

A PHENOMENOLOGY OF PROGRAM
MAINTENANCE AND EVOLUTION

[C.K.S.]^{by} CHONG HOK YUEN

IMPERIAL COLLEGE OF SCIENCE AND TECHNOLOGY
(UNIVERSITY OF LONDON)

DEPARTMENT OF COMPUTING

A PHENOMENOLOGY OF PROGRAM
MAINTENANCE AND EVOLUTION

by

C.K.S CHONG HOK YUEN

A Thesis submitted for the Degree of Doctor of Philosophy
November, 1980.

ABSTRACT

This work is inspired by Belady and Lehman's Program Evolution Dynamics concepts and attempts to expand them. After these concepts have been reviewed and updated, two dissimilar software systems, in terms of age, size, structure, etc., were analysed by means of a method which attempted to blend the hypothetico-deductive with the empirical-inductive, the synthetic with the analytic and the qualitative with the quantitative.

By partitioning each system in various ways and by studying it at several of its levels, diverse results were obtained; each component type was distinguishable by some of its properties. None of these evolves in a very predictable manner, the evolution of each being not too similar to that of the other and being somewhat dependent upon the observed component and versions of the system. Adding these complex patterns with the equally complicated influxes of new and outfluxes of some old components led to a global behaviour devoid of distinctiveness.

The study of some activities representing the system's interactions with its environment revealed intricate causal patterns and inter-relationships whereby an action somewhere may induce a prompt or delayed compensatory or sympathetic reaction elsewhere. Although the activities from a unit of the environment or upon a component of the system could be sufficiently explored for meaningful models, yet, when, for each activity, all the units' contributions are added together, depending upon whether they are synchronised or not, one observes either global dynamics or invariance.

The results led to an abstract model which also attempts to capture the existing Evolution Dynamics concepts as well as several software engineering and systems ideas in terms of the system, its environment and their interactions.

ACKNOWLEDGEMENTS

It is with great pleasure that we acknowledge the assistance of the following people.

First and foremost, my supervisor, Professor Meir M. Lehman, has consistently shown acute interest and offered encouragement, advice and constructive guidance. My special thanks go to him and to Dr. G. Benyon-Tinker, Dr. P.G. Harrison and Mr. J. Newland for their helpful comments on parts of this thesis.

I would also like to thank both Centrefile Limited and International Computers Limited for making their data accessible to me. The enthusiastic co-operation and assistance given by Mr. D. Benest and his colleagues at Centrefile and Messrs. A.F. Hutchings, G.D. Pratten, R.A. Snowdon and their colleagues at ICL are gratefully acknowledged.

I am particularly indebted to the management of Centrefile for their generous financial support during my first year of research and to that of ICL for theirs during these last two years.

And I owe most to Andrea Chong for many patient hours of typing and careful proof-reading and whose sacrifices, encouragement and devotion were invaluable. And, last but not least, to Nicholas Chong, born at about the same time as this project started and whose needs were in constant conflict with those of this research and in spite of/because of whom I simply plodded on. This work is dedicated to them, my wife and child, who both prevented and inspired this work and without whom it might have been completed much sooner or never at all.

CONTENTS

CONTENTS

PREFACE	(iv)
CHAPTER 1: COMPUTER SCIENCE, SOFTWARE ENGINEERING, LARGE SOFTWARE STUDIES AND EVOLUTION DYNAMICS	1
1.1 Introduction: the problems posed by large software systems and their possible solutions through quantitative methods	1
1.2 Review of Lehman's (and others') ED work	3
1.3 Retracing Lehman's footsteps	5
1.3.1 The plots	5
1.3.2 Method of analysis	6
1.3.3 The analysis proper	6
1.3.3.1 Cumulative modules handled	6
1.3.3.2 Handle rate	8
1.3.3.3 Fraction of modules handled	9
1.3.3.4 Size	10
1.3.3.5 Net growth	10
1.3.3.6 Release interval	12
1.3.3.7 Size against RSN	12
1.4 Summary of our results	12
CHAPTER 2: THE GENERAL APPROACH TO THE INVESTIGATION	14
2.1 Introduction	14
2.2 Objectives	14
2.3 The four aspects of the approach	14
2.4 Blending the empirical-inductive with the hypothetico-deductive approach	16
2.5 The need to emphasise the empirical method	16
2.6 The empirical method	17

CONTENTS

2.7	Deciding which variables to measure	17
2.8	The extent to which variables can be measured	19
2.9	Statistics gathering	20
2.10	The historical data base	21
2.11	The types of data available and the analyses envisaged	23
CHAPTER 3: ANALYSIS OF THE BD SYSTEM		25
3.1	Introduction to BD	25
3.1.1	The relative simplicity of the system	25
3.1.2	BD is a collection	25
3.1.3	Classifying the components of the system	26
3.1.4	BD is non-release based	26
3.1.5	The available data	27
3.2	Analysing the changes data	27
3.2.1	Looking for patterns in the series	28
3.2.2	Fitting the best polynomial to each series	28
3.2.3	Time series analysis	29
3.2.3.1	Analysis in the time domain	29
3.2.3.2	Analysis in the frequency domain	32
3.2.4	Fitting the normal distribution to each variable	33
3.2.5	Looking at the CCSS data	34
3.2.6	Summary of the results	35
3.3	Analysis of the size measures	36
3.4	Micro analysis: distribution analysis	38
3.4.1	The various sets' attribute distributions at the same point in time	39

CONTENTS

3.4.1.1 Comparing the properties of the (100) MFCM and of the (331) UM with those of the population, P	39
3.4.1.2 Comparing the properties of different subsets after partitioning	41
3.4.2 Distribution evolution	43
3.4.2.1 Some complexity measures, suggested and used in previous work	44
3.4.2.2 Using a new method to test for increasing complexity	49
3.4.3 Correlation and multiple regression	50
3.4.3.1 Correlating the attributes	50
3.4.3.2 Linear regression models	51
3.4.3.3 Non-linear multiple regression models	52
3.5 Module analysis	53
3.5.1 Analysing the attributes distributions at the same point in time	53
3.5.2 Analysing the modifications data with respect to the modules	53
3.5.2.1 Analysis of all the changed modules and the old modules	54
3.5.2.1.1 Analysis within each time period	55
3.5.2.1.2 Evolution of mf/f , f/m and Max_f over time	56
3.5.2.2 Analysis of the new modules	57
3.5.2.2.1 Repeat of the analysis of the evolution of the global change parameters over time	57
3.5.2.2.2 Exploring the best fits to the global parameters	58
3.5.2.2.3 Deriving the exponential fit from purely theoretical considerations	59
3.5.2.2.4 Convenience and uses of the exponential fit	59

CONTENTS

3.5.2.3 Repeating the analysis for the typical new module	61
3.5.2.3.1 Deriving the data for the typical new module	61
3.5.2.3.2 Results and remarks from Figure 34	61
3.5.2.3.3 Using the typical module's results to predict other modules' behaviour	62
3.5.2.4 Repeating the analysis for the typical new micro	62
3.5.2.5 Comparing the 3 methods	63
3.5.2.6 The work of Shooman	63
3.5.2.7 Exploring the usefulness and adaptability of the model	64
3.5.2.7.1 A more general model: providing for the introduction of new errors	64
3.5.2.7.2 Using the model to explain the old modules' behaviour	66
3.5.2.7.3 Proposing a quality measure	67
3.5.2.7.4 Refinements - when the quality of the module is a function of time	68
3.6 Summary of Chapter 3	69
CHAPTER 4: ANALYSIS OF THE B SYSTEM	72
4.1 Introducing the system, its history, environment and sources of data	72
4.1.1 Introduction to B	72
4.1.2 The structure of B	72
4.1.3 The project log and the SDL descriptions as the sources of structure data	74
4.1.4 Holon versions and phases	74
4.1.5 Other sources of data	74

CONTENTS

4.2 Analysis of the holon attributes	75
4.2.1 The releases, subsystems, holons and data we worked on	75
4.2.2 Analysing the frequency distributions from one release to the next	77
4.2.3 Analysing the holons classified by the types of changes sustained	78
4.2.3.1 The possible evolutionary patterns for the changing holons	80
4.2.3.1.1 A general and uniform shift of the frequencies from left to right	80
4.2.3.1.2 A general but uneven shift of the frequencies from left to right	81
4.2.4 The global behaviour as explained by the different types of holons classified by changes category	82
4.2.4.1 Interpreting the distributions' complexities in terms of the holons' complexities	83
4.2.4.2 Relating the complexities of the different types of holons	84
4.2.4.3 The ageing and rejuvenating processes	84
4.2.4.4 Applying the ideas of ageing and rejuvenation to software	84
4.2.5 Correlation and multiple regression analysis	87
4.2.5.1 Correlation analysis	87
4.2.5.2 Regression models for cumulative PATCHES and DAF2 of K32	88
4.2.5.2.1 The linear models	88
4.2.5.2.2 The non-linear models	88
4.2.5.2.3 The incremental models	90
4.2.5.2.4 Applying a theory of stability or instability to B	91
4.3 Analysis of daf2's	93

CONTENTS

4.3.1 Randomness tests and polynomial fits	94
4.3.2 Time series analysis	95
4.3.3 Investigation of the daf2's distribution	95
4.3.4 The typical holon's profile	96
4.4 Analysing the software notices	97
4.4.1 Purpose and contents of software notices	97
4.4.2 Time series analysis of the software notices	97
4.4.3 Analysis of each release's software notices	98
4.4.4 Models of release and system behaviour in terms of number of software notices	99
4.4.5 The trends in the global time series and modelling them	100
4.5 Bug report analysis	101
4.5.1 Introduction	101
4.5.2 Analysis of the distributions of 4 chosen months' data	102
4.5.2.1 The different behaviours which the variables exhibit over time	102
4.5.2.2 Further analysis with the response time variable	102
4.5.2.2.1 Comparing different periods' response time distributions of each priority type	103
4.5.2.2.2 Comparing different priorities' response time distributions over the same period	103
4.5.3 Correlating one variable with another	104
4.5.4 Analysis of the bug reports and the bug responses time series and the outstanding queue	105
4.5.4.1 Analysing each series on its own	105
4.5.4.1.1 The time domain results	105

CONTENTS

4.5.4.1.2 The analysis of the outstanding bug reports	106
4.5.4.1.3 The typical week's REP and RES	107
4.5.4.2 Crosscorrelating the REP and RES series	108
4.5.4.3 Analysing response time	109
4.6 Analysis of the daf6's data	110
4.6.1 Introduction	111
4.6.2 The analyses we performed	112
4.6.3 Choosing the groups, variables and time period to use	113
4.6.4 Clustering the groups	114
4.6.5 The global daf6 distribution over time	114
4.6.6 Analysis in the time domain	115
4.6.6.1 Analysis of group 5	115
4.6.6.1.1 Choosing a Δt	115
4.6.6.1.2 Choosing the variables to explore	115
4.6.6.1.3 Analysing the trend in each category of daf6's	115
4.6.6.1.4 Analysing the evolution of the daf6 distribution and of its diversity for each variable	116
4.6.6.1.5 Analysis of the degree of association between pairs of the variables	118
4.6.6.2 The in-depth analyses of G3 and G6	118
4.6.6.3 Comparing the results for the 3 groups	119
4.6.7 Analysis in the release domain	120
4.6.7.1 Analysis of the variables forsaken in the time domain analysis, with the emphasis on ATTEMPTS	120
4.6.7.2 Analysis of the MTHS, UTIL, CAT and MODE variables in the release domain	120

CONTENTS

4.6.7.2.1 Analysing the diversity of each variable	121
4.6.7.2.2 Testing for pairwise correlation among the diversities of the variables	122
4.6.7.2.3 For each group, the evolution of the distribution by each variable from one release to another	122
4.6.7.2.4 For each release, the group to group comparisons of the distributions by each variable	122
4.6.8 Generalising the main results of the daf6 analysis	123
4.6.8.1 The properties	123
4.6.8.2 The parts	124
4.6.8.3 The releases	124
4.7 Summary of Chapter 4	125
CHAPTER 5: POOLING THE ANALYSES OF THE TWO SYSTEMS STUDIED	132
5.1 Introduction	132
5.2 The structural characteristics	132
5.2.1 The 20-80 law	132
5.2.2 The properties are not necessarily concordant	133
5.2.3 The properties of the partitions	133
5.3 The evolution of the structure characteristics over time	135
5.4 The evolution of the activity characteristics	136
5.4.1 Each activity on its own	137
5.4.2 The interrelationships among the activities	142
5.5 The evolution of the output characteristics	142
5.5.1 Net growth characteristics	143

CONTENTS

5.5.2 The characteristics of modifications to the BD code	144
5.5.3 Explaining why complexity for the BD system has not increased	147
5.5.4 Characteristics of modifications to the B code	148
5.5.5 Complexity investigation for B	148
5.6 The characteristics of the system-environment interactions	149
5.6.1 Relating the outputs with the inputs	149
5.6.2 Investigating how the different characteristics of a variable change from part to part and from release to release	151
5.7 Concluding remarks	153
CHAPTER 6: THE FLAWS IN THE DATA, TOOLS AND RESULTS	155
6.1 Introduction	155
6.1.1 The need to devise a methodology for analysing the data	155
6.1.2 Working with distributions and the associated advantages and problems	156
6.2 Static structure analysis	158
6.3 Evolution analysis	161
6.3.1 Structure evolution analysis	161
6.3.2 Activity dynamic analysis	161
6.3.2.1 Tests which are used on each individual series	162
6.3.2.2 Testing for inter-relationships in the series	164
6.3.3 Output dynamic analysis	164
6.4 The data used to represent system and environment interactions	167

CONTENTS

6.5 Concluding remarks	167
6.5.1 Some points for further exploration	167
6.5.2 Some results which may be useful	168
CHAPTER 7: THE WAY AHEAD	169
7.1 Introduction	169
7.2 Improvements concerning the existing data	169
7.3 Improvements concerning the existing analyses	170
7.4 Enlarging the analysis	171
7.5 Automate data collection and analysis	173
7.6 Where do we go from here?	174
CHAPTER 8: A THEORY OF PROGRAMMING SYSTEMS' STATIC AND DYNAMIC BEHAVIOURS	176
8.1 Building the theory from empirical results	176
8.2 A very general model	179
8.3 The state coordinates	181
8.3.1 The state coordinates refer to measurable properties rather than traits of the system	181
8.3.2 How the state coordinates contribute to the system's static complexity	182
8.4 The environment	184
8.4.1 The reasons for including the environment in the study	184
8.4.2 The system-environment boundary	185
8.4.3 Some interesting environmental properties	186
8.5 Interfaces with the environment	187
8.5.1 Not all environmental factors are inputs	188

CONTENTS

8.5.2 The nature of the interfaces	189
8.6 The dynamics of the exogenous factors	189
8.6.1 The need to model the environment	190
8.6.1.1 The multi-layers of the environment	190
8.6.1.2 Recursiveness is also applicable to the system's multi levels	191
8.6.2 The effects of the environment's dynamics upon the system's complexity	191
8.7 The dynamics of the system's structure	192
8.7.1 Interdependencies among the structural properties at any given time	192
8.7.2 Feedback effects	193
8.7.3 The smoothing of the individual effects at the global level	193
8.7.4 Different models may be required for different parts of the system	194
8.8 The effects of time	195
8.9 Other aspects of the model	196
8.9.1 The long term effect of the initial system state	197
8.9.2 Steady state	197
8.9.3 The effect of the system upon its environment	197
8.10 Conclusion	197
TABLES	200
FIGURES	228
REFERENCES-	289

PREFACE

Under normal circumstances, one would not have required a preface but we felt that there was a need to clarify at least 3 points before the reader can be allowed to peruse the text. First of all, we need to explain why this thesis is so bulky. Then, the justification for structuring it the way it has been done needs be given, too. And, finally, it seems that we had to give some idea of the types of contributions we have made. As this could not be fitted within the main text, it is perhaps appropriate to include it here as well.

This research was not intended to last more than 2 years nor the text of the thesis to exceed more than 80 pages. In fact, we feared that it might not even reach that target... There was a distinct possibility of not exceeding it had we taken a purely hypothetico-deductive approach but how meaningful the effort would have been is a matter of pure conjecture. Instead, rightly or wrongly, we chose the empirical approach. But then, we found that the deeper we dug, the more we discovered and had to report, the more questions were prompted by our findings and the less we found that we knew so that we had to dig deeper once more until, Parkinson's law having been broken again and again, we had to eventually call a halt.

During the writing-up, while initially drafting this thesis from the hundreds of pages of results, [36-50] among others, we found ourselves torn between two polarized methods in which to present the account of our work. Like most statistical and most of the software engineering publications with a statistical flavour, our thesis could have put the emphasis on the results while very conservatively evaluating and interpreting them. The other approach is to do away completely with the results, thus saving us a lot of work and reducing this thesis to perhaps half of its actual size, in order to concentrate solely upon their interpretations and deductions. After much thought, we decided that neither was appropriate. The first method would have been too sterile and not of sufficient relevance for a Computing Science thesis while the second would much weaken our arguments since there would have been no evidence to back them up. We have, therefore, attempted to tread the middle ground although, by doing so, we may have run foul of those who prefer the first as well as those who prefer the second method. On the one hand, we could be accused of not presenting enough factual evidence and of going too far in our interpretations while, on the other, we could be accused of providing too many results and of not being adventurous enough in our deductions. Although it is not easy to please everyone, we have tried our best and, in consequence, the thesis has become bulkier than it would have been had either the first or the second method been selected.

The second point concerns the structuring of the thesis and, as we shall see, the particular structure we have selected, has probably lengthened the thesis, too. Roughly, we wanted the topics we treated to narrow down from the general to the particular and to expand from the particular back to the general. Chapters 1 and 2 were initially intended to be a single chapter of a few pages where we would have briefly situated our work within the narrowing confines of Computing Science, Software Engineering, the study of large software and Evolution Dynamics, ED, and also give some ideas of the philosophical considerations behind our approach. However, while attempting to summarise the existing ED ideas and findings, we found that the last attempt to pool them together was made more than 3 years ago. As much has happened since then, we had to repeat that work and discovered that, in doing so, we have already been able to contribute directly, in some small amount, towards the advancement of ED. We also took the opportunity to attempt to strengthen the already existing methodology and results by introducing new tools for the analysis. Thus, our introductory chapter, swollen beyond expectations by these additions, had to be split into Chapter 1, dealing with what we have described above, and Chapter 2.

Chapter 2 attempts to justify the approach we took, why we allied the hypothetical with the empirical, the analytic with the synthetic, the quantitative with the qualitative and so on. We also needed to discuss why we started from the available data rather than to attempt to gather our own, what measures were extracted from the data, how good they could be and what sort of reliability and importance one could place upon them. In Chapter 2, we may not have contributed any original idea but we, at least, attempted to clearly and coherently describe those intuitive principles which may have guided most researchers.

Coming to our own work, we again found that there were two distinct ways of presenting it. The first method would be to list the work done and the results first. After that, we could pass on to the interpretations and then to the generalisations and theory building. Presumably, the first stage would produce tedious reading, specially for those not so interested in the actual results and methods of arriving at them, while, for the remainder, we would need frequent cross-references, again adding tedium to the reading. The alternative is to present and discuss the results, one at a time, each with its own interpretations and generalisations. This could have been more interesting for the reader except that, when we attempted it on the first draft, we found that we had to repeat the same arguments all over the place. The result was that the thesis grew more and more repetitious.

Although repetitions could be minimised by reverting back to

the first method, we preferred to compromise once more. A three-tiered approach was finally decided upon whereby we steadily moved away from the particular to the general. Chapters 3 and 4 constitute the first tier, Chapter 5 the second and Chapter 8 the final one.

In Chapter 3, we concentrated upon the analysis and results from the data we had for the first of the 2 systems we had studied, although, at the same time, we tried to provide adequate interpretations for all the results. However, whenever interpretations were similar or could be exploited more fully and more appropriately in Chapter 5, we did not attempt to develop them more than strictly necessary. In return, results which led to interpretations not corroborated anywhere else or which could not be exploited in Chapter 5 were given the full treatment. Thus, the reader should not be troubled more than necessary by the unevenness with which the results are exploited in this chapter. Similar remarks apply to Chapter 4 which is for the second system we have studied.

The second tier of the approach allowed us to, at last, remove ourselves from the actual data and results. By pooling and integrating all the findings and conclusions in a single place, their interpretations are all the more strengthened since not only do they support each other but they can be viewed in much better perspective. To relate these with the actual results in Chapters 3 and 4, liberal use of cross-references had to be made.

The third and final tier of the approach was to remove ourselves as far away from the data as possible and, starting from the findings summarised in Chapter 5 as well as existing results, ideas and concepts and some thoughts we have been entertaining, to attempt to generalise them such that what we have learnt can be converted into a theory of large systems behaviour. This was what we attempted in Chapter 8.

However, theorising should be our last activity and Chapter 8 has more appropriately been situated as the very last chapter of this thesis. Before generalisation can be appreciated, one must situate its context and know what are the limitations of its foundations. Anyway, whatever we have been able to conclude is conditioned by the approach, the data, the tools and the results. Chapter 6 discusses whatever flaws we could perceive in these while, in Chapter 7, we proceeded with the exploration of possible ways of refining and extending our work.

As to the contributions, the initial one is in Chapter 1. We compiled an up-to-date account of all ED findings, tightened them up in certain cases and introduced some more tools for analysing the data. Section 1.4 gives a brief summary of these.

The second concerns our methodology. In Chapters 3 and 4, we attempted to show how, following the philosophical considerations of Chapter 2, one may systematically attempt to exploit all facets of the data that are available. Our methods, which are really of general applicability, could eventually lead to a truly useful methodology which may become as important as the results of this particular study. Section 2.11, among others, contains a short description of the 3 aspects of a system's study that we have concentrated upon. And, in Section 6.1.1, we briefly pointed out a few strong points which our methodology seems to possess.

There are probably very few works that resemble either Chapter 5 or Chapter 8, quite a lot of whose contents could be novel but, then, these are based upon the results of Chapters 3 and 4. As these are both summarised and extended in Chapters 5 and 8, we shall not proceed with the details although a few points should be highlighted:-

- 1) Section 3.3's results which seem to have strengthened previous ED size evolution concepts.
- 2) Section 3.4.2 which has improved the use of modifications data for measuring increasing complexity, specially by the introduction of Jonckheere's test.
- 3) Our exponentially decreasing modifications rate model, Section 3.5.2.2., which can be useful in more ways than one.
- 4) How the introduction of the ageing and the rejuvenation concepts applied to the whole system can clarify how its complexity is evolving, Section 4.2.4.
- 5) The introduction of Sutherland's ideas, Section 4.2.5.2.4.
- 6) Our discovery that most measures at the global level tend to behave as random variables, Sections 3.2, 4.3.1, 4.4.2 and 4.6.5, although they tend to be more deterministic at the sub-levels, Sections 3.5.2.2, 4.3.4, 4.4.3 and 4.6.6.1.4, and the use of the latter to explain the former, Section 5.4.1.
- 7) The relevance of sub-level analyses which not only help to explain the behaviour at a more global level but also seem to establish that the analysis of a system at any level is worthy of study on its own, Sections 3.4.1, 3.5.2.1.1, 4.2.4, 4.3.4 and 4.4.5.
- 8) The "explanation" of increasing dynamics and invariance of increasingly large systems in terms of the increasing number of contributing sources, Sections 4.5.4.1.2 and 5.4.1.
- 9) That both time and release-based concepts of evolution are conducive to equally meaningful results, Section 4.6.7.

CHAPTER 1

COMPUTER SCIENCE, SOFTWARE ENGINEERING,
LARGE SOFTWARE STUDIES AND EVOLUTION DYNAMICS1.1 INTRODUCTION: THE PROBLEMS POSED BY LARGE SOFTWARE SYSTEMS
AND THEIR POSSIBLE SOLUTIONS THROUGH QUANTITATIVE METHODS

The computer age has been with us for about 30 years only yet one cannot but marvel at the astonishing technical accomplishments that have been achieved by hardware technology. Sophistication of usage has grown with a pace to match the machines' increasing capabilities. Unfortunately, the technology of building large software systems to handle the growing capabilities and needs has not progressed as rapidly.

The US military were the first to develop very large software systems but, since these were shrouded with secrecy, it was not until the advent, in the late sixties, of large scale manufacturer-supplied operating systems such as OS/360 and of specialised real-time systems such as airline reservation systems that the problems, among others, of cost overruns, schedule slippages and over-expectations became widely known.

Because more and more large systems were being developed, mostly on an ad hoc basis and because most ran into several, if not all, of the problems associated with large software, understanding the phenomenology of software development became a high priority, so much so that the Study Group on Computer Science, set up in 1967 by the NATO Science Committee, decided to provide a badly needed opportunity to discuss these problems. This materialised as the Garmisch 1968 Software Engineering Conference [132] which has since been viewed as a turning point, an event which has had a significant and lasting effect on the subsequent work in the software field. It was there that a "software crisis" was fully acknowledged and that a radical new term, "Software Engineering", which emphasises that software can and must be engineered, was coined. The band wagon had been started and has been rolling ever since [149].

Following that conference, there was a flurry of activity in Software Engineering [34] but little work was done concerning its quantitative aspects, although it was realised that already existing techniques similar to those of the craft of Computer Performance Evaluation, which was developed to measure hardware utilisation, as exemplified in Freiburger's book [63], could be applied to measure the system behaviour itself.

It was not until 1973 that Boehm [26] gave fresh impetus to quantitative methods when he reported that the status of software error analysis was virtually non-existent. Because of the increasing economic impact of software errors, such a state

of affairs could not continue to be tolerated for very long and his urging generated a proliferation of studies, including such topics as error analysis [1,7,22,27,59,64,70,71,114,115,122,126,130,152,161,171,172,173], costing [184], programmer productivity [169,179], program quality [28,153] and software complexity [68,120,125].

Endres [59], for instance, analysed error data in terms of error occurrence rate and frequency of software modification for several module distributions. Shooman and Bolsky [161] classified their data by frequency of error occurrence rate for each type of error and so on. Two important subdisciplines that have emerged from these activities are software reliability [31,57,78,116,119,121,122,126,129,155,156,160,171,173,175] and the study of large software systems [6,8-10,13-20,93-110,134-135,145-148] although there is much overlap between the two [121,122,126,160,175].

Large software systems have been studied by many authors from various viewpoints [5,30,33,55,74,91,123,151,180] but the (different) quantitative approaches of Putnam, Basili and Lehman seem more promising in solving the problems outlined in the opening paragraphs. These problems are still very real: large software is still being built largely as before, the same problems of building, using and maintaining these are being encountered [176] and we still do not perform enough quantitative research and know too little about large program behaviour.

It, thus, seems that, despite the Garmisch conference, the shock waves which Boehm sent through the software community and the research that has been going on, the lessons of these last twelve years have not yet been learnt. Realising that we need to understand how a large program behaves before we can attempt to devise better ways of constructing, modifying and managing it, Lehman, in particular, has concentrated his studies upon the evolutionary behaviour of large programs during maintenance. In recent years, his ideas have coalesced into a theory, called "Evolution Dynamics", ED, which appears to have great potential. It is in the context of attempting to enlarge its basis that we have undertaken this research project.

The initial part of our work consisted of assimilating the existing ED theory. To describe what we have learnt, we can either attempt to summarise the contents of all the relevant ED publications one at a time or, more appropriately, since a lot of repetition will be avoided, by providing:-

- 1) A historical survey of Lehman's (and others') ED work, as in Section 1.2.
- 2) A pooling and an up-to-date account of the various findings, as in Section 1.3. In addition, we verified the

results, where applicable, performed further tests on the data and made a few comments, too.

1.2 REVIEW OF LEHMAN'S (AND OTHERS') ED WORK

To avoid repetitions, in making our brief historical survey of all ED publications, we have kept the descriptions of each paper's contents to a minimum. Instead, we have concentrated upon the significant additions of each paper and upon the development of the research, without giving too many details about each feature. In consequence, the review may appear mechanical but this was a necessary evil. Anyway, the essence of the theory will be extracted and examined in the next section.

The origins of ED theory can be traced back to 1969 when Lehman carried out a study of the programming process within a particular environment (IBM's) and published the analysis of some data on OS/360 [93]. This led to a (static) macro-model, the Yorktown model, of the growth of large programming system projects. The model was reported in 1971 in [13] which, in addition, mentioned a micro-model and introduced the phrase "Programming Systems Growth Dynamics" (later to be changed to "Evolution Dynamics"). More extensive and some of the definitive OS/360 data and plots, upon which the principles and laws of ED came to be based, were published in early 1972 [14]. Observations such as the smoothness and linearity of some plots (e.g., of size against release sequence number, RSN), the exponential growth of complexity and the ripples indicative of a self stabilising feedback process, all to be later integrated within the concepts of ED, were already made.

In [15], we have the first hint of the forthcoming laws although none was, as yet, formulated and the modified ideas of Baumol concerning unbalanced productivity growth [11], to be more fully developed later, are briefly introduced. [94] collates all previous work and ideas and sets the pattern for the subsequent papers: there is a noticeable shift from the purely empirical investigation towards attempts to interpret and understand the observations and to knit them into a coherent theory. One novelty is the presentation of a dynamic macro-model (the Berkeley model), built around the progressive and anti-regressive activity ideas as adapted from Baumol. [16] is a revision of [94], mainly through the discard of some of the early ideas and plots retained in [94].

[16] is itself overlapped by [95] which, besides, introduces the new term "Evolution Dynamics" and enunciates, for the first time, the first 3 of the 5 laws, Figure 1. Another contribution of this paper is the description of yet another model, the London model, but none of these first four models, byproducts of the various phases in the development of the

theory, are to be mentioned again. It was only at the start of 1975 that a second system, DOS, was studied from an ED point of view [113]. However, the analysis was restricted to some plots and some of their associated regression and/or exponential fits. Discussions and interpretations were minimal.

In late 1975, a third system, Executive, was studied [73]. Although it was not formally compared with OS/360 (or DOS), it was obvious that, although the same terms (e.g., size, change) did not measure exactly the same things for the two systems, yet quite a few, though by no means all, of the plots and observations were very similar.

[17] presents a fifth model, a fault-penetration model. However, its more significant contributions to ED are 1) the regression fits to net growth and fraction of modules handled in terms of RSN and to handle rate and size in terms of days, 2) the introduction of new ideas such as the cyclicity of the growth process and the observations on the associated critical growth mass and invariance and 3) the inclusion of 3 Executive plots to confirm the OS/360 results.

Based on the idea of progressive and anti-regressive activities and that of increasing complexity, an early and perhaps premature non-linear differential equation model of ED was developed [150]. [18], which is really a reworking of [17], is probably intended for a wider distribution. It is only in [108] and [109] that we see rather extensive comparisons, of some systems' behaviours: OS/360's, Executive's and Omega's. Most of the themes exposed previously, but treated more thoroughly in [18], are recapitulated with the main aim of demonstrating the generality of the observations although the authors noted several unexplained anomalies. These two papers cover much the same ground, although [108] attempts to be more interpretative.

From then on, some momentum was gathered and 3 more systems have been studied (each on its own), using similar techniques and plots as used previously. ED has been looked at in [36], CCSS in [110,72] and B in [51]. Further data for Executive have been analysed in [36] and still newer data are available in [23]. However, no study analogous to those in [108,109] has been attempted, despite the elapse of 3 years and the availability of more data on both further and already studied systems. It is probably opportune to attempt to fill this gap specially as previous results (and, hence, the theory itself) may be considerably strengthened if the number of systems that can be shown to behave similarly and as predicted by the theory is doubled. The work in Section 1.3 is an effort in that direction.

[98] is partly concerned with the introduction of the 4th and 5th laws, the former being the generalisation of invariance and

the latter being that of critical growth mass. CCSS's net growth per release plot is used together with those of the 3 previous systems to illustrate the 4th law, but like [97-107,19,20], [98] concentrates upon building up from the basic results, concerning itself with explanations, discussions and interpretations of cyclicity, invariance and long range trends and of the laws and what the consequence of violating the latter would be. From then on, increasingly new ideas and views are exposed, e.g., attempts at defining programming methodology and software engineering and both static and dynamic views of how an engineering discipline develops and evolves are in [99] while in [107] radical new views of program types are presented. Although interesting and probably revolutionary, too, these views are not directly related to the basic ED theory. Consequently, we shall not dwell upon them. [186] is a refinement of Riordon's model.

1.3 RETRACING LEHMAN'S FOOTSTEPS

The work in this section has, in fact, 2 purposes:-

- 1) To extend the work done in [108,109] to the most up-to-date data for the systems already examined and for the unexamined systems.
- 2) To attempt to strengthen the results by applying further tests to the data.

1.3.1 The plots

Most of the observations and interpretations which led to the development of the ED theory were based on 7 types of plots. These and the latest versions for the 6 systems (OS/360, Omega, Executive, B, CCSS and DOS) we shall consider are as follows:-

- 1) Cumulative modules handled, CMH, against age, A, Figure 2.
- 2) Modules handled per day, or handle rate, HR, against RSN, Figure 3.
- 3) Fraction of modules handled or simply fraction handled, FH, against RSN, Figure 4.
- 4) Size, S, in modules against A, Figure 5.
- 5) Net growth per release, NG, in modules against RSN, Figure 6.
- 6) Release interval, RI, against RSN, Figure 7.
- 7) S against RSN, Figure 8.

Notes

- 1) For Executive, instead of modules handled, we had changes and size was in lines of code.
- 2) The first four families are those examined at length in [108,109].

1.3.2 Method of analysis

It is probably a good thing to present the method we adopted. Taking each family in turn, in the same order as in [109], we shall:-

- 1) Summarise the previous results and observations made. This is why, to avoid repetitions, we have minimised in Section 1.2 the discussion of the results themselves.
- 2) Analyse the data (for all 6 systems, wherever possible) according to the best tools in our arsenal.
- 3) Reach our own conclusions.
- 4) Compare our conclusions with the existing ones.
- 5) Make such comments and/or proposals as appropriate.

Note As B has at most 6 points, it shall be omitted from any analysis.

1.3.3 The analysis proper

1.3.3.1 Cumulative modules handled

At one point or another, the OS/360 [97,98,108,109], Omega [97,98,108,109], Executive [97,98,109], DOS [97] and CCSS [98] plots had been described as straight lines which were interpreted as indicating "the invariance of global work rate" [98] or taken as examples of "constant work rate" [102,108,109] and led to the enunciation of the 4th law [98].

The tool we applied is that of successive polynomials. The analysis of variance, F-test, is used to determine whether a higher degree polynomial is a significantly better fit than the next lower one, proceeding in a step-wise fashion. We retain a polynomial only if its estimated regression coefficients are significant by the t test.

The detailed results are in Table 1. The test was not performed for Omega (no data). The fits, up to the cubic, for the other 4 show that:-

- 1) In all cases, except Executive, the cubic is a better fit than the quadratic which is better than the straight line (for Executive, the cubic is better than the straight line). Although the linear fit has an R^2 upwards of 91%, the fact remains that, statistically, either or both the cubic and the quadratic fits are better.
- 2) On the assumption that the quadratic is better, then for OS/360, CCSS and DOS, there is a maximum point somewhere in the plot as the fit has the form

$$\alpha + \beta_1 A - \beta_2 A^2.$$

Thus the original observation on OS/360 [18] which was subsequently dropped in favour of linearity may be more adequate: "The data at the extremes suggest that in the early life of the system and in the most recent 2 releases, the handle rate may have been a little lower... It appears that even though the straight line fit is adopted as an initial model, an S-curve provides a more faithful representation over the life to date of the operating system".

But it seems as equally important to note that the technique of cumulating the data in the sequence that they occurred is not conducive to very meaningful results despite the very strong reason (that of dealing with the release overlap problem) which caused Lehman to resort to its use. This is because, for any random variable, the cumulative plot will yield a straight line and we believe that MH is indeed a random variable except at the early and late stages of a system's life. Unfortunately, there are too few data points for any protracted analysis here. However, subsequent work on the two systems we shall describe in Chapters 3 and 4 gave ample support for this hypothesis.

Consider the random variable, y , which is normally distributed, $N(\mu, \sigma^2)$,

then y_t , i.e. the t -th value of y is given by

$$y_t = \mu + \varepsilon_t \quad \text{where } \varepsilon_t \sim N(0, \sigma^2) \quad \text{and} \\ \text{is usually known as white noise.}$$

$$\sum_{i=1}^t y_i = \sum_{i=1}^t (\mu + \varepsilon_i)$$

$$\text{i.e. cum } y = \mu t + k \quad \text{where } k=0 \quad \text{since } E(\varepsilon_i)=0 \text{ and } E(\sum \varepsilon_i)=0.$$

Indeed, both the "statistically invariant" phrase used in the enunciation of the fourth law and the "constant work rate" which Lehman has found appear to imply that MH is a random variable although this has not been stated explicitly. It then seems that the 4th law could be more accurately recast, perhaps as:-

The global activity rate over the major portion of the maintenance phase of the life cycle of a large programming project, by and large, shows no time dependent behaviour.

The "by and large" qualification is necessary to cover exceptional circumstances such as the organisation's closedown due to strikes, Christmas vacations and so on.

That work rate may behave like a random variable is probably

contrary to most managements' expectations. This may not only come as a surprise but as a big shock to them. Although, for maximum impact, it seems appropriate to discuss at once the cause of this behaviour yet this is premature. This is a result which will appear again and again so that we prefer to accumulate all our evidence and, as stated in the Preface, we shall reserve the interpretations of the common results in Section 1.3 and Chapters 3 and 4 to Chapters 5 and 8, except when only locally relevant. Besides, the purposes of this section were clearly stated in its opening paragraph. The impatient reader may, of course, jump immediately to Chapters 5 and 8 but we believe that nurturing him gently through all the work we have done allows the results to be appreciated more fully and the fruits which will be ultimately gathered to be more rewarding.

1.3.3.2 Handle rate

Lehman's observations on OS/360 and Omega were about a "well defined average" handle rate and a "cyclic variability" whose descriptions range from a rather precise "zig zag shape with peaks being shortly followed by troughs" [108,109] to a vaguer "cyclic effects, not necessarily with a pure period" [104].

Rather than talk about well defined averages, it seems more convenient to talk about stationarity [29] in a random variable. About the second observation, one may attempt serial correlation as well as deviations from randomness significance tests as different methods to directly or indirectly verify or support it. Unfortunately, there are too few data points to perform autocorrelation [29] or spectrum analysis [79] to identify serial dependencies. It is therefore not possible to directly verify for the presence or absence of serial correlations (This, as will be seen later, could, however, be attempted on the longer series we had for the 2 systems that we subsequently studied.): one can only test the null hypothesis that the sequence in which the measures are taken is random.

No satisfactory significance test which can support the cyclicity hypothesis seems to exist, specially as cyclicity does not appear to have been too rigorously defined. At best, what can be achieved, unless one attempts to define the term and to devise a test for it, is to interpret it literally (the zig-zag effect). Then, some existing randomness tests whose alternative hypotheses can be arranged to favour the cyclicity idea can be employed. Even then, significance tests can never establish the truth or otherwise of any hypothesis. It was in such a context that we had to proceed.

The Runs test [163] was a previously used randomness test which was said to give significant results [108] but we found that it was wrongly applied in [108,73]. A hopefully correct use is now

made of the test which we supported with two further tests, the Turning Points and the Phase Length tests [88]. All three are based on the order or sequence in which the observations are obtained but each examines a certain property such as the number of runs (or of turning points or of phases of different lengths) in the data. That property would then indicate whether the data is similar to that obtained from a random process or otherwise. For example, too few runs would suggest either a trend or some bunching due to lack of independence or both while too many runs would indicate systematic short-period cyclical fluctuations [163].

For each of these tests, the null hypothesis is that the observations are in random order while the alternative hypothesis depends upon whether one selects a two-tailed test - when one does not wish to predict the direction of the deviation from randomness, i.e one is content that the data is non-random - or a one-tailed test - when one wishes to test for either too few or too many runs, for example.

A conjunction of these three tests has been used because each is better than the others under certain circumstances but not under others. For example, the Turning Points test is a poor test against trend but is better against cyclicity since "turning" is a local property and would not be much affected by the whereabouts along a line of gentle trend [88].

Not a single result was significant, Table 2. Statistically, this means that there is no evidence against the hypothesis that the data is random. The phrase "the data is consistent with" the null hypothesis is also often used. This, however, does not really constitute evidence against cyclicity (interpreted in the strict sense), except, perhaps, in a roundabout way. Further discussions about cyclicity are in Section 1.3.3.5. Also, see more discussions about evidence for or against a hypothesis in Section 5.4.1.

1.3.3.3 Fraction of modules handled

The OS/360 plot was said to have "a clearly visible non-linear" upward trend which suggested that the system was growing rapidly in complexity [95] and to support the 2nd law [18]. The Omega plot had an upward trend [108], was another example of increasing complexity and supported the 2nd law as well [98] but it also exhibited a "cyclic component" [105]. The CCSS plot also "varies cyclically" but only about a constant amount [110]. That FH for CCSS did not exhibit the characteristic upward trend and was, therefore, an "inappropriate application of the complexity measure" [98], was explained by the different kind of architecture for CCSS [98]. In Section 8.3.2, we shall take up that point again and discuss the effect of a system's

architecture upon its complexity at greater lengths.

Unfortunately, it has not been possible to verify that, except for CCSS, the plot should show an increasing trend because the B plot has too few points, Executive has no plot and DOS can either be interpreted as being in 2 phases or as increasing.

We shall only apply the significance tests to the data and, as there are obvious trends in OS/360's and Omega's plots, the Runs test is inapplicable to them. There is only one significant result out of 10, Table 2, and this leads us once more to suggest that there is little statistical support for the cyclicity idea.

1.3.3.4 Size

The systems were said to exhibit persistent [18] but diminishing growth [18,97]. The "limits to growth" [108] were interpreted as due to increasing difficulty/complexity [18,97,109].

Our results, Table 1, show that, for OS/360, Executive, CCSS and DOS, the quadratic is the best fit. Of these, OS/360 and DOS have a maximum, cf. the CMH results, while EXECUTIVE (contrary to previously and this new result is also found in [23]) and CCSS are exponential. Omega, from the shape of its plot, behaves similarly. Thus, there is a nice division of 2 against 3! Could it be that only IBM products exhibit limits to growth? Or perhaps the largest, assuming OS/360 and DOS are the largest, which do so?

1.3.3.5 Net growth

The OS/360 plot was found to be cyclic with the net growth points lying within a band bounded at about the 400-module level, called the critical growth mass, which had not changed over time [18]. Any increase above this led to a subsequent decline suggesting that a system develops a resistance to growth which increases rapidly (non-linearly) with the absolute size of incremental growth [108]. In conjunction with the CCSS plot, it was used to illustrate the ideas of cyclicity, critical growth mass and invariance and forms the basis of the 5th law [98].

Our results on both these plots and DOS's, too, show little support for the cyclicity idea, Table 2.

Having looked at the two main variables, HR and NG, on which the cyclicity idea is based, it becomes appropriate to present some additional ideas which we happen to hold on the subject. If we accept that cyclicity should not be taken as meaning a sequence of above-median values alternating with below-median ones (or something to that effect) but rather that, while most

values fluctuate about the mean, the larger a value is, the greater is the probability of having a smaller value following it and that Lehman's laws should not be taken as having the same sort of accuracy as the laws of Physics or Chemistry but should, instead, be viewed as having the same sort of applicability as Social Science or Economics laws, then Lehman's main conclusions would retain all their validity.

However, we believe that the cyclicity, critical growth mass and invariance ideas as well as the 3rd, 4th and 5th laws, probably much more commonly occurring phenomena than is currently thought but not highlighted before the advent of ED, can perhaps be more appropriately expressed and summarised in terms of probability density functions.

Assume, for the present, that each of the above variables is either normally or exponentially distributed, probably two of the most frequently encountered distributions and, certainly, the two that predominantly represent most software data, as we shall see in Chapters 3 and 4. For the exponential distribution, suppose that the mean is $1/\lambda$. Then, for a given value x of the random variable X , the probability of obtaining a value greater than x , which we may appropriately term the critical value, is given by $P(X > x) = e^{-\lambda x}$, i.e. the larger the critical value, the lesser is the probability of obtaining a value larger than it, encapsulating the concepts of critical growth mass and resistance to growth when X is net growth.

For the normal distribution, as the value deviates further from the mean, its frequency grows smaller. This implies that, if the individual values are randomised, picked and plotted against the sequence in which they had been picked, then, not only is there no trend, but there are/should be an equal number of values above and below the mean, with very few values much larger or much smaller than it and the majority of values being close to it. Thus invariance is quite nicely explained in terms of this distribution but why an effective statu quo should be maintained despite all efforts to the contrary will, as pointed out earlier in Section 1.3.3.1, only be treated in an interpretative fashion in Chapters 5 and 8. The normal distribution also explains the looser (and, probably, more accurate) meaning of cyclicity [104], i.e. why a large value tends to be followed by a smaller one and that of critical growth mass, i.e. why, given a sufficiently large value x , there are few values greater than it. Thus, suppose we are dealing with the handle rate of CCSS (mean 7.07, standard deviation 3.71) and we have observed a handle rate of 13.07, then the probability of obtaining a higher value is only 5.3%.

$$P(X > 13.07) = 1 - F((13.07 - 7.07)/3.71) = .053$$

If the observed value is set higher, to 14.07, say, then the

probability of obtaining a higher value is even smaller, 3%. When one thinks about it, most phenomena, whether exponentially or normally distributed, behave in similar fashions. If the mean lightbulb life is 100 hours, then most probably one will find that, if the last bulb which one has been using has lasted 200 hours, then the next one will probably last much less. If the average height of grown-up males is 5 ft 8 ins, then most male grown-ups one meets will not exceed 6 ft 3 ins, and so on.

As to the explanations and arguments, e.g., exceeding the critical growth mass results in a bad product, penalties will have to be paid in terms of a following clean-up release, etc., these are in line with the suggested distributions.

1.3.3.6 Release Interval

The release interval for OS/360 was seen as increasing exponentially [18] and, because of increasing complexity, other systems were expected to behave likewise [109]. From the obvious shapes in Figure 7, it was not necessary to trend test the plots. It is fairly obvious that, for both IBM systems, there is evidence towards exponential increase of the release interval (cf. the behaviour of size) but Executive and CCSS do not conform: they have peculiar shapes of their own.

1.3.3.7 Size against RSN

The OS/360 plot suggested cyclic ripples which demonstrated that what was being observed was the output of a multi-loop self-stabilising feedback system [18,109]. It was later found that the Executive plot also exhibited the "ripple effect" [36] and, therefore, lent support to the observation. However, it is not easy to say whether the 5 additional points obtained from [23] show additional ripples. The subsequent developments of the theory: growth rate and work pressure build up resulting in reduction in quality of design, coding, testing, documentation, etc., and in increasing complexity and finally leading to clean-up and restructuring of the system, are very appealing. Yet one feels that, in addition to the presence of certain ripples in some of these plots, some further evidence for the multi-loop self-stabilising feedback process would be welcomed.

1.4 SUMMARY OF OUR RESULTS

- 1) Even if one has very strong reasons for doing so, the cumulative plot, as a statistical technique, should only be used with care since it is liable to be misleading.
- 2) A few of the results would be in need of some modification:-

a) We have found little evidence in favour of the permeating cyclicity. In fact, all our results seem

to point to the contrary.

b) Invariance seems to be too strong a term to describe something which does not show any determinism. Not only is the expected value important but the range within which the value may fluctuate is equally significant. For example, modules handled per day which has a range of 0 to 27 for OS/360 and one of 2.5 to 17.5 for CCSS cannot quite be described as constant or even as being "statistically invariant".

c) About the notion of limits to growth: the systems are evenly distributed between those that show increasing and those that show decreasing growth.

d) Release interval seems to increase exponentially only for the two IBM systems.

3) Encapsulating the properties of each variable within those of its probability density function seems to be more appropriate and concise. Thus, invariance simply says that most of the values are within a certain distance of the mean, that distance being the "critical growth mass" in the case of net growth. Cyclicity just means that most values fluctuate about the mean while those which deviate far from it are few and far between. Another advantage is that it can be viewed as a law about the 3rd(?), 4th and 5th laws, since it implicitly contains them. But should one prefer to be more specific and wish to highlight certain aspects of the system, e.g., work rate and release content, then one may, by all means, still think in terms of these laws. In that case, they may, perhaps, be slightly modified to reflect the empirical evidence. In Section 1.3.3.1, the 4th law was tentatively recast. In the same vein, the phrase "is statistically invariant" should once more be replaced by that of "shows no time dependent behaviour" in the enunciation of the 5th law.

Thus, although Lehman's observations are, in general, representative of the majority of the systems studied yet some of the analyses need to be tested with more accurate tools, some of the results to be tightened up, some of the terms to be more accurately described, if not fully defined, and the exceptions to the common behaviour to be explained not as exceptions but within the framework of a slightly enlarged theory, if possible. But this is what Science is all about. Any theory must eventually be submitted to empirical validation and, since there can be no ultimate theory but only improved theories, factual observations, when they do not quite coincide with those predicted from the model, can only serve to improve it.

CHAPTER 2

THE GENERAL APPROACH TO THE INVESTIGATION

2.1 INTRODUCTION

This chapter is a prelude to the descriptions of our own work, linking it with that outlined in Chapter 1. Its main purposes are to explicitly enumerate the principles which have guided us throughout this study and to provide a panoramic view of the issues of concern. More specifically, we shall take a broad look at the objectives of this study, the approach we adopted, the problem and subproblems involved, the tactics employed to solve the individual subproblems and the strategy used to crack the problem.

It goes without saying that our approach draws much of its inspirations from that of Lehman, although the writings of Sutherland [170] and Weinberg [183], among others, have been revelatory, too. We must also acknowledge that many of the philosophical considerations to be mentioned below have been practiced by Lehman during these last ten years. Indeed, he has been adopting the empirical and quantitative approaches well before the publication of Boehm's provocative paper [26], which, to most of the community, marks the start of the quantitative era. However, we have attempted to improve the method, to be more systematic, to apply more tools and so on, mainly to achieve the objective of deriving a useful methodology as mentioned as one of our aims in the next section.

2.2 OBJECTIVES

The purpose of this study has not been to propose yet another method to build up large software systems [76,131,136,137] but rather to investigate the properties (still largely unknown, despite the work of Lehman and others) of existing large software systems, to understand these, their effect on complexity and similar measures, how to control the latter, how to make predictions, and so on. In short the main objective remains still that of ED: to understand and further clarify the programming and maintenance process of large software systems.

Another objective was to attempt to settle some of the outstanding issues we were concerned with in Chapter 1. Clarifying, verifying and strengthening previous results could only lead towards the enrichment of the theory of ED. A final objective was to derive a methodology for the systematic study of the evolution of large software systems, this being the legacy of the analysis we have performed.

2.3 THE FOUR ASPECTS OF THE APPROACH

The guiding principles of Lehman's studies has been those of the systems scientist [170]. Systems science, which is not much older than ED, has attempted to be a science of sciences and tried to integrate them. The system's approach has been to avoid the polarisations which exist from one scientific discipline to another or, even, within some particular branches of the same discipline. Instead, it attempts to combine the best in all the disciplines of science. We shall not be able to discuss all the facets of systems science; we restrict ourselves to a brief discussion of four of the main aspects of the approach which one may take to solve a complex problem:-

1. Congruence of the tool with the problem

As far as possible, we stuck to the fairly obvious principle of allowing the properties of the problem at hand to determine the analysis and the selection of instruments. If not powerful enough, these lead to ineffectiveness while too powerful tools would result in a waste of our resources - something we could ill-afford.

2. Combining the analytic with the synthetic approach

Historically, scientific advancement has mostly been through partitioning real-world phenomena into smaller and smaller units which are then studied in fine details in the belief that the analysis of individual parts is more manageable than the synthesis of the whole and that the global picture can eventually be built up from the different analyses. Although this "Divide and Conquer" approach undoubtedly works for simple systems, yet it is woefully inadequate for systems with any degree of complexity. Although necessary for elucidating and unravelling the mechanics of the global behaviour, analysis is certainly insufficient for large software investigations. It has thus been complemented with that other aspect of the system's approach - that of taking a global view.

3. Interdisciplinary

Recognising that systems problems rarely fall within the neat but artificial demarcations of academic disciplines, a search for the best tools available from a wide spectrum of disciplines was undertaken.

4. Syncretic

As a rule, quantitative methods were preferred but when quantification was not possible, we had no option but to use qualitative constructs. Ideally, a good mix of both

constructs would provide an adequate vehicle for investigation and this is what we attempted to achieve.

2.4 BLENDING THE EMPIRICAL-INDUCTIVE WITH THE HYPOTHETICO-DEDUCTIVE APPROACH

On a broader dimension, we concur with Sutherland's [170] strong recommendation that both of the two usually isolated arenas of scientific enterprise should be used: the empirical-inductive and the hypothetico-deductive methods.

The empirical-inductive method, which has monopolised science since Bacon's and Galileo's "Age of Reason", attempts to elicit theory from facts and facts from observations, such that the derogatory influences of prejudice, opinion or bias are effectively minimised. Unfortunately, although facts are verifiable, the empiricists are often at a loss to explain their results: "This is demonstrably so but we cannot indicate its significance". More importantly, the more complex phenomena, by their sheer size and complexity and by that of the work which their analyses would entail, are not proper targets for the empirical-inductive method. Thus, if scientists continue to bind themselves to this method they may never get to grips with the most significant of the systems which plague our age: the organic systems which include software systems, the objects of our special interests.

Therefore, to have any hope of solving large systems problems, we are forced to consider the hypothetico-deductive method which is characterised by the generation of hypotheses through the intellect: imagination, fabrication, intuition, speculation and so on. These hypotheses sometimes border on the fantastic and, even when severely restricted, may be either too difficult to verify or simply unverifiable. But, provided that the concepts arrived at by the philosophers of science are not treated as dogma but as true hypotheses whose ultimate validity will depend on the extent to which they will be subsequently empirically validated, this method has always been accepted as being scientific.

Just as nothing is gained by restricting our competence to a single discipline, to either quantitative or qualitative methods, to either analysis or synthesis, both modalities have been considered. As they are complementary rather than competitors, little can be achieved by using one to the exclusion of the other.

2.5 THE NEED TO EMPHASISE THE EMPIRICAL METHOD

In spite of our expressed intention to blend the empirical with the deductive method yet we had to put the emphasis on the

former mostly because of the counter-intuitive nature of software and because of the relative abundance of (mostly untested) ideas and paucity of empirical results in our branch of science. Boehm [26] gave various examples of software's counter-intuitiveness and expressed his surprise at the little effort that had gone into collecting and analysing data. Five years after Boehm, Belady [12] was still preaching that the most important next step in software science was to gather and analyse facts.

2.6 THE EMPIRICAL METHOD

Software engineering being still in its infancy, we took the view that the goal for any experimental work was primarily to gather system knowledge. This was the second reason which caused us to rely more heavily upon the empirical method which is more indicated, at least initially, for entering into largely unexplored fields, despite the weaknesses outlined in Section 2.4.

The empirical method is generally characterised by the collection of data concerning the phenomena in the domain of discourse without much idea of what to expect and before much speculation as to their significance. From the analysis of the observations, similarities and differences that arise lead to a classification scheme, to the generation of hypotheses and to the development of models. These are said to be data driven since they are in the spirit of curve fitting and it is only through data that evidence can be built up for or against any thesis. However, although they represent only empiricism, these models are the starting points which will aid understanding of the phenomena and stimulate theoretical investigation.

According to this method, any hypothesis one arrives at will further be used to structure and discipline an experiment for its empirical validation. Unfortunately, really worthwhile experimentations with software do not seem to be possible. They require such a level of detail that few can afford it, the problem is extremely difficult to isolate and the extrapolation from small to large projects is not meaningful [148].

2.7 DECIDING WHICH VARIABLES TO MEASURE

Let everything that can be objectively identified be called a variable and let us assume for one instant that all variables are measurable, then we are faced with the immediate problem of what to measure. Choosing (by intuition?) an a priori set of variables to measure seems unsatisfactory as:-

"Experimental Computer Science is a young discipline. There is little experience to draw upon when deciding what attributes of a software system are important to measure

with respect to maintenance performance" [61].

Weinberg [183] rightly points out that:-

"When you don't know much about a subject, you don't know in advance what to measure. So your early experiments are not what they might seem to be - they are not to measure things but to determine what can be measured and what is worth measuring".

He may be indirectly preaching the measurement of everything that can be perceived although he warns that "see it all" does not mean "know it all" for knowing means to have the ability to extract the essentials from the unnecessary details. By measuring and examining everything the researcher may be sure to see the fine details but his view may become cluttered and he may easily get bogged down in irrelevant matters so that he misses the significant features and hence the big picture. Fortunately or unfortunately, because there is a limit to the availability of resources at one's disposal, one may hardly proceed to exhaustively examine each and every possible variable. One may, however, make a bold attempt at achieving that unreachable goal provided that Weinberg's warning is observed. Then, whenever some variable does not lead to any concrete result, this, in itself, is significant since we, at least, know that the variable in question does not have any importance and may be discarded from our model.

Therefore a suitable compromise is required. On what basis, then, do we choose the variables to analyse and those not to? From the vast number of possible variables which may be studied, the natural tendency is to reject those that appear to be too insignificant or too small to be of any importance. But importance is not necessarily associated with the conspicuous or the large: the physiologist may not ignore the pineal gland while studying the brain nor may the biologist ignore the DNA from the study of the living cell.

One may start with a suitably chosen set of variables to investigate a certain trait of the system. By a process of successively increasing or decreasing the number of variables in the set one may arrive at a set over which the trait has remained fairly stable. This trick is extensively used in the well-known method of stepwise regression model building.

Another well-known method in new fields of research to reduce the number of variables to a manageable figure is to find the zero-order correlation matrix of a large set of variables and then to discard those variables which have low correlations with the variable of interest. Unfortunately, as Kendall and Stuart [87] have pointed out, this procedure may be seriously

misleading. Whilst the whole set of zero-order correlations completely determine the whole set of partial correlations, the small zero-order correlations of the variable of interest with the other variables do not guarantee similar higher order coefficients. We may thus be throwing away some very valuable information.

To complicate matters, because of the distinct possibility that the set of significant variables may change from one instant to another, there is the time factor to consider as well. This implies that, even if the Principle of Parsimony is adhered to, we are forced to consider many more variables than required for a parsimonious model.

2.8 THE EXTENT TO WHICH VARIABLES CAN BE MEASURED

Let us now assume that unlimited resources are available for measurement and analysis so that one may attempt to measure everything. Then, one is justified in asking to what extent are the phenomena visible in the first place and quantifiable in the second.

In the physical sciences, everything is expressible in well defined and precisely measurable units of size, time and mass. Economics, Sociology and Computing are less well blessed: we are forced to think in terms of value, utility, quality, complexity and so on and have to recognise that even if these concepts are quantifiable there is something essentially subjective about them. Some, for example, manpower or work are measurable in principle yet, even when the primary units, e.g., human beings or changes are countable, some regard must be given to quality if we are to be at all realistic. Others which embody psychological values, e.g., complexity, are less capable of measurement. Still others such as those embodying high ethical principles are much less adaptable to measurement if at all. Sorokin [167] not only states that sociocultural phenomena are empirically transparent but asserts that they are generally unamenable to treatment by quantification. He has therefore cautioned against the importation of methodological principles or analytical instruments from the hard to the soft sciences.

Thus, although we shall do our utmost to quantify as much as possible, following the advice of Boehm [26] who quotes Lord Kelvin's famous words:-

"When you can measure what you are speaking about, and express it in numbers, you know something about it; but when you cannot measure it, when you cannot express it in numbers, your knowledge is of a meagre and unsatisfactory kind: it may be the beginning of knowledge, but you have scarcely, in your thoughts, advanced to the stage of science",

and Gilb [65] who quotes:-

"The measure you give will be the measure you get, and still more will be given to you", (St Mark, 4:24),

yet, aware of the difficulties of measuring soft science variables and of the limited applicability of quantitative methods, we shall have to supplement the latter with qualitative methods as declared earlier in Section 2.3. In this sense, we shall not only escape the narrow confines of the physical sciences but shall at last make a stab at solving more complex and worthwhile problems. Thus Platt [141] as quoted by Weinberg [182] is probably right in saying:-

"Today we preach that science is not science unless it is quantitative. We substitute correlation for causal studies and physical equations for organic reasoning. Measurements and equations are supposed to sharpen thinking, but ... they more often tend to make the thinking non-causal and fuzzy. They tend to become the object of scientific manipulation instead of auxiliary tests of crucial inferences.

Many - perhaps most - of the great issues of science are qualitative, not quantitative, even in physics and chemistry. Equations and measurements are useful when and only when they are related to proof; but proof or disproof comes first and is in fact strongest when it is absolutely convincing without any quantitative measurement.

Or to say it another way, you can catch phenomena in a logical box or in a mathematical box. The logical box is coarse but strong. The mathematical box is a beautiful way of wrapping up a problem, but it will not hold the phenomena unless they have been caught in a logical box to begin with."

Jevons [80], earlier than any of these, sums up the arguments thus:-

"Every question of science is first a matter of fact only, then a matter of quantity and by degrees becomes more and more precisely quantitative."

As to the quantifiable variables, they are measurable on any of four scales which, ranging from the strongest to the weakest are the ratio, interval, ordinal and classificatory scales. It is obvious that the stronger the measurement scale the more powerful will be the conclusions from each analysis [163].

2.9 STATISTICS GATHERING

Having shown a preference to start with the gathering of real data, and fully aware of the problems of choosing which variables to measure as well as the problems of measurement, we then had to confront the problem of deciding whether to observe real-world applications and gather data as we proceed with the observation or to limit ourselves to digging up and analysing what had already been recorded and is available.

Apart from sharing the problems we have outlined in Sections 2.7 and 2.8 above, each method has its own disadvantages. About the former, it is probably true to say that a complete and intensive observation of an actual system may require both a long time and so many people that it would not be suitable for a PhD research program. Although, after living the system for some time, one would start to understand it and, therefore, be confident enough to limit the observations to only those felt to be significant, yet this would tacitly assume a swing towards hypothetico-deductive methods. A final problem is that of the interference on the process itself. This is known to social scientists as the "Hawthorne Effect" and it occurs in all sciences where human interactions are involved. However, this factor has largely been neglected in our field, with the possible exception of Weinberg [182] and Lehman [96] who goes one step further. He has no doubt about the applicability to software systems of Heisenberg's "Principle of Uncertainty" whereby:-

"The mere act of observing leads to a change in the system which, in the sense of exact science, results paradoxically in the invalidation of the earlier results and models".

About the latter method, we would have to resign ourselves to the inevitability of forsaking some much desired information because that information was never considered to be of enough importance to be recorded in the first place. About what is available, one may find that they are scattered, inaccurate, incomplete, not manipulable, not easily retrievable, may require some transformation into a suitable form for analysis and may be as costly to gather as in the former method: too much data, too complicated to extract and so on. Despite these difficulties, historical data have mostly been the type of data available and, like previous researchers, these were what we had and we had to put up with them.

2.10 THE HISTORICAL DATA BASE

We could neither extract all possible variables nor, more ideally, the exact set of significant variables from the available historical data base. Some variables were in the wrong scales or were not the ones we would have preferred. Worse: they could have formed an entirely wrong set of

variables. Therefore, since the subsequent analyses and conclusions are functions of and limited by the quality, richness and amenability, among other things, of that data base, we could not afford to lose further precious information. In that spirit, we had to make the best use of what was available and an effort was made to extract all that we could and to measure all that was measurable from the data base, even at the risk of having to deal with too many variables, many of which may not have been relevant. But, since we are still exploring the mostly unmapped, the non-relevance of these non-relevant variables is, in itself, relevant!

We were able to study two systems. For each, the data were in fact held in more than one data base. These have been described with sufficient thoroughness in several of our progress reports [37,46,47,49,50] and briefly recapitulated in Chapters 3 and 4. Here we only wish to emphasise two points.

The first concerns the frustrations, the loss of time and the wasteful exertion of energy caused by the previously outlined problems that are inherent to manually recorded data. In view of these difficulties it is not surprising that similar large scale data retrievals have rarely, if ever, been undertaken. We, therefore, strongly recommend that future data be not only more consistently and accurately recorded but be in a form that can be easily, and perhaps automatically, retrieved. Otherwise, we fail to see any progress being made, as even the most dedicated and persistent researcher will soon give up hope.

The second point is that all the data relate to the maintenance phase of the software life cycle. Thus, perforce, our work will have to be restricted to that particular phase alone but this does not need to worry us. Besides that maintenance is precisely the subject of ED studies, there are several other good reasons for studying it. The high cost of maintenance quoted as 49% of all software effort by Leintz and Swanson [111], as 60% by Wolverton [185], as more than 60% by Boehm [26], as 75% by Staudhammer [168], as 2 to 4 times the cost development by Wasserman [181] and as rising to about 80% of total costs by 1985 by Morrissey and Wu [127] shows that, unquestionably, the most important factors in maintenance need to be examined closely and that "it is necessary to work towards prevention and reduction of costs associated with system evolution" [181]. There is also more potential payoff in improving the efficiency of this phase than in any other [26]. Wolverton [185] went so far as to state that, although maintenance is the most visible and costly phase, it is also the least documented and understood. Therefore, our investigation stands a fair chance of not only contributing to the enrichment of the field of Software Engineering but also of being of some practical utility.

2.11 THE TYPES OF DATA AVAILABLE AND THE ANALYSES ENVISAGED

The type of maintenance data we have gathered can be divided into two main categories. There are those that concern structure: the components of the system can be counted, related to each other or examined individually. Also available to a certain extent are data concerning changes in structure such as the increase or decrease in size of each component or of the system itself from one point in time to another. The other category mainly describes the types of activities that have been going on. Quite a few of these concern changes such as the fixing of bugs or local enhancements to the code but other activities may not involve any modifications to the system at all.

Although the time is not ripe to get involved with the details of the analysis, it is however necessary to briefly outline the aspects of the system that may be investigated from those data.

The structure data lead to the study of the structural dimension of the system where the primary aim is to draw a map showing which components comprise the system and the nature of the prevailing relationships among them. However this only provides a "snapshot", i.e. a static picture of the system. Having spoused the cause of ED, we are inevitably committed to consider the dynamic aspects of the system as well although this is bound to be a difficult assignment: no one besides Lehman has concerned himself with a system's dynamics and time based techniques are not only few but, when available, they require some rather advanced statistical sophistication from the practitioner. Thus, on the structural front, dynamic analysis would involve the articulation of transformation functions that explain the way in which a given system state translates into another system state over time.

On the activity front, we may aggregate each measure by an appropriate time period to obtain a time series from which we may attempt to detect any sort of observable or deterministic behaviour such as a trend, cyclicity or seasonalities. We may also be interested in any feedback process within each activity, i.e. whether the intensity of the activity at any time is influenced by any previous happenings. Then, we are interested in the nature of the relationships among the activities, pairwise or several together and in any changes in the causal patterns or interactions prevalent among these. Finally, one must attempt to see how the structure dynamics are related with those of the activities. After all, the ultimate aim is to obtain a model of the system behaviour in terms of the activities which affect it.

It is premature to discuss here the theoretical aspects of the analyses we performed nor is it appropriate to describe which

tool was used to detect which feature of the data. We only wish to state that, being more interested in gathering system knowledge, we explored as many tools as we could and took a not too dogmatic approach, on occasions permitting ourselves to be less rigid than required for, say, formal statistical inferences. Nevertheless, the methodology we applied in arriving at the conclusions is perhaps as important as the conclusions themselves.

We thus proceed forthwith to summarise the separate analyses we performed on the two systems we have studied. Only after these have been described, in Chapters 3 and 4, shall we attempt to discuss, in Chapter 6, the data, tools and techniques together with their weaknesses. Otherwise the reader may not know what we are talking about. Since empirical results should not be reported as isolated facts without any attempt at generalisation and integration, the results are pooled together and interpreted as much as possible in Chapter 5. Unfortunately, because only 2 systems with sometimes very different data were investigated, the generality of the results is somewhat restricted. But, combined with previous observations, these allow us to attempt, in Chapter 8, a comprehensive theory of large systems' dynamic behaviours, however, only after we have explored some possible ways of pursuing this work in Chapter 7.

CHAPTER 3

ANALYSIS OF THE BD SYSTEM

3.1 INTRODUCTION TO BD3.1.1 The relative simplicity of the system

The BD system was built, about ten years ago, by Centrefile Limited, a subsidiary of the National Westminster Bank, with the intention of automating the accounting aspects of the borrowing and lending schemes of Building Societies and of servicing the latter. Several of its features contribute towards making this system relatively more stable than those which have already been studied:-

- 1) The Building Societies' functions are laid down by law. They also have their own association which may enforce additional rules to ensure uniformity. It is therefore not surprising to find that, in general, the Building Societies make basically similar demands upon the system.
- 2) Although, from time to time, changes do occur in the Building Societies' statutes and operations, these are both infrequent and do not lead to major upheavals.
- 3) Centrefile, the system implementer and owner, retains control over the system and is really its sole user since the Building Societies do not have hands-on access to it. They can more accurately be described as clients of Centrefile since, for a fee, the latter arranges for their inputs to be processed, their files to be kept and their outputs to be produced on its own computer situated at its own premises.

Thus, 1) indicates that, with the well defined missions and operations of the system, implementing its computerised version, BD, was not a major difficulty and that most of the common defects due to the inaccurate modelling of the real world can be avoided while 2) and 3) ensure its relative stability. In addition, compared with most of the systems studied within the ED context, BD is really quite a moderately-sized system.

3.1.2 BD is a collection

Another major aspect in which BD differs from other systems (except CCSS) is that it is really a suite of programs which are physically independent of each other but which are, nevertheless, still linked, though very loosely, in the sense that the output from one may become the input to another. This type of system is fairly common of commercial data processing systems though not of scientific, real-time or operating systems.

We, therefore, have a collection of programs not all of which may be used for any given run and whose scheduling will be determined by the needs at hand. In fact, each program's functions and importance will determine its usage frequency: some are essential and run at all times while others may be optional and very infrequently run. For example, the real-time "module", which is the centrepiece of the system and which captures most of the input data, is run during all normal office hours. Other essential programs are run nightly to process the data captured by the real-time module and to do the updating to the accounts, among other tasks. Further essential programs will be run weekly, monthly or half-yearly (e.g., posting of interests to each investor's account). The optional modules may include those which reorganise the files, the odd-job ones and so on.

3.1.3 Classifying the components of the system

Thus, it is possible, as done by Centrefile, to artificially divide the modules into physical subsystems by the criterion of frequency of usage, but partitioning by some other means such as by the programming language used is also legitimate - because of their independence property, some modules were implemented in Cobol or PL/1 although most were in Assembler.

Each Assembler module is generated by a top-level "micro", the module generating micro, which calls other micros which call other micros, etc. This technique allows each unit of code, the micro, to be the product of one assembly.

It is possible that a micro may represent a function common to several modules. It will then be link-edited to all of them. This commonality property may provide one partitioning criterion for the micros. The classification property which Centrefile has used depends upon the micros' functions instead and the following interesting classes, among others (see [37] for more details) are obtained:-

- Non-executable micros, e.g., tables.
- Bats, i.e. the IBM-provided real-time micros.
- Tugs, the Centrefile standard output routines.
- Macros.
- Dummies.
- Aliasing micros (same function but differently named micros).
- Hll (which are the high level language modules).
- Withdrawn - no longer used micros.
- Ordinary/core/normal micros.

3.1.4 BD is non-release based

As should now be clear, there was no justification for the release concept within the context of a system such as BD. Each and every module was modified as and when required, albeit with due regards to the possible effects of the modification upon the other modules which share the same data base and/or some common micros.

Thus, any data which we collected could only be investigated on an actual time basis, contrary to the release to release basis which has been the domain in which all previous ED studies have been carried out.

3.1.5 The available data

There exist 3 main sources of data, all of which were adequately complete for a static study as at the time of the survey but, for only one of them, were there sufficient archival records to justify a dynamic analysis. This concerns the amendments to existing micros or modifications records which have been preserved since the beginning of 1973. Listed on each record are the amendments made (there can be more than one amendment per record), the micros amended by each amendment, the types of amendments as well as the date.

Then there are data concerning the micros, see Section 3.4 for a full list. Finally, we have data concerning the modules themselves: the date each is implemented (and withdrawn), its constituent micros and its size. For the latter two items, we have a few lists which, combined with the information gleaned from the other two sources, allow an approximate recreation of each module's and the system's past sizes in micros, record counts and bytes. The analysis of these reconstituted data is in Section 3.3.

3.2 ANALYSING THE CHANGES DATA

From the amendment records, these monthly series were extracted: the changes, enhancements, fixes, micros changed, micros enhanced and micros fixed series (c, e, f, mc, me and mf respectively) and from them, 3 further ones mc/c, me/e and mf/f (average micros changed per change and so on) were derived in the obvious manner.

Note See [37] about 1) the reason why shorter periods of aggregation, e.g., 1 week, have not been investigated and 2) the Centrefile change classification scheme.

The approach which will be commonly adopted for all time series investigations will be:-

- 1) To look for patterns such as trends, seasonalities, cyclicities and stochasticities in each series.

2) To fit the family of polynomials to the actual series itself and to its corresponding cumulative series.

and, depending upon the success of the above:-

3) To attempt time series analysis, including the pairwise crosscorrelation of the series, in both the time and the frequency domains.

4) To fit typical distributions to the data.

3.2.1 Looking for patterns in the series

The first stage involved examining each scatter diagram but this did not suggest any observable pattern amidst the wild fluctuations in the data (Figures 9 to 12). We, therefore, proceeded to the statistical testing of each of the series, as in Section 1.3.3.2. Note that, although not formally included here as randomness tests, the autocorrelation function, acf (or correlogram), the periodogram [29] and the polynomial fits are further techniques which we have used (see later).

Whether on a 6-monthly basis or on a monthly basis, we did not obtain a single significant result, i.e. in each case, the data are consistent with the randomness hypothesis. The relevant statistics for mc, me, mf, c, e and f are in Tables 3 to 5.

3.2.2 Fitting the best polynomial to each series

In addition to the fits to the actual series, the best polynomial fit to each of cmc, cme, cmf, cc, ce and cf (Figures 13 & 14), the cumulative series of mc, me, mf, c, e and f were also found. All the results for the monthly series, including two further series, the micros handling fraction, mhgf, and the micros handled fraction, mhdf (see Section 3.4.2.1), are in Table 6.

Although the number of changes per month shows no trend, that of enhancements is increasing while that of fixes is decreasing. The ratio of enhancements to fixes is therefore increasing. There is a very slight decreasing trend in mf while me and the overall mc show no trend. Thus, of the 11 actual series, only mf, e and f exhibit trends and require quadratic fits (which are the integrals of the straight line fits for the actual series) for their cumulative series. Each of the others was best fitted with its mean while its cumulative series was invariably best fitted by a straight line with its slope equal to the mean.

As the linear fits to mf, e and f gave low R^2 values (11.1, 22.3 and 14.2% respectively), other types of fits (polynomials of higher degrees, exponential, logarithmic, etc.) were

attempted but did not result in improved R^2 [38].

One can either assume or attempt to show that, for series like e, the disturbances are normally distributed. The disturbances can be represented by the residuals from the best fit but, because of the low R^2 values, it is futile to analyse the residuals from our fits. (Differencing the actual series is an alternative method of removing the trend but see Section 3.2.3.1.)

Thus, the conclusion must be that, mostly, there are no trends and that, although a trend may exist in some series, it is mild and dominated by the marked random fluctuations imposed upon it.

Although our randomness tests do not suggest that the series resemble anything else except random series yet the conclusion cannot argue in this favour in any definite sense (see Section 5.4.1). Stronger evidence for or against cyclicity and seasonalities may be provided by the autocorrelation function. This, however, is part and parcel of the time series analysis in the time domain which is discussed next.

3.2.3 Time series analysis

Both the time domain [29] and the frequency domain [25,79] approaches have been adopted because, although complementary and equivalent, they are capable of shedding lights on different aspects of the data.

3.2.3.1 Analysis in the time domain

To apply time series analysis, the series should be what is called stationary (roughly "no trend"). As observed in Sections 3.2.1 and 3.2.2, this condition is satisfied in all cases except mf, e and f.

Differencing is a technique used by Box and Jenkins [29] to remove any trend in a series and to bring about stationarity. However, with each of mf, e and f, differencing increases the standard error (see Table 7 for that of e*, the differenced (d=1) series of e), indicating that white noise is being added to the series. For each of these series, the trend, although significant, is, therefore, sufficiently small for it to be considered stationary for time series analysis purposes. However, both the differenced (d=1) and actual series have been analysed.

The autocorrelation results indicate that no acf value for any variable is significant at the 5% level except for e at lags 1 to 7 (Table 7). These are due to the stronger trend in e and implies that it should be differenced, despite the increasing

standard error. The results for the other variables tend to support the hypothesis that they are independent random processes and, therefore, that building time series models for them will largely be futile, the essence of time series analysis being that successive observations are dependent so that future values may be predicted from past observations. None of the acf and pacf (partial autocorrelation [29]) plots would therefore resemble the characteristic plots which could give a clue to the best model required and they did not.

Nonetheless, some effort, if only to demonstrate its futility, was required to attempt to identify a good model. There was no other means than to build up several of the simpler models, none more complex than ARIMA (2,1,2), the least parsimonious we could afford to experiment with, and choose the best on the criteria described in [41]. Because of the time consuming process, not all series could be modelled. The best models for those which were investigated are in Table 8.

We started with m and fitted 6 models, (1,0,0), (2,0,0), (0,0,1), (0,0,2), (1,0,1) & (2,0,2), to the data. We found that none was adequate. The barely passable ones are (1,0,0) and (0,0,1) but, since their estimators are non-significant, they reduce to

$$z_t = 23.70 + a_t \quad \text{where } a_t \sim N(0, \sigma^2)$$

, lending further support to the theory that the process is simply white noise.

Similar results and arguments apply for c : for both (1,0,0) and (0,0,1), the estimators are both small and redundant while the other models are gross overfits. The best model is therefore

$$z_t = 13.32 + a_t \quad \text{where } a_t \sim N(0, \sigma^2).$$

Eleven models were fitted to e and e^* , the best being

$$z_t = z_{t-1} - 0.79a_{t-1} + a_t.$$

That it has no z_{t-2} term but the two terms a_t and a_{t-1} indicates the dominance of stochasticity over autoregressive influence. It is shown below that, for series showing stochastic trend (as opposed to deterministic trend), IMA of order 1, (0,1,1), is theoretically the best model.

Of the 10 models fitted to f and f^* , no good one was obtained for the actual series while of the differenced models, (1,1,0), (2,1,0) and (0,1,1) were adequate, the best being again (0,1,1),

$$z_t = z_{t-1} - 0.83a_{t-1} + a_t$$

, although the white noise model would have been just as good.

In conclusion, we deduce that, from the retained models, stationary series like m and c can be considered as white noise, best described by fits of the type

$$\nabla z_t = \mu + a_t \quad a_t \sim N(0, \sigma^2).$$

This result is in accordance with those in Sections 3.2.1 and 3.2.2.

A significant estimator for the constant term, θ , in $(0,1,1)$: viz.

$$\nabla z_t = \theta + a_t - \theta_1 a_{t-1}$$

, would indicate a deterministic linear trend in the presence of non stationary noise [29]. That there is no such constant in the model fitted to either e or f implies that, for each series, the description of a stochastic trend, typified by random changes in the level and slope, is better than that of a deterministic trend.

There is, a priori, no particular reason why a deterministic component should be present, a stochastic trend being probably more realistic as it does not require the series to follow the pattern it has developed in the past.

The most common form of the $(0,1,1)$ process, $\nabla z_t = (1-\theta B)a_t$ is in terms of previous z 's and current and previous a 's by direct use of the difference equation. This is

$$z_t = z_{t-1} + a_t - \theta a_{t-1}$$

which we have been using. However, expressing z_t in terms of current and previous shocks a 's allows some special aspects to be appreciated.

$$1-\theta B = (1-\theta)B + (1-B) = (1-\theta)B + \nabla = \lambda B + \nabla$$

where $\lambda = 1-\theta$

$$\nabla z_t = (\lambda B + \nabla)a_t$$

$$\Rightarrow \nabla z_t = \lambda a_{t-1} + \nabla a_t$$

On summation, this yields

$$z_t = \lambda S a_{t-1} + a_t$$

$$z_t = \sum_{j=0}^{\infty} \psi_j a_{t-j}$$

$$\text{where } \psi_0 = 1 \\ \psi_j = \lambda \quad \text{for } j \geq 1$$

Since the process is nonstationary, it possesses no mean but λSa_t can be thought of as measuring the level of the process at time k and as telling us what is known of the future value of the process at time t based on knowledge of the past at time k . At time k , our knowledge of the future behaviour of the process is that it will diverge from this level according to $\lambda \sum_{j=k+1}^t a_j + a_t$ whose expectation is 0 and whose behaviour we cannot predict.

Now, the two equations

$$z_t = \lambda Sa_{t-1} + a_t$$

$$\text{and } \lambda Sa_t = \lambda Sa_{t-1} + \lambda a_t$$

provide a useful way of thinking about the generation of the process. The first shows how, with the "level" of the system at λSa_{t-1} , a shock a_t is added at time t and produces the value z_t . However, the second equation shows that only a proportion, λ , of the shock is actually absorbed into the level and has a lasting influence, the remaining proportion $\theta (=1-\lambda)$ of the shock being dissipated.

By some elementary algebraic manipulation, it can be shown that

$$\lambda Sa_t = \lambda z_t + (1-\lambda) \lambda Sa_{t-1}$$

, indicating that each new level is arrived at by interpolating between the new observation and the previous level. If λ is equal to unity, $\lambda Sa_t = z_t$ which would ignore all evidence concerning location coming from previous observations. If λ is close to 0, λSa_t relies heavily on λSa_{t-1} , which would have a weight $(1-\lambda)$. Only the small weight λ is given to the new observation. From our results, λ is small (.21 & .16 respectively).

3.2.3.2 Analysis in the frequency domain

The periodogram for each series shows no or extremely few frequencies with great intensities. Table 9 summarises the salient features of the periodograms by listing all the significant periods, while giving, together with other statistics, the percentage variation which each of these explains.

Notes

- 1) The low percentage of variation explained by each or by all of the significant periods in each of the series indicate that quite a lot of the variation can be accounted for by random noise.
- 2) The large amount of power concentrated at the

fundamental frequency of the e series can again be explained by its trend. The f series behaves likewise but, as its trend is less strong, so is the power at the fundamental frequency.

The overall impression, which ties up with all our previous results, is that most signals are weak and are rather uniformly scattered over the various frequencies. The cumulative periodograms, Figure 15, illustrate this point strikingly: in most cases, the dots hug the central diagonal, the plot expected for a pure noise series, very closely. Only for e does the actual cumulative periodogram cross the (10% confidence) limit lines. Although f and mf also show some deviation, they do not cross these.

Crosscorrelating [29] the series by pairs provided mostly non-significant results. At the 95% confidence limits, only values outside the approximate range $[-2/59, +2/59]$, i.e. $[-.260, .260]$ are significant while, at the 99% confidence limits, the non-significant range becomes $[-.335, .335]$. The significant results at $\alpha=.01$ are in Table 10.

Of a total of 14 significant crosscorrelations involving the different pairs of the 6 series, 9 are at lag 0, i.e. are the ordinary simple Pearson's correlation while 3 of the remaining 5 involve e^* , whose behaviour is largely influenced by the smoothing method employed, e.g., see Godfrey [67]. In conclusion, it seems that this work has shown that significant crosscorrelations are unlikely and that, even if they are present, they can be expected to be weak.

Given the mostly non-significant Fourier frequencies and crosscorrelations, there was no need to proceed to the more complex spectrum or cross spectrum analysis [79].

Note Analyses for 5 further series: new "normal" modules, new "normal" micros, new "all" modules, new "all" micros and net growth in bytes, were also carried out but the results, all non-significant, are not included here (see [45] for the detailed results).

Thus, while the work in the previous two sections suggested that the patterns in most of the series were random, here we seem not only to have found confirmation of these results but also to have shown the absence of interdependencies among the variables and dependencies of future on past values as well. It then seems that the series we have examined can, for most intent and purposes, be treated as random variables.

3.2.4 Fitting the normal distribution to each variable

The next logical step is to attempt to fit a normal distribution to each series to see whether the usual assumption that the disturbance in any model is normally distributed is plausible. Strictly speaking, the normal distribution is for continuous random variables but it has often been fitted to discrete random variables which are treated as if they are continuous. Quite a few of our series are made up of discrete values. Furthermore, the data must be truly random and independent. Despite that these properties are violated to some extent by mf, e and f yet the trend in each series is weak enough for us to attempt the fit.

Except in the case of me, very good fits are obtained for all 6 variables. In fact, the goodness of fit, in each case, was comparable to that obtained for corresponding pseudo-randomly generated data (see [42] for the details).

Why the normal distribution is not a good fit to me can be observed from its frequency distribution plot: months with moderately large me's are fewer while those with small me's are more frequent than expected. The mc/c distribution shows a similar behaviour although that of e is as expected. The apparently anomalous behaviour of me (and me/e) is, at present, unexplained but we shall see later, Section 3.4.2.2, that there is a definite evolution towards smaller me (and me/e) although e itself shows no trend.

3.2.5 Looking at the CCSS data

CCSS is the only previously analysed system for which sufficient data exist for rigorous statistical tests. As significant differences from or similarities with BD may be very meaningful, the analyses described in Sections 3.2.1 to 3.2.4 were repeated for this system. However, two problems exist with its data. They do not form discrete time series and there are missing values in some of these.

These 6 variables were investigated: release interval, modules changed, fixes/day, fraction of modules handled/day, net growth/day and handle rate (modules changed/day).

Release interval, whether the whole, the pre-release 27 or the post-release 27 (release 27 is when management decided to lengthen the interval [72,110]) sample is taken as well as fixes/day, net growth/day and handle rate gave "ideal" results: no significant results from the randomness tests, no trend in the series and a very good normal fit. Modules changed has a good normal fit and shows no trend but appears to be cyclic from the results of the Turning Points and Runs tests. For fraction of modules handled/day there is a slight decreasing trend. Note that a somewhat similar measure was used as an approximate measure for complexity in the analysis of OS/360

[18]. Clearly either this measure is inadequate for CCSS (see [98] for the explanation in terms of system architecture) or its complexity is decreasing over time.

The remainder of the analysis was only possible and carried out on the three longest series: release interval, modules changed and fixes/day. No significant autocorrelations were obtained except with modules changed whose previously detected cyclicity is reflected in significant negative autocorrelations at lags 1, 4, 6, and 9 (see [45] for the full results). At the 95% confidence limits there was a total of only 4 significant but very weak crosscorrelations (Table 11). At the 99% confidence limits, none was significant. Fourier analysis shows that the percentage of variation explained by each of the significant frequencies in the CCSS series (Table 12) are, in general, higher than those obtained for the BD series. This could very well be taken as indication of a fundamental difference between the two series. The following interpretation is tempting: BD, being "simpler" in most of its aspects, smaller system size, smaller size of maintenance team, smaller number of users and simpler external environment, among others, than CCSS, reacts much more readily to random influences while CCSS, a much "larger" system, has a moment of inertia and a dynamic behaviour which are less easily disturbed so that its behaviour on the whole is more regular than that of BD.

3.2.6 Summary of the results

Of the 12 series (6 BD and 6 CCSS) considered, there is a suggestion:-

- 1) Of a mild trend in mf, e, f, and fraction of modules handled/day.
- 2) Of cyclicity in modules changed.
- 3) That the normal distribution is a good fit to all except me.
- 4) That, despite the behaviour of some of the series in one way or another, the most striking feature which we have observed is the apparent strong stochastic element in each of the series. As pointed out in Section 1.3.3.1 where we first exposed the idea of stochasticity, the cause of this behaviour, probably surprising to most, and certainly to those engaged in work whose purpose is to ensure that this should not be so, will only be tackled in later chapters.

The questions which still need be answered are:-

- 1) Do the similarities between BD and CCSS extend to all

other programming systems?

2) Are the differences significant? If so, what are the factors which cause these? Do, as suspected, the sizes of the system and of its environment play significant parts in affecting the system's behaviour?

It was mainly as an attempt to provide some answers to these questions that we undertook the analysis of B (see the next chapter), a large system for which an adequate amount of data exists.

3.3 ANALYSIS OF THE SIZE MEASURES

Eight alternative measures were used for size and plotted as functions of time. The first three are:-

$\sum_{all\ mods} mods$, i.e. the sum of all the modules, each module counting as 1, Figure 16,

$\sum_{all\ mods} mics$, i.e. the sum of all the module sizes in micros, each module size being measured by the number of micros it contains, Figure 17,

$\sum_{all\ mods} bytes$, i.e. the sum of all the module sizes in bytes, Figure 18.

Since the system comprises certain non-typical micros (e.g., tugs and hll, Section 3.1.3), these three measures are repeated by excluding the unusual micros to find out what is the effect on the system of these. The terms "core" modules and "core" micros have been used to describe modules and micros which are not atypical while "all" modules and micros will include all types. We then have these 3 new size measures $\sum_{core\ mods} mods$, $\sum_{core\ mods} mics$, $\sum_{core\ mods} bytes$, Figures 19-21.

The last two measures are:-

$\sum_{all\ mics} mics$, i.e. the sum of all the micros where each distinct micro is counted only once, Figure 22, and

$\sum_{all\ mics} bytes$, i.e. the sum of all the micro sizes in bytes, Figure 23.

The similar shapes of the plots seem to indicate that the size measure which one selects is immaterial as long as it reflects the gross pattern of growth for it is the latter which matters, not the measure used. Other features were observed, too:-

- 1) The increasing upwards trend, denoting continuous growth, whichever way we choose to measure it.
- 2) In each plot, 3 ripples or cycles (not to be confused

with the "cyclicity" or oscillatory, an altogether better term, effect) can be observed more or less clearly. They do not appear to occur over equal time periods, although the first cycle started before 7301 while the latest one did not seem to have ended. For each cycle, the rate of growth, fast at first, decreases with time until it is nearing zero at the end.

3) Each cycle's concavity is in the opposite direction to OS/360's [18] and Executive's [36], indicating that if the behaviour is to be explained in terms of cyclically self-regulating feed-back loops, then it is negative, not positive, feedback which is the major contributor to this system's behaviour. It is possible to relate each cycle to a distinct redevelopment phase to boost the capabilities of the system, each phase being characterised by a build-up to a peak followed by a decline which may overlap the next phase's build-up. Indeed, therein lies the explanation of the observed ripples.

4) The plots of Σmicros and Σbytes were different to the others' in one respect: it is possible to observe, at the end of each cycle, a trough which is due to the decrease in the number of micros and hence of bytes. This behaviour is very reminiscent of the tidying-up process during the dying stages of each cycle observed in OS/360 [18].

5) Figure 24 shows that the module's average size in micros or bytes has been decreasing steadily mainly due to the smallness of the newer modules but the average byte size of the modules in terms of the core micros has been slightly increasing, despite a decrease in its average core micro size. The latter increase is due to two factors:-

- a) The newer (and smaller) modules, consisting mainly if not solely of noncore micros, are not considered for this part of the analysis.
- b) Most modifications to a micro usually result in an increase in its byte size.

6) That new modules and new micros account for much more of the total growth than the increasing sizes of the changing micros.

7) The "shareability" of the micros is increasing, Figure 25. The system is tending to evolve through replication of already existing micros.

In this section, we have made several observations similar to those of Lehman, the most striking of which are the ripples and the tidying-up process. Thus, quite a few of his ideas seem vindicated. As predicted by the theory, BD shows continuous growth. This, however, given the nature of the system and most of the growth, which is through the addition of new modules, does not necessarily imply increasing complexity of any

consequence. Perhaps a greater cause for concern is the increasing size of the core micros. If we assume that they get increasingly unstructured after amendment, and, as stated in the second law, that their complexity is increasing, too, then the situation may be worsening although, for such a small system, it may take quite a long time to get out of hand. But, in parallel with the tendency towards increasing complexity, there appear to be compensatory activities which tend to annul it out. These are what Lehman has called anti-regressive activities. Thus, the tendency to write smaller-sized modules, to use high level languages and so on can be regarded as on-going activities to combat increasing complexity or as the net results of or the reactions to such an increase. They may also be interpreted as indicative of a decreasing need to modify the system. We shall again refer to the different ways in which one may interpret most of our results, time and time again. For the present, we shall only state that it is probably too early to decide which of the two interpretations reflects reality more closely. However, it seems without doubt that, even if the previously described activities cannot be taken as examples of on-going anti-regressive work, the end-of-cycle tidying-up process is a good one of periodic anti-regressive activity.

3.4 MICRO ANALYSIS: DISTRIBUTION ANALYSIS

Having looked at the activities data in the previous three sections, we now concentrate upon the data which concern structure. For each micro, we have the following:-

- Micro number.
- Date implemented.
- Number of modules that use the micro.
- Hierarchical level in which the micro is found in its calling module.
- Number of calling micros, i.e. the no. of micros which call that micro.
- Number of called micros.
- Size in terms of record count (i.e. the number of cards, including comment cards).
- Size in terms of bytes.

Two most interesting questions are:-

- 1) How do the properties of the micros evolve over time? (The dynamics of the structure characteristics.)
- 2) Do different subsets of these have properties which are dissimilar at any point in time? (The static structure properties.)

A third one will be a combination of the above two, i.e. how do the properties of the different subsets evolve? Do they get

increasingly different from one another, tend to resemble one another or do they remain the same?

Figure 26 shows that the general 20-80% (or 80-20%) law applies to this system in the same way as it does to other systems (software or otherwise) in the sense that a minority of the micros account for a majority of the modifications (and that a majority of the micros account for a minority of the modifications): 36.5% of the micros have never been modified (according to records available as at 7301) while 20% of them account for 67% of all the changes. This led us to ask whether the micros form a continuum or whether they may be sufficiently differentiated for us to identify subclasses in the population. These may be so different as to require separate investigations. An important first step towards the characterisation of the amendment frequency of a micro by its structure properties is to isolate the micro types by amendment frequency before examining the common features of each group. But it is not easy to obtain well-defined micro clusters. In fact, the number of subsets to explore is virtually infinite and, although it is very tempting to look at several, because of our limited resources, we restricted ourselves to only 2 arbitrary choices:-

- 1) The most frequently changed micros, MFCM, and the unchanged micros, UM.
- 2) The mutually exclusive subsets of withdrawn, "aliasing", "hll" and "normal" micros.

3.4.1 The various sets' attribute distributions at the same point in time

3.4.1.1 Comparing the properties of the (100) MFCM and of the (331) UM, with those of the population.P

From their relative frequencies by each of their six attributes, Figure 27, we observe the predominant negative exponential shape which seems to denote great simplicity in the data. For example, most micros, whether MFCM or UM, are called by 1 or 2 modules only and by no or 1 (or sometimes 2) other micros.

Attempts at distribution fitting (exponential, normal and so on) were not quite successful [44], probably because of the great degree of randomness which causes the irregularity in the distributional shapes but we note that the UM distributions are more regular than those of MFCM, more specially for called micros, byte size and record count and that, as expected, for each variable, the distribution of P is intermediate between those of the others. For the hierarchical level attribute, the shapes reflect the natural method by which nesting in programming is achieved: there is an increasing number of

micros from the first to some lower level, here the third. Afterwards, it is downhill all the way.

MFCM have these characteristics: they are used at only one location in the system, are large, are at the highest hierarchical levels and call many other micros. These are confirmed by correlation analysis, see [38]. The analysis also shows that the most frequently used micros are the least frequently changed. That they are frequently used but not change often could be because they are simple, bugfree, well defined and deterministic while the most frequently changed micros, being at a higher hierarchical level, do not have such clear cut and simple properties and are usually unique. Or the reason may simply be that one is afraid to tamper with frequently used components. From what follows in the next paragraph, one prefers the former deduction.

In the theory of hierarchical systems, the higher level components are said to dominate the lower level ones [170]; their variables influence those of the lower ones but these do not influence those of the higher ones. The variables in the lower levels are then the "causes" of those in the higher levels. Then, as each level influences and imposes constraints on the structure and behaviour of that immediately below it, the deeper-nested levels become more and more deterministic. Levels are thus connected by a unilateral chain of communication, each successively lower level becoming the "slave" of the higher one. The innermost level is that whose behaviour is the most regulated or deterministic while that situated at the top is the one which has the most extensive repertoire and which is the least deterministic. That the higher level components' environments are more open also implies that they are more susceptible to modifications, as verified by our results.

This is exactly how, as most programmers know, the majority of programs are structured: the main body of the program and those routines which it directly uses are the principal means by which the gross functions of most applications are implemented while the more deeply-nested routines deal with the finer and more specialised aspects of the functions. In that respect, the latter routines tend to be smaller in size and not to call other routines. Thus, whenever the program needs to be enhanced to reflect modifications required in the application, it is usually the higher level components which are affected. The deeply-nested ones, unless they are faulty, tend to be modified only very occasionally and can be considered to be more or less stable. Since, for systems whereby the application is implemented not as a single whole but as a collection of smaller entities, there is bound to be certain overlaps in function and similarities among these entities, then there is a tendency to share some of the more specialised functions, e.g.,

the bats and tug micros.

The Kolmogorov-Smirnov 2-sample test [163] was used as a formal procedure to confirm the preceding observations. The results show that the UM values for called micros, byte size and record count are stochastically less than those of MFCM. For hierarchical level, as justified in the preceding paragraphs, it is opposite while there is no difference in the patterns in which the 2 sets are called by modules or by other micros. We, therefore, conclude that, for a given attribute, different subsets of the population under study may have some similar properties but some different properties, too.

3.4.1.2 Comparing the properties of different subsets after partitioning

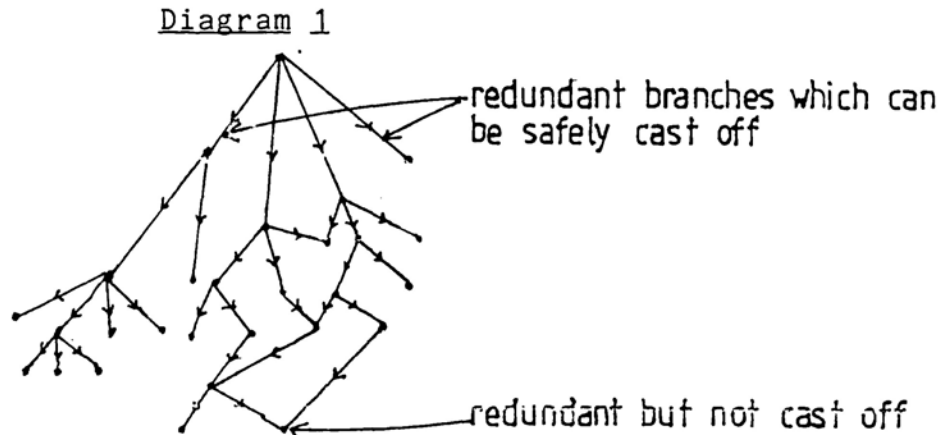
The purpose of this section is to attempt to show whether any differences exist in the properties of the partitioned sets of micros, according to the types defined in [37] (see also Section 3.1.3). Of these sets, only 3 were large enough for detailed analysis: normal, withdrawn and aliasing, Figure 28.

Somewhat successful exponential fits were obtained for each distribution (see [44]), except for byte size and for hierarchical level. We note that calling modules, calling micros and number of changes represent very simple situations, specially for the withdrawn micros, which, in addition, appear to be generally simpler than the other two types by all of the attributes. For each set of micros, the other attributes have less simple distributions although record count, called micros and changes still manage to look exponential. However, byte size and hierarchical level can better be described as skewed normal. The latter's skewedness provides a very interesting insight: while that for the normal micros is very slightly to the right, that of the withdrawn micros is more pronounced (indicating that they tend to be found in even higher levels) and that for the aliasing is to the left, indicating that duplicated micros tend to be found in the lowest hierarchical levels instead.

The casting off process is really related to the risk which one takes while removing redundant code. It is probably not worth, for a few economies, the risk of disrupting the system and tampering with code whose interactions are ill-understood. However, well-understood redundant code can quite easily be chucked off (together with any ancillaries). Inevitably, deeply-nested code is treated with care and probably retained, even if redundant, since one is less certain of one's understanding of their interactions and dependencies with the rest of the code. Redundant code situated at the highest hierarchical levels are easier to understand; one is, therefore, more confident that no problems will be caused upon

removal and readier to take the risk. For analogous reasons and, also, because the probability of them being required elsewhere is small, redundant micros tend to be those which are called by only a few modules or a few other micros.

Diagram 1 summarises the above thus:-



For similar reasons, instead of "sharing" a deeply-nested micro between several modules, one may prefer to duplicate it. The resulting copies may then be given different names and, existing as separate entities, may be modified, independently of one another, without any incompatibility problems arising among the sharing modules.

Because of its greater efficiency, the Kruskal-Wallis one-way analysis of variance test [163] was chosen to test whether the 3 subsets belong to the same population. Non-significant results were obtained for called micros, byte size and record count [44]. Pairwise testing (Kolmogorov-Smirnov test) of the samples confirmed the differences we noted while discussing the plots and fits: that the withdrawn micros have simpler distributions by calling modules, calling micros, hierarchical level and number of changes than the other two subsets.

The test also indicated that aliasing micros, apart from their tendency to appear in the lowest hierarchical levels, behave no differently from the normal micros [44]. This may be because duplication is mostly the taking of a single copy of the original (90.0% had only one duplicate each) and most of it (89.5%) occurred right at the start of the system's life. For most intent and purposes, these micros could be, and were, as shown by our analysis, treated as ordinary ones. The rarity of duplication during the rest of the system's 9 years of operational life is indirectly reassuring since it may otherwise be interpreted as an indicator of increasing

complexity. The increasing shareability results in Section 3.3 reinforces the view that the contingency measure of duplication which was initially provided for to combat anticipated compatibility problems was unnecessary. Within the same context, increasing shareability should be interpreted as indicative of increasing confidence by the maintenance team in its own understanding of the system's functions.

The work in these two subsections has clearly demonstrated the usefulness of studying individual subsets of the component population although it also gives an idea of the tremendous amount of work which would be involved if more than a few subsets were examined. At the same time, it is also obvious that taking a global view is an approach which is conducive to equally meaningful results. The general implication is that there are two ways of studying the properties of a population of software components. The first is to concentrate upon the behaviour of each of the mutually exclusive subsets of components. The number of possible subsets and the degree to which they resemble one another will play a large part upon determining the amount of effort required for this approach. This might require so much work that it would have to be curtailed. It may even be preferable to adopt the other approach, that of considering all the population together (or a sample of it). In this case, each identifiable subset's properties is virtually ignored and smoothed out of the study through the global view. Uncluttered by details, the analysis will be much easier but how applicable the results will be will depend upon the homogeneity of the population. If one is interested in the fine details of the system or in only particular parts of it and/or if the subsets do really show significant differences from one another, then the former approach is indicated. If one wants only a global view and/or if the components resemble each other anyway, then the latter is to be preferred. Since both approaches lead to significant findings, it really depends upon whether one wants to (and can afford to) look at the trees themselves or only at the forest. Although the latter approach is the one which has been favoured in ED studies, the former should not be excluded since it can do much to explain why the forest should be as it is observed. Thus, there is a case to look at both the forest and different varieties of trees.

3.4.2 Distribution evolution

The only variables amenable to this sort of analysis are the modifications (mc, me, mf, c, e and f), since past records for the others are virtually non-existent. The aim here is to detect any observable trend towards more complex distributions, although such trends may not necessarily imply an increasingly complex system.

3.4.2.1 Some complexity measures, suggested and used in previous work

The purpose of this section is to attempt to discuss the relative merits and weaknesses of the measures which have been extracted from the modifications data and used to explore complexity evolution. Following the discussions, possible improvements are proposed but, however far we may go, it must be remembered that the information provided by the modifications data only reflects one aspect of complexity. Thus, even if a comprehensive metric is derivable from this part of our work, it may, at best, be considered only as a partial contribution to the search for an all-encompassing complexity metric, if this can ever be achieved.

Heretofore, Lehman [18] has been using modules handled fraction as a crude measure for complexity, although it is pointed out that modules handling fraction would be a better measure. (The difference is that a module accounts for only 1 possible handle but several handlings.) Similar measures applied to BD showed no trends, Table 6. Several reasons suggest themselves for this sort of behaviour:-

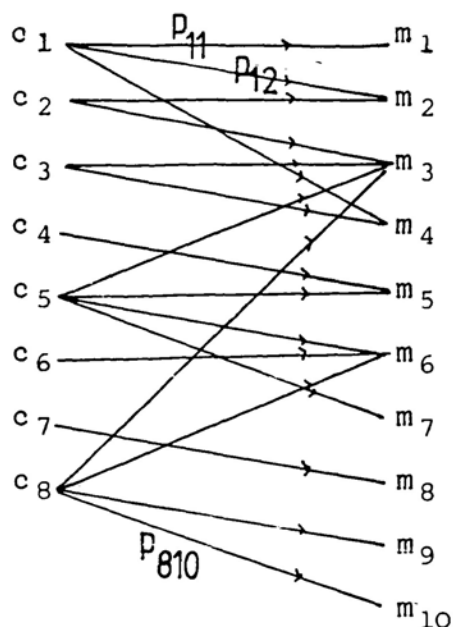
- BD is similarly structured as CCSS and, hence, behaves likewise.
- BD is very much smaller in all of its aspects than the other systems and, hence, may not qualify for the "large program" term and may not exhibit their characteristic syndromes.
- BD, contrary to the other systems, is not release based and, consequently, falls outside a theory of evolution developed largely around the release to release behaviour.

Probably, there is a measure of truth in all of these statements, although which is the more significant can perhaps only be determined through a careful design of experiments.

Other measures which we used to attempt to detect increasing complexity were mc/c, me/e and mf/f, measured in handling terms [43]. Their usage was justified by the belief that, as a system ages, its structure deteriorates with the consequence that any new change will not only require increasing effort but will also affect more and more micros, "the propagation effect". Such a state of affairs was presumed to be reflected by increasing trends in those 3 measures. As shown below, this is not quite true but, as quick and rough measures, they seem adequate. No trends were detected in them, Table 6.

Example 1

Consider the changes situation depicted in Diagram 2.

Diagram 2

where $C = \{c_1, c_2, \dots, c_8\}$ is the set of all changes

and $M = \{m_1, m_2, \dots, m_{10}\}$ is that of all micros affected by C

while $P = \{p_{ij} | \exists i, j \text{ such that } c_i \text{ affects } m_j\}$

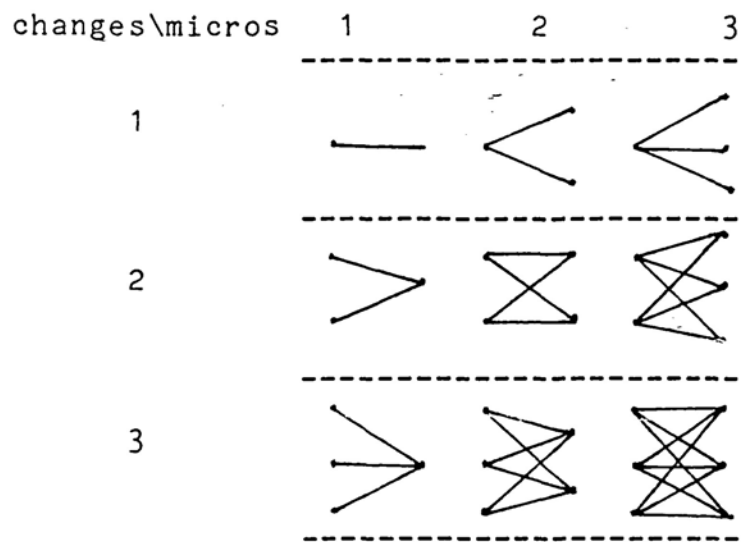
,e.g., c_1 is a change which affects the 3 micros m_1, m_2 and m_4 while c_2 affects m_2 and m_3 while c_4 affects only m_5 , etc.

Now, in terms of handlings, mc/c is 2.25 while in terms of handles it is 1.25. Although these are different, it would matter little if there exists a well defined relationship between handlings and handles. This is what we set out to explore next. Of the several examples given in [44], only two are reproduced to demonstrate that the measures become more inadequate as we depart further away from a one to one mapping.

Example 2 - The most complex cases - when each micro is affected by all the changes.

Consider only the cases of 1 to 3 changes and 1 to 3 micros, thus:-

Diagram 3



leading to the following table:-

Diagram 4

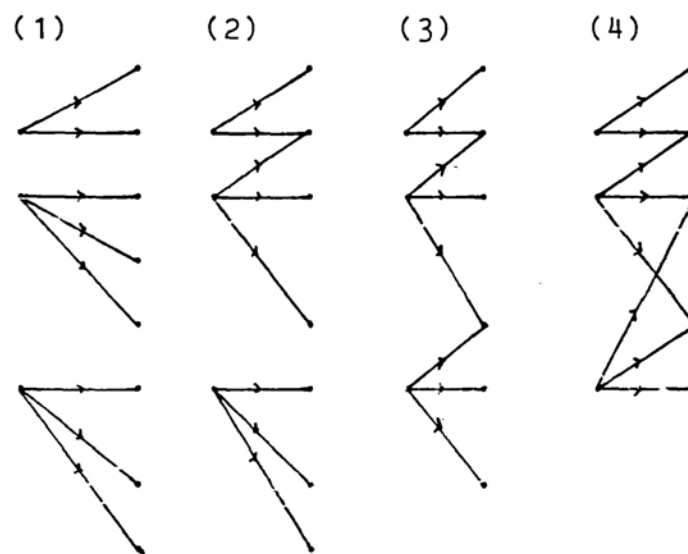
		Handlings		Handles	
c	m	mc	mc/c	mc	mc/c
1	1	1	1	1	1
	2	2	2	2	2
	3	3	3	3	3
2	1	2	1	1	.5
	2	4	2	2	1
	3	6	3	3	1.5
3	1	3	1	1	.33
	2	6	2	2	.67
	3	9	3	3	1

It is clear that (1,1) is the simplest and (3,3) the most complex of the 9 cases shown in Diagram 3, yet handles give $mc/c=1$ for both, Diagram 4. It is also clear that, on moving downwards along any column in Diagram 3, increasingly complex situations are encountered, yet handlings give constant mc/c while handles give decreasing values, Diagram 4!

The defects seem to be that each measure only uses 2 of the 3 elements involved: the number of changes, the number of micros affected and the mappings from the former to the latter. Required is some measure which uses all 3 elements and which is capable of unequivocally ordering the situations which may arise. Example 3 shows that both measures cannot do this.

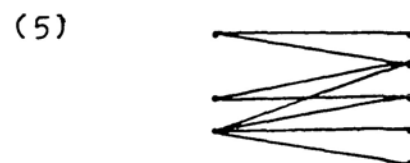
Example 3

Diagram 5



Handlings mc/c	8/3	8/3	8/3	8/3
Handles mc/c	8/3	7/3	6/3	5/3

Clearly, the constant handlings complexity measure (and - worse - the decreasing handles measure) conflicts with the intuitively clear increasing complexity from (1) to (4). However consider



which gives a handlings of 8/3 but a handles of 5/3. How is it possible to say whether (5) can be equally ranked with (4) or not? It thus seems that a fourth element, that of the nature of the mappings themselves, must also be taken into consideration. A better way of conveying the information represented by the

diagrams is to employ two complementary frequency tables, e.g., all 4 elements concerning (3), (4) and (5) can be summarised thus:-

Diagram 6

micros affected\changes	(3)	(4)	(5)
1	0	0	0
2	1	1	2
3	2	2	0
4	0	0	1
Total changes	3	3	3

Diagram 7

changes\micros	(3)	(4)	(5)
1	4	2	3
2	2	3	1
3	0	0	1
Total micros changed	6	5	5

Diagram 6 shows that in (5), 2 changes affect 2 micros each and one change affects 4 micros. The number of mappings, 8, is obtained from the sum of the products 2×2 and 4×1 . Diagram 7, the inverse of Diagram 6, shows that in (5), 3 micros were changed only once, one was changed twice and another thrice. Again, the sum of the product $(1 \times 3 + 2 \times 1 + 3 \times 1)$ gives the value 8.

We have not been able to extract a totally satisfying metric from these tables. This seems a hard problem and nicely illustrates the problems which one may encounter, even in limited situations, while trying to discover an all-encompassing complexity metric. Despite the interesting nature of the problem, this was not the object of our research and, having to move on, we did not pursue this part of the investigation any further.

3.4.2.2 Using a new method to test for increasing complexity

Instead of attempting to derive a single metric to represent complexity evolution, we preferred to deal with the frequency distributions themselves. By 1) constructing frequency tables similar to Diagrams 6 and 7 at regular time intervals and 2) testing for a shift over time towards increasing f_i 's for large

x_i 's in both cases we can test for and claim that the trend leads towards increasing complexity, if there is other supportive evidence.

However, any progress along this line of investigation was dependent upon the existence of a significance test which, while rejecting the null hypothesis that the samples are all drawn from the same population, at the same time lends support to the specified alternative that the situation is getting more complex. The customary parametric one-way analysis of variance test as well as all the non-parametric tests discussed by Siegel [163] fail to satisfy this demand. Fortunately, we were able to discover a relatively unknown test, Jonckheere's test [82], which seems to satisfy our purposes. Instead of simply rejecting the null hypothesis that the samples belong to the same population, it is a test against ordered alternatives, i.e. it is specifically designed to test that the samples occur in a specific order.

It was, therefore, possible to proceed further and the necessary tables were constructed for c, e and f separately, using a cumulation period of 6 months. The choice of this rather large period is justified by the resulting 10 distributions for each variable, a more than adequate figure for blanket distributional tests.

There was only one significant result [44]: that there is a trend towards enhancements involving fewer micros. (This then explains why we did not have a good normal fit, Section 3.2.4.) In each of the five other cases, it seems likely that the samples have come from the same population. That Jonckheere's test indicates no increasing complexity confirms the results from previous work and shows that mc/c, etc., although not too appropriate, nevertheless still provide meaningful results, probably because most of the mappings are 1 to 1 (see note 1) below).

Further examination of the tables [44] led to other noteworthy points:-

- 1) More than 80% of the errors can be corrected by handling only one micro per error; 95 to 100% of them require at most 2 micros to be fixed while in no case were more than 3 micros handled. These low values confirm the results which Endres [59] obtained with DOS/VS data. However, an enhancement may involve up to 27 micros.
- 2) Depending upon the period considered, 71 to 91% of all the micros were fixed only once each, there being only 1 micro which was ever fixed 4 times in any 6 month period. From 77 to 100% of all the micros enhanced needed at most 2 enhancements. Apart from the first period, which was somewhat exceptional, no micro was ever enhanced more than

6 times during any period.

3) There is evidence of a trend, among the more complex enhancements, towards attempts to involve an increasing number of micros. However, this does not mean that, as a whole, the enhancements are getting more complex. The frequency distribution and Jonckheere's test have already revealed another trend: the number of enhancements that involve a few (1 or 2, say) micros is increasing. Then, while more and more complex enhancements are being attempted, the proportion of simpler enhancements is increasing, too, giving an overall me/e ratio which is unchanging.

It thus seems that, as the system ages, despite an overall impression of simplicity, the enhancements become increasingly divided into 2 different classes. There are those which involve fewer and fewer micros and those which involve more and more of them. The former, of the tinkering type, indicates that the final retouches to remove the actual defects in the system are getting simpler and simpler. This could be interpreted as showing that the "perfect" state is being reached, or more precisely, since there may be no perfect state, that the final adjustments are increasingly of the minor type. The other class of enhancements seems to indicate that the maintenance team is gaining in experience and confidence, enough to attempt bolder and bolder enhancements which involve the handling of increasing amounts of code. The opposing interpretation is that, because of increasing communication among the components, system unstructuredness and complexity, tinkering, unless involving very simple tasks, tends to be increasingly avoided while the enhancements increasingly tend to involve more of the components.

3.4.3 Correlation and multiple regression

The remaining work concerning the attributes treats all the micros for all the periods together. One reason is that the modifications are too few for the different subsets to be explored individually. Others, such as the exponential explosion in the analysis effort and so on, are discussed in Chapter 7.

3.4.3.1 Correlating the attributes

An attempt to correlate the average changes/month for each micro with the variables - calling modules, hierarchical level, calling micros, called micros, record count and byte size - gave very low coefficients which were nevertheless significant [38]. It seems that more changes are made to a micro if it is called by fewer modules and micros, if it calls more micros and/or if it has a larger size (record count or byte), but, of

course, these properties, as shown in Table 13, are not all interdependent. The most significant correlations, in order of magnitude, are between the following pairs:-

- 1) Number of calling modules and number of calling micros.
- 2) Record count and byte size.
- 3) Number of called micros and byte size.
- 4) Number of calling modules and number of called micros, (negative correlation).
- 5) Number of calling micros and number of called micros, (negative correlation).
- 6) Hierarchical level and number of called micros.

1) and 2) are obviously explained, 3) indicates that larger micros tend to call other micros more often, 4) and 5) show that the simplest micros (i.e. those that called none) tend to be used more often in other micros and modules, see also Section 3.4.1.1, while 6) is as expected.

3.4.3.2 Linear regression models

We needed only to construct models for changes/month since e is highly correlated with c while f is too small. There being too much data for the whole population of 883 micros to be considered, we had to take a sample whose maximum size for computer memory manipulation (64K) was shown, by a simple calculation, to be about 112 micros. Such a large sample can safely be assumed to be fairly representative of the population. However, another sample (of 100) was used as a check.

With the sample of 112, non significant F and t values are obtained when all the variables are considered. The very small R^2 of 7.6% indicated a very unsatisfactory model. The best stepwise model [56] has an R^2 of 5.7% and is as follows:-

changes/month	=	41.26	+	0.059	record count	-	3.99	calling modules
s.e				0.030				2.63
t				2.01				-1.52
significant at				4%				12%
R^2				5.7%				
F				3.32				

Analogous results were obtained with the smaller sample, whose best stepwise model is

changes/month	=	44.08	+	0.054	record count	-	3.91	calling modules
s.e				0.031				2.75
t				1.72				-1.42
significant at				7%				15%
R^2				5.1%				
F				2.62				

3.4.3.3 Non-linear multiple regression models

Because no new independent variables were available for further exploration, the only way to attempt to improve the models is to transform the data. From the plots of the dependent variable against each of the independent ones, it was quite evident that not much can be made of them and no idea of the required transformation could be gained. We, therefore, proceeded by attempting several of the more usual transformations before recreating the models. [40] gave the details of the transformations and the results obtained; none were very good, the log-log model being, by far, the best:-

$$\log(\text{changes}/\text{mth}) = -12.85 + 4.85 \log(\text{record count}) - 2.79 \log(\text{calling modules})$$

se	1.11	1.88
t	4.36	-1.48
significant at	0.1%	15%
R ²	17.29%	
F	11.39	

, i.e.

$$\text{changes}/\text{month} = 1.42 \times 10^{-13} (\text{record count})^{4.85} (\text{calling modules})^{-2.79}$$

which implies that the contribution of each independent variable in explaining the behaviour of the dependent one and its effect upon the latter is, after being raised to a certain power, multiplicative rather than additive. The other sample gave the very similar model:-

$$\text{changes}/\text{month} = 3.83 \times 10^{-13} (\text{record count})^{4.67} (\text{calling modules})^{-3.40}$$

, R² being 17.86%.

Note that the independent variables in the parsimonious model in this section are not only the same as those in Section 3.4.3.2 but also conform to the correlation results, Section 3.4.3.1.

The model can be interpreted thus: all things being equal, doubling the micro's size causes its frequency of changes to increase by 26 while doubling the number of modules calling it decreases its number of changes by 8.6. It is, therefore, advisable to keep the size of the micros small but whether it is the large number of modules calling a micro which causes it to be changed infrequently (because of the difficulties involved in keeping the changed micro compatible with each of its calling modules) or whether it is that simple, well-defined and, hence infrequently, changed micros which tend to be used over and over again in several modules is not known.

Note, however, that ours are only empirical models, which, arbitrarily created, should make no strong claim to explain why changes/month responds to variations in the independent variables. That even the best model is still grossly inadequate indicates that other factors have much more influence upon the independent variable. This leads us to the belief that, although the frequency of changes to a micro is certainly influenced by its structural properties, there are other, external, undiscerned and not easily measurable factors which may be the main causes of the changes made to a micro. Until they are properly identified and modelled, there is no alternative but to describe the micro changes process as being stochastic. (The recurrent theme of randomness and its explanation as well as the contributions of structural and external influences in governing the behaviour of the system will be amply discussed in Chapters 5 and 8.)

3.5 MODULE ANALYSIS

3.5.1 Analysing the attributes distributions at the same point in time

Work similar to that described in Section 3.4.1 for the micros was carried out for the modules, too, but the only directly available variables for these relate to their size - in micros or in bytes. We have used all the modules, AM, because there are only 131 of them. The two other subsets which were also analysed are the 60 modules which have been amended, C, and the 71 which have not, U, in order to find out whether they differ significantly by size. U includes such freaks as the tugs, utility and hll modules [37] (see also Section 3.1.3).

The analysis shows that the sizes, in both bytes and micros, of the U modules are very significantly smaller than those of the C ones, [44]. It could be that the latter were initially small but that, through modifications, they have increased significantly in size or that larger modules have greater probabilities of being changed. The truth probably lies somewhere in between but, through lack of data, it has not been possible to find evidence to support either hypothesis.

3.5.2 Analysing the modifications data with respect to the modules

(A much more detailed account of the work in this section can be found in [43].)

The purpose of this section is similar to that of Section 3.2 where the series related to the global activities on the system. Here, however, we wish to explore the behaviour of the modules. The attributes selected for analysis are 1)-20) below

but the initial problem was to find an appropriate aggregation period. One month is probably too fine: 59 monthly periods being excessive for the analysis which we propose to carry out. Choosing 1-yearly periods will provide us with a measly 5 points which are too few for meaningful statistical inferences. A suitable compromise seems to be the 6-monthly period; henceforth simply referred to as the period or Δt .

3.5.2.1 Analysis of all the changed modules and the "old" modules

From the raw data, we then compiled, for each module, its number of micros and its modifications data per each time period. From these, we extracted 20 parameters to summarise the behaviour of the (modified part of the) system:-

1) Number of micros,	m.
2) Number of micros changed,	mc.
3) Number of micros enhanced,	me.
4) Number of micros fixed,	mf.
5) Number of changes,	c.
6) Number of enhancements,	e.
7) Number of fixes,	f.
8) Number of micros changed per change,	mc/c.
9) Number of micros enhanced per enhancement,	me/e.
10) Number of micros fixed per fix,	mf/f.
11) Number of modules which have been changed,	M.
12) Number of changes per module,	c/M.
13) Number of enhancements per module,	e/M.
14) Number of fixes per module,	f/M.
15) Number of modules changed,	Mc.
16) Number of modules enhanced,	Me.
17) Number of modules fixed,	Mf.
18) Number of changes per module changed,	c/Mc.
19) Number of enhancements per module enhanced,	e/Me.
20) Number of fixes per module fixed,	f/Mf.

Note 15) $\neq$ 16) + 17) but 15) = 16) v 17).

Correlation provided no surprising results while polynomial fitting gave the mean as the best fit in each case except for mf, f, f/M and f/Mf - all involving fixes - whose best fits are straight lines with negative slopes. The R^2 values for the latter, as always, are rather low, indicating the presence of marked noise. Figure 29 exhibits the time plots whose variations, on the whole, seem unlikely to be systematic.

Because new modules seem to be amended more frequently, specially early in their existence, than existing ones, those newer modules whose date of birth could be retraced were weeded out and the analysis repeated. This was in an attempt to determine to what extent the injection of new modules

influences the whole system's behavioural patterns: very little, since almost similar results were obtained, Figure 30. The analysis concerning the new modules themselves shall not be referred to until Section 3.5.2.2.

3.5.2.1.1 Analysis within each time period

The emphasis, here, is not on the evolutionary behaviour of the variables but rather on whether there exist any instantaneous relationships among them and on how these are affected by factors like module size. It may be, for example, that although c and mc may both be stochastic time series yet, at any point in time, one is strongly related to the other, whatever module is being changed.

The analysis is performed on 3 sets of modules: all of them, AM, only the "old" ones, OM (those predating 7301) and only the "new" ones, NM (those postdating 7301), because we wished to compare the behaviour of NM with that of OM and to observe how much the addition of NM disturbs the settled pattern of behaviour, if any, of AM.

For the subsequent analysis, the only useful variables are mc , me , mf , c , e , f and m ; mc/c , me/e and mf/f shall be derived from the results.

Correlation shows that, whether we are dealing with AM or OM, for each time period, mc or me or mf is most strongly correlated with c or e or f first and then secondly with m . This most convenient result is as expected and allows us to proceed to the next step of the analysis, which is performed for each of the 10 periods for both AM and OM. As the method is the same for all of them, it needs be described for only one time period.

Using c , e and f in turn as the independent variable, their best regression models are found. They are invariably of the following nice forms:-

$$\begin{aligned} c &= \alpha + \beta mc + \gamma m \\ e &= \alpha + \beta me + \gamma m \\ f &= \alpha + \beta mf + \gamma m \end{aligned}$$

and gave R^2 values upwards of 96%. Such high values imply that no transformation of the data is required.

In addition, the following linear regression fits were also found:-

$$\begin{aligned} c &= \alpha + \beta m \\ c &= \alpha + \beta mc \\ e &= \alpha + \beta m \end{aligned}$$

$$\begin{aligned} e &= \alpha + \beta me \\ f &= \alpha + \beta m \\ \hat{f} &= \alpha + \beta \hat{mf}. \end{aligned}$$

The results show that, in general, AM gives slightly worse fits than OM (whose R^2 are, in general, better by about 2%). This is interpreted as showing that NM does indeed disturb the settled pattern of AM, although minimally. Linked with the time based results where the R^2 have much lower values, one concludes that, although the variables may all appear to vary stochastically over time, yet their relationships are strong and preserved from one period of time to the next.

Interpretation of $f = \alpha + \beta mf + \gamma m$

As c , e and f can all be interpreted similarly, the discussions will be confined to fits for f only. We first noted that the coefficients of both mf and m were positive, indicating that as either mf or m increases or both, so does f .

Interpretation of $f = \alpha + \beta mf$

The value of β gives the average f/mf for the modules considered over the given period. Hence $1/\beta$ gives the more meaningful mf/f (which compares well with the mf/f 's obtained by other means).

Interpretations of $f = \alpha + \beta m$

The average f/m is given by β but the expression also gives the size of the smallest module which contains errors thus:-

$$f \geq 0$$

$$\text{if } \alpha + \beta m \geq 0$$

$$\text{i.e. if } m \geq -\alpha/\beta.$$

In other words, the maximum size of the error free module, henceforth denoted by Max_f , is given by $-\alpha/\beta$.

3.5.2.1.2 Evolution of mf/f , f/m and Max_f over time

From the various fits, 10 possible values (one per period) are obtainable for each of mf/f , f/m and Max_f , whose behaviour over time may then be investigated. The same applies to the c and e variables, too.

Table 14 and Figure 31 show that AM and OM behave very similarly and that:-

- 1) mc/c , me/e and mf/f , as usual, appear to be stochastic

series.

2) f/m is decreasing over time while c/m and e/m remain constant. This could indicate that errors are being gradually smoked out of the system while the average number of enhancements per micro has not really changed over time, and could be one of the first non-dually interpretable results which could tip the main conclusion in favour of decreasing, instead of increasing, difficulty and complexity.

3) Smaller and smaller modules are being enhanced and changed while the size of error free modules has remained constant. The latter finding is indirectly reassuring as enhancing smaller and smaller modules has not implied that these are tending to get more frequently fixed as well. Again, there is cause to favour the increasing simplicity argument.

As the modules seem to tend to become fix free and, probably, bug free over time, although this may not be necessarily so, one may conclude that the system is getting better and better. It is not known (and not possible to find out from our data) what factors are at work. These could be, among other things, that the system is small, that it is stable, that its environment is well defined and understood, that the maintenance team is small and experienced, that the personnel turnover is low or maybe that the architectural characteristics of a collection, BD being one rather than a system, is the overriding factor.

3.5.2.2 Analysis of the new modules

There were 14 NM, half of them having been in existence over 6 (6-monthly) periods, 4 over 5 periods, 2 over 3 and 1 over 8. Since their introductions had been staggered in time, they have been brought to the same birth date and "normalised" to the same lifetime, too. The normal life has been taken as 6 periods, the period spanned by the majority of NM. While scaling, the total number of mc , me , mf , c , e and f for each module is maintained and the relationships between each period's elements are retained. Some controversy was caused by our methods and by the choice of 6 months as the period, too, but these seem to matter little, as repeating our work with simpler methods, e.g., as used in Section 4.3.4, than normalising and shorter periods led to substantially the same results [159].

3.5.2.2.1 Repeat of the analysis of the evolution of the global change parameters over time

The work described under Section 3.5.2.1 has also been carried out for NM. From the plots in Figure 32, one cannot fail to notice the exponentially decreasing shape of the curves. This

is to be contrasted with the plots in Figures 29 and 30. The parameters extracted, as in Section 3.5.2.1.2, from the regression fits to each of the 6 time periods are plotted in Figure 33. While it is difficult to interpret the other plots yet mc/c , me/e and mf/f , once again, show unmistakable exponential decrease over time.

3.5.2.2.2 Exploring the best fits to the global parameters

The general shape exhibited by the 9 sets of data, mc , me , mf , c , e , f , mc/c , me/e and mf/f , obtained under Section 3.5.2.2.1 could be best fitted, among others, by these: $y = \mu$, $y = a + bt$, $yt = c$ and $y = ke^{-\lambda t}$. By comparing the fits, it was obvious that, for the first six variables, the exponential fit was the best while the conceptually more satisfying hyperbolic fit with asymptote $y = a$, $a \neq 0$, was slightly better than the exponential fit for mc/c , me/e and mf/f , Table 15.

Some noteworthy points are:-

- 1) The exponential fit for e is the worst, indicating that it is the variable which is most strongly influenced by external and stochastic factors.
- 2) e is the least rapidly decreasing series, indicating that, although the fix rate may quite soon become vanishingly small, that of making enhancements, although decreasing, would remain substantial.
- 3) That me/e is decreasing as well indicates that initially the enhancements to the new modules could be somewhat more difficult (each involving several micros), but that, as time goes by, either the enhancements are getting simpler or major enhancements tend to be avoided because of increasing complexity. In the context of all our results, notably here and in Section 3.5.2.1.2, one prefers the former explanation.

That the enhancements seem to be getting simpler and that the number of fixes is decreasing rapidly indicate that, contrary to what may be feared, few, if any, new errors are being added to each module. This also indicates that the number of residual errors is decreasing and that the system is probably getting bug free. The alternative view that errors are still being introduced but are not being discovered or fixed seems implausible.

The system, then, seems to be getting better and better: there are fewer and fewer fixes and, while the number of enhancements is not decreasing at such a fast rate, the enhancements are, nevertheless, getting simpler and simpler. Since it looks as if that of increasing simplicity is the more probable of the two ways of interpreting some of our results, it may be a good idea to attempt to bring together all the previous simplicity

conclusions.

As the system is being perfected, the newer modules which are being written tend to be small and of the "bells and whistles" type. They also tend to replicate already existing functions, to be variations on the same theme and to be easier to write since they tend to incorporate already existing micros as "off the shelf" components, Section 3.3. Their enhancements tend to become simpler while those to existing modules tend to involve a larger number of the smaller ones, too, Section 3.5.2.1.2.

Tinkering with already existing code for minor improvements tends to become scarce while the increasingly experienced maintenance team attempts bolder and bolder enhancements of the large scale type, Section 3.4.2.2. There is also a decreasing need to fix a micro, Section 3.5.2.1.2, or a module; these are probably becoming error free.

3.5.2.2.3 Deriving the exponential fit from purely theoretical considerations

The exponential fit, which we have best fitted empirically from the data, is well known and can be obtained by starting with purely theoretical considerations. Let y represent any of our 6 variables. Then, assuming that the rate of change of y is proportional to y , we have

$$dy/dt = -\lambda y$$

$$\Rightarrow y = ke^{-\lambda t}.$$

When $t = 0$, $y = k$.

While k is the initial y value, λ can be interpreted as a constant of proportionality which reflects certain structural aspects of the system. The importance of λ is more fully discussed in Section 3.5.2.7.3.

3.5.2.2.4 Convenience and uses of the exponential fit

Total at time t and at infinity

The sum of y at any time t or during any interval t_1 to t_2 can be found directly from the formula. There are 2 methods: that of treating the y 's as discrete time values or that of treating them continuously.

Discrete treatment

$$\text{Cum } y = \sum_{t=0}^t y = \sum_{t=0}^t ke^{-\lambda t} = k(1 - e^{-\lambda t}) / (1 - e^{-\lambda})$$

Let T be the total number of y when $t = \infty$. Then it is given by

$k/(1-e^{-\lambda})$ and therefore has a finite value.

Continuous treatment

$$\text{Cum } y = \int_0^t k e^{-\lambda t} dt = ((k e^{-\lambda t}) / -\lambda) + c$$

When $t = 0$, $\text{cum } y = 0$

$$0 = -k/\lambda + c$$

$$c = k/\lambda$$

$$\text{cum } y = (-k/\lambda) e^{-\lambda t} + k/\lambda$$

$$\text{cum } y = k(1 - e^{-\lambda t})/\lambda$$

$$\text{and } T = k/\lambda$$

Using the Binomial distribution for forecasting

The expression for $\text{cum } y$ can be interpreted thus: the total number of fixes, say, until t is the product of all possible fixes and the (increasing) probability of a fix being made by that time. Then the probability of a fix being made until t , $p(t) = (1 - e^{-\lambda t})$ and that of a fix not being made by that time is $q(t) = 1 - p(t) = e^{-\lambda t}$. Both $p(t)$ and $q(t)$ are exponentially distributed with $p(0) = 0$, $p(\infty) = 1$, $q(0) = 1$ and $q(\infty) = 0$.

We can now use the Binomial distribution to make forecasts. Thus, the formula

$$P(x) = {}^T C_x p(t)^x q(t)^{T-x}$$

gives us the probability of making x fixes until time t .

Note All that we need to know are the estimates for k and λ as each of the above is expressible in terms of these two only.

The exponential fit may also be used to estimate the time to make a total of x fixes and that for the fix rate to reach the value r . These are given by $(\log_e(T/(T-x)))/\lambda$ and $(\log_e k/r)/\lambda$. Although expressions for $\text{cum } y$ are also derivable from all the other fits, they are not as useful as the exponential. Table 16 compares the predicted cum values with the real ones at $t=6$.

The predicted T values

The various T values obtained from both the discrete and continuous treatments are in Table 17. These show that although the modules are settling down, 3 years after their introduction, yet only about 75% of all the modifications to be affected to them have already been made, the remainder being

likely to stretch over a long period of years, as indicated by the distribution. That f and mf have the highest percentage of work completed while e has the lowest indicates that, while errors tend to be repaired early, the process of making enhancements is a more drawn out affair. Thus, the code becomes fix free before it can be enhancement free. Also, it implies that the rate of introducing errors is lower than that of removing them.

3.5.2.3 Repeating the analysis for the typical new module

The analysis so far described in Section 3.5.2.2 has proceeded at the system level. One may now ask how well the exponential fit is able to explain each module's or each micro's behaviour. Finding the answer to this question is our next task. As it is both impractical to repeat the analysis for each module and necessary to smooth out the fluctuations in each module's behaviour, it was best to proceed by devising a typical module to represent the whole set. Then, only its analysis would be necessary.

3.5.2.3.1 Deriving the data for the typical new module

The typical module's data are derived by first weighting each module's data by its size and then by adding them. This sum is finally divided by the total size of the modules. The method, of course, assumes that the number of changes to a module is, more than anything else, related to its size. From the regression models in Section 3.4.3, there is sufficient evidence to believe that this may be the case. Anyway, size was the only available module attribute, Section 3.5.1.

Let y_{ij} represent mc , me , mf , c , e or f for module i , $i=1$ to 14, in period j , $j=1$ to 6, and let w_{ij} be the size of module i in period j . Then the weighted value x_j , for the period j is given by

$$x_j = \left(\sum_{i=1}^{14} w_{ij} y_{ij} \right) / \left(\sum_{i=1}^{14} w_{ij} \right).$$

The x 's will then be typical of a module of size

$$\left(\sum_{j=1}^6 \left(\sum_{i=1}^{14} w_{ij}^2 \right) / \left(\sum_{i=1}^{14} w_{ij} \right) \right) / 6$$

, averaged over the 6 periods. This is found to be 20.4 micros.

3.5.2.3.2 Results and remarks from Figure 34

- 1) The results, in every single case, confirm those of the whole system.
- 2) There is one interesting feature in the curves which has not been previously noticed. This is that, for the second period only, the mf , f and mf/f values are above expecta-

tions while those of the other variables are below. The interpretation is as follows. It is well known that, on field release, all the flaws, which the system implementors and internal testers may not have noticed, are suddenly exposed. These, after taking some time to filter through from the users, will cause somewhat more fixes than should have been usual. To cope with that increase, probably other, lower priority work, will have to be left aside for the moment, as, for example, indicated by the lower than expected e value for the same period.

3.5.2.3.3 Using the typical module's results to predict other modules' behaviour

The best fit to the typical new module may be used to predict the behaviour of any module, provided that one can assume that all the modules behave similarly and that their different modifications values are mostly dependent upon size. Then, adjusting $y = ke^{-\lambda t}$ for the typical module, we have $y = (mke^{-\lambda t})/20.4$ for the module of size m . This, of course, is simplistic. Probably, the methods of both creating an ideal module and of parametrising the model of its behaviour so that the latter can be adapted to represent several types of modules can be considerably honed. However, we had our own quest so that although, en passant, several interesting topics and problems such as this (see also Section 3.4.2) were encountered, yet it had not been possible to spend sufficient time on them for complete solutions; they were either circumvented or solved only in a very broad sense.

3.5.2.4 Repeating the analysis for the typical new micro

The method adopted to derive the typical new micro's data is:-

- 1) To obtain, for each module i , the average modification/micro in each period j , thus

$$x_{ij} = y_{ij} / w_{ij}.$$

In other words, each item of a module's data is divided by the module's size.

- 2) Then, to smooth out the fluctuations in each x_{ij} , each period's results are averaged, i.e.

$$x_j = \left(\sum_{i=1}^n x_{ij} \right) / n$$

where $n = 14$ for $j=1,2,\dots,6$

is found.

The results can then be assumed to be typical of a micro representing all the micros in the 14 modules, giving them equal weighting, irrespective of the module's size.

Exactly the same results and remarks as for the typical module apply, Figure 35.

3.5.2.5 Comparing the 3 methods

As the data were differently treated in each of the 3 methods, the numerical results, as expected, did not turn out to be quite the same. The important thing is that each method brought out the fundamental fact that modifications to the system sustain an exponential fall over time. All 3 methods have their merits and particular circumstances will determine which one to choose. However, the following remarks seem pertinent:

1) The system

Although the most straightforward method, it may be the least useful since the results do not seem to be generalisable to other systems which may be of different size, architecture, environment and so on. But, if one needs not forecast the future behaviour of any system other than that being studied, then this method seems best.

2) The module and the micro

These are more generalisable since the context of each module/micro is more similar to that of another module/micro than that of a system to that of another.

3.5.2.6 The work of Shooman

In [43], we briefly discussed several works which, in some sense or the other, treated the same general subject of modelling the number of modifications as time functions, whether empirically or theoretically. As an example of the latter, we may take Shooman's construction of a simple residual error model on the way to build up a complex reliability model [162]. It was derived thus:-

Let $\rho(\tau)$ be the error detection rate at time= τ , $\rho(\tau)$ being measured in errors/instruction/month

$\epsilon_c(\tau)$ the cumulative error function

E_T the total errors in the system

and I_T the total number of instructions.

$$\text{Then } \epsilon_c(\tau) = \int_0^{\tau} \rho(t) dt \quad \text{-----(1)}$$

$$\rho(\tau) = d\epsilon_c(\tau)/d\tau \quad \text{-----(2)}$$

and the residual errors/instruction, $\epsilon_R(\tau)$ at $t=\tau$ is then given by

total errors/instruction - corrected errors/instruction at $t=\tau$

$$\text{, i.e. } \epsilon_R(\tau) = (E_T/I_T) - \epsilon_C(\tau) \quad \text{-----}(3)$$

Shooman, for simplicity, assumed that the error rate is constant over time. In a subsequent paper [160], he says that:-

"Obviously, examination of the figures show a generally decreasing trend in error rate against debug rate...In addition, it seems reasonable to postulate that the number of errors debugged per month should decrease as the number of errors present decreases. If we assume that the number of errors detected per month is proportional to the number present, we obtain an exponential error detection rate."

$$\text{, i.e. } \rho(\tau) = k_1 \epsilon_R(\tau) \quad \text{where } k_1 \text{ is the constant of proportionality}$$

$$\text{, i.e. } \rho(\tau) = k_1 (E_T/I_T - \epsilon_C(\tau)) \quad \text{from (3)}$$

$$\text{, i.e. } d\epsilon_C(\tau)/dt = k_1 (E_T/I_T - \epsilon_C(\tau)) \quad \text{from (2)}$$

$$\text{, i.e. } d\epsilon_C(\tau)/dt + k_1 \epsilon_C(\tau) = k_1 E_T/I_T$$

$$\text{, i.e. } \epsilon_C(\tau) = E_T/I_T + A e^{-k_1 \tau}$$

Now, when $\tau = 0$, $\epsilon_C(0) = 0$

$$A = -E_T/I_T$$

$$\text{then } \epsilon_C(\tau) = (1 - e^{-k_1 \tau}) E_T/I_T$$

which is analogous to $\text{cum } y = T(1 - e^{-\lambda t})$.

The simplifying assumptions:-

- 1) All errors in the system are detectable.
- 2) The total number of instructions is fixed and finite.
- 3) The functional corrections do not introduce new errors.

which Shooman makes, of course, do not quite hold. But this is not the place to discuss reliability theory. The point which we really want to make is that, for simple and gross models, the empirical approach is likely to yield models similar to those obtained from purely theoretical considerations.

3.5.2.7 Exploring the usefulness and adaptability of the model

3.5.2.7.1 A more general model: providing for the introduction

of new errors

That the assumptions cannot hold is amply demonstrated by the deviations from the ideal which the empirical data exhibited. We shall now give one example to demonstrate how the applicability of the model can be extended. This concerns, contrary to Shooman's third assumption, allowing for the introduction of new errors.

Let us represent the rate of error introduction by the general function, $f(t)$.

Then $dy/dt = f(t) - \lambda y(t)$

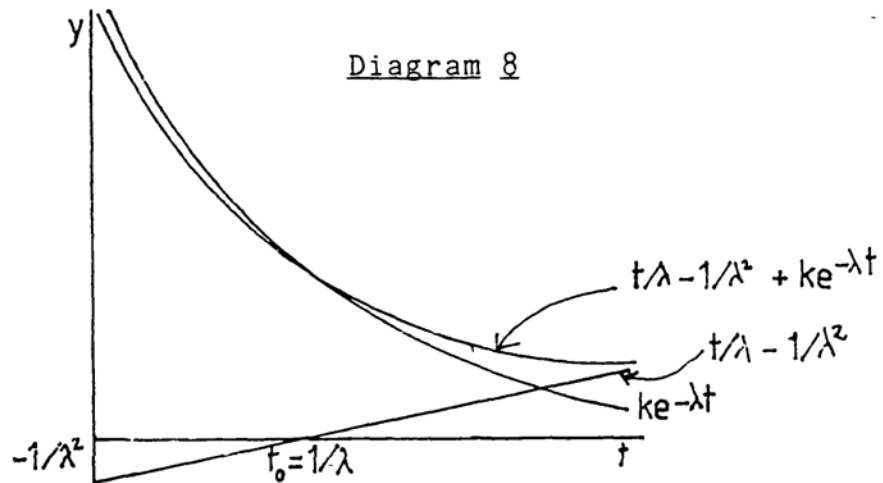
$$y = e^{-\lambda t} \left(\int f(t) e^{\lambda t} dt + k \right)$$

If $f(t) = t$, then $y = e^{-\lambda t} \left(\int t e^{\lambda t} dt + k \right)$

$$\text{i.e. } y = t/\lambda - 1/\lambda^2 + k e^{-\lambda t}$$

$$\text{and cum } y = t(t/2 - 1)/\lambda + k(1 - e^{-\lambda t})/\lambda$$

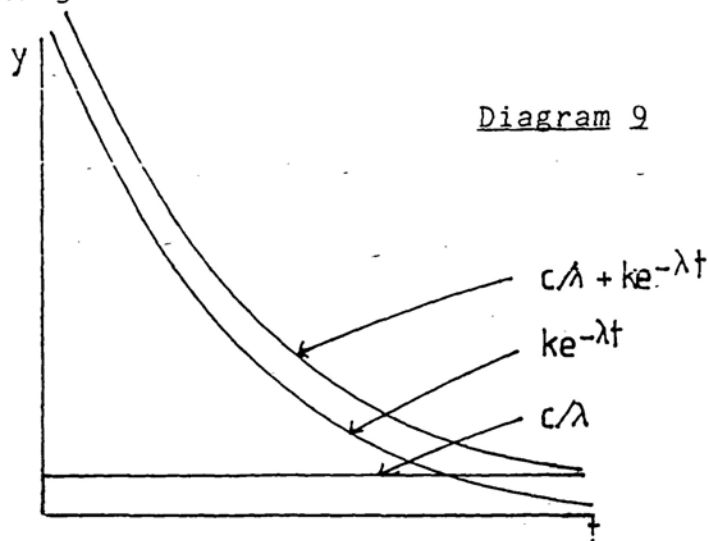
The shape of y is as shown in Diagram 8 below.



Thus, until $t_0 = 1/\lambda$, there are less y 's than in the case of the $y = k e^{-\lambda t}$ model but afterwards not only are there more y 's than in the simple case but as from the point of inflection, it is monotonically increasing. It is then not true to say that the situation is improved until t_0 .

If $f(t) = c$

Then $y = e^{-\lambda t} \left(\int c e^{\lambda t} dt + k \right)$, i.e. $y = c/\lambda + k e^{-\lambda t}$ whose shape is:-

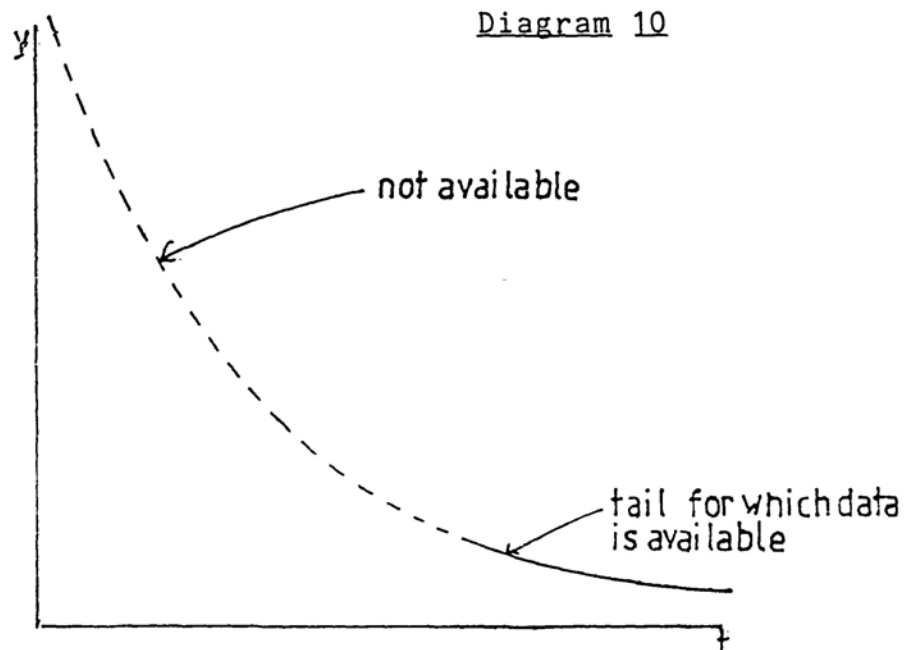


Then, when $t \rightarrow \infty$, $y \rightarrow c/\lambda$.

As the curves for NM show no upward trends (as yet), one may assume that the contribution of $f(t)$ is as yet (and maybe forever) quite negligible for them.

3.5.2.7.2 Using the model to explain the old modules' behaviour

It is, by now, quite natural to believe that OM also behaves according to our models. Although it was not possible to devise a typical old module to prove it, yet, assuming that the OM do behave similarly as the NM, then the decreasing trend in the global number of fixes to OM, Section 3.5.2.1, or in that to the whole system, Section 3.2.2, can quite easily be explained. As the old modules were mostly at least 4 years old when the modifications data were collected, it is almost certain that they had already settled down. (NM takes about 2 years to do this.) If we assume a similar behaviour as NM, we, then, through lack of data, have missed the initially spectacular steep decrease in their curves and are left with only the tail part where the fix rate has slowed down considerably. This is shown diagrammatically below.



Then, when all such decreasing tails of the OM are added together, the net effect is likely to be a decreasing trend, specially as quite a few of the OM were implemented simultaneously, either right at the start or at other times. Although the older components are settling down and need hardly to be fixed, that the number of fixes to the whole system has not become vanishingly small but is only decreasing is due to the continuous addition of new components into the system.

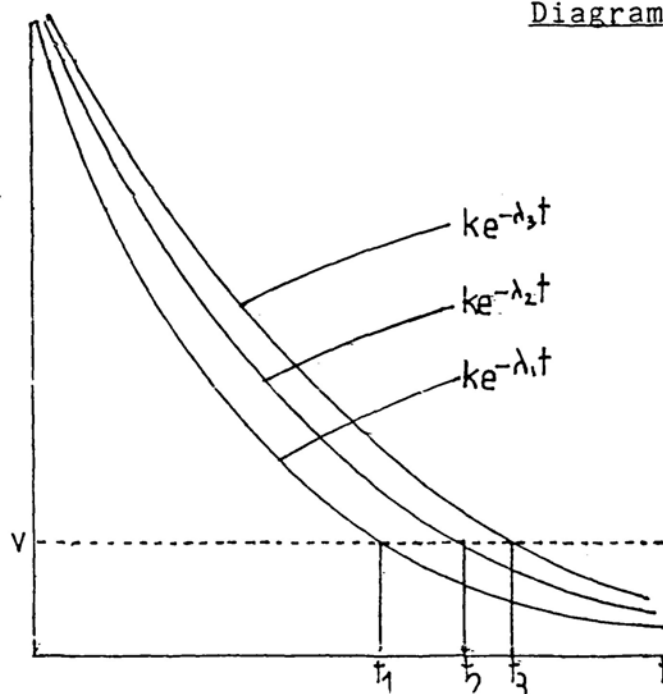
The global number of enhancements to the system is, however, increasing, Section 3.2.2, despite the fact that each new component's enhancements per unit time decreases in the same fashion as the number of fixes. This is due to a conjunction of factors:-

- 1) The rate of decrease in the number of enhancements for each micro is much less than that of fixes.
- 2) There is a tendency for reassessing and rehauling the already settled modules. Increasingly smaller modules are being selected for enhancements and more and more ambitious enhancements are being performed, too. It is then possible that, for the enhancement model, we shall have to provide for a reverse from a decreasing to an increasing trend somewhere in the tail.
- 3) The continuous addition of new modules.

3.5.2.7.3 Proposing a quality measure

The constant, λ , is a factor whose magnitude can be interpreted as representing the quality of the program. For the model, $y = ke^{-\lambda t}$, for constant k , the larger λ is made, the smaller will be the area under the curve. Also, the time t_i , to reach any given value, v (e.g., the maximum user tolerance error rate) is attained faster by the larger λ_i . Thus a large λ could represent a high quality module while a small λ would represent a poor quality one. Note that T also decreases as λ increases.

Diagram 11



Given k and v ,

$$\lambda_1 > \lambda_2 > \lambda_3 \Rightarrow t_1 < t_2 < t_3 \Rightarrow T_1 < T_2 < T_3$$

For the extended models, the contribution of errors introduced through repairs also depends upon λ , e.g., in

$$y = t/\lambda - 1/\lambda^2 + ke^{-\lambda t}$$

, the term $(t/\lambda - 1/\lambda^2)$ has a smaller slope, and hence contributes less to y , with larger λ . At the same time, t_0 , the time when $(t - 1/\lambda)/\lambda$ crosses the t -axis becomes smaller, so that postponing the y 's until later is minimised. Although T remains infinite, cum y is also reduced by increased λ .

In $y = c/\lambda + ke^{-\lambda t}$, when λ increases, not only does the area under $ke^{-\lambda t}$ decrease but c/λ decreases as well. The ultimate value of y , given by c/λ , is also reduced by increasing λ .

3.5.2.7.4 Refinements - when the quality of the module is a function of time

One can now suppose that the quality, λ , of the module is not fixed over time, and can explore various time functions for it. We would then enter the realm of differential equation.

To try and explain the fluctuations, instead of assuming them to be noise, one may let

$$\lambda(t) = A + B\sin wt$$

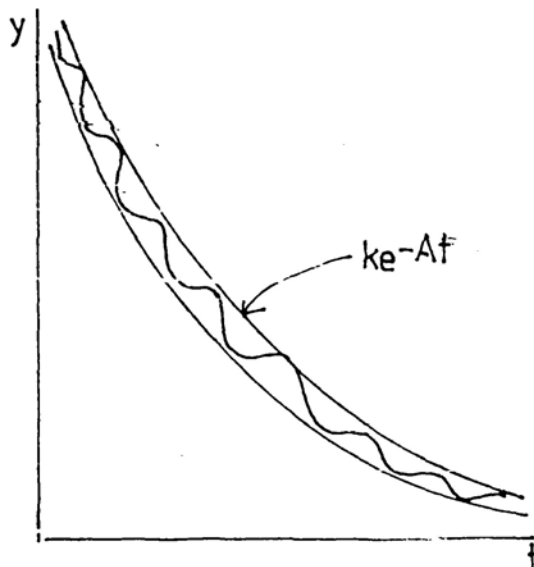
, then, when $f(t) = 0$

$$dy/dt = -(A + B\sin wt)y$$

$$y = ke^{-At - 2B\sin^2(wt/2)/w}$$

behaving as shown in Diagram 12.

Diagram 12



where ke^{-At} is an upper bound and the factor

$$e^{-2B\sin^2(wt/2)/w}$$

oscillates between $e^{(-2B/w)}$ and 1.

There are many other possible refinements, but as these are not along the mainstream of our present work, they are left for another time when they may be explored at greater leisure.

3.6 SUMMARY OF CHAPTER 3

We round off this chapter with a summary of the results and interpretations. This has not been an easy task. First, a summary of a summary (since this is what it is, this chapter being itself a summarised account of the work contained in 9 previous reports [37-45]) proved a thankless task. Then we had to avoid tackling what were intended for Chapters 5 and 8.

The activity measures of the BD system do not seem to show any time dependent behaviour of any consequence; they exhibit no or very weak trends, appear little affected or unaffected by one another and by their own past behaviour. Indeed, there is cause to believe that, should there be any influence upon such activities, the major portion of it gets dissipated rather than absorbed. Systematic signals at given frequencies which could allow the activity series to be termed cyclic (in the sinusoidal sense) or oscillatory were mostly undetectable. It, thus, seems that the best description for these activities is statistical invariance, or, more precisely, random behaviour. In addition, they appear to be normally distributed.

The comparison of the behaviours of these activity series with those of others from CCSS led to evidence which suggest that "larger" systems, such as CCSS, could exhibit more dynamics and inertia at the global level than "smaller" ones such as BD. However, the behaviour of smaller systems still do tend to resemble that of the larger ones; BD is continuously modified, the ripples in the growth pattern as well as the tidying-up process are common to it, too. The analysis of BD's growth also shows that since the different size measures employed show similar evolutions, the measures must all be strongly related and more or less equivalent to one another. This leads us to the conclusion that, for most studies, concentrating the analysis upon one (any one) of the obvious candidates for measuring size would probably provide most of the information that could be extracted. A final result concerning growth is that the larger share of it is through the addition of new components, the contribution of the increasing size (through modifications) of the existing components being quite modest in comparison.

The components population can be characterised in several ways. One finds that frequently changed components tend to be large, multifunctioned and situated at the highest hierarchical levels while the unchanged ones are small, highly specialised, unifunctioned and deeply-nested. One also finds that withdrawn components tend to have smaller attribute values than those still in use, probably because the latter continue to be changed and exhibit growing structure properties. This was also the case with the hierarchical level property; deeply-nested functions, however well understood in themselves, when redundant, tend not to be removed because their relationships with the other components may not be well understood and it may be preferable to avoid the risk of disrupting the functioning of the rest of the code.

Although various component subsets may exhibit certain distinct differences in their properties, yet they all appear to obey the 20-80% law. Indeed, the exponential distribution is roughly approximated by most. Furthermore, because of their potential usefulness in explaining the global behaviour, there is a case for carrying out the study of each of them (or, at least, the more important ones) in parallel with that of the whole population. Whether one looks at different subsets, at the whole population of the components or at both will largely be governed by the resources available, the number of subsets of possible interest, the homogeneity of the population and by whether one is more interested in studying the trees or the forest. Probably, a suitable compromise will have to be found.

By proposing the use of Jonckheere's test, we seem to have found a better way of exploiting the information concerning the modifications to the code to study the evolutionary behaviour of the complexity trait. This, as well as the other previously used tests, provided no evidence in favour of the hypothesis that modifications tend to involve an increasing number of components. On the contrary, there is reason to believe that, as the system ages, the tendency of most of the enhancements is to be of the tinkering type, involving fewer and fewer components although the number of those that involve more and more components tend to increase, too. There are then two types of improvements to the code: that of polishing it up (which is decreasing) and that of extending its capabilities (which is increasing). We also found that fixing existing code tends to disappear over time. The behaviour seems to agree more with the concept of a decreasing need to improve (in the polishing sense) and repair the system and with that of growing experience and understandability rather than with that of an increasing difficulty to modify it. There were further evidence which suggested that some sort of ideal state may be reached: the newer modules increasingly tend to be small, of the "bells and whistles" type and to replicate already existing functions. Their fix and enhancement rates as well as those of the newer

micros were found to decrease as exponential time functions although the decrease in the former is both more rapid and less influenced by external factors. The types of enhancements to the new modules also tend to become simpler but a trend, similar to that exhibited by the ancient modules, towards increasingly ambitious enhancements, although not ruled out, has not yet been discerned.

The exponential decrease model to the modifications behaviour of the newer components lent itself to predicting the time to attain a certain rate or for a given modifications total and to giving the probability of making a certain number of modifications by a certain time as well as providing a quality measure with which to compare the components. Also, it allows itself to be extended in several directions: to explain the behaviour of the already-settled components, to more realistically account for observations which cause the data to deviate from the idealised situation, e.g., when errors can be introduced through the modifications.

Finally, the individual component's exponential decrease model is compatible with the observed decreasing trend in the total number of fixes for the system and with that of increasing trend in the total number of enhancements. Because of the continuous addition of new components, the total number of fixes never quite vanishes. That of enhancements is increasing because, in addition to the enhancements to the newer components, the enhancements to the older ones remain appreciable, etc.

CHAPTER 4

ANALYSIS OF THE B SYSTEM

4.1 INTRODUCING THE SYSTEM, ITS HISTORY, ENVIRONMENT AND SOURCES OF DATA4.1.1 Introduction to B

VME/B or, simply B, is a large operating system which drives the most powerful of ICL's new 2900 series computers. In several of its aspects, it approaches OS/360's gigantism: it consists of the order of 4 to 5 thousand components. of 2 to 3 million lines of code, services scores of users and keeps busy a staff of more than 200 which is organised with at least 3 levels of management and which was, at one time, spread over 3 or 4 locations. But there are significant differences, too: the designers of this system have attempted to draw from the experience of older systems and to avoid their pitfalls. A disciplined design technology called "structural modelling" [75,140,142-144,165], developed in-house by ICL, has been used to take the design from the initial specifications through each refinement step to the actual implementation itself, which has been written in S3, a high level language very similar to Algol 68. Furthermore, the design had been supported by the facilities provided through an automated vehicle. This has been continuously improved not only to provide better assistance for the development of new software but is being currently used to assist the enhancement and maintenance processes pertaining to B.

Compared with the other systems, B is relatively young. Only 3 years separate the first user release date (beginning of 1976) from the time we completed the gathering of the data (beginning of 1979) and there had been only 4 user releases. That there has been such a small number of releases and that very few release-based data are available anyway, Section 4.2.1, show that there is not really much point in attempting a release-based analysis. Besides, this has already been the subject of a previous work [51]. Therefore, we have more or less been forced to adopt once more a time-based approach for the main part of our work concerning this system. Although we have attempted, whenever possible, to adopt the release-based approach, too, yet our emphasis on evolutionary behaviour will have to be that through time instead of that from one release to the next. What difference this fundamental departure from previous investigations makes can only be appreciated once the study has been completed.

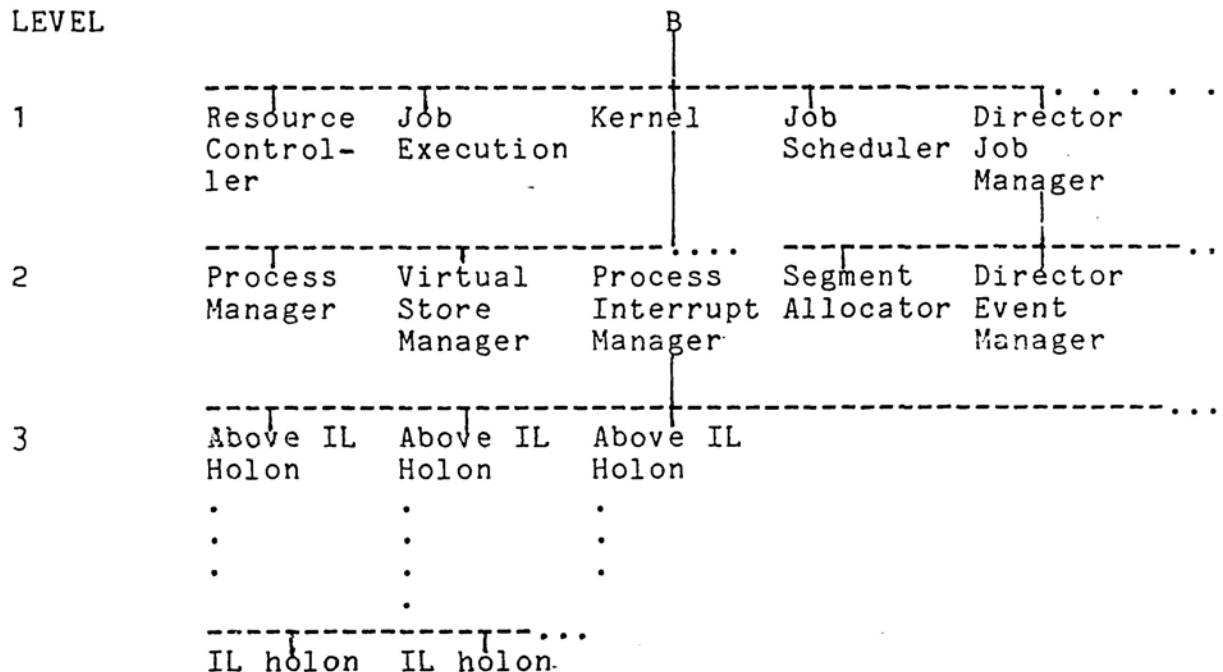
4.1.2 The structure of B

The structure of B is arrived at via the already mentioned

design process called structural modelling which, starting from the initial data specifications, proceeds to expand these and then the corresponding functional specifications from one level to the next, until the lowest level, where the data items are sufficiently small to be implemented, is reached. Then, each level is, in itself, a complete and self-standing representation of the design. Thus, only one level, the lowest, will appear finally as code and, although the approach and decomposition principle are similar, the end-product is totally different from that of structured programming.

The structural modelling approach led to a hierarchy of B "holons" basically illustrated thus:-

Diagram 12



At level 1 are groups which break up, at level 2, into major functional holons called subsystems. The subsystems then differentiate through several levels (of "above implementation level" holons) before reaching the "implementation level", IL. Because IL holons are the entities with which the staff work [143] (in preference to modules although strictly speaking there should have been a 1 to 1 correspondence, see [143] for a discussion of the anomaly) and because their structural characteristics, Section 4.2.1, are more easily measured, they are the basic units we chose to work with.

4.1.3 The project log and the SDL descriptions as the sources of structure data

To describe the holon-holon or holon-data relationships, a formal notation, SDL (System Descriptive Language), was developed. For the IL holons, that description merges into the implementation programming language, S3.

The project log is a manual file which was meant to be the prime source of information concerning B. It contains information about almost anything concerned with the system: its design, construction, testing and so on, complemented by SDL listings for all the holons, quite often several per holon, chronologically filed according to the changes they have suffered. For each holon, there are supposed to be as many SDL descriptions as there are versions of the holon (see Section 4.1.4 below).

Since by definition, above-IL holons are not actual part of the code (they only reduce ultimately to IL holons), their updating has not been enforced whenever changes are made to the system. For this reason, they have been ignored from our study. We, as mentioned earlier, concentrated upon the IL holons, henceforth simply referred to as holons.

4.1.4 Holon versions and phases

To identify each IL holon, there is not only a unique index number but also a phase number and a version number. The necessity for these is obvious from what follows. As B developed, it became necessary to support several instances (slight variants to suit the needs of special installations, the standard version in current use, the version under development, the test bed versions, etc.) of the product simultaneously. A new phase number is assigned to each redevelopment (major enhancement) of a subsystem to identify it from previous ones. A similar number is assigned to its holons but these are also given an initial version number of 0 to start with. Each individual holon's version number will then be incremented by 1 whenever it is modified. From the available SDL's in the project log, we could then extract all the relevant data for each holon for each of its phases and versions.

4.1.5 Other sources of data

Before describing the holons data analysis in detail, let us first introduce the other sources of data which were available. These relate mostly to activities that have been going on. Thus, dafs (Design Amendment Forms) form part of the mechanism which allows the supervision and control of updates to the software. Of prime interest to us are daf2's which formalise

the introduction of new SDL's into the data base used to construct the system and daf6's which are mostly used to indicate actual coding changes due to fixes, enhancements, hardware errors and so on to the individual modules. There, however, exist some daf6's which merely record change proposals. On the other hand, software notices are mostly used to advertise to the users some problems which have been found too complex to resolve (at least in the existing system releases). The deficiency is documented, restrictions of use are imposed and circumventions of the problem are usually suggested. However, there exist a few software notices which incorporate changes to the system! We also examined the bug reports from the (internal) users and bug responses to them. These last two types of documents are self explanatory.

More detailed descriptions of the software notices, the bug reports (and responses) and the daf6's are in [47], [49] and [50] respectively. Daf2's are examined in Section 4.3, software notices in Section 4.4, bug reports (and responses) in Section 4.5 and daf6's in Section 4.6.

Note Bug reports usually lead to daf6's (or software notices) which map on to daf2's but, as is obvious, there is no simple one to one relationship: several or none of each type may map to several or none of the other.

4.2 ANALYSIS OF THE HOLON ATTRIBUTES

4.2.1 The releases, subsystems, holons and data we worked on

Since it was impossible to tell from the project log which holons and holon versions belong to each release, we had to get hold of this information by some other means. This could be obtained from certain listings but only the complete lists of releases 5X27 and 5X32 Kernel holons, (K27 and K32), releases 5X23, 5X27 and 5X32 non-Kernel holons, (D23, D27 and D32) were available. (The non-Kernel list is known as the Director list although it comprises all the remaining groups in the system.)

We first compared the lists against each other to discover the new and withdrawn holons during each release. From the sums of new, withdrawn and all holons for the 3 releases, it was impossible to detect any trend although, if the pre-5X23 versions are considered, one sees that there is a definite increase in the total number of holons. More conspicuous from the lists is the observation that both the removals of the old holons and the additions of the new ones appear to be in bulk rather than on an individual basis, i.e. subsystems rather than holons tend to be the units in which facilities are either removed from or incorporated into a succeeding version.

But these lists were mainly intended as some sort of index to

allow the identification of the relevant SDL's in the project log. It would, then, be a simple, though lengthy, matter to extract each holon's new attribute values from these before passing on to the study of their evolution over the releases. From the voluminous lists and project log, it was clear that only a sample of the 5,000 or so holons (each with about 5 SDL's although only a maximum of 3 would have been examined here) could be analysed. We would have preferred random sampling and, at the 95% confidence interval, a sample size of about 1,000 would be required. This is not unmanageable, yet the data will be too thinly spread for us to pursue the analysis at the group (20 groups) and the subsystem (80 subsystems) levels. We, thus, opted for cluster sampling, the next best method. While choosing the clusters, we attempted to encompass all aspects of the system by:-

- 1) Including Kernel and Director holons in, as much as possible, the same proportion as they appear in the population.
- 2) Including groups at different protection, ACR, levels.
- 3) Including groups with 1, a few or several subsystems.
- 4) Taking only 1, a few or all the subsystems in a group.

Hopefully, a heterogeneous population, giving us the maximum chance of detecting dissimilar behaviour in the various subsystems under study, is obtained.

The set of the selected subsystems is {G1SSB, G1SSC, G1SSD, G5SS1, G5SS2, G5SS3, G5SS7, G9SS8, G13SS1} where GxSSy denotes subsystem y of group x. The group 1 subsystems would in some way represent the Kernel, K, while the rest the Director, D. However, to more adequately represent K, D or B, a truly random sample would be more appropriate.

After selecting the subsystems, we then went back to the project log to look up the relevant SDL's (i.e. those versions of the holons in the selected subsystems from 5X23 to 5X32). Unfortunately, to prevent the project log from fattening beyond bounds, it has been the policy not to retain many more SDL's per holon than strictly necessary; usually keeping only one for every phase was found to be adequate. This may be sound management since, in most cases, the change from one version to the next is only very slight: a line, a word or even only a character may be changed. But it did not quite suit our purposes. As there was no change of phase from 5X23 to 5X27 (for any of the subsystems we selected) and, although the version number of a holon may have been incremented several times, more often than not, it had only one SDL filed for both releases. There was, therefore, no option but to take the same SDL for both releases.

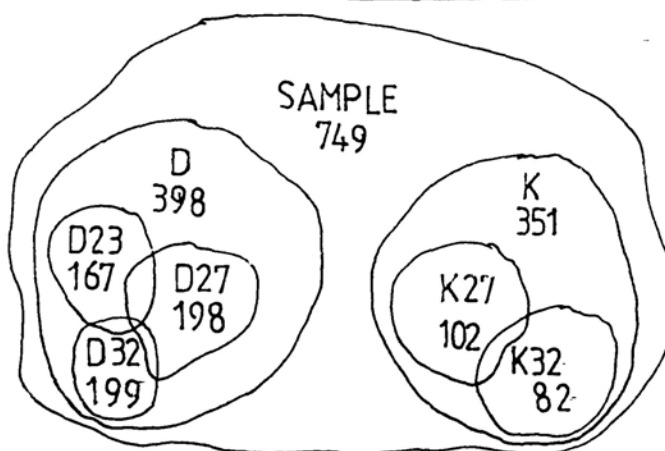
From the SDL of each relevant IL holon version, these

quantifiable attributes have been extracted:-

- 1) EXTS - the number of holons interacting, from the same subsystem, with this holon.
- 2) EXTO - the number of holons interacting, from other subsystems, with this holon.
- 3) IOE - the number of data items made available to this holon by other holons.
- 4) IOG - the number of data items made available to other holons by this holon.
- 5) IOP - the number of parameters of this holon.
- 6) CALLS - the number of calls made by this holon to other holons.
- 7) PATCHES - the number of temporary patches to-date to this holon (PATCHES are daf6's which actually led to changes in the holon).
- 8) SIZE - the size of this holon in number of lines of coded instructions.
- 9) DAF2 - the number of daf2's to-date to this holon.
- 10) AUTHOR - the implementer of this holon.
- 11) DATE - the date of implementation.
- 12) EXT - which is the sum of EXTS and EXTO.
- 13) IO - which is the sum of IOE, IOG and IOP.

Diagram 13 represents the breakdown of the 749 holons we have examined.

Diagram 13



4.2.2 Analysing the frequency distributions from one release to the next

There are thus 3 releases, 9 subsystems and 13 variables to explore. Examining the histograms for D23, D27, D32, K27 and K32 (see Figures 36 to 38 for 3 samples) indicates:-

- 1) That most distributions take a shape that ranges from

the exponential to the skewed normal.

2) That there is not much difference either among the 3 D distributions, for each variable, or between its 2 K distributions.

3) That, for each variable, the D and K distributions appear to be different.

Jonckheere's test, when used to test whether the D distributions are "increasing" from 5X27 to 5X32, provided no significant results (see [46] for the details). Three main factors may have contributed simultaneously to the non-significant results:-

1) The high proportion of holons which were unchanged over the 3 releases.

2) The effect of using the same data for many of the changed holons over 5X23 and 5X27, through lack of appropriate records, Section 4.2.1.

3) The general smallness of the changes.

On proceeding to pairwise tests, the D23 sample was dropped. The Kolmogorov-Smirnov 2-sample test, when used as a 2-tailed test, indicated that only CALLS and DAF2 differ from D27 to D32. A one-tailed test confirmed that the distributions for these 2 variables are larger for D32. This is what we expect for DAF2 while the CALLS result might be the first indication we have for an ever increasing number of calls made by each holon from release to release. This cannot remain unchecked if we wish to take note of the often repeated warning in the literature that doing so results in establishing a combinatorially increasing number of communication links, the direct cause, it is said, of increasing complexity in software systems.

When the K distributions were tested, no significant results were obtained.

It, therefore, seems that as a whole the structural characteristics that we have been able to identify, quantify, measure, analyse and infer from, have remained fairly static from 5X27 to 5X32 (and possibly as far back as from 5X23).

4.2.3 Analysing the holons classified by types of changes sustained

By classifying the holons according to the activities they have sustained from one release to another, we arrive at these 4 groups:-

- 1) The unchanged holons.
- 2) The changed holons,

with 3 subgroups:-

- a) Those that have exhibited growth in the attribute concerned. This is what is expected in most cases.
 - b) Those that have not.
 - c) Those that have undergone a reduction in the size of the attribute concerned. This may happen either consciously or unconsciously. Conscious attempts would fall in the category of "anti-regressive activity" [95], although one must not rule out an unlikely but possible increase in complexity despite a reduction in the values of all perceived attributes.
- 3) The removed holons, i.e. those that have been dropped from the previous release.
- 4) The new holons, i.e. those that were not in the previous release(s). If the system is getting less manageable and more difficult to add on to, then it is realistic to hold the view that the new holons will take on increasingly large attribute values, exhibit more "complex" distribution shapes from release to release and that, for any release, they may even appear as, if not more, complex than the existing holons.

These have been realised as follows for D:-

- UCD32, i.e. the group of Director holons which were unchanged from 5X27 to 5X32.
- CCD27, i.e. the 5X27 (Director) holons which were changed in 5X32.
- CCD32, i.e. the changed holons which are the new versions of those in CCD27.
- DD27, i.e. the 5X27 holons which had been deleted from 5X32.
- ND32, i.e. the 5X32 new holons.
- D27, i.e. the set of UCD32, CCD27 and DD27.
- D32, i.e. the set of UCD32, CCD32 and ND32.

Similar groups, UCK32, CCK32, etc., were found for the Kernel holons.

Note The changed holons have not been subclassified.

By choosing and comparing some pertinent pairs of distributions, (new, unchanged), (unchanged, (before) change), (before change, after change), using the Kolmogorov-Smirnov test, we have been able to show, from 5X27 to 5X32 at least, that:-

- 1) There seemed to be no difference between the new and

unchanged holons (whether K or D). If this result is representative, then very much the same sort of holons are being produced as at the start.

2) That, generally speaking, the attributes for either the unchanged or the new holons have smaller values when compared with the (before) changed holons, although a few of the results were not statistically significant.

3) That the changed holons have "simpler" distributions before than after change but the differences were slight, except in the case of PATCHES and DAF2 for D.

and that the following relationships generally hold

new = unchanged < before < after.

The next question concerns whether these relationships hold at all times instead of from 5X27 to 5X32 only. This will require many more analyses along the same vein as in this section with quite a few future releases before it can be finally resolved. However, the impressions which we have gained from the project log while looking at different SDL listings for the previous versions of each holon are that it is very likely that, at least, the "before" and "after" relationship holds and that the unchanged holons have, in the large, very rarely or never been modified at all.

4.2.3.1 The possible evolutionary patterns for the changing holons

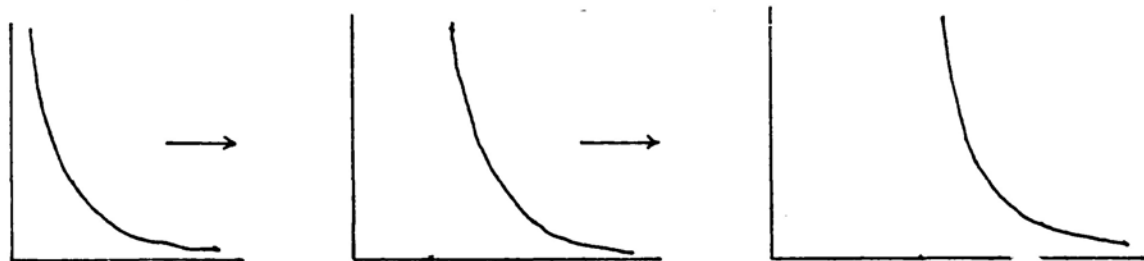
Assuming that a holon which is changed usually assumes a larger structure attribute value, then a consideration of:-

- 1) The negative exponential shapes we have obtained for all distributions.
- 2) The variations in these shapes (sometimes approaching skewed or regular normal distributions) over several attributes and several groups of holons.
- 3) The way in which the exponential shape can evolve.

leads us to believe that the distribution evolution of the changing holons by each structure attribute is a mixture of the following:-

4.2.3.1.1 A general and uniform shift of the frequencies from left to right

This is summarised by Diagram 14.

Diagram 14

4.2.3.1.2 A general but uneven shift of the frequencies from left to right

This may occur in several fashions depending upon the unevenness and the irregularity of the shifts over all the frequencies. Two contrasting examples are:-

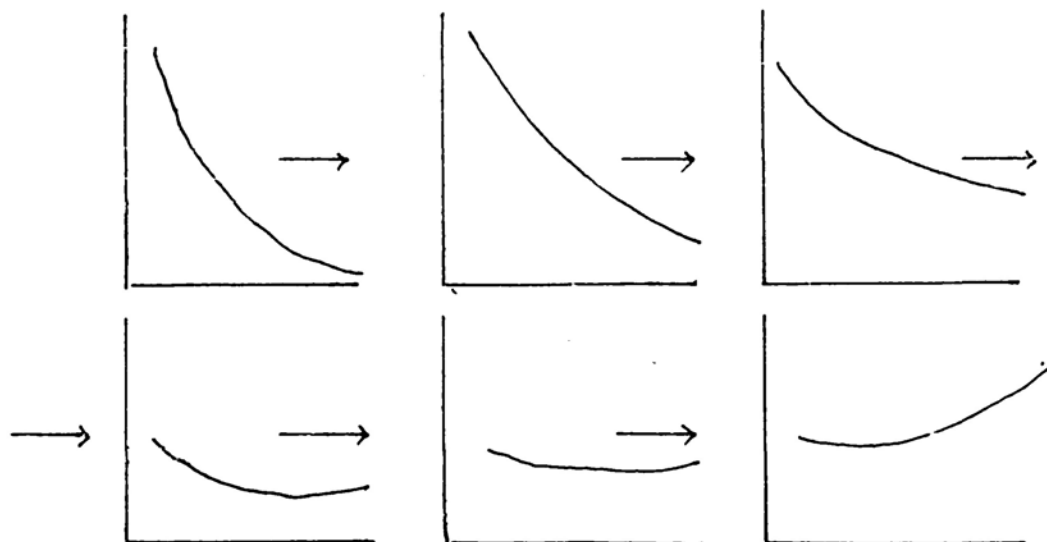
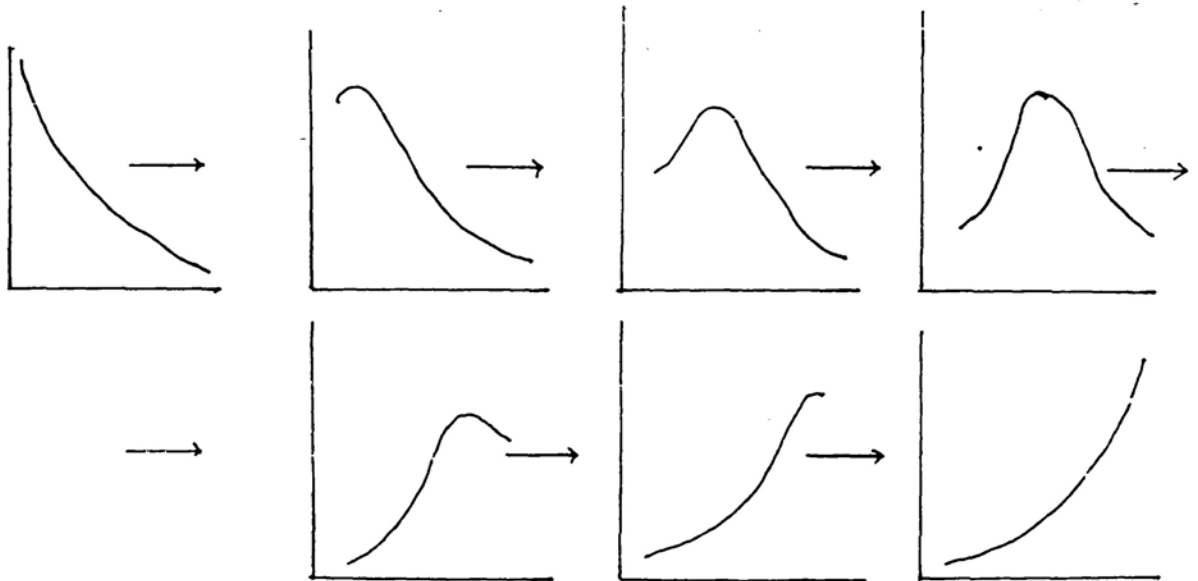
Diagram 15

Diagram 16

Note No statement is made about the rate at which the distributions evolve from the simplest to the most complex shape.

4.2.4 The global behaviour as explained by the different types of holons classified by changes category

Having looked at the behaviour of the different types of components in the population as in Section 4.2.3 and found a general relationship among them concerning the relative values of each of their attributes, it is now convenient to investigate the effect of adding them together. Using the (D27, D32) and (K27, K32) pairs, we found that, although most 5X27 distributions seem "simpler" than the 5X32 ones, yet quite a few behave otherwise. Thus we may conclude that there is by no means an inexorable rightwards shift in the overall holon distribution: the shift is not only checked but reversed in some cases by the combined effects of the new, unchanged and other types of holons.

Clearly, the more holons there are in each category, the greater will be the latter's effect upon the global distribution. The unchanged and the "changed but no growth" holon-sets each has a frequency distribution whose effect will be to dilute any shift in the overall distribution.

The effect of the "changed but reduced" holons will be to "simplify", i.e. cause a left shift, the next release's overall distribution. That of the new or removed holons will depend upon how each of their distributions compares with the overall one. Should the new holons be simpler (they probably are,

according to our results) or the removed ones be more complex, then each will simplify the overall distribution. Should it be the reverse, then the new overall distribution will become more complex.

The dilution/simplification effect of the new and unchanged holons is indeed important since 107/204 and 41/81 of the resulting (or 107/177 and 41/74 of the previous) D and K holons were unchanged. Similarly, 31/204 and 13/81 of the D and K holons in 5X32 were new and made significant contributions to the overall distribution. There are unfortunately too few deleted holons to allow us to see how deletion affects the overall distribution.

4.2.4.1 Interpreting the distributions' complexities in terms of the holon's complexities

Now if we assume, as is likely, that the distribution of the holons which have changed from one release to the next is indeed getting more "complex" or even if that of the overall population is behaving likewise, there is still a bridge to cross before we can assert that the complexity of the changing holons themselves (to be contrasted to that of their distribution) or that of the whole system itself is increasing. This is because a more complex distribution does not necessarily imply a more complex situation. First of all, a holon which has larger attribute values than another needs not be more complex than the latter. Then, a distribution analysis does not provide any information about the actual population involved. We may find a definite trend towards larger and larger structure values by the examination of the distributions yet counting the population may show that there could have been an actual decreasing trend in the number of components per release. The latter, from what we have seen, Section 4.2.1, is unlikely and, even if it could be realised, would probably not be on a sufficiently large scale to mess up the conclusions.

Then, even if an increasing trend in the attribute values has been found, one still needs to establish its links with increasing complexity before the latter can be claimed. Since increasing attribute values and unstructuredness are both consequences of changes to the code, one seems justified in using general increasing attribute values as a rough indicator for increasing complexity. Indeed, we needed this note of caution because too many researchers had been too ready to assume an increase in a trait whenever an increasing trend is found in a metric which could, at best, only be taken as a possible indicator of the increasing trait. Although the association between metric and trait may appear obvious, yet one is not the other and, unless there are absolutely unquestionable reasons for doing so, the behaviour of the former may not be taken to always reflect the behaviour of the

latter.

4.2.4.2 Relating the complexities of the different types of holons

Bearing the above in mind, one may then attempt to interpret how the complexities of those holons which are no longer wanted and dropped from one release to another, of the remaining ones which get changed and of the new ones which are added relate to each other. For each release, the new ones have been of a complexity which is comparable to that of the unchanging ones. Thus, if one could assume that increasing system complexity would imply increasing difficulty in constructing new components and the increasing complexity of the latter themselves, then, until now, the overall complexity of the system has probably not changed significantly. The changing holons, however, do get more complex after being changed and, as they were already more complex than the unchanging ones (whose complexities have been unchanging, too), it seems likely that the changes do not impact the holons uniformly but are, in the large majority, restricted to a minority of the holons only. (This has much in common with the 80-20 law found applicable to the BD micros, Section 3.4.)

4.2.4.3 The ageing and rejuvenating processes

With collections of machines, living things and so on, if there is no change in the constitution of the population, then, through wear and tear, the individual components will get less and less efficient and more and more unmanageable and creaky, all symptoms of ageing, until there comes a time when it becomes more profitable to replace the population by a new one (if non-living) or when it dies of old age (if living). But all the components need not be replaced (die) simultaneously and there is also the possibility of adding new ones (giving birth to them) to the population at any time. Then, whether the population is deteriorating (or ageing) as a whole or improving (or being rejuvenated), instead, will be determined, to a large extent, by the replacement rate when compared to the ageing process. Should the older (let us call them "older" since, assuming that they are utilised at the same rate, longer existing components will be more worn out than shorter existing ones) components be replaced by newer components at a rate which exceeds the ageing rate, then the whole population will be rejuvenated and may last forever. Otherwise, the time for replacing it will creep a little bit nearer.

4.2.4.4 Applying the ideas of ageing and rejuvenation to software

The analogy which one may draw between the behaviour of such systems and large software is attractive but perhaps not as

obvious as it may seem at first sight. Ageing, as applied to software, should imply more than a mere passage of time: if a software system is only repeatedly used over time but is not being modified at all, then it will not have "aged" in any sense since software has the property of not deteriorating through repeated usage. However, software does get changed and, in the process, does show increasing attribute values, Section 4.2.3. If, in consequence (see our note of caution, Section 4.2.4.1, again), it gets more unstructured, complex, creaky and unmanageable, then ageing for software would appear to be similar to ageing for all types of systems.

Should it be allowed to age unchecked, then there will soon be a case for killing the system off, Lehman's first law, and for creating a new, simpler system to replace it. But one may, more judiciously, adopt a policy of replacing the oldest components (the suitable cases for extermination) by new and less complex ones at a sufficient rate to keep the system in a reasonably spritely state. Thus, in parallel with the ageing process, rejuvenation is constantly required. Should the former be allowed to dominate at all times, then the march is relentlessly towards senescence and death. The statu quo may be maintained by balancing the two processes while, if the rejuvenation process is allowed to dominate, then, the system would become decreasingly complex. How the two processes relate may be indicated by how the inclusion of the new components with the changing ones affects the overall distribution. If the rightwards shift in the evolution of the latter is irremediable, then it will be ageing which has been the dominant factor throughout, if the distribution does not seem to be changing over time, then the statu quo has been maintained, otherwise the rejuvenation process would dominate. For B, heretofore, it seems that statu quo has been the more likely outcome.

The above concepts of ageing and rejuvenation are very close to Lehman's progressive and anti-regressive activities [95]. Progressive work is that which changes the system to improve its immediate capabilities, e.g., functional power, productivity and so on. Anti-regressive work, on the other hand, does not necessarily provide instant or visible returns but, instead, is a long-term investment to prevent future unmaintainability and decay. Lehman has asserted that progressive work must be complemented by anti-regressive activities, otherwise there shall be "unbalanced productivity growth", the consequences of which can be dire. This is exactly what our analysis suggests.

That B, as yet, has shown no propensity towards larger or smaller average attribute values and, hence, seems to be in the statu quo state concerning its overall complexity may be due to a combination of several factors:-

- 1) The system may be too young to start exhibiting the symptoms of increasing complexity as detailed by Lehman [18].
- 2) Increasing complexity is already prevalent but, as it is in no way related to the values of the attributes, it does not cause them to show a shift in their distributions to more complex shapes. However, it seems contrary to common sense to categorically deny that increasing attribute values may be related to increasing complexity.
- 3) We have too few releases to work upon to arrive at firm results in one sense or the other.

There is probably a degree of truth in all of the above statements.

Before proceeding to the next section, let us conclude by sounding a few notes of warning:-

- 1) We have not concerned ourselves with the actual ageing rate. Whether it is fast or slow has little effect upon the discussions - it may take an eternity for the system to decay for all that we may care. From the analysis of B which shows that a majority of the holons are unchanged from one release to the next and that the attribute values of the changing holons increase only very slightly, it seems that the changing rate is indeed slow. It is tempting to infer that the ageing or complexity increase rate is likewise slow but, again, one must be cautious: slowly increasing attribute values do not necessarily imply a slowly increasing complexity, too.
- 2) New components may be added not to replace old ones but to implement new functions and facilities. Even if each of them is as simple as can be, they will cause the overall complexity of the system to increase. Unfortunately, the overall distribution will be simplified. Indeed, this highlights one of the difficulties in attempting an analysis based upon the study of distributions only.
- 3) There are many other shortcomings concerning the use of distributions. For example, the actual size of the population is completely ignored so that, if, say, a system has reduced its number of components to one tenth of the original number, then, if the attribute values of those which are retained have sensibly increased, examination of the distributions only would perhaps wrongly imply that the complexity of the system has increased, too, while it may well have been otherwise (see also Section 6.2).
- 4) The components may not be of equal "importance" (the 20-80 law). Then the ageing and rejuvenation of the system will depend more upon the behaviour of the more important components than upon that of the others.
- 5) The actual attribute values themselves may not be too

meaningful. For the same attribute values, one can imagine all sorts of components or systems with varying degrees of complexity. Fortunately, in real life, most things conform to the norm.

6) One must not assume that we have shown that substituting a new component for an ancient one will decrease the overall complexity. Such an action may well decrease the complexity of the component in question but, if, e.g., it establishes many more new links with other components, then the overall complexity could increase. But, of course, methods of detecting this are provided for in 2 ways: we re-examine the whole new population and we study several attributes simultaneously, one of which is the number of links which a component establishes.

4.2.5 Correlation and multiple regression analysis

4.2.5.1 Correlation analysis

Further to the arguments in Section 4.2.4, one may add that the extent to which a system decays is not necessarily proportional to the quantity of properties which change but is, instead, a function of the changes to those, and only those, critical properties which govern structural degradation. Thus, although one may have observed and analysed thousands of changing properties, if the pertinent ones are not among them, then, whatever conclusions may have been reached, these would be mostly irrelevant as far as degradation or increasing complexity is concerned (see our remarks in Section 2.7, too).

Our next task was to attempt to sort out the pertinent from the other attributes. The chosen method was to try to identify a convenient degradation or, alternatively, quality measure first and then to relate the others to it. Given the objective of this research and realising what a tough task finding an all-satisfying metric would entail, we did not pursue the search beyond that of choosing the most likely from the already available variables. PATCHES seems to be our best bet, the next choice being DAF2. The obvious way to relate PATCHES with the other variables is to use the correlation analysis tool, despite its obvious weaknesses, Section 2.7. We also took the analysis one step further by trying to model PATCHES in terms of the other variables through regression analysis, mainly as an attempt to discover how appropriate such a model may serve as a forecasting tool. DAF2 was analysed similarly, but, it seems that, should we require one measure to represent quality and another for work, then DAF2 can be put to better use as a stand-in for the number of changes, an unavailable measure, and, therefore, as an approximate measure for work.

The correlation matrix in Table 18 is self-explicit. Because of

the high positive correlations which PATCHES exhibits with most of the other attributes, our decision to include all the attributes in the analyses in Sections 4.2.3 and 4.2.4 is justified. Furthermore, the feeling that increasing complexity may cause a general increase in a component's attribute values is strengthened, too.

4.2.5.2 Regression models for cumulative PATCHES and DAF2 of K32

4.2.5.2.1 The linear models

Because previous analyses have shown that the attributes of the holons have changed little from 5X23 to 5X32, Section 4.2.2, and because creating regression models is not much fun and rather lengthy, we concentrated for this, the first of 3 stages on regression analysis, with the examined K holons of 5X32 only, the latest version and, therefore, the most useful for predictive purposes for the as-yet-to-be-released (at the time of the study) 5X36.

The best PATCHES model, excluding DAF2 as an independent variable, is

$$\begin{array}{rcccc} \text{PATCHES} = & -3.295 & + & 0.602\text{CALLS} & + & 0.410\text{IOE} & + & 1.352\text{EXTO} \\ \text{s.e} & & & 0.072 & & 0.113 & & 0.469 \\ \text{t} & & & 8.36 & & 3.64 & & 2.88 \end{array}$$

The total R^2 of 77.92% is quite reasonable compared with those obtained with BD and with what other workers have been obtaining. In [178], Waldbaum only managed values of about 70%, even after extensive transformations. We note from the model:-

- 1) The absence of AGE and SIZE from the explanatory variables and the preponderance of the structural links of the holon: the number of calls which it makes and that of external data elements which it uses.
- 2) The coefficients on the right hand side are positive indicating that, as the value of each variable increases, so does that of PATCHES.

The best DAF2 model is

$$\begin{array}{rcccc} \text{DAF2} = & 0.220 & + & 0.332\text{PATCHES} & + & 0.073\text{AGE} & + & 0.236\text{EXTS} \\ \text{s.e} & & & 0.051 & & 0.021 & & 0.112 \\ \text{t} & & & 6.47 & & 3.42 & & 2.11 \end{array}$$

giving an R^2 of 70.04%. EXTS is still another communication attribute despite the inclusion of other communication attributes in the model for PATCHES.

4.2.5.2.2 The non-linear models

While the linear models are acceptable, further functional forms for the regression equation were explored in a bid to obtain even better fits.

Plotting, as before, Section 3.4.3.3, was not very useful and, therefore, the same transformations as for BD were used. In all cases, we came up with worse models, except when all the variables are squared beforehand. The total R^2 for the PATCHES model is now 86.56%, an improvement of about 9%, although we have 1 fewer explanator. Both remaining variables are concerned with the communication links of the modelled holon: its number of calls to other holons and the number of external data elements which it uses.

$$(\text{PATCHES})^2 = -52.881 + 0.675(\text{CALLS})^2 + 0.928(\text{IOE})^2$$

s.e	0.038	0.109
t	17.56	8.49

The new DAF2 model, $R^2 = 72.32\%$, is also an improvement.

$$(\text{DAF2})^2 = 30.946 + 0.201(\text{PATCHES})^2 + 0.163(\text{IOE})^2$$

s.e	0.021	0.053
t	9.61	3.07

Implications

ICL may think that to improve the quality of future holons, it is only necessary, from the previous results, to reduce CALLS and, say, IOE. However, these models should not be used for what they were never intended for. That the various variables are mostly highly correlated among themselves, Table 18, would imply complex relationships among them so that tampering with any one of them may result in compensatory or sympathetic action elsewhere. There is, then, quite a distinct possibility that any problem solving strategy may prove dysfunctional after all. For example, reducing CALLS may very well result in some other variable, say SIZE, to become the dominant factor in the next release (and model) and cause problems of its own.

Finally, that R^2 is not 100% leads us to make these 3 important and not mutually exclusive conclusions:-

- 1) There may be other structural characteristics which may also have an influence upon the quality of the holon and the amount of work done to it.
- 2) There may be exogenous forces or external determinants, e.g., market pressures, management plans, that impinge upon the system. These, of course, are much more difficult to measure or may even be too far removed both spatially and temporally to be empirically observable.
- 3) It may be inherently complex to characterise in

numerical terms the different factors that influence a programming system. An exact count may not be the correct answer. To illustrate, take DAF2 as our first example (see also Section 2.8). As another example, compare a holon which calls 3 others which themselves do not call any with another holon which calls only 2 but these in turn call many others.

4.2.5.2.3 The incremental models

We have, so far, used the "as at" values for each attribute (the cumulative approach) but one may also work with the changes from one release to the next - the incremental approach. We need not concern ourselves with the pros and cons of each approach here (see [48] instead). The data we shall use to create the models will be the following two sets: the differences between CCD32 and CCD27 and those between CCK32 and CCK27. One fundamental difference with the cumulative models is that, for the new PATCHES models, we shall no longer force DAF2 out of the list of possible explanatory variables. Since these 2 variables are usually highly correlated, whenever one becomes the dependent variable, the other can be expected to be among the explanatory ones. In whatever way the data is treated, there will have to be 4 models: D and K will each have one DAF2 and one PATCHES model.

The first models

We started by modelling Δ DAF2 and Δ PATCHES from 5X27 to 5X32 in terms of both the increases in the other variables and the actual variables themselves, on the grounds that the latter's magnitudes may also have some influence on the independent variable.

Additional models were created by including the rates of change as independent variables.

The absolute models

Next, the models were repeated with the absolute values of all the differenced data. The advantage of using absolute values becomes clear when one considers that large Δ DAF2 values will be negatively correlated with big reductions in size but positively correlated with big size increases while they are positively correlated with $|\Delta$ SIZE|.

The results and the conclusions

16 models were therefore created, Table 19. They gave the overall impression of great variety in the relationships which they exhibit, e.g., the independent variables include all

types, the delta, the rates and the actual attributes themselves in various combinations. It also seems that the models are improved whenever the rates are included as independent variables and whenever the changes are in absolute terms.

Generally, the best models are those in terms of the absolute changes and absolute rates of change attributes. However, the independent variables obtained for each model are not quite consistent with those obtained for their corresponding models. Although, for interpretative purposes, this is a disappointing result, as interpretations become inconsistent, yet it was entirely expected. The variables are so highly intercorrelated that only a slight manipulation of the data shifts the emphasis and is sufficient to cause one variable to replace another in the model.

The lessons learnt

The main one is that multiple regression models may not be very useful for interpretative purposes, most specially when the independent variables are highly intercorrelated. It is thus safest to stick with their use as empirical formulae to derive the value of the dependent variable, given those of the explanatory ones.

4.2.5.2.4 Applying a theory of stability or instability to B

Sutherland's different classes of systems

According to Sutherland [170], any phenomenon is amenable to modelling on any of 4 levels of analysis:-

- 1) The state variable level where the determinants of the system are identified.
- 2) The parametric level which involves assigning numbers to the state variables for specific points in time or space.
- 3) The relational level where the nature of the relationships among the state variables is established.
- 4) The coefficient level where specific numerical or magnitudinal values are assigned to the interfaces among the state variables.

For example, in the model

$$y = (0.5w + vx)/6z$$

,the state variables are v, w, x and z, the relational components are +, X and /, the coefficients are 0.5, 1 and 6 while, at the parameter level, we have to assign a specific value to each variable.

For deterministic systems, the simplest class identified by Sutherland, all but the parametric level are fixed. The moderately stochastic systems, the next more complex class, are those where the coefficients and the parameters may change but where the state variables and the relationships remain constant. The severely stochastic systems, even more complex, allow the relationships to change as well while the worst, the indeterminate systems, allow the state variables themselves to be inconsistent so that the basic nature of the system may change through time. Such systems are the most difficult to deal with.

The practical problems in adopting Sutherland's ideas

Unfortunately, the tools for building the models where any functional relationship is allowed are, to our knowledge, unavailable. At this state of the art, we shall have to settle for multiple regression models. Some sort of sophistication could be achieved by transforming the data but, even then, there would be no real mix of different types of relationships.

Although we shall bear in mind the 4 provisos outlined below (and already dealt with elsewhere), Sutherland's ideas are attractive and one has to see how they may be usefully applied in the context of ED studies.

The provisos are:-

- 1) Whether the state variables are adequately represented by the variables which we have.
- 2) Linear models provide only for the addition and subtraction relationships.
- 3) The weaknesses of regression models: no causal relationship, the strong intercorrelations making it easy to replace one explanator in the model by another without much affecting R^2 .
- 4) Only 2 releases, instead of the necessary several, are available for comparison.

One further complication is that any comparison of the models will have to be based on 3 considerations. In addition to the variables and their coefficients, we also have to consider the R^2 of each model. This task would be simple if each factor can be considered singly while the other two are fixed but, unfortunately, this is impossible.

Deductions from the incremental models

To find whether the same determinants are at work upon both D and K, we compared each Director incremental model against the corresponding Kernel one, Table 19. For each pair, there is usually only 1 common independent variable, (either Δ DAFS

or ΔEXTS). Could this be taken to indicate that, apart from the great influence of DAF2 upon PATCHES, (and vice versa), the other influences upon either of the two attributes differ from D to K?

Deductions from new cumulative models

The next (and last) step was an attempt to compare the influences at work over different classes of the system. Due to lack of data, only cumulative models could be built and used for that purpose. Table 20 exhibits the best linear models fitted to the sets listed under Section 4.2.3. Of these, CCD27 and CCD32 (and CCK27 and CCK32) have an additional feature: that of providing the only opportunity to compare the same things from one release to the next.

When the models in each of the 4 pairs for the changing holons are compared, it is found that, at most (and sometimes never) only one explanatory variable is substituted by another (from a total of up to 4) and that the directions of the relationships are unaltered although the coefficients are never the same. However, 4 explanatory variables are rather too many. If we reduce these to 3, then, a straight interpretation of Sutherland's ideas would suggest something between a moderately and a severely stochastic system. Before we conclude that there is no hope of predicting the system's behaviour with much certainty, we must remember that there is still much to be done in attempting to apply Sutherland's ideas. The main pitfall, at present, is, we believe, due to the unsuitability of regression models when the variables are highly intercorrelated. But are there better tools?

From the above models, it looks as if the subsystems within each release may not be very similar to one another. This is confirmed by the widely different histogram shapes (an example is Figure 39) we obtained for each subsystem for any variable. Although the results are as expected and pursuing the analysis at the sub-global level has previously led to very meaningful results, Sections 3.4.1, 3.5.2.2 and 4.2.3, we have not felt it necessary to repeat all our analyses at the subsystem level here. That would have taken too much of our time and creating a regression model for each subsystem would probably add little to this research although it may be of some usefulness for management purposes. However, this simple but laborious task is left to whoever may be interested in performing it.

4.3 ANALYSIS OF DAF2'S

(The detailed results of the analysis in this section can be found in [46].)

This section is analogous to that in Section 3.2. In addition, we shall be investigating, from the moment of its introduction into the system, the changes pattern over the life history of a "typical" holon as depicted by its number of daf2's per month, cf. the investigations concerning the typical new BD module and micro, Sections 3.5.2.3 and 3.5.2.4.

Before proceeding further, let us remark that any result which may be obtained cannot be strictly interpreted as showing the work input pattern into each subsystem. Not only is there no 1 to 1 mapping of a change on to a daf2 but a count of daf2's (or of changes) does not reflect the fact that changes can have various magnitudes and varying degrees of difficulty, Section 2.8. There is no consequence upon the interpretation of the results if, as regards to the magnitudes and difficulties of the changes, true stochasticity exists. If, however, one expects that, due to increasing complexity, it is getting more and more difficult to make a change, then a diminishing number of daf2's cannot be expected to indicate that the amount of work is decreasing. On the other hand, one cannot say that a diminishing number of daf2's is indicative of increasing complexity, if there is no proof that the actual amount of work is not decreasing. Thus, before being able to assert that our deductions concern the amount of work put into the system, we must determine first how the amount of work put in a daf2 is evolving over time. This is not an easy task and, as before, Section 4.2.4.1, the problems of how to unequivocally relate the behaviour of a certain measure with that of an unmeasured (and, perhaps, unmeasurable) trait of the system, of which it is a possible indicator, and perhaps convert it into an all-satisfying metric for the latter, like so many problems raised, Sections 3.4.2, 3.5.2.3 and 4.2.4.4 among others, are not part of our quest and could not be pursued thoroughly.

First, it was necessary to express the daf2's data in terms of time series appropriate for analysis. A weekly grouping of the daf2's was too fine since there were too numerous 0 or near zero values. Since the subsystems are only about 5 years old, the rather large 6-monthly or even 3-monthly period appeared not only in danger of smoothing out some possibly interesting features but there would not have been sufficient data points to work with. Therefore, the monthly period had been chosen.

4.3.1 Randomness tests and polynomial fits

The randomness tests, Section 1.3.3.2, showed that, at the 99% confidence limits, all the results were non-significant. We next fitted polynomials, Section 1.3.3.1, to each of the 9 series (one per subsystem) and to its cumulative sum. In each case, we obtained the mean and its linear integral as best fits to each series and its cumulative series.

These results then concord, in every manner, with those obtained in Chapters 1 and 3 (Sections 3.2.1 to 3.2.2) and vindicate our previous conclusions.

4.3.2 Time series analysis

The only significant acf's are in Table 21. Of the 9 series, 4 showed no significant autocorrelations from 1 up to 20 lags, 3 showed only 1 significant result while of the remaining 2 series, one showed 2 and the other 3 significant results.

The few significant autocorrelation values which were obtained were not only very small but may have been spurious results (1 spurious result is expected out of 20 results at $\alpha = .05$). Thus, the general impression is that, for any subsystem, the number of daf2's in any month neither has any special effect upon that of any subsequent month nor has been affected by that of any preceeding month. This non-dependency upon past or future behaviour result also implies that the fluctuations in the number of daf2's per month for each subsystem is probably not of a systematically cyclic or oscillatory nature. Instead, as implied by the previous section's results, it seems to fluctuate unpredictably within a certain range. This being so, there was no point in attempting to persue this line of investigation to its end by building the elaborate Box-Jenkins model.

The analysis in the frequency domain resulted in periodograms which, for each subsystems' series, show very few frequencies with significant intensities, all such intensities being low. The conclusions which we reached from the correlograms are therefore confirmed.

4.3.3 Investigation of the daf2's distributions

That any time-based signals, if they exist at all, have been found to be so weak as to be considered insignificant implies that, for practical purposes, the time factor, as for the BD modification series, Section 3.2.4, can be ignored so that the sequence in which the observations are made becomes immaterial. We are then in a position to investigate the distribution of the daf2's. From the histograms in Figure 40, it appears that, for any subsystem, the daf2's are approximately exponentially distributed. In terms of probability, this means that, for each subsystem, few daf2's per month are more likely than large numbers of them. Furthermore, it is again clear that each subsystem's distribution is different from that of the others, cf. the results in Section 4.2.5.2.4. Various tests confirmed the observation.

That the distribution for each subsystem appears to be exponential rather than normal should not be worrying: we have

been carrying out our investigations at a sub-sublevel. The global distribution, indeed, looks normal and, in that sense, concords with previous results obtained with global measures. The explanation is, of course, found in the Central Limit Theorem.

4.3.4 The typical holon's profile

This part of the investigation is to determine how the number of daf2's per month varies over the life time of each subsystem's representative holon. This is done by ignoring the calendar dates, shifting all the holons to the same origin, finding the average of the number of daf2's for these holons from month to month and finally plotting these average values against time. (One technique among several others that may be used, cf. the work done in Sections 3.5.2.3.1 and 3.5.2.4.)

Again, common exponential shapes, but differently parametrised for each subsystem, are obtained, despite the simpler method adopted for this system's analysis. These shapes are more or less regular for the first 5 to 7 months after "birth", after which they tend to become random. By breaking up each typical life history into its corresponding phases or redevelopments, Section 4.1.4, not only is an exponential plot obtained for each phase but the plots become smoother, Figure 41. Finally, smoothing the data results in even smoother shapes, Figure 42.

- 1) That, if there are no new phases, then the number of daf2's peters out over time.
- 2) That new redevelopments of existing code do indeed occur.
- 3) That there is a sudden increase in the number of changes after each such new phase.
- 4) That the decline is subsequently resumed.

,together imply that the tendency to make small changes and enhancements to existing code is to fade out and that this is probably not due to an increasing resistance to change and increasing complexity. Otherwise, how can we account for the new phases? As more substantial work which include new subsystems and rehaults to existing ones are always followed by a large number of modifications to the new code and by an increase in that to the remaining code, then, if the number of daf2's was increasing for the whole system, it would not be due to an increasing need to perform more and more work on older and older subsystems but rather to work on new subsystems or on new phases of existing ones.

Note The analysis in Section 4.3 was also performed with various sums of the subsystems to represent K, D and B (see Section 4.2.1 about the suitability of the sums and [46] for the details) but the results have not been of sufficient

importance to be included in this report.

4.4 ANALYSING THE SOFTWARE NOTICES

(A more detailed account of the work described this section can be found in [47].)

4.4.1 Purpose and contents of software notices

When, for a current version of B, a problem which is too complex to be resolved until a subsequent release is encountered, then a software notice documenting the deficiency, the restrictions of use to avoid the problem, circumventions of the error, among other things, is issued to all relevant users.

From the contents of the software notices, the only information which is potentially of interest and which is quantifiable consists of:-

- Reference number.
- Release version(s) of the product to which the error relates. It can be:-
 - All to date.
 - <release>.
 - <release> and before.
 - <release> onwards.
 - <release1> to <release2>.
- Date raised. Should the notice be updated and reissued, this date remains unchanged.

Although classifiable by the components affected, by whether they were reissued, not issued or withdrawn and so on, the notices were overwhelmingly of the ordinary kind: issued only once, never withdrawn except when the release version becomes obsolete and so on. As, besides, there were not enough of them, categorisation and the analysis of each category were not justified and were not carried out.

4.4.2 Time series analysis of the software notices

Choosing the calendar week as the aggregation period gave us 168 discrete data points, sequenced from 7601 to 7903 - quite a respectable figure. Visual inspection of the plot of the number of software notices per week as a time function indicates reasonable stationarity, i.e. no obvious seasonality or cyclicity was noticed, although further analysis revealed a mild decreasing trend.

The acf and pacf plots of the actual series, Figure 43, show that the most appropriate model would be AR(3), (3,0,0). As this might be an overfit, (2,0,0) and (1,0,0) were also explored for adequacy. To provide for the possibility that the acf and pacf plots of the differenced and actual series may convey other information, we also investigated the (1,0,1) and the (0,1,1) models. Table 22 exhibits all five of them: AR(3) is indeed the best.

Notes

- 1) The (1,0,1) model, the next best, provides very similar forecasts to the chosen AR(3) one.
- 2) Models (1,0,0) and (2,0,0), particularly the former, are obviously underfits, although not as bad as (0,1,1).

The Fourier frequency results, as usual, did not provide much additional insight as most of the signals were either weak or insignificant.

4.4.3 Analysis of each release's software notices

Some of the release version entries in the header are ambiguous, notably the "onwards" one. When this is coupled with the fact that there is no precise date over which each release is introduced, each release's time series becomes rather imprecise. This is why their time series analysis was not envisaged. Instead, we concentrated upon qualitative observations from the releases' time plots to get some ideas of the intra-release behaviour of the software notices.

We first noted the absence of seasonality and cyclicity. There were, however, interesting observations to be made on three patterns which were superimposed one upon the other:-

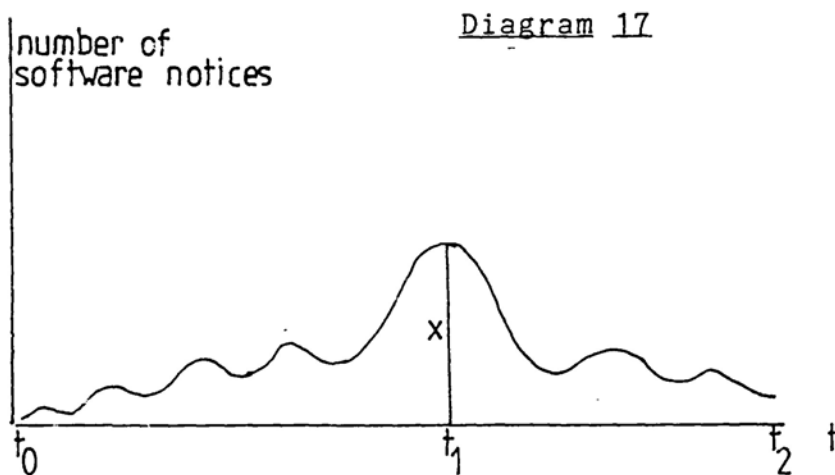
- 1) The data exhibit wild and irregular fluctuations.
- 2) Also obvious are the changes in levels at which the fluctuations occur. These levels may indicate periods when the rate at which software notices are being issued are high and others when it is low.
- 3) For versions predating 7601, i.e. 4X25, 4X30 and 5X10, there is an apparent decreasing trend. For 5X32, a recent release, there appears to be an increasing trend while for 2 of the 3 others, 5X23 and 5X27, whose whole life histories are covered during the period investigated, the plot is bell-shaped. 5X18, short lived, restricted to few users and so on [51], is anomalous and may be discounted.

By using moving average techniques (see [35,86,88] for a thorough discussion of filters), the local fluctuations were smoothed out and, further to confirming, very strikingly, the preceeding paragraph's second and third remarks, the new plots,

Figure 44, also contributed considerably to explain the various peaks in the plot of the global series, Figure 45.

4.4.4 Models of release and system behaviour in terms of number of software notices

Following the remarks in the preceeding section and collating the behaviours of all the existing releases' software notices time series, we arrive at the individual release's model which can be neatly summarised by Diagram 17.



It is easy to interpret the diagram. It takes some time (shipment to customers, their getting used to the new version, the elapsed times in feed-backs from finding errors to signalling them to identifying them to preparation of the notices and to distributing these and so on) for the rate of issuing software notices to build up to an eventual peak which should be the outcome of maximum usage experience, fault discovery and fault reporting. There, however, inevitably comes a time when the most visible faults would have been detected so that the fault reporting rate and, hence, the issue of software notices rate, decline. Of course, there are ups and downs in the process and the random element is strong, too, although, for clarity, we have not included the latter in the above diagram.

As should be obvious:-

- 1) The figure is not symmetrical about x .
- 2) The "run-in" time, $(t_1 - t_0)$, i.e. the time to the maximum notices issue, is not necessarily equal to the "run-out" time, $(t_2 - t_1)$.
- 3) The number and amplitudes of the waves to the left of x are not necessarily equal to those on the right.

Parametrising the model seems virtually impossible so that none

of run-in time, run-out time, x , area under the curve between t_1 and t_2 and so on can be predicted. The usefulness of the model is, therefore, mainly to explain and to qualitatively monitor some particular aspects of system behaviour. Monitoring may be in terms of a comparison of succeeding releases' run-in times, run-out times, peaks and so on.

4.4.5 The trends in the global time series and modelling them

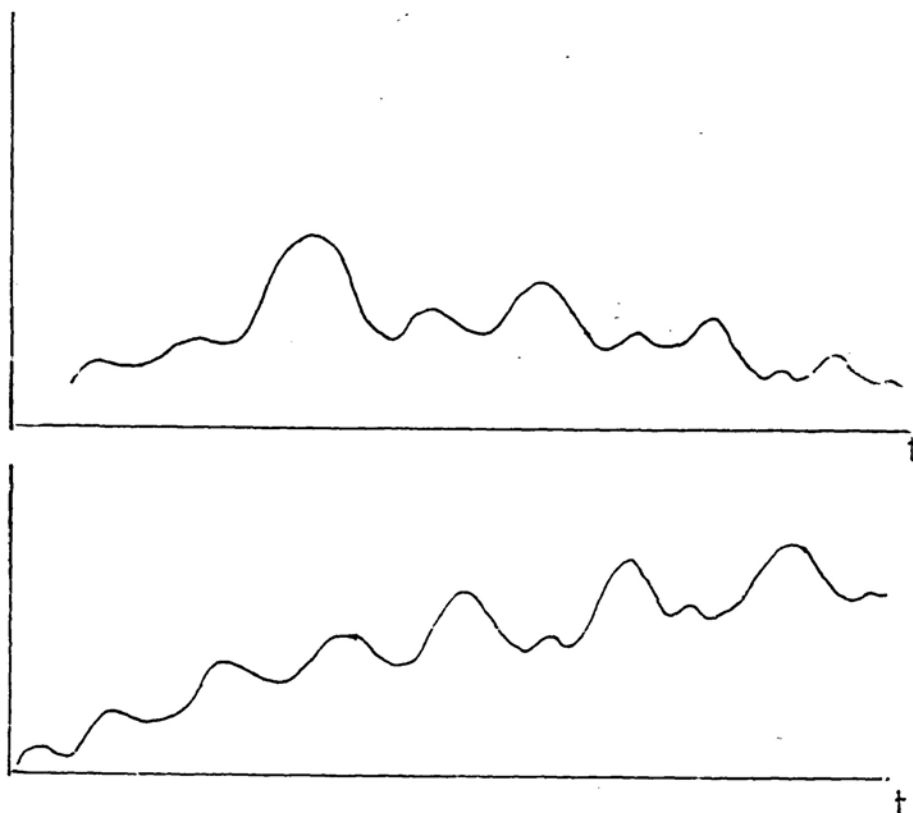
Having obtained some general ideas about the software notices' intra-release behaviour, it may be helpful to perform some inter-release comparisons, i.e. to observe how the software notices intra-release behaviours differ from one release to the next.

The similarities in the shapes of 4X25's, 4X30's and 5X10's plots, and the reflection of the latter's in 5X18's imply that problems tend to be common. They are, perhaps, omissions or global or conceptual problems reflecting defects in the initial design itself. There are indications that, with the passage of time these defects are being ironed out:-

- 1) The peaks, the means and, quite interestingly, the standard deviations, from 5X10 to 5X32 are regularly decreasing in magnitudes, Table 23.
- 2) In the global series, there exists a decreasing trend which is reflected in the Fourier analysis carried out and confirmed by regression analysis.

Thus, so far for B, there seems to be a general improvement. This could imply decreasing complexity but would not necessarily contradict Lehman's postulates [18]. This may be because the system is too young to start exhibiting the symptoms of increasing complexity. After all, it seems that most software notices have, so far, been issued to deal with mistakes or defects which have been handed down from the initial design and version and which are only being uncovered through increasing usage and experience.

Whether the decreasing trend in total number of software notices per period will continue or will be reversed remains to be seen. The effect of this upon the model for the system's behaviour which automatically follows from that of the individual release is obvious. Below are two idealised examples of the system's profile: when the trend is downwards and when it is upwards.

Diagram 18

4.5 BUG REPORT ANALYSIS

(A more detailed account of the work in this section can be found in [49].)

4.5.1 Introduction

Bug reports are forms which are filled in to signal possible errors and, therefore, problems in the use of the system. Each one results in a bug response. For this research, only internal bug reports (i.e. those emanating from within ICL) could be analysed so that no claim can be made about the applicability of the conclusions to external bug reports (i.e. those coming from ICL's customers).

The available data, covering the period 7708 to 7902, were so vast that, to keep this study within manageable bounds, we had to severely restrict both the analysis and the samples to be investigated. On the one hand, subcategories (and combinations of these) of the bug reports were not analysed while, on the other, only 4 different months' (7708's, 7803's, 7809's and 7902's) data spanning the full period and as distantly apart as possible were looked at in Sections 4.5.2 and 4.5.3.

The analysis is on 7 variables extractable from each bug report-bug response pair, each dealing with a different aspect of the data. There are 6 variables of classificatory nature: the priority class and the originator of the report, the release(s) and component affected, the machine on which the supposed bug was found and the type of the bug discovered, if any. A seventh variable, the response time, is of the interval scale.

4.5.2 Analysis of the distributions of the 4 chosen months' data

4.5.2.1 The different behaviours which the variables exhibit over time

The relative frequency plots in Figure 46 show great diversities, confirmed by the χ^2 test [163], among the four periods' distributions by any variable.

Breaking the distribution frequencies into the different classes of each variable shows that the variables are probably of 3 broad types:-

- 1) Those where each class within the variable shows no trend but is, nevertheless, fluctuating violently, e.g., priority and type of errors.
- 2) Those where each category shows some increasing trend which is then followed by some decreasing trend, e.g., release, originator and component. This behaviour seems easy to understand. In a development environment, effort is mainly concentrated upon the release being developed. Hence, in 7708, most work and bug reports were on 5X23, in 7809, they were mainly on 5X32 while 5X23 was already past history, and, in 7902, 5X27 had joined 5X23 as past systems while 5X36 was the new system given greater emphasis. The similar fluxes of the reports over the components can be explained thus: at different points in time, new components are introduced or/and old ones are modified, both resulting in more problems (reflected by the percentage of bug reports referring to these) relating to these components rather than to others. (This is very similar to the results we obtained in Sections 3.5.2.2.1 and 4.3.4.) As their problems get ironed out and as new components which, in turn, will require attention are introduced, there is a gradual shift of bug reports from the more ancient to the newer components.
- 3) Those whose behaviour mirrors the history of the system, e.g., machine usage directly reflects the distinct periods over which different machines were used for development or for testing.

4.5.2.2 Further analysis with the response time variable

The χ^2 test may also be used to investigate any differences in the response time distributions of bug reports of different priority types, Figure 47. There are 2 questions that one may ask:-

- 1) Whether, for each priority category, the response time has stayed the same over time. This provides for an analysis similar to that applied to the variable response time but at the level of each priority instead of at the global level.
- 2) Whether, for any month, hence, at any instant in time, the response time is a function of the priority category.

4.5.2.2.1 Comparing different periods' response time distributions of each priority type

Only priorities B and C bug reports were numerous enough for analysis according to 1). The evidence, on the basis of the χ^2 test, was in favour of the hypothesis that both the priority B and the priority C distributions, like that of the global data, have not remained the same over time. Since these constitute the majority of the bug reports, these results are as expected.

4.5.2.2.2 Comparing different priorities' response time distributions over the same period

For the orthogonal analysis, apart from comparing the distribution of the B priority bug reports with that of the C priority ones for each of the 4 months considered, the only workable alternative is to compare that of the A and B priority bug reports taken together with that of the combined C, D and E priority ones. We have chosen to work with the latter alternative since it is less likely to favour the hypothesis that there is no difference between the 2 distributions.

Ambivalent results were obtained. Only 2 periods provided significant results. This was probably because for all priorities, except A, the mean response times were comparable, whatever the period. Actually, except for D and E, the targets are nowhere met, the discrepancy being greatest for B (target of 1 day), the most common priority, despite the fact that we have ignored response times exceeding 30 days from our computations. Although ICL's objectives are to meet the targets in 90% of all the cases, yet the discrepancies still remain, Table 24. Either other factors such as the administrative procedures to be followed and the inelastic time required to sift through all the diagnostic evidence take so much time that the targets become unrealistic or priority, except A, means little to the responders. Could priority be a placebo whose sole purpose is to give a sense of reassurance to the users?

4.5.3 Correlating one variable with another

For each period, the contingency coefficient [163] between each pair of attributes is determined. Although the 4 contingency coefficients (one per period) between any pair did not show any apparent trend yet they were useful in confirming each other. Of the 21 possible correlations, only 7 were found to be non-significant, 6 of them involving response time.

Some of these correlations did not make much sense in themselves and could be wrongly interpreted but not if one remembers that they may be correlated only through their associations with another common variable. Thus, the (machine, type of error) correlation is meaningful by virtue of the fact that, due to the type of task each originator is undertaking, he has to use an assigned machine and that he brings along his skills and working habits with him.

The results seem all explainable by these 3 postulates:-

- 1) Originators (single individuals or groups of persons) have different working habits, they assign priorities differently, are worse at making some categories of error than others and so on. This, with the originator as the cause, explains the (priority, originator) and (originator, type of error) correlations.
- 2) At any moment in time, each originator is assigned a specific task: development or testing, among other things, of specific component(s) within specific release(s).
- 3) Specific machines are allocated for specific tasks. Some are for development and others for testing, etc.

Such correlation work may be useful in identifying to management the particular entities or aspects of the system that could be most usefully monitored. From the significant results, one can easily identify the originators who were most active (such an originator would have more than the expected number of bug reports to his name during most periods) or, at any given time, the component or release which was undergoing the most development, the machine on which development was carried out and so on.

The previous findings that priority A and B bug reports tend to have quick response times (up to 1 day) were confirmed but, thereafter, there were no distinctions among the various priorities. As appropriate, there was no correlation either between priority and type of error (priority being assigned by the reporter while categorisation of error is performed by the responder) or between originator and response time (nobody had preferential treatment). The absence of correlation between release and response time may be taken as meaning that the time taken to repair bugs has so far been independent of the release

sequence. There has thus been no increase in complexity (confirming previous results) so far as the response time measure shows.

4.5.4 Analysis of the bug reports and the bug responses time series and the outstanding queue

4.5.4.1 Analysing each series on its own

4.5.4.1.1 The time domain results

As no categorisation of the data is required for the work described in this section, it was possible to work with the whole sample. The weekly and monthly REP (bug report) and RES (bug responses) series, Figure 48, indicate no systematic patterns, were best fitted with their means and their cumulative plots were straight lines, all being results similar to those we have come to expect. (The daily series, too numerous to be plotted, gave similar results.)

The shape of the correlogram for daily REP, Figure 49(a), is typical of series which exhibit short term correlations. These are characterised by a fairly large value of r_1 followed by 3 or 4 more coefficients which, while significantly greater than 0, tend to get successively smaller. AR models are appropriate for series of this type. One other feature of great interest is the value of r_5 , which is conspicuously larger than its neighbours.

These 2 aspects of the behaviour of the daily REP series can be translated thus:-

- 1) Observations above the mean tend to be followed by some further observations above it and vice versa for observations below it.
- 2) There is some weekly pattern in the sense that, for certain days of the (5 day) week, any observation tends to be followed by another of similar magnitude one week later. A glance at the data shows that, for Mondays, on the same day of the following week, in particular, this effect is real enough though by no means is it regular or acute.

The daily RES series' correlogram, Figure 49(b), is different from the daily REP's in 1) that there is no weekly effect and 2) that, although its pattern is similar to that of daily REP, the significant correlations start only as from r_2 .

The weekly series' correlograms show that longer term dependencies were non-existent, r_1 being the only significant result, Figure 50.

The possible time series models which were suggested by the acf

and pacf plots of the daily and weekly series were explored. The models, Table 25, confirm that:-

- 1) (2,0,0) is an underfit (the acf of the residuals at lag 5 is still significant) while (5,0,0) is an overfit (see the non-significance of the third and fourth autoregressive terms).
- 2) The first autoregressive term of (3,0,0) is non significant.
- 3) (1,0,1) for both the weekly series are overfits. In each case, the correlation between the z_{t-1} and the a_{t-1} terms is more than .9. The t statistic in 3 out of 4 cases indicates the non-significance of the term.

Since, for each weekly series, the (1,0,0) model is to be preferred, the interpretation should, therefore, be that each week's number of bug reports (or responses) is unaffected by that of any previous week, except the most recent's. Weekly REP and RES are then Markov processes. Finally, before passing on to the next section, let us remark that although the dependencies which we have uncovered may be real enough, yet their effects are so short-termed that it seems doubtful whether they may be successfully exploited for effective planning purposes.

(Apart from confirming that the random element tends to dominate, the frequency domain results did not provide any further information.)-

4.5.4.1.2 The analysis of the outstanding bug reports

The outstanding bug reports values at the end of each week, Figure 51, show that, after an initial and steady rise during the first 3 weeks, they remained fairly stationary, or maybe slightly increasing, for about 26 weeks until a sudden rise for about 4 weeks and, thenceforth, a steady decline. The increasing queue length during the initial period indicates an excess of REP rate over RES rate over that period while the final decrease is due to the excess of RES rate over REP rate. Two conclusions can be derived:-

- 1) That the present set-up is quite capable of dealing with an extra (unspecified) load without undue hardship.
- 2) That RES has been the more unstable series. This is confirmed by its larger variance. Coupled with the weekly effect found in the daily REP series but not in the daily RES series and the closer fitting model for the weekly REP series, this leads to the very interesting insight that the greater stability in the REP series could be due to the greater inertia of the system in this aspect of its behaviour. This point, on which we shall capitalise in Chapter 8, is sufficiently important to be developed

further in the succeeding paragraphs.

Bug reports emanate from many more sources than responses and there may be so many of these that any individual contribution tends to become negligible. In the sense that each source's contribution does not account for very much to the global behaviour, some sort of global stability is achieved. One would expect REP to conserve its characteristics unless some drastic changes occur in the future. As bug response sources are more limited in number, each source exerts much more influence upon the whole behaviour. RES has therefore less inertia and is more manipulable. Thus, while the effect of removing or adding a single responder has quite a visible impact upon the global behaviour, yet changing the number of reporters by several may be barely noticeable.

This highlights the difference that exists generally between those entities which are subject to a large number of influences and those which are not. Our theme is not only closely related to the observation that large programs have "dynamics" (to use the opposite word to inertia) of their own and are not directly under the control of management [18] but takes it one step further in that some explanation as to why large programs and not small ones exhibit dynamics and are worthy of ED study is provided. We should perhaps also state that dynamics should refer only to those properties which are under multiple influences. Then, dynamics or inertia or statistical invariance is not really due to the actual physical sizes of the programs or of their environments. But, since, on the level we are dealing with, gigantism in size goes hand in hand with gigantism in almost everything else, this subtlety is not practically important. Further support for the dynamics or invariance theme is provided by the Central Limit Theorem, one form of which states that the sum of a large number of independent variables, none of which predominates, tends, in the limit, to have a normal distribution.

One may also speculate that the weekly effect in REP is probably due to a similar effect, that the cumulation of the very many simultaneous REP sources causes the amplification of some slight but concrete pattern in the working habits of the development teams. There may be some deterministic element or pattern in each person's working habits, probably due to human nature, the work environment and whatever. The same element probably exists in the responders' working habits, too, but there may be too few of them for the pattern to emerge from the stochasticities when their outputs are added together. We shall return to this point in the next section.

4.5.4.1.3 The typical week's REP and RES

An attempt was made to see whether there is a pattern in the number of bug reports (and responses) for the consecutive days of each week and to identify that weekly effect. This work was

extended to find out whether the pattern, if any, could have evolved over time.

The typical week's REP and RES values as represented by the averages for each day for the whole period considered are depicted in two of the bottom plots in Figure 52. Apart from the negligibility of the Saturday and Sunday activities, there is no truly observable pattern.

We then divided the series by quarterly periods and replotted the profiles for each week within quarter. The REP profiles showed no pattern but indicated that Monday is almost invariably the highest activity point in the week, thus accounting for the significant r_5 autocorrelation value. With the RES series, however, Mondays are not usually high points and no other patterns are observable. One can only speculate why more bug reports are received on Mondays. The developers may work in such a fashion that crescendo is reached only at the end of the week so that quite a lot of bug reports are only raised at the last moment and received the next Monday. The weekend rest may provide an opportunity to conceive, consciously or subconsciously, more bright ideas. Once Monday arrives, there is a flurry of activity and a mini flood of bug reports. Probably the same effect exists for the responses but it has not been discovered. Thus, the add-on effect which tends to smooth out random fluctuations and to amplify any slight deterministic element becomes important only when the number of sources is significantly large as for the bug reports but not when it is small as for the bug responses.

4.5.4.2 Crosscorrelating the REP and the RES series

So far, each of the two series has been considered on its own. We now turn our attention to the possible relationships between them. The cross-correlograms, Figure 53, show that there is no monthly pattern. At the weekly level, the most significant correlations are for the same week, one week later (and possibly one week before). Thus, weekly RES lags behind weekly REP by 1 week at most.

There are several significant crosscorrelation values between the daily series. Most of them are for positive lags, the most significant of which occurs at no lag. Thus, daily RES lags behind daily REP by 0, 1 up to 4 and 7 and 8 (working) days and leads it by 1 or 2 days. Of the two ways to interpret the results: 1) that both series have independent rhythms which are in step with one another and 2) that one leads the other and causes it to follow a similar pattern, it is probably the latter which is the better alternative. Thus, a high (or low) value in REP causes a value of a similar magnitude in RES the same day and/or 1, 2, 3, 4, 7 and 8 days later.

One direct application of this result is in terms of short range RES forecasting. There is also the possibility of obtaining a useful Box-Jenkins transfer function model using REP as the input to and RES as the output from the process. But such short-lived relationships do not seem to be of great applicability, however real they may be (see also Section 4.5.4.1.1).

In situations of inputs and outputs to a process, one may also attempt a queuing model. However, queuing theory does not seem to be very useful in our case because not only are the input bug reports and output bug responses values each serially correlated but the mean service time (see the next section) and the outstanding queue length are steadily decreasing. There is, therefore, no equilibrium solution and no estimate is possible for the traffic intensity, the expected queuing time or the probability that the service is idle. Furthermore, the service time distribution could not be easily characterised: a Poisson fit, the one usually assumed, proved unsuccessful. Thus, it seems that the Box-Jenkins transfer model would be more useful but it would have involved too many resources which were unavailable anyway. Therefore, we did not proceed along this road any further.

4.5.4.3 Analysing response time

After the response time to each bug report has been computed, the results were grouped by month of response and the monthly mean tested for time trend. Since there is an adequate number of months (19), our problem lies in the fact that, although most responses occurred within a few days of bug report reception, quite a few response times take up to one year and cause a lot of distortion to the calculations. Large outliers, therefore, needed to be excluded from our analyses. But where lies the demarcation between a large and a small value? Instead of answering this question, we analysed the data in three different ways. These should confirm each other's results.

We first found the monthly means of the times for increasingly larger ranges, Table 26, all of which when plotted, Figure 54, show decreasing trends. Further support is provided by the increasing trends in the plots of the percentage of reports requiring response times up to 3, 7, 11, 14, 22 and 30 days, Figure 55. That the slopes of these get less steep implies that the shift from longer to shorter response times is mainly concentrated in the low values. A third method of looking at this data is the comparison of the short (to fix), average (to fix) and long (to fix) response times. If one was able to associate "short" times with the reports concerning the easy to understand (and to fix) part of the system and the "long" times with those concerning the hard to understand part of the system, then a steady decrease in the average of the short

times would indicate growing familiarity with the easy to understand part of the system while an increase in the average of the long times would indicate growing difficulty with the hard to understand part of the system, i.e. growing complexity despite increasing familiarity.

One could take σ , or 2σ as the demarcating limits but the distributions were far from normal and, since taking σ would have been arbitrary in some sense, we decided to take the first 20% of the values as being short, the next 60% as being average and the rest as being long. The short times, being mostly 0, were forgotten altogether. The plots of the averages of the others are in Figure 56. The average plot is decreasing (albeit stochastically) confirming previous results and indicating increasing understanding of certain parts of the system (if these can be linked to the average bugs). The long plot is difficult to interpret: it may be slightly increasing in the first half and then slightly decreasing during the second. Therefore, it could not be used to infer increasing complexity, even if the latter is indicated by longer long times.

One other interpretation for the decreasing trend in response time is that the responders are getting more sloppy. This is implausible because using less care would increase the probability of making an unsatisfactory response. This would have surely boomeranged causing an increase in the number of the bug reports. This would have been detected by significant crosscorrelations with RES leading REP. It then seems more likely that the responders are getting more experienced. Probably, a combination of factors like greater familiarity with the code, problems tend to have been encountered before, the responders have acquired skills of detection is at work. The absence of a decreasing trend in large response values indirectly supports this interpretation: they seem associated with rarer problems.

Another possible indicator of increasing complexity may be an increasing trend in bug reports which have been handled more than once. It seems plausible to assume that an increase in the number of times a bug report is transferred from team to team would indicate increasing difficulty in locating faults and, hence, growing complexity. Unfortunately, the lists from which we extracted our data contained at most 2 component entries: the true one affected and the last of any previous ones which may have been unsuccessfully examined. Nevertheless, analysing the proportion of bug reports that have been passed on more than once will still lead to valuable results and similar conclusions, which, upon analysis, are found not to favour the increasing complexity idea, Figure 57.

4.6 ANALYSIS OF THE DAF6'S DATA

(A more detailed account of the work in this section can be found in [50].)

4.6.1 Introduction

Daf6's are used to indicate actual coding changes such as fixes or enhancements to individual modules. There, however, exist some "information only" daf6's where some proposals or malfunctions are described. We have concerned ourselves with the main maintenance site's daf6's only and these impact 31 groups (see Section 4.1.2) which have been introduced into and removed (if removed) from the system at different points in time. Other diversities among the groups can be seen in Table 27.

The daf6's, spanning the 5 years between 7403 and 7902, concerned the pre-customer releases 3X and 4X as well as the customer releases 5X10, 5X18, 5X23, 5X27 and 5X32.

From the daf6's, these 12 items of information are obtained:-

- 1) INIT: who "raised" the daf6.
- 2) MODULE affected. Usually, but not always, a module consists of a single holon.
- 3) MTHS: the time (in months) between the last compilation of the module under modification and the raising of the present daf6.
- 4) UTIL(ity), which consists of 2 parts:-

Availability, which can be either Gen(eral) or Sel(ective), i.e. for certain specific users, and
Applicability, which can be Gen(eral), Opt(ional) or Diag(nostic).

- 5) CAT(egory) of change, due to and/or categorised as: specification bug, implementation bug, other software bug, hardware interaction, enhancement or test aid.

Note Frequently, specially during the early months, the bugs were not subcategorised.

- 6) MODE: source or object mode modification or whether "information only" daf6.
- 7) VERSION(s): of product affected.
- 8) ATTEMPTS: number of attempts until successful.
- 9) WD: whether withdrawn or not.
- 10) BUGREP: the references of the bug reports which are associated with (caused) this daf6.
- 11) REPLACES: the reference of the daf6 being replaced.
- 12) DEPENDENCIES: the references of any daf6's which depend in some way or another upon this one.

Apart from collecting the above information from the daf6 forms, we further derived NV, the number of releases affected from VERSION, and TIMES, the number of times each module is "daf6ed" per unit time, from MODULE.

4.6.2 The analyses we performed

So far, most of the evolutionary behaviour we have investigated has been over time. But evolution does not occur solely as a time function: there are significant changes from one release to the next, too, whatever the release interval, the releases acting as some sort of anchor points. It is, therefore, equally meaningful to investigate release-based evolution, as Lehman has done. Here, we have an opportunity to adopt both the time and pseudo-time (release) approaches, which may then be compared and contrasted. However, one must immediately point out that it is irrelevant to ask which is the better of the 2 approaches and to attempt to choose between them. Each investigates evolution on a different dimension, which, although similar in many aspects to the other, is, nevertheless, an altogether different thing. What is, therefore, observed in one may not be observable in the other. It would, then, perhaps be preferable to combine both approaches for comparing the time evolutions of succeeding releases' variables. As one can easily imagine, the complexities involved are such that only the idea can be formulated at present. The nearest we have come to achieving such an aim is to qualitatively examine each release's software notices behaviour (the intra-release behaviour), Section 4.4.3, and then to qualitatively compare them together (the inter-release behaviour), Section 4.4.5.

Then, whether on a real or pseudo-time basis, there are, for each variable that we may care to select, 2 main analyses, orthogonal to each other, that may be performed: one may make group to group comparisons, given a suitable time period or pseudo-time (i.e. a static study) or one may make comparisons over time/pseudo-time (i.e. study the evolution) for any group one may wish to explore, cf. the analyses in Sections 4.5.2.2.1 and 4.5.2.2.2. These comparisons will be mainly in terms of the distribution of the daf6's among the various categories of each chosen variable though a simplified version of the analysis, involving only a single measure - that of diversity - was also performed.

In classifying all the daf6's according to any variable, one finds that a 3 dimensional frequency box, with the groups forming one axis, time/pseudo-time another and the categories of the variable itself the third, emerges. As each variable allows a new reclassification of the daf6's, it results in a new frequency box and allows a new facet of the data to be explored. Of the 8 possible ways of combining the 3 dimensions,

those of taking none or all 3 at the same time constitute too simple or too complicated a task. They were therefore ignored. Some examples of analyses which involve taking 1 dimension at a time are:-

- 1) Comparisons over the group dimension with any combination of categorised/uncategorised daf6's and chosen/whole time periods.
- 2) Comparisons over the time dimension with any combination of categorised/uncategorised daf6's and group/whole system.
- 3) Comparisons over the category dimension with any combination of group/whole system and chosen/whole time periods.

Evidently, a casual examination of the data would suffice to provide all the necessary answers to some of the above while others, notably those where the categorisation of all the daf6's would be required, would involve an amount of work which is beyond our scope.

The main part of our work here, therefore, has been concentrated upon the simultaneous consideration of 2 dimensions, notably:-

- 1) When group and category are combined to make distributional comparisons over the groups: how does the composition of a variable into its various categories vary from group to group (given a certain time period)?
- 2) When time and category are combined to study the distributional evolution over time of the daf6's for a given entity (group or whole system).

The third combination was deemed of lesser importance and only slightly dealt with.

Transcending the frequency boxes were further analyses: the most interesting were that of exploring the degree of association between different pairs of variables (for a given group and time period) and that of exploring the concordance among the variables and among their diversities, too, for a given group over all the releases or for a given release over all the groups.

4.6.3 Choosing the groups, variables and time period to use

Prior to attempting the analyses, we had to face several problems, all related to too much of everything: too much data (thus ruling out the analysis of the whole system as a single entity), too many groups, too many variables and too many time periods if the aggregation period is not chosen judiciously. Because the amount of analysis to be performed increases explosively whenever there is a slight increase in the quantity

in any dimension, we had to:-

- 1) Choose only a few groups to explore. We hypothesised that the groups can be arranged into a few clusters only. The analysis of a typical group from each cluster would then lead to results generalisable to the whole cluster.
- 2) Exclude from the analysis the uninteresting or the analytically intractable variables.
- 3) Choose an appropriate Δt with possibly conflicting aims: so that the daf6's are as evenly distributed over the periods as possible, that there are not too many periods to explore and smoothing out the stochasticity while preserving the characteristic trends, if any.

4.6.4 Clustering the groups

To cluster the groups, we ruled out the numerical taxonomy methods [3,53,60,166] as too complicated and not necessarily relevant for the job, Section 2.3. The groups' daf6's time plots, Figure 58, provided a crude method of clustering the groups and resulted in 4 discrete clusters, each having a plot which is characterised by:-

- A) A single early peak, e.g., groups 9 and 40.
- B) Several peaks which are neither necessarily equispaced nor ordered in their magnitudes, e.g., group 5.
- C) A single peak which does not occur early on, e.g., group 31.
- D) Uniformly low levels.

Alternatively, we ordered the groups by their figures of merit which, for each, has been taken to be the sum of its ranks on each of its observed properties, Table 27. These ranks were found to be neatly demarcated into groups which correspond to the discrete clusters obtained graphically, Table 28.

When the figures of merit are recomputed without the daf6's information and reranked, the groups, apart from the top 6 or 7 B ones, get mixed up. The latter, but not the others, are characterised by small ACR level numbers and large sizes, these results being corroborated by significant positive Kendall's rank correlations [163]. It is, therefore, probable that the most useful and functionally important groups are not only the most complex ones and those which are at the very heart of the system (at its lowest ACR levels) but are also the ones most prone to modifications and redevelopments. One wonders whether they need to be that large at all. As largeness compounds the difficulty in performing a change, would it not have made life easier to have designed each of these groups into several smaller, independent and functionally distinct ones?

4.6.5 The global daf6 distribution over time

The global daf6's plot, Figure 59, shows that, after an initial sharp rise, there is a steady decrease in the rate of raising daf6's for the whole system. As most of the active groups, e.g., 2, 5, 6, 9, 22 and 44, were introduced around 7403, we conclude that there is a decrease in the need for new groups and for further redevelopments of existing groups. Thus, both ways, the system is settling down but that may not remain so indefinitely.

4.6.6 Analysis in the time domain

4.6.6.1 Analysis of group 5

We started with the time domain and analysing group 5, G5, since it was at the top of Table 28, belongs to cluster B (potentially the most interesting cluster), has a fairly uniform daf6 spread over time and has already been studied in Section 4.3.

4.6.6.1.1 Choosing a Δt

After experimenting with several different weights to smooth the daf6 time plots of several groups, we came to the conclusion that a Δt of 3 to 6 months for comparing the distributions would be best in smoothing out all the random fluctuation while preserving the basic characteristics.

4.6.6.1.2 Choosing the variables to explore

For the time domain analysis, we discarded ATTEMPTS, WD, BUGREP, REPLACES and DEPENDENCIES since each of these is dominated by a single category, that of the norm, and, with a large number of time periods, there would have been too few daf6's in the other categories for any meaningful results. We were, therefore, left with these 9: INIT, MODULE, MTHS, UTIL, CAT, MODE, VERSION, NV and TIMES.

4.6.6.1.3 Analysing the trend in each category of daf6's

After constructing and examining the relevant frequency tables (one per variable per time period), we extracted, from them, data relating to the proportion of daf6's in each category to find answers to questions such as: Is the proportion of implementation bugs or that of optional reps each increasing over time (perhaps at the detriment of enhancements or diagnostic reps)? Is that of daf6's which affect only 1 release version decreasing? The usual techniques of polynomial fits and multiple regression fits are applied to the set of relative frequencies for each of the categories.

Some typical results are in Table 29. As usual, we note that R^2 is small.

4.6.6.1.4 Analysing the evolution of the daf6 distribution and of its diversity for each variable

We then turned our attention to studying the evolution, for each variable, of all its categories simultaneously. The initial analysis consisted in using the χ^2 test to blanket test each variable's different distributions. The result, in each case, had been found to be significant. The next analysis involved 2 stages:-

- 1) For any given period, measuring the spread (diversity) of the daf6's among the various categories.
- 2) Applying our standard method of polynomial and regression fits to the sequence of such results.

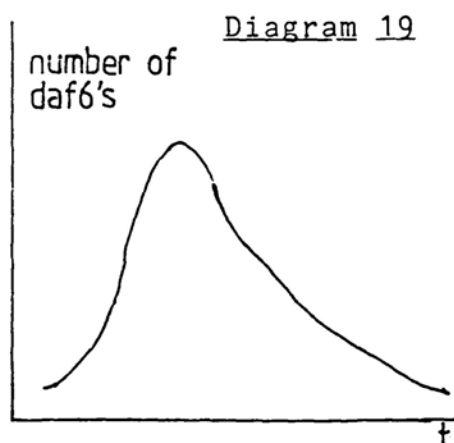
The rough 80-20 law which exists for each variable (as so often found for the other data, too) translates into a negative exponential distribution for those variables which were scored on a ratio scale (MTHS, NV and TIMES) and could, therefore, be characterised by a single parameter, viz. the mean. Since a small mean indicates a sharp fall from frequencies of low to those of large values, hence a small spread, while a large one denotes a flatter fall, hence a larger spread, the mean seems to be an adequate diversity measure in such cases.

However, to our knowledge, there exists no equivalent ideal measure for nominal data. Shannon's diversity index, H , so useful in the field of Information Theory, seems the best there is. That H can adequately measure the spread is shown in Table 30 where 7 arbitrary samples spread over 5 categories are compared. However, H lacks one feature which prevents it from being all-satisfying in our case. This can be readily observed from the table: if, for any sample, the 5 frequencies are reshuffled, the same H would be obtained. In reality, the spread is unaltered yet it is important that, for us, such reshufflings be reflected in the measure. Thus, although we had no choice but to use H , we made sure that the observations do not lose their significance by checking their distributions, too. As expected, there were no extreme cases of reshufflings and it was, therefore, legitimate to use H . For each variable, H , for each month, was found, plotted and polynomial fitted. The best fit to each variable's diversity evolution over time is in Table 31.

That each variable's distribution has not stayed the same can quite easily be explained. For INIT, there exist periods when daf6's originate from only a few individuals and others when they are more evenly distributed among the originators. Concentrating upon the originators rather than the periods

allows us to classify them variously: some are only spuriously active while others are active all the time, some have quite low outputs while others raise quite a large quantity of daf6's. The (slight) decreasing overall diversities in UTIL, CAT and TIMES, Table 31, can respectively be accounted for by the slightly increasing trends in the proportions of the GenGen type of daf6's at the expense of the other UTIL categories, of those correcting implementation and specification categories of bugs (at the expense of the other CAT categories) and of modules which were changed only once per unit time (at the expense of those changed, say, twice), Table 29. All these results, taken together, imply a simplifying process. The variables NV, VERSION and MTHS, on the other hand, show increasing diversities (up to a certain point only for the latter two, see Table 31). The trends in NV and VERSION are attributable to the shift from daf6's affecting only 1 release version towards those of the "onwards" class and those which affect two versions. This is one area which could provide problems in the future since there will be an increasing need to adapt each change to suit it to each of the proliferating product versions. The trend in MTHS suggests that most daf6's are first time ones rather than repeats of failed corrections and, hence, would again imply a simplifying process.

From the time plots of the various categories of VERSION for G5, Figure 60, one instantly recognises a common behaviour among each release's daf6's: a sharp rise to a peak followed by a less steep fall to low and almost negligible monthly values, spanning a somewhat longer period. Diagrammatically, this can be represented thus:-



Now, if we refer back to the G5 plot in Figure 58, we can unmistakably match the peaks with those in Figure 60 (one for one). It is, thus, clear that we have positively identified the causes of the peaks in G5's daf6's time plot. Examining a few of the other groups confirmed the conclusion. In view of this

result, each cluster type is easily explainable. In fact, the categorisation scheme itself becomes unnecessary. D (no major problems and no redevelopment since first time out) is just a special case of A (problems only during first release and no redevelopment) which is itself a special case of B (redevelopments at different periods). The interpretations to C are as evident as those to the others.

A further observation which one may make is that, if a group is not redeveloped, then its number of daf6's eventually fades to a small, almost uniform, trickle which, for most purposes, can be considered to be negligible. Thus, one may conclude that the system, if left alone in its present state, will present no more problems - whether this means no more bugs or no more perceived bugs is another matter. However, given its mission, it is extremely unlikely that B will stay unchanged for any appreciable length of time.

4.6.6.1.5 Analysis of the degree of association between pairs of the variables

Most of the correlations, contingency coefficients, we obtained were non-significant, Table 36. One would, therefore, dismiss the thought of any consistent, permanent or strong association between any pair of the variables. One would rather think that, between most pairs, any observable association, in case it is significant, is not only weak but of a strictly temporary nature. For example, over a short period of 1.5 years, from the start of 1976, UTIL was positively associated with both CAT and MODE. This was due to quite a few proposals for enhancements (GenOpt Information only enhancement daf6's) that were being made during that period. The initial 1.5 years also saw more enhancement and information only daf6's than expected but, as the early daf6's were uncategorised by UTIL, we could not confirm that they were of the GenOpt UTIL type as well. More important, however, is the interpretation that periodically (at least, for this group) defects (room for improvement) are noticed and proposals for remedy (if not the actual steps themselves) are made.

4.6.6.2 The in-depth analyses of G3 and G6

The results obtained with G5 strongly suggest that the manner of the daf6 distribution per group over time owes more to external influences than is due to any internal or structural property of the group itself. We would then be at liberty to choose any group for comparison with G5. We had to accept that one could not possibly subject all of them to a treatment similar to that inflicted upon G5. But comparing it with only one other group would not provide us with the sort of definite results we are seeking. However, taking 2 further groups would go a long way towards meeting that goal. We preferred them to

come from cluster B once more because:-

- 1) Although the behaviour of the uncategorised daf6's for 2 different groups may exhibit the same patterns, this would not imply that their daf6's would categorise similarly.
- 2) The daf6's in a cluster B group would divide more evenly over time.

G3 and G6 were chosen mainly because they are among the cluster B groups with the most daf6's (and, after experiencing the analysis of G5, we found that we had to have as many daf6's as possible per period).

Using the same Δt , variables and categories as for G5, we arrived at these results. The best time fits to similar categories as those for G5 for both groups for each of the remaining 7 variables are in Table 32 while those to the variable diversity measures are in Table 33. As for G5, all the distributional tests gave significant χ^2 results. The results of the correlations are discussed in the next section.

4.6.6.3 Comparing the results for the 3 groups

A visual comparison of the frequency and/or relative frequency tables, each to each, shows that, all in all and in general terms, there appear to be similar characteristics among the 3 groups (and, by extension, over all the groups). For example, for each, most of the daf6's are of the GenGen utility type while there are very few of the other utility categories except GenOpt, that the most common CAT type is implementation bugs with enhancements and specification bugs lying in second and third place, that most daf6's affect one version only until very recently when the proportion of onwards daf6's has been on the increase, etc.

How the proportion of each category or how the diversity of each variable evolves over time has also been compared and is best summarised in Tables 34 and 35. In each slot, whenever no trend is obvious, a "K" is entered, otherwise a $\nearrow$ indicates a straight line fit with a positive gradient, a $\searrow$ one with a negative gradient, a $\curvearrowright$ is for a quadratic (parabolic) fit (with a maximum) while a $\cup$ is for one with a minimum. Although the behaviours of the groups are not identical in every sense, one sees that the general tendency is to be so. Usually a trend in the diversity of one variable or in the proportion of one category transcends most, if not all, three groups.

One readily observes, from Table 36 which compares the 3 groups' correlation results, that most variables are interdependent. With G3, for periods 5, 6 and 7, we have tentative confirmation that information daf6's tend to be of optional applicability. However, for G6, although, for the last

10 months, most daf6's were of the information type (see in Table 32 that the fit for this category has a positive gradient), yet no significant results are obtained by correlating MODE with either UTIL or GAT (cf. G5). Closer inspection of the G6 data shows that these information daf6's refer to bugs (which are not of optional applicability), more probably a large bug which affects several modules. Thus, information daf6's also serve to describe collective module defects which, because of their complexity, may not be immediately dealt with.

4.6.7 Analysis in the release domain

4.6.7.1 Analysis of the variables forsaken in the time domain analysis, with the emphasis on ATTEMPTS

There being insufficient data for WD, BUGREP, REPLACES and DEPENDENCIES to reach any conclusive results, we do not repeat the details here. Tables 37-39 give each group's daf6 distribution by ATTEMPTS from one release to the next. The χ^2 test shows that, for each group, the distribution has not been stable over the releases. Exhaustive comparisons of each group's distributions would require 21 pairings per group. However, the workload can be substantially reduced by concentrating upon our main interest which is to find out whether the distribution has been changing from one release to the next. The number of pairings per group then reduces to 6.

From the results in Table 40, it seems that, while the distribution of G3's daf6's by ATTEMPTS appears to remain stable from 5X10 to 5X32, that of G5 has appreciably changed, notably during each of releases 5X23 and 5X27. G6 has been discounted as there were only two releases, 5X10 and 5X23, where ATTEMPTS greater than 1 was substantial for it.

By grouping and χ^2 testing the corresponding columns of Tables 37-39, the 3 groups' daf6's distributions by ATTEMPTS during each release were compared. Table 41 summarises the results. One very quickly observes that, although, over most releases, the distribution over the 3 groups appears to be the same, this is not always true, 5X10 being the most notable exception.

Pairwise testing, Table 42, shows that, the distributions for G5 and G6 are very similar, whatever the release, G3's differing from the others' during 5X10 and 5X18. The conclusion is, therefore, that the groups are mostly, but not always, similarly distributed, with G5 and G6 showing more common behaviour with each other than with G3.

4.6.7.2 Analysis of the MTHS, UTIL, CAT and MODE variables in the release domain

This is a repeat of the work performed in Section 4.6.6 with release replacing time as the domain of the analysis. Some other differences are:-

- 1) As there are only 7 releases (6 for G3 and very few data for 5X32 for any group), it has not been possible to fit polynomial and regression models to the evolution of the diversity of each variable or to that of the proportion of daf6's in each of the variables' categories.
- 2) We have felt it unnecessary to attempt pairwise correlations of the variables mainly because of previous non-significant results.
- 3) We have further reduced the variables to be explored to MTHS, UTIL, CAT and MODE.
- 4) Because of these substantial reductions, it becomes feasible to fully tackle the group to group comparisons, which, so far, has been performed for ATTEMPTS only and to proceed along with further analyses similar to those performed with ATTEMPTS.

4.6.7.2.1 Analysing the diversity of each variable

The measured diversity values are in Table 43. It is difficult to say whether the trends observed in Section 4.6.6.3 are confirmed by the sequence of values down each column for a given (group,variable) pair, although there is some corroboration of previous results, e.g., MTHS for G5 and G6, UTIL for G3 and G6 and CAT for G3.

Concordance among the diversities would occur if a small/large diversity in one variable can be associated with small/large diversities in the others. Kendall's coefficient of concordance [163] allowed us to investigate whether, for each group, there is any concordance among the diversities of the variables, whatever the release. Then, for any group, should there have been concordance, one would need to only concentrate upon comparing only one, any one, variable's diversities over the releases, to investigate evolution. If, for that variable, say, the ordering $R_1 < R_j < R_k < \dots$ is obtained and the ordering is consistent, whatever group is taken, then this provides a way of comparing releases. Unfortunately, for each of our 3 groups, the evidence is against the existence of concordance among the diversities of the variables, whatever the release.

The same method may be used for the orthogonal analysis, i.e. whether over a given release, there is any concordance among the diversities of the variables of each of several groups. If this was so, then, during a particular release, the diversities of the variables for group i would all be consistently less than those for group j, etc., allowing the groups to be ordered thus: $G_i < G_j < G_k < \dots$. Should the ordering be consistently the same for all the releases, then this may provide a way of

comparing the groups. Unfortunately, there were insufficient data for formal tests, yet judging from the data, the variables, although not necessarily concordant, for G6 during all releases, but more strikingly during 5X10 and 5X23, seem to be less diverse than for the other groups. G5 is possibly the next least diverse.

Note We have been careful not to claim that increasing diversity denotes increasing difficulty or complexity, although this may just be the case (see Section 4.2.4.1 for a similar note of caution).

4.6.7.2.2 Testing for pairwise correlation among the diversities of the variables

Despite the lack of general concordance among the diversities of the variables, it is possible that, at least, some of them may be related in some sense or the other. We have explored such possible associations using Kendall's r [163] but the results overwhelmingly show that, in general, there is no association between the diversities of any pair of variables, whatever the release.

4.6.7.2.3 For each group, the evolution of the distribution by each variable from one release to another

Focusing on the distributions, we started by constructing, for each group and each release, its daf6 distribution by each variable. Then each group's distributions by any variable were blanket tested. The results were all significant whatever the group or variable examined. Pursuing the analysis with pairwise tests of succeeding releases' distributions, we have the results shown in Table 44. Apart from MTHS, G6 shows greater stability in the distribution by each variable from one release to the next than the other two groups, although, by no means, is the distribution the same over all the releases. G3 and G5 show that, for certain variables, e.g., MTHS, the distribution changes radically from one release to the next while, in other cases, e.g., CAT for G5 and MODE for G3, only occasionally. However, all in all, G3 shows greater instability than G5. All these results are, of course, related to those already obtained with the diversity concordance analysis.

4.6.7.2.4 For each release, the group to group comparisons of the distribution by each variable

The results of the release to release comparison of the groups on all of their variables are in Table 45. We then proceed to obtain Table 46 (the orthogonal equivalent of Table 44) which shows the results when the groups are taken two at a time. Thus, G3's distributions are mostly significantly different from those of G6 while G5's distribution sometimes resembles

that of G3, sometimes that of G6 and still at other times resembles neither or both. This seems to confirm our previous results.

However, it is perhaps more productive to examine how each group's distribution varies from one release to the next. The more striking feature seems the great changeability in the distribution and this, itself, depends upon the variable examined. For example, with CAT, the distributions tend to resemble each other while, with MTHS, they tend to differ, whatever the release. Furthermore, for other variables, the release itself seems to exert some influence upon the distribution. For example, for UTIL, the distributions show great diversity of behaviour whatever the release yet, in 5X27, they seem to resemble one another.

Thus, it may be true to generalise in saying that most groups have basically the same distribution which is, however, subject to external and internal influences differing from group to group. The more stable groups would have a distribution which does not change often from release to release. Unfortunately, the releases themselves may be disruptive to the whole system and a group may be stable in terms of variable X but quite unstable in terms of variable Y.

4.6.8 Generalising the main results of the daf6 analysis

On ending, it is perhaps necessary to briefly recapitulate and generalise the results in as simple terms as possible. The emphasis of the analysis of the effects of certain observable influences, the properties as represented by the variables we analysed, can be put on any of 3 different dimensions: the influences themselves, the different parts of the system and the releases affected.

4.6.8.1 The properties

Any activity on a system can be categorised according to any of several types of influences which are being exerted. They may be compared against one another in the same part of the system or each may be compared from one release to another (the dynamic part of the analysis), given the same part, or from one part to another (the static part of the analysis), given a release. We found that, for each part, during any release, the properties bear little relationships to one another. Some may be large-valued while others are small-valued. Some may be spread out over a large range of values while others are less diverse. In addition to their values and diversities not being concordant, these do not change from one release to the next in a very orchestrated fashion, e.g., while for one variable, its value and/or diversity may increase from one release to the next, for another, it may decrease but, the next time around,

their behaviours may interchange. However, in most cases, a property is comparable from one part to the next and tends to evolve in much the same manner over the releases, for all the parts, indicating that, in a general sense, the parts are all similarly affected by external influences.

We also found that some properties were very volatile, changing all the time and from part to part, while others may hardly change either from release to release or from part to part. Other findings concerning the properties are that some may show more similar behaviours over the parts than others and that some are more stable from one release to the next. Thus, the external influences themselves vary in intensity and in stability although these are also conditioned by the parts and releases affected.

4.6.8.2 The parts

In general, one finds that there are some parts over which the values and diversities of each property resemble each other more than over other parts. There are also certain parts over which the situation is generally simpler than over others. Then, there are parts over which the properties tend to remain stable while others over which they are permanently changing. However, as before, these findings are subjected to the influences of the releases and the properties, too, so much so that they may sometimes be completely reversed.

4.6.8.3 The releases

About the same sort of findings concerning the parts apply. Over certain releases, some variables and parts are less diverse and resemble each other than over others so that there are, in effect, quieter and more turbulent releases, too.

What we have, above, is just a brief summary of the most salient of the behaviours that we had been able to note. Although the variations are more limited than those which would have been obtained if all sorts of combinations were allowed, yet they are sufficiently rich to suggest how complex the problem really is and that it may well be impossible to attempt solving it completely. One may perhaps reduce it by eliminating one or more of the dimensions. For example, a global view (i.e. studying the whole system instead of its individual parts) will simplify matters considerably. Although for the other data, the BD modifications, the daf2's, the software notices and the bug reports, we did combine the global with the sub-level approach, here, our concentration upon the sub-level was intentional. Indeed, what we had been trying to obtain from the daf6's provided the empirical evidence for what formed quite a substantial part of the theory exposed in Chapter 8.

4.7 SUMMARY OF CHAPTER 4

(Note The same remarks as in Section 3.6 apply here.)

Like BD, B can change through the modifications of existing components as well as through the additions of new or the removals of redundant ones. The effect of such changes on size is mostly an increase. The contribution of local enhancements and fixes to each component upon the global system size is quite modest, however. Indeed, most of the change in size (mostly growth) is via grand scale major changes whereby whole sections of code, such as subsystems, get chucked off or added in (mostly the latter). Dropping, replacing or adding components on an individual basis is a rather rare activity.

It is not only the average size in the number of instructions of each component that is increasing. We have found that, for most components, the attribute values that we were able to measure show general increases whenever the component is changed. However, as far as the whole population is concerned, the average value for each attribute shows a much less perceptible increase. Some attributes show no increase at all while, in a few cases, there may even be a decrease. This is due to a combination of factors:-

- 1) The large majority of the components which remain unchanged from one release to the next and are, perhaps, unchanging over all the releases, too.
- 2) The components which are newly introduced to the latest system version have, until now, been exhibiting attribute values very similar in size to those of the unchanging ones.
- 3) The removed components are probably large-valued ones.
- 4) The increases for the changing components, although genuine enough, are really very slight.

It may seem plausible to assume positive correlation between largeness and complexity and to associate, for each component, its increasing number of changes and, hence, its increasing attribute values with increasing unstructuredness and complexity. But since examples of small programs which are more complex than larger ones abound and since it is easy to decrease the attribute values of a changing component, if that was the aim of the change, then one should beware about finding more in these results than they contain. They only give some approximate idea of possible relative complexities thus: in a general sense, the new components are probably of the same complexity as the unchanged and unchanging ones while those which had been removed were probably more complex than the remaining ones and those which are changing are increasing in complexity although, as a whole, due to the combined effects of the different types of components, the complexity of the system

may not have changed very much from one release to the other.

The changes in attribute values which we have found cannot be taken to reflect faithfully the changes in the complexity of the system. What we have found is, at best, a possible complexity indicator which is incomplete in itself and which needs to be supported by further indicators. Thus, following the theorising over the ageing and the rejuvenation processes of the population, it will not be sufficient if we just attempt to replace large-valued components with smaller-valued ones, whether on a one-to-one basis or in bulk, as a means to combat senescence and to prevent the ultimate decay and unmanageability of the system. More about the theory and certain shortcomings can be found in Section 4.2.4.4.

Although we have investigated the evolution of several attributes, the important aspect regarding complexity and complexity evolution should not be too much concerned with the number of properties which take on increasingly large values. Rather, one would feel much more comfortable if it was possible to identify and concentrate upon those pertinent properties that can be firmly associated with complexity. Of those we have used, PATCHES, by intuition, seems to be a good choice, although DAF2 may be just as acceptable. Since correlation analysis shows that most of the other attributes are positively correlated with it, then one may interpret this as meaning that the increasing complexity in a component causes its attribute values to increase, too. Modelling PATCHES (and DAF2) in terms of the others led to linear and non-linear regression models whose explanatory variables are preponderantly of the communications variety and whose R^2 's are of the order of 80%. Compared with the BD models and those obtained by other workers, these are eminently satisfactory but that there remained quite an appreciable amount of unexplained variation led to 3 not mutually exclusive conclusions:-

- 1) There may be other meaningful attributes which had not been measured (unmeasurable?).
- 2) There may be exogenous forces which complement the influence of the structural properties.
- 3) The available measures may not represent their corresponding structure properties as well as they might.

We attempted to create more sophisticated models by using increments, whether actual or absolute, and rates of change in the attributes as the variables but, although the general impression is that better models are obtained, yet the variables from one model to another are highly inconsistent. This, indeed, allowed us to deduce that maybe B and software systems, in general, are of the moderately to the severely stochastic class of Sutherland's types of systems. Further work which involved the comparison of the corresponding models over

two different releases for each of several sets of components reinforced the conclusions.

Passing on to the analysis of the number of daf2's/month for each subsystem, we found that, as time series, they exhibited no patterns worthy of attention. Like the other measures examined in Chapters 1 and 3, they mostly behave as random variables. However, they seem to be exponentially rather than normally distributed but this should not be worrying. The global distribution (not shown), indeed, looks normal and, in that sense, concords with previous results. The explanation of the exponential distribution being more common at the sub-levels and the normal one at the global level is, of course, found in the Central Limit Theorem.

Rearranging the data so as to investigate the number of daf2's to a typical holon over its lifetime led to an exponentially decreasing model over each of its different phases. A similar model was obtained for all the subsystems although the rates of fall differ from one to another and from phase to phase, too. The model, therefore, shows that, despite certain differences in rates and quantity, which may be due to both the properties of the subsystems themselves and the different requirements to which they are subjected by external sources, the number of daf2's per component tends to vanish, if it is not redeveloped, and that this is not due to increasing complexity. Rather, it is just symptomatic of the settling-in process. Otherwise, how can we account for the recrudescence of new phases from time to time?

The next type of data which we analysed were the software notices. Although their global weekly time series, like other series, showed no patterns of any consequence, yet one could fit a good AR(3) model to it. The decomposition which we performed with the software notices was to separate them by the release affected. But, because of some ambiguities and of the possible imprecision of the separation process, we refrained from attempting too rigorous analyses on the new series. Instead, we concentrated upon qualitative observations from the releases' time plots. These led to the construction of a qualitative model of the number of software notices as a function of time to each release over its life history. The model is bell-shaped showing a build-up to a peak and a decline thenceforth. Superimposed on that main shape are alternating peaks and troughs which were shot through and through with random fluctuations. The interpretations are obvious, Section 4.4.4. Other results are that when a defect is found in a version, it seems to be relevant to the other versions, too, and that the number of software notices and its variability appears to be decreasing from one version to the next.

We then analysed the bug reports. Categorising them according

to each of 7 different variables, one finds that, in each case, the constitution of the population has not been stable over time; at different moments, the attention was on different components and versions of the system and so on. Correlating the variables with each other resulted in the following deductions:-

- 1) Different people have different working habits; they assign priorities differently and are worse at making certain types of errors than others.
- 2) At any time, each team is involved with a specific task: the development or testing of given component(s) within specific version(s) and so on.

Further analysis concerning the response time to a bug report showed that, for any priority type, the distribution, like that of the global data, has not been stable and that, apart from the priority A ones, the response times to each priority type of bug reports have always been of comparable magnitudes, despite the existence of different targets. It also showed that, as a whole, the response time has been slightly decreasing. This is yet another evidence against the increasing complexity idea. That the proportion of bug reports which has been handled more than once has not increased over time seems to support the finding.

When treated as time series, neither the bug reports, REP, nor the bug responses, RES, showed noticeable patterns, except slight increasing trends, amidst the marked stochasticities, the trend in RES being slightly larger than that in REP. Early on, the REP rate exceeded that of RES so that the outstanding bug report queue lengthened but the situation has since been reversed and the queue length has been steadily decreasing. There is, as yet, no steady state and it seems that the present set-up of responders is still capable of dealing with some extra load without any undue hardship. In REP, there exists a weekly effect of more bug reports on Mondays than on any other day of the week. This could be because people tend to carry on reviewing their problems, consciously or subconsciously, over the sabbatical weekends, resulting in a spate of activity on the first day of work resumption. Why the same effect had not been detected in RES, too, is probably due to the fact that, although the responders may behave likewise, there are too few of them, when compared to the responders, for the add-on effect, which tends to smooth out the random fluctuations and to amplify any slightly deterministic element among several simultaneous sources, to be effective. A similar add-on effect explains the greater stability of the REP series. Since there is a vast number of REP sources, each of these contributes only minimally to the global behaviour. Indeed, adding or removing a few REP sources has a negligible effect upon the latter. However, for RES, it is an entirely different matter: as there

are much fewer RES sources, each contributes appreciably to the global behaviour and adding or removing one of them has a marked influence upon the latter. Thereby lies the explanation of dynamics, inertia or even statistical invariance attributed to large programs, although largeness should not be taken to mean actual physical sizes but should, instead, be a function of the multiplicity of influences. However, at the level we are dealing with, one, usually, implies the other.

When considered on its own, each series appears to be a weekly Markov process and, as one might expect, any REP value has a certain amount of influence upon the RES values the same and/or 1, 2, 3, 4, 7 and 8 days later. However, because of the ever-decreasing queue length, the non-stationary response time distribution, the marked stochasticities in the process and the not-easily-characterisable nature of the bug report arrival process, a queuing model for the reporting-responding process seems unsuitable. A Box-Jenkins transfer function model seems more appropriate but could not be undertaken within the scope of this study.

The last type of data which we examined were the daf6's. Because there were too much data available, too many groups, too many variables and all sorts of possible agglomeration periods for an investigation of the evolution of the daf6's behaviour over time, we had to severely restrict the number of elements investigated along practically any dimension, specially as the two main tasks which we had to do:-

- 1) Investigating how the composition into the various categories of a variable varies for the daf6's of one part of the system to those of the next (given a certain time period) - the static part of the analysis.
- 2) Investigating how the composition of the daf6's into the various categories of a variable varies over time (given a certain part of the system) - the dynamic part of the analysis.

are, in themselves, quite complex and extremely labour intensive. Besides, other tasks such as exploring the degree of association between different pairs of the variables and the concordance among the variables and among their diversities, too, were envisaged. There, also, loomed the spectre of doubling whatever work was performed when the time domain dimension would have been replaced by that of pseudo-time (or release), another essential part of the analysis we wanted to perform.

As a start, we concentrated upon the dynamic part of the study with the analysis of a single group. Our choice was G5, the largest and most frequently daf6'ed. The aggregation period problem was solved on the basis of the shortest one, among

several we experimented with, which provided the smoothest time plots. Because we felt that they were of sufficient importance, most of the variables were retained.

Predictably, the (roughly exponential) distribution of the daf6's for G5 has been stable over time by none of the variables. This being so, we penetrated below the global level to study the evolutionary behaviour of each variable's different categories. One could, indeed, detect quite clear increasing trends in the proportions of daf6's in some categories, such trends being compensated by decreasing trends in some of the others. Since, for each variable, the daf6's shift from one category to another, we wanted to check whether these were continually from the more frequently occurring categories to the less frequently occurring ones (hence, an increasing diversity, as the distribution is mainly negatively exponential) or vice versa. This was accomplished by measuring each variable's mean (for those measured on a ratio scale) or Shannon's H (for nominal data) for each period and, then, by looking for time trends in these. Some of the variables show stochastic behaviour, e.g., INIT where the originators can be spuriously or constantly active with varying degrees of intensity. Others, such as UTIL, CAT and TIMES show weak decreasing diversities over time. This is because the proportions of the most frequently occurring categories GenGen (for UTIL), implementation and specification bugs (for CAT) and modules which were changed only once (for TIMES) were increasing at the expense of most of the other categories of the variable concerned. Taken together, these results hint at a simplifying process rather than at one which is getting more complex. The increasing diversity in MTHS led to the same conclusion although that in NV (or VERSION) which is due to a shift from the daf6's which affect only one version to those affecting several should be carefully monitored. This is one area which might spell future trouble, although the anticipated increasing difficulty to effect a change would not be due to the increasing complexity of the system itself. Rather, it will be caused by the need to coordinate the change over each of the ever-increasing number of versions that would need to be supported.

Although our analysis shows that, over certain short periods, there were periodic increases in the number of daf6's which were used to suggest possible improvements, by and large, the variables show no or weak and strictly temporary associations among themselves.

Further work concerning the uncategorised daf6's of the groups led to the association of each new redevelopment phase which a group has undergone with a recrudescence of daf6's for that group. One finds that those that have had most redevelopments and daf6's are precisely those which are the largest, the most

functionally important and those which are situated at the very heart of the system, too. It was also evident that, should a group not be redeveloped, then its daf6's tend to eventually vanish, this behaviour being very similar to that of the modifications to a BD micro and to that of the daf2's, too.

Obviously, the groups that have been redeveloped often and that provide the most daf6's, at the same time, are those which are the most interesting in the context of this analysis. We selected two of them, G3 and G6, for a treatment similar to that inflicted upon G5 and for the static part of the analysis as well as for completing this investigation by passing on to the release domain analysis. All in all, there appear to be similar general characteristics among the 3 groups (and, by extension, over the other groups). First, their daf6's categorise to comparable proportions. Then, although the behaviours of the groups concerning the evolution of the daf6 categories and of the variable diversities are not always identical in every sense, yet the trends tend to be replicated. Finally, the correlation results for these two groups were comparable to those of G5. Thus, we may conclude that the impacts of the daf6's on different parts of the system tend to be similar at each point in time and to evolve similarly, too, although there are variations which can be to a great or to a small extent, depending upon both the variables and the parts concerned. For example, there exists the odd variable which may exhibit completely different behaviours over each part. There also exist certain parts which are, more or less, influenced similarly and others which are not. And then, there are parts over which the influences are less felt than over others.

Analysis in the release domain basically confirmed the same impressions. However, a third element is added to complicate matters. It is that the categorisation of the daf6's is affected by the release, too, although in no systematic fashion. This is easy to understand; during a given release a lot of work of a certain category may need to be performed while during another the emphasis may be on other types of work. Then, although a variable may affect different parts similarly, its own behaviour is partly conditioned and affected by the releases involved. Also, although, most of the time, two parts may exhibit closer behaviours with each other than with another one, whatever the variables, yet, due to certain requirements in each release, only one of the two similar parts may be affected and, this, in a similar fashion to the third. The generalisation of these results are not repeated here but can be recapitulated by re-reading Section 4.6.8.

CHAPTER 5

POOLING THE ANALYSES OF THE TWO SYSTEMS STUDIED

5.1 INTRODUCTION

In the Preface, we have already explained why the emphasis in Chapters 3 and 4 was on summarising the results of our analyses in the sequence they were performed and why most of the interpretations of the findings and most of the conclusions had been left to this chapter. It was that, because many of the findings led to similar interpretations and conclusions, the arguments in those two chapters were becoming too repetitive. By pooling the findings together in a single place, instead, not only would we avoid the repetitions but each argument would be strengthened by virtue of the fact that similar findings would support each other. Furthermore, since some results, specially when they could be interpreted in several ways, required confirmation or support from further results, one finds that the results can be seen in better perspective and it becomes easier to fit the pieces of the jigsaw after the studies have been completed. The global view becomes much clearer, too.

5.2 THE STRUCTURAL CHARACTERISTICS5.2.1 The 20-80 law

Whenever any sample of BD or B elements, e.g., modules, micros or holons, was taken, the predominant shape of the frequency distribution by any structural property (size, number of calls, hierarchical level) is always found to be exponential with a few tending towards the normal shape, Sections 3.4.1 and 4.2.2. This rough exponential shape is frequently encountered in many fields of human endeavour and is known, in social parlance, as the 20-80 law, e.g., most (80%) of a nation's wealth is held by a few people (20%) and vice versa. In the software field, many workers, e.g., Knuth [90], Anderson and Sargent [4] and Samet [154], among others, and closer to us, Endres [59], Elshoff [57] and, closest, Benyon-Tinker [23] have all come across it. Why such a shape should be preponderant is probably due to the fact that software are artifacts which, because they are humanly engineered, tend, in their majority, to reflect their creators' needs of making things as simple as possible. However, as we have seen, things created simple do not necessarily stay simple, Section 4.2.3, unless they are left alone. Indeed, although humans may have the (conscious or unconscious) gift to produce things which are largely simple, it is their inability of devising ways of keeping these simple when changed and of adequately managing the changes which causes all the problems associated with the maintenance of large evolving software. However, it is fairly obvious, from

the 20-80 law which our results suggest, that most components (the 80% which account for the 20% of the activity or property being studied) are of the simple category and can safely be excluded from any monitoring exercise. Of course, there is every possibility that, because of evolution, some of these will overspill into the other category in the future but this is another story. Then, we can perhaps hope that, just as Knuth's results have influenced the construction of compilers, ours may persuade software management to mostly concern themselves with only those few components that exhibit the largest values of the attribute they wish to control.

5.2.2 The properties are not necessarily concordant

However, although most components may be simple according to each property examined, yet there is no concordance among the properties, i.e., for any component, the values of all its attributes are not of the same magnitude. Thus, a component is neither uniformly large nor uniformly small over all of its properties though there do exist certain definite associations. For example, a BD micro which calls many others is almost always large-sized, Section 3.4.3.1, and a large-sized holon usually has many data elements and issues many calls to other holons, Section 4.2.5.1. These are, in truth, much more realistic than an expectation of all the properties being concordant specially as largeness in one property may be associated with smallness in another, e.g., a BD micro which calls many others is itself called by only a few ones. It must be noted that we have, on purpose, avoided writing about any large property which caused largeness/smallness in another one. This is really difficult to resolve; probably, it is the requirements of a component's functions which cause it to have, for some properties, certain values which are associated with one another.

5.2.3 The properties of the partitions

The B holons being too numerous, it was impossible to attempt, as for BD, to characterise the whole system. Consequently, only certain holon sets, where each set represents a larger unit, the subsystem, were investigated, Section 4.2.1. Although still predominantly exponentially distributed, these sets show great variability in their properties, Section 4.2.5. Some may be large in one property but smaller in another and, where a set is smaller than another in any particular property, that relationship may not hold for another property, i.e. the ordering of the sets is not necessarily consistent over all properties nor is that of the properties consistent for all sets. This seems relatively easy to understand. Just like the individual components, different subsystems have different functions to perform so that, in some, there may be certain features, and, hence, attributes that are well developed while,

in others, the emphasis is upon other features. Undoubtedly, other factors such as the fact that the components may have been developed by different persons/teams whose products are not usually similar contribute significantly to the differences, too.

The BD micros, when partitioned by the actual/thrown away criterion or by the frequency of change, also led to the same sort of results, Section 3.4.1. Here, however, the interpretations are different and, perhaps, more interesting. People are probably cautious about excising components whose interactions with the remaining code may not be well understood. It is, also, probably true to say that components which are large, multi-functioned and complex do lead to frequent modifications while those which are small and simple do not invite many changes. The conclusion may appear to be that one should strive for small and simple components while constructing software but whether those which sit at the highest hierarchical levels (as most of the large and complex ones do) can be made small and simple is another matter. It would also not do if we were to replace a few large ones by quite a number of smaller ones, each of which may be infrequently changed, but whose total number of changes may exceed that expected from the large components. Thus, the problem is not really as simple as it may look.

The meaningfulness of the results which we have obtained by considering various subsets of each of the populations available as well as these themselves makes a strong case for the study of the "trees" at the same time as that of the "forest". Most of the relevant arguments have already been exposed in Section 3.4.1.2 and it may be useful to turn back the pages to refresh the memory. However, we wish to add here, now that we may call upon the 20-80 law, that although a priori there may exist no basis upon which to select a partitioning criterion and particular partitioned parts for analysis, yet, because of that law, one would quite easily be able to sort out the interesting cases from the obviously useless ones. The problem of the unlimited number of subsets which may be investigated, although not solved, will be simplified, too, although how many of the remaining subsets may be investigated, as we have said before, will largely be determined by the resources that may be spared.

One further usefulness of the sub-level approach is that it allows the global behaviour to be inferred from those of the subsets whenever the former could not be investigated because of largeness as in the case of B. Assuming that most of the B subsystems, like those we have studied, have properties which are approximately exponentially distributed, then the mathematical tractability and reproductivity of that distribution tell us that the mixture of such distributions, i.e. the distribu-

tion of the global system itself, is exponentially distributed [66]. The deviations from the exponential to the normal distribution and the ultimate tendency to the latter, are explained by the Central Limit Theorem which states that the sum of any distributions is asymptotically normal.

5.3 THE EVOLUTION OF THE STRUCTURE CHARACTERISTICS OVER TIME

Structure evolution investigation can be achieved by repeating the work described in Section 5.2 either for each of various points in time or over successive releases so that successive snapshots of the system's component distribution by each of its structural properties could be taken and compared. Because of the scarcity of data, only two snapshots for B could be taken, Section 4.2.2.

We found that in terms of all the holons we have examined there was little concordance among the attributes' behaviours; some showed generally larger mean values (we really wish to say: significantly different distributions characterised by larger means) while others do the opposite as we move from the previous to the more recent release, Section 4.2.3.

Such ambivalent results were probably due to the mixture of different types of holons whose possibly contrary behaviours have tended to nullify each other's influence on the global behaviour. Thus, once again, to unravel the intricacies of the system's behaviour we needed to depart from the holistic view. This was achieved by partitioning the holons into unchanged, changed, removed and new classes. Examining each partitioned set in turn, we found that the new and unchanged holons have similar attribute values which are less than those of the changing holons, the latter getting more complex after the change has been effected. (There were too few deleted holons for these to be analysed adequately).

The important question is how general indeed were these results? Undoubtedly, many more than two releases will have to be examined for firm conclusions but, from our own qualitative observations of the holons over their entire history spanning about six releases and from the 20-80 law (in fact 36.5% of the BD micros have never been changed), we are fairly satisfied that the changes are concentrated on to a few and probably the same changing holons which take on increasingly larger structural values (albeit, as our analysis shows, at a slow rate) while there remains a very substantial pool of largely unchanging holons.

We have already considered how the distribution by each of their attributes of the changing holons may evolve into more complex shapes, Section 4.2.3.1, and how these evolutions may be used as possible indicators of increasing complexity in the

holons themselves, Section 4.2.4.1. Our cautionary remarks about the latter may be summarised by saying that one should take care before inferring increasing complexity (whether functional, computational, implementational or psychological) from the increasing trends in the structural attributes. Although this may probably be the case, we need not only to be absolutely clear in our minds that these trends are indeed true manifestations of increasing complexity but also to show that this is demonstrably so.

However, even if, as in all likelihood, increasing attribute trends can be associated with increasing complexity, it seems that this applies only to the changing holons. The global complexity, from our results, does not seem to have been changing markedly in any direction. That the new components are not more complex than the unchanged ones is another indication that the system's overall complexity may not have really increased significantly from its inception. The continuing influx of new and simpler holons, the dampening effect of the large mass of unchanging holons (and the casting off of old and "complex" ones?) all contribute to compensate for the behaviours of the changing holons. One could say that the ageing (meaning getting worn out or more inefficient rather than the mere passage of time) of the whole population due to the behaviour of the changing components is being counter-balanced by the rejuvenation process caused by the influx of new and simpler components and the outflux of the old and worn-out ones. These ideas were treated at great lengths in Sections 4.2.4.3 to 4.2.4.4 although, there and in Sections 6.2 and 6.3.1, we repeatedly stressed that further refinements to these apparently promising ideas were required.

5.4 THE EVOLUTION OF THE ACTIVITY CHARACTERISTICS

Having successively looked at the structural properties and their evolutionary behaviours, it is now time to deal with another major topic. This is the evolutionary behaviours of the activity characteristics. The activities which we have studied are mainly those by the agents directly affecting the system under study and perceived by them as being of sufficient importance to require the setting-up of procedures for generating and carrying them out and the productions of documents for recording them. Such activities are usually associated with changes to the system but, by no means, were we able to study all the activities that cause the system to change. Indeed, in each of the two organisations that produced and maintain the systems we studied, we were unable to obtain data to satisfactorily represent the activities relating to the production of new functions, the major contributor to system growth, as we have seen, Sections 3.3 and 4.2.1. On the other hand, we studied some activities which, although very important, account for little or no direct and immediate change

to the system, e.g., bug responses and software notices.

The ultimate purpose of studying these activities is, of course, to relate their behaviours with those of the changing characteristics of the system itself. But, before attempting such relationships, it is necessary to examine the dynamics of these activities by themselves. We first noted the contrast between the two organisation's reporting and recording activities. They seem to reflect the greater need of the larger organisations and/or of the larger systems for control. The BD activity measures, mc, me, mf, c, e and f, were extracted from a single document while, to represent B's activities, we were forced to work with various types of forms: dafs, software notices, bug reports and bug responses. Since the list is incomplete and there exist some other activities that are not recorded anyway, it was impossible to integrate these and futile to attempt extracting measures similar to those used for BD. We, therefore, dealt with each type of form on its own and extracted whatever data and conclusions we could from it. Even then, the bulkiness of the data represented a formidable problem.

5.4.1 Each activity on its own

Apart from the e series which showed a rather significant increasing trend, the randomness tests we applied to the various BD activity series (whatever the aggregation period), Section 3.2.1, to each B subsystem's daf2 series, Section 4.3.1, and so on, showed that they all exhibited behaviours consistent with that of white noise. In the formal statistical jargon, our results merely show that there is no evidence against the null hypothesis that each series is a random process. We have already mentioned in Section 1.3.3.2 that we would have preferred the null hypothesis itself to be that the series is cyclic or whatever, since, then, we could have asserted that there is evidence against it, should the results have been significant. That, from the existing tests, our non-significant results only allow us to say that there is no evidence against the white noise process is not the same as saying that there is, therefore, evidence against the alternative hypothesis is quite annoying. Add to that the known fact that significance tests can never establish the truth or otherwise of the null hypothesis, then one gets an idea of how frustrating the endeavour really was. However, we had to make the most of what was available.

Thus, despite obtaining so many similar results from the significance tests, we still needed to seek the support of other tools such as correlograms, periodograms and spectra which do not depend upon significance tests and, in that sense, could lead to "stronger" results. These results, Sections 3.2.3 and 4.3.2, overwhelmingly confirmed those inferred from the

significance tests so it would probably be true to say that each of these series does not show any cyclicity (of the up and down type) or seasonality and probably no trend either. As we have said before, these may appear to contradict Lehman's findings. However, we pointed out that Lehman had probably employed the cyclicity term in a much looser sense than with a strict up and down meaning, Sections 1.3.3.2 and 1.3.3.5, and that phenomena observed on a release-to-release basis may not be necessarily observed on a real-time basis, Section 4.6.2. The latter remark indeed applies to all the results included in this section.

One interesting feature concerning quite a few of the B series (several of the subsystem's monthly daf2 series, the weekly software notices, the daily REP and the daily RES series) is the presence of some weak and short-term serial correlations. Since these are positive, they indicate clumping of similar values together rather than cyclicity. This is, as yet, the first indication of some sort of activity "dynamics" which may be present in B.

When REP and RES were aggregated into larger periods, the daily serial dependencies disappeared turning each series into an effective Markov process, Section 4.5.4.1.1. There is a possibility of obtaining similar short-term effects in the other series had they also been analysed on such fine mesh. This was not carried out both for practical reasons and because of the doubtful practical usefulness of such results which merely indicate high responsiveness, Section 4.5.4.1.1.

Work on another front, the Box-Jenkins stochastic models we obtained for me, c and mf, only confirmed the impression of no apparent effect except white noise, Section 3.2.3.1. That e is best fitted with an IMA model of order 1 seems to indicate that any trend should be viewed as a stochastic trend (i.e. with random changes in level) rather than as a deterministic trend. But it is really the interpretation, from the models, that the greater portion of each "shock", if not all of it, being dissipated rather than absorbed to have a lasting influence that is more important. It may be that this behaviour, which implies that the system does not learn from past events is peculiar to BD only and/or to systems which have similar properties to it. One can only speculate whether it is caused by BD's special architecture, its smallness and/or that of its environment, among other things. This particular problem illustrates nicely the enormity of the task that still needs to be accomplished.

AS with the BD series, we could only afford to model a few of the B series (weekly software notices, both daily and weekly REP and RES). The models obtained were definitely more adequate and, very interestingly, are autoregressive with positive

coefficients, in accordance with the correlograms results and interpretation of positive activity dynamics. Two explanations are offered for the difference in types of models we obtained for the two systems. The first, which rests upon the usage of finer meshes for the B series, seems unlikely since some monthly daf2 series show significant autocorrelations, too, Section 4.3.2. The second is that larger systems (and B is larger than BD in every sense of the word) could exhibit some sort of dynamics while smaller ones do not. This seems more likely since not only has the whole concept of ED been based upon the dynamics of systems much larger than BD but our own (too brief) look at another system, CCSS, again, implied the existence of dynamics and inertia which were stronger than those in BD, Section 3.2.5.

Why larger systems should exhibit more dynamics and inertia than smaller ones was attributed, in Section 4.5.4.1.2, to the fact that a large system's activity, as opposed to those of the smaller systems, usually expresses the sum of the efforts of a multitude of sources which each contributes an infinitesimal amount to the total. (The more such sources there are, the greater will be the dynamics (and inertia) observed and the less easy will it be to deviate the system's behaviour from its settled patterns.) Thus, adding or removing one or even several of the sources do little to affect the settled patterns whether in motion (dynamics) or static (inertia), specially as stochasticity has an important role to play as well. As it is in the power of management to only make certain adjustments here and there but not to drastically modify the whole set-up; it is not surprising to find, as Lehman has observed so many times, that, despite all efforts, management's control over the system's future behaviour is not great, specially as the application of these efforts may not have been consistently in the same direction to either accelerate the dynamics or slow it down. However, this is quite understandable if we realise that the managements were largely unaware of the existence of dynamics in their systems and of their causes. Before passing on to other effects which the addition of different sources' outputs may cause, let us again repeat that the dynamics of a system depend more upon the multiplicity of these sources than upon the system's actual physical size, but, of course, these are related to one another.

Another aspect of the analysis involves choosing the best polynomial fit to each of the series we have discussed. In every case, the mean of the series was the best choice except mf, e and f for BD, Section 3.2.2, and software notices and daf6's for B, Sections 4.4.2 and 4.6.5. The R^2 's for these were all low, so low in some cases, e.g., mf and f, that the trends could be considered negligible, as in fact treated by the other tests. Then, what is the chief feature of these series if they exhibit neither cyclicities nor seasonalities and if trends are

either non-existent or weak? The answer, inevitably, has to be that the dominant effect, if not the only effect, is randomness. This discovery has, indeed, been one of the highlights of this research.

Most previous researchers have failed to recognise its importance, neglected to study it or could not cope with a "randomly discrete set of events in an uncertain environment" [133]. According to Needham and Aron [133], this is the main reason why the work of the theorist had had minimal impact on that of the software engineering practitioner.

More succinctly, software activities may effectively be considered as random variables, whose value y_t at time t can be represented by

$$y_t = \mu + a_t$$

where μ is the expected value and a_t is the noise effect. It can also be shown that the cumulative form of that equation is a straight line, whatever the variance of a_t , Section 1.3.3.1.

An attempt at identifying the characteristic distributions of these random variables (and, hence, of the noise too) showed that all BD variables except me could be well fitted with a normal distribution, Section 3.2.4. However, the B series were less well behaved, their distributional shapes ranging variously from the exponential to the normal. Nevertheless, whatever the probability density function, its properties implicitly include the concepts of invariance and critical growth mass and the fourth and fifth laws, too, Section 1.3.3.5.

Now that we have seen that white noise characteristics are exhibited by most of the variables we have studied, even more so by those of the smaller BD system, it is appropriate to explain why randomness should be the predominant feature which we have observed.

Because of the relative simplicity associated with the BD system, it would probably have been easier to influence its behaviour in one way or another so that, unlike the other systems, it shows little inclinations to exhibit dynamical trends. That, despite that property, there has been no inclination to cause the system to move in any intended direction seems to be due to the more or less steady state which the system has been in for quite some time. There has been no major upheavals. Instead, there has been a continuous flow of new components, each of which gradually tends to become enhancement and fix free, Section 3.5.2.2. Although there is some degree of determinism in them, the behaviours of the components or modules, Section 3.5.2.2, and sub-components or

micros, Section 3.5.2.4, are not reflected at the global level. This is due to the staggering of their introductions into the system at not-too-regular intervals as well as to the presence of the inherent stochasticities.

The same irregular staggering and inherent fluctuations led to the mostly indeterminate behaviour at the global level of most of the activities of B, too. Thus, although we were able to obtain models that show an unmistakable decreasing trend from a (sudden or rapidly reached) peak for the daf2's per holon, Section 4.3.4, for the software notices per release, Section 4.4.4, for the daf6's per group per phase, Section 4.6.6.1.4, yet the daf2's per subsystem, Section 4.3.1, the software notices for the whole system, Section 4.4.2, and the daf6's for the whole system, Section 4.6.5, show mostly stochasticity.

Although it is obvious that the more there are components which are introduced in random sequence, the more random the global effect becomes, yet it is the associated idea of the values tending to resemble one another, i.e. of invariance or of inertia that we shall stress. We previously asserted that the larger the system (or more exactly, the more there are contributing parts), the more dynamics, if in "motion", or inertia, if at "rest", it exhibits. Assume that, as for each subsystem's daf2's, the sequence is sufficiently randomised for each part. Then, if their means are $(m_1, m_2, \dots, m_n)$ and their variances $(s_1^2, s_2^2, \dots, s_n^2)$, the mean of the sum of the parts is $m_1 + m_2 + \dots + m_n$ while its variance is $s_1^2 + s_2^2 + \dots + s_n^2$, assuming the mutual independence of the parts. To simplify matters, let us assume that the parts have the same mean, m , and variance, s^2 . Then, the mean of the whole becomes mn and its variance is ns^2 . Now the variability or coefficient of variation (ratio of the standard deviation to the mean) for the part is s/m while that of the system is $s/m/n$, i.e. the ratio for the system is smaller than that for the part by $1/\sqrt{n}$. But it is that it becomes smaller as n increases that provides the statistical interpretation of the idea of increasing invariance or inertia due to increasing contributing sources.

Of course, we would not have been able to explain the invariance at the global level and would not have known that the various sources smooth each other's deviations out if we did not attempt to take a deeper look by descending to various sub-levels. From our results, this has been more than justified. Studying the trees indeed led to a greater understanding of the forest itself.

However, one should not get the impression that agglomeration always leads to randomness. The study of the bug reports gave a beautiful example of the case when various sources are in phase with one another instead of being irregularly staggered so that summing up their simultaneous outputs led to any slight but

concrete determinism to stand out in relief among the inherent stochasticities, Sections 4.5.4.1.2 and 4.5.4.1.3. That there must be a sufficiently large number of these sources for the pattern to emerge is provided by the fact that the examination of the bug responses which emanate from only a few sources did not lead to the same finding.

As regards to the distributions, the Central Limit Theorem indicates that the predominant exponential shapes which we observed for the sub-levels will lead to the normal ones which we observed at the global level.

5.4.2 The interrelationships among the activities

After examining each variable on its own, we investigated them in tandem and in groups in an attempt to determine any functional relationships among them. Even if these are non-existent, we were still interested in summarising the data in some simple way which may not have any physical meaning but which may be extremely valuable for predicting the values of some variables by the values of other variables.

Correlation analysis which had been applied to the structural properties was also used here but in an extended form. Not only are we interested in the effects of each variable's value upon those of the others which occurred simultaneously (simple correlation) but the effects of its past values upon those of the others could also have been important and were investigated (cross-correlation). Most of the latter were negligible while the few simple correlations that were significant were easily explained: *c* is largely constituted of *e*, there is almost a one to one mapping from *c* to *mc* and from *REP* to *RES*. In fact, those last two series provided the only significant cross-correlation results: daily *REP* affecting the *RES* values on the same day and those which lag behind by a few days, too, Section 4.5.4.2. This is due to both the spill-over from one day to the next and the provision of a certain delay which may be accepted in receiving a response from a report.

Thus, some of the series remain closely coupled at all times even though they appear to vary stochastically on their own from one moment to another.

Note: Although similar analyses and results exist for some CCSS activity series, they have not been included here since the system has not been investigated fully by us.

5.5 THE EVOLUTION OF THE OUTPUT CHARACTERISTICS

Most of the activity variables we have just analysed (e.g., *c*, *e*, *f*, *daf2*, *daf6*, bug reports) can be viewed as inputs into the system. However, the maintenance process of software systems

is unlike normal transformation processes that treat their inputs as raw materials to produce visible outputs. Rather, the inputs are intended to transform the system itself - to render it more powerful, more efficient, more modifiable, less faulty, less complex and so on. Thus, the net result of all such efforts is a new instance of the system itself and the outputs, if one really insists upon them, are just the new traits in terms of power, efficiency and so on of that new instance. We have already envisaged in Section 2.11 that relating the outputs with the inputs will be the final goal of this type of research. Indeed, in Chapter 8, we shall attempt to lay the foundations by building the major part of the theory which we shall develop around the inputs-outputs concept. At present, the practical problems associated with any attempt to relate the outputs with the inputs, even qualitatively, appear to be insurmountable.

We shall discuss these problems more appropriately in Section 6.3.3. Here, we merely point out that the traits one would have liked to study are probably not analysable. Some of these have been most talked and bandied about. Although many metrics have been devised and proposed, the fact remains that none of them has been quite successful. The concepts are as fuzzy as ever and it may never be possible to obtain a totally satisfying measure for any of them, Sections 2.8 and 6.3.3. We, thus, preferred to concentrate upon the study of the behaviour of some attributes which, on their own, could be construed as outputs from the system.

5.5.1 Net growth characteristics

Growth is, by far, the most visible and easiest to measure of all these variables. The data for B are limited but still indicate that the system has grown more often than not during the 6 releases or so that it has been in existence. The major part of system growth is through the additions of whole chunks of new code such as subsystems while the rest is accounted for by the actual modifications to the existing code, Section 4.2.1. BD exhibits analogous behaviour: change in size (mainly increase) is mostly through the additions of new modules, Section 3.3.

Because it is the addition of whole chunks of new code and not the modifications of the existing code which accounts for most of the growth, one may ask whether the concept of the system getting more and more unstructured and complex as it ages holds as strongly as it has been made to be. Certainly, the addition of new chunks or the removal of redundant ones will nearly always increase the system's complexity, however minimally. The existing code will have to be modified, however slightly, to accommodate the additions or the removals and the new code will increase the system's size and establish its own lines of

communication with the existing code. Yet given that clear and clean interfaces were provided for in the first place among the subsystems in systems constructed such as B and that the surgical knife or the stitching needle is applied cleanly, then removal of obsolete subsystems or additions of new functions, facilities and code through entire subsystems, the main causes of change as we have seen, should affect the structure, hence the complexity, of the system in modest rather than marked fashion. On the other hand, it is plausible to assume that components which are actually modified do increase in unstructuredness and complexity. This is probably a more important contributor to the overall system complexity and its increase would probably have been quite noticeable had this type of change been predominating. For BD, which is more a collection of modules than a system as we know it, the situation is much simpler. As the modules are mostly independent of one another, it is easy to add or withdraw a few without too much concern about the whole system. Also, although each module may be changed quite often and may become very unstructured and complex, since it does not interconnect with the others, the whole system's complexity is not affected in too marked fashion.

As it is, for BD, there is no uniform rate of development; development occurs in waves or cycles but these are in no way regular and therefore cannot be modelled quantitatively, Section 3.3. The intensity and duration of each development cycle is probably governed by many complex factors such as the needs, the available resources and the timeliness of starting the cycle.

Each cycle shows a gradual build-up to a climax, when growth is most intense, which sooner or later runs down to, sometimes but not always, almost nil growth, when most of the planned activity has been accomplished, until the next cycle starts. This sort of behaviour is reminiscent of a negative feedback process rather than of a positive one as in OS/360's growth [18]. Despite these differences, the end-of-cycle clean-up process that has been described for OS/360 also seems to exist for BD. The growth plot by micros generally shows a drop in the number of micros at the end of each cycle. Thus, although modules may not be systematically reviewed and removed from the system at the end of each cycle yet it seems that an effort is made to rid them of their redundant micros.

As asides to the main theme let us remark 1) that, apart from the clean-up feature, all the 8 BD size measures we have used exhibit very similar general qualitative behaviour. We, therefore, conclude that, for qualitative results and interpretations, one needs not worry too much about which of several alternatives is the best measure, each being as good as the other and 2) that "core" micros are increasing in size while

"non-core" ones are not, probably because the former are the most frequently changed and most prone to increase in size while the latter, mostly unused, need rarely, if ever, be modified.

It is important to note that in both our systems and in most of those studied by others, too, the activities relating to the major growth factor in the system's size have nowhere been documented on the same level as those due to fixes and enhancements to already existing code. Thus, it is possible to account for the variations that exist in the system's size in terms of the major activities pertaining to it (see the opening paragraph of this section, once more) only very broadly and the situation will not change as long as such activities shall remain largely undocumented and, therefore, unamenable to analysis.

5.5.2 The characteristics of modifications to the BD code

The activities, already examined at a global level in Section 5.4, both individually and interacting among themselves, can be reconsidered from the point of view of their impacts upon the system and whether the traits of the latter have been changing in any definite manner. In this section and the next, we shall concentrate upon grouping the various indicators together and interpreting them in terms of the evolutions of the traits of the BD system while those of B will be considered in Sections 5.5.4 and 5.5.5.

On the other aspect of the maintenance process, i.e. the fixing and enhancement of already existing components, we already know that, for BD, although c is stationary, e is increasing while f does the opposite, i.e. the ratio of enhancements to fixes is on the increase, Section 3.2.2. One way of interpreting this shift in favour of the enhancements could be based upon the assumption that the fix rate is proportional to the number of residual errors in the system, (roughly speaking, since errors get increasingly difficult to discover and a zero fix rate is no proof that the system is error free). Then it would appear that, as the system ages, the errors are gradually smoked out and the personnel increasingly finds time on its hands and, therefore, concentrates upon refining the system. The anti-thesis is that the system is getting more complex, that fixes are becoming increasingly difficult to effect and therefore less frequently carried out but this ill-fits the increasing number of enhancements and is therefore less favoured. (A situation where enhancements are increasing while it is getting more difficult to fix errors must indeed be odd.)

The former interpretation is further supported by two trends in the individual enhancements: most involve fewer and fewer micros while a few do the opposite, Section 3.4.2.2. Probably,

the final retouches are getting simpler while, in parallel, growing confidence is causing bolder and bolder developments to be undertaken. Other results such as those showing the increasing frequency with which already existing micros are incorporated into the new modules and the decreasing size of the latter, Section 3.3, imply that most developments are of the bells and whistles type and consequently lend support to the settling down theory. There were still other results which suggest that the process of enhancing and fixing the code has, contrary to belief and as the theory would have it, probably not introduced any significant number of new errors, Section 3.5.2.1.2. All these results, and others, which had all been summarised in Section 3.5.2.2.2 could indicate, as hinted at in Sections 3.2.5, 3.4.2.1 and so on of Chapter 3 and here, too, Section 5.3, that BD is probably an atypical system. We shall come back to the same theme at the end of Section 5.5.3.

The model of fixes for a typical new module or micro shows that they decrease exponentially to negligible values within 3 or 4 years after implementation, Section 3.5.2.2. The fall of e is much less steep and it takes longer to fade out so that the component is still being enhanced while it may no longer be fixed. This not only confirms the continuing need for further enhancements but also shows, as we have said before, that, in the case of BD, tinkering with the code does not seem to introduce new errors.

Another difference between the f and the e models is that the fit to the former was better. This indicates that while fixes are determined mostly endogenously, i.e. by the properties of the component itself, yet enhancements are more largely due to external forces, Section 3.5.2.2.2.

One last aspect of these models concerns the observation that not long after release there is a swell in the number of fixes accompanied by a decrease in that of enhancements. It is likely that the process of transferring a component from the testing to the usage environment causes its shortcomings to be suddenly exposed. This creates a pressing problem that leads to a spate of fixing activities while other activities are put aside for a while, Section 3.5.2.3.2.

The models could be used to predict the time to a specific fix rate, to a specific number of fixes and so on, Section 3.5.2.2.4. They could also be refined to include the possibility of introducing errors and the cycles or oscillatory patterns, Sections 3.5.2.7.1 and 3.5.2.7.4. However, perhaps their most useful property is that a quality measure for the components modelled can be extracted, Section 3.5.2.7.3.

Why these rather well defined behaviours are not reflected in the global e or f series has already been dealt with quite

fully in Section 5.4.1. Besides, we also have the evidence that the new modules form only a small portion of the whole system, at any time, and their introduction does little to affect the already settled pattern, Section 3.5.2.1.

5.5.3 Explaining why complexity for the BD system has not increased

From all our analyses (and it would be too repetitious to relist all the results here), we concluded that system complexity has probably not changed over time. This may be that the increasing system size, total number of enhancements and number of difficult enhancements are balanced by the diminishing number of fixes, the increasing number of easy to effect enhancements and of fix free modules and so on, to say nothing about factors like increasing familiarity and experience.

The learning factor may explain why complexity may not have increased despite the fact that the system is continuously growing in size and that changes are being made to its components. In strict terms, there is, indeed must be, an increase in part of what constitutes the complexity factor yet the true complexity itself, according to our results, does not seem to be affected. This is, however, not contradictory as it is the psychological complexity of the system that we are mainly interested in. Because the maintenance team has been largely unchanging and the people that constitute it have been interacting with the system for quite a long time, it is not surprising that what would certainly appear, to any external observer or newcomer, as a more complex system than the one which was initially brought into use so many years ago, is to those people something which they now understand much better than previously. Thus we can draw 3 lessons:-

- 1) That any complete study of the complexity of a software system must include a study of its metasystem.
- 2) That, since human behaviour is involved, we may have to enroll the help of people such as psychologists who are more expert at understanding such behaviour.
- 3) That, because of the very involvement of human beings and of their subjective appreciation of what is complex and what is not, there is practically no hope of ever finding any true complexity measure. We have already touched upon this theme early on in Section 5.5. Sections 2.8 and 6.3.3 discuss it more fully.

But there dwelt the niggling thought that we can expect complexity not to increase despite continuous growth and modification only for systems such as BD, which, we have already said, is quite different from most of those that have been studied. Is the cause due to the small size of the system so that the threshold of a reasonably intelligent person's

ability to take in and understand the whole system is not exceeded? Or is it its structure which is the overriding factor? Or that the maintenance team is small and has been stable? Each probably does contribute significantly to the behaviour but one can only speculate about which is the more important. Finding out seems to be an important direction in which to pursue any future research but we anticipate that only after slow and painful work shall we be able to know. A quick way may be to design a certain experiment but can we really afford to, Section 2.6?

5.5.4 Characteristics of modifications to the B code

Since the B data did not quite resemble BD's, the latter's analyses could not exactly be duplicated. The activity models provided one of the areas where we could contrast the results from each system. Although the various B models were in terms of different entities (the daf2, software notices and daf6 models were for holons, whole system and groups respectively) yet, if one ignores the redevelopment phases, all their shapes are characterised by a rapid rise to a peak which then gradually falls off, and, therefore, resemble the BD entities', too. The phases are due to functional redevelopments for the purposes of a new release although there are no rules concerning them. Which part of the system to redevelop, when to do it and the resources to be involved (in labour and time) are driven, to a large extent, by factors external to the system itself, Sections 4.4.4 and 4.6.6.1.4. Therefore, as could be expected, there were no patterns among the phases of each part considered. Our models, as before, were, consequently, only qualitative in nature.

Two important deductions from the shapes of the phases and their frequencies are:-

- 1) The tail in each new phase indicates that, as in BD, there would be a vanishing of activities, should there have been no new phases.
- 2) That new phases appear from time to time indicates that any decreasing activity would probably be due to the settling-in process rather than to increasing difficulty.

Because of its sheer size, its existence in a competitive field, of the evolving technology and needs of clients and so on, B probably cannot escape from metamorphosing into newer instances of itself - releases - at frequent intervals. For BD, which has well-defined and relatively unchanging functions and caters for a more stable clientele, the release concept is not even entertained.

5.5.5 Complexity investigation for B

One early indication that total system complexity might have been decreasing concerns the software notices whose intensities and variabilities from week to week have been decreasing on a release-to-release basis, Section 4.4.5. This simplifying process finds parallels in the decreasing trend in the response time for most bug reports, Sections 4.5.3 and 4.5.4.3, in the queue of outstanding bug reports, Section 4.5.4.1.2, in the software notices, Section 4.4.2, in the total daf6's as well as new redevelopment phases, Section 4.6.5 and in the diversities of the daf6's according to several variables, Section 4.6.6.1.4. Coupled with our failure to detect any increasing trend in the number of components examined per bug report, Section 4.5.4.3, or in the number of attempts required to effect a change correctly, Section 4.6.7.1, among others, we decided that this system, like BD, shows no evidence to be increasing in complexity. One way of explaining this sort of behaviour has already been given in Section 4.2.4 where we theorised how the ageing (due to the changing components) and rejuvenation (due to the newer and simpler components) rates may cancel each other out to cause the system's attributes and behaviour to remain largely unchanging, as found in Section 4.2.2. But why the system may not have been increasing in complexity could be attributed to several factors: the team is getting more experienced, the system is settling-down or maybe it is still so young that the complexity threshold has not yet been reached and/or the dynamics/inertia of the system have not been given sufficient time to really get going. Since it is the factor which sets B apart from most of the systems studied in the ED context, the latter could be the more important factor, although, at present, there is no way to prove it. Then, one would still like to see whether, as the system ages, the threshold would remain unexceeded and whether increasing ossification could be prevented.

Thus, as far as the data allowed, the components of BD and B have shown quite similar qualitative behaviours. The subtle but important differences which show that, at the global level, BD is more responsive to external factors while B possesses dynamics (trend) and inertia (invariant) properties to a greater extent must not be forgotten either. A most major step would be undertaken if these similarities or differences could be precisely related to and explained by each of the size, structure, age, environment factors and so on. This has, of course, not been possible from the data we had.

5.6 THE CHARACTERISTICS OF THE SYSTEM - ENVIRONMENT INTERACTIONS

5.6.1 Relating the outputs with the inputs

The effect of the system's particular maintenance environment, together with that of the world further outside, upon the

system cannot be totally ignored from our studies if we wish these to be exhaustive. As it is, data concerning the environment, whether immediate or further away, and its interactions with the system are hard to come by although the activity variables which we have been using and which could be viewed as inputs into the software system are, after all, embodiments of external factors acting upon the system, although they are partially endogenously driven, too, being, say, feedback effects from previous system outputs. Take, for example, the number of fixes to the system at any time: its value is dependent upon the state of the system at that time and this is, itself, dependent upon previously carried-out fixes. It is in this context that the remarks concerning the large stochasticity in the behaviour over time of the enhancements, whose behaviour is conditioned to a larger extent by external determinants rather than internal ones, but not so much in that of the fixes to a typical component apply, Section 3.5.2.2.2.

We have already remarked that it is extremely difficult to relate inputs such as enhancement activities with the new system traits which result from these activities in Sections 2.8 and 5.5. At this moment in time, it has only been possible to investigate the behaviours such as long-term trends of the inputs and outputs (or, rather, easy to measure indicators for these), mostly on their own, to qualitatively, and only in general terms, infer the behaviour of one from that of the other. What we would have liked is to predict, more or less accurately, what sort of effect a certain change in an input, say a given increase in the number of enhancements, will have on a given trait, say complexity. This, at present, cannot be accomplished because there may be so many factors acting upon each trait that what effect each may have may not be determinable since one does not know what these factors are, Section 2.7, and also what the effect of each may be since one may neither be able to isolate each factor from all the rest for specific treatment nor are they all visible. Some may, for example, be too far removed both temporally and physically for them to be noticed, Section 2.8. Furthermore, their interactions among themselves may be too complex to unravel.

We were confronted with similar problems while attempting to model the behaviour of the activities themselves. Thus, the number of enhancements, at any time, can be determined by the internal properties of the system at that time (its traits), other activities within the metasystem or the same environment (e.g., the number of fixes to be carried out during the same period), activities in the meta-metasystem or the outer-environment (e.g., users' pressure for change), as well as past activities. The situation may indeed be too complex for an empirical approach to succeed (see our quote from Forrester in Section 7.4). Then, it would indeed not be surprising to find that, for each activity that we have analysed, randomness is

the main feature we have observed. One anticipates that the more we shall be able to explain the system's and the metasystem's activities behaviours in these terms, the less will be the degree of residual randomness which, at present, is intolerably high.

In an early and rather crude effort, multiple regression models were used to determine the extent to which the exogenous factors affect both the structural and the behavioural properties of the system. That our best BD model gave an R^2 of only 17%, Section 3.4.3.3, while those of B were upwards of 70%, Section 4.2.5.2, lends further support to the idea that B is less responsive to external influences than BD.

But, of course, these were only static models. Despite the weaknesses of regression models - Regression Theory only purports to find and display patterns of relationships which can be used for prediction but which may, on the face of it, be nonsense so that the models created may not be interpretable at all - they provided the only means by which we could attempt to adopt Sutherland's ideas about his 4 classes of systems. From the few models for the same thing but taken at different points in time we found that B could best be classified as of the third, i.e. the severely stochastic class, Section 4.2.5.2.4. Thus, without more accurate and further data as well as better tools, we have little hope of predicting future system behaviour with any degree of accuracy.

5.6.2 Investigating how the different characteristics of a variable change from part to part and from release to release

Of the various data we have worked with so far, some such as bug reports and daf6's allow us to study the system-environment interactions according to each of several different properties or aspects of the same data. These range from those that are almost totally environmentally determined and controlled, e.g., the originators of both bug reports and daf6's, the priority and machine characteristics of the bug reports, through those that are influenced by both system and environment, e.g., response time or number of attempts for both types of activities, to those that are almost exclusively determined by the system's properties, e.g., the faulty components and the bug categories in both bug responses and daf6's.

Strictly speaking, f and e are subcategories of c. In that respect, then, the distribution of the BD changes is shifting from fixes to enhancements, Section 3.2.2. For B, at whatever level of the system the investigation was carried out (subsystem, group or global level), we also found that, for each property of the data, whether bug reports or daf6's, the distribution among its categories has not stayed the same over time, Sections 4.5.2.1, 4.6.6.1.4 and 4.6.7.2.1.

There do exist some deterministic inner trends in some of the categorising schemes, e.g., the response time of the "short to respond" bug reports is decreasing, Section 4.5.4.3, while the daf6's of general utility is increasing, Section 4.6.6.1.3. However, the real pointer to whether the interactions between the system and its environment are getting more complex is on two more global fronts, the first of which englobes the inner trends while the second englobes the first itself. To elaborate, we wished to find out, in the first place, whether, in each method of categorising the data, the shift is towards the categories of the more complex types and/or whether the data is getting increasingly diversified and, in the second, whether this diversification is general over all aspects of the data.

Alas, there was no great concordance among the diversity trends in the various properties of the daf6's data, Section 4.6.6.1.4. Some were increasing, others were decreasing while the rest did neither. However it is not the trend which is the important feature, it is what interpretation can be attached to it: some increasing diversity trend may mean the same sort of process as the decreasing one in another aspect of the same data. Thus, the increasing diversity in MTHS is taken as good since it implies that the modules are getting redaf6'ed at longer intervals and together with the decreasing diversities in UTIL, CAT and TIMES, it probably implies a simplifying process in the system.

The contrary behaviour of the other properties' diversities which lead to the no concordance results and the weak pairwise correlations show that the environment-system interactions, as interpreted from the daf6's, are probably independent of each other, any associations which they may form being weak and of a temporary nature, Section 4.6.6.1.5. Bug reports, on the other hand, have thrown up contrasting results. There are definite associations - each team has its own working habits and at a certain given time can be found to work upon a specific release, machine and component(s), Section 4.5.3.

After investigating how characteristics $C_1, C_2, \dots, C_n$ behave over part P_i , we then explored whether the behaviour of each property is uniform over different parts of the system, i.e. how C_j behaves over parts $P_1, P_2, \dots, P_m$. One finds that, qualitatively, C_j generally preserves its behaviour, not only in the evolution of its spread among its categories, e.g., the diversity of CAT is decreasing for all 3 groups we examined while that of VERSION is increasing for 2 and constant for the last one, but also in its inner trends, e.g., for all groups, the proportion of daf6's repairing implementation bugs is increasing, Section 4.6.6.3. These, therefore, indicate that the influence of external factors is more or less system wide.

But are they distributed out in equal doses? I.e., for a given period or release R_k can we order $P_1, P_2, \dots, P_m$ by their diversity ranks on all the characteristics $C_1, C_2, \dots, C_n$? This was certainly possible although the l th ranked part is not necessarily l th ranked on all its properties. The same ranking, however, does not necessarily hold over all releases, e.g., for R_k , we may have $P_a < P_b < P_c \dots$ while, for R_{k+1} , we may have $P_b < P_c < P_a \dots$ although there is a tendency for some parts to occupy the lower ranks and for others to occupy the higher ranks. The changes in rankings from one release to another are probably due to different sorts of emphasis being put on different parts of the system at different times (exogenous factors) while, on the other hand, the tendency to preserve the rankings indicate that the endogenous properties of each part have a role to play as well. They probably help to preserve some element of stability within each part. Besides the properties of the part, the stability also depends upon which property of the data we have been considering in the first place.

Having more or less established the influence of exogenous factors, we made an attempt to determine when they are strongest and when they are weakest. This was done through ranking $R_1, R_2, \dots, R_p$ by their diversities on all properties $C_1, C_2, \dots, C_n$. Just as before, the introduction of time as a factor also introduces a greater degree of randomness and it was impossible to obtain consistent rankings among the properties for each release, whatever P_i is used. We then had little confidence in the validity of the ranking $R_a < R_b < R_c \dots$. Furthermore, emphasising different P_i 's at different times also meant that there were as many such different rankings as P_i 's.

In summary, we may say that there is great variability in the data. First of all there is a great stochasticity in the behaviour of each factor and of each part over time. Then the properties, ranging from the relatively stable to the very unstable ones, do not show concordant behaviour over time although each of them interacts in a similar fashion or not too different fashions with different parts of the system. Finally, these parts themselves range from the stable to the unstable although their stabilities are related to those of the properties.

5.7 CONCLUDING REMARKS

For either Chapter 3 or 4, apart from the introduction, the summary and certain sections which are concerned with the description of the data or some mathematical development, we have made sure that we have dealt with all the sections and included all their results here. There were only a few exceptions which were left out because they did not quite fit in the development of ideas in this chapter, e.g., Section 3.4.2. There were also some sections such as 3.4.1.1, 3.4.1.2, 3.5.2

and 4.2.4 which had already been thoroughly dealt with. We, therefore, decided not to include more than what had been necessary here. However, the reader must still note that, although we have attempted to interpret the results as far as we could, we have refrained from crossing over too much to their generalisations. The theorising that our conclusions would have to eventually lead to is more appropriately left to Chapter 8.

CHAPTER 6

THE FLAWS IN THE DATA, TOOLS AND RESULTS

6.1 INTRODUCTION

After pooling all the results of Chapters 3 and 4 together, interpreting them and attempting to put them in perspective in Chapter 5, the natural progression from there seems to attempt to generalise from the conclusions and to extract a general theory for large software systems behaviour over the maintenance cycle.

But any theorising must be made within the context of basic results and realities, i.e. the extent to which we can build vertically up will depend upon the foundation stones - how strong they are, what are their flaws and so on. It is therefore a priori necessary to discuss the weaknesses in the data, tools and analysis, the possible ways of strengthening these, the directions in which the study may be pursued, among other things.

The discussions can be conducted anywhere between the two extremes of treating each of the topics singly and all together. With the former, the reader with specific interests, e.g., the researcher who only wants to know about our recommendations or the manager whose sole aim is to find some tools for monitoring the system, needs not wade through the entire discussions. Unfortunately, repetitions are bound to occur even if the same ground will be covered from different points of view.

The best compromise we found is to delay the recommendations until the next chapter while the contributions, weaknesses and usefulnesses are taken up, in the same step, here. Simultaneously patting ourselves on the back and kicking our backsides ensures that neither is the ego too flattered nor the pain too unbearable! Both have, of course, to be moderated. While we particularly want to draw the reader's attention to the stronger results yet we must not antagonise him by sounding too self-congratulatory. Where there are weaknesses we want to convince him that what we achieved were the best under the circumstances, yet we may not list too many of them for fear of being accused of building our thesis uniquely upon weak results.

6.1.1 The need to devise a methodology for analysing the data

We found that to analyse the data the way we did, we had to practically invent our own methodology. This was based upon a mix of statistical tools and techniques from other disciplines: Shannon's H from Information Theory, Clustering from Taxonomy, the Delphi Method [24,54] from Management Science as well as

the systems ideas of Sutherland and Forrester, not to forget the techniques of Control Theory and differential equations among others. We leave it to the reader to assess the sufficiency and success of the borrowing and whether we have been able to match the appropriate tool with each problem as we wanted to, Section 2.3.

After we have performed our analyses, we found that, as we had anticipated in Chapter 2, there has quite naturally emerged a methodology which could be of general applicability for systems evolution research. This methodology, which seems to go beyond any which has been previously applied towards the systematic study of large software systems (and, possibly, of general systems, too) from empirical data derives its strength not from the invention of any new tool or technique but, rather, from attempting to blend already known tools from statistics and other disciplines into a complete kit. To be sure, most of the tools we have used are very widely known. Probably, of the most widely-known and commonly-used are correlation and multiple regression analyses. Others are not so well-known but, nevertheless, have been applied before in the software engineering field, e.g., the Kolmogorov-Smirnov test by Littlewood [118], the Delphi method by Scott & Simmons [158] and Clustering techniques by Lee, Slagle and Mong [92]. But to our knowledge, some of those we have used, e.g., Jonckheere's test, the randomness tests, distribution comparison tests, time series techniques, have not been applied before in our field.

Of course, there are several statistical books which describe quite a few of the techniques we have been using (and others, too). There are even several statistical packages which have been in existence for several years, now, but neither the books nor the packages, to our knowledge, give much indication about which tools to apply, in what sequence, etc., to achieve one's goals. Even in a recently published book (which came to our attention just as we were about to complete these investigations) intended for software engineers, Kobayashi [89] mainly concentrates upon the theoretical aspects of the tools rather than upon their application or applicability, either singly or as a complete kit. Indeed, no work such as ours seems to have been undertaken, despite the existence in the literature of data, admittedly scarce which could, say, be treated as time series or compared by more formal means than visually [21,32,77,169] and, despite calls for more formal techniques, e.g., at the second SLCM workshop, Belady [12] said that the use of time series analysis "must be introduced".

6.1.2 Working with distributions and the associated advantages and problems

On another front, we need to emphasise that the proceedings have been carried out at the distribution level. We felt that

this was justified although working with tendency measures such as the mean would have been simpler. In the first place, there is the attack on sub-disciplines like Software Reliability where such measures as MTTF have been given too prominent roles [117]. In the second, one has little use for the mean alone. Weinberg [183] gives this example: most people prefer to wait 10 minutes each morning for the bus than to wait 1 minute on 4 days and 26 minutes once a week, although the average wait in the latter case is 6 minutes. Thus, any study should measure both the mean and the variability - not just the former "as most studies do".

But even after we had decided to work with distributions rather than with means, we still trod carefully. We wanted to make sure whether, like most workers, we could regard our distributions as normal and therefore call upon the relative abundance of theoretical results which exist in its case. That the classical theory of probability has been one-sided in properly studying only the normal distribution is, of course, due to its dominating role in both theoretical and applied questions. However, we found that, in many cases, our data were by no means normally distributed, Sections 4.2.2 and 4.3.3, and that using the standard normal theory was likely to be seriously misleading. Besides, the classificatory nature of quite a few of our data, Sections 3.4, 4.2.2, 4.3.3, 4.5.2 and so on, precluded the use of parametric techniques. Thus, most of the well-developed statistical methods, including those along the mainstreams of statistics, e.g., multivariate methods, component analysis, factor analysis and factorial design [128] could not be used. We had to content ourselves with non-parametric tools [163]. These are generally weaker but are the proper tools if we want to continue matching them with the job, Section 2.3.

Perhaps one compensation for having gone into all these pains is that distributions, albeit under very strict circumstances, allow us to infer the global from the individual and vice versa, e.g., see the reproductivity properties of the normal, Poisson and gamma distributions in [89].

Concerning the data themselves, we prefer to retain the main details until after the methodology itself has been discussed. Here we shall only reiterate a few general points. We have already discussed, in a general sense, the inevitability of working with what we have instead of with what we want in Section 2.9. Thus, no attempt was made to define and identify exact and complete structural properties or input and output activities to measure and analyse. Instead, already available or readily countable measures which could be perceived as reasonably related to structure and behaviour were judged to be appropriate enough. And, because we were sufficiently willing to measure and analyse anything that could be reasonably

measured, to the extent that it was possible, we have tried to keep to the spirit of approaching the problem with an open mind (Section 2.5), letting the data and the results of their analyses mould our subsequent thinking more than anything else.

We have already briefly summarised in Section 2.11 the 3 fronts: the structural, the dynamic and the ecological, upon which our methodology attempted to attack the problem. It is really unnecessary to explicitly spell out how it is done as the logical sequence in the analyses and, hence, the methodology itself, can quite easily be grasped from reading Chapters 3 and 4. Therefore, we preferred to concentrate, in this chapter, upon the problems associated with each aspect of the analysis rather than to attempt to describe an exact recipe for large system analysis.

6.2 STATIC STRUCTURE ANALYSIS

On the structural front, the idea is to understand how the system is structured and to draw, if possible, a map whereby the configuration and the static properties of the system can be revealed, i.e. we needed some sort of spatial distribution that is able to show which components comprise the system, where each component resides relative to every other and that can represent the prevailing relationships among the components.

The most advanced and appropriate tool that, to our knowledge, can best accomplish these tasks is the interaction or connection graph and its associated matrix of Graph Theory. Although it provides us with the fundamental mathematical basis for the determination of system structure, Graph Theory, in the context of this research, suffers from 3 disadvantages:-

- 1) Although it may model the components' interfaces through both calls and data bindings yet there are other aspects of structure such as size which do not properly transform into connection matrices.
- 2) Although the theory is well developed for any matrix yet there is nothing which allows the comparisons of several matrices, exactly what we want if the analysis is to progress naturally, i.e. for the static as well as for the dynamic part of the analysis. Thus, connection matrices do not seem to be very useful for structure evolution investigations.
- 3) This concerns the difficulties in manipulating such matrices. An initial appraisal indicated that the data were probably not accurate enough to build representative matrices. Then, at present, we could not construct the matrices other than manually and this was a very time consuming process. Also, a thorough investigation will require the construction of a matrix for each of the many

structural properties that lend themselves to such analysis. But the greatest difficulty concerns the sheer size of any matrix which would have to be constructed. There must be $n(n-1)/2$ entries to represent n components in a non-directed situation (and twice as much in a directed one). As n exceeds 4,000 for B , the task of constructing even a single matrix is literally impossible, even if a program and a data bank were available.

Persuing the analysis along this path, had it been possible, would not only have presented a somewhat novel approach but it would have allowed the introduction of other ideas such as the use of spatial correlation [52] to provide solid evidence in favour of or against the so often talked about increasing communication among the components and the related ripple or propagation effect [69]. But, alas, this was not to be.

Perhaps, working on a small subset of the system, to start with, could have been both acceptable and manageable. But, since the problems really concern developing the techniques to a sufficiently adequate level to allow the analysis to proceed (the means) rather than the better understanding of the system (the ends), we preferred, so as not to risk devoting all our time to it and so that we could pass on to other things, to choose more expedient ways of dealing with that particular problem. However, it must not be forgotten that there may be enough material here for a worthwhile future research project which could lead to very significant contributions, too.

The alternative and more expedient method of structure analysis was found in the form of constructing and examining the relative frequency distributions of the components by what could be perceived as structural properties of the system. Frequency distribution not only shows the frequency of the components within each class and is hence perhaps superior to connection matrices for characterising features such as size but can be compared (pairwise by the Kolmogorov-Smirnov and χ^2 tests or in group by the Kruskal-Wallis and the χ^2 tests, say) and even tested for some particular order (Jonckheere's test). Thus, it has the advantage of allowing evolution to be investigated. Unfortunately, it suffers from several weaknesses: it does not tell us anything about the system's actual size, about the relationships among the system components, about the characteristics of or changes in each component and so on. Two examples, both of which are extreme, are illustrative. The system's doubling in size is transparent to the frequency distribution if the frequency of the components in each class is likewise doubled. The distribution remains unaltered even if all the components change their values, provided that the set of final values is the same as that of the original values. Although we know that such extreme

cases are unlikely yet we must be aware of the fact that substantial actual structure changes may be reflected only to a limited extent in the frequency distribution.

Perhaps the correct approach is to combine the frequency distribution analysis with that of the connection matrix, complementing them with other tools, if necessary. Unfortunately, for already stated reasons, we could only deal with frequency distributions.

Although we have investigated the components' distribution by more than a dozen properties, we found that, we could not say that we have measured all or most of the properties that mattered, e.g., with respect to a given trait such as complexity, and not measured those that did not. We could not even affirm that what we have measured are good indicators of the traits we are interested in or even distinguish the more important ones from those that were not. We were, however, able to establish, through correlation analysis, that quite a lot are interdependent although they are not concordant. As is well known, correlation cannot establish causality nor is it able to detect non-linear relationships. Furthermore, there is insufficient extraneous knowledge to supply the causal nexus. Therefore, failing the availability of more sophisticated tools, we shall have to assume that there probably exist some complex causal patterns among them whereby a change in one property may cause complementary, sympathetic, compensatory or counter-active changes in the other properties. Added complications may be caused by the possibility that these relationships may evolve over time, but we do not need to discuss them here.

Another problem concerns the many possible levels at which the system can be studied. Although a global level gives a holistic view, it does not allow us to see inside, to understand a little bit better what is really going on and, because of the system size, may not be practically feasible. Selecting any lower level to explore is probably valid since it serves to highlight some particular aspect of the system but it immediately begs the question. Who is to say which level is more important than the others and, given the amount of analysis which each entails, one can simply not attempt to explore all of them. The same problem arises when, instead of considering the whole population together at whatever level we are looking at the system, different subsets of it are considered separately and/or compared with each other. There are unlimited ways in which to partition the population and because it may not be homogeneous, each partitioning probably leads to new and equally meaningful results, too.

Then, we come to the question of whether the global behaviour can be inferred from the subsystems'. Despite that the latter

may roughly follow the exponential or the normal law and that the reproductivity of each of these laws allows us to infer that, globally, the behaviour is the same yet the components may not all be on an equal footing. There may be some of them which, by virtue of, say, their central positions, are very critical and, if these should be anomalous in their properties, then inferring that the behaviour of the whole system is similar to the common behaviour of the majority of its components will be quite incorrect.

6.3 EVOLUTION ANALYSIS

Dynamic evolution analysis was the second front on which we attacked the system analysis problem. One can choose to work on either a time or a pseudo-time basis. The main ED proponent has always maintained that release-based analysis is as meaningful and as susceptible to lead to significant results as the time-based approach [18]. We do concur, Section 4.6.2, and as our results themselves show, his statement seems quite reasonable. However, the problem was that we had to choose whether we preferred release-to-release models of behaviour or continuous-time models or whether we could afford to conduct two parallel analyses at the same time. As it was, we let the circumstances dictate our approach: BD, not release based, could only be investigated on an actual time basis, with B, depending upon the amenability of the data, some analyses such as structure evolution analysis were principally dealt with on the release basis while for others, e.g., activities dynamic analysis, the emphasis, again because of the nature of the data, had to be placed on the actual time basis and still for others, such as the daf6's analysis, evolution over both domain could and was explored.

6.3.1 Structure evolution analysis

Our structure evolution analysis has not been as thorough as we would have wished: it has been performed for only one system and on the release domain only, just 2 releases have been used and, as already discussed, because of the lack of theoretical developments, either the tools are non-existent or they are crude, Section 6.2. Thus, the conclusions and hypotheses, based on these few empirical data and results alone, cannot be said to rest on solid foundations. But they seemed well supported by qualitative evidence and that was the best we could achieve under such circumstances.

6.3.2 Activity dynamic analysis

Let us start by considering the domain of the analysis. We have already said that both the time and the release-based approaches are probably conducive to equally meaningful results; they simply lead to different types of models. However, as BD is not

release-based and as there are only a handful of B releases, we had, more or less, been forced to restrain the major portion of our evolution analysis to the time domain. The time-based method suffers from the problem of choice of the aggregation period. A finer mesh also implies a finer analysis in the sense that shorter and shorter term effects can potentially be identified but unfortunately stochasticity increases, too [79]. Of course, enlarging the aggregation period achieves the contrary effects. Therefore, a proper balance must be achieved. That we concentrated upon finding each activity's best aggregation period led to inconsistencies among these and to the uneasy feeling that the short-term effects found in the finer-meshed series could also have been found in the other series had the same finer mesh been employed on them, too, but these were the prices we had to pay. Probably, they were less than that of choosing a uniform mesh.

There is probably no point to repeat once more the general weaknesses of empirical data, Chapter 2, or to discuss what could have been done had certain types of information been available. The points we want to make here are that only what was available could be used and our results should be judged on that basis, that, for B, measures not quite similar to those of BD had to be used so that strict comparability is not possible and that, even within the same system, consistency could not be maintained, e.g., the measure may relate to holons, modules or other components.

It is also important to stress that whatever behaviour these series exhibit it is in terms of quantities alone. For example, if the quantity of changes is decreasing over time, this is all that one can infer. One may not say that work rate is also decreasing unless one knows something about the relative quality of the changes. It is possible that the difficulty of effecting a change has been increasing and that the work rate has therefore remained constant.

Concerning the tests themselves, let us remark that, compared with structure analysis, there are many more tools here, although not nearly enough or as strong enough as we may have wished.

6.3.2.1 Tests which are used on each individual series

Scatter diagrams and line plots have not been very useful because, due to the great stochasticity in the data, it is virtually impossible to deduce anything useful through their visual examination.

Of the several tests to find whether the series are subject to well-defined influences such as long-term tendencies to increase or decrease, seasonal variations which occur in

regular sequence at specific intervals of time or cyclical fluctuations which represent consistently recurring rises and declines in activity, quite a few could be classed as significance tests and, as we know, they cannot definitively establish the truth or otherwise of a hypothesis. Each test, better for investigating some particular aspect of the data but worse for others, may even be misleading on its own, Section 1.3.3.2.

We, therefore, had to ally these tests with other techniques such as correlograms and periodograms (we did not proceed to spectra since the periodograms gave more than a hint that they were not required) to attempt to form a stronger tool which could allow little doubt on the global conclusion if each individual result leads precisely to the same conclusion.

While looking for trends, testing for the significance of the regression coefficients of the fits was one further method we used to support that family of tests. Such fits are however only based upon historical data, cover only the period looked at and may not be used for other than very short term forecasting. It is also impossible to know whether the trend will be maintained: should some kind of innovation occur, there may be a real break with the past and forecasting the future from the past would become useless. Perhaps a greater defect of curve fitting is that it does not aspire to explain behaviour.

Thus, more complex models were required. One form was provided by time series analysis techniques in both time [29] and frequency domains [25,79]. However, these involve a fair degree of statistical sophistication and the end products, specially the Box-Jenkins stochastic models, are both elaborate to produce and remain restricted to short-term forecasting. Thus, only a few models could be explored.

One can only indirectly search for evidence of seasonality and cyclicity through examining the plots, the correlograms and the periodograms as well as through significance tests. Seasonality, apart from a regular dive in activities at Christmas, probably does not exist. Abstracting the trend and seasonal components, should they be present, would leave a fluctuating series which may, at one extreme, be purely random and, at the other, a smooth oscillatory (term to be preferred to cyclic as used by Lehman [18]) movement [86] which represents some kind of systematic effect in the sense that successive observations are not independent. We found that software activities, like most economic series, are probably not oscillatory: we have something between the two extremes.

The final work while considering each variable on its own was fitting a distribution to it. Sometimes this was not quite justified since the presence of a trend or of serial

correlations, however weak, still indicates some significant time effect and, therefore, violates the condition of independence. We could perhaps have used more distribution types than we did but the extra work seems hardly justified since one does not necessarily select the distribution which best fits the data, the physical interpretation of the fit being of greater importance. Besides, each set of data can probably be equally well-fitted by several distributions [9].

6.3.2.2 Testing for inter-relationships in the series

On our attempts to analyse the series more than one at a time, we felt very much hindered by the lack of adequate tools. Cross-correlation suffers from the same weaknesses as simple correlation, Section 6.2, and there is the problem of the choice of the proper aggregation period, too. The problem was compounded by the fact that different types of activity series covered different periods, were differently aggregated and referred to different entities. Thus, we had not been able to relate the variables as much as we would have liked.

6.3.3 Output dynamic analysis

Of possibly greater importance is our failure to relate the system's inputs, as most activities effectively were, with its outputs. System outputs were hard to measure and to come by. One of the most easily obtainable outputs was growth in system size but, paradoxically, its related activities (inputs) were among the least documented. Structure changes to the already existing parts of the system could be viewed as the outputs from the fixing and enhancing activity measures but whereas the data for the latter cover a long period of time, that for the former did not.

Obviously, one would have preferred to measure and to work with outputs such as changes in power, understandability, quality and complexity. These, specially the last two, have been the subject of much research but, despite some noteworthy work, e.g., that of Boehm et al. on quality [28] and those of Halstead [68] and McCabe [120] on complexity, the truth is that they still fall short of the ideal and that, as yet, no universally accepted measure for any of these concepts exists. Some may take the view that, with perseverance, the goal will sooner or later be reached but we tend to side with those who believe that this search for the Holy Grail is bound to be fruitless. Our argument is that, because of their very nature, such concepts are unmeasurable: they are subjective and require an element of appreciation. And the involvement of human beings always implies that no precise measurement is possible. Just contrast the hard sciences with Psychology, Sociology, Economics or, for that matter, any science that has something to do with people.

We, therefore, come to the important conclusion that one should abandon such efforts and concentrate, instead, upon identifying good and easy to measure indicators of increasing complexity, etc. and upon investigating whether they show concordant behaviour. Other, more mature sciences, have, after all, already shown the path: when investigating whether a nation's health is improving, doctors usually resort to finding out instead how the life expectancy, the percentage of deaths caused by cancer, the proportion of infants who do not survive their first year, etc., have evolved.

Thus, system outputs, in terms of more realistic measures such as size growth rate, rate of making changes, average size of error free component, components changed per change, backlog of outstanding bug reports, bug report response time and number of attempts required to perform a fix should also lead towards equally fruitful results. Such measures are all probably manifestations of one or other of the concepts of complexity and so on but analysing only one of them would not be sufficient for any strong inference. This is because of the possibly different ways of interpreting the results, e.g., a decreasing change rate may indicate either increasing difficulty or a decreasing need for changes. One would be much happier to infer increasing complexity if all first 3 measures exhibit decreasing trends while an increasing one is in the others.

It is doubtful whether increasing complexity as represented by trends in these measures can be clearly detected amidst the strong stochasticities in the data. Probably not in the case of the seventh measure where most fixes require only one attempt to be successful, Section 3.4.2.2, so that the trend, even if real, would likely to be small and submerged by the stochasticities and in the fourth, too, because it does not adequately take into account all the information contained in the complicated changes to components mappings, Section 3.4.2.1. And then, taking only a global view and, therefore, seeing only the surface may be insufficient. Undercurrents such as the decreasing trend in most bug response times, Section 4.5.4.3, and the decreasing number of micros affected by the majority of enhancements, Section 3.4.2.2, can be completely hidden by the greater variability of the fewer but much larger values.

We have already pointed out that the inputs-outputs relationships are undoubtedly of a very complex nature. A great deal of work is still required to identify and accurately measure the relevant inputs and outputs. Techniques for relating these empirically are few and/or cannot cope with problems of this complexity. Although there exist some techniques such as the Box-Jenkins transfer models that, at

least, allow us to make a stab at the problem, we found that we could make little progress. Even though we had a few input variables to work with and could identify certain easily obtainable measures as indicators of the unmeasurable traits for the outputs, we found that, in contrast to the activity variables, these spanned too short periods for us to be able to seriously relate them with the inputs. Anyway, the Box-Jenkins transfer function model required more resources than were available to us. Possibly the bug reporting and responding process was where an input-output model seemed more likely to succeed but we were frustrated once more. Of the two techniques available, the first, the Box-Jenkins transfer function model, could not be attempted because of lack of resources while the conditions of the second, Queuing Theory, could not be satisfied, Section 4.5.4.2.

Although the models of behaviour at the sub-levels seem to have been important by the way they complemented each other, by the insight they provided and by their usefulness yet they rested mostly on the elementary techniques of averaging, smoothing and plotting. Furthermore, except for the modifications model, Section 3.5.2, they are mostly qualitative in nature. We may know whether a variable is to increase, decrease or remain the same in the near future, whether the peak activity has been reached, whether it is time to start the next cycle, whether there is enough slack in the system to absorb possible activity upsurges and whether, in the error correction process, the point of diminishing returns has been reached but, unfortunately, we cannot pin any values on these.

For analytic models there are probably too many complex factors and relationships to consider. See, for example, the limited success experienced by Thibodeau and Dodson [174] on attempting similar work. Also the common life cycle curves, e.g., the logistic and the Rayleigh curves, the latter so central to the treatises of Norden [134,135] and Putman [145-147], do not seem to quite suit our purposes and, as Parr [38] and Basili and Zelkowitz [9] observed, the Rayleigh curve would be no better or worse than any other.

One type of model which we have used rather extensively are multiple regression models. However, we were acutely aware that, even if they have been extensively used elsewhere [81], they still fall short of the ideal and do not quite satisfy "the need to identify the proper variables of the environment, which will be the prime determinants of the system's performance and to qualify the relationship" as Yeh [187] claims. Rather, they are more likely to depict a black-box situation and consequently may not relate much to reality or tell us anything that is meaningful about it. Nevertheless, they have been successful enough in more than one fashion, notably in attempting to apply Sutherland's ideas about

different types of systems.

6.4 THE DATA USED TO REPRESENT SYSTEM AND ENVIRONMENT INTERACTIONS

Whether the data we have been using to represent environment and system interactions are adequate is open to debate. Should each different classification of a variable be taken as illustrative of a particular aspect of the interactions between the system and its metasystem, then because of the possibility that different factors may not interact similarly with the same part of the system, that the same factor may not interact similarly with different parts of the system and that the interactions between a particular factor and a particular part of the system may change from release to release, we are forced to work on 3 dimensions implying an explosive exponential number of combinations to explore. It is probably not humanly possible to cope with except a few traits, parts and releases.

The only property which could be extracted and studied from these interactions data was diversity. Even then, Shannon's H, as the diversity measure for classificatory data, has its limitations, Section 4.6.6.1.4. Finally, very few values in each of the categories implies that a single unit value change from one category to the other is equivalent to a large percentage change. For such cases, exploring for trends in the categories is not very useful.

6.5 CONCLUDING REMARKS

What we have tried to do in this chapter is to review, in a general sense, the flaws in the data, tools and results and some usefulnesses, too, but we could not consider any particular point on its own. Nor is it possible to do so now. However, as a post-script to this chapter, we shall provide as examples, two short lists, one of points for further exploration and one of results which can be of some applicability.

6.5.1 Some points for further exploration

- 1) Correlation has shown that changed modules are larger than the unchanging ones, Section 3.5.1, that infrequently changed micros are called by many modules, Section 3.4.3.3, and so on. It would be quite an improvement if one could establish which is the cause and which is the effect, e.g., is it because of its largeness that a module is changed frequently or is it because it is changed frequently that it is large.
- 2) It is already acknowledged that both BD and B are somewhat unusual systems, Sections 3.2.5, 3.4.2.1, 4.2.4.4, 4.4.4 and so on. The main questions concerning

them will revolve around whether we can identify which are the more important properties that cause them to be peculiar systems.

3) How to refine our method based upon regression analysis or to replace it by a more adequate one to classify systems according to Sutherland's ideas, Section 4.2.5.2.4.

4) We have used the number of patches as a rough measure for quality, Section 4.2.5.1, daf2's as one for work, Section 4.3, increasing largenesses in component structure values as indicative of increasing complexity, Section 4.2.4.1, and so on, and, more than once, drawn attention to the weaknesses, too. A real step forward would be taken if they could be improved or even converted into all-satisfying measures.

5) Related to 4) is how to improve the work in Section 3.4.2 and extract a simple metric so that comparing distributions can be dispensed with.

6) Then, concerning the models, there is probably much room for improvement: more refined ways of constructing a typical module, Section 3.5.2.3.1, micro, Section 3.5.2.4, or holon, Section 4.3, are required. Then, hopefully, one could start parametrising each entity's model so that it can be adapted to predict the behaviour of all the various types of the entity.

7) On the same subject of model refinement, many other ways than as in Section 3.5.2.7 can be explored for the exponential decrease model.

6.5.2 Some results which may be useful

1) It may be a waste of time to have more than 2 priority categories for the bug reports, Section 4.5.2.7.

2) It may be possible to reduce the number of bug responders by one or two, Section 4.5.4.1.2.

3) Given that people have shown themselves to be better at certain tasks and not so good at others, Section 4.5.3, there is some scope for trying out various possibilities. After monitoring each team, one could attempt to reorganise them so that homogeneous teams with equivalent abilities (and a few super-teams to act as trouble-shooters) can be constituted. More realistically, however, one could try to more closely match future tasks with the existing teams.

4) The models may be useful for giving some broad indications as to whether some future types of action will be opportune or not, e.g., it would be relatively easy to tell, for each phase or release, whether the peak in a certain activity (issuing of software notices, daf2's, daf6's, etc.) has been reached, whether most of it has actually occurred and so on. Thus, some valuable help may be obtained in deciding whether it would be expedient to embark upon the next development stage.

CHAPTER 7

THE WAY AHEAD

7.1 INTRODUCTION

Chapter 5 may have given the impression that the study has been quite successful while Chapter 6 may have had the opposite effect. More probably, the degree of success has been neither high nor low but it seems to have been reasonable, particularly in view of the little that existed before and, therefore, of the great difficulties which we experienced.

While Chapter 6 specifically discussed the weaknesses in each aspect of the data, the analyses and the methodology one at a time, the objective of this chapter is to explore how matters can be improved generally. At the same time, we shall use these discussions to see how various additional ways of analysing the data can be explored. These can be regarded as recommendations. However, we shall make no attempts to suggest which particular aspects of these extensions of our study would be more useful, important or practicable; we shall leave it to the reader to decide for himself. This is because we see our main task as attempting to list the possible complementary and supplementary analyses. Since, as the reader will soon find, there are limitless possibilities, it is for him to decide which of the few, that he will only be able to deal with, he wants to explore.

One of the recommendations that we shall not make is for full scale experimentation. This is probably infeasible, Section 2.6.

7.2 IMPROVEMENTS CONCERNING THE EXISTING DATA

The existing data base may be improved by using:-

- Better measures

We would recommend that the already available data be recorded more accurately and more consistently, that they be quantified as much as possible and that the measures extracted from them be refined. Not only should the recording manner of each type of data be preserved over time but each of these types should be consistent with the others as much as possible, e.g., for exploring their interrelationships, it would help if the activities referred to the same entity. Refinements are needed because crude measures like number of calls or net growth, commonly used by many investigators, including ourselves, cannot adequately represent the subtler and perhaps more important aspects in the data such as the chaining of the calls or the addition and removal of code. If these goals can be

achieved, then it is likely that at least some of the unexplained variations in the models constructed in terms of these measures will be removed.

The best method to achieve these goals seems through the establishment of definitions, prior to data collection and measurement. But, points which particularly concern Section 7.4, one must first identify what one wants to define and measure and, although definitions lead to consistency and, therefore, to the comparability of systems, they must first be accepted, if not universally, at least over the systems studied. Of course, the need for definitions has been stressed countless of times but there has always been more talk than action.

- Longer periods and/or more releases

Should the resources be available, there is no reason why both the equally meaningful pseudo and real time approaches should not be investigated to the same extent. Release-based analysis for B should certainly be explored to a greater extent than we have done, once the number of releases reasonably exceeds the paltry 6 that exist at present. Unfortunately, it could be a long wait before that happens. Using (calendar) time as the alternative domain may provide as many data points as one may wish but it may turn out to be totally useless if they are all cramped within such a short period that the characteristic patterns in the system's behaviour would remain unobservable. With more refined data, one may even proceed to create other types of models, e.g., those based upon execution time!

7.3 IMPROVEMENTS CONCERNING THE EXISTING ANALYSES

Here we mostly discuss, under several brief topics, how we can supplement and complement the already performed analysis to extract even more information from the existing data. Not only does each of these require the analysis to be repeated all over again but they can all be combined together. One therefore quite quickly sees the exponential explosion of possible ways of expanding and refining our work as well as the impossibility of carrying all of them out. We could not regard any of them as a strong recommendation. They may simply be viewed as ways in which already existing or further facets of the data may be explored more deeply.

- More ways of using the aggregation period

There is that of exploring different aggregation periods for the same activity series and that of using the same aggregation period for different activities.

- Investigating each subcategory on its own

Instead of, say, all the bug reports together one may investigate each priority class separately and instead of all the changes together, one may investigate the 66% that took the average time and at most one standard deviation away from the latter to effect, the 17% of changes that took the least time and the 17% that took the most time, all separately.

- More combination of subcategories

E.g., management may be particularly interested to monitor the behaviour of release 5X23's priority A bug reports.

- More levels, entities and subsets of entities

Probably more thorough understanding will be achieved by studying more levels, more types of entities and more subsets of the latter. Ultimately, one may begin to identify the (more) proper level(s) at which to study the system, the proper type(s) of entities and the proper characteristics of each subclass of entities, too.

- Smoothing the data

One may attempt each of several filters to the data.

7.4 ENLARGING THE ANALYSIS

- More properties within the same system

We have already seen that the available data may represent only a small proportion of that required for a truly effective study. One special area of concern is that there is virtually no recording of changes in the environment, whether these are in the technical, economic or social field of the outer environment or in that of the business itself, e.g., organisation, staffing or policy changes. If such data remain unavailable (and it is very difficult indeed to record, let alone perceive, the significant changes in these, specially the outer-environment), the inputs-outputs relationships will stay unknown and largely unmodelled and we may have to continue describing the behaviour of the system as being highly stochastic.

- Other systems

Although a tremendous amount of effort would be required to investigate a single system with sufficient thoroughness, yet it is necessary to extend the study to other systems, too. There may be no basis for comparison at present and, because of the absence of previous results, one may, as we could have been, be somewhat inhibited in the interpretations of one's own

results. However, the what and the how of the comparisons would become more evident once other systems have been similarly analysed. Also, the complementation and supplementation of different systems' results could provide a better perspective for interpreting the various findings and for improving the consequent conclusions, generalisations and theory.

- Other lines of investigation within the same system

With new, more accurate and more extensive data, it may become possible, in some cases, to go further than we were able to and, in others, to even perform new types of investigations. What the latter may be, we cannot elaborate but, about the former, the tools which we were able to use to only a limited extent could find greater applicability while new tools could be required, too. One can expect, as a consequence, a refinement in our methodology and, possibly, the transformation of our qualitative models into quantitative ones.

Of those techniques already used or mentioned, the construction and manipulation of connection matrices may be attempted, perhaps only as a pilot study, on a small component subset. There are also some problems associated with the use of the others: Weibull, Rayleigh, Gamma and other fits, Cluster analysis, Figure of merit method, Delphi Process, differential equations, cross-spectral analysis, Queuing Theory, Box and Jenkins transfer functions and spatial correlations, e.g., deciding which clustering algorithm to select and implement, which weighting factors to use to compute the Figure of Merit. However, there are definite uses for them and there may be some other methods which may be useful, too, to mention a few: Bayesian Theory, Factor Analysis, Maximum Likelihood Methods, Psychology, Forrester's models and simulation.

We mention psychology because of the inevitable involvement of human beings with software systems and because our study hints that there is a threshold in the understandability or complexity of software, whether linked or not to the magic number 7, which is the approximate limit of issues Man can consider simultaneously [124].

But, even if we were able to include all these tools in our kit, it may still not be possible to understand all aspects of the system's behaviour. According to Forrester [62], an actual historical time series is not the best starting point because the situation may be too complex for empirical methods:-

"A successful dynamic model of non-linear, noisy information feedback systems probably cannot be derived by statistical analysis of the overall, aggregate, system performance".

As we ourselves have made the point quite often, incredibly complicated cause and effect sequences where all parts are potentially connected and where an action in one part sets off complementary, sympathetic or compensatory reactions elsewhere cannot be unravelled by the relatively simple tools of Statistics. This is why we sought help from other disciplines. In another effort, instead of only taking a holistic view, we attempted to look deeper into the system. Thus, while we concur with Forrester, we have attempted more than just a statistical analysis of global data. Forrester started from the opposite direction, of course, and while it has its advantages over our favoured approach, it does not allow his models to be built without some a priori hypotheses and we believe that, in the blind, the better start is provided not by the hypothetico-deductive modality but by the empirical-inductive one since it, at least, endeavours to provide us with some initial bootstrap rooted in reality, Section 2.6. Nevertheless, Forrester's models rest upon feedback loops and on the system having closed boundaries, the second of which software systems do not possess and the first of which is very difficult to identify empirically, specially as they seem to affect one another in complex fashions, as in software systems. Therefore, they may not be applicable at present but they must still be kept in mind.

Mathematical models and simulation, if one accepts the bankruptcy of empirical methods, may be the only possible avenue left to explore. (In a way, Forrester's models are truly simulation models). But the argument that to make a start certain hypotheses are necessary, and to have any credibility, they must, at least, be partially empirically based, still holds.

Although we have attempted to bring in knowledge from other disciplines, our effort could not have been exhaustive and there probably remain relevant tools from still other disciplines, about which we may be unaware. This shows that a researcher in our field needs to be a well-rounded person. There are probably very few such rare birds around and this may be why so much still remains to be done. Rather than depending upon them, it may be wiser to follow Belady's advice:-

"... we cannot rely exclusively on computer scientists and software engineers: other experts must also participate. Psychologists, for example, are trained to evaluate complex situations using the rigorous and well established methods of experimental design and statistics. What we, therefore, propose is a multi-disciplinary approach... Experts from sociology and management sciences should also play an increasing role in life cycle related research" [12].

7.5 AUTOMATE DATA COLLECTION AND ANALYSIS

It should be obvious by now that, to make significant headway, the best, if not the only, way to proceed should involve the automation of both data collection and analysis.

7.6 WHERE DO WE GO FROM HERE?

Most of this chapter only seems to show that, despite the great amount of work accomplished, we have merely scratched the surface and that our research has raised more questions than it has solved. For example, there was always the question of whether the variables - whether of structure, activity or exogenous factors, the components - at any level, the releases and the systems are classifiable as good or bad, as important or not important, as stable or not stable and whether we can identify the factors that cause each entity to be characterised as it is. Or which is the best level, aggregation period or clusters to use to study the system.

From what we know, it is difficult to tell whether these are solvable or unsolvable problems. In the first place, quality, importance and stability are relative concepts and ultimately depend upon the observer [139]. In the second, looking at any new level, any new partitioning, as we have seen, is equivalent to a redefinition of the problem and will throw up new results about some other significant aspect of the system. In the last, the time factor needs to be considered as well. If the characters of each object we want to classify (and have classified) change over time like the outside of an amoeba, then a situation equivalent to a taxonomic nightmare results.

We thus come full circle to asking ourselves, once again, whether our techniques are what is needed to crack the problem. If there has been failure to adequately deal with the problem, is it due to the wrong selection of methodology and tools or is it simply that the current state of science is not sufficiently advanced to deal with the study of software systems?

Software systems being a special class of systems, our techniques should be of general applicability for systems study. They are, as Lehman says, the fruit flies [107] of systems study, representing a microcosm of larger and more complex, e.g., economic and sociological, systems. Turning the argument around, we should have been able to use Econometrics and Systems Science techniques. But it is precisely the scarcity of tools from these disciplines that forced us to devise our own methodology.

For example, in Systems Science, following Von Bertalanffy's landmark paper [177] where open systems were first formally described, was a surge of activity where quantisation was attempted but this effort had not been sustained. Nowadays, the majority of what is being written in the subject area is

qualitative or conjectural rather than quantitative [16]. Some of the most significant works in that field have concentrated upon simplistic systems, such as simple geometrical shapes [157], single-goal simple minded organisms [164] and so on. Sutherland's book [170], in contrast, is on a grand scale and has been a great source of inspiration but it still remains an abstract treatise; quantitative techniques are infrequently discussed while not a single result or analysis is reported.

Turning to Systems Science is, therefore, not likely to be very useful. There are then 2 possible conclusions, the first of which is pessimistic and the second optimistic:-

- 1) Since such well recognised branches of Science which have been developing for over 30 years have not progressed very far in solving large systems problems, then we probably have little chance of doing better. Because Systems Science is still immature, we may have come too soon on the scene. Should we abandon hope or wait until the necessary tools are developed and become available or should we simply plod on? Such shall be our dilemma.
- 2) Previous systems study have floundered because the rates of evolution of the systems studied have been too slow for an adequate number of generations to be studied. Now, that software systems, the fruit flies of systems study, have become available for proper work to be envisaged and that we have been able to list some of the problems, here, we hope that, jolted, others, more qualified than us, may now be persuaded to join in our efforts and in the fun, too.

CHAPTER 8

A THEORY OF PROGRAMMING SYSTEMS' STATIC AND DYNAMIC BEHAVIOURS

8.1 BUILDING THE THEORY FROM EMPIRICAL RESULTS

Most of the work that deals with software systems and/or their environments falls into one of these 3 categories:-

- 1) The descriptions of the author's insights and thoughts.
- 2) The mathematical models.
- 3) The empirical results.

2) is really a more formalised form of 1) and both may be classified as hypothesis making - a step along the way of theory formation. However, with the latter, it is easy to let the imagination run wild, Section 2.4, while with the former, one may err by making too simplistic assumptions, probably so that mathematical manipulation may become possible. Even then, the derivations may be so complex that the only absolute conclusion one may draw is that mathematics, the most sophisticated of all tools, is still too elementary for the problem at hand. Kac [84] describes abstract models thus:-

"Models are, for the most part, caricatures of reality, but if they are good, then, like good caricatures, they portray, though perhaps in a distorted manner, some of the features of the real world.

The main role of models is not so much to explain and to predict - though ultimately these are the main functions of science - as to polarize thinking and to pose sharp questions. Above all, they are fun to invent and to play with, and they have a peculiar life of their own. The 'survival of the fittest' applies to models even more than it does to living creatures. They should not, however, be allowed to multiply indiscriminately without real necessity or real purpose.

Unless, of course, we all follow the dictum, attributed to Oswald Avery, that 'you can blow all the bubbles you want to provided you are the one who pricks them'".

The construction of abstract models thus provides a formal structure which, although idealised, is, nevertheless, consistent and generates hypotheses which can, indeed must, be subjected to verification by comparison against observational data and experimentation. In contrast, empirical observations are in search of solid theoretical foundations but, all too frequently, the close logical connection between the purely theoretical and the applied models has been masked by the fact that they appear so markedly different. This is how Leontief

[112], in his Presidential Address to the American Economic Association, describes the situation facing Economics:-

"But I submit that the consistently indifferent performance in practical applications is in fact a symptom of a fundamental imbalance in the present state of [economics]. The weak and all too slowly growing empirical foundation clearly cannot support the proliferating superstructure of pure, or should I say, speculative economic theory".

Our discipline, although comparatively very young, is in danger of being similarly polarized. One way of bridging the gap is to empirically verify the abstract model that one has constructed (see also Section 2.4). Because abstract models can be so easily constructed, this usually results in a more restricted abstract model which can again be subjected to empirical tests, and so on. We again quote Leontief [112]:-

"True advance can be achieved only through an iterative process in which improved theoretical formulation raises new empirical questions and the answers to these questions, in their turn, lead to new theoretical insights. The 'givens' of today become the 'unknown' that will have to be explained tomorrow".

One may also start at the other end (see also Section 2.6) with the applied models which are usually more restrained than the theoretical models: their functional form must be specific, the parameters realistic yet estimated from real data, the number and complexity of the arguments restricted by the problems of statistical estimation and model application and so on. From these empirical models, one may attempt to extract a theory that will be broadly accepted. This will be iterative between the form, i.e. the broad framework obtained by interpreting the empirical results, and the content of the theory which is to be matched to that form. If during the theory formulation process, (one's preferred) content is found to conflict with the form, then it will have to be amended and/or augmented. However, one may also have to consider the modification of the form itself: reinterpretations, retestings or further tests may be required. Two things can be said about the initially chosen form, with which the content is to be associated:-

- 1) It does not have to be totally correct, but only reasonably adequate to permit the content to be developed.
- 2) Without it, the iterative process needed to develop the theory cannot proceed.

As there has been many more (untested) ideas than results in our field, it has always been understood that our approach, as that of ED, was to be empirically-based and that we had to work our way through any available data to construct our models. And

we were fully aware, right at the beginning, of the difficulties in extracting a theory therefrom (Section 2.4 and Leontief's first quote above), yet it is one of the requirements of our task to endeavour to do so. In formulating our theory, we have attempted to follow the iteration process between form and content as described in the preceding paragraph but we have more probably made quantum leaps on occasions, specially where evident holes can be found to exist. In this, we ask the reader to be understanding and to bear with us. The remainder of this chapter (of this thesis!) describes what we have been able to come up with.

8.2 A VERY GENERAL MODEL

So that there may be no misunderstandings, let us state right away that the model which we shall present is in very general terms only. There is no attempt to identify the exact forms of the transformation functions and the individual variables to be considered. At the present stage, it is probably impossible to do either and to attempt to be all-encompassing would be a titanic task probably requiring a lifetime of labour. But these are not our immediate concerns. Although we do recognise that there is quite a gulf between the exploratory state of our model and one that can be used for predictive purposes, we insist that the latter cannot be formulated at present and that its formulation must be preceded by some tentative theory to compactly account for the observations in the subject matter. This is what all our work has been aimed at and the ensuing general model has the main purpose of providing a framework around which most of the ideas that we have been entertaining and that are vouched by our results, those that exist in ED theory, and some of the relevant ones in systems theory and some topics of software engineering may crystallise. Indeed, the model is not really the main contribution of this chapter, it is just the means to an end, that of allowing the discussions to proceed smoothly.

As may be recalled, there were 3 aspects to our investigations: that concerning the properties of the system, that concerning their dynamics and that concerning the dynamics of the inputs or exogenous factors. Although how systems look like is of primordial importance, since we must understand the form before we can understand the behaviour, yet our real interest and that of ED is in dynamic concepts, with the focus shifted from form to function, from organisation to action and from being to behaving. Thus, time is a fundamental element in the model. We have chosen to represent the model in the discrete-time form because it appears simpler to formulate and to describe. It is also an easy task to derive the continuous-time form from it. A more important reason is that each discrete time point can be taken to represent a specific release so that, whether one prefers the real-time or the pseudo-time approach, the model is

equally appropriate. Indeed, the discussions, which do not deal with specifics, are applicable to both cases. For convenience, we shall use the term "time" rather than "release".

Let us assume that, at any given time, the system state (or how the system looks like) is representable by a set of state conditions or coordinates which may be called the state vector. The ultimate aim of ED is, then, to be able to predict the state of the system at time $(t + \Delta)$ from that at some time t .

The values of the elements in the state vector, at any given time, may depend upon:-

- 1) Those of the other elements at the same time.
- 2) The previous state(s) of the system.
- 3) The inputs and their values from time t to time $t + \Delta$.
- 4) The transformation process itself.

Just as the state coordinates can be taken to represent the system's relevant instantaneous structural properties and traits, the inputs will represent those of the effective environment. Then, our general model to represent the dynamics of a system's structure can be expressed as follows:-

$$\underline{s}(t+\Delta) = f(\underline{s}(t+\Delta), \underline{i}(t), t+\Delta, t)$$

where

$\underline{s}(t)$ is an n -dimensional state vector which represents all the relevant endogenous variables at time t .

$\underline{i}(t)$ is an r -dimensional input vector representing all the relevant exogenous variables from time t to time $t+\Delta$, as well as lagged values of $\underline{s}$.

f is a non-linear n -dimensional vector function.

Thus, for $s_j(t+\Delta)$ where $j=1$ to n , we have

$$s_j(t+\Delta) =$$

$$f_j(s_1(t+\Delta), s_2(t+\Delta), \dots, s_n(t+\Delta), i_1(t), i_2(t), \dots, i_r(t), t+\Delta, t)$$

Before going into the details, let us briefly make 3 comments:-

- 1) It is necessary to point out that $\underline{i}(t)$ can be partitioned into 2 parts, the first of which represents genuine inputs, $\underline{e}$, into the system during the time interval considered and the second being lagged values of $\underline{s}$. We thus have a set of difference equations which causes the time path of $\underline{s}(t)$ to be determined by the changes in the exogenous factors as well as by the dynamic structure, repre-

sented by both its lagged and actual values, of the system itself.

2) Many of the $s_k(t+\Delta)$ and $i_l(t)$ in f_j , $k=1$ to n , $l=1$ to r , will have zero coefficients, signifying that they are not important in influencing the value of $s_j(t+\Delta)$. This a necessary condition, otherwise, it would be a case of everything depending on everything else with a vengeance!

3) Some may complain that the model is too general, so general that it caters for all occasions. They may be justified in their complaint but each of its aspects is necessary in explaining the observations we have made.

8.3 THE STATE COORDINATES

Since there must, as pointed out previously, be structure before there is a system that can have behaviour, we shall start, as in Chapters 4, 5 and 6, by discussing the static structure. Defining the system in terms of state conditions is subjective to the extent that one always has a choice among the probably unlimited number of properties to study yet, there must be a minimum, though unspecified number, n , of properties whose specific forms can be taken as values of the variables $s_1, s_2, \dots, s_n$ such that the specific forms of all the properties at any time are uniquely determined by these n properties at that time, the state coordinates.

8.3.1 The state coordinates refer to measurable properties rather than traits of the system

Conceptually, the model is the same whatever the state vector that we wish to explore and there are probably quite a few different ways of constituting the state vector, too. The state coordinate could be in terms of system traits or of more readily accessible and measurable properties. Examples of traits are quality, power, efficiency, understandability, robustness, modifiability and complexity while true system properties would include the degree of homogeneity among the structural properties, the degree of symmetry in the distribution of system components, the density with which these are distributed. We have already, Sections 2.8 and 6.3.3, expressed our pessimism about the measurability of the traits. At present, such properties as we have listed above are probably not measurable as well but, as they are devoid of subjectivity, there is at least some hope that, in the not too distant future, satisfactory measures for them will be found. Until that happens, we propose to discuss $s(t)$ in terms of size, number of communication links, data bindings [8], data elements and so on, properties that can be readily measured, though not reflecting the situation as accurately as one may wish, Sections 2.8, 4.3 and 7.2 among others.

Of course, several of these properties may map on to each trait, each mapping perhaps on to several of the latter. Determining the actual mappings and assigning a value to each property, even on a classificatory scale, so that the value of the trait can be calculated, as done by Boehm et al. [28], seems both hazardous and arbitrary and is tantamount to recognising the quantifiability of the traits, yet it must be admitted that this may be the best that can be achieved in the present state of the art. Because of these difficulties and because the problem of converting our abstract model into something more concrete probably represents a mammoth task on its own, it would be better to start with the easily measurable properties we proposed to discuss.

8.3.2 How the state coordinates contribute to the system's static complexity

We shall return to discussing the tractability of s and its influence upon the degree of randomness that one encounters during empirical investigations in Section 8.10. For the moment, let us see how some of the different aspects of the static system state contribute to the system's static complexity. (Other traits may be discussed along similar veins but as there may be no end to it, we have limited ourselves to one only, that which has most intensely haunted most of us: complexity.)

First there is the question of the absolute size of the system. Whether it affects the order of s , i.e. the number of elements in it, is conjectural but it seems reasonable to assume that the values of the elements in s will be larger for a large system than for a similar smaller system, e.g., all other things being equal, a larger system will have more components than a smaller one. Hence, size will directly affect the (static) complexity of the system. So would the average component size, number of data elements and so on. Such deductions seem borne out by the facts. BD, a small-sized system, exhibits none of the behaviour associated with complex systems while OS/360, probably the largest studied within the ED context, has all the trappings of a very complex system, indeed.

Then there is that of the organisation, form and being of the system. As one may anticipate, the architecture of the system must have a profound influence upon its level of complexity. Segmented systems where all the parts of the system tend to be similar in both structure and function and, more importantly, are virtually independent of each other and communicate and, therefore, influence each other's behaviour only through narrow channels tend to be simple. They also tend to evolve through replication while the new parts minimally affect the complexity of the existing parts and of the whole. In general, one can say

that they are less expensive to manage and control. A hierarchical system, in contrast, is a highly organised and differentiated whole where each part has a definite function to perform. Usually, the chain of command from the top to the bottom is not only complex but no two chains are similar. Removal of any old part or addition of a new one may, therefore, be a complex operation, specially when that part communicates with several others, directly or indirectly, and when it is nested in the lowest levels. That a deeply-nested piece of code is difficult to understand may be due to the inability of human beings to remember more than a sequence of about 7 [124] while complex links between various parts of the system exist because, although we profess to build our computer programs and our software systems in a tree-like structure, yet we always allow a module to be called by several others so that we end up with semi-lattices, which are much more complex structures. For example, while a tree on 20 elements can contain at most 19 further subsets, for a semi-lattice, there can be more than one million different subsets. Thus, the more the components are allowed to communicate with one another, the more will the structure depart from that of a tree and the more significant will be the contribution of that aspect of the structure to the complexity trait. Of course, even more complex systems, e.g., that where each component communicates with all the others, do exist, but the two that we have dealt with, the former being typical of data processing systems like BD and CCSS and the latter representing the operating systems and so on seem to cover most cases concerning software. We have quite adequate evidence that BD does indeed behave like the segmented systems described above while, for CCSS, the non-increasing "fraction of modules handled" variable ties up nicely with the behaviour expected from systems built along the same lines.

About homogeneity, it is probably true to say that the greater is the diversity of the parts by each structural property, the less homogeneous are they by that property and the greater will be the contribution of that factor to complexity. As is obvious, the greater the medley of components, the disharmony of the design, the multiplicity of missions the software has to carry out, the greater is the diversity over each of these aspects of the system. But diversity or variance (one may also talk in terms of density) in the distribution of the parts is but one factor. The actual values as well as the type of distribution which they assume are also important complexity factors. Being humanly engineered - artifacts - the systems probably follow the 20-80 law and, therefore, tend to exhibit a simplicity facet to each of their properties, but, paradoxically, because of the same artifact property, no part has either uniformly large or uniformly small properties. This further aspect of structural heterogeneity - the degree of concordance of the parts over different properties - is yet another contributor to the complexity trait. Then there is also

that of functional homogeneity. Dogmatically, one could insist that there can be no such thing since each component of a software system has quite a specific and unique task to perform. But there obviously are some components which perform more or less similar tasks and so on so that one may always find ways to classify the components functionally, although this seems subjective and dependent upon what one aims to achieve.

Other properties which may be considered for inclusion as state coordinates could be the methods used for parameter passing, the scope of the global variables, the amount of documentation, whether the interfaces among the components are well defined and closely protected, whether the program is modular, structured, written in a high level language and so on. Most of these have been intensively studied by many of the most eminent researchers in the field and should therefore be carefully considered as suitable state coordinates although there are obvious problems. However this is another matter. Since it is generally acknowledged that structuring and modularising the system, rigorously defining and constraining the interfaces among the components, minimising the use of global objects, among others, all contribute to reduce the value of the complexity trait, then an increase in their corresponding state coordinates, in our model, would be equivalent to an increase in complexity while a decrease would indicate a decrease in complexity.

8.4 THE ENVIRONMENT

8.4.1 The reasons for including the environment in the study

The second aspect of the model relates to the nature of the environment. Most researchers in our field, like those in other fields, have completely ignored the environment and its possible effects upon the object of their interest.

"In the realm of social theory there has been something of a tendency to continue thinking in terms of a 'closed' system, that is, to regard the enterprise as sufficiently independent to allow most of its problems to be analysed with reference to its internal structure and without reference to its external environment," [58].

As we have said above, the failure to include the environment in most studies has not been restricted to social science. Why it has stretched over most branches of science could be attributed to 2 main factors:-

- 1) The influence and success of Physics as a science is so strong that scientists from other disciplines tend to imitate its techniques, yet we fail to understand that its

very success is precisely due to its limiting itself to the study of simple and closed systems.

2) "Context (a name I prefer to environment because it sounds less like a thing separate from ourselves) is the hardest thing to perceive, because it includes us, our ways of thinking. The fish cannot see the water. 'It' is the source of change, of unexpectedness, the real generation of newness, designs of evolution," [83].

In closed-system studies, anything which is unaccountable by the internal properties has been attributed to error variances.

"According to this conception, they should be controlled out of studies of organisations. From the organisation's own operations they should be excluded as irrelevant and should be guarded against. The decision of officers to omit a consideration of external factors or to guard against such influences in a defensive fashion, as if they would go away if ignored, is an instance of this type of thinking. So is the now outmoded 'public be damned' attitude of businessmen toward the clientele upon whose support they depend. Open system theory, on the other hand, would maintain that environmental influences are not sources of error variance but are integrally related to the functioning of a social system, and that we cannot understand a system without a constant study of the forces that impinge upon it," [85].

In fact, systems theory started by dealing with the analysis of internal processes in organisms or organisations relating parts to whole through close-system models. However, when wholes had to be studied and related to their environments, these types of models soon proved inadequate and, inevitably, they gave way to open-system models, first proposed by Von Bertalanffy [177]. We contend that, since software systems are inanimate objects and therefore cannot self-induce the changes to their compositions, it is a tautology to state that the latter can only occur via external agents. Thus, to understand how a software system evolves, one cannot exclude its environment and its interactions with the latter from a proper study: it must be approached as an open system.

8.4.2 The system-environment boundary

Having decided to treat software systems as being open, there is a need to identify its environment as well as the system-environment boundary. Quite often, both of these are hazy and how and where to draw the line between the system and its environment will depend upon the special factors in each individual case. Sometimes it may be quite an arbitrary decision. Generally, depending upon the scope and purpose of the investigation, it may be drawn anywhere.

In some cases, such as those we are dealing with, however, there is little difficulty in drawing the boundary. Thus, the software system (and perhaps all its documentation too) is the system being studied and all the rest is the environment.

8.4.3 Some interesting environmental properties

One can perhaps define the environment as a set of elements or components which affect the system, although they lie outside the system. Thus, one implicitly recognises that, because of the effect of the environment, the software system will continually evolve (require maintenance) even if its first instance was perfectly conceived, designed and implemented, Lehman's first law, Figure 1.

Some relevant questions concerning the environment could include:-

- 1) What are the components of the environment? What are the most important characteristics of the environment? How protean is it? Is it placid or turbulent? Are the components highly differentiated, autonomous?
- 2) With what is the system interacting?
- 3) What are their key characteristics in terms of quantity, size, strength, objectives, desires, stability, predictability, capabilities, tractability, etc.?
- 4) What are the more general features of the environment? What is the state of technology and of fundamental knowledge? What are the other systems with which the system must coordinate, the geographical, political, legal, technological and social economic factors, and so on? What are the trends in these?

One method to classify the environmental factors is into the economic, geographical, etc., classes as in 4) above, but, for our model, it is more important to recognise that there are some which directly influence the system's behaviour and hence must be included in e while the effect of the others can be indirect or/and remote to any degree and hence need not be in e. One can therefore view the environment as being multi-layered where a factor belonging to one layer or level may directly influence not only other factors within the same layer but factors in the next higher and lower layers, too. For most of our discussions, we shall not distinguish among the layers which, anyway, probably form a continuum and, therefore, are not separable from one another, though, in Section 8.6.1.1 we shall divide the environment into 2 parts: the task or maintenance environment which can also be identified as the metasystem (those factors relating to questions 2) and 3)) and the wider or outside world environment, the meta-metasystem, (those factors relating to question 4)). There is probably a case for going further, too, by subdividing the task environment into the project which designs, programs, tests and

validates the software system and that which manages the former. Of course, the boundary between the meta-metasytem and the metasytem is arbitrary to a certain extent. It is also much fuzzier than that between the latter and the system itself.

An alternative method of classifying the environmental inputs is to divide them into progressive and anti-regressive activities [95]. The group of latter activities can be taken as the anti-thesis of that of the former in the sense that its purpose is to combat entropy while increased complexity is probably the usual net effect of progressive activities. Translated in our model, progressive activities will cause the values of the state coordinates which they affect to increase while the anti-regressive activities will have the opposite effect. If the expression is linear, then those elements of e which represent the progressive activities will have positive coefficients while those which represent the anti-regressive ones will have negative coefficients. Thus, to simplify matters, we need only discuss one of these two types of activities. What follows is in terms of the progressive ones since they are much more common, anti-regressive activities being undertaken rarely, if ever at all.

The effects of the environment characteristics upon the system's static complexity are analogous to those of the state coordinates, i.e. they may be discussed in a similar manner as $s(t)$, since given the same interfaces, a simpler environment would generally imply simpler and fewer influences upon the system, simpler evolutionary behaviour of the system and, possibly, a simpler system, too. Indeed, small (simple?) systems seem to be associated with small (simple?) organisations/environments while the contrary is true for large systems, e.g., contrast BD with OS/360. For large systems, there is a further complication - the maintenance team (metasytem) may be geographically scattered over several locations. Also, an environment where the factors at any time are homogeneous or concordant, however turbulent, or a placid environment, however complex, is easier to understand and its influence upon the system more readily allegorised than one which is heterogenous, even if somewhat calm, or turbulent, even if small. Indeed, when the environment is completely homogeneous, i.e. when all its components are similar and behave and, therefore, influence the system, similarly (e.g., the building societies in BD's environment), then its effect upon the system is considerably simplified, even if the latter is totally permeable to environmental factors. Also, a totally placid environment where all of its components are unchanging over time may be ignored but the dynamics of inputs will be more fully considered in Section 8.6.

8.5 INTERFACES WITH THE ENVIRONMENT

The next focus of interest concerns the interfaces which the system maintains with its milieu. Interfaces may be defined as the points where exogenous determinants pass into the system and where endogenous forces pass into the environment.

8.5.1 Not all environmental factors are inputs

The first question that one may ask how open the perimeter of the system can be to influences arising from the environment. It is obvious that the extent to which the system's interchanges with its environment are constrained and selective has a large part to play in determining its complexity. The selectivity of the interfaces is their ability to distinguish acceptable from unacceptable inputs and outputs, and to let pass, in either direction, only those that have been deemed desirable. The reception of inputs into any system must be selective to some degree since not everything can be totally absorbed, e.g., the digestive system of living organisms assimilates only those inputs to which it is adapted. Similarly, all systems, in general, must be attuned to only some specific inputs. If the system is highly dependent upon external events, then it may be called an open system. A closed system, on the other hand, is one whose behaviour is totally determined by its own properties. Of course, there is no such thing as a truly closed system. Nor is it possible to imagine an object which is so decadent as to be permeable to any input. Thus, openness and closure represent only relative concepts.

It is our contention that the analytical tractability of any system's behaviour is also dependent upon the extent to which it is master of its own behaviour. One whose perimeter is well defined and relatively impervious to environmental intrusions will be minimally affected by external influences so that its interactions with the latter will, in general, be easier to understand than one whose perimeter is more or less "open" to environmental or exogenous influences, specially when these are themselves complex. Furthermore, the more open is the system, the greater is the difficulty of identifying the external forces since at least some of them may not be visible as they may be far removed in time and/or space and so on. This aspect of closure or openness is represented in our model by the order of $\underline{e}$. If $\underline{e}$ has few elements, then the system being represented will be more closed than one where it has quite a lot of elements. If the order of $\underline{e}$ in any f_j is zero, then the system will be totally closed and the model reduces to

$$\underline{s}(t+\Delta) = \underline{f}(\underline{s}(t+\Delta), \underline{l}(t), t+\Delta, t)$$

which means that the system's state is some calculable function of its present and past states only. (Some good examples of totally closed software are the mathematical subroutines where

$\underline{s}(t + \Delta) = \underline{s}(t)$ for all t but, of course, these are not worthy of our attention.) If that of $\underline{e}$ is infinite (or those of $\underline{s}$ and $\underline{l}$ are zero), then it will be totally open. (In case one is looking for a measure of openness/closure, it is perhaps more meaningful to consider the ratio of these two orders.)

8.5.2 The nature of the interfaces

So far we have only considered openness or closure with respect to the quantity of inputs which may affect the system. But there is one other aspect of the interfaces which we need to consider. It is that the system not only selects its inputs but, for each input, not all of it is absorbed; there is always some portion that is dissipated. It is as if the system attempts to maintain itself within well-defined, tangible bounds, and to protect itself against the incursions of unpredictable or uncontrollable forces arising from the environment. These properties are incorporated in our model within the transformation functions, $\underline{f}$, which may be of any form. For simplicity, let us assume that the functions are all linear. Then, if, for any s_j , the coefficient of an input in its f_j is small, one may say that most of the input's influence is being dissipated rather than being absorbed. But one should remember that the same input may have a large coefficient in the expression of s_k , $k \neq j$. When all the inputs have large coefficients, the behaviour would be determined mostly exogenously and, as in Section 3.2.3.1, we could say that the system does not learn from past events (at least in that particular aspect of its properties). Of course, the same sort of reasoning applies if the functions are more complex than linear but, since very complicated relationships among the factors in each expression must be provided for, there is a chance that the dissipation/absorption effect might not be quite apparent.

8.6 THE DYNAMICS OF THE EXOGENOUS FACTORS

It is time to consider the fourth aspect of the model which concerns the dynamics of the inputs. It seems that however the exogenous factors may interact among themselves, this needs hardly concern us since identifying and measuring the relevant inputs from the environment to the system at any given time and seeing how they affect the system but not each other are all that we need. Unfortunately, one may not always identify the complete input set nor, as our investigations show, is there any simple and accurate way of representing the inputs as functions of time. Each input, more often than not, exhibits neither patterns of seasonality and cyclicity nor any trend. It was also not an easy task to positively identify the feedback effects of a variable, either upon itself or upon other factors. Indeed, it seems that each input's dominant feature, at least when viewed at the global level, is that of

stochasticity. Lehman [18] has previously interpreted this situation of no recognisable pattern as "statistical invariance" or constancy. This, we hasten to add, in no way conflicts with our notion of stochasticity. Stochasticity, however, seems to be a more precise term, despite its possible-pejorative undertones, since like invariance, it implies a certain average among the values but it goes further by asserting that these are not predictable, except, perhaps, as lying within some broad range.

8.6.1 The need to model the environment

That randomness is the main feature of our observations is probably due to the fact that each factor's behaviour was mainly examined at the global level and as a function of time and, hence, other factors excluded from its analysis except when correlation and regression analyses were applied. If the unaccountable portion of the behaviour is small, then it is not a tragedy to dismiss it as due to error variations. This is, however, no longer justifiable if it is as large as our results have implied, unless we are, of course, prepared to accept and to state that the variables affecting each input are so great in number, so inaccessible and so complex in their interactions that we have no choice but to consider the behaviour of the inputs as subject to chance.

To more accurately predict the inputs than can be presently done, then there seems no escape but to try and understand what affects their behaviour and to model each of them as functions of those factors from an outer layer of the environment which are in part responsible for its behaviour as well as of the other inputs and their lagged values. Thus, $\underline{e}$ can be represented by a model similar to that for $\underline{s}$:-

$$\underline{e}(t+\Delta) = \underline{g}(\underline{e}(t+\Delta), \underline{i}(t), t+\Delta, t)$$

where $\underline{i}(t)$ now represents the factors from the outer layer together with the lagged values of $\underline{e}$. In the case of system B, the elements of $\underline{e}$ could be dafs, bug reports, software notices and so on while the elements of $\underline{i}$, apart from those already in $\underline{e}$, may include management decisions, user requirements, business considerations, the state of technology and so on. These, in effect, are the factors (the inputs from the outer-environment) that affect the working of the maintenance team, the new object of study.

8.6.1.1 The multi-layers of the environment

It is fairly evident that the state of the technology influences users' requirements which influence management decisions which, in turn, influence the functioning of the maintenance team and the actual modifications to the system.

Thus, as pointed out in Section 8.4.3, the environment can be viewed as being multi-layered, each layer obtaining its inputs from the layer immediately above it, ad infinitum, and representable by our general model, too. Our model is therefore recursive, all aspects of each layer describable and interpretable as previously, Sections 8.3 to 8.5. It, however, seems unnecessary to distinguish more than 2 layers: that immediate to the system, the task environment, and that representing the rest, the outer-environment, or, at most three, if the organisation that manages the task environment is a layer that is inserted between the afore-mentioned two. Notwithstanding the common problems of identifying the relevant input factors, of determining the exact functional forms, of using readily identifiable properties instead of the traits we are interested in, because of the increasing fuzziness of the inputs, outputs and interfaces, it becomes more and more difficult to model the outer layers of the environment.

8.6.1.2 Recursiveness is also applicable to the system's multilevels

A further similarity between the software system and the environment models is that recursiveness is not restricted to the multi-layers of the environment. The system itself is divided into several levels, consisting of groups, subsystems, individual components and so on, each of which satisfies the definition of a system in its own right, whatever that may be [2]. Each of these units is not only worthy of study on its own but is capable of providing us with quite meaningful results, too. The classic examples in Physics (matter, compounds, molecules, atoms, sub-atomic particles such as electrons and so on) and Biology (living organisms, organs, cells, and even finer structures) prove the point. In a less spectacular manner, so do our results, which, in addition, show that, as may be anticipated, the lower the level, the simpler is the unit and its behaviour too. In fact, the latter gets more and more deterministic so that it would be easier to model the behaviour of the micro or holon rather than that of the system. This is relatively easy to explain: each component is small and, hence, simple, does not consist of subcomponents which may be heterogeneous in their properties, etc. But it still remains to be seen whether we prefer to look at the trees rather than at the forest itself. There may be a case for constructing these simpler models since it could well be that, in the same way as our analyses at the sub-levels were helpful in understanding the behaviour at the global level, subcomponent models could be useful in the construction of the system model, too.

8.6.2 The effects of the environment's dynamics upon the system's complexity

We now return to the dynamics of the input factors. However

allegorisable and exactly predictable the inputs may be, the fact is that, in addition to their static properties such as quantity and homogeneity (already dealt with under the section concerning the nature of the environment) and, apart from the nature of the interfaces, the dynamics of their properties have a very substantial part to play, too. The frequency with which events occur in the environment, the rate of change of each input will determine the degree of placidity or turbulence of the environment and, hence, that of the system.

8.7 THE DYNAMICS OF THE SYSTEM'S STRUCTURE

We now come to the dynamics of the system structure, i.e. the changes that take place on the structural dimension through time. We have already seen in Sections 8.3, 8.4 and 8.5 that the static complexity of a system depends upon its own properties, those of its environment and those of the interfaces it maintains with the latter. How a given system state at time t translates into a different one at time $t+\Delta$ also depends upon these same properties and the changes in them. The greater the change exhibited by any of these during any time interval, the greater will be the rate of change of the structural properties, hence the more important and complex the dynamics of the system. Then, complexity on a dynamic dimension depends upon the rate at which the structural properties change. Although a statically simple system existing in a turbulent environment may exhibit more complex dynamic behaviour than one which is statically complex but which exists in a placid environment, yet one could imagine that simple systems in extremely turbulent environments and complex systems in placid ones would be rare. Anyway, the former type of systems would soon become complex (B is perhaps an example of a system which has not yet done so) while the latter will soon become obsolete.

Let us now take a look at some aspects of the model which we have not yet discussed.

8.7.1 Interdependencies among the structural properties at any given time

It must be noted that, for each $s_j(t+\Delta)$, $\underline{s}(t+\Delta)$ is included in its expression to allow for the possibility that, at any time $t+\Delta$, $s_j(t+\Delta)$ may not be independent of $s_k(t+\Delta)$, $j \neq k$. Should the state coordinates all be independent of one another, then there would be no need for including $\underline{s}(t+\Delta)$ in each expression, thus considerably simplifying the model but this is unlikely. Also, should a given state coordinate be exactly correlated with another, then we can simplify our model, once again, by reducing the state vector, and, hence, the number of expressions in the model, by 1. Unfortunately, as our work has shown, the state coordinates are neither fully dependent upon

or independent of one another. It is obvious that the nature of the prevailing relationships among them has a direct effect upon the complexity of the system (and upon that of the model, too).

8.7.2 Feedback effects

The studies of Forrester [62] show how one may borrow feedback concepts from Control Theory to successfully build dynamic models of behaviour. Lehman, in several of his papers, resorted to feedback loops, both positive and negative, to explain some of his concepts, e.g., the provision of computing facilities which lead to more sophisticated applications, hence to more demands for facilities and hence to more facilities. However, the problem is that, although feedback loops do seem to exist in software systems and that the very stability/instability in a system may depend upon them and the time lags involved, yet feedback effects are among the most difficult things to measure. We have to decide whether the very weak or non-existent auto and cross correlations which empirical investigations may show (and have shown, according to our results) are to be interpreted as demonstrating the absence of any feedback or whether they indicate that the process is simply too complex and not empirically tractable, i.e. we have to decide whether software are open-looped or complex closed-looped systems. An open-loop system does not learn and cannot draw from its experience, i.e. the present state of the system is independent of its past states and, at first sight, this seems to fit with our results. In closed-loop systems, past behaviour does affect future behaviour. However, should there be even a few such loops intricately intermingling with one another, then empirically, an examination of the resultant output gives little hope of discerning them. That this may reflect the true situation is provided by several indications (notably the waves in the growth processes of OS/360, CCSS and BD and that management must surely draw some lessons from the past), most of which may not be of any great significance, but which together add much weight to the interpretation. Therefore, former states of g must also be accommodated in the model. We have not specified how many lags we need to consider. Although we may need to go back as far in the past as is necessary, in practice, only a few lags will be all that is required.

8.7.3 The smoothing of the individual effects at the global level

That the resultant from several loops affecting each other is not liable to provide meaningful interpretations brings us once again to the notion that a too macroscopic view may have been taken, Section 8.6.1, and that a more detailed analysis is required. Yet, paradoxically, this does not seem possible.

Although, on its own, each input factor or system component may be studied with increasing detail and accuracy (as we have done) yet, any attempt to relate them together causes the situation to get hopelessly out of hand because their interactions seem so complex. Consider the ridiculous example of an army of ants, each of which can join in and leave at any time to pull and push with varying strengths and in every direction on a crumb of bread. While the behaviour of a typical ant is relatively easy to describe: either it is in contact with the crumb or it is not, when it is in contact, it is either pulling or pushing, the strength with which it pulls or pushes depends upon its own weight, size and so on, that of the crumb is not so, its net movement can only be described as random, unless one is able to relate its movements in terms of those of the individual ants. In the same way, when long term goals have not been clearly set, when management is reactive and driven by events and when uncoordinated practices exist (and in a plural task environment where several teams co-exist, it is indeed very difficult to coordinate, say, the changes made to the system), everyone works on his own and does what he thinks is best for the system, and the state of the system remains "invariant". (Incidentally, it is also well known that localised efforts at increasing a system's efficiency has very often quite the opposite global effect.)

The net global result of all these multi-directional efforts, randomness (or stability or invariance), is due to the individual effects cancelling each other out, unless, of course, when they are harnessed and applied in the same direction. Then, some sort of dynamics is created. As each individual contribution is only a tear drop in the ocean, each doing only a mole's job, this explains why, once ways of doing things have been set up, it is only with extreme difficulty that old ways can be eradicated and replaced by new ones: one sees little change in the movements of the bread crumb if a few more ants join in the pulling and the pushing. It is evident that the more there are individual factors contributing to the net global effect, the greater will be its dynamics or inertia and resistance to change.

It is almost a tautology to conclude that, if we need more accurate predictions, then more complex models are required. However, we should not grow despondent, if we have to settle for the simpler but less accurate global model: many branches of science, not only those where human beings are involved, are in the same boat, e.g., the predictions of meteorology, for all its great importance, still lag far behind those of physics.

8.7.4 Different models may be required for different parts of the system

Being restricted to a global level of analysis may be a

blessing in disguise. We have seen, Section 8.6.1.2, how subentities, at any level, are systems in their own right and are, by themselves, worthy of study. Do we, then, take a look at each level of the system? Can we really afford to?

But different models are required not only to study the depths of the system but its breadths, too. At any level, there may be such a degree of plurality among the components that any method of clustering is legitimate and conducive to meaningful results. Do we, then, attempt to create a model for each of these clusters? The number and variety of such models will very much depend upon the homogeneity among the various parts. That they all exist within the same environment ensures that, to a certain extent, they are subjected to much the same sort of inputs, but their behaviours, as should now be obvious, are also controlled partially by their endogenous properties, which may not be the same since they have different functions, have different sizes, are differently built and are managed by different teams, among other things. Now, it is easy to imagine that an attempt to increase the efficiency trait of one part of the system, although it will unaffected most of the rest, may result in a decreased efficiency somewhere else. The change is then called functional with respect to the efficiency trait of the first part, dysfunctional to the second part and nonfunctional to the rest of the system.

More generally, nonfunctionality with respect to any trait or part of the system occurs if the change is not causally relevant to the trait or the part. Otherwise, it may be either functional or dysfunctional. Functionality or dysfunctionality can also be with respect to a trait or a part. A change is functional (or dysfunctional) with respect to a trait if it functionally (or dysfunctionally) affects that trait, whatever the part considered (or with some relaxation, over the majority of the parts). It is functional with respect to a part, if it functionally affects the most important trait of that part, however it may affect the other traits. E.g., a change may be functional with respect to the power of a part of the system, dysfunctional with respect to its efficiency but nonfunctional with respect to its complexity. If we consider the power of the part to be more important than its efficiency, then the change is functional.

If the traits are the elements in $\underline{s}$, then an input which is functional, nonfunctional or dysfunctional to s_j will cause the latter's value to increase, to decrease or to remain unchanged, assuming positive correlation between functionality and trait value. If, at the same time, the input is dysfunctional to s_k , then the latter's value will decrease.

These ideas are, of course, not restricted to the models of the

parts. Should a change be nonfunctional over all the parts, at all times, then it is not functional over the whole system and it may be omitted from the system's as well as from the parts' models. If some traits (or parts) are similarly functionally/dysfunctionally affected by the changes, then there are fewer traits to consider (and fewer parts models to build). The extent to which the traits are related with one another, to which the inputs similarly affect the traits or the parts and to which the system is self-maintaining with respect to its traits will, like other factors, determine how complex the problem is, how complex the models need to be and, hence, how complex the system really is (surely these 3 are positively correlated?).

At the global or any other level, when the effect of each input upon the traits is heterogenous and the traits cannot be classified in order of importance, then it is not possible to devise an objective function to maximise. The worse situation that can arise is a zero sum game, although, realistically, there are always some traits which are more important than others and it seems possible to maximise (or minimise) any particular trait without too serious side-effects.

8.8 THE EFFECTS OF TIME

The final parameters of the model are concerned with time. It is obvious that the changes in s are functions of the time period considered. Less obvious is the effect of the moment in time itself. Should we, in our model, allow for the possibility that the transformation processes, f , are themselves functions of time and, therefore, make provision, as we have done, for time in the model? Most models assume f to be taken as time invariant but, for software when each state change results in a redefinition of the system itself, of its structural properties and of its interfaces, f is likely to be modified, too, so that t may have to be included as a parameter. On the other hand, it seems logical to believe that two exactly similar systems, interacting with exactly similar environments, must evolve in the same manner, even if one is old and the other is young. Then, the time parameter in the model could be dispensed with. This may be a minor point but, for the sake of completeness, it had to be included and discussed. Not that it matters very much at present, too. We are so far away from being able to identify the elements of e and s , from measuring them accurately and we know so little about f that whether f evolve as time functions should be a low priority problem.

8.9 OTHER ASPECTS OF THE MODEL

We terminate with the discussions of a few concepts and ideas that are not explicit from the models but which seem of sufficient importance to merit a few lines.

8.9.1 The long term effect of the initial system state

From our model, it is easy to see that $s(0)$, the initial system state, influences that of the next which influences that of the next, ad infinitum. The degree to which a system state influences that of the next will determine to a large extent how much the design and initial implementation hold the keys to the system's future behaviour. From what we have seen, this seems to be quite considerable.

8.9.2 Steady state

Another time related concept is that of steady state. A system is said to be in a steady state if its behaviour in one time period is the same as that in any other time period. From our model, this will be when $s_j(t+\Delta) = s_j(t)$, $\forall j$. In truth, the type of stability which a complex system may approach is rather a quasi-stationary equilibrium whereby a movement in one direction is only approximately countered by another in the opposite direction. If we were to plot the time series of the global effect, then a fluctuating series with no trend rather than a constant straight line will result. Thus, although changes may occur on all its dimensions, a system may, in the main, still remain in a steady state if its general character has not been affected too much. One of the conclusions of this work is that the more global the level of study, the more likely are we to encounter a steady state-like behaviour.

8.9.3 The effect of the system upon its environment

Just as an enterprise may achieve steady state or even induce changes in its environment by moving into new markets or developing new customer needs, among other strategies, a software system probably exerts some influence upon its environment, too, e.g., maintenance effort is a function of the number of residual bugs. Since the interface between the system and its immediate environment (and, by induction, between any 2 layers of the environment) allows traffic in both directions, the inputs for the model at any level will have to be expanded to include those that emanate from the layer immediately below it with those coming from that above it. For simplicity, we have largely ignored to discuss this set of the inputs.

8.10 CONCLUSIONS

In this chapter, we have seen how the complexity of a software system is not a simple concept but that there are several facets to it, Sections 8.3 - 8.7. Indeed, the complexity of modelling the behaviour of the system and its success will be paced by the rate of advancement in our descriptive knowledge

and insight into the static and behavioural aspects of the system and its environment and the interactions between the two. Our models must simultaneously deal with all these factors if they are to be at all successful. At present, the efficiency with which we can model the dynamics of software systems is not very high. This is because we cannot, as yet, fully identify the relevant structural and environmental properties, are not quite able to reduce and codify these to numerical values and cannot extricate the complex relationships that exist among them.

Indeed, the more tractable the system on any of its behaviour dimensions, the less complex is it on that plane, the less difficult is it to allegorise the behaviour and the less is the behaviour attributable to randomness or complexity. That our work has shown that globally most effects smooth each other out and reduce to invariance or white noise is a measure of how much still needs to be done.

To what degree, then, is it possible to attempt to unravel a software system's behaviour? Shall we then, accept, as Lehman [96] says that:-

"The theory of man-made systems, of systems including people or operating in an environment that involves people can never be complete or precise",

without attempting any effort to make it as precise as it can be? Or shall we just state, as our studies suggest, that software systems are of Sutherland's severely stochastic types [170] and, therefore, exhibit behaviours which cannot be predicted with any degree of accuracy? From Weinberg [183], who offers an alternative classification of objects into 3 types:-

- 1) The "organised simplicity" ones (machines) which are amenable to analysis.
- 2) The "unorganised complexity" ones (populations, aggregates) which are amenable to the statistical approach.
- 3) The "organised complexity" ones (systems) which are both too complex for analysis and too organised for statistics.

,it seems that we shall have to accept to live with unsucccess.

Although the empirical approach and the presently available tools seem inadequate to deal with the problem on their own, Section 7.4, one needs not really be so pessimistic. Hitherto unexplored, complementary approaches could be attempted. Mathematical models may still not be adequate, Section 8.1. Is simulation then the great white hope? Some system thinkers view it as the ultimate weapon since constructing a model that exhibits the behaviour that one wishes to understand may be

equivalent to demonstrating that the behaviour is fully grasped.

Unfortunately, simulation, which has been used for quite a while by systems engineers, does not seem to have helped us to understand socio-economic systems, say, as well as we should, despite its inroads. If it has met with only qualified success with the study of more complex systems than software and which are niched higher on the complexity gradient, then do we really have a better chance, precisely because ours are much less complex systems and because we have more generations to work upon? This may be answered by what seems to be the next logical step in ED research but it cannot be part of this effort.

That others, working on other types of systems, have preceeded us and seemingly, not fared much better should not be too much of a comfort. Do our approach and conclusions contribute anything to the software system solving problem and, in a wider context, to that of the general system?

This is not for us to answer. But this is not the way we wish to conclude. However, the words fail us. How best, then, but to quote Sutherland [170] who neatly puts down better than we shall ever be able to what we wished to express thus:-

"Many of you will walk this same ground again, hopefully with greater thoroughness and alertness than we have done here. So much in this world remains to be done, and we cannot even begin to estimate that which remains to be learned. All that can be hoped is that what has been written here will stimulate others - of greater insight, talent, or energy - to search deeper and wider. The greatest hope for what you have just read is not, then, that it should be conclusive, but that it should be germinal in some small way of better-conceived and better-executed efforts. And, perhaps paradoxically, our fondest hope is that the reader knows less now than when he began so many pages ago... that he is less certain about the answers and more sensitive to the questions."

TABLE 1

THE FIRST THREE POLYNOMIAL FITS TO THE CMH AND SIZE SERIES OF 4 PREVIOUSLY ANALYSED SYSTEMS

	OS/360	EXECUTIVE	CCSS	DOS
C	$1.57 \times 10^3 + 2.40A$	$1.93 \times 10^2 + .48A$	$-7.47 \times 10^2 + 7.11A$	$-1.19 \times 10^3 + 5.34A$
$r^2 =$	.17	.03	.06	.10
$t =$	13.68	16.77	111.51	53.33
$R^2 =$	91.67	97.57	99.66	99.27
M	$1.25 \times 10^3 + 3.68A - 7.29 \times 10^{-4} A^2$	NO GOOD QUADRATIC	$-1.24 \times 10^3 + 8.24A - 5.11 \times 10^{-4} A^2$	$-1.67 \times 10^3 + 6.22A - 2.89 \times 10^{-4} A^2$
$r^2 =$	.561 3.06×10		.244 1.08×10	.409 1.31×10
$t =$	6.55 -2.33		33.77 -4.74	15.23 -2.21
$R^2 =$	93.84		99.78	99.41
H	$5.94 \times 10^2 + 8.08A - 6.98 \times 10^{-3} A^2 + 2.16 \times 10^{-7} A^3$	$1.09 \times 10^2 + 1.32A - 1.67 \times 10^{-3} A^2 + 1.77 \times 10^{-7} A^3$	$-2.83 \times 10^2 + 4.54A + 3.32 \times 10^{-3} A^2 - 1.47 \times 10^{-7} A^3$	$-2.46 \times 10^2 + 8.58A - 2.09 \times 10^{-3} A^2 + 3.8 \times 10^{-7} A^3$
$r^2 =$	5.68 7.42×10^{-4} 2.65×10^{-7}	.18 3.38×10^{-4} 1.80×10^{-7}	.54 5.39×10^{-4} 1.59×10^{-7}	1.16 8.50×10^{-4} 1.8×10^{-7}
$t =$	14.23 -9.40 8.55	7.43 -4.92 4.93	8.434 6.19 -7.21	7.37 -2.46 2.1
$R^2 =$	98.95	99.59	99.90	99.53
S	$-7.27 \times 10^2 + 11.84A$	$3.26 \times 10^4 + 14.38A$	$1.02 \times 10^3 + .131A$	$7.87 \times 10^2 + .60A$
$r^2 =$	.15		.01	.08
$t =$	79.76		5.60	8.00
$R^2 =$	99.73	98.2	55.65	77.12
Z	$-1.01 \times 10^3 + 12.94A - 6.26 \times 10^{-4} A^2 + 3.58 \times 10^{-4} A^3$	$+6.05A + .004A^2$	$1.23 \times 10^3 - 3.26 \times 10^{-1} A + 1.98 \times 10^{-4} A^2$	$3.39 \times 10^2 + 1.70 A - 3.96 \times 10^{-4} A^2$
$r^2 =$	.47 2.58×10		.06 2.54×10	.75 2.59×10
$t =$	27.34 -2.42		-5.42 7.77	22.79 -5.30
$R^2 =$	99.80	99.20	87.39	98.37

TABLE 2
RANDOMNESS TESTS RESULTS FOR THE HR, FH AND NG SERIES

			OS/360	OMEGA	EXECUTIVE	CCSS	DOS			
HR	turning	no.of pts	20	9	9	43	22			
	points	expected	12	5	clearly	27	13			
	test	observed	14	6	not	29	12			
		variance	3.23	1.9	applicable	7.32	3.59			
		t	1.11	.73		.62	-.70			
		signif (5%)	NOT	NOT		NOT	NOT			
phase		exp	obs	exp	obs	exp	obs	exp	o	
length	1	7	11	3	5	18	19	8	6	
test	2	3	2	1	0	7	8	3	3	
	>3	1	0	0	0	1	1	1	2	
TOTAL			11	13	4	5	26	28	12	11
runs	> median	9	5			21		11		
	< median	10	4			22		11		
	runs	11	6			27		12		
	signif range	<5v>16	<2v>10			<14v>28	<7v>17			
FH	turning	no. of pts	19	10	not	17	12			
	points	expected	11	5	avail-	10	7			
	test	observed	11	6	able	12	6			
		variance	3.0	1.46		2.70	1.81			
		t	0	.55		1.21	-.49			
		signif (5%)	NOT	NOT		NOT	NOT			
phase		exp	obs	exp	obs	exp	obs	exp	o	
length	1	7	8	3	4	6	9	4	3	
	2	3	1	1	1	2	2	1	1	
	>3	0	1	0	0	1	0	1	1	
TOTAL			10	10	4	5	9	11	6	5
runs	> median	not	not	not		9		6		
	< median	appli-	appli-	appli-		8		6		
	runs	cable	cable	cable		14		7		
	signif range					<5v>14	<3v>11			
NG	turning	no. of pts	19	not	too	16	12			
	points	expected	11	avail-	few	9	7			
	test	observed	10	able		8	8			
		variance	3.06			2.52	1.81			
		t	-.76			-.84	.99			
		signif (5%)	NOT			NOT	NOT			
phase		exp	obs	exp	obs	exp	obs	exp	o	
length	1	7	8			5	1	4	5	
test	2	3	1			2	6	1	2	
	>3	0	0			1	0	1	0	
TOTAL			10	9			8	7	6	7
runs	> median	10	not	too		8		6		
	< median	9	avail-	few		8		6		
	runs	8	able			10		9		
	signif range	<5v>16				<4v>14	<3v>11			

TABLE 3

RUNS TEST RESULTS FOR THE 6 MAIN MONTHLY BD SERIES

	No. pts >median	No. pts <median	No. runs	z-value
micros changed	26	33	28	-0.56
micros enhanced	28	31	26	-1.16
micros fixed	26	33	26	-1.09
changes	27	32	28	-0.60
enhancements	25	34	24	-1.56
fixes	22	37	24	-1.66

TABLE 4

TURNING POINTS TEST RESULTS FOR THE 6 MAIN MONTHLY BD SERIES

	mc	me	mf	c	e	f
n(no. of points)	57	58	53	57	54	53
obs. no. turning pts	38	38	34	39	38	34
expected trng pts	37	37	34	37	35	34
variance	9.81	9.99	9.10	9.81	9.28	9.10
t-value	0.32	0.32	0	0.64	0.98	0

TABLE 5

PHASE LENGTH TEST RESULTS FOR THE 6 MAIN MONTHLY BD SERIES

	mc		me		mf		c		e		f	
	Obs.	Exp.	Obs.	Exp.	Obs.	Exp.	Obs.	Exp.	Obs.	Exp.	Obs.	Exp.
1	25	22.5	25	22.9	18	20.8	26	22.5	26	21.3	20	20.8
2	11	9.7	10	9.9	14	8.7	10	9.7	10	9.2	11	8.7
>2	1	3.5	2	3.5	1	3.5	2	3.5	1	3.2	2	3.5
	37	35.7	37	36.3	33	33	38	35.7	37	33.7	33	33

TABLE 6
BEST POLYNOMIAL FITS TO ALL THE MONTHLY BD SERIES

mc	30.86
me	23.69
mf	8.11 - 0.07t
c	13.32
e	3.52 + 0.12t
f	7.45 - 0.07t
cmc	10.53 + 29.84t
cme	-18.53 + 22.48t
cmf	-7.24 + 8.22t - 0.03t ²
cc	-10.79 + 13.60t
ce	24.69 + 1.25t + 0.10t ²
cf	-10.43 + 7.72t - 0.03t ²
mhgf	0.04
mhdf	0.03
mc/c	2.32
me/e	3.34
mf/f	1.13

TABLE 7

ACF VALUES FOR THE SIX MAIN MONTHLY BD SERIES

	mc	me	mf	c	e	e*	f
diff	0	0	0	0	0	1	0
mean	30.86	23.69	6.03	13.32	7.08	0.09	5.34
s.e	2.71	2.55	0.46	0.72	0.56	0.65	0.42
variance	432.88	384.46	12.72	30.77	18.66	24.33	10.23
dgr frdm	59	59	59	59	59	58	59
s.e(acf)	0.13	0.13	0.13	0.13	0.13	0.13	0.13
lag 1	0	0.030	0.153	0.116	0.348	-0.524	0.240
2	-0.076	-0.027	-0.023	0.054	0.394	0.144	0.155
3	0.014	-0.007	0.095	0.136	0.226	-0.181	0.181
4	-0.042	-0.004	0.161	0.066	0.378	0.029	0.176
5	0.098	0.134	0.183	0.176	0.426	0.050	0.205
6	0.036	0.060	0.037	0.083	0.373	0.021	0.150
7	0.202	0.212	0.126	0.104	0.319	0.013	0.198
8	0.036	0.108	0.042	-0.025	0.254	-0.068	0.094
9	-0.232	-0.142	-0.020	-0.087	0.265	0.053	0.009
10	-0.076	-0.074	0.024	0.055	0.189	-0.103	0.092
11	-0.036	0.025	-0.026	-0.065	0.271	0.231	0.043
12	-0.018	0.022	0.104	-0.046	0.052	-0.192	0.076
13	0.081	0.148	-0.038	-0.146	0.092	0.022	-0.089
14	-0.016	-0.004	-0.087	-0.146	0.091	-0.019	-0.095
15	-0.063	-0.036	0.038	-0.012	0.098	0.154	0.036
16	-0.259	-0.231	-0.115	-0.271	-0.079	-0.142	-0.139
17	-0.071	-0.056	0.011	-0.129	-0.079	0.004	-0.017
18	-0.012	-0.004	-0.012	-0.075	-0.062	0.011	-0.020
19	0.056	0.032	-0.020	-0.073	-0.097	-0.059	-0.059
20	0.049	0.053	-0.020	-0.099	-0.028	0.156	-0.106

TABLE 8
THE BEST BOX-JENKINS MODELS TO SOME BD MONTHLY SERIES

	TYPE	RESID- UAL VAR.	SIG. LEV- EL	CHI SQUARE VAL.	MODEL
me	1,0,0	377.60	.917	14.34	$z_t = 23.00 + 0.029z_{t-1} + a_t$ t= .22
	0,0,1	377.59	.917	14.33	$z_t = 23.70 + 0.029a_{t-1} + a_t$ t= -.22
c	1,0,0	29.84	.873	15.57	$z_t = 11.77 + 0.116z_{t-1} + a_t$ t= .89
	0,0,1	29.79	.875	14.69	$z_t = 13.32 + 0.118a_{t-1} + a_t$ t= -.90
e	0,1,1	14.13	.939	14.31	$z_t = z_{t-1} - 0.790a_{t-1} + a_t$ t= 10.21
f	1,1,0	12.13	.800	18.06	$z_t = .559z_{t-1} + .441z_{t-2} + a_t$ t= -3.77
	2,1,0	10.59	.975	11.66	$z_t = .415z_{t-1} + .226z_{t-2} + .359z_{t-3} + a_t$ all t's significant
	0,1,1	9.15	1.0	7.24	$z_t = z_{t-1} - 0.832a_{t-1} + a_t$ t= 11.23

TABLE 9
THE SIGNIFICANT PERIODS (ALL PERIODOGRAMS)
FOR EACH OF THE 6 MAIN BD SERIES

SERIES	PERIOD	F-VALUE	PROB.	% VAR.	No. Line Crossings
mc	6.56	4.10	.979	13.9	
	2.56	3.22	.954	10.9	
	3.69	3.13	.950	10.7	
TOTAL				35.5	0
me	6.56	3.89	.974	13.2	0
mf	NONE				
c	NONE				
e	59	12.73	1.00	43.2	11
e*	2.23	5.69	.994	19.7	0
f	59	4.97	.990	16.8	0

TABLE 10

THE SIGNIFICANT CROSS CORRELATIONS
BETWEEN PAIRS OF THE MONTHLY BD SERIES WHEN $\alpha=0.01$

x series	y series	lag(k)	$C_{xy}(k)$
mc	c	0	.76
mc	e*	-1	-.58
		0	.55
mc	me	0	.98
c	f	0	.63
c	mf	0	.63
c	e*	-1	-.55
		0	.52
c	me	0	.66
f	mf	0	.96
f	me	-9	-.36
mf	me	-9	-.35
e*	me	1	-.58
		0	.56

TABLE 11

THE SIGNIFICANT CROSS CORRELATIONS
BETWEEN PAIRS OF THE CCSS SERIES WHEN $\alpha = .05$

x series	y series	lag(k)	$C_{xy}(k)$
Rel Int	Mod chgd	NONE	
Rel Int	Fixes	0	.37
		-7	.31
Mod chgd	Fixes	1	.33
		-8	-.32

TABLE 12
THE SIGNIFICANT PERIODS (PERIDOGRAMS) FOR 3 CCSS SERIES

SERIES	PERIOD	F-VAL.	PROB.	% VAR	L.C.
Rel. Int.	2.26	3.90	.973	18.2	0
Mod. Chged	2.53	10.38	1.00	48.3	8
Fixes Rate	21.50	4.81	.987	22.3	
	8.60	3.35	.956	15.6	
				37.9	0

TABLE 13
CORRELATION MATRIX OF THE MICRO STRUCTURE PROPERTIES

	HIER. LEVEL	CALLING MICROS	CALLED MICROS	RECORD COUNT	BYTE SIZE
CALLING MODULES	0.02	0.65	-0.14	-0.04	-0.05
	HIER. LEVEL	0.05	0.08	-0.09	0.05
		CALLING MICROS	-0.09	-0.03	0.01
			CALLED MICROS	0.04	0.40
				RECORD COUNT	0.62

TABLE 14
COMPARING THE BEST POLYNOMIAL FITS
TO EXTRACTED MEASURES FROM SECTION 3.5.2.1.1

	AM	OM
mc/c	2.579	2.535
c/m	0.0786	0.0786
Max _c	12.02-0.74t t=-1.99 R ² =44.17	13.10-0.83t t=-4.14 R ² =77.42
me/e	3.34	3.27
e/m	0.0527	0.0526
Max _e	14.98-1.27t t=-2.82 R ² =61.40	16.03-1.56t t=-3.12 R ² =66.03
mf/f	1.28	1.23
f/m	0.0470-0.00382t t=-3.82 R ² =64.56	0.0468-0.00378t t=-3.74 R ² =63.56
Max _f	11.9	12.0

TABLE 15

R² VALUES WHEN EACH OF 4 TYPES OF EXPRESSIONS FITTED TO DATA

	$y=a+bt$	$1/y=a+bt$	$yt=c$	$y=ke^{-\lambda t}$
mc	76.79	52.73	55.82	63.88
me	75.49	56.67	33.20	80.67
mf	63.59	73.64	48.90	83.93
c	76.79	52.73	55.82	63.88
e	41.53	42.12	58.28	42.03
f	75.39	70.83	58.15	81.70
mc/c	87.32	97.86	47.58	95.16
me/e	85.98	96.75	42.19	94.29
mf/f	60.28	71.21	45.44	66.24

TABLE 16

NEW MODULES - PREDICTED CUMY VALUES WHEN $t=6$

VAR	REAL	$y=a+bt$	$1/y=bt$	$yt=c$	$y=(1-e^{-\lambda t})k/\lambda$	$y=k(1-e^{-\lambda t})/(1-e^{-\lambda})$
mc	217	217.9	167.6	201.9	278.8	350.4
me	179	179.5	142.1	172.0	222.7	278.1
mf	38	38.4	24.4	30.0	46.2	58.6
c	101	96.9	92.6	95.2	112.8	127.7
e	75	71.1	71.0	73.2	79.0	87.2
f	26	25.8	20.3	22.1	30.8	37.3

TABLE 17

NEW MODULES- PREDICTED T VALUES
AND ESTIMATION OF "FRACTION OF WORK COMPLETED"

variable	$T=k/\lambda$	$T=k/(1-e^{-\lambda})$	(real) $100\lambda/k$
mc	295.9	371.8	73.3
me	237.7	296.8	75.3
mf	48.6	61.7	78.2
c	144.2	163.3	61.9
e	112.9	124.6	66.4
f	34.1	41.2	76.3

TABLE 18

CORRELATION MATRIX OF THE HOLON STRUCTURE PROPERTIES

[illegible]

TABLE 19

THE INCREMENTAL MULTIPLE REGRESSION MODELS FOR PATCHES AND DAF2

DIRECTOR PATCHES									
Δ & ACTUAL	-1.23	+ .62 Δ DAF2	+ .08 IOE	+ .14 EXTO	+ .10 DAF2	$ \Delta $ & ACTUAL	-1.01	+ .58 Δ DAF2	+ .10 IOE
$\sigma =$	.09	.03	.04	.04		$\sigma =$	.08	.03	.04
$t =$	7.00	2.22	3.67	2.30		$t =$	6.95	3.69	4.67
$R^2 =$	73.02					$R^2 =$	76.25		
Δ , ACTUAL & RATE	-.06	+ .81 Δ DAF2	-.08 IOE			$ \Delta $, ACTUAL & RATE	-.13	+ .81 Δ DAF2	+ 2.29 $ \Delta $ IOE / IOE
$\sigma =$	.10	.04				$\sigma =$	.09	.87	
$t =$	8.39	-2.30				$t =$	8.64	2.64	
$R^2 =$	59.53					$R^2 =$	60.48		
KERNEL PATCHES									
Δ & ACTUAL	-9.2	+ .02 Δ SIZE	+ .6 Δ DAF2	+ .19 EXTS		$ \Delta $ & ACTUAL	-1.04	+ .02 $ \Delta $ SIZE	+ .54 Δ DAF2
$\sigma =$	.002	.17	.08			$\sigma =$	.002	.11	.01
$t =$	9.27	3.51	2.30			$t =$	10.38	4.82	2.97
$R^2 =$	90.76					$R^2 =$	95.78		
Δ , ACTUAL & RATE	.14	+ .01 Δ SIZE	+ .89 Δ DAF2	+ .45 Δ CALLS / CALLS	+ .33 Δ CALLS	$ \Delta $, ACTUAL & RATE	-.16	+ .03 $ \Delta $ SIZE	+ .29 Δ DAF2
$\sigma =$	.002	.09	.06	.06		$\sigma =$	.001	.09	.27
$t =$	6.07	9.79	7.90	6.15		$t =$	18.99	3.09	3.08
$R^2 =$	95.55					$R^2 =$	97.26		
DIRECTOR DAF2									
Δ & ACTUAL	1.76	+ .54 Δ PATCHES	+ .26 Δ EXTO	-.23 Δ CALLS	+ .01 Δ SIZE	$ \Delta $ & ACTUAL	1.27	+ .60 Δ PATCHES	+ .005 $ \Delta $ SIZE
$\sigma =$	.07	.11	.06	.002		$\sigma =$	.07	.002	
$t =$	7.64	2.38	-3.77	2.86		$t =$	8.53	2.34	
$R^2 =$	65.96					$R^2 =$	59.61		
Δ , ACTUAL & RATE	.82	+ .60 Δ PATCHES	+ 1.03 Δ DAF2 / DAF2			$ \Delta $, ACTUAL & RATE	.82	+ .60 Δ PATCHES	+ 1.03 Δ DAF2 / DAF2
$\sigma =$	.06	.25				$\sigma =$	.06	.25	
$t =$	9.41	4.19				$t =$	9.41	4.19	
$R^2 =$	65.69					$R^2 =$	65.69		
KERNEL DAF2									
Δ & ACTUAL	.21	+ .52 Δ PATCHES	-.01 Δ SIZE	+ .61 Δ EXTS	+ .12 IOE	$ \Delta $ & ACTUAL	-.46	+ .28 Δ PATCHES	-1.48 $ \Delta $ EXTO
$\sigma =$	.10	.003	.15	.04		$\sigma =$	.06	.22	.13
$t =$	5.42	-3.62	3.97	2.68		$t =$	4.63	-6.73	5.52
$R^2 =$	81.48					$R^2 =$	87.48		
Δ , ACTUAL & RATE	.56	+ .79 Δ PATCHES	-.35 Δ CALLS / CALLS	-.01 Δ SIZE / SIZE	+ .15 IOE	$ \Delta $, ACTUAL & RATE	.85	+ .58 Δ PATCHES	+ 1.45 Δ DAF2 / DAF2
$\sigma =$	.08	.06	.002	.05		$\sigma =$	.09	.29	.09
$t =$	10.11	-5.63	-4.92	2.78		$t =$	6.13	4.95	-2.98
$R^2 =$	88.24					$R^2 =$	78.47		

TABLE 20
THE CUMULATIVE MULTIPLE REGRESSION MODELS FOR PATCHES AND DAF2

----- PATCHES -----											
CCD27	-.55+.01SIZE+.27DAF2					CCK27	1.73+.94CALLS+.83IOE-.85EXTS-.95IOP				
	$\sigma =$	.002	.12				.16	.19	.35	.43	
	t=	2.45	2.15				5.96	4.42	2.43	-2.19	
	R ² =	18.78				84.47					
CCD32	-.80+.41DAF2+.006SIZE					CCK32	2.93+.96CALLS+.88DAF2-1.07EXTS-1.17IOP				
	$\sigma =$	.10	.002				.20	.23	.44	.50	
	t=	3.97	3.33				4.80	3.77	-2.44	2.31	
	R ² =	39.90				86.89					
UCD32	-.87+.49DAF2+.22IOE+.18IOG					UCK32	-.85+1.07EXTS				
	$\sigma =$	.04	.08	.07			.22				
	t=	11.45	2.83	2.55			4.94				
	R ² =	60.92				38.52					
ND32	-.92+.70DAF2-.01AGE+.004SIZE-.14EXTS					NK32	-.78+1.18DAFS-1.78IOG				
	$\sigma =$	.18	.01	.002	.07		.08	.33			
	t=	3.80	2.00	2.61	2.15		13.87	-5.35			
	R ² =	82.01				96.66					
----- DAF2 -----											
CCD27	2.54+.39IOE+.23IOP+.26PATCHES					CCK27	1.05+.16CALLS+.18PATCHES+.06AGE				
	$\sigma =$	.08	.07	.09			.10	.10	.05		
	t=	4.90	3.23	2.86			2.29	2.41	2.77		
	R ² =	45.68				62.21					
CCD32	4.71+.30PATCHES+.12IOE+.007SIZE					CCK32	2.29+.43PATCHES+.10AGE				
	$\sigma =$	.11	.03	.002			.08	.06			
	t=	2.80	3.66	3.09			5.60	2.73			
	R ² =	49.17				68.42					
UCD32	1.13+1.02PATCHES+.43EXTS+.51EXTO					UCK32	4.14+.50EXTS-.23IOE				
	$\sigma =$	.08	.07	.07			.19	.12			
	t=	12.94	6.02	7.00			2.65	-1.97			
	R ² =	66.68				16.62					
ND32	.89+.52PATCHES+.03AGE+.002SIZE					NK32	.58+.70PATCHES+1.63IOG+.003SIZE				
	$\sigma =$	.11	.01	.001			.06	.16	.001		
	t=	4.59	2.70	2.00			11.11	10.01	2.46		
	R ² =	81.60				98.97					

TABLE 21

THE SIGNIFICANT ACF'S FOR B'S
SUBSYSTEMS DAF2'S/MONTH TIME SERIES

SERIES	CONFIDENCE LIMITS(±)	LAG(K)	R(K)
G1SSB	.248	1 6	.289 .255
G1SSC	.280	1	.356
G1SSD	.292	2 3 16	.354 .310 -.296
G5SS1	.252	NONE	
G5SS2	.244	NONE	
G5SS3	.252	2	.267
G5SS7	.292	1	.331
G9SS8	.304	NONE	
G13SS1	.312	NONE	

TABLE 22

THE BEST BOX-JENKINS MODELS TO THE WEEKLY SOFTWARE NOTICES

TYPE	RESIDUAL VARIANCE	SIGNIF LEVEL	CHI- SQUARE	MODEL
1,0,0	22.57	.02	39.57	$z_t = 3.54 + .47z_{t-1} + a_t$ $\sigma = .07$ $t = 6.88$
2,0,0	21.24	.30	24.87	$z_t = 2.61 + .36z_{t-1} + .24z_{t-2} + a_t$ $\sigma = .07$ $t = 4.77 \quad 3.23$
3,0,0	20.58	.58	19.14	$z_t = 2.13 + .32z_{t-1} + .18z_{t-2} + .18z_{t-3} + a_t$ $\sigma = .08$ $t = 4.13 \quad 2.25 \quad 2.37$
1,0,1	20.91	.36	23.77	$z_t = 1.11 + .83z_{t-1} - .47a_{t-1} + a_t$ $\sigma = .07$ $t = 11.04 \quad 4.02$
0,1,1	22.52	.34	26.22	$z_t = z_{t-1} - .60a_{t-1} + a_t$ $\sigma = .06$ $t = 10.10$

TABLE 23

SOFTWARE NOTICES STATISTICS FOR DIFFERENT RELEASES

	5X10	5X18	5X23	5X27	5X32
size	87	123	119	126	78
maximum	34	19	19	15	11
mean	4.64	2.56	3.03	2.51	2.90
variance	36.67	11.89	11.90	8.62	5.86
s.d.	6.06	3.45	3.45	2.94	2.42

TABLE 24

MEAN RESPONSE TIME (DAYS)
FOR FIRST 90% OF PRIORITY B BUG REPORTS

period(mth)	7708	7803	7809	7902
mean	3.52	2.61	3.33	2.20

TABLE 25

THE BEST BOX-JENKINS MODELS TO THE DAILY AND WEEKLY REP AND RES SERIES

TYPE	RESIDUAL VARIANCE	SIGNIF LEVEL	CHI- SQUARE	MODEL		
DAILY REP						
2,0,0	80.90	.39	23.15	22	$12.71 + .23z_{t-1} + .20z_{t-2} + a_t$ $\sigma = .07$ $t = 3.33 \quad 2.47$	
5,0,0	75.50	.72	15.04	19	$8.64 + .20z_{t-1} + .19z_{t-2} - .01z_{t-3} + .02z_{t-4} + .22z_{t-5} + a_t$ $\sigma = .07$ $t = 2.87 \quad 2.64 \quad -.20 \quad .26 \quad 3.24$	
DAILY RES						
3,0,0	106.70	.90	13.30	21	$13.71 - .0009z_{t-1} + .21z_{t-2} + .20z_{t-3} + a_t$ $\sigma = .07$ $t = -.01 \quad 3.25 \quad 2.99$	
WEEKLY REP						
1,0,1	906.02	.97	11.07	22	$88.04 + .23z_{t-1} + .15a_{t-1} + a_t$ $\sigma = .31$ $t = .74 \quad -.47$	
1,0,0	907.82	.97	11.93	23	$73.48 + .36z_{t-1} + a_t$ $\sigma = .10$ $t = 3.42$	
WEEKLY RES						
1,0,1	1090.6	.90	13.94	22	$52.92 + .54z_{t-1} - .17a_{t-1} + a_t$ $\sigma = .22$ $t = 2.44 \quad .64$	
1,0,0	1095.8	.94	13.67	23	$70.45 + .39z_{t-1} + a_t$ $\sigma = .10$ $t = 3.88$	

TABLE 26

THE % OF VALUES INCLUDED WITHIN EACH RESPONSE TIME RANGE

RANGE (DAYS)	0-7	0-14	0-30	ALL	30+
% VALUES INCLUDED	70	80	90	100	10

TABLE 27

SOME STATISTICS CONCERNING THE B GROUPS

TABLES

GROUP	START DATE	END DATE	AGE (MTHS)	ACR LEVEL	NO. OF S/S	NO. OF HOLONS	NO. OF DAF6'S	AVERAGE DAF6/MTH	ACR RANK	S/S RANK	HOLS RANK	DAF6'S RANK	DAF6'S /MTH RK	FIGURE OF MERIT
1	7601		38	1,2	13	1004	1382	36.4	---OMITTED BECAUSE OF ITS UNIQUENESS---					
2	7403		60	4	6	198	324	5.5	4	4.5	5	10	10	33.5
3	7509		41	4	1	380	508	12.4	4	17.5	3	5	4	33.5
4	7403		60	4	1	150	279	4.7	4	17.5	9	12	12	54.5
5	7403		60	5	6	687	1325	22.1	11	4.5	1	1	2	19.5
6	7405		58	8	6	180	1315	22.7	17	4.5	7	2	1	31.5
7	7403		60	5	3	364	419	7.0	11	9.5	4	8	9	41.5
8	7412		51	5/9	3	51	80	1.6	19.5	9.5	16	17	17	79
9	7408		55	9	8	56	585	10.8	19.5	1	15	4	5	44.5
10	7406		57	9	6	65	296	5.3	19.5	4.5	14	11	11	60
11	7501	7601	13	15	6	?	14	1.1	TOO SHORT LIFE SPAN, PROBABLY WITHDRAWN GROUP					
12	-----WITHDRAWN GROUP FOR WHICH NO DATA ARE AVAILABLE-----													
13	7607		30	5	1	188	134	4.5	11	17.5	6	14	13	61.5
14	7404		59	4	4	21	37	.6	4	7	17	20.5	20.5	69
16	7504	7612	9	NONE	1	?	7	.9	-----BOOTSTRAP AND WITHDRAWN GROUP-----					
18	7403	7710	43	5	2	116	62	1.4	11	12.5	12	18	18	71.5
20	7605		34	8	1	4	14	.4	-----TOO FEW DAF6'S-----					
21	7406		57	7	7	123	120	2.2	15.5	2	11	15	15	58.5
22	7403	7712	45	4	2	?	434	9.5	4	12.5	?	6	7	36.9*
23	7406		57	4	1	80	48	.9	4	17.5	13	19	19	72.5
24	7507		43	4	1	508	427	9.9	4	17.5	2	7	6	36.5
26	?	?	?	?	1	-----NOT ENOUGH DATA AVAILABLE FOR THIS (WITHDRAWN) GROUP-----								
27	7506	7603	10	10	1	?	40	4.0	TOO SHORT LIFE SPAN, PROBABLY W/D					
29	7603	7608	6	8	1	?	8	1.3	TOO SHORT LIFE SPAN, PROBABLY W/D					
31	7407		56	5	1	158	143	2.6	11	17.5	8	13	14	63.5
32	7403	7406	4	?	?	?	3	.7	INSUFFICIENT DATA, TOO SHORT LIFE SPAN					
33	7404		59	5	3	143	112	2.0	11	9.5	10	16	16	62.5
34	?		?	?	1	-----INSUFFICIENT DATA-----								
35	?		?	4	1	17	-----INSUFFICIENT DATA-----							
37	7405	7505	13	?	?	?	20	1.5	INSUFFICIENT DATA, TOO SHORT LIFE SPAN					
40	7403	7704	37	5	1	?	351	9.2	11	17.5	?	9	8	56.9*
44	7405		58	9	3	16	775	13.4	19.5	9.5	18	3	3	53
45	7404		59	7	1	?	37	.6	15.5	17.5	?	20.5	20.5	92.5*
47	?		?	3	3	13	-----INSUFFICIENT DATA-----							
49	?		?	?	3	-----INSUFFICIENT DATA-----								
50	7506	7605	12	10	1	?	10	.8	TOO SHORT LIFE SPAN, PROBABLY W/D					
51	7404	7508	17	?	?	?	18	1.1	INSUFFICIENT DATA, TOO SHORT LIFE SPAN					
53	-----NO DATA AVAILABLE-----													
54					2	INSUFFICIENT DATA								
55					6	INSUFFICIENT DATA								

* WEIGHTED TO ACCOUNT FOR MISSING HOLONS RANK

PAGE 215

TABLE 28
CLUSTERING THE GROUPS

GROUP	FIGURE OF MERIT	CLUSTER TYPE
5	19.5	B
6	31.5	B
2	33.5	B
3	33.5	B
24	36.5	B
22	36.9	B
7	41.5	B
9	44.5	A
44	53	A
4	54.5	B
40	56.9	A
21	58.5	A
10	60	A
13	61.5	B
33	62.5	D
31	63.5	C
14	69	D
18	71.5	D
23	77.5	D
8	79	D
45	92.5	D

TABLE 29

BEST POLYNOMIAL FITS TO THE MONTHLY TIME SERIES
OF THE FRACTION OF G5 DAF6'S IN SOME CATEGORIES

CATEGORY		BEST POLYNOMIAL FIT			
UTIL GENGEN		73.73 + .58T			
		$\sigma =$	.22		
		t =	2.61		$R^2 = 18.48$
UTIL GENOPT		27.15 - .78T			
		$\sigma =$	.25		
		t =	-3.13		$R^2 = 26.58$
CAT	BUGS	THE MEAN			
CAT SPEC BUGS		7.98 + .38T			
		$\sigma =$	.18		
		t =	2.17		$R^2 = 11.88$
CAT IMPL BUGS		29.27 + 3.35T - .06T ²			
		$\sigma =$	1.05	.03	
		t =	3.20	-2.63	$R^2 = 27.46$
CAT	ENHANCEMENTS	THE MEAN			
MODE CHANGE		THE MEAN			
MODE INFORMATION		THE MEAN			
VERSION ONWARDS		1.70 + 9.89T - .31T ²			
		$\sigma =$	2.78	.10	
		t =	3.55	-3.09	$R^2 = 7.44$
NV 1		112.77 - 1.48T			
		$\sigma =$	.19		
		t =	-7.74		$R^2 = 50.82$
NV 2		-1.05 + .60T - .01T ²			
		$\sigma =$	.24	.004	
		t =	2.53	-2.12	$R^2 = 12.58$
TIMES 1		62.30 + 1.29T - .01T ²			
		$\sigma =$	.35	.006	
		t =	3.70	-2.29	$R^2 = 41.92$
TIMES 2		18.72 - .29T			
		$\sigma =$	.06		
		t =	-5.20		$R^2 = 31.82$

TABLE-30
DEMONSTRATING HOW SHANNON'S H MEASURES THE
DIVERSITY OF A SAMPLE OVER ITS CATEGORIES

CAT\SAMPLE	1	2	3	4	5	6	7
1	80	70	20	60	50	100	96
2	5	10	20	15	25	0	1
3	5	5	20	15	15	0	1
4	5	10	20	15	5	0	1
5	5	5	20	15	5	0	1
H VALUE	77.8	100.0	161.0	138.6	127.7	0	22.3

TABLE 31
BEST POLYNOMIAL FITS TO THE DIVERSITY MONTHLY
TIME SERIES OF EACH G5 DAF6 VARIABLE

VARIABLE	TYPE OF MEASURE	BEST POLYNOMIAL FIT			
INIT	H	THE MEAN			
MTHS	MEAN	$39.27 + 7.46T - .08T^2$ $\sigma = 2.18 \quad .04$ $t = 3.41 \quad -2.21$			$R^2 = 36.47$
UTIL	H	$56.38 - .98T$ $\sigma = .41$ $t = -2.39$			$R^2 = 16.03$
CAT	H	$54.90 - .39T$ $\sigma = .173$ $t = -2.23$			$R^2 = 8.05$
MODE	H	$96.48 - 4.61T + .07T^2$ $\sigma = .89 \quad .01$ $t = -5.19 \quad 5.14$			$R^2 = 33.02$
VERSION	H	$-4.58 + 5.36T - .07T^2$ $\sigma = .87 \quad .01$ $t = 6.18 \quad -5.16$			$R^2 = 47.05$
NV	MEAN	$.94 + .01T$ $\sigma = .002$ $t = 4.28$			$R^2 = 24.02$
TIMES	MEAN	$1.35 - .01T$ $\sigma = .001$ $t = -6.10$			$R^2 = 39.07$

TABLE 32
BEST POLYNOMIAL FITS TO THE MONTHLY TIME SERIES OF
THE FRACTION OF G3 AND G6 DAF6'S IN SOME CATEGORIES

BEST POLYNOMIAL FIT				
CATEGORY	GROUP 3		GROUP 6	
UTIL GENGEN	110.85 -	5.70T + .11T ²	104.51 -	.52T
	$\sigma =$	1.72 .04		.18
	$t =$	-3.32 2.88		-3.00
	$R^2 =$	24.92		17.64
UTIL GENOPT	THE MEAN		THE MEAN	
CAT BUGS	THE MEAN		95.45 -	.84T + .02T ²
	$\sigma =$			.35 .006
	$t =$			-2.36 2.79
	$R^2 =$			15.30
CAT SPEC BUGS	THE MEAN		THE MEAN	
CAT IMP BUGS	-5.73 +	5.62T - .10T ²	16.92 +	5.53T - .12T ²
	$\sigma =$	1.74 .04		1.27 .03
	$t =$	3.22 -2.51		4.35 -3.82
	$R^2 =$	30.05		37.53
CAT ENHMTS	39.02 -	2.35T + .04T ²	THE MEAN	
	$\sigma =$	.66 .01		
	$t =$	-3.57 2.54		
	$R^2 =$	40.81		
MODE CHANGE	101.18 -	1.92T	110.03 -	.89T
	$\sigma =$	.41		.33
	$t =$	-4.72		-2.69
	$R^2 =$	35.74		11.60
MODE INFO	6.60 +	1.01T	-16.80 +	1.48T
	$\sigma =$	.41		.23
	$t =$	2.45		6.41
	$R^2 =$	13.03		42.76
VERSION ONW	-12.58 +	2.29T - .04T ²	-20.83 +	1.55T
	$\sigma =$	.91 .02		.24
	$t =$	2.53 -2.15		6.42
	$R^2 =$	16.79		42.85
NV 1	124.31 -	4.89T + .08T ²	102.87 -	.88T
	$\sigma =$	1.52 .03		.27
	$t =$	-3.22 2.43		-3.23
	$R^2 =$	31.85		15.93
NV 2	-2.50 +	1.48T - .03T ²	-8.73 +	.97T
	$\sigma =$	.61 .01		.23
	$t =$	2.44 -2.54		4.16
	$R^2 =$	14.17		23.97

TABLE 33

BEST POLYNOMIAL FITS TO THE DIVERSITY MONTHLY
TIME SERIES OF EACH G3 AND G6 DAF6 VARIABLE

BEST POLYNOMIAL FIT				
VARIABLE	TYPE	GROUP 3	GROUP 6	
INIT	H	$1.41 - .01T$	$2.86 - .89T + .002T^2$	
	$\sigma =$	.006	.02 .0003	
	$t =$	-2.32	-5.53 4.07	
	$R^2 =$	13.31	53.16	
MTHS	MEAN	THE MEAN	$.38 + .29T - .004T^2$	
	$\sigma =$		.08 .001	
	$t =$		3.52 -3.14	
	$R^2 =$		20.13	
UTIL	H	$.01 + .08T - .002T^2$	$-.02 + .006T$	
	$\sigma =$	.02 .0006	.003	
	$t =$	3.47 -3.76	2.40	
	$R^2 =$	30.20	11.36	
CAT	H	$.86 - .02T$	$.40 - .004T$	
	$\sigma =$	.005	.002	
	$t =$	-4.42	-2.06	
	$R^2 =$	35.81	8.40	
MODE	H	THE MEAN	$.36 - .005T$	
	$\sigma =$		.002	
	$t =$		-2.49	
	$R^2 =$		10.16	
VERSION	H	THE MEAN	$-.05 + .05T - .0008T^2$	
	$\sigma =$		.01 .0002	
	$R^2 =$		23.98	
NV	MEAN	$-.12 + .05T - .001T^2$	$-.15 + .03T - .0005T^2$	
	$\sigma =$	.02 .0004	.01 .0001	
	$t =$	3.17 -3.00	3.40 -3.21	
	$R^2 =$	20.58	17.74	

TABLE 34

COMPARING THE TRENDS IN THE 3 GROUPS'
FRACTION OF DAF6'S MONTHLY TIME SERIES

CATEGORY	GROUP 3	GROUP 5	GROUP 6
UTIL GENGEN	↘	↗	↘
UTIL GENSEL	K	↘	K
CAT BUGS	K	K	↘
CAT SPEC BUGS	K	↗	K
CAT IMPL BUGS	↗	↗	↗
CAT EHMNTS	↘	K	K
MODE CHANGE	↘	K	↘
MODE INFO	↗	K	↗
VERS ONWARDS	↘	↘	↗
NV 1	↘	↘	↘
NV 2	↗	↗	↗

TABLE 35

COMPARING THE TRENDS IN THE 3 GROUPS'
DIVERSITY MONTHLY TIME SERIES

VARIABLE	GROUP 3	GROUP 5	GROUP 6
INIT	↘	K	↘
MTHS	K	↘	↗
UTIL	↗	↘	↗
CAT	↘	↘	↘
MODE	K	↘	↘
VERSION	K	↗	↘
NV	↗	↗	↘

TABLE 36

CORRELATION MATRIX FOR FIVE DAF6 VARIABLES

UTIL			CAT			VERSION			MODE			
	G3	G5	G6	G3	G5	G6	G3	G5	G6	G3	G5	G6
1					NOT	NOT		NOT			NOT	
2					NOT	NOT		NOT	.001		NOT	NOT
3					NOT	NOT		NOT	.01		NOT	NOT
M 4	NOT		NOT	.01	.001	NOT	.01	NOT	.01	.001	NOT	NOT
T 5	.01	NOT	NOT	NOT	NOT	NOT	.05	.001	NOT	.01	NOT	NOT
H 6	NOT	NOT	NOT	.01	NOT	NOT	.01	NOT	.02	NOT	NOT	NOT
S 7	NOT	NOT	NOT	NOT	NOT	NOT	NOT	.001	NOT	.001	.01	NOT
8	NOT	NOT	NOT	NOT	NOT	NOT	NOT	NOT	NOT	NOT	NOT	NOT
9	NOT	NOT	NOT	NOT	NOT	NOT	NOT	NOT	NOT	.01	NOT	.01
10	NOT		NOT	NOT	NOT	NOT	NOT	NOT		NOT	NOT	NOT

			1									
			2			NOT			NOT			
			3									
U 4		NOT		NOT		NOT	NOT		NOT	NOT		NOT
T 5		NOT		.001	NOT	.01	NOT	NOT	NOT	.001	.001	NOT
I 6		NOT		.001	NOT	NOT	NOT	NOT	NOT	.05	.001	NOT
L 7		NOT		.01	NOT	NOT	NOT	NOT	NOT	.001	.001	NOT
8		NOT		NOT	NOT	NOT	NOT	NOT	NOT	NOT	NOT	NOT
9		NOT		NOT	NOT	NOT	NOT	NOT	NOT	NOT	NOT	NOT
10		NOT				NOT	NOT			NOT		NOT

						1		NOT			.05	
						2		NOT	.05		.05	.001
						3		NOT	NOT		.05	.01
						4	.02	NOT	.01	.001	NOT	.01
C 5		NOT		NOT		NOT	NOT	NOT	NOT	.001	NOT	NOT
A 6		NOT		NOT		.02	.02	.01	NOT		NOT	NOT
T 7		NOT		NOT		NOT	NOT	NOT	NOT	NOT	NOT	NOT
8		NOT		NOT		NOT	NOT	NOT	NOT	NOT	NOT	NOT
9		NOT		NOT		NOT	NOT	NOT	NOT	NOT	NOT	NOT
10		NOT				NOT	NOT			NOT	NOT	NOT

									1		.01	
									2		NOT	.001
V 3									3		NOT	NOT
E 4		NOT		NOT		NOT	NOT		4	NOT	NOT	NOT
R 5		.01		NOT		NOT	NOT		5	.01	NOT	NOT
S 6		NOT		.001		.05			6	NOT	.001	.05
I 7		NOT		.01		NOT	NOT		7	NOT	.01	NOT
O 8		NOT		NOT		NOT	NOT		8	NOT	NOT	NOT
N 9		NOT		NOT		NOT	NOT		9	NOT	NOT	NOT
									10	NOT	NOT	

TABLE 37

RELEASE DISTRIBUTION BY ATTEMPTS (GROUP 3)

ATTEMPTS	4X	5X10	5X18	5X23 & ONWARDS	5X27 & ONWARDS	5X32 & ONWARDS
1	37	177	31	53	63	4
2	0	32	4	12	15	0
3	0	13	2	2	6	0
4	0	0	0	1	1	0
5	0	1	0	0	0	0
TOTAL	37	223	37	68	85	4

TABLE 38

RELEASE DISTRIBUTIONS BY ATTEMPTS (GROUP 5)

ATTEMPTS	3X	4X	5X10	5X18	5X23	5X27	5X32
1	247	257	174	78	81	105	58
2	0	1	8	1	23	13	6
3	0	1	5	1	12	2	2
4	0	0	1	0	4	1	0
5	0	0	0	0	2	0	0
6	0	0	1	0	1	0	0
TOTAL	247	259	189	80	123	121	66

TABLE 39

RELEASE DISTRIBUTIONS BY ATTEMPTS (GROUP 6)

ATTEMPTS	3X	4X	5X10	5X18	5X23	5X27	5X32
1	243	488	102	20	77	12	28
2	0	2	6	0	19	0	0
3	0	0	5	0	5	0	0
4	0	0	0	0	1	0	0
TOTAL	243	490	113	20	102	12	28

TABLE 40

RESULTS OF THE PAIRWISE COMPARISONS
OF THE RELEASE DISTRIBUTION BY ATTEMPTS
(CHI-SQUARE TEST) FOR EACH OF THE 3 GROUPS

GROUP	3X^4X	4X^5X10	5X10^5X18	5X18^5X23	5X23^5X27	5X27^5X32
3		.01	NOT	NOT	NOT	NOT
5	NOT	.001	NOT	.001	.001	NOT
6	NOT	.001		.05		NOT

TABLE 41

RESULTS OF THE BLANKET TEST (CHI-SQUARE)
OVER THE 3 GROUPS' DISTRIBUTIONS BY ATTEMPTS
FROM RELEASE TO RELEASE

RELEASE	3X	4X	5X10	5X18	5X23	5X27	5X32	GLOBAL
RESULT	NOT	NOT	.01	.05	NOT	NOT	NOT	.001

TABLE 42

RESULTS OF THE CHI-SQUARE PAIRWISE COMPARISONS
OF THE GROUP DISTRIBUTIONS BY ATTEMPTS
FOR EACH OF THE RELEASES

	3X	4X	5X10	5X18	5X23	5X27	5X32	GLOBAL
G3^G5		NOT	.001	.02	NOT	.05	NOT	.001
G5^G6	NOT	NOT	NOT	NOT	NOT	NOT	NOT	.001
G6^G3		NOT	.05	.05	NOT	NOT	NOT	.001

TABLE 43

DIVERSITY VALUES OVER THE RELEASES FOR EACH
OF 4 VARIABLES AND EACH OF THE 3 GROUPS

RELEASE		GROUP 3	GROUP 5	GROUP 6
MTHS	3X		1.32	2.39
	4X	7.95	3.71	3.28
	5X10	5.65	4.95	4.33
	5X18	2.32	2.84	3.10
	5X23	7.29	5.16	2.82
	5X27	6.09	3.59	5.45
UTIL	3X		0	0
	4X	0	7.7	0
	5X10	85.51	60.44	9.58
	5X18	69.12	64.00	0
	5X23	116.12	53.60	14.02
	5X27	54.66	35.88	56.61
CAT	3X		35.56	18.72
	4X	85.04	73.97	46.17
	5X10	92.08	58.81	54.15
	5X18	64.37	51.27	19.85
	5X23	59.73	52.42	57.04
	5X27	26.02	40.34	28.68
MODE	3X		48.66	6.65
	4X	0	33.43	40.27
	5X10	49.66	50.19	31.92
	5X18	67.51	66.16	19.85
	5X23	69.29	46.99	19.56
	5X27	68.98	32.15	28.68

TABLE 44

RESULTS OF THE PAIRWISE COMPARISONS OF THE
RELEASE DISTRIBUTION BY ANY OF 4 VARIABLES
FOR EACH OF THE 3 GROUPS

VARIABLE	PAIRS	GROUP 3	GROUP5	GROUP6
MTHS	3XΛ4X		.001	.01
	4XΛ5X10	.001	.001	.001
	5X10Λ5X18	.02	.001	.01
	5X18Λ5X23	.001	.001	NOT
	5X23Λ5X27	.02	.001	.05
UTIL	3XΛ4X		NOT	NOT
	4XΛ5X10	.01	.001	NOT
	5X10Λ5X18	.001	NOT	NOT
	5X18Λ5X23	NOT	NOT	NOT
	5X23Λ5X27	.001	NOT	NOT
CAT	3XΛ4X		NOT	NOT
	4XΛ5X10	.05	.001	.001
	5X10Λ5X18	.05	NOT	NOT
	5X18Λ5X23	NOT	NOT	NOT
	5X23Λ5X27	NOT	NOT	NOT
MODE	3XΛ4X		.001	.01
	4XΛ5X10	NOT	NOT	NOT
	5X10Λ5X18	.001	.05	NOT
	5X18Λ5X23	NOT	.02	NOT
	5X23Λ5X27	NOT	NOT	NOT

TABLE 45

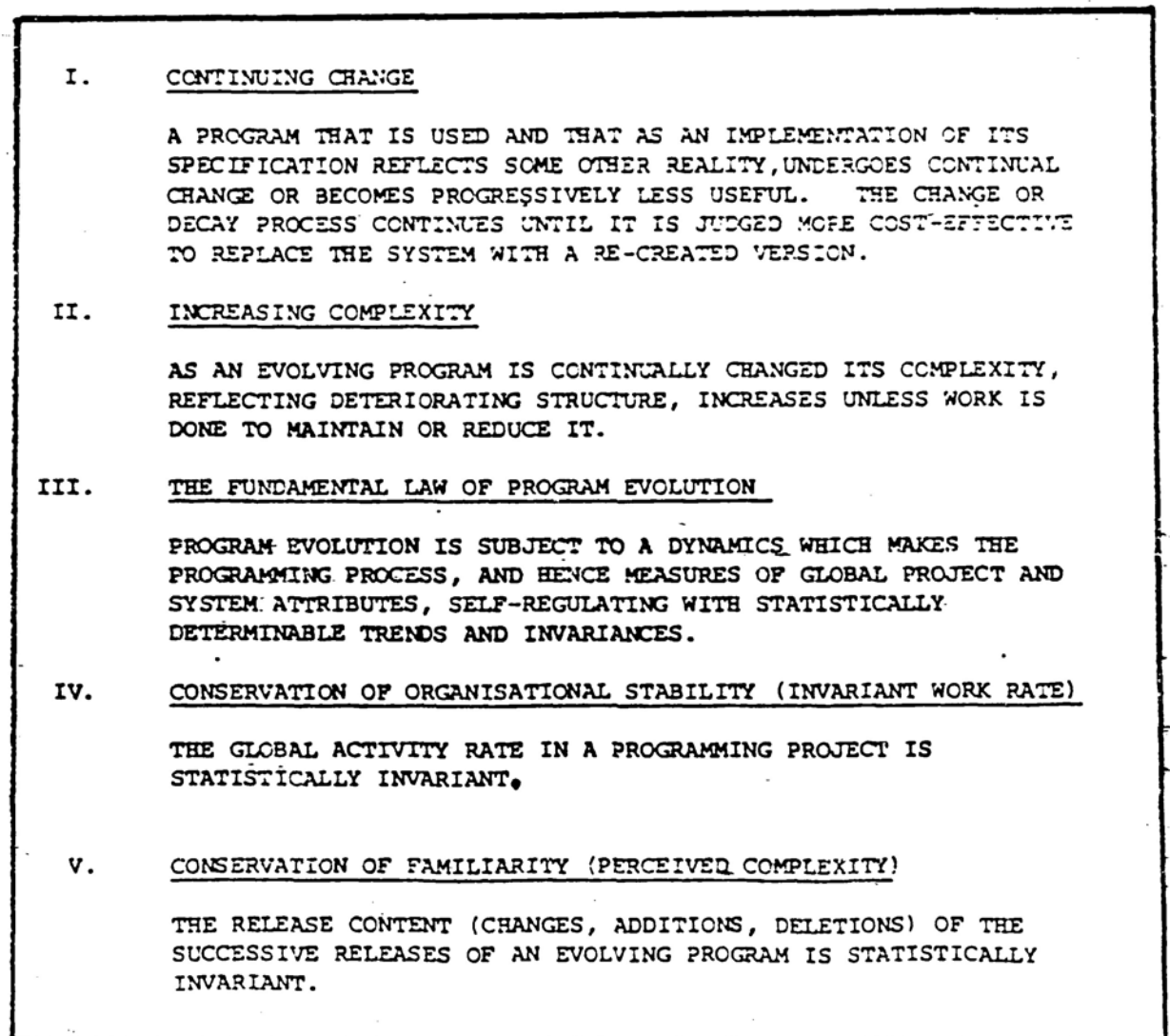
RESULTS OF THE BLANKET TEST OVER THE 3 GROUPS' DISTRIBUTIONS
BY ANY OF 4 VARIABLES FROM RELEASE TO RELEASE

VARIABLE	3X	4X	5X10	5X18	5X23	5X27
MTHS	.001	.001	.001	.01	.001	.001
UTIL			.001	.001	.001	NOT
CAT	.02	.001	.001	NOT	NOT	NOT
MODE	.001	.05	.05	.001	.001	.001

TABLE 46

RESULTS OF THE PAIRWISE COMPARISONS
OF THE GROUP DISTRIBUTIONS BY ANY OF 4 VARIABLES
FOR EACH OF THE RELEASES

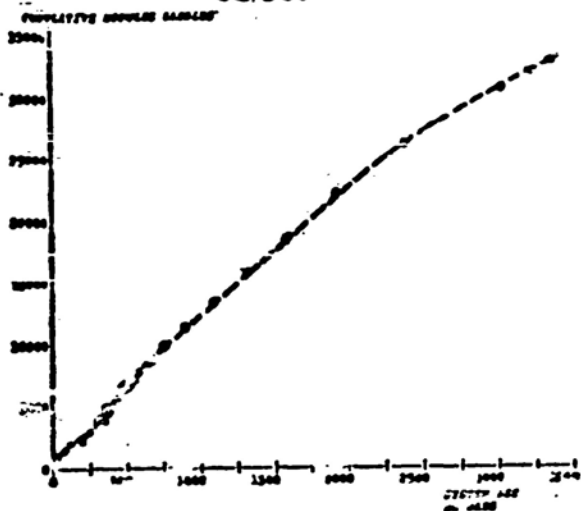
VARIABLE	GROUPS	3X	4X	5X10	5X18	5X23	5X27
MTHS	G3AG5		.001	.001	.001	.001	.001
	G5AG6	.001	.001	.001	.001	.001	.001
	G6AG3		.001	.001	.001	.001	.001
UTIL	G3AG5			NOT	.001	.001	NOT
	G5AG6			.001	.001	.001	NOT
	G6AG3			.001	.001	.001	NOT
CAT	G3AG5		NOT	.001	NOT	NOT	NOT
	G5AG6	.02	.001	NOT	NOT	NOT	NOT
	G6AG3		.001	.001	NOT	NOT	NOT
MODE	G3AG5		.02	NOT	.05	.001	.001
	G5AG6	.001	NOT	.02	.01	.01	NOT
	G6AG3		.02	.02	.001	.001	.001



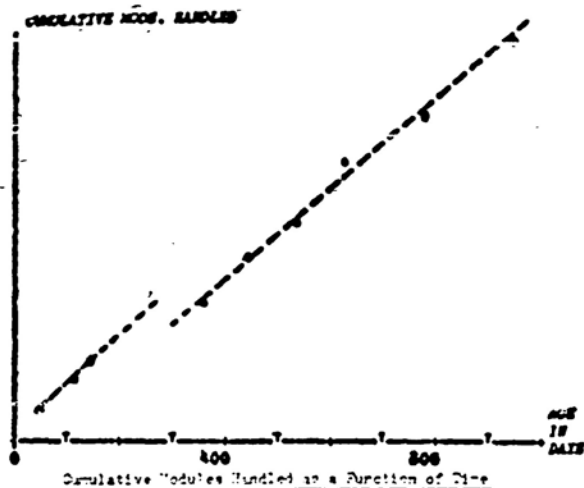
Laws of Program Evolution

FIGURE 1

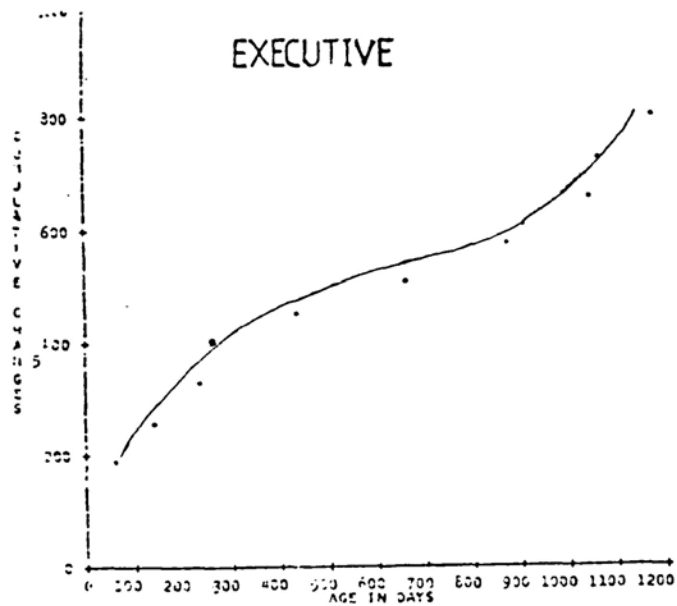
OS/360



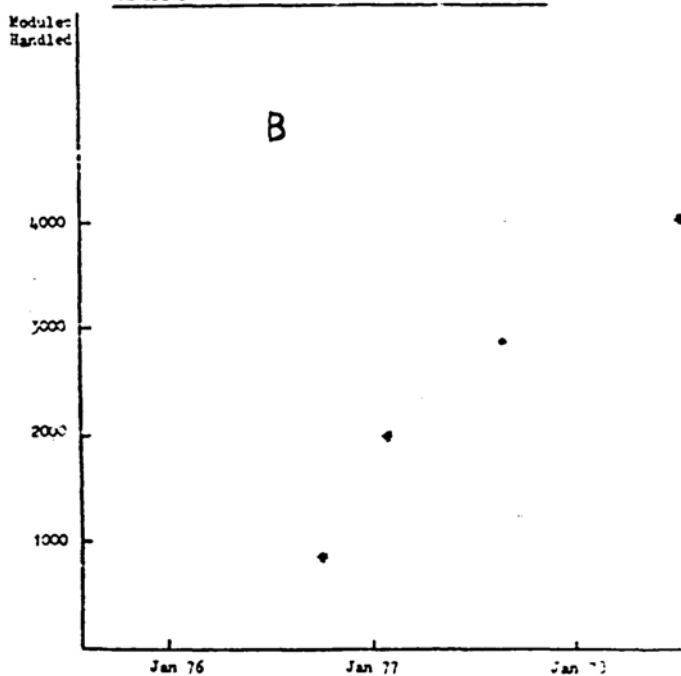
OMEGA



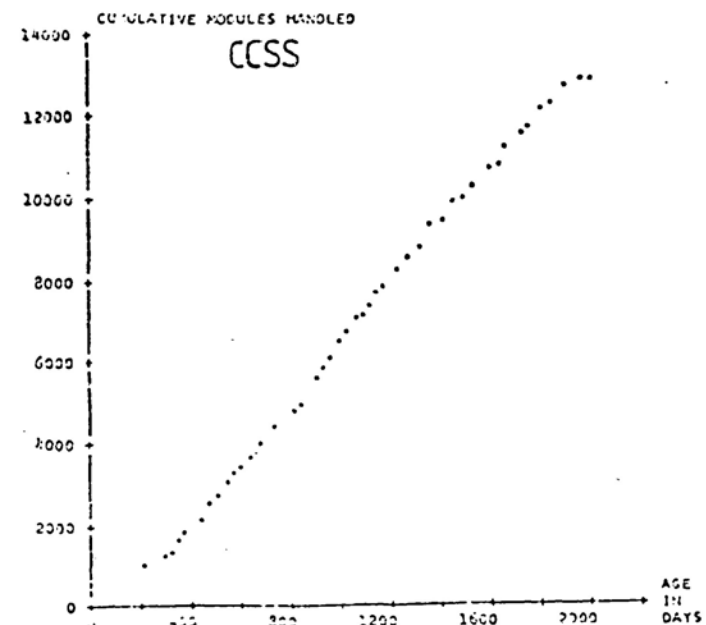
EXECUTIVE



B



CCSS



DOS

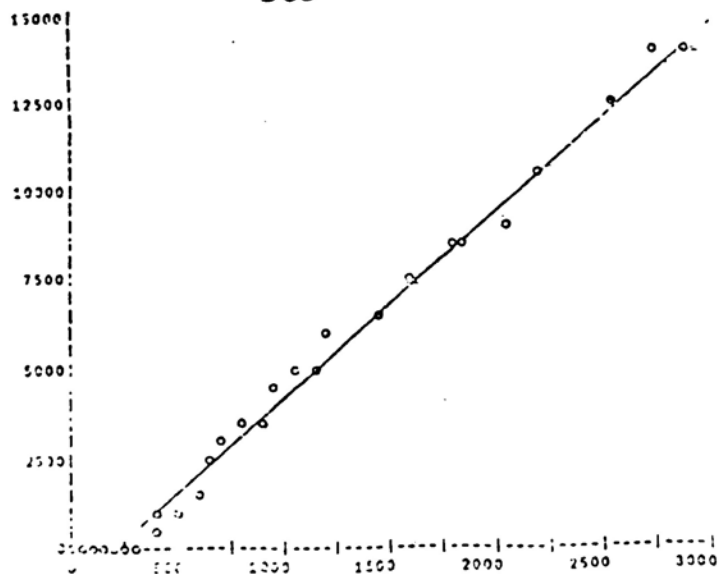
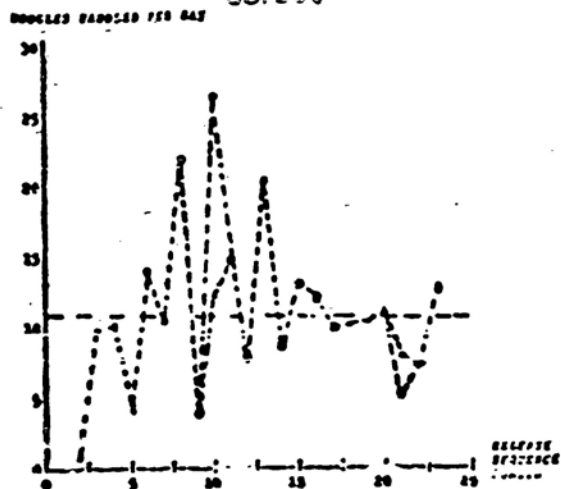
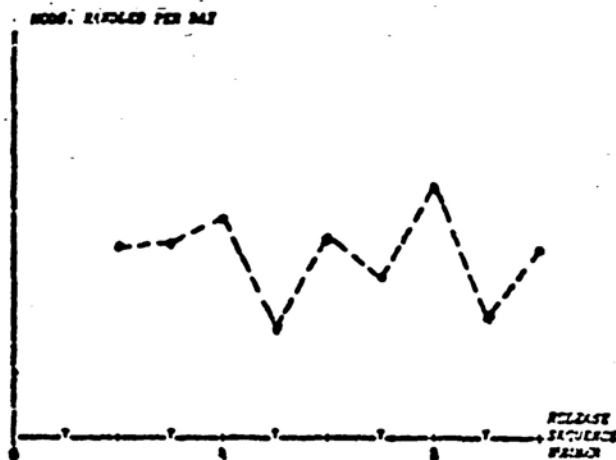


FIGURE 2

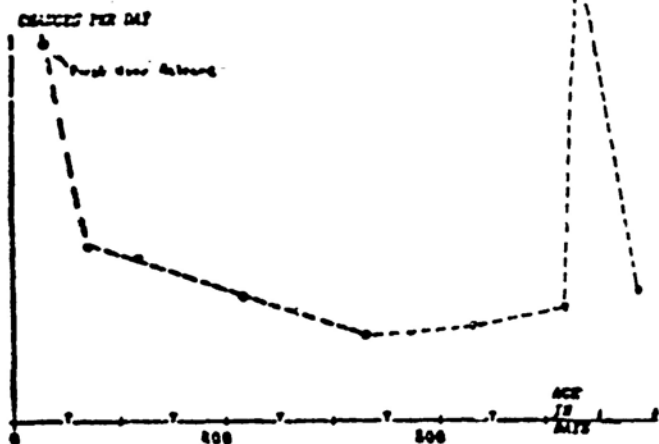
OS/360



OMEGA

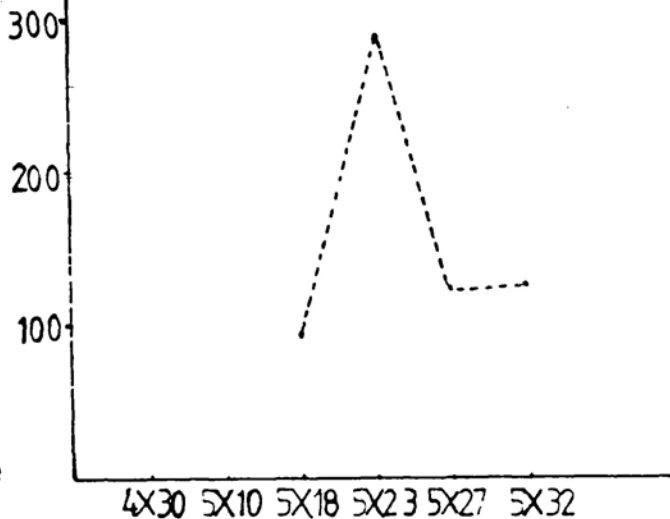


EXECUTIVE

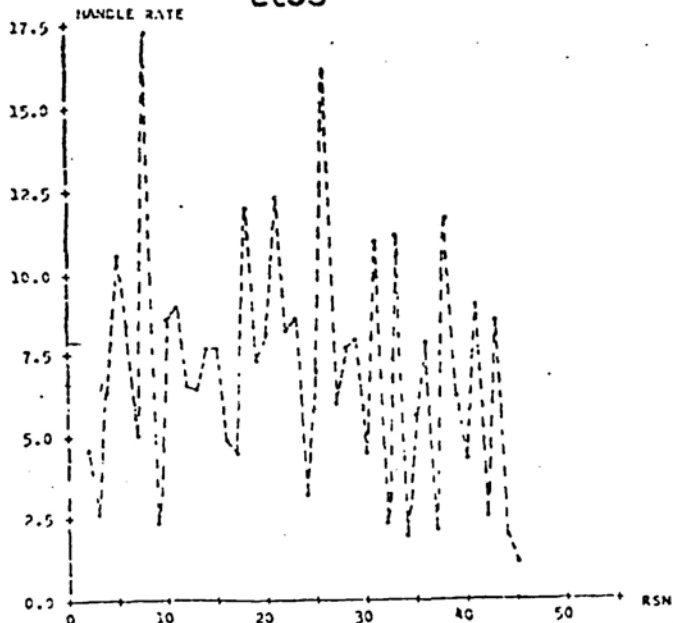


B

MODULES HANDLED PER MONTH



CCSS



DOS

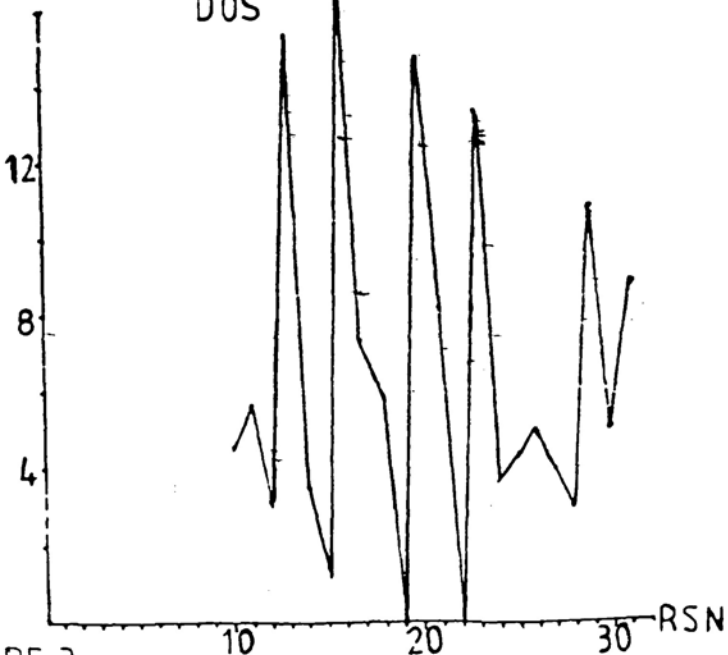
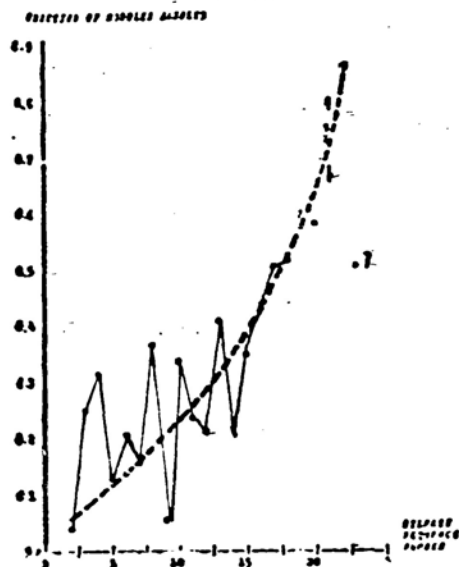
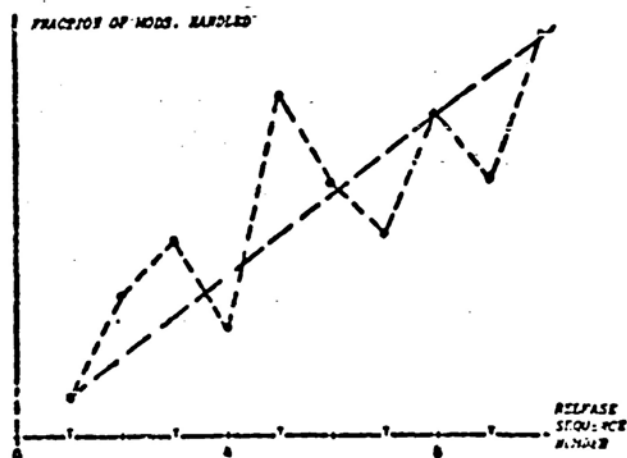


FIGURE 3



EXECUTIVE
NOT
AVAILABLE



Fraction of Modules Handled as a Function of
Release Sequence Number

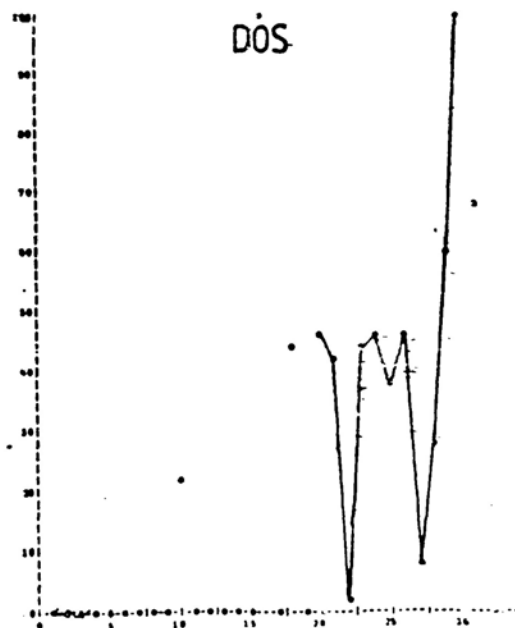
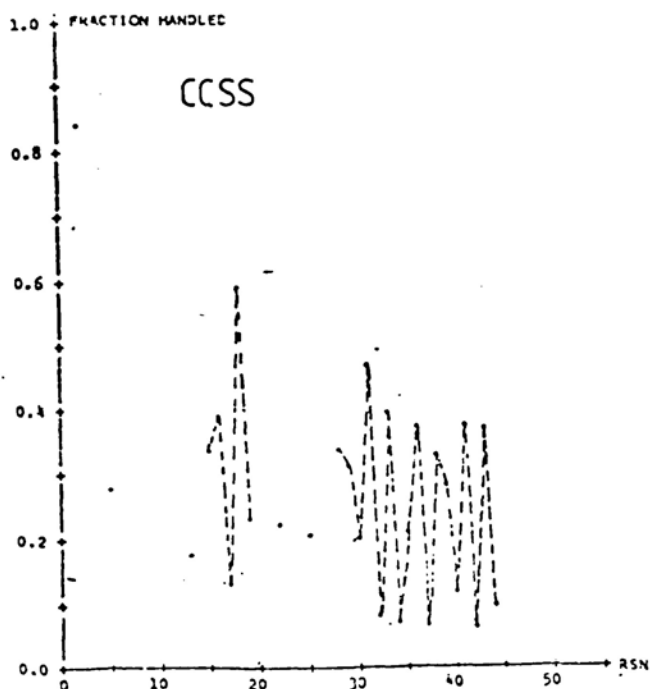
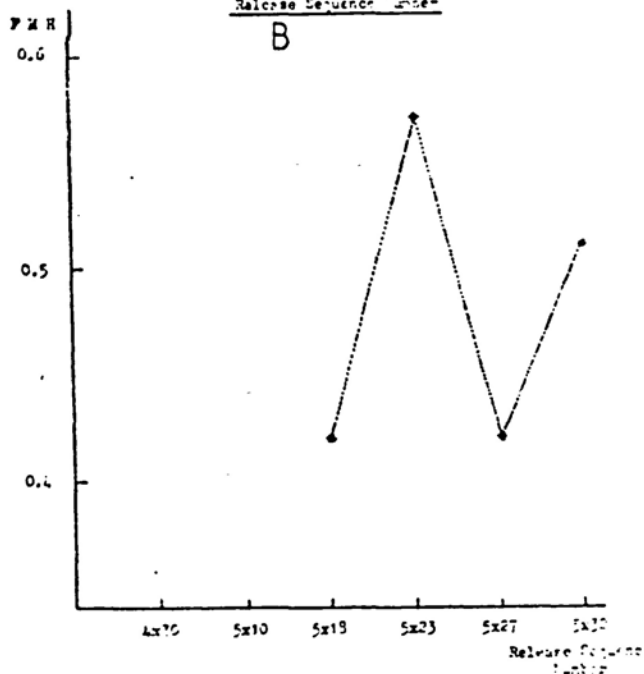
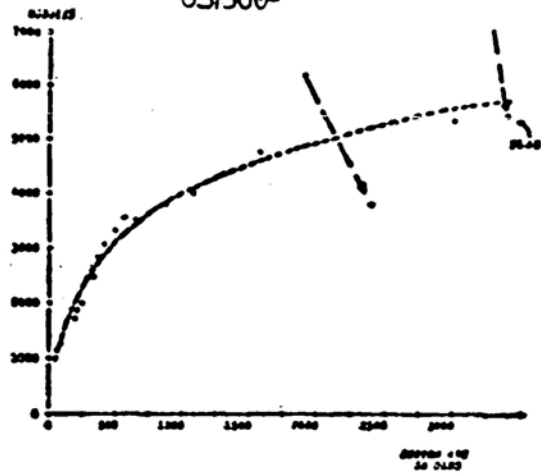
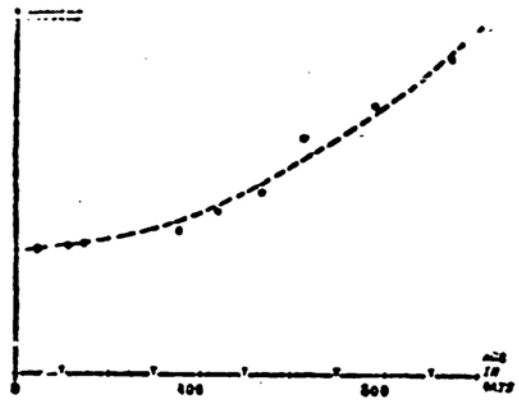


FIGURE 4

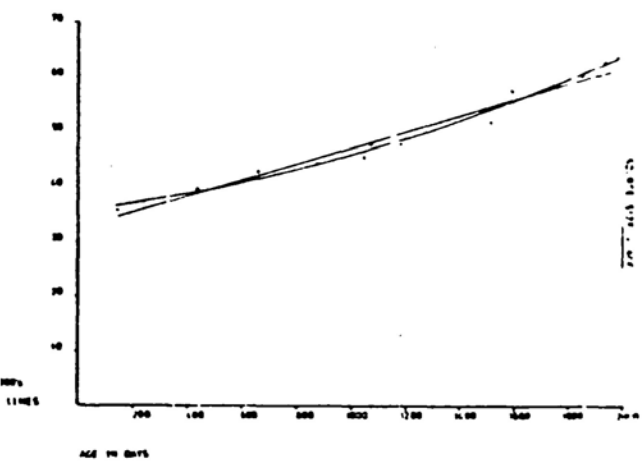
OS/360



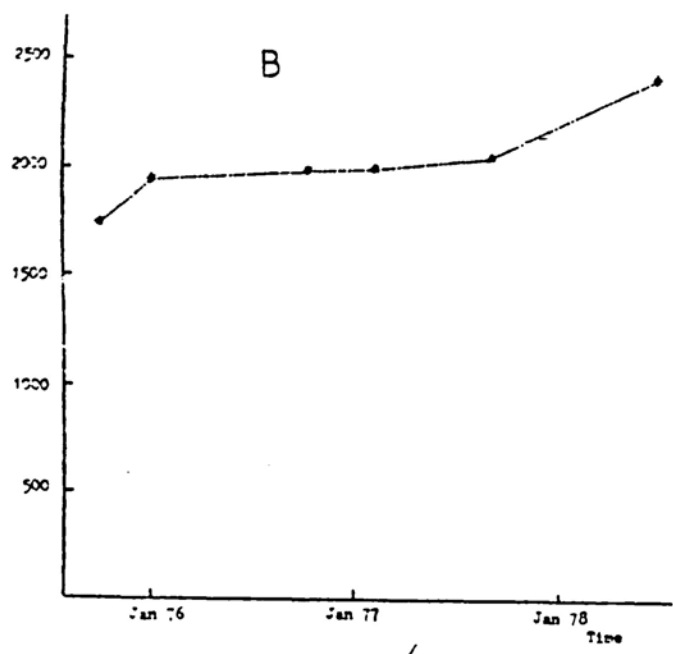
OMEGA



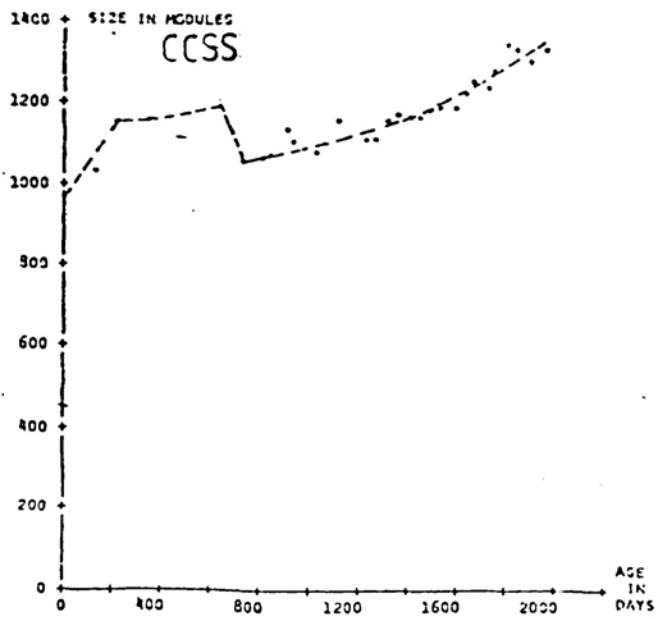
EXECUTIVE



B



CCSS



DOS

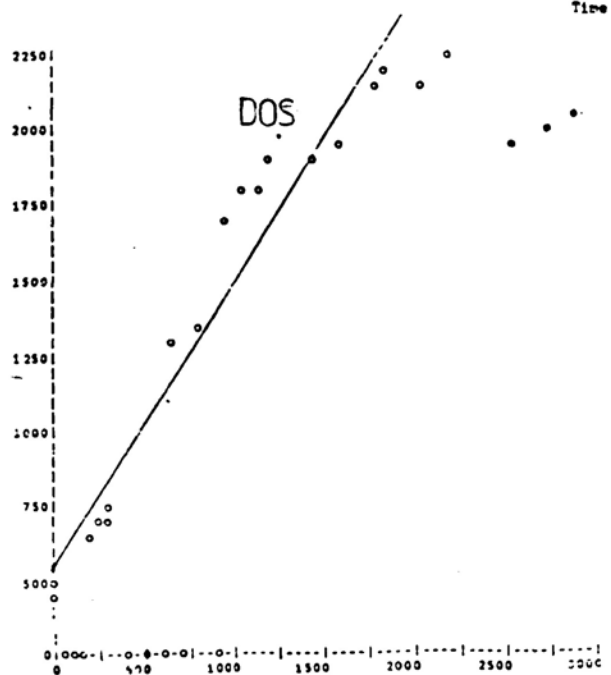
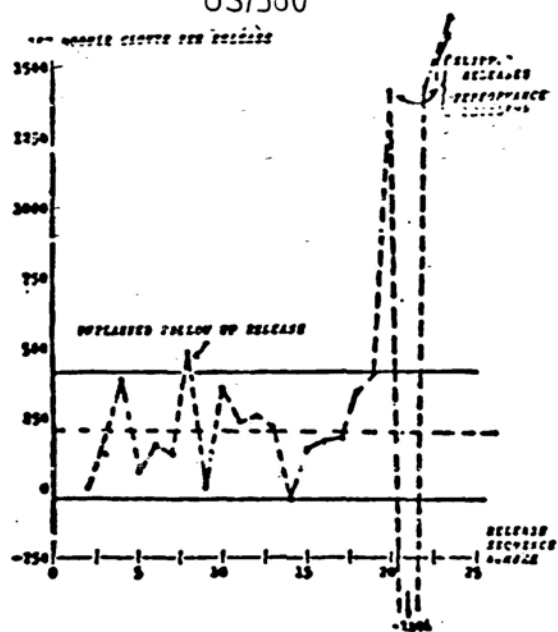
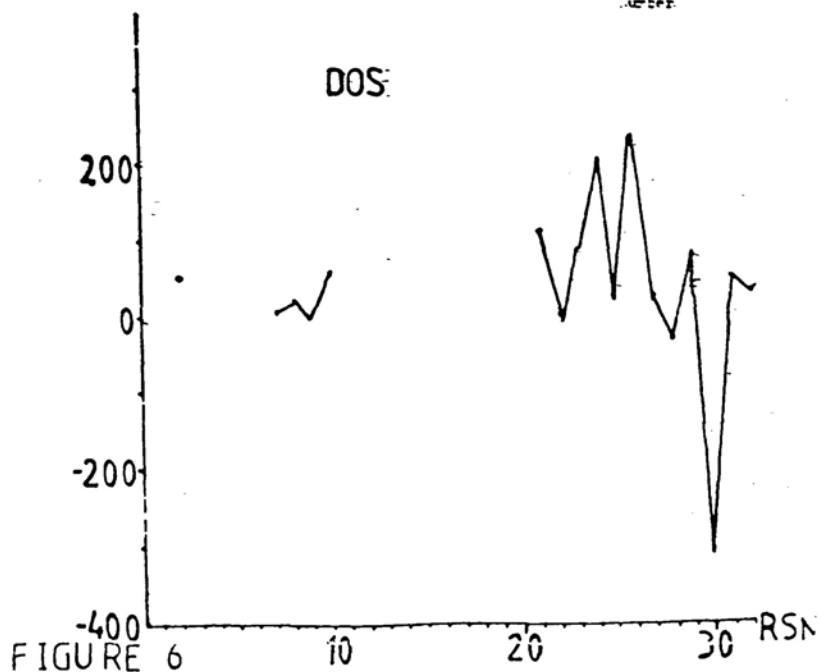
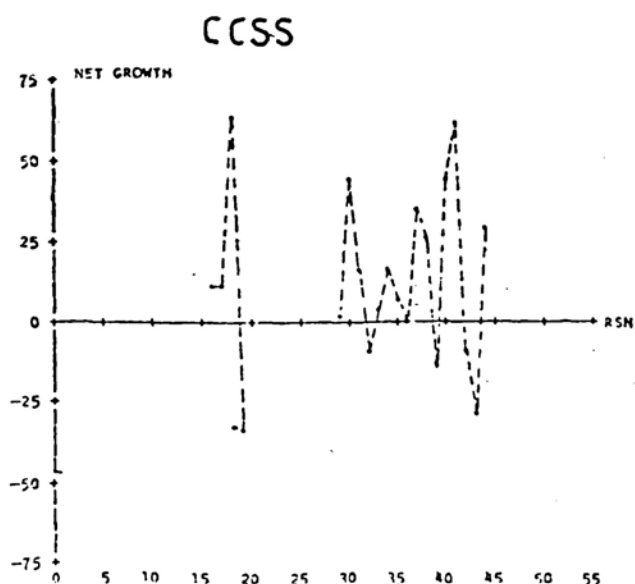
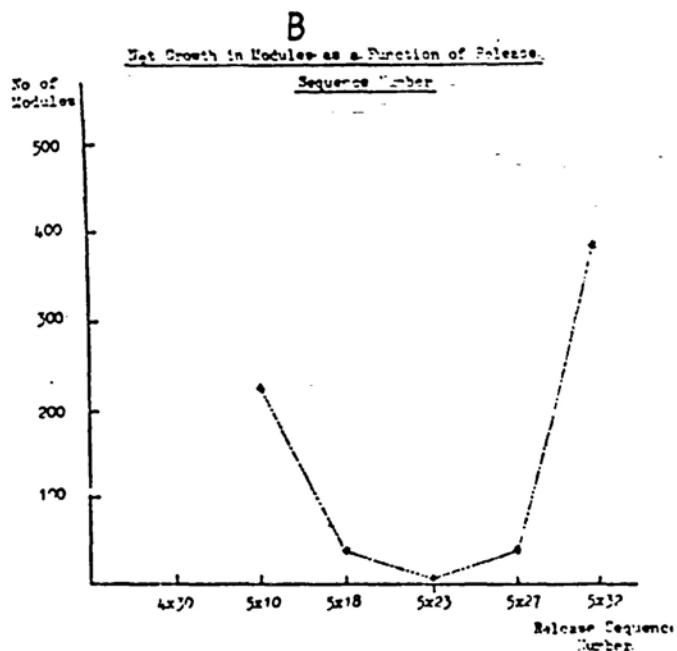
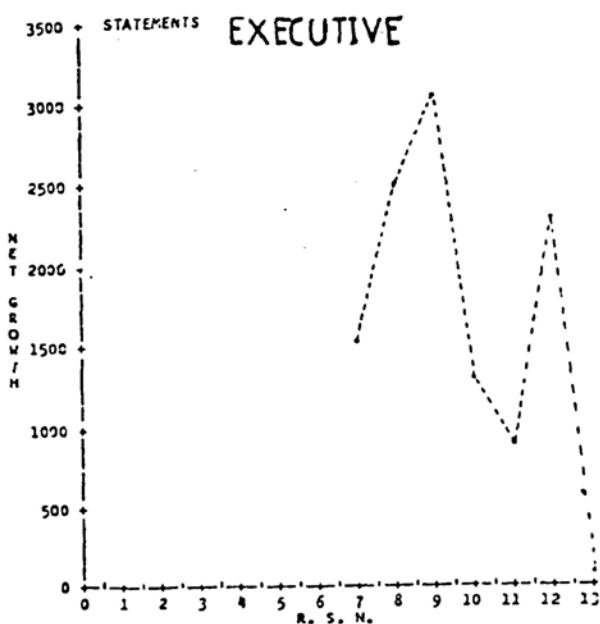


FIGURE 5

OS/360

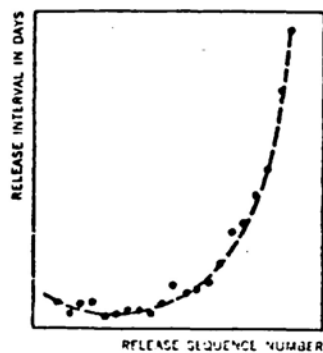


OMEGA
NOT
AVAILABLE



09360

Increasing release
interval



OMEGA
NOT
AVAILABLE

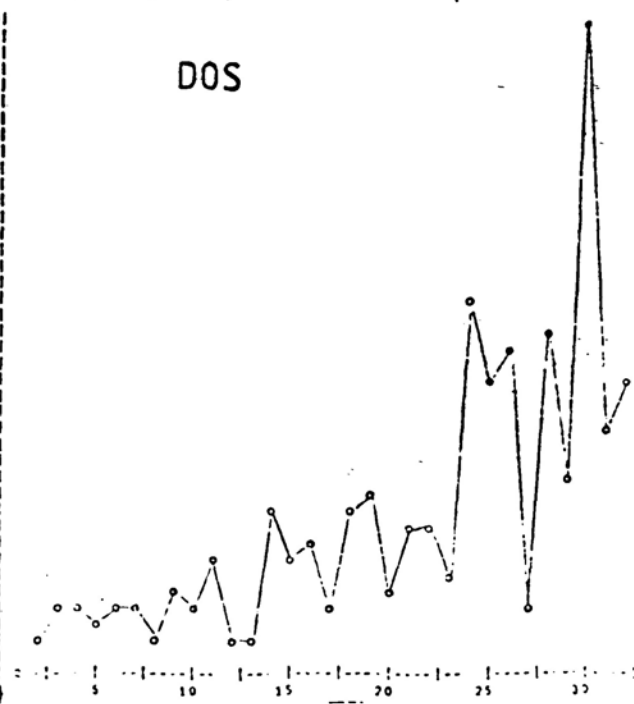
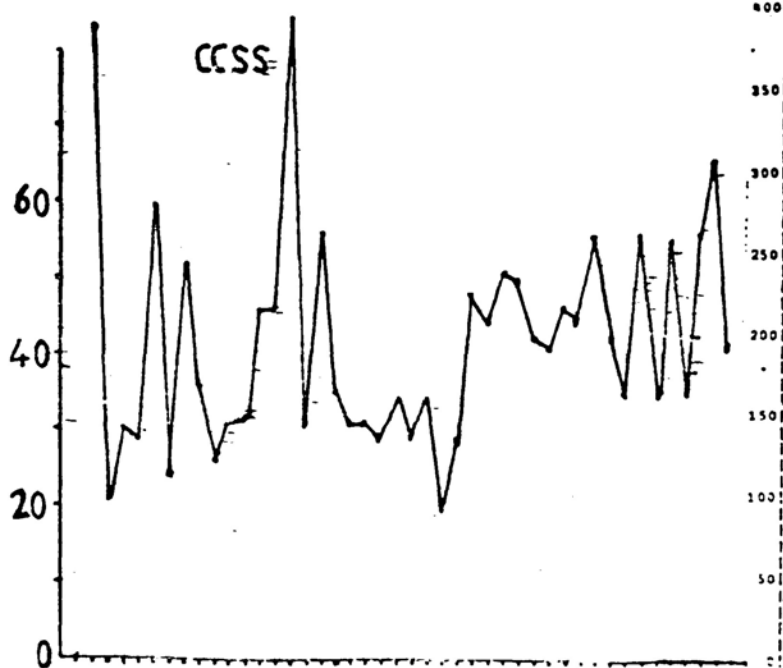
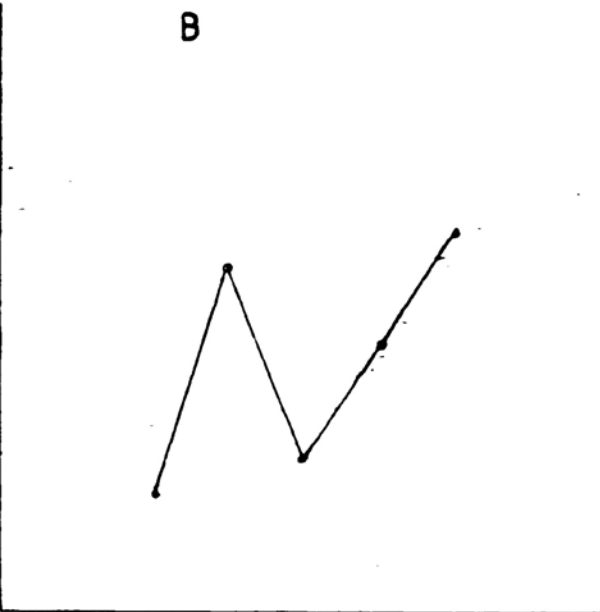
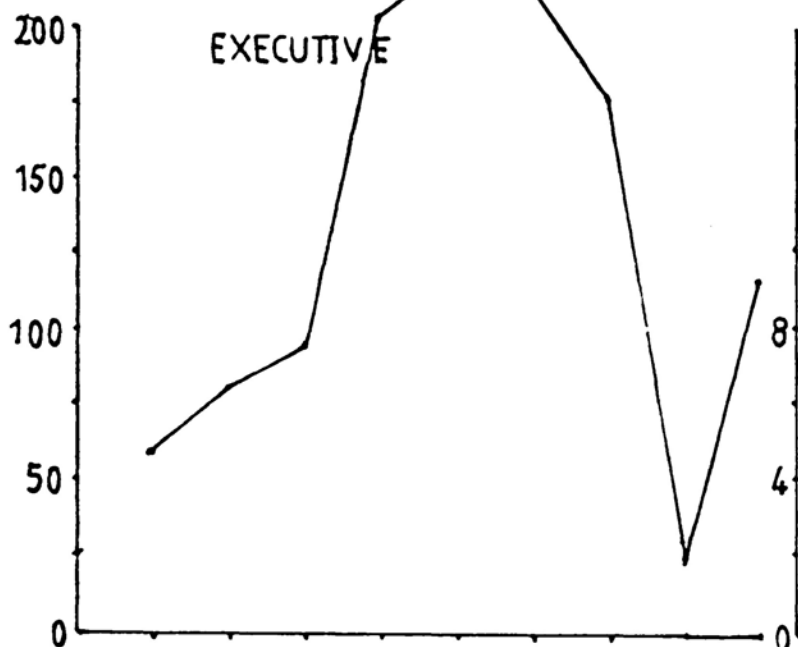
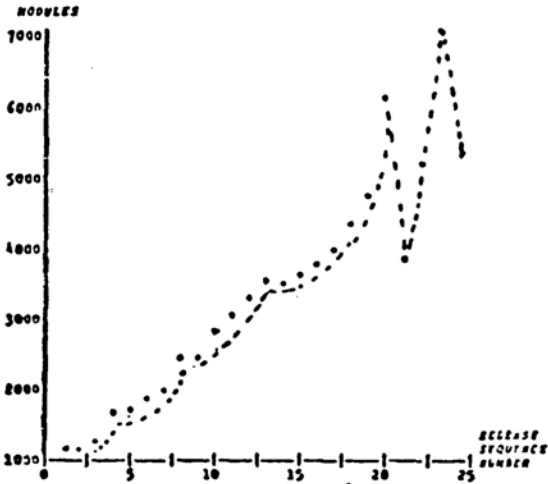


FIGURE 7

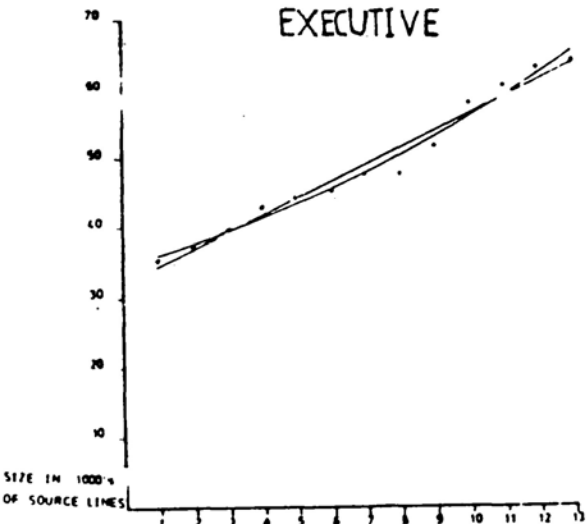
FIGURES

OS/360

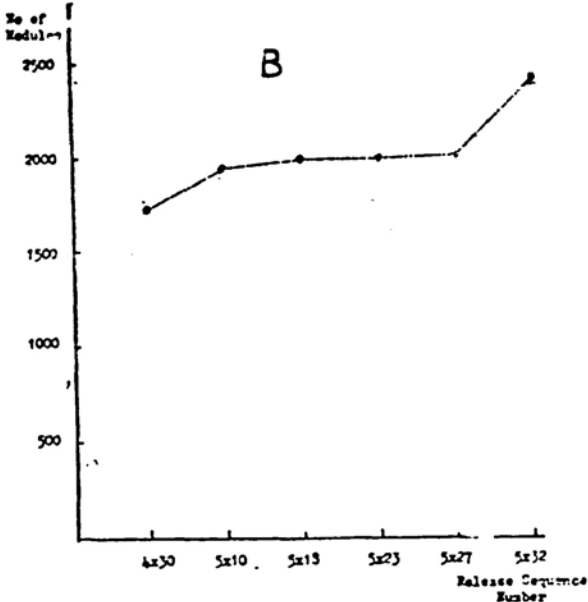
OMEGA
NOT
AVAILABLE



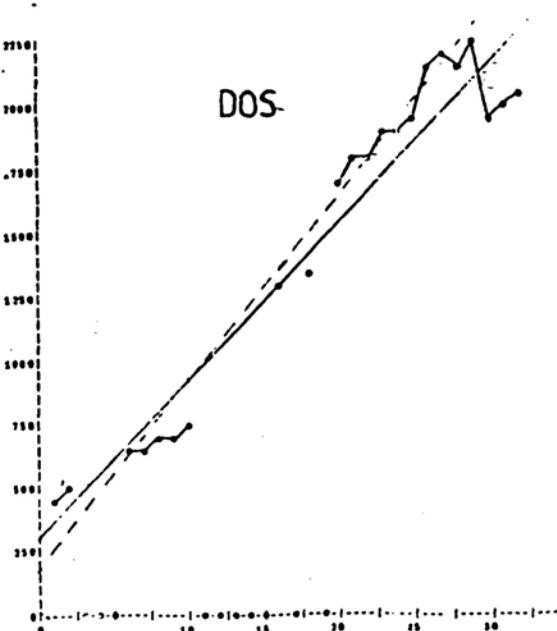
EXECUTIVE



Size in Modules as a Function of Release Sequence Number



DOS



CCSS

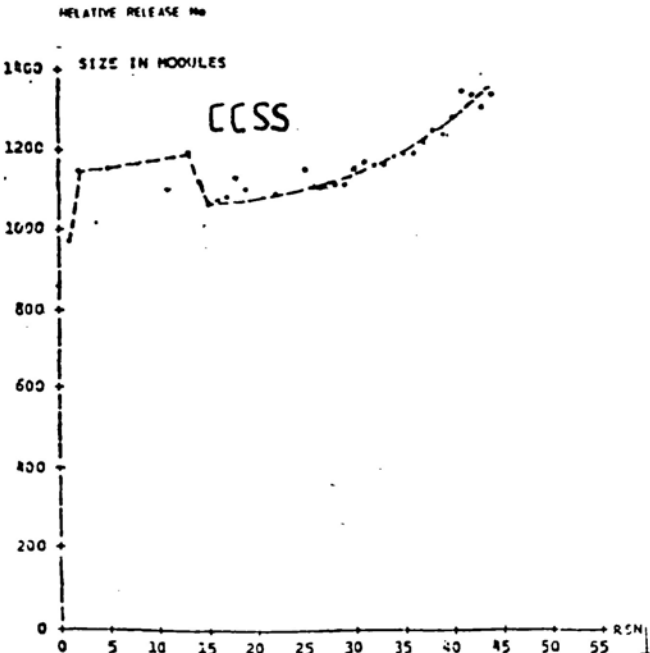
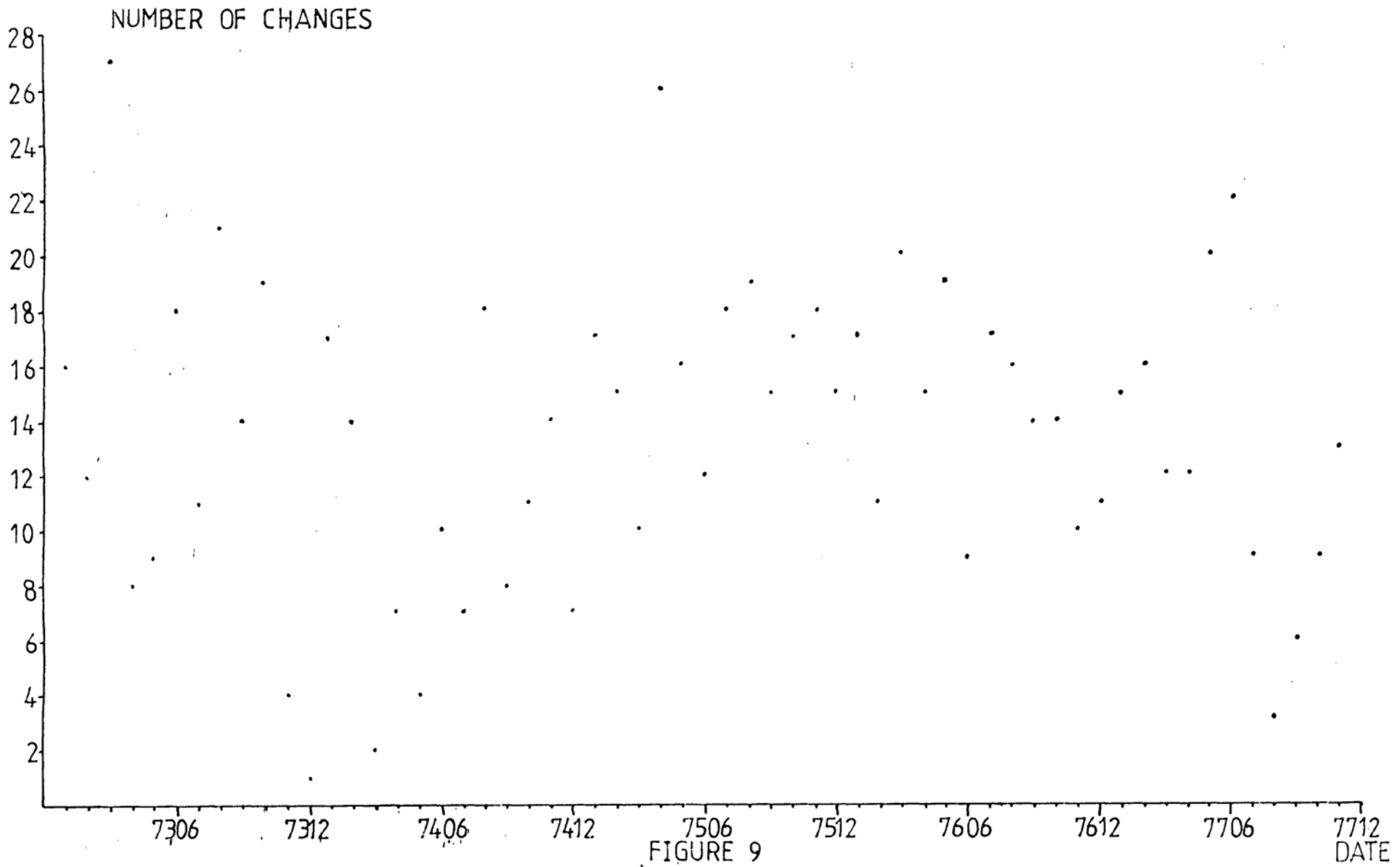
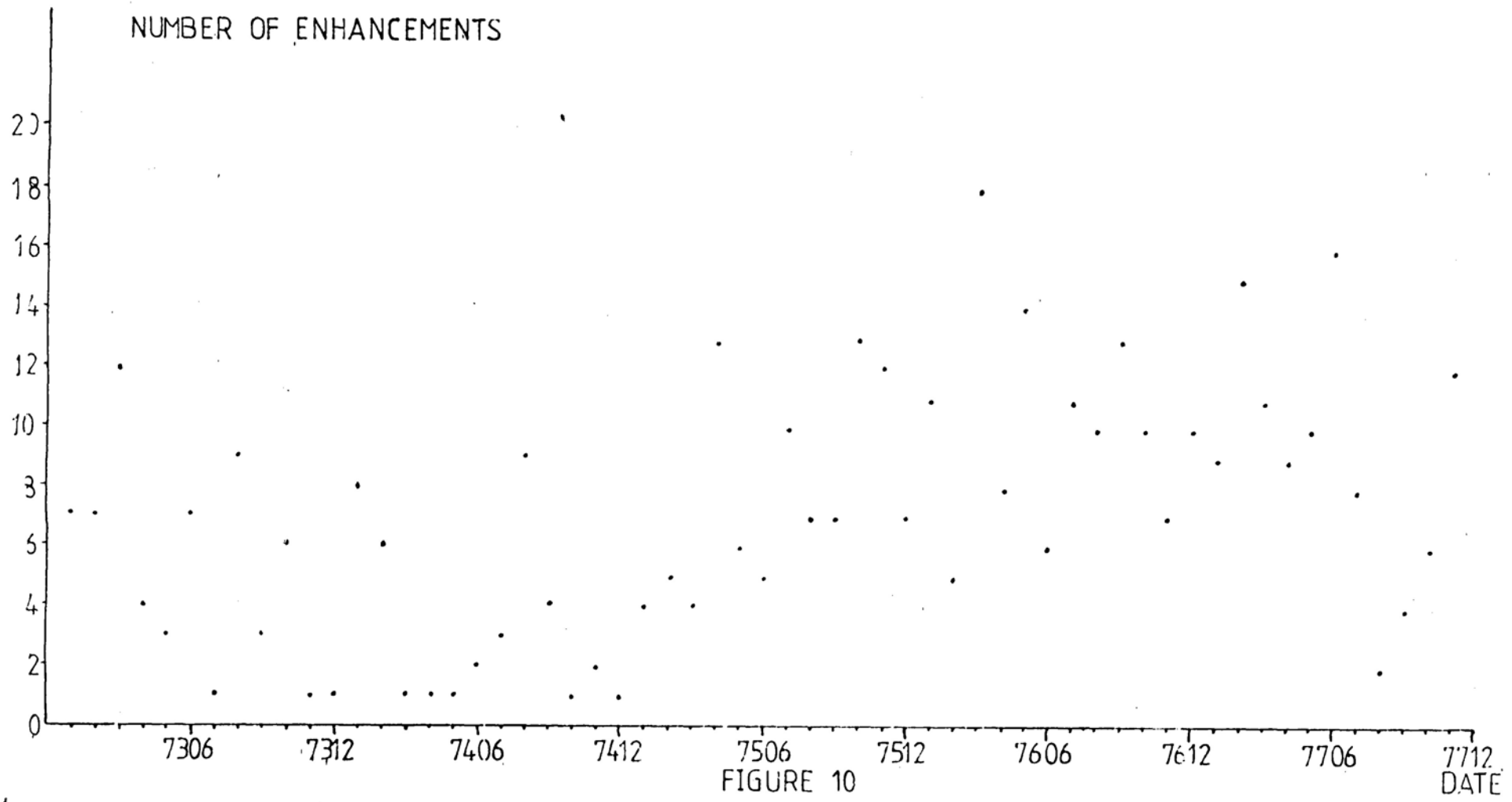
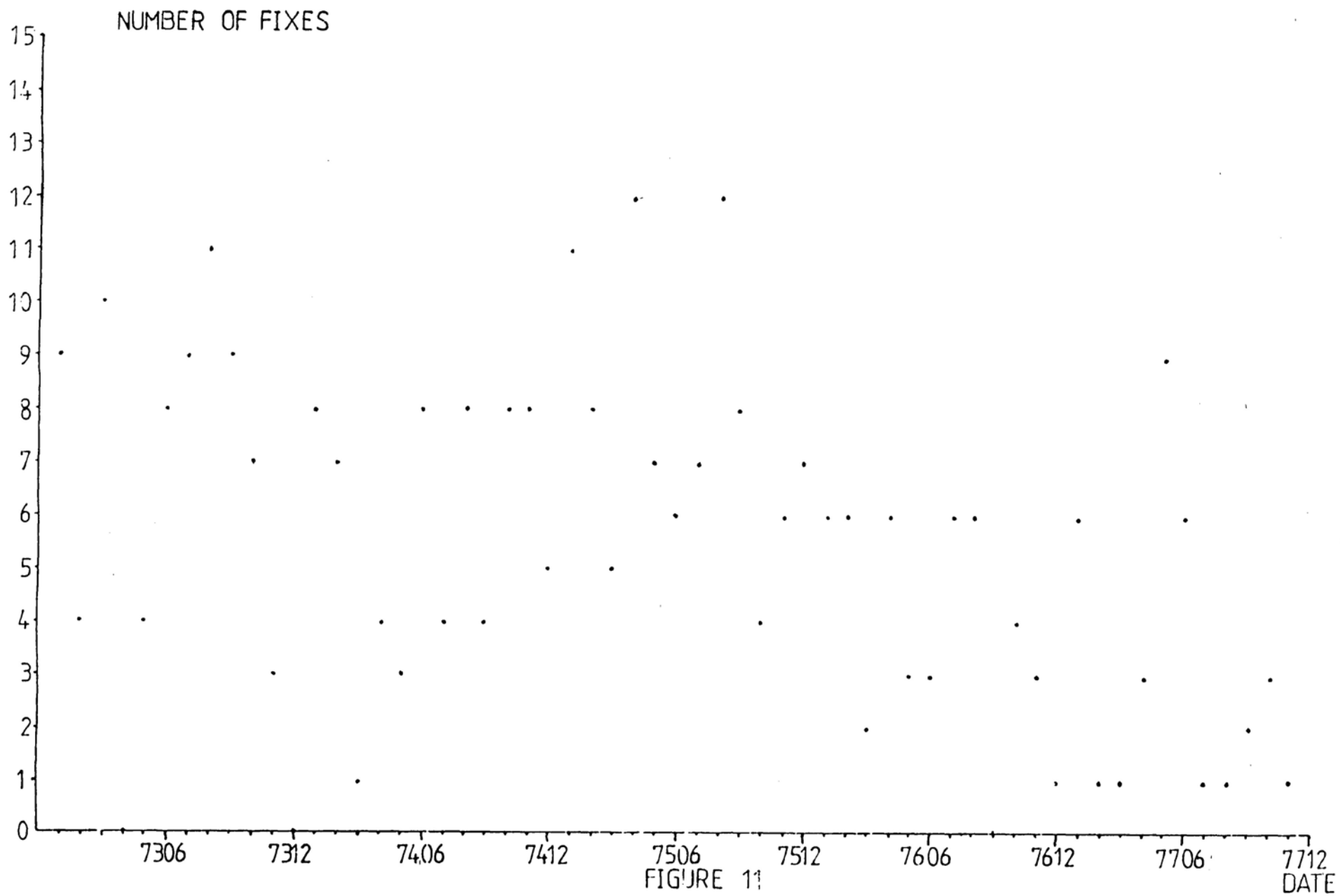
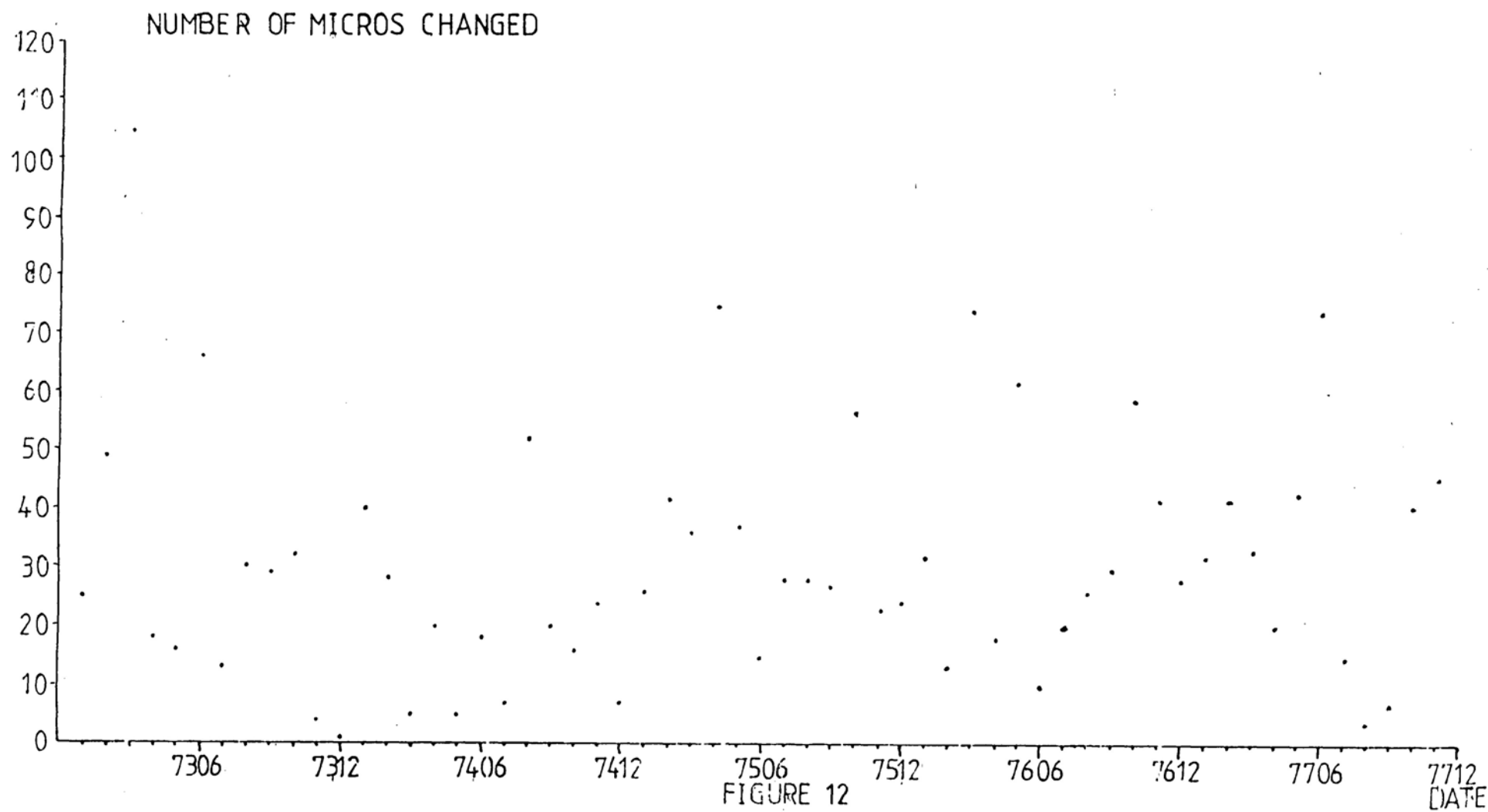


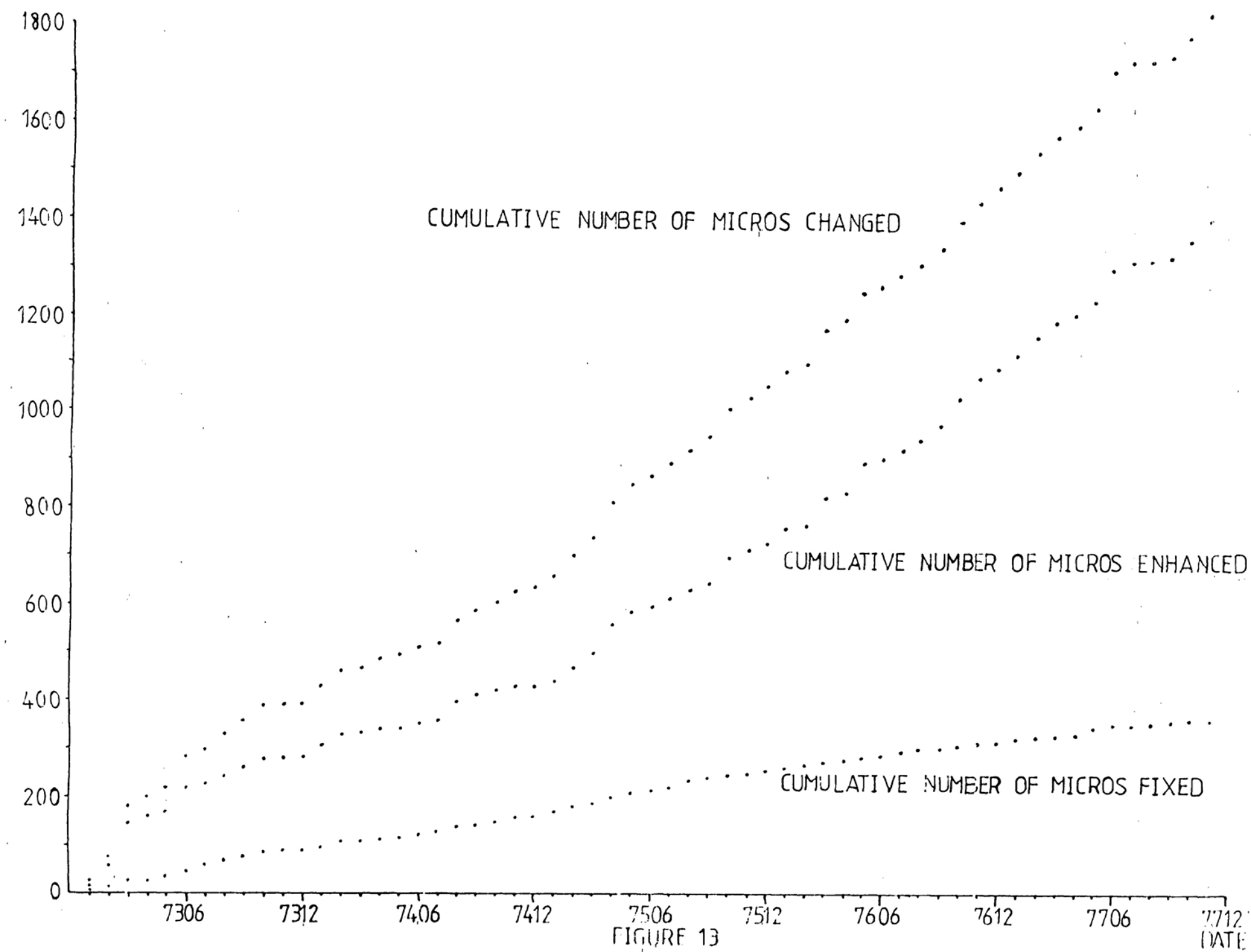
FIGURE 8











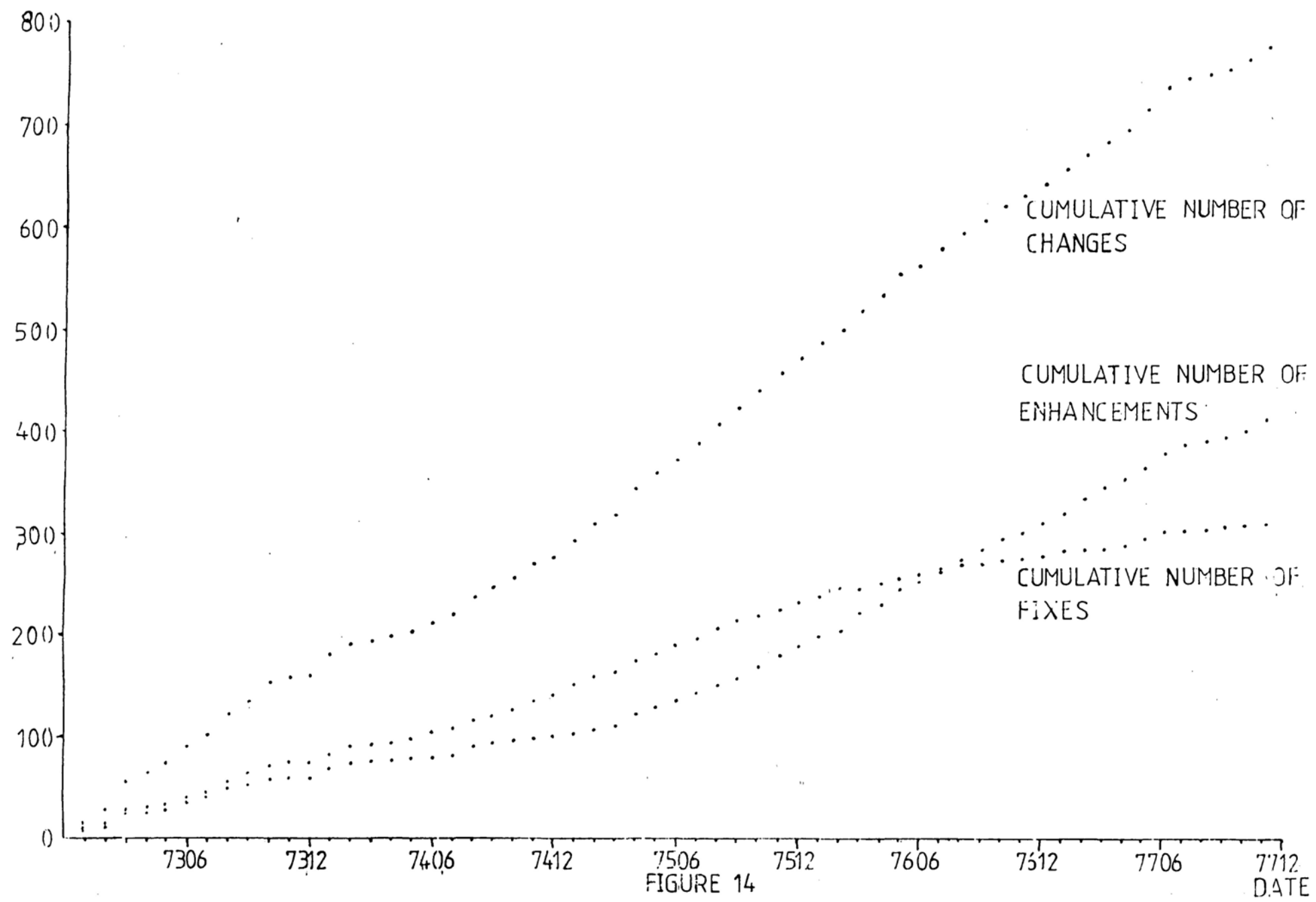
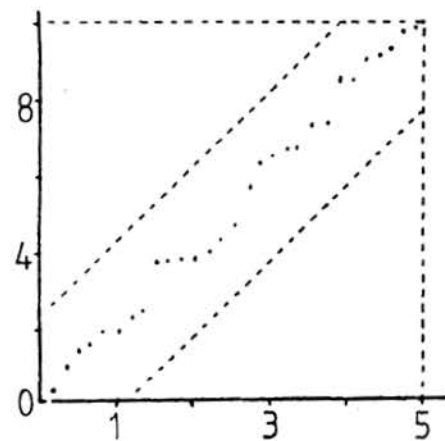
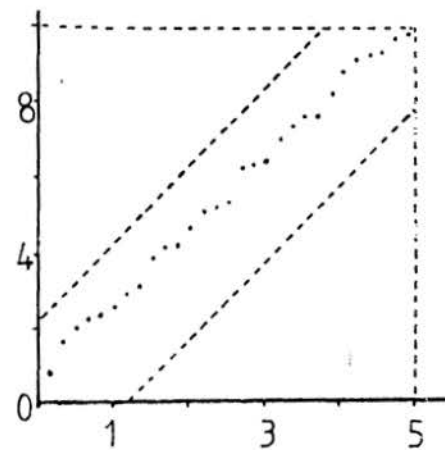


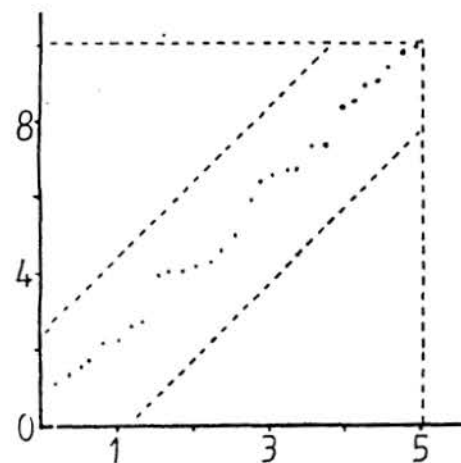
FIGURE 14



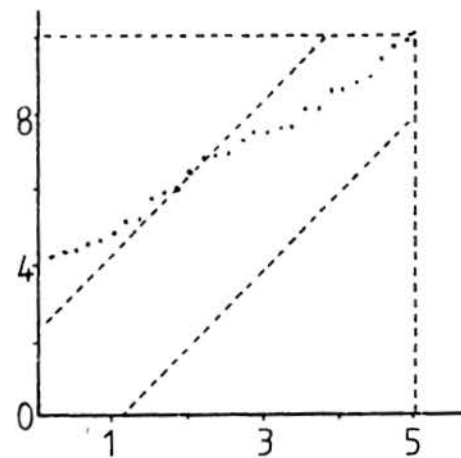
MICROS CHANGED



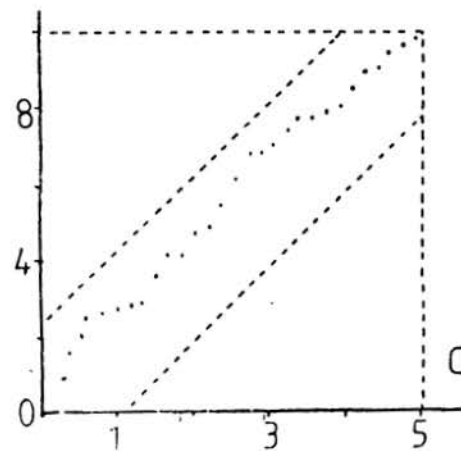
CHANGES



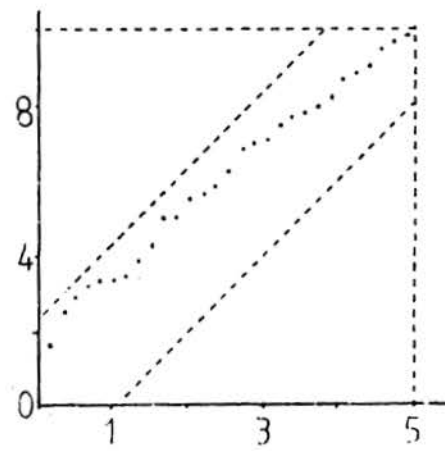
MICROS ENHANCED



ENHANCEMENTS

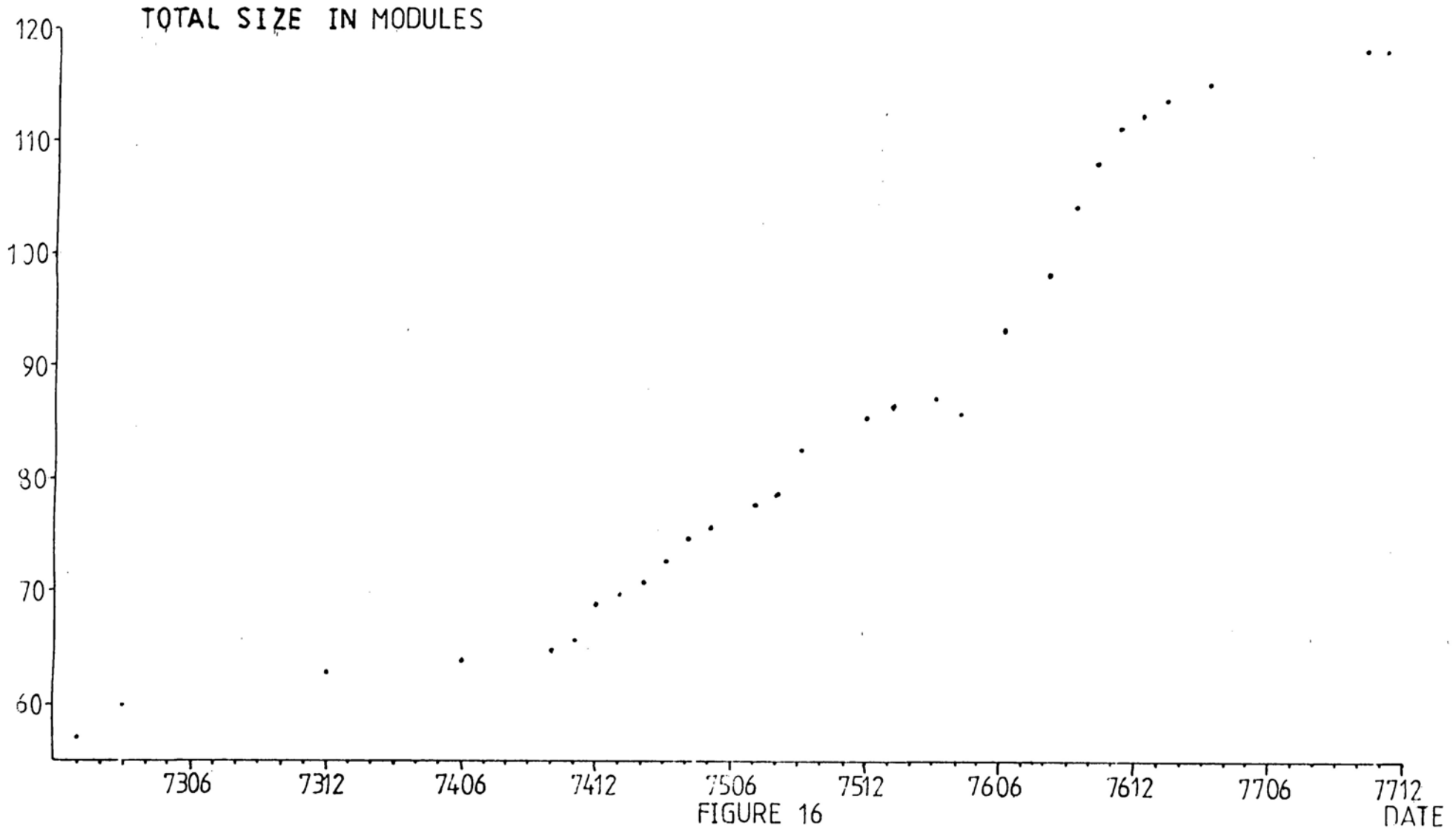


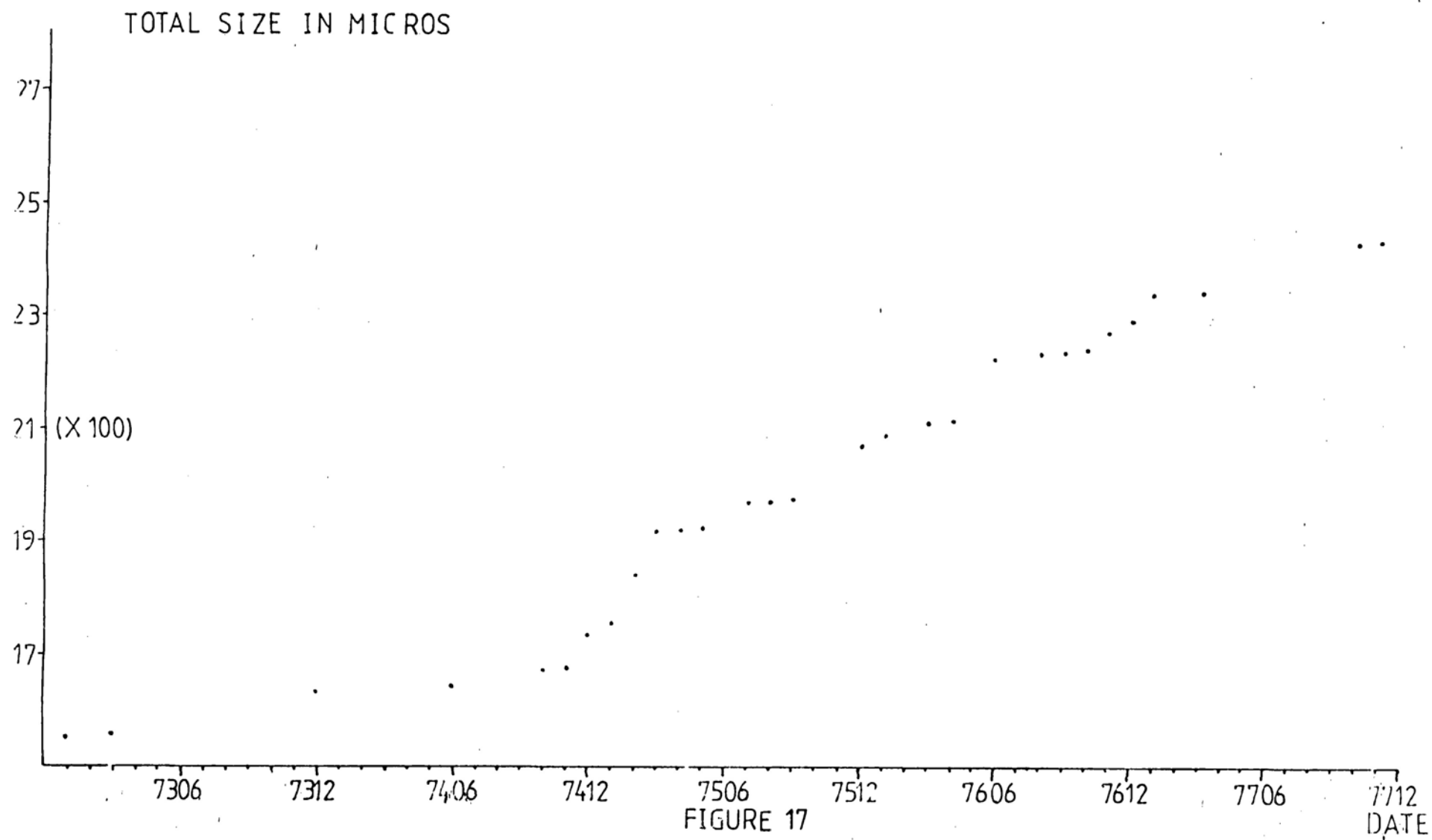
MICROS FIXED

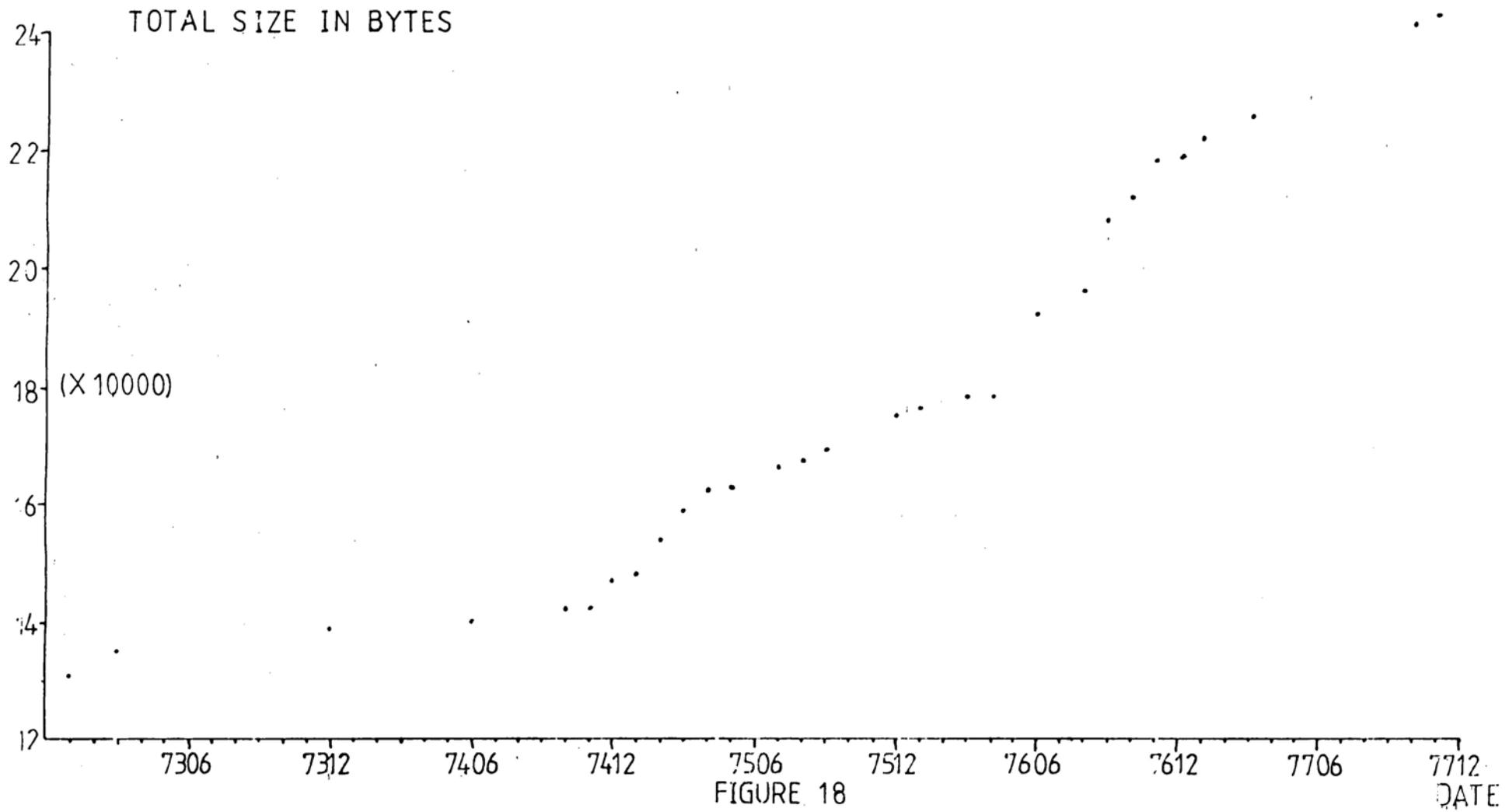


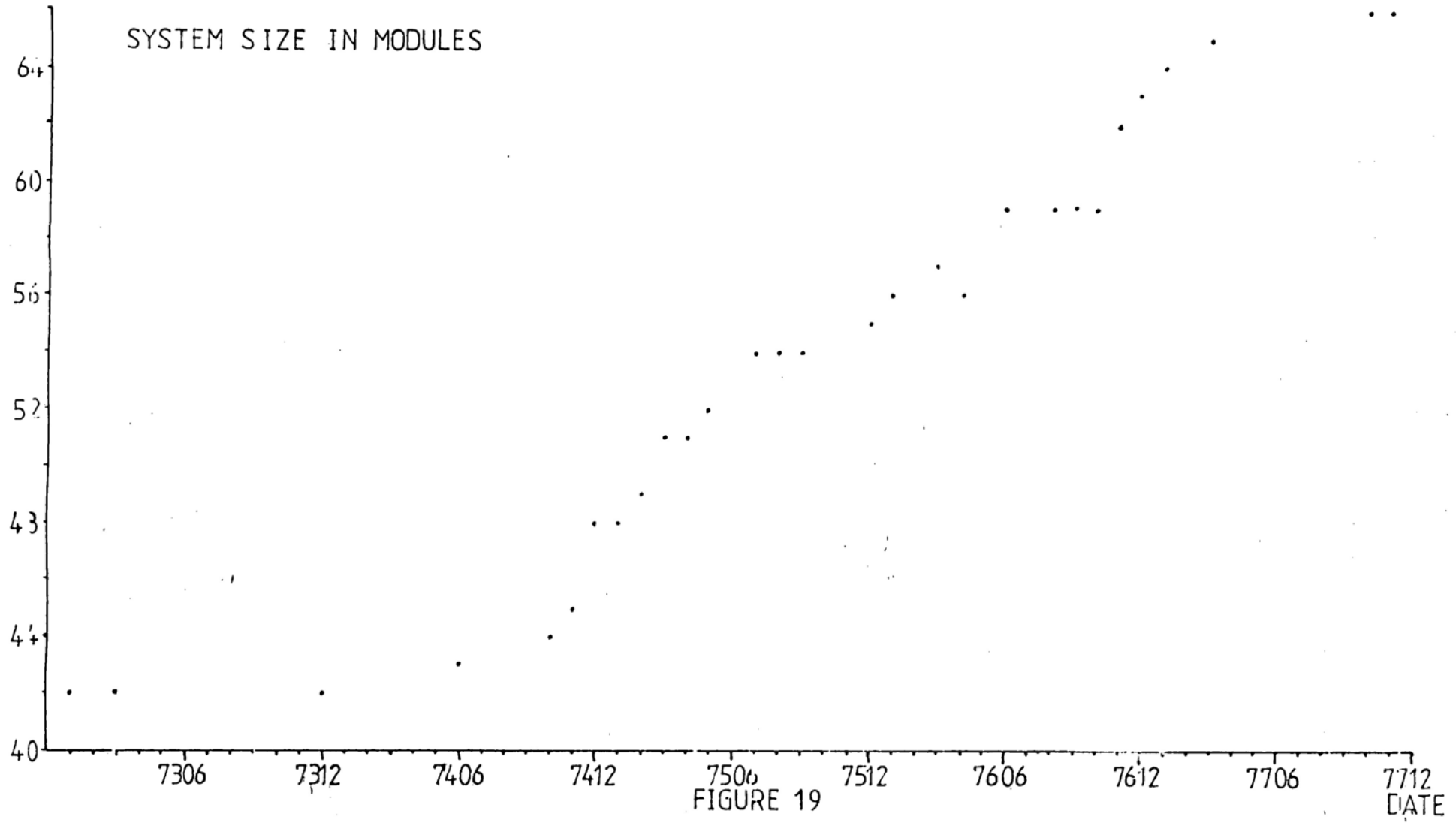
FIXES

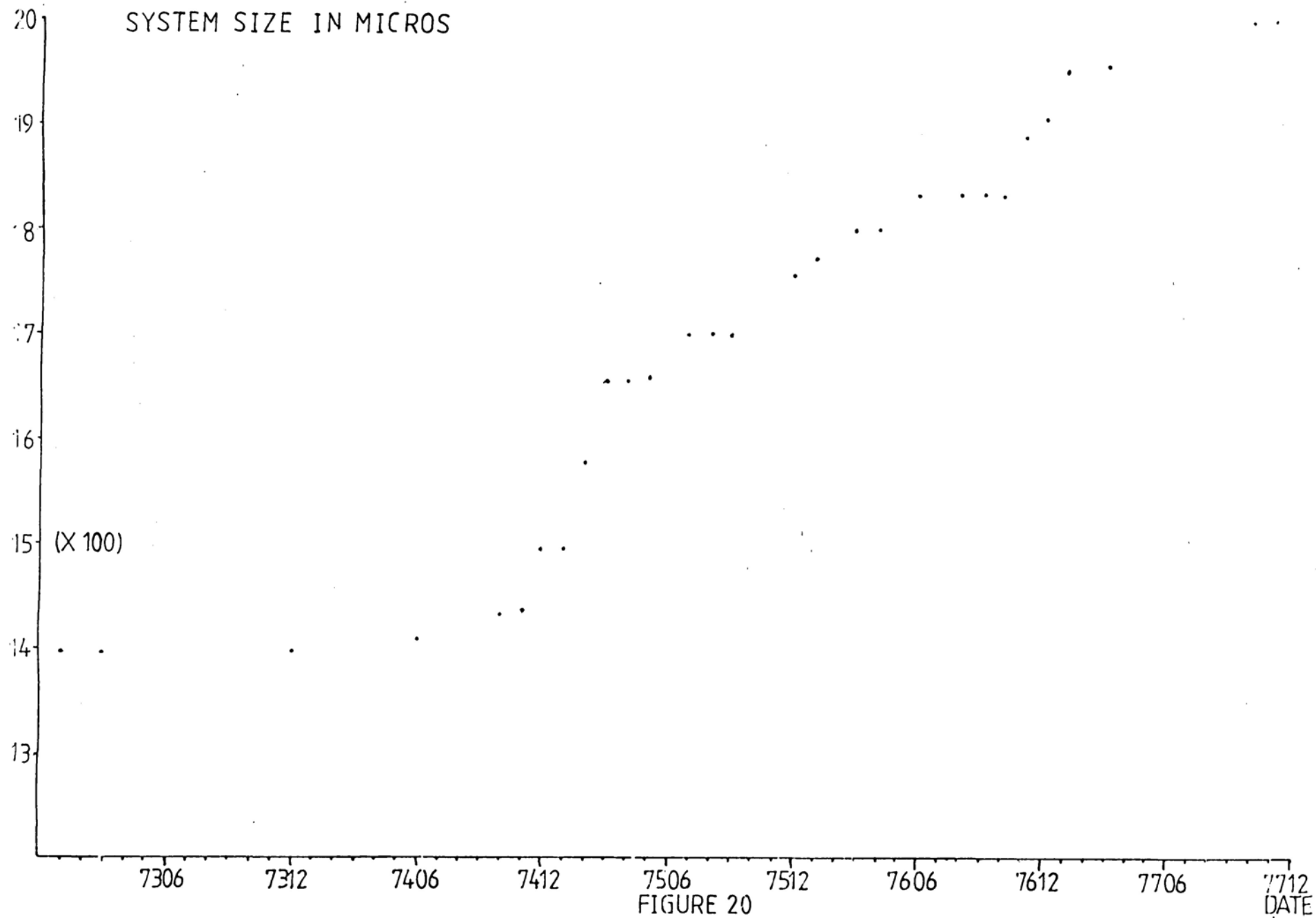
CUMULATIVE PERIODOGRAMS
FIGURE 15

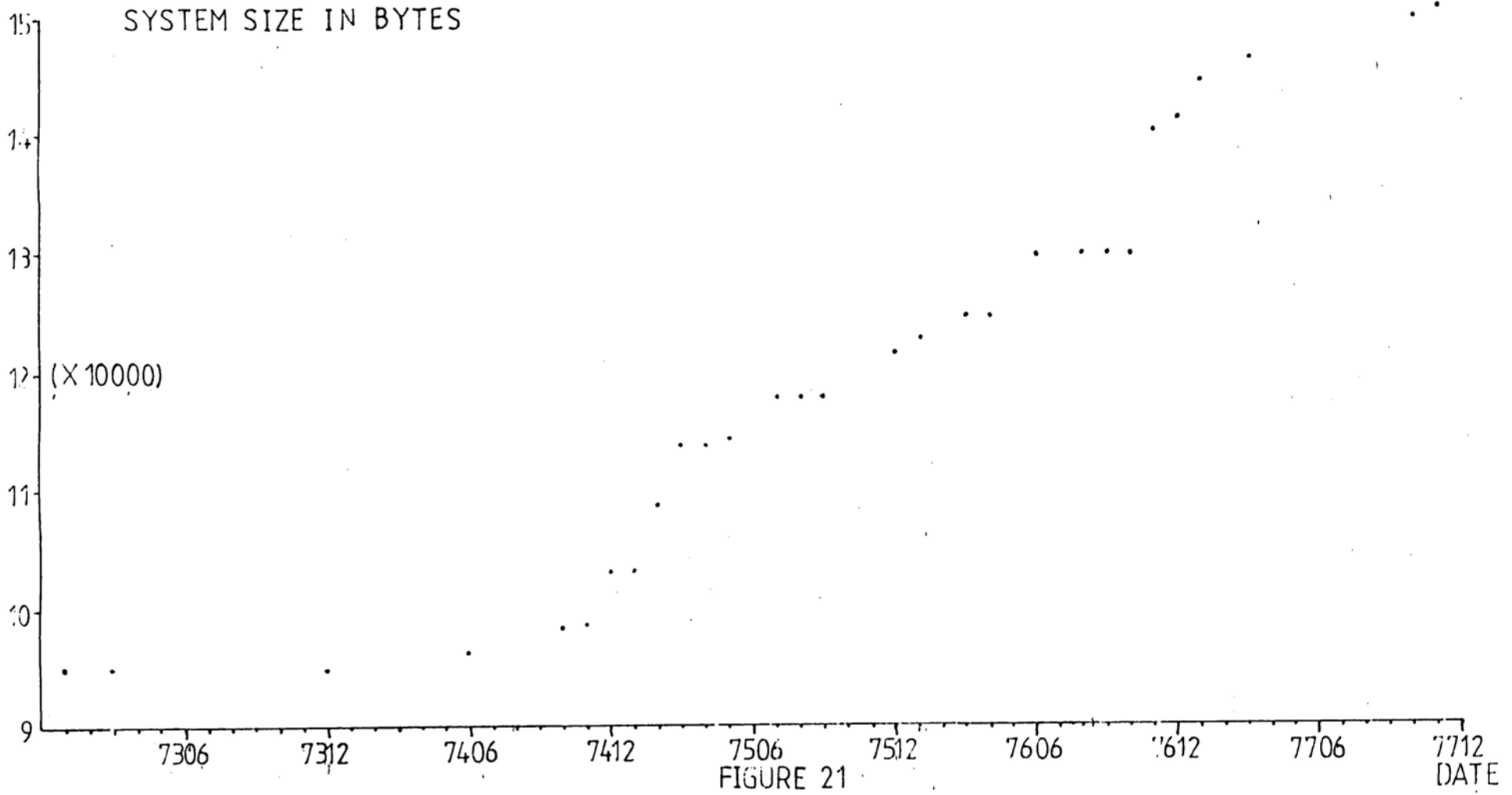


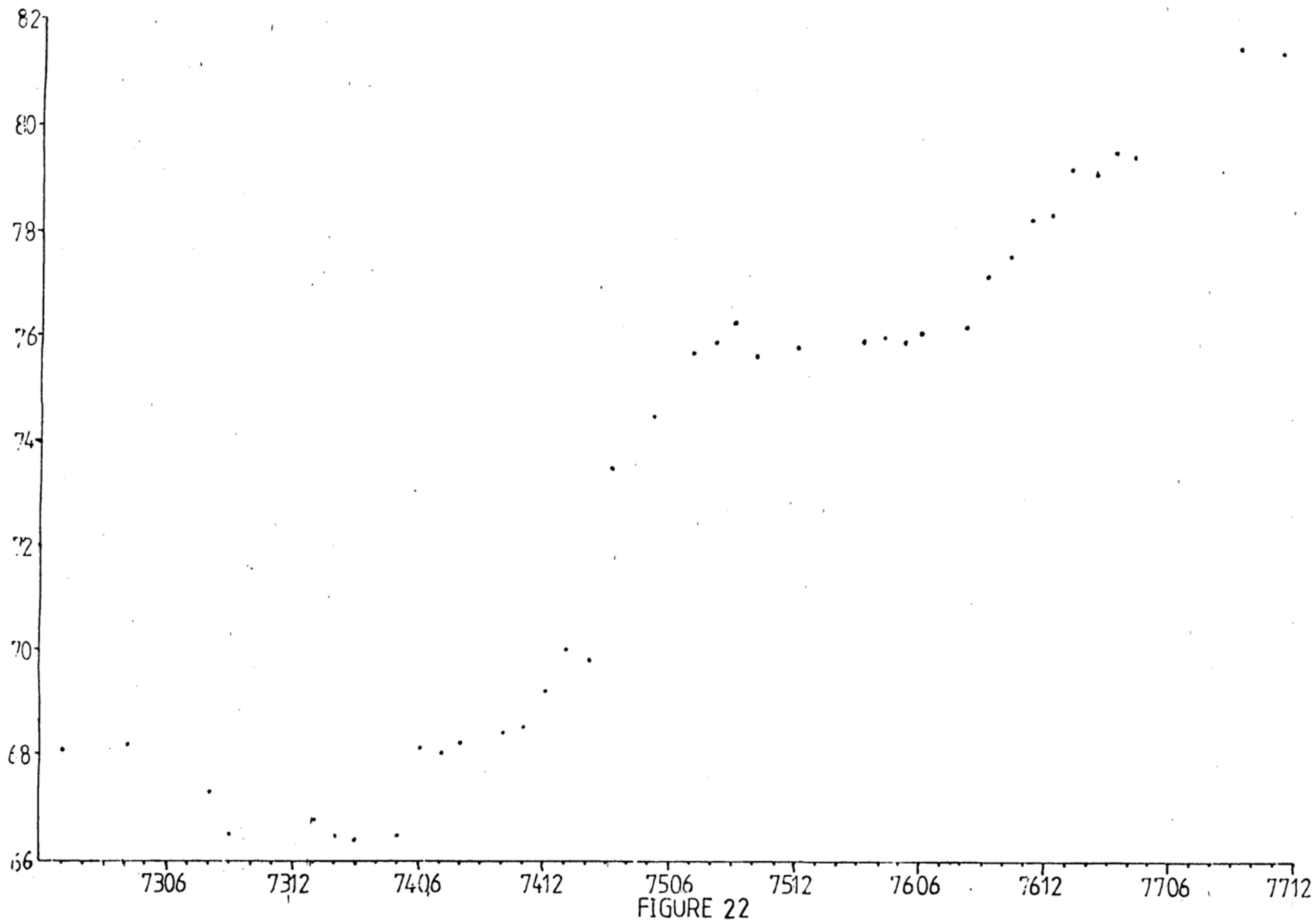






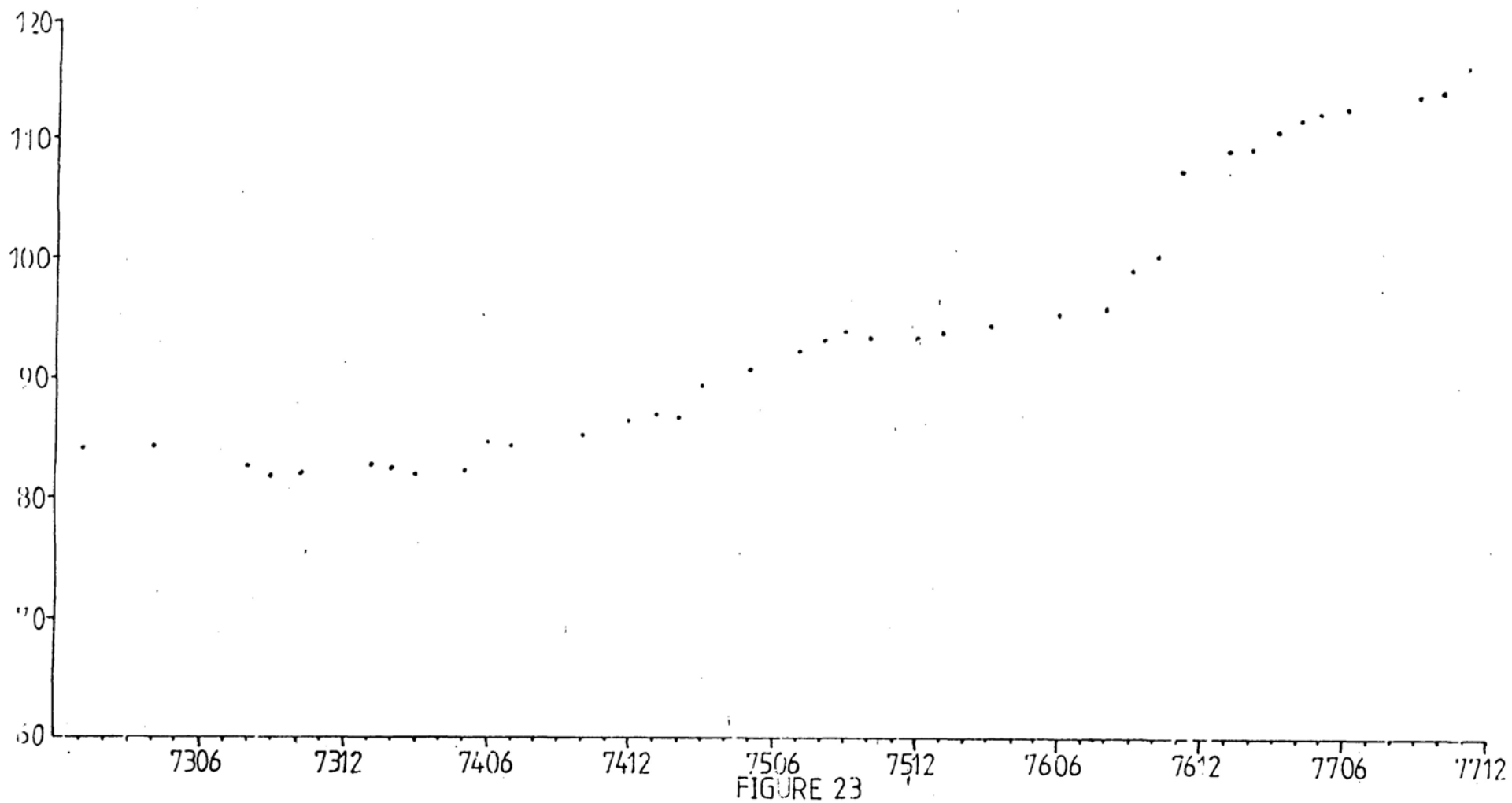






FIGURES

PAGE 249



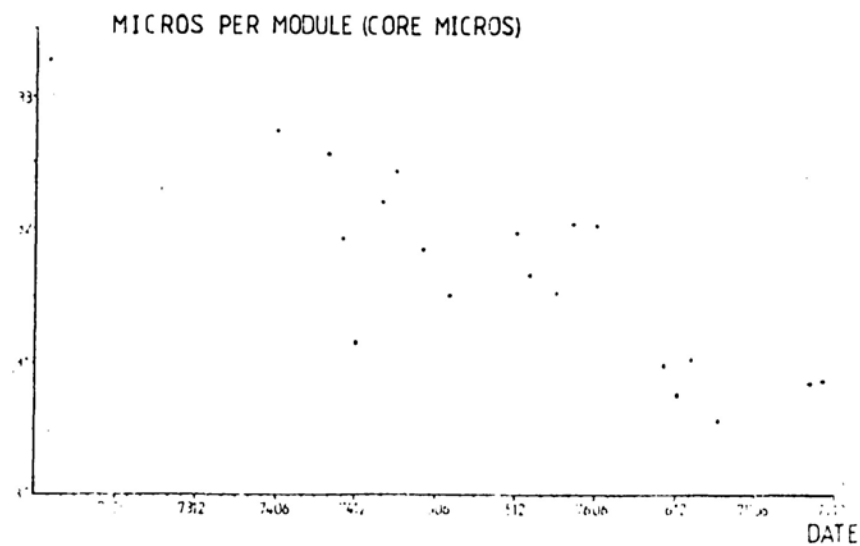
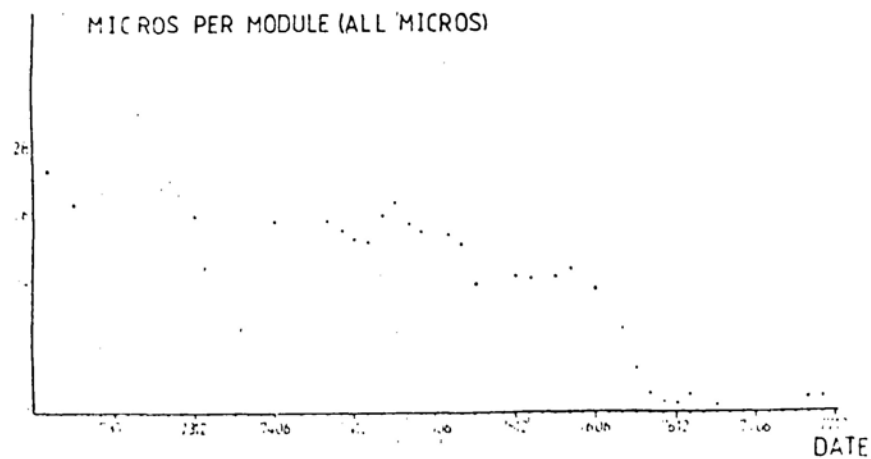
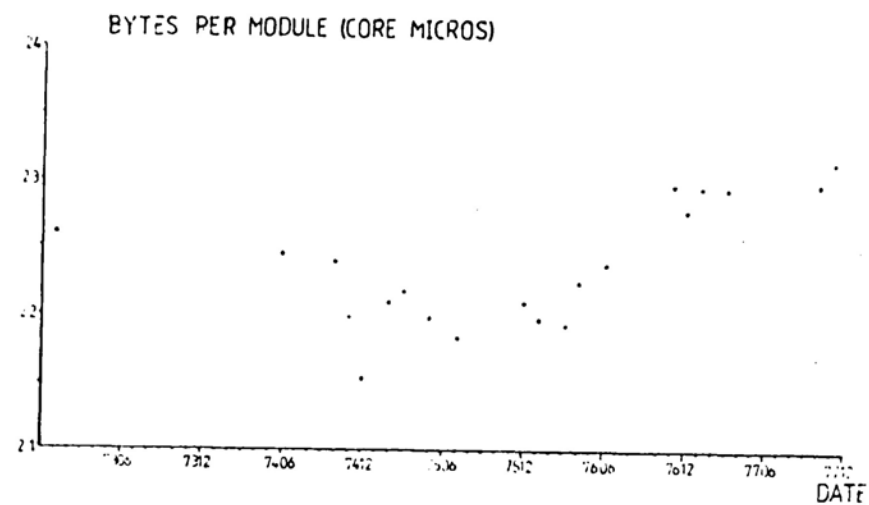
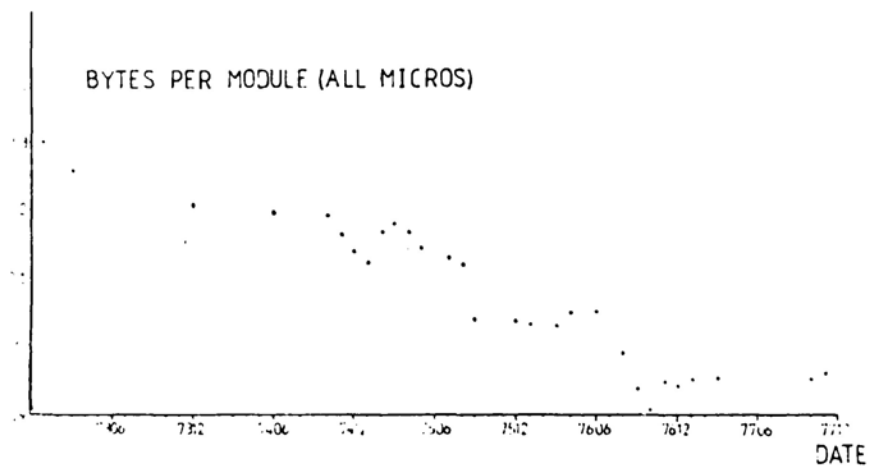
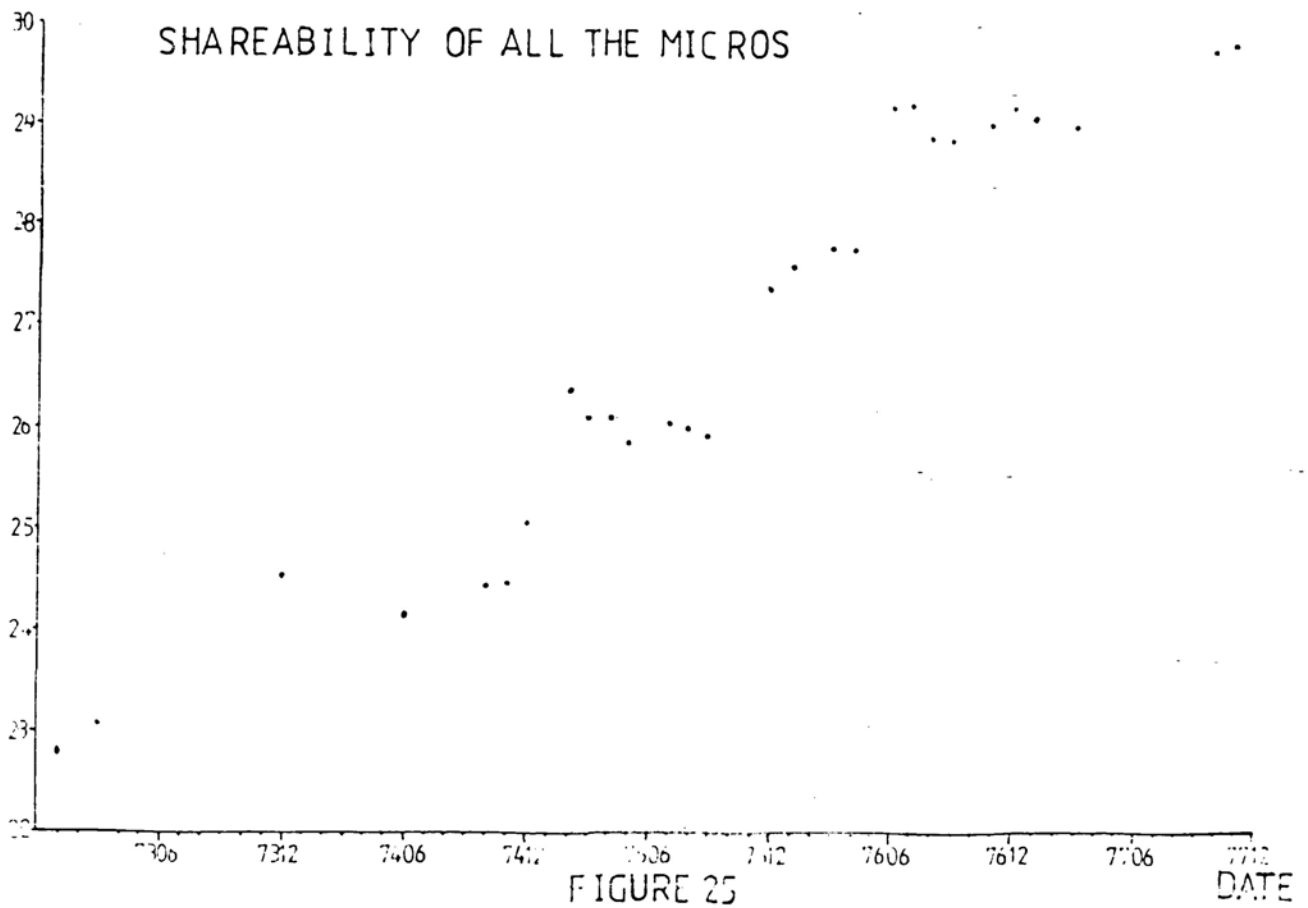
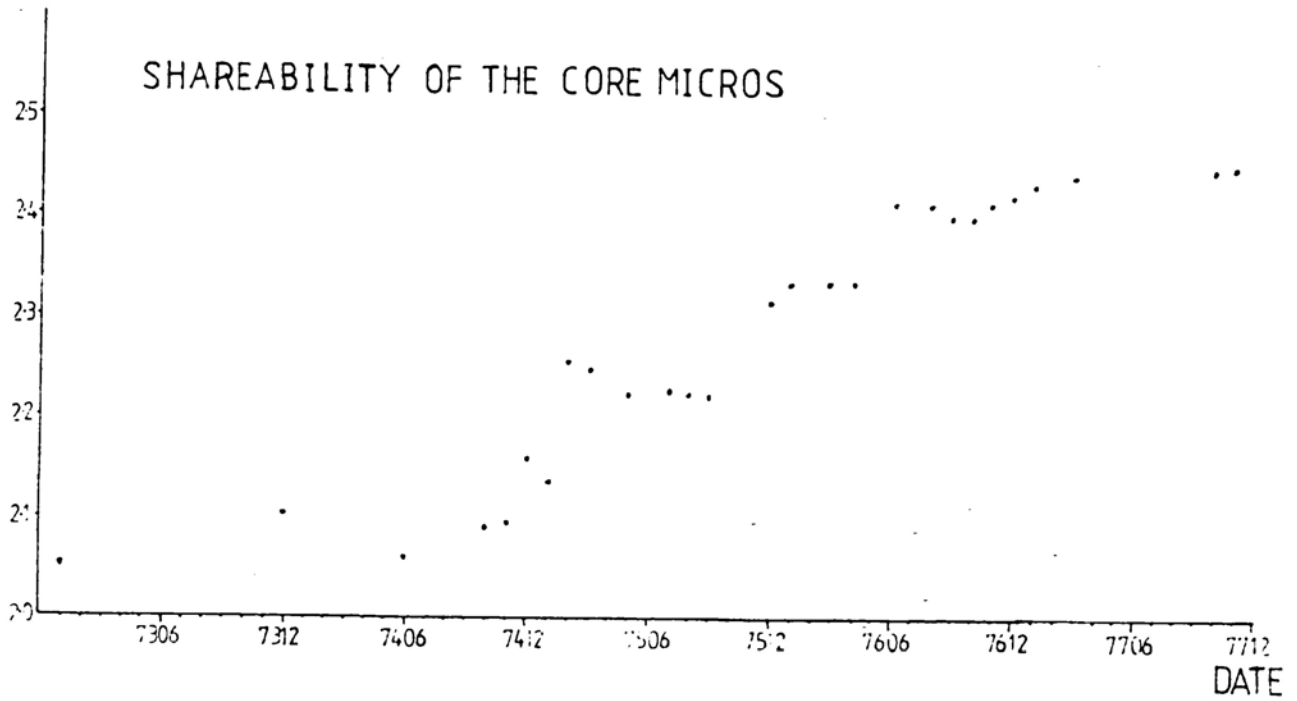


FIGURE 24



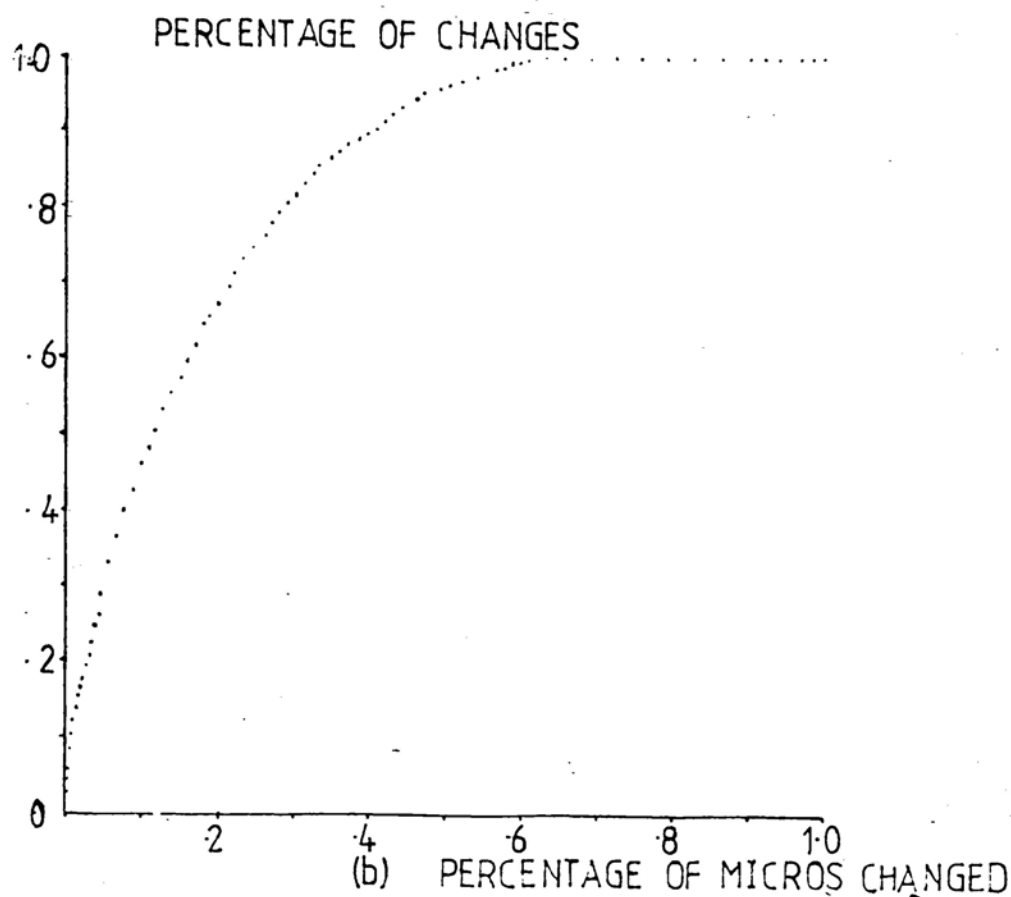
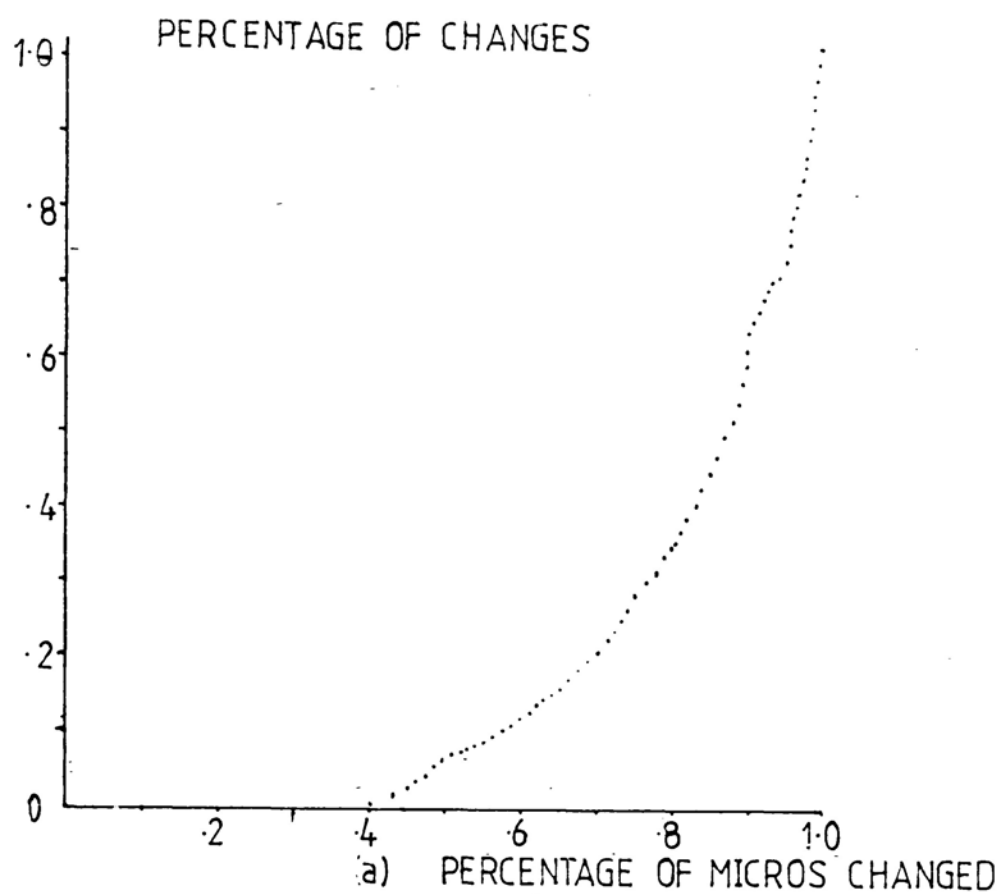


FIGURE 26

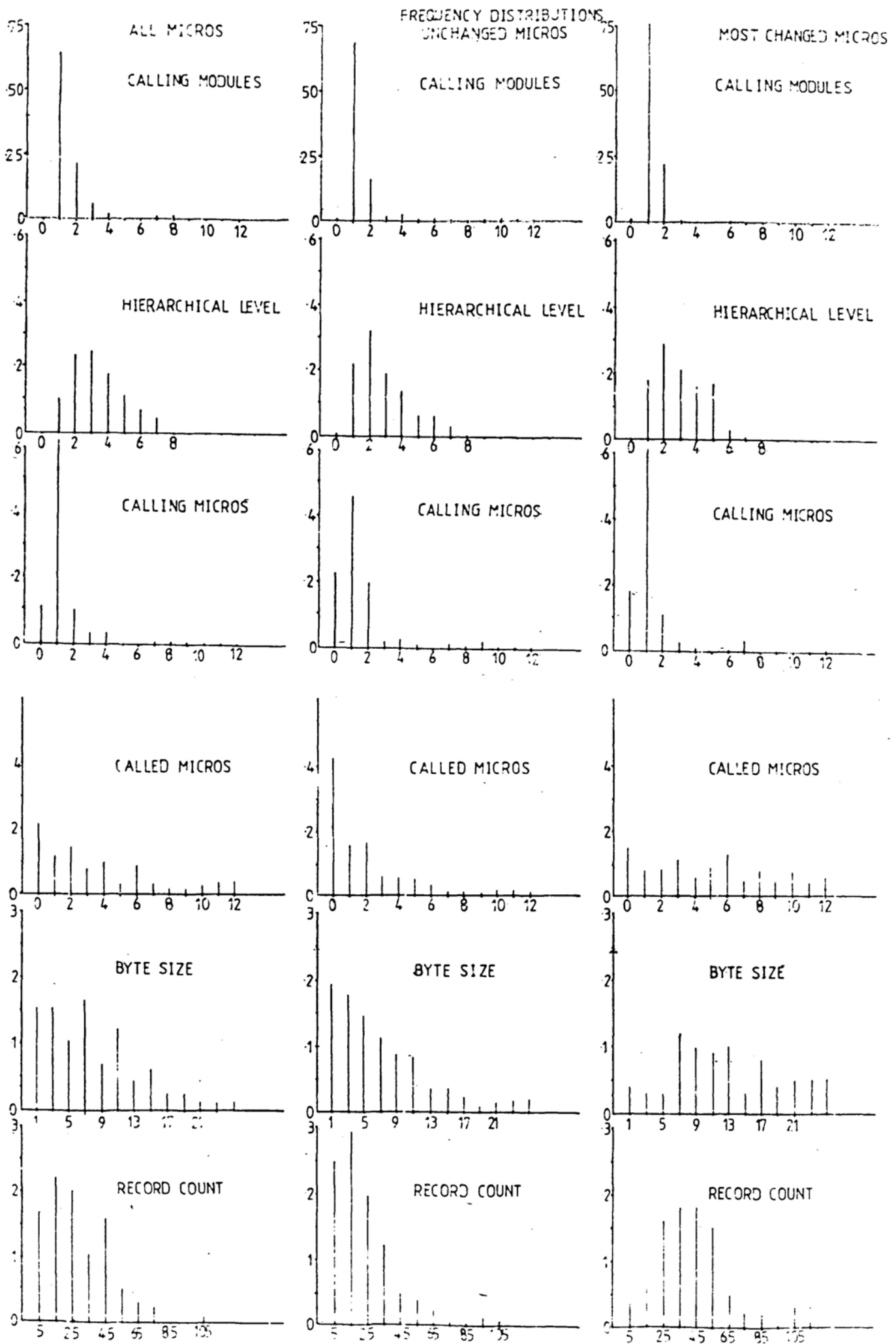
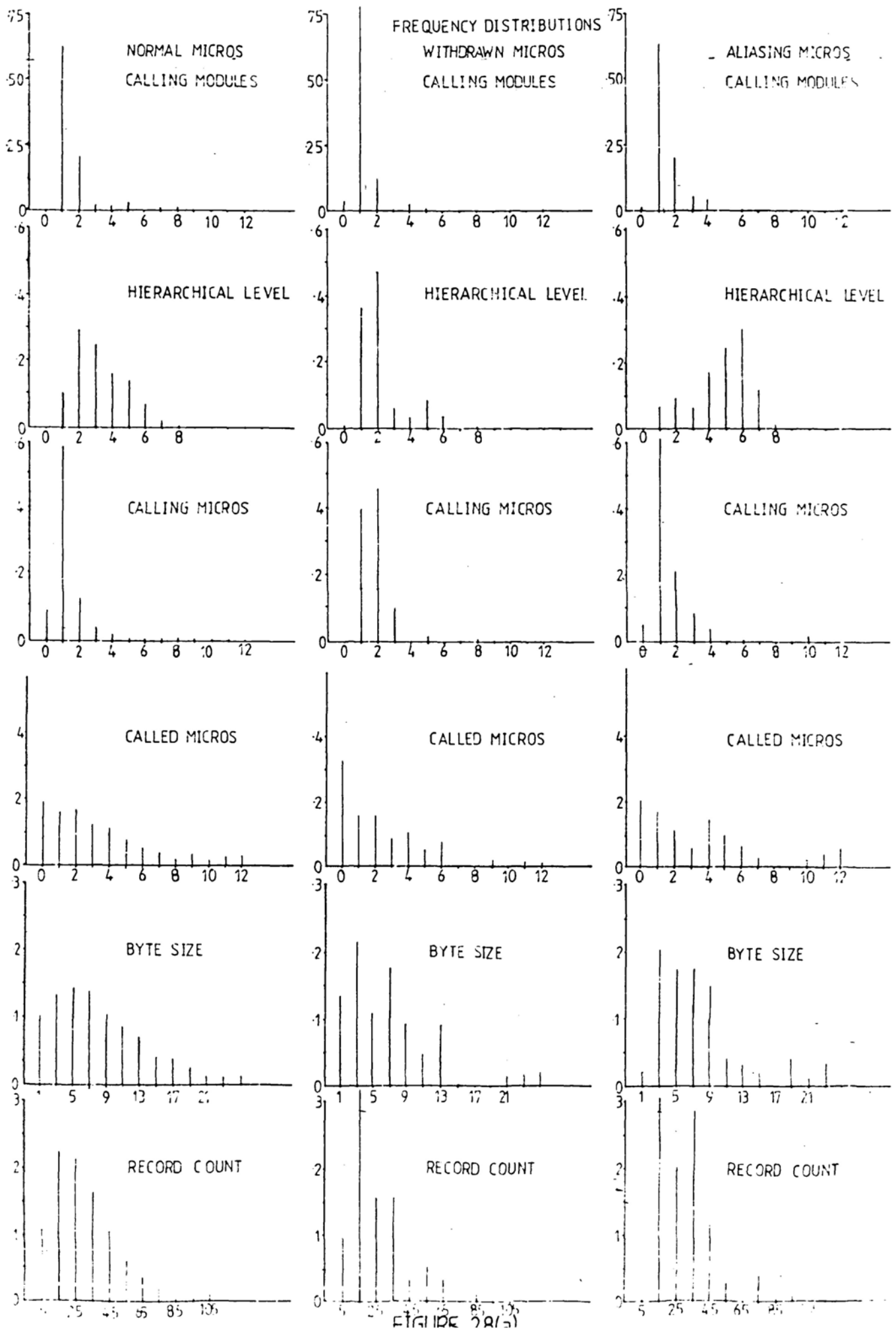
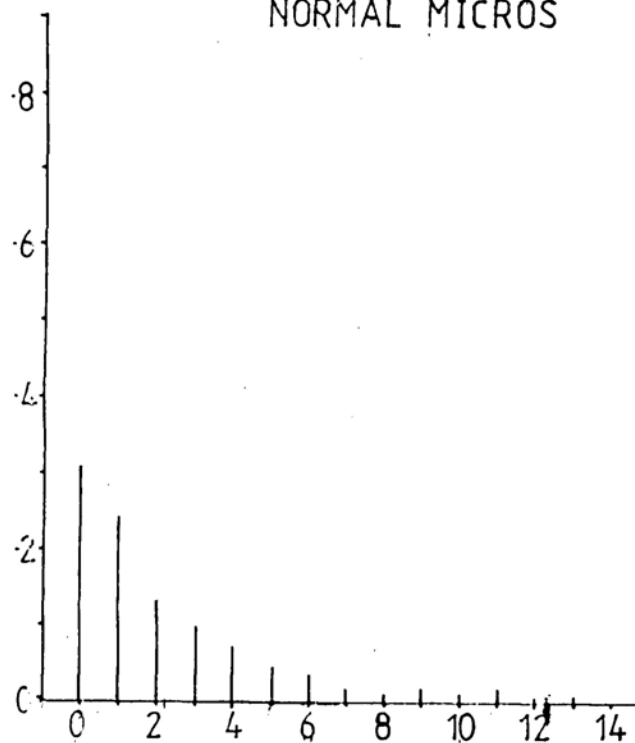


FIGURE 27

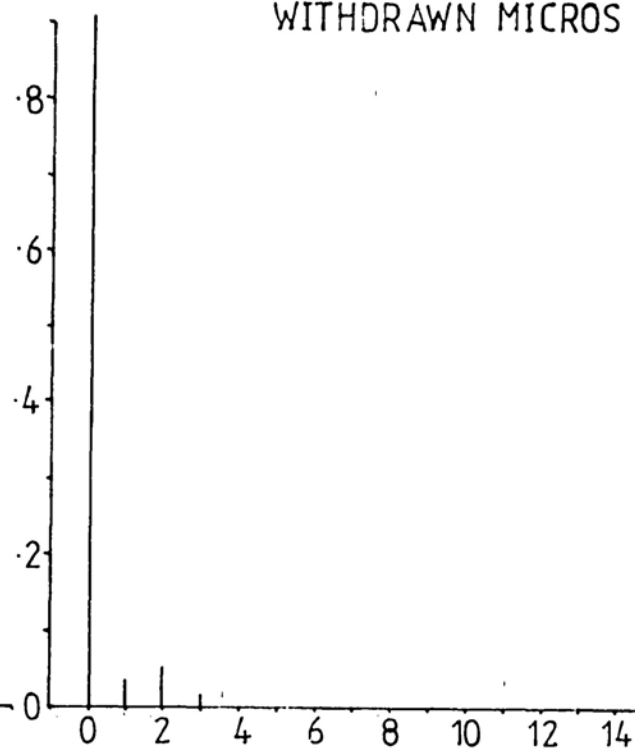


FREQUENCY DISTRIBUTIONS BY NUMBER OF CHANGES

NORMAL MICROS



WITHDRAWN MICROS



ALIASING MICROS

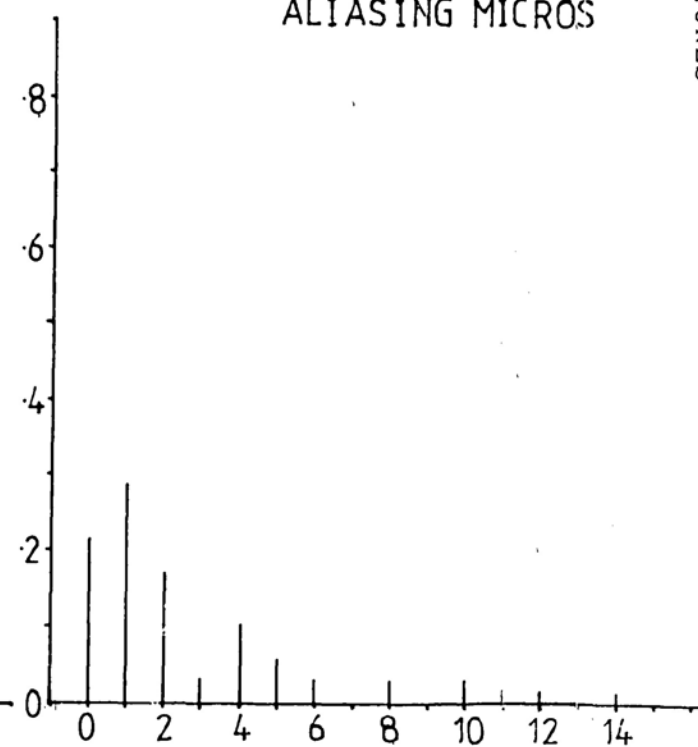


FIGURE 28(b)

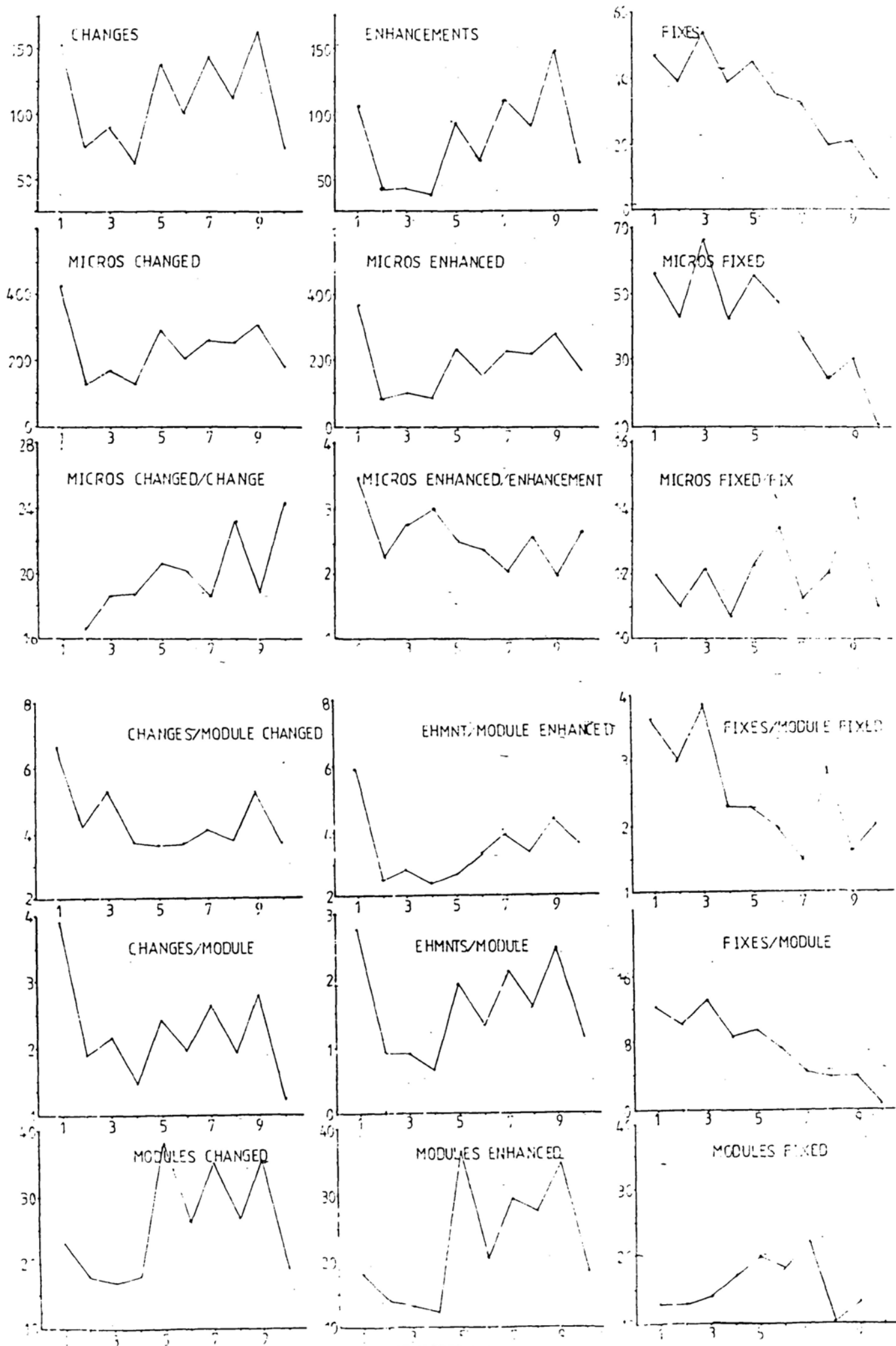


FIGURE 29

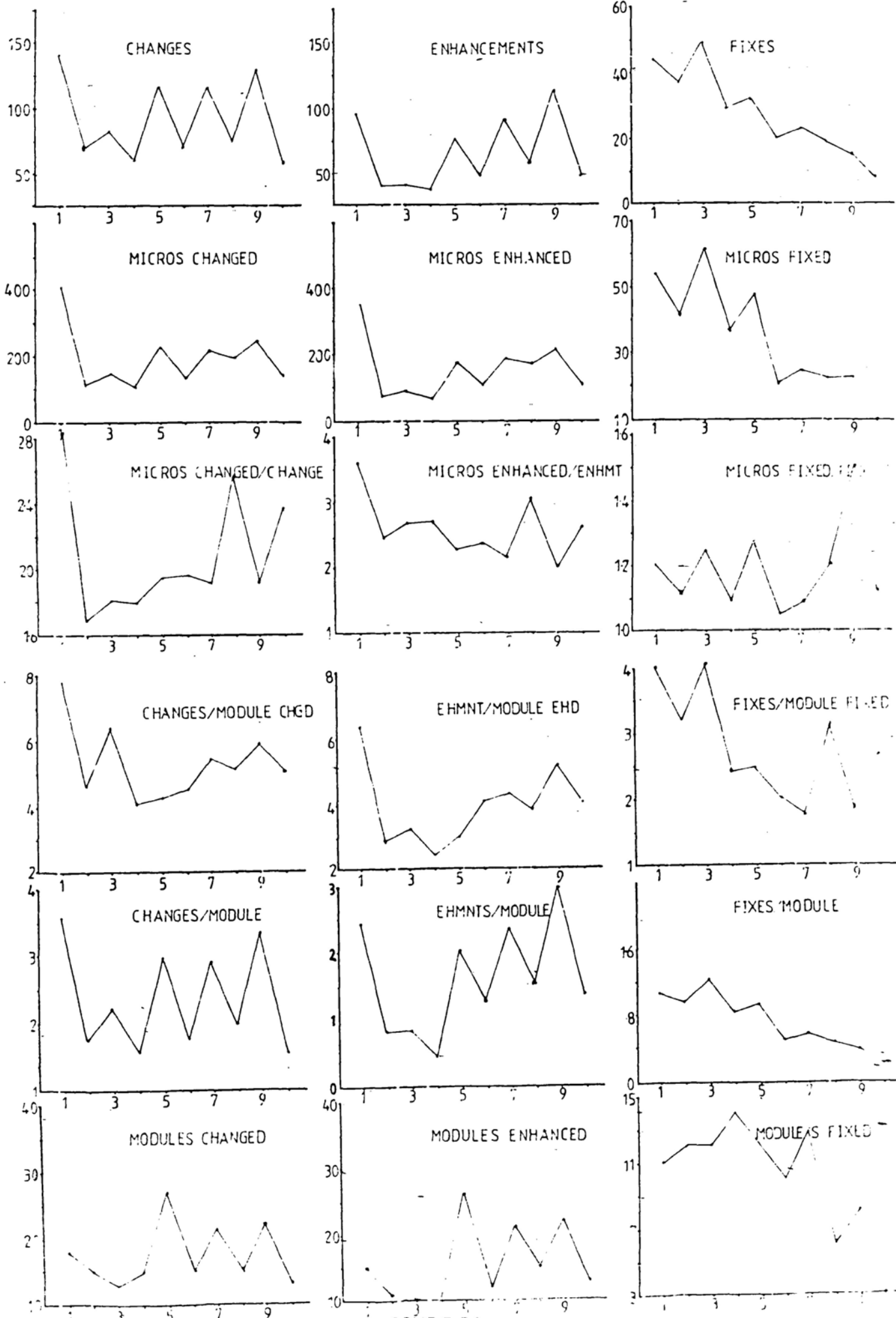
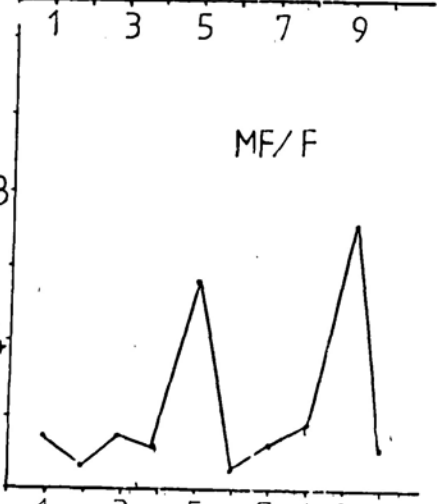
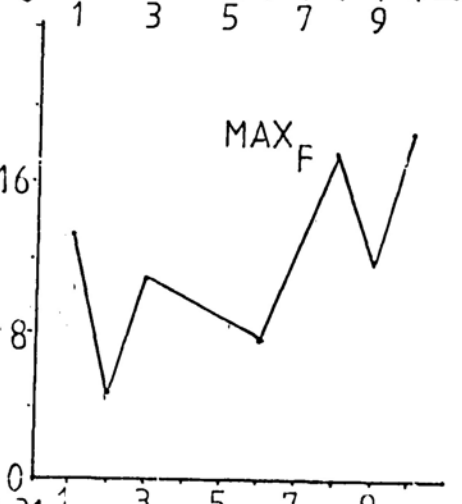
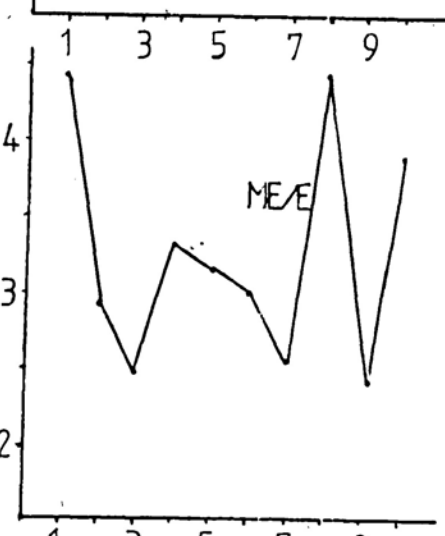
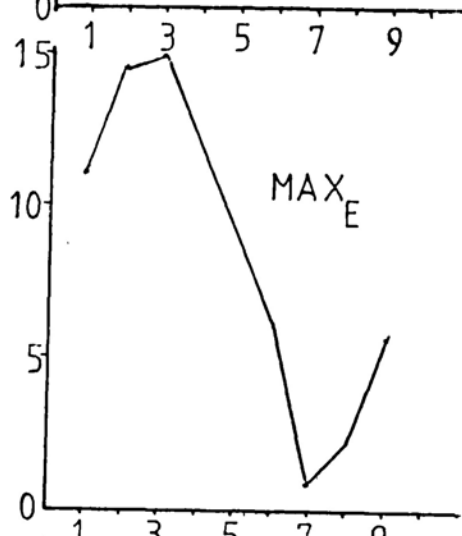
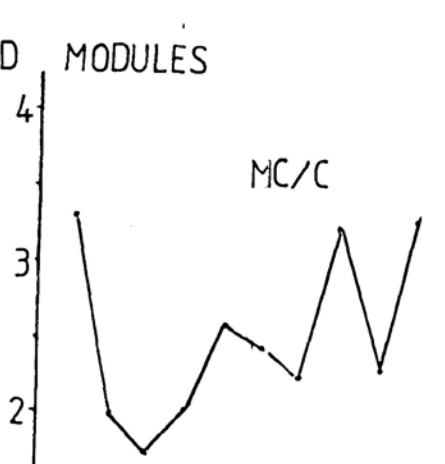
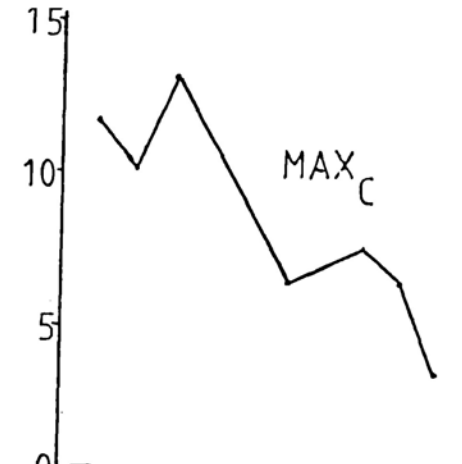
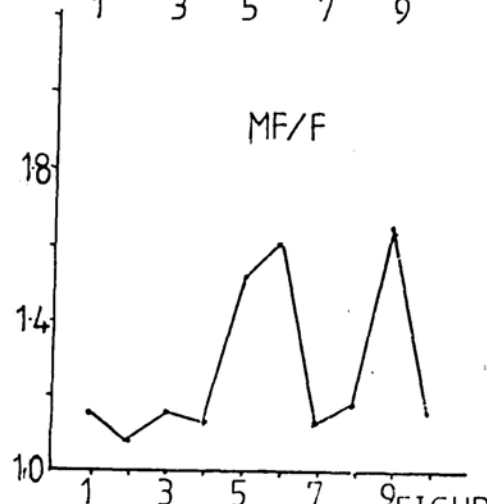
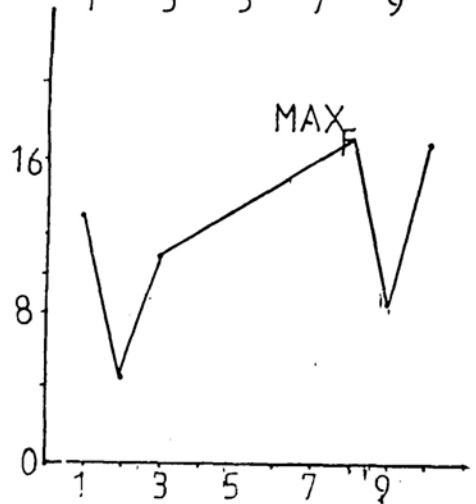
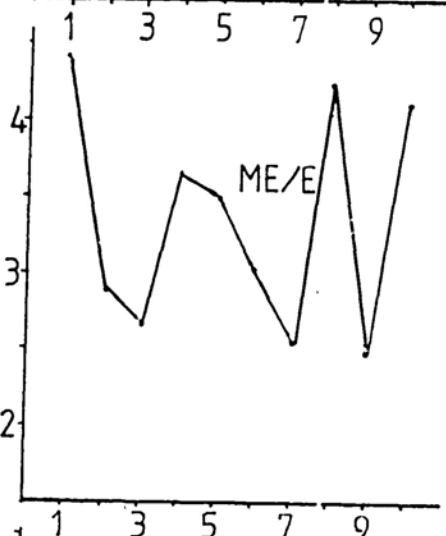
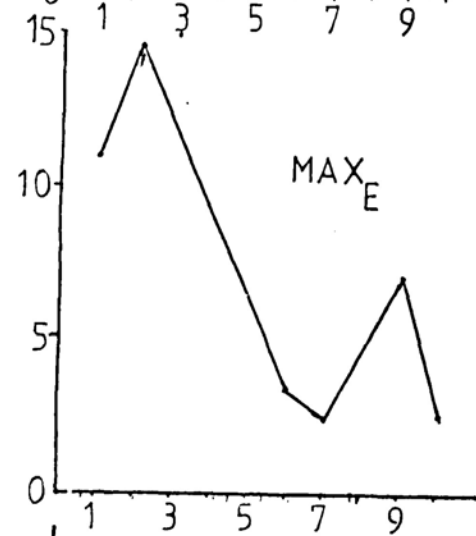
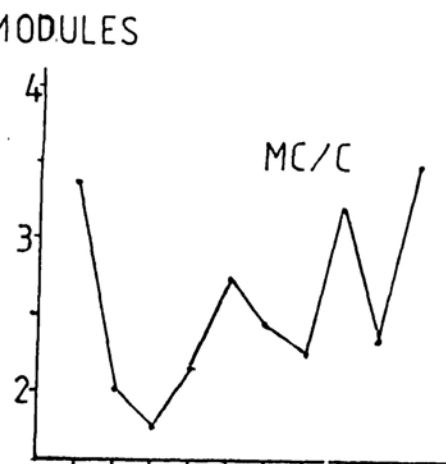
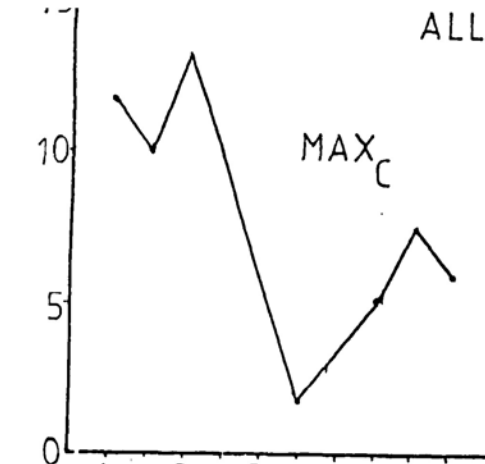


FIGURE 30



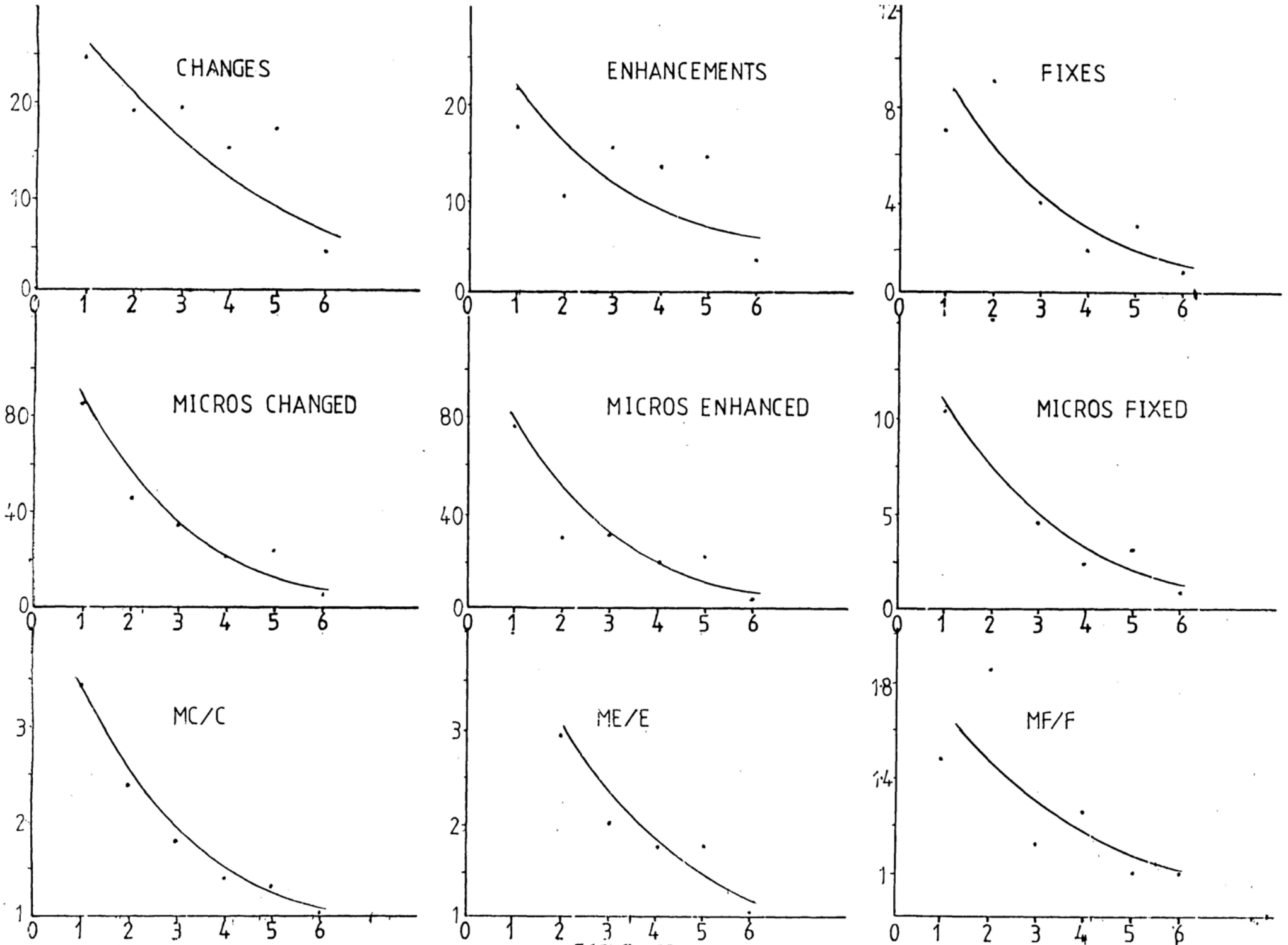
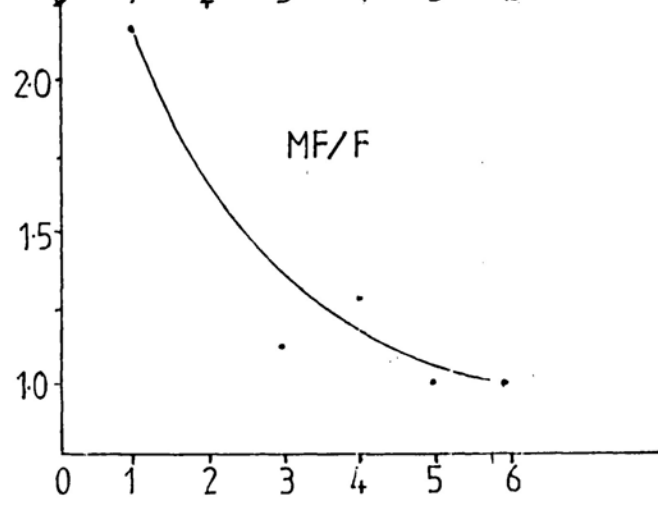
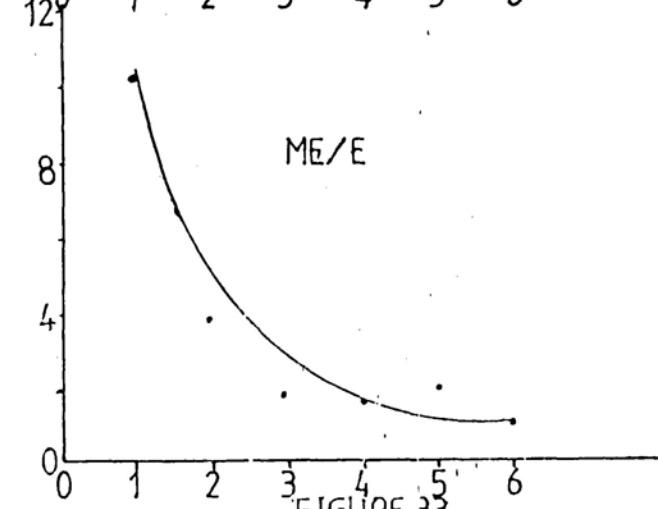
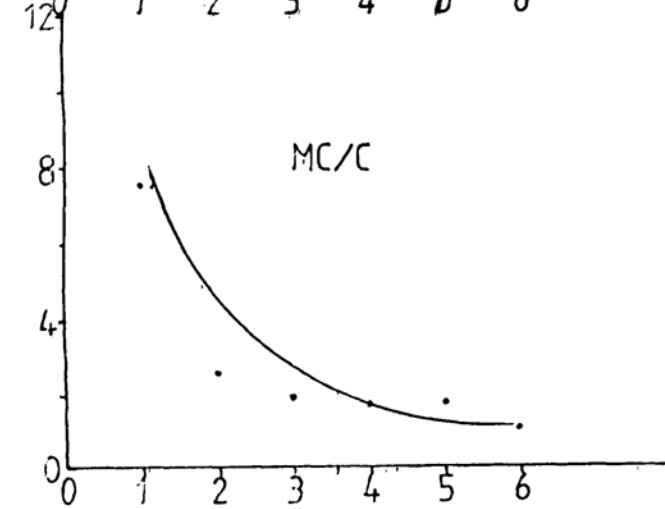
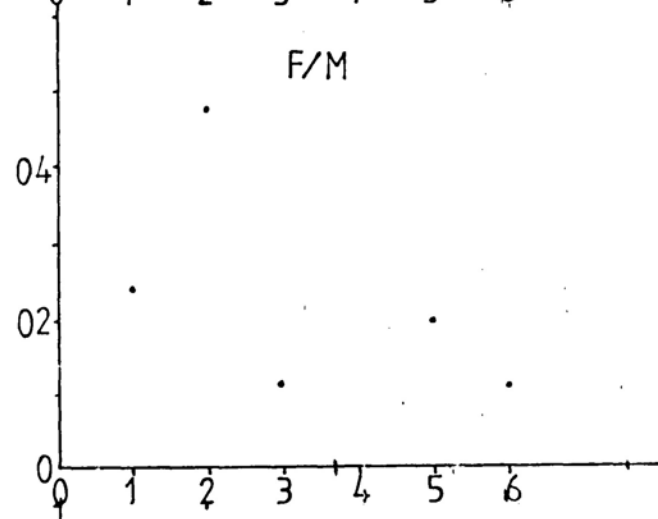
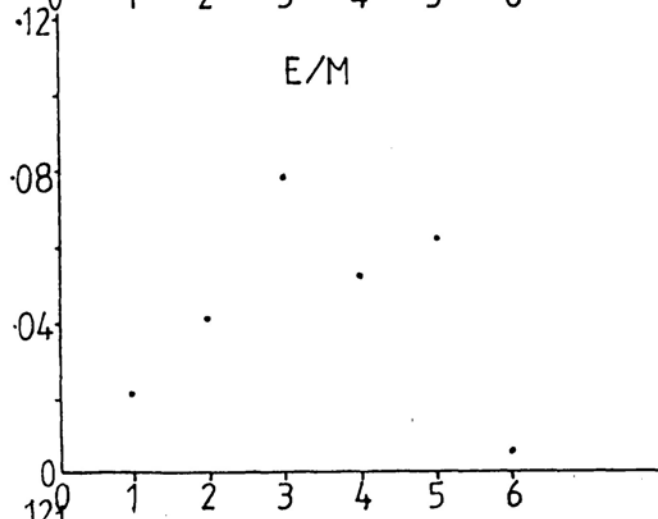
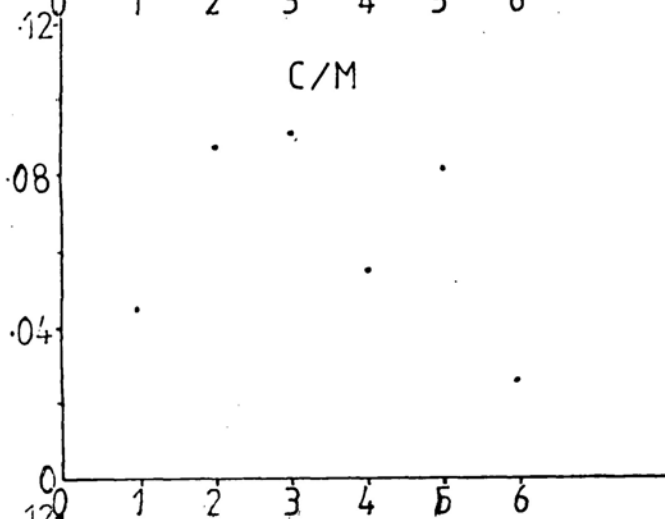
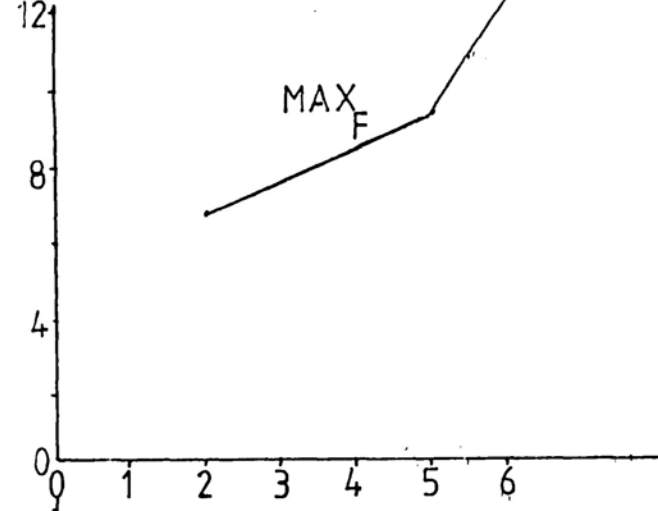
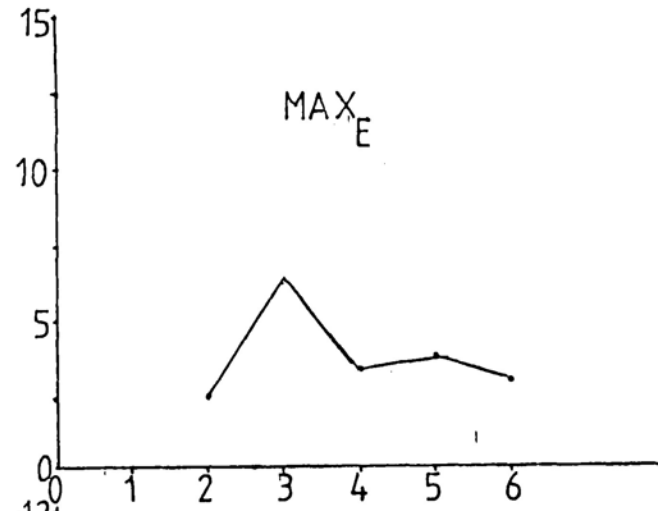
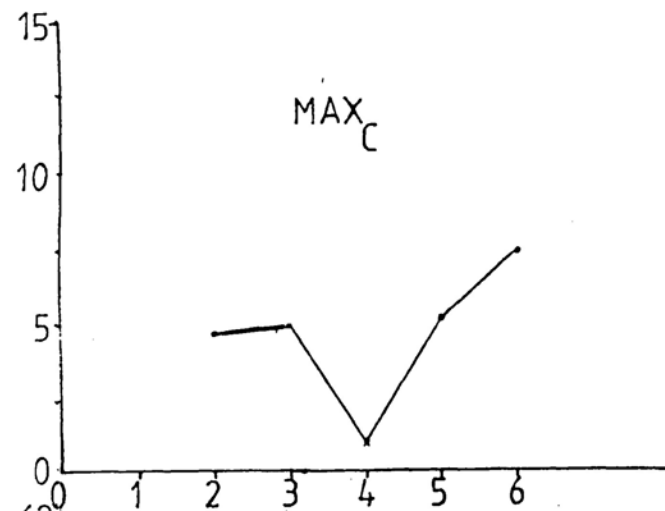
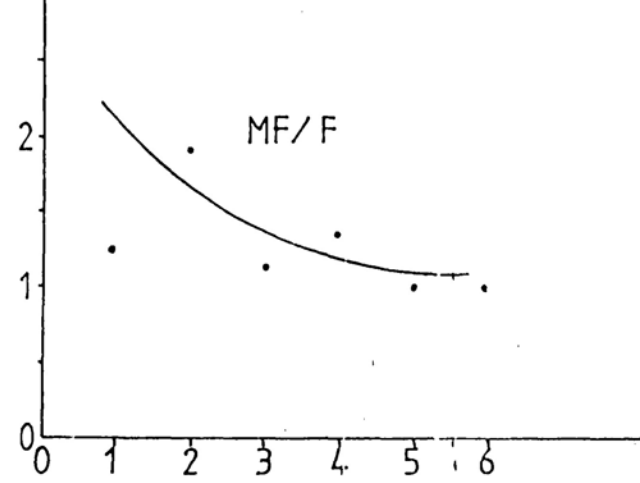
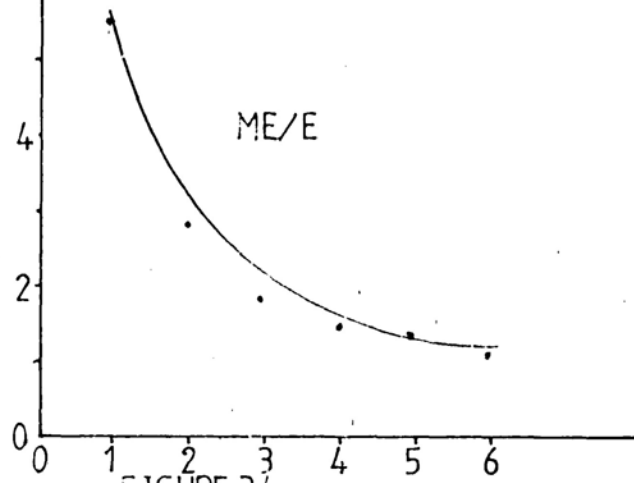
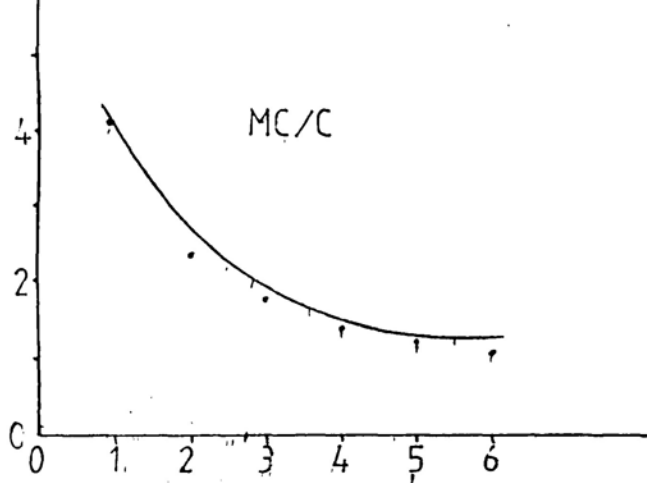
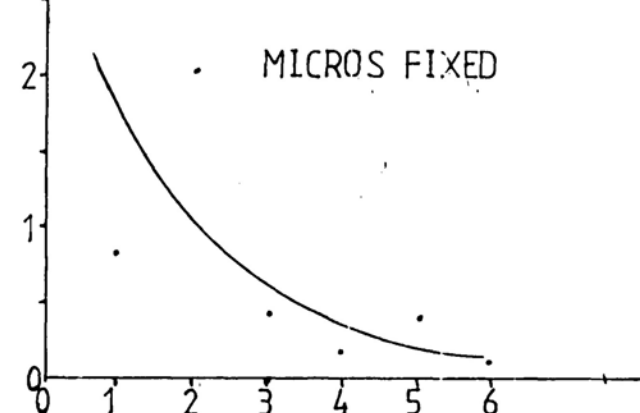
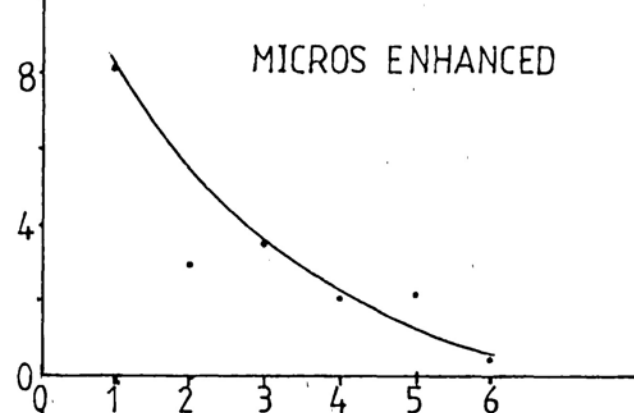
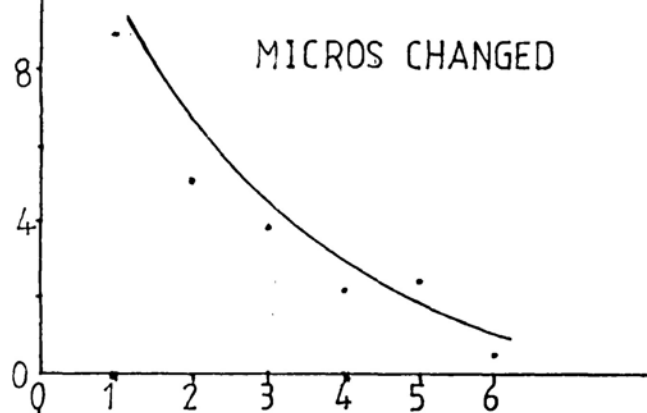
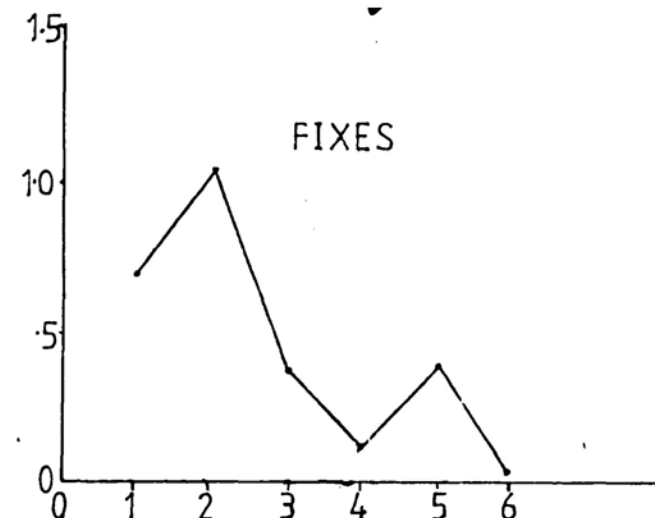
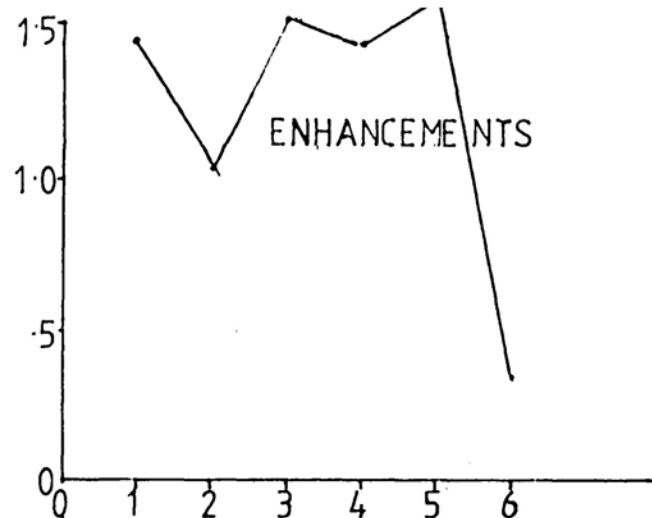
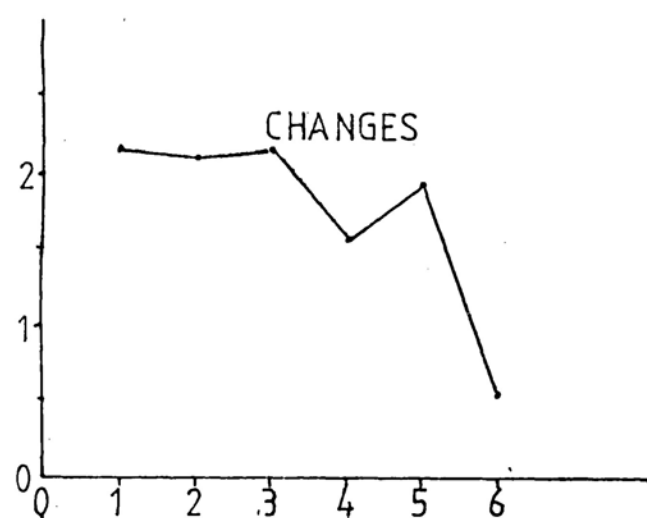
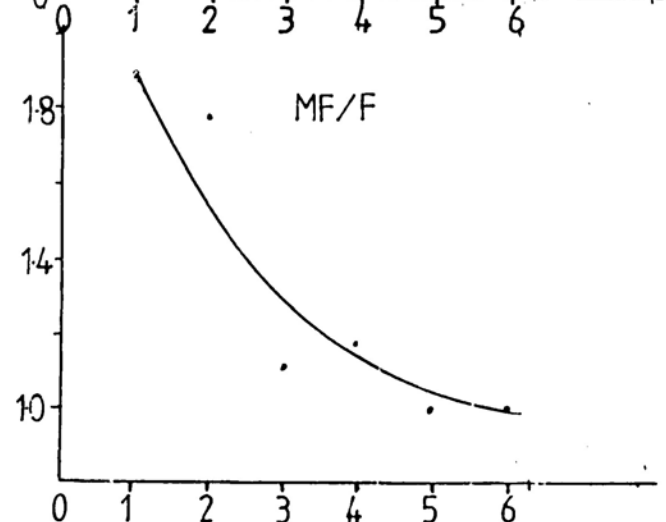
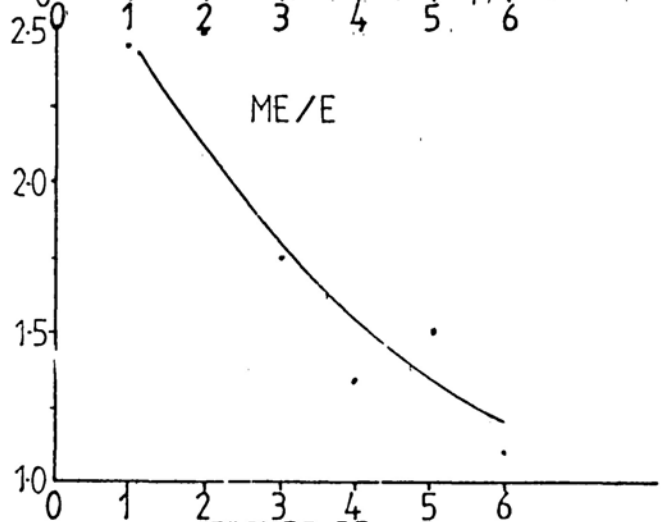
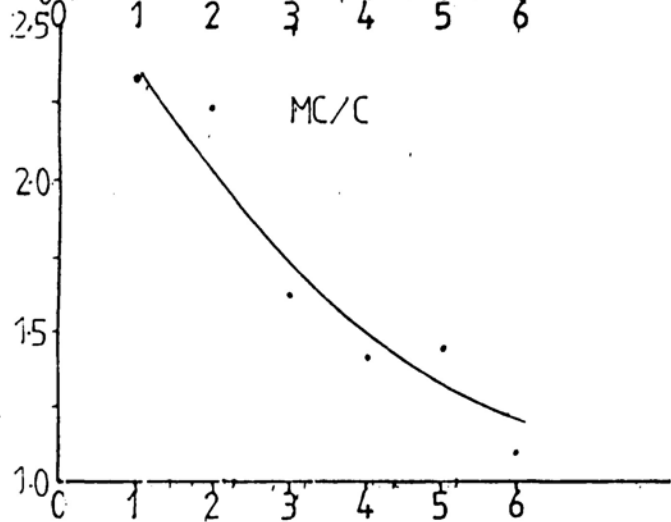
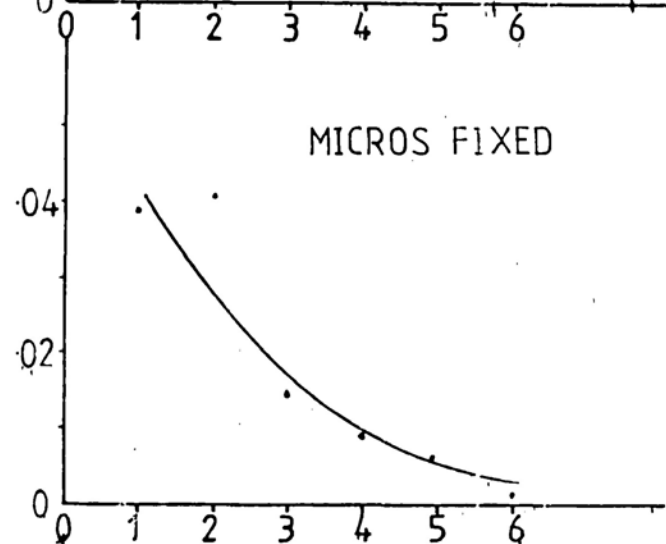
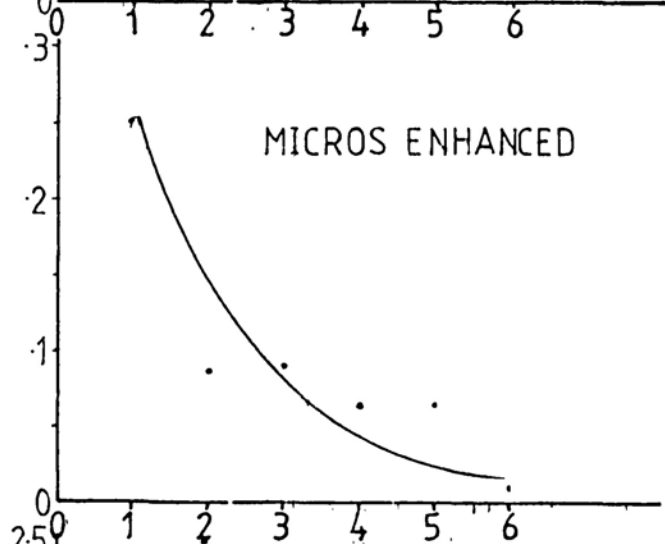
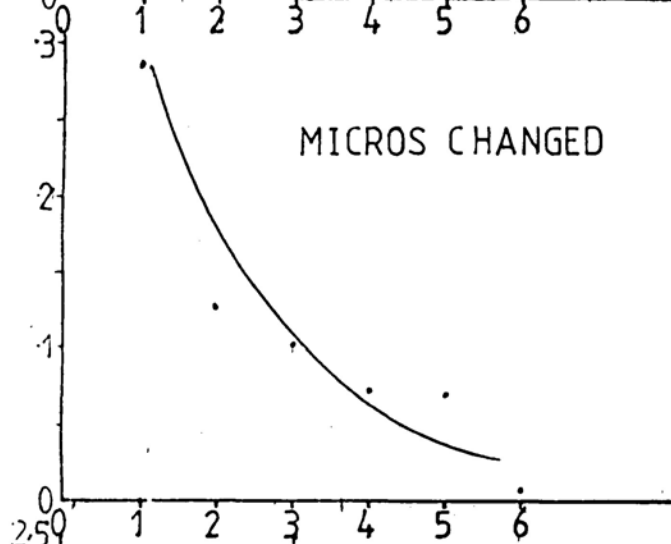
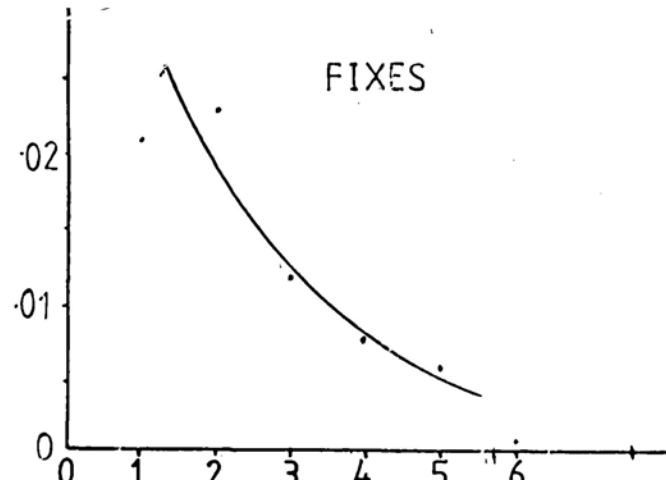
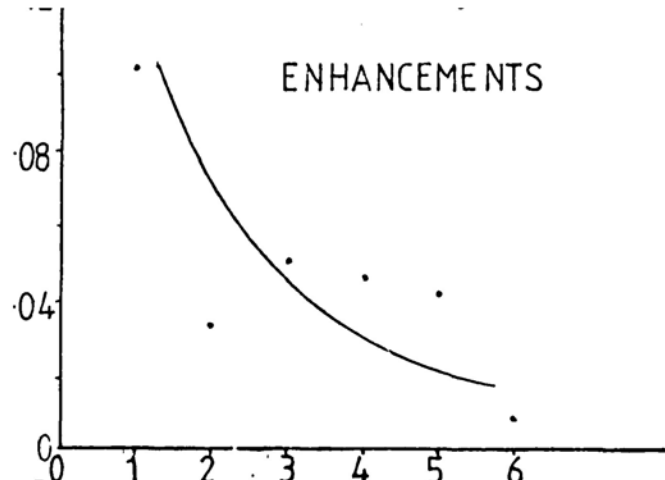
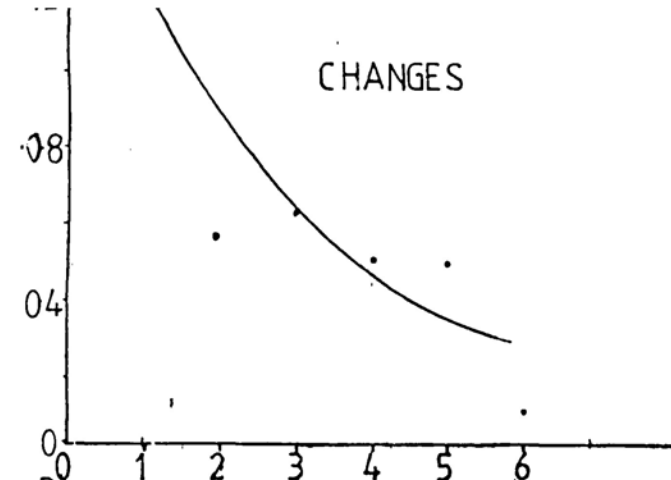


FIGURE 32



FIGURES



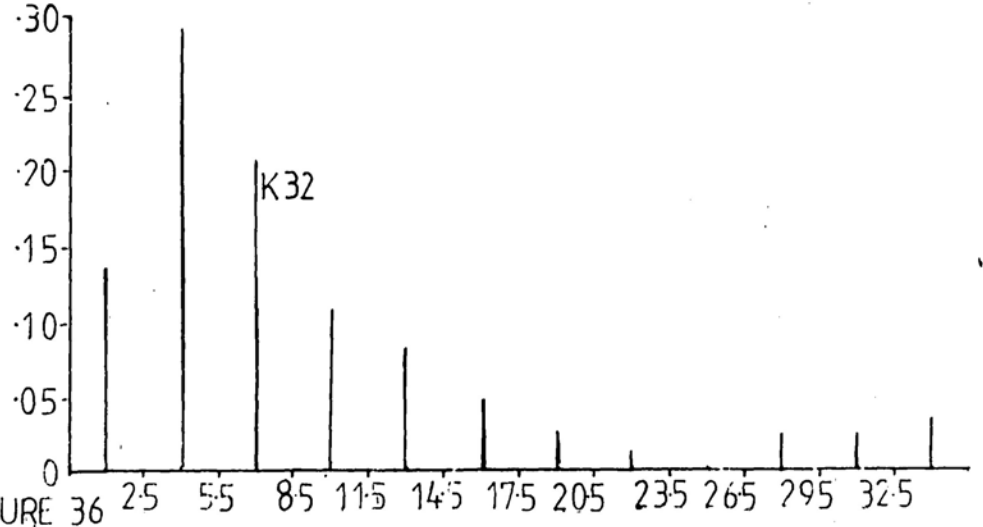
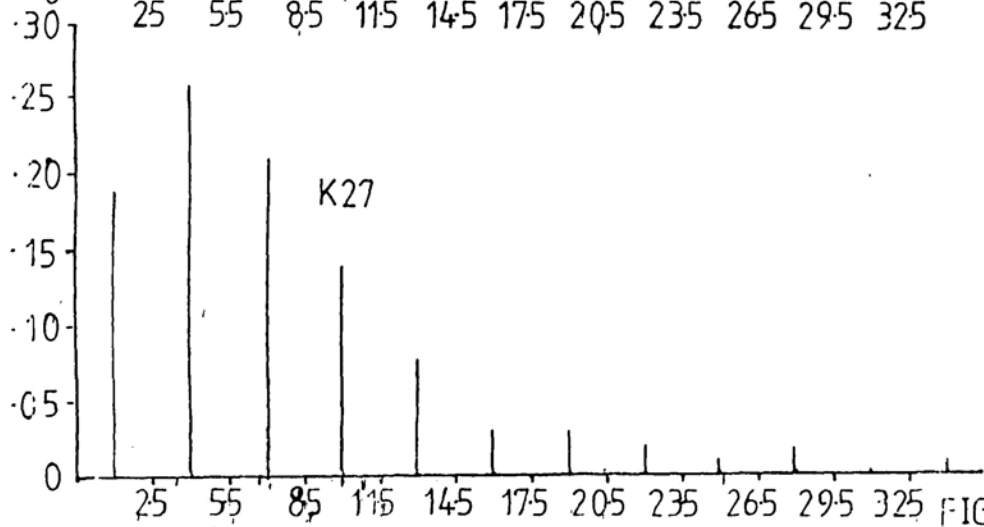
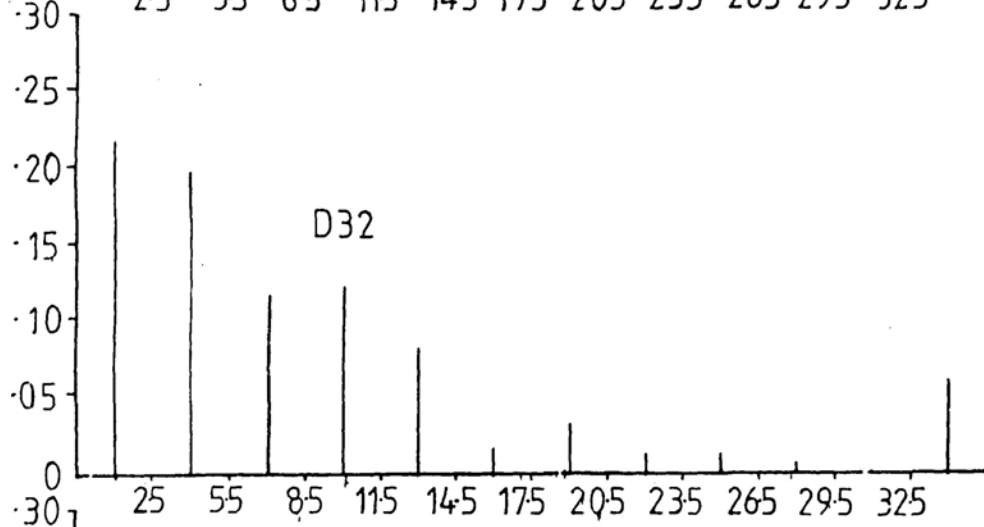
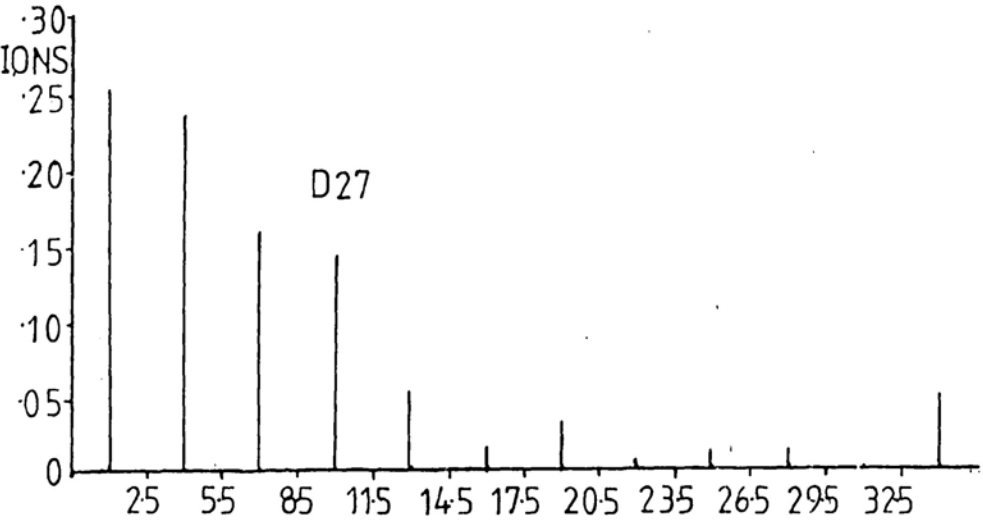
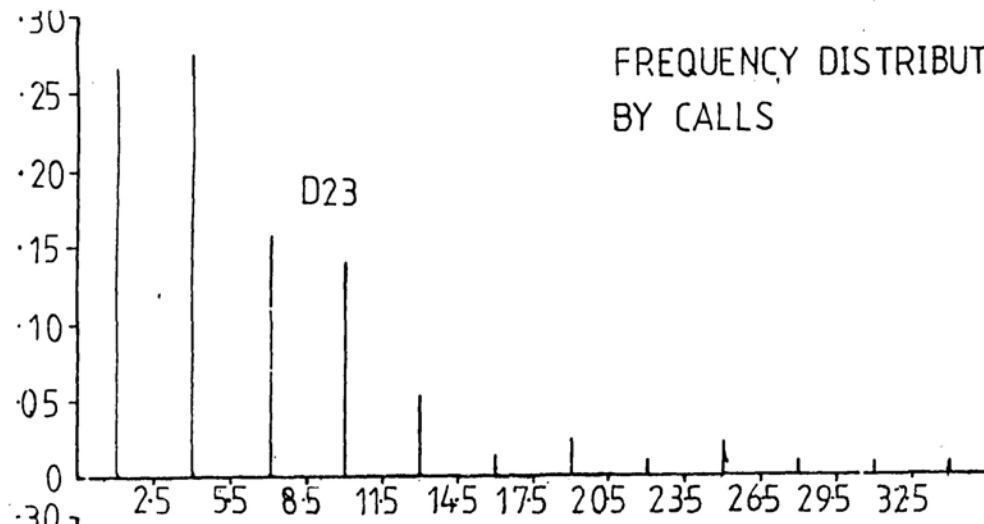


FIGURES

PAGE 263

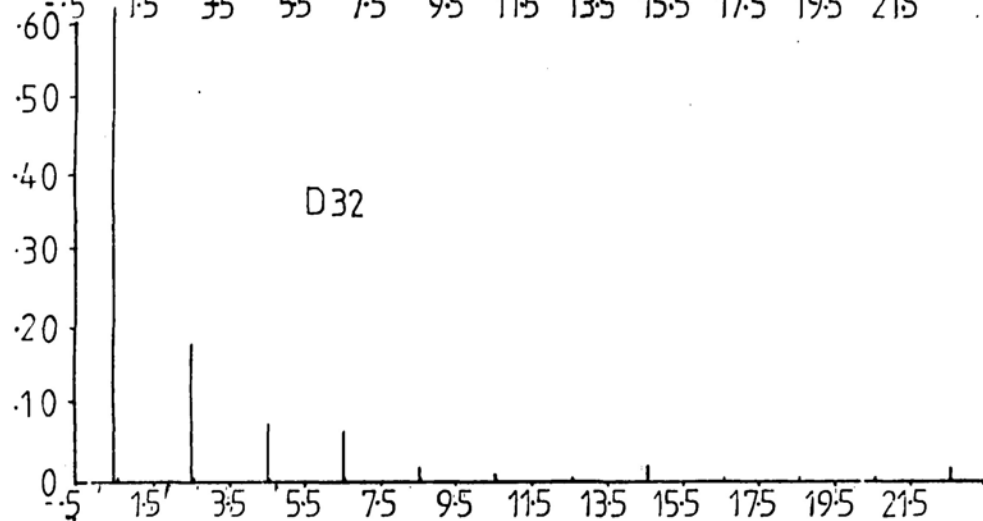
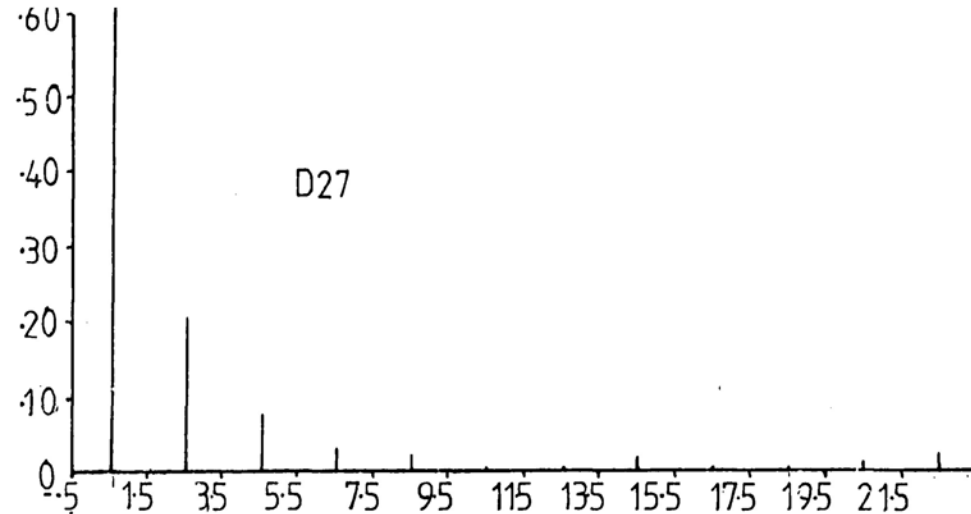
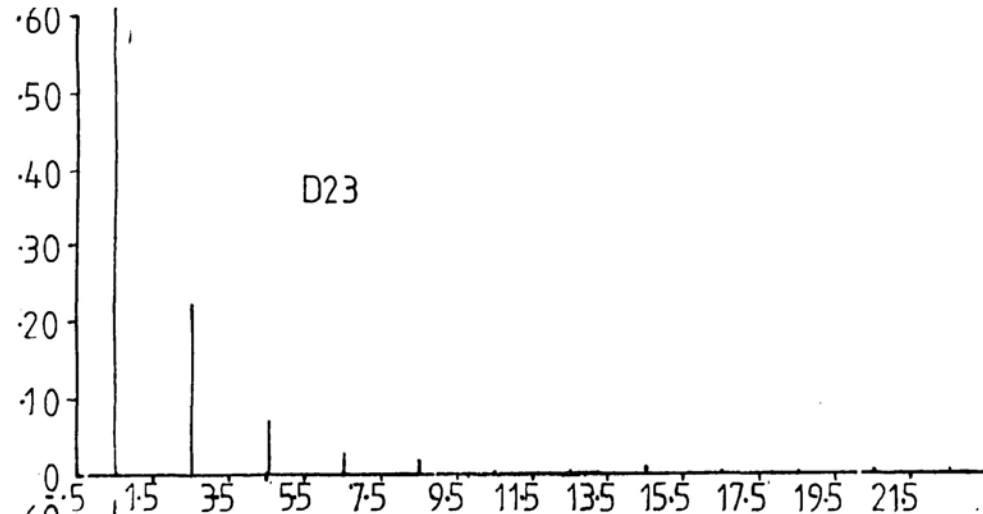
FIGURE 35

FREQUENCY DISTRIBUTIONS BY CALLS



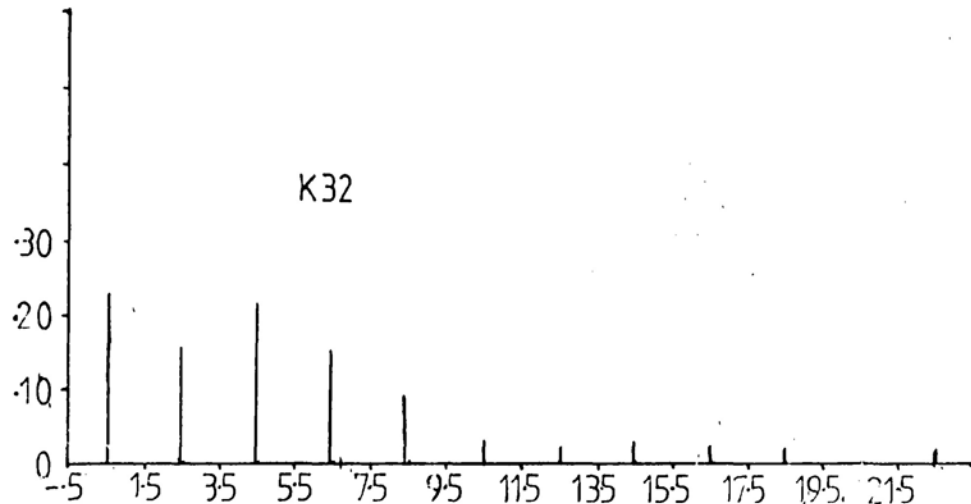
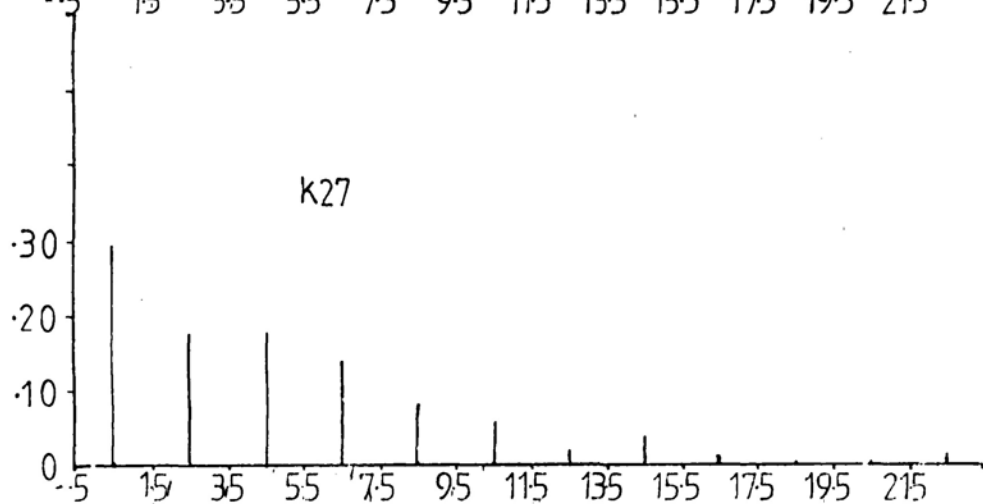
FIGURES

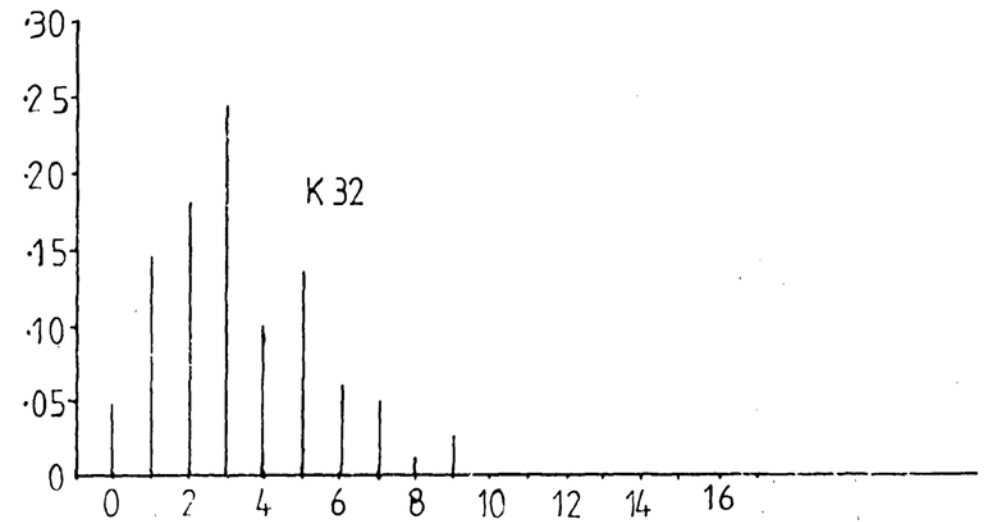
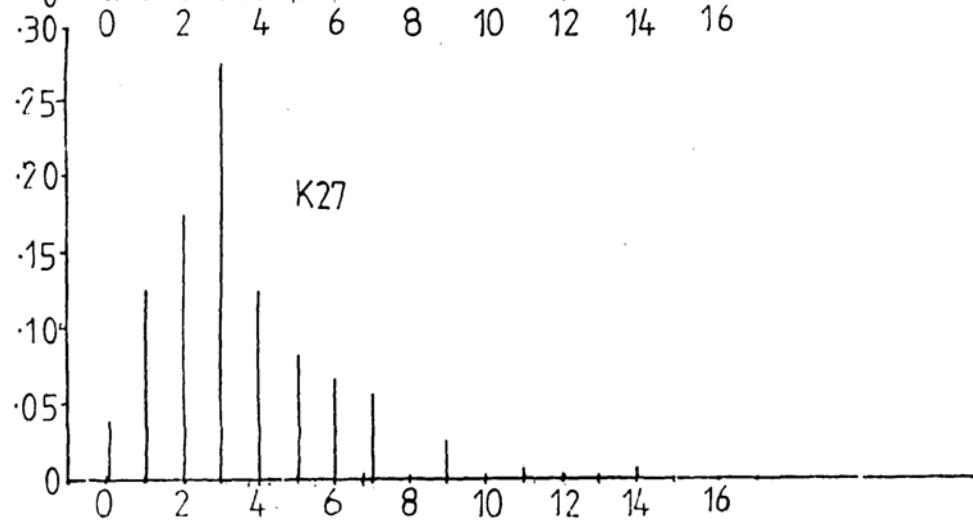
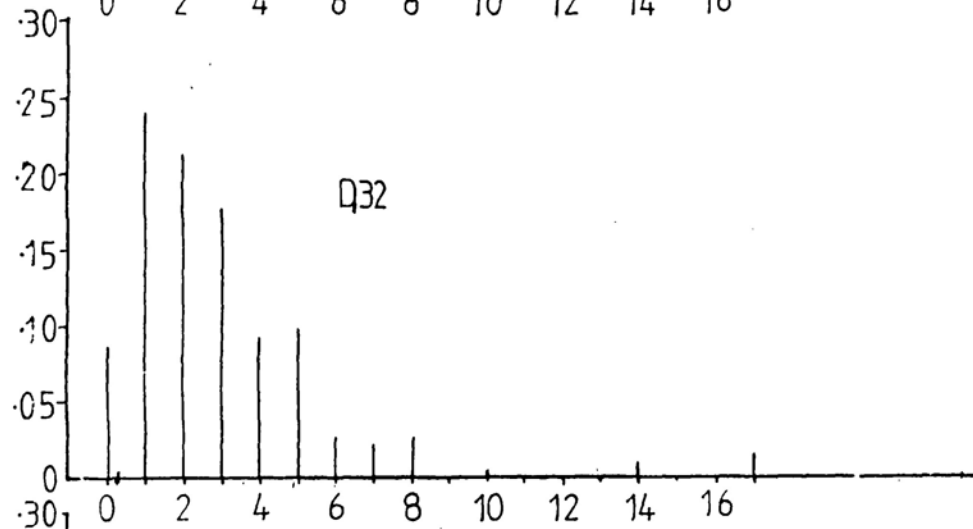
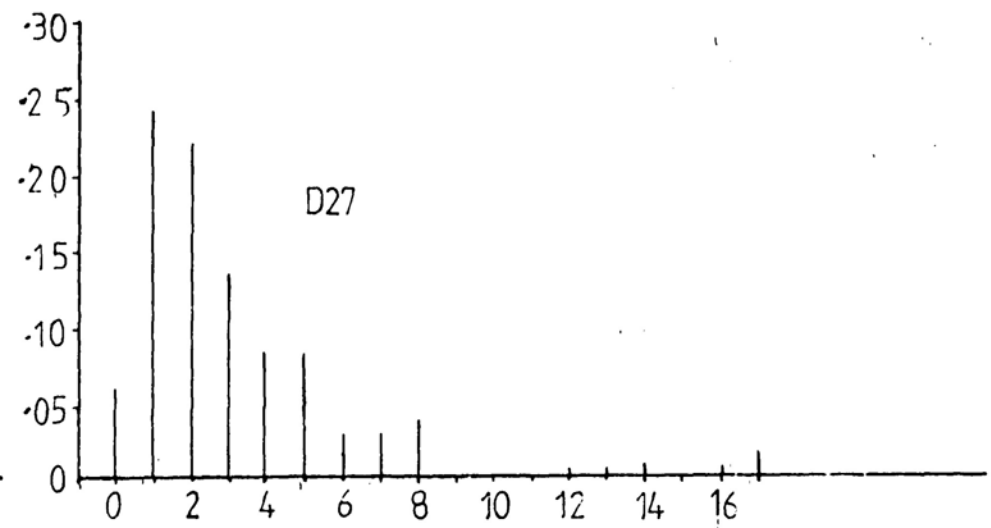
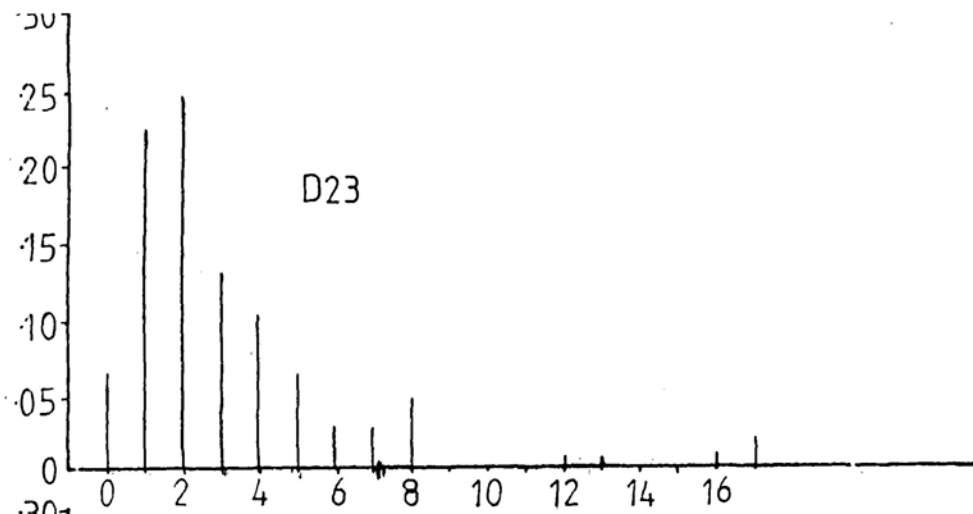
PAGE 264



FREQUENCY DISTRIBUTIONS
BY EXTS

FIGURE 37





FREQUENCY DISTRIBUTIONS
BY 10(P)

FIGURE 38

FREQUENCY DISTRIBUTIONS OF RELEASE 5X32 SUBSYSTEMS BY EXTS

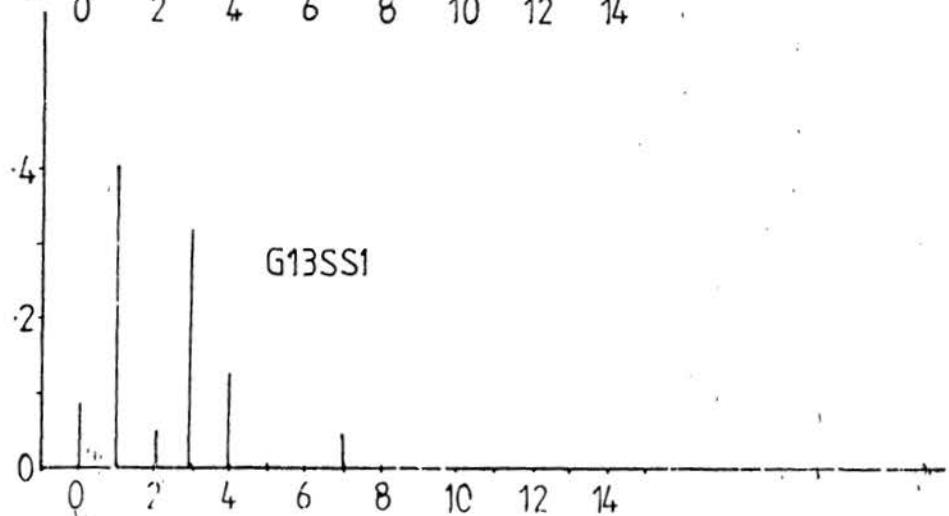
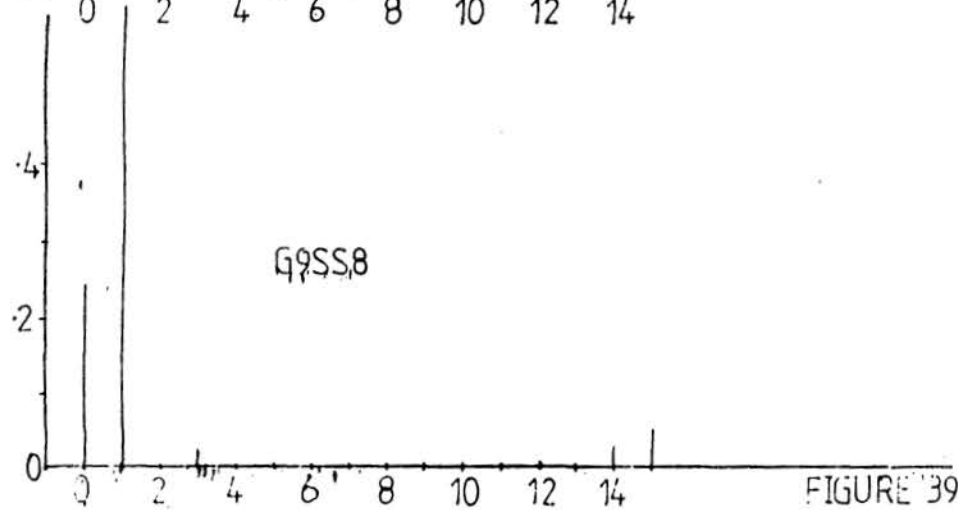
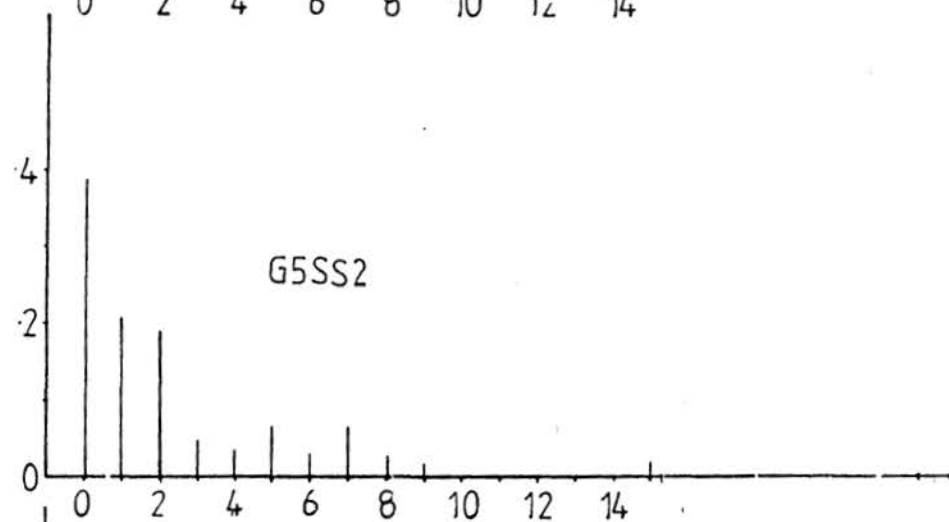
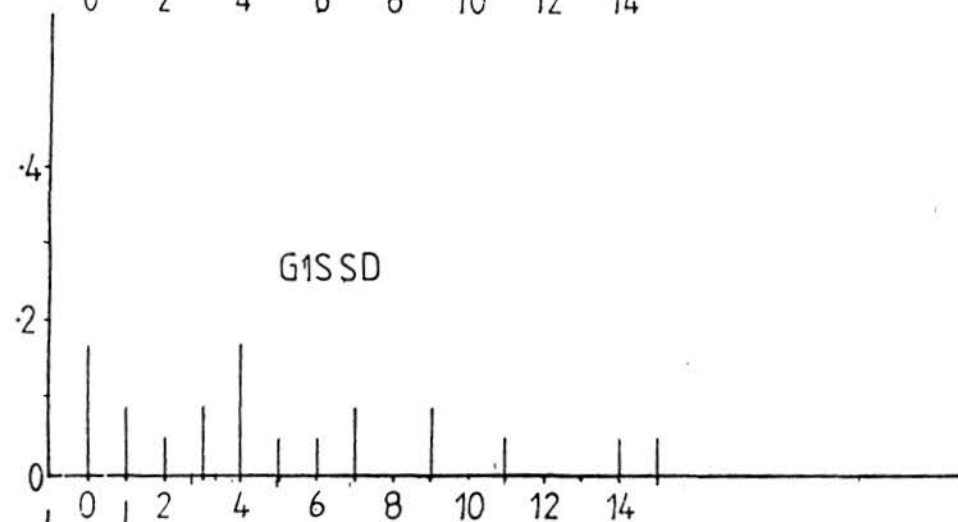
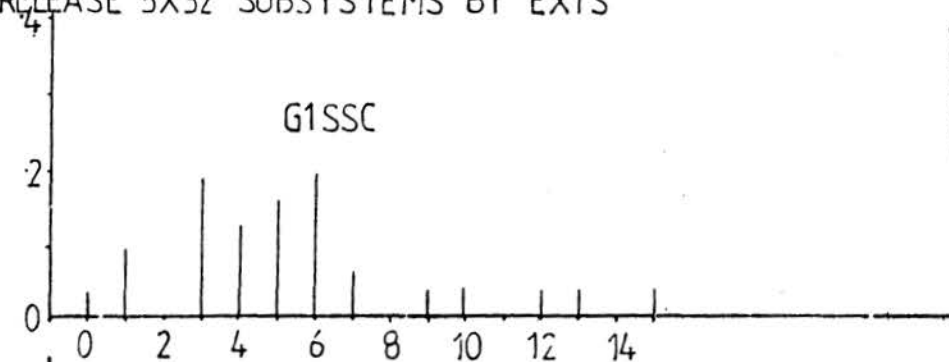
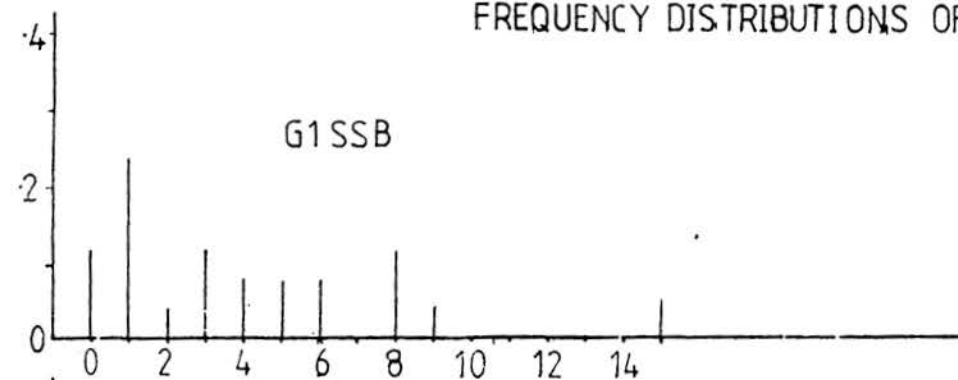


FIGURE 39

SUBSYSTEMS DAF2 DISTRIBUTIONS

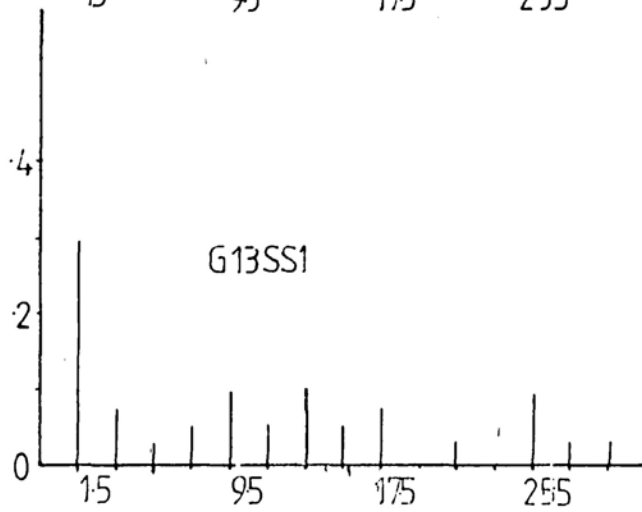
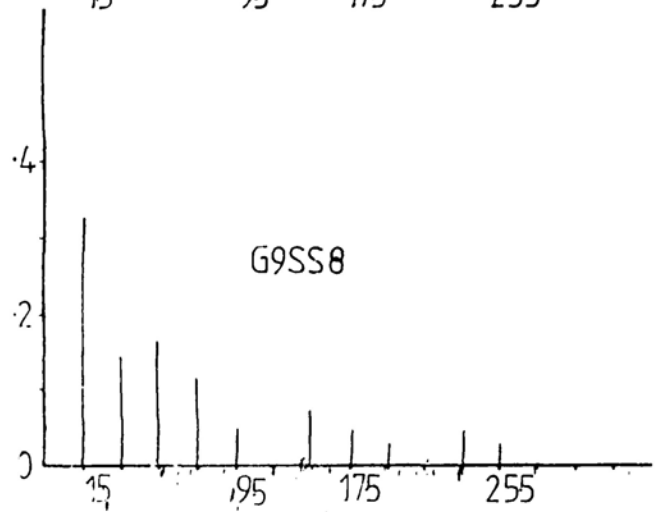
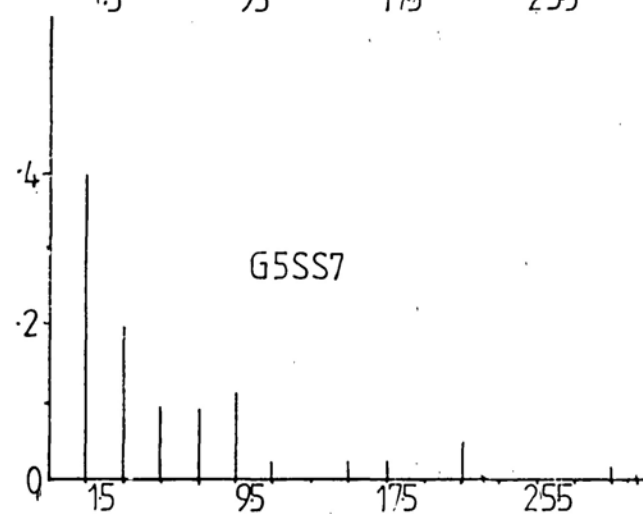
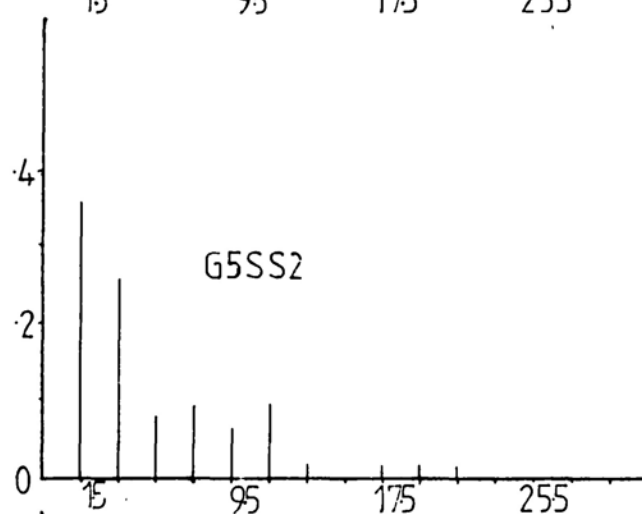
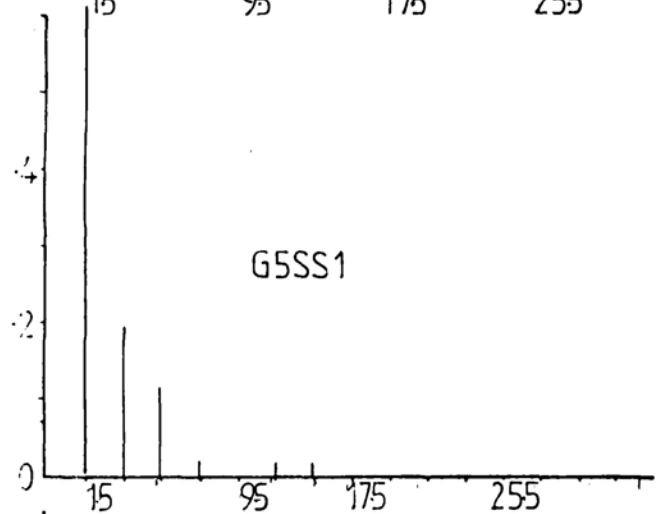
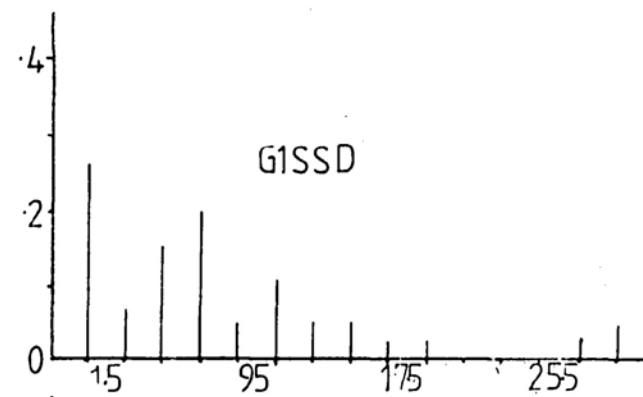
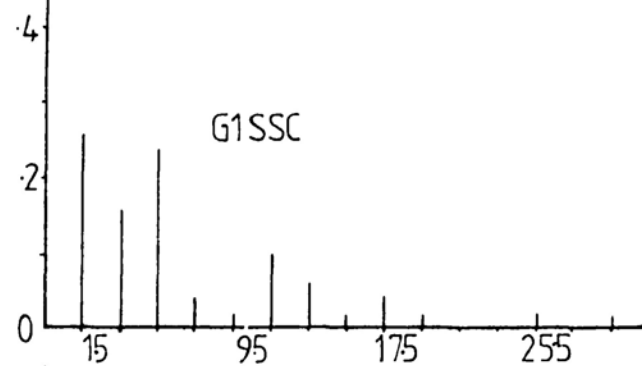
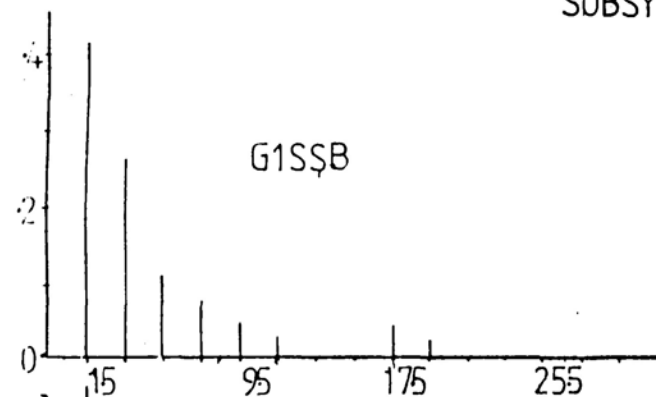


FIGURE 40

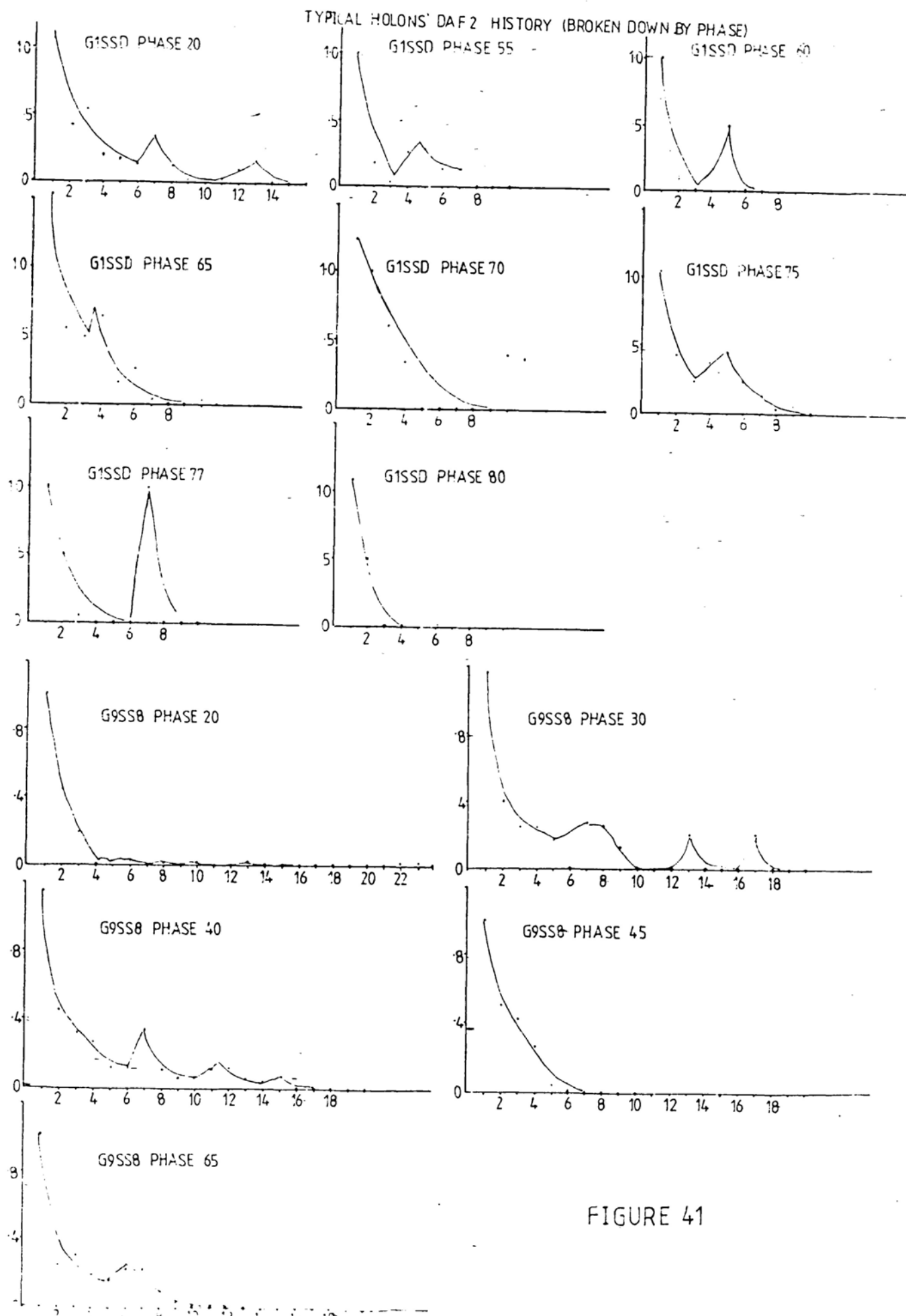
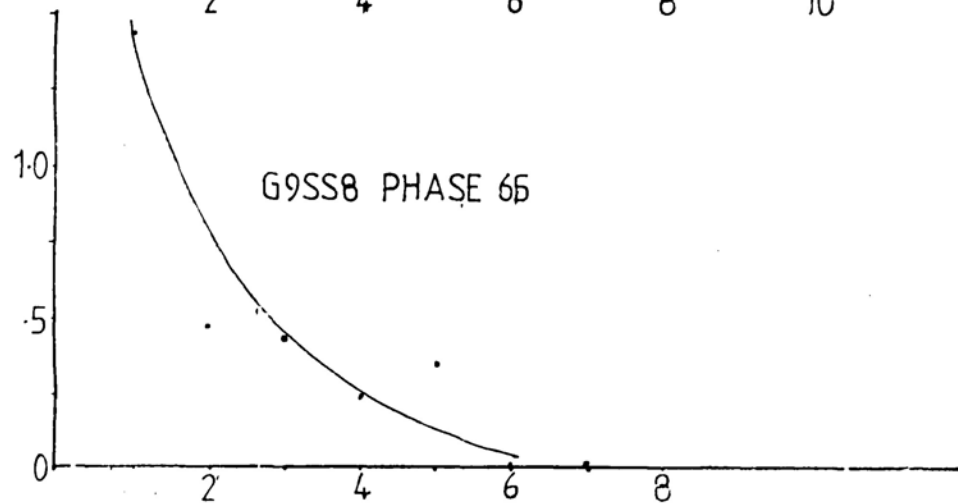
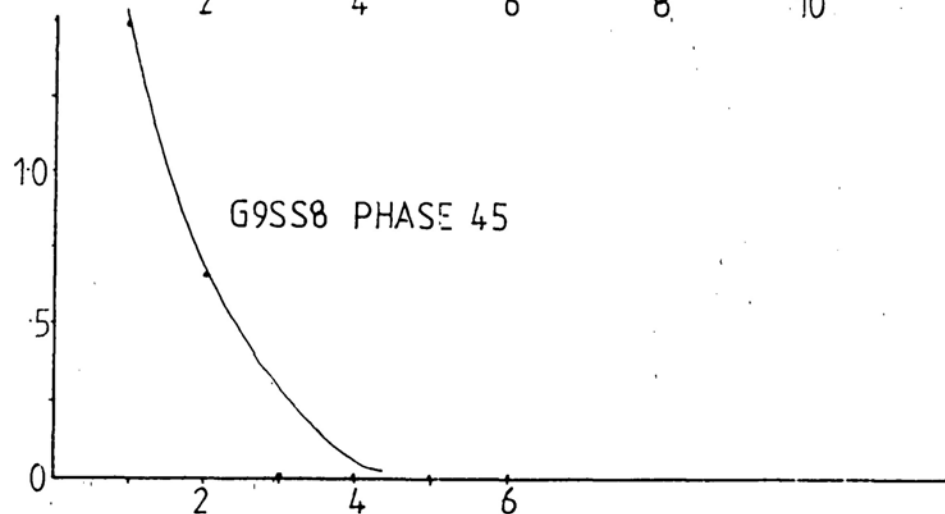
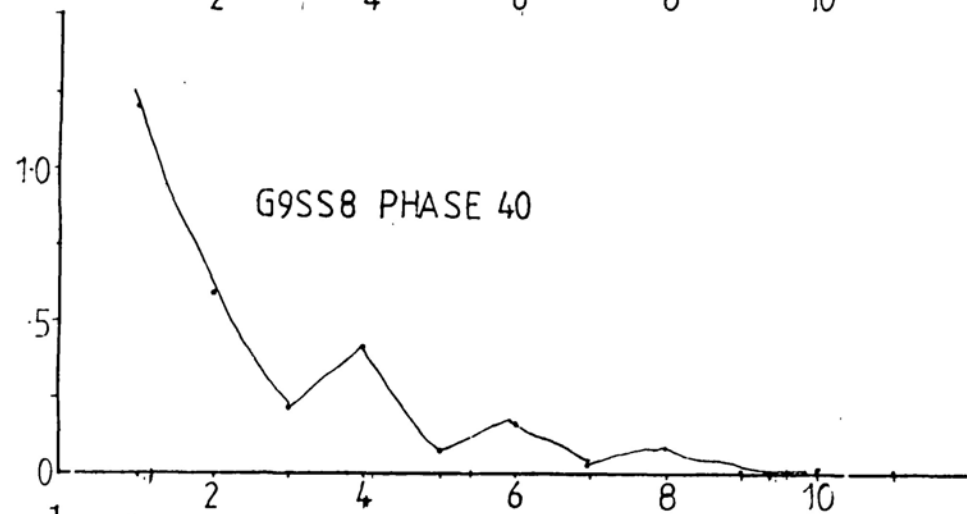
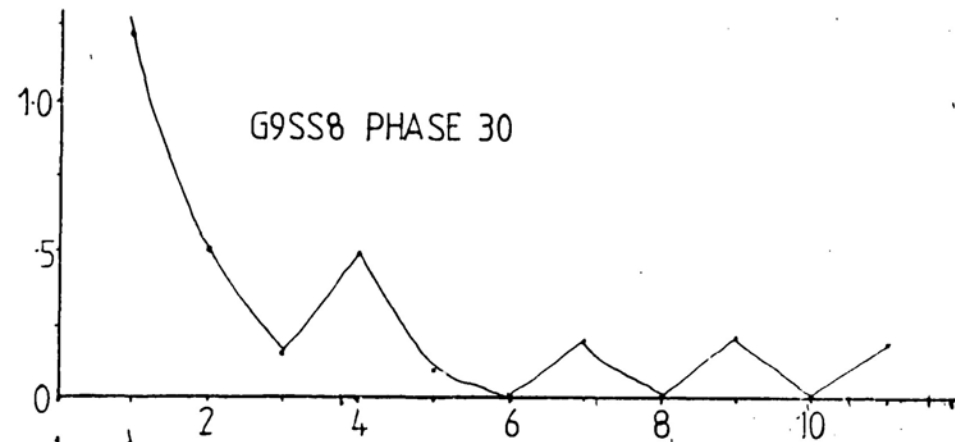
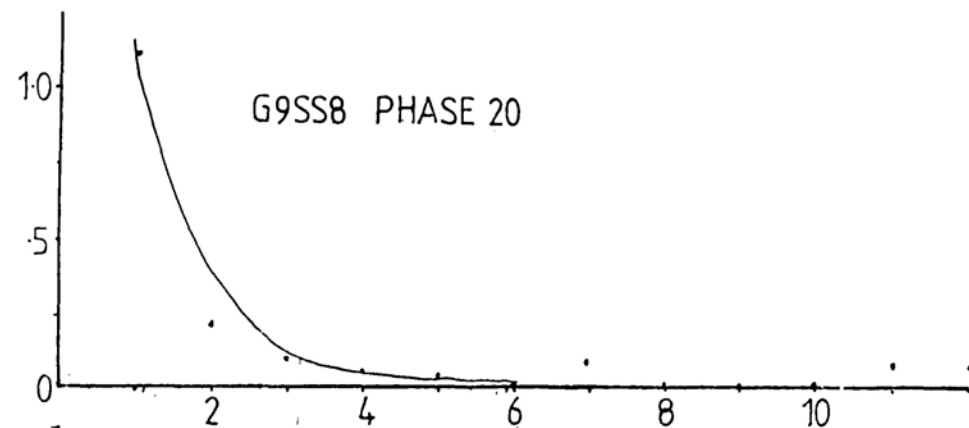


FIGURE 41



TYPICAL HOLON S DAF2 HISTORY
(SMOOTHED DATA)

FIGURE 42

SOFTWARE NOTICES

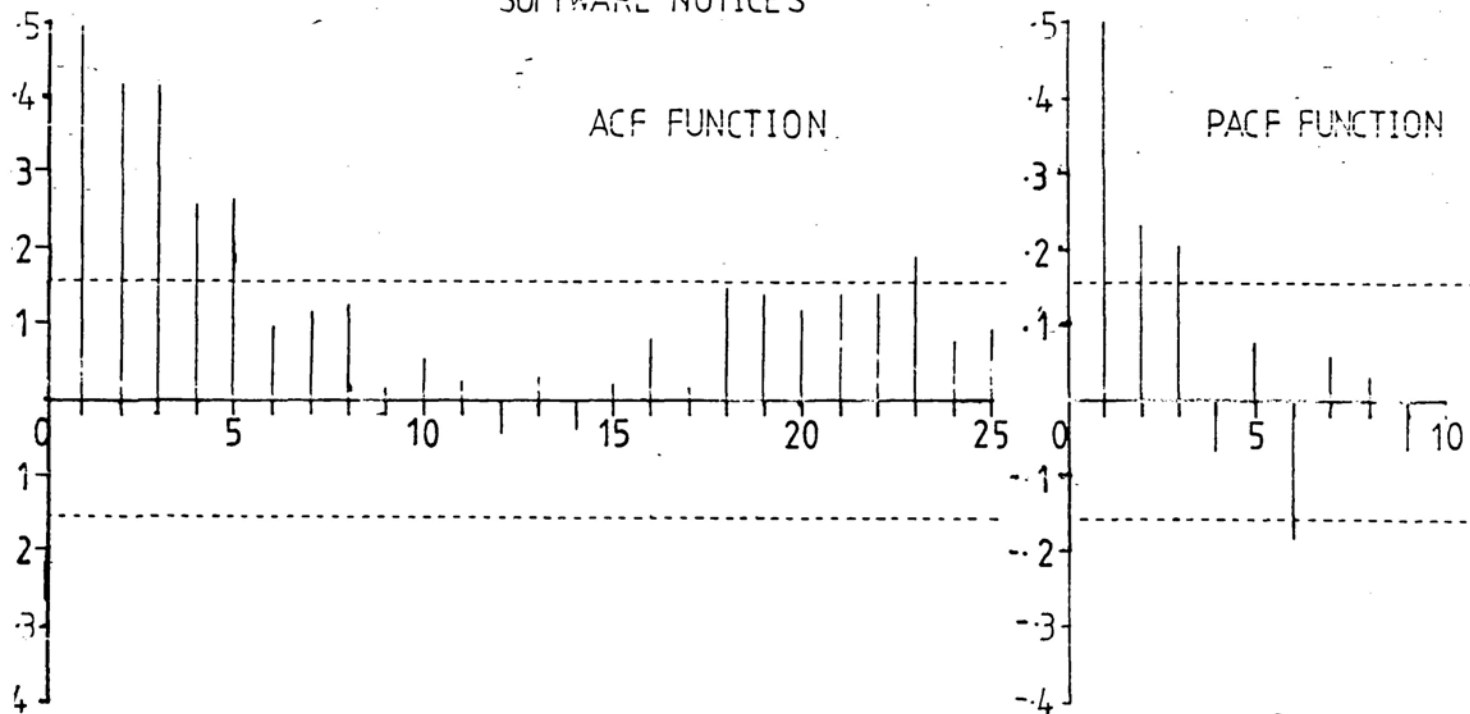


FIGURE 43

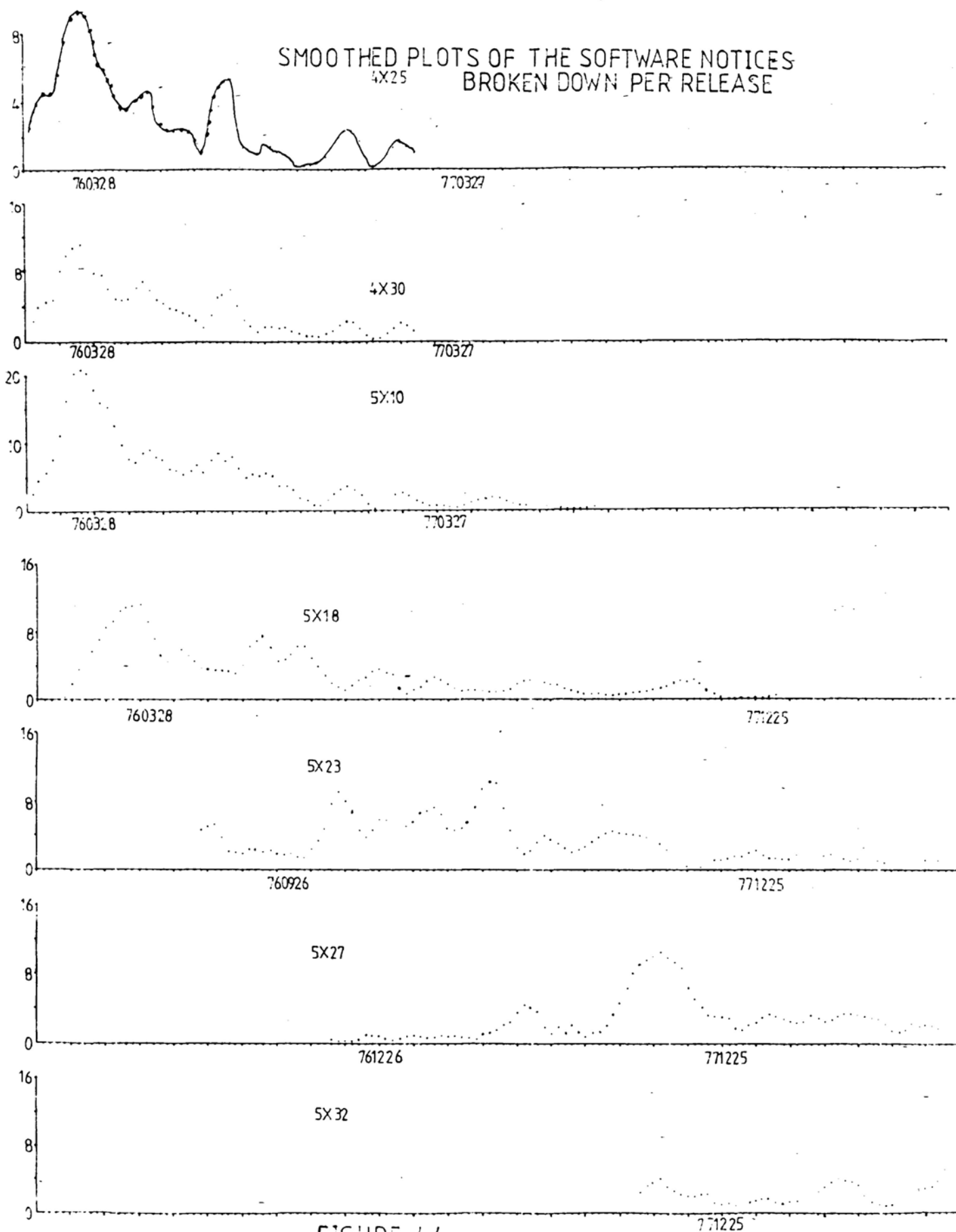
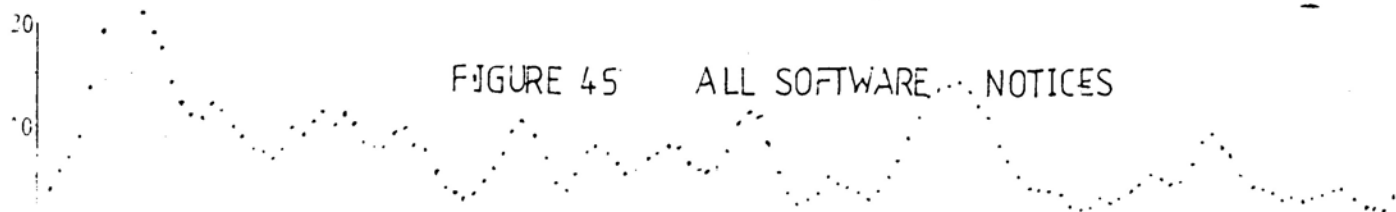
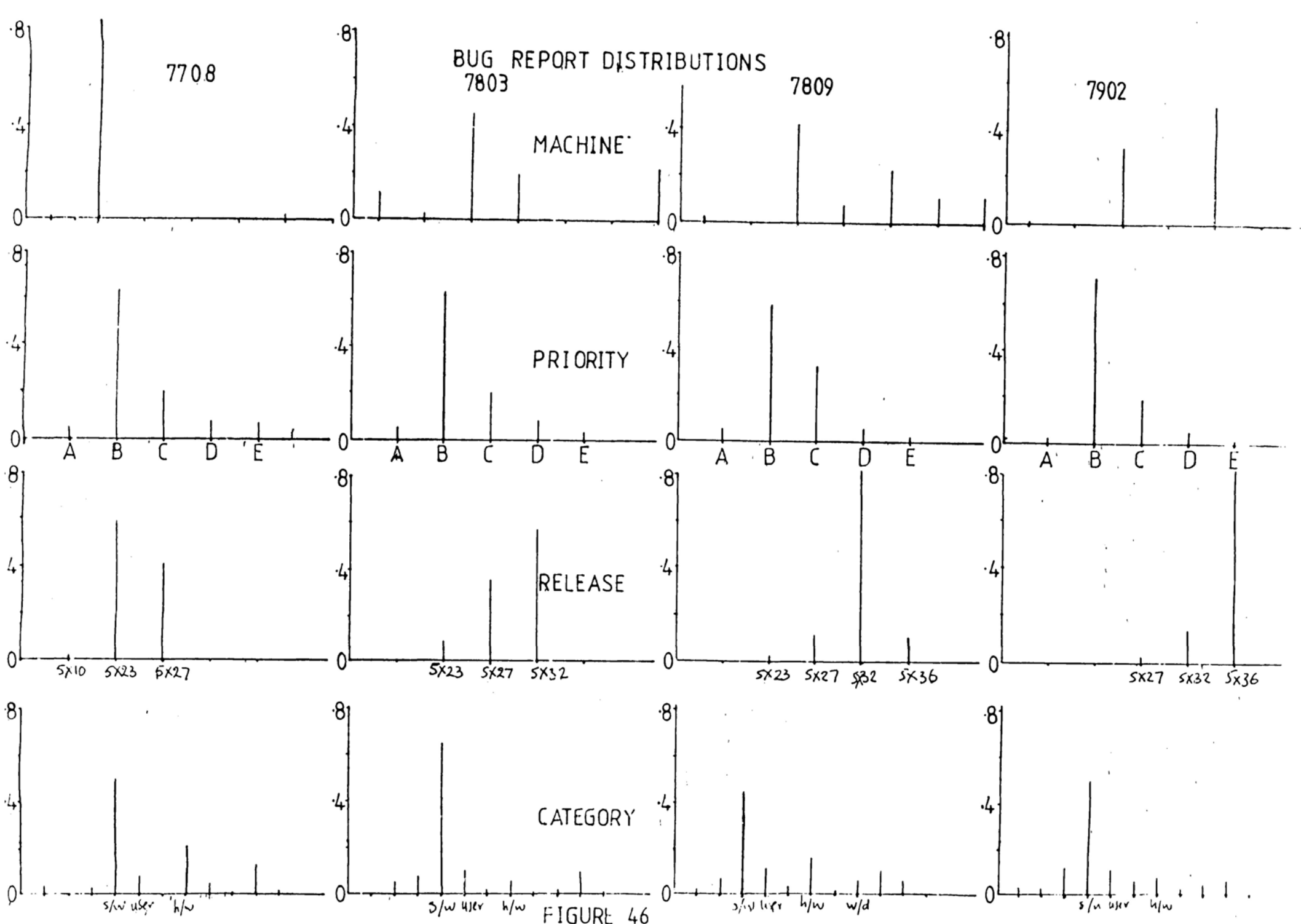


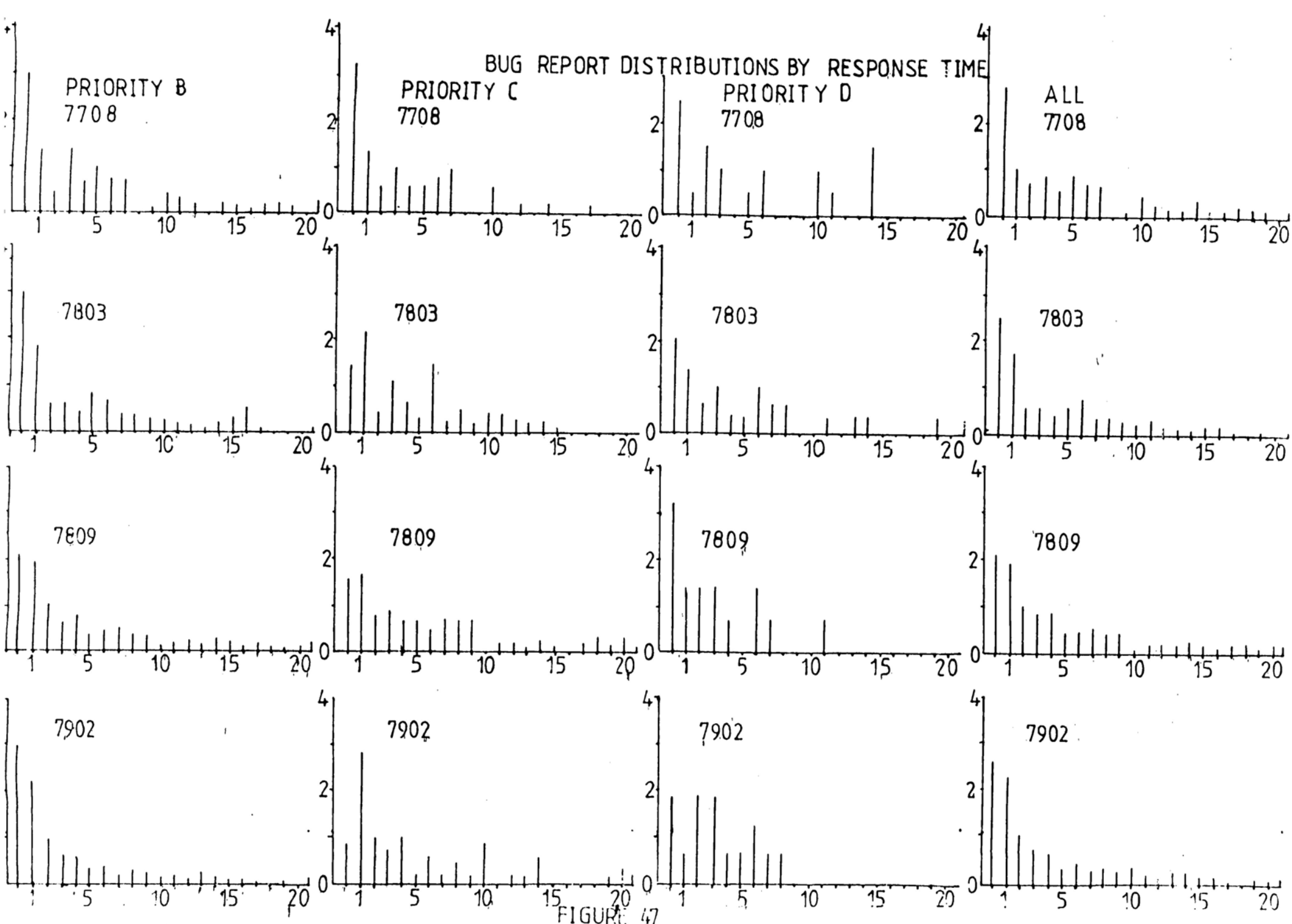
FIGURE 44

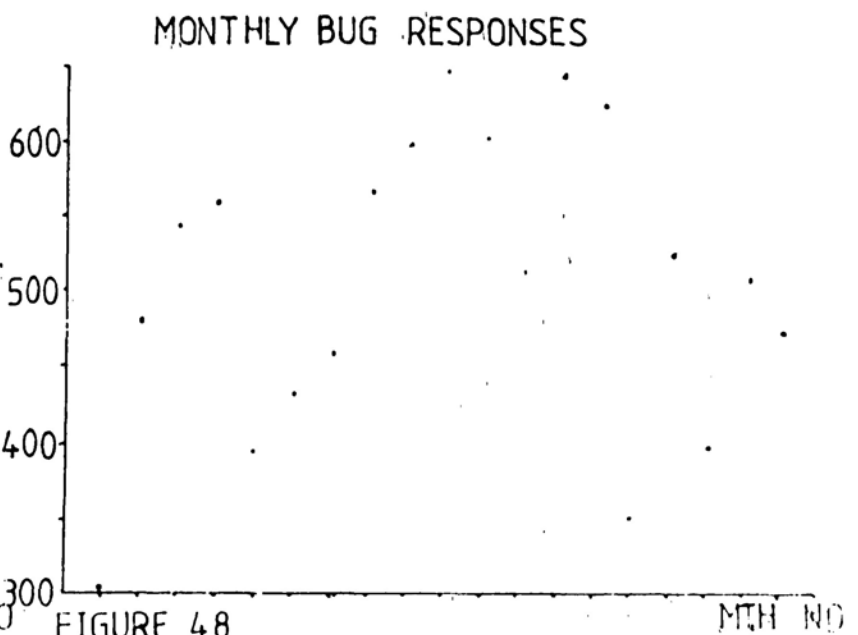
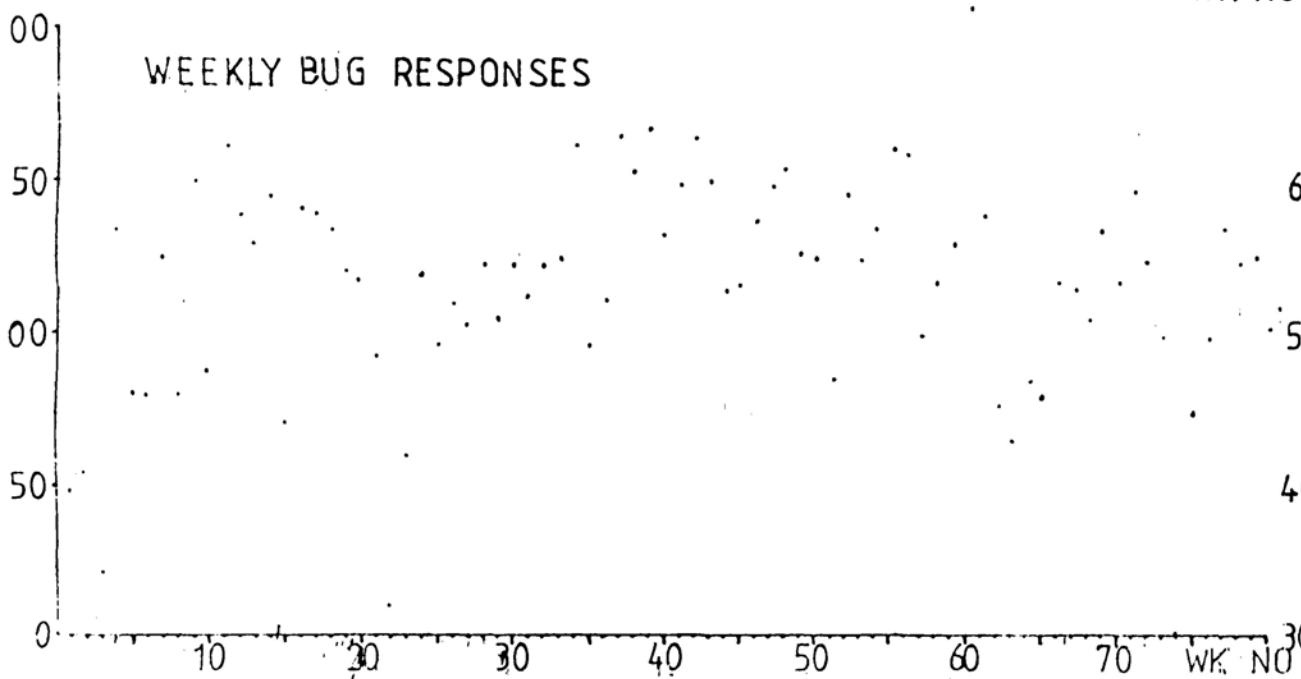
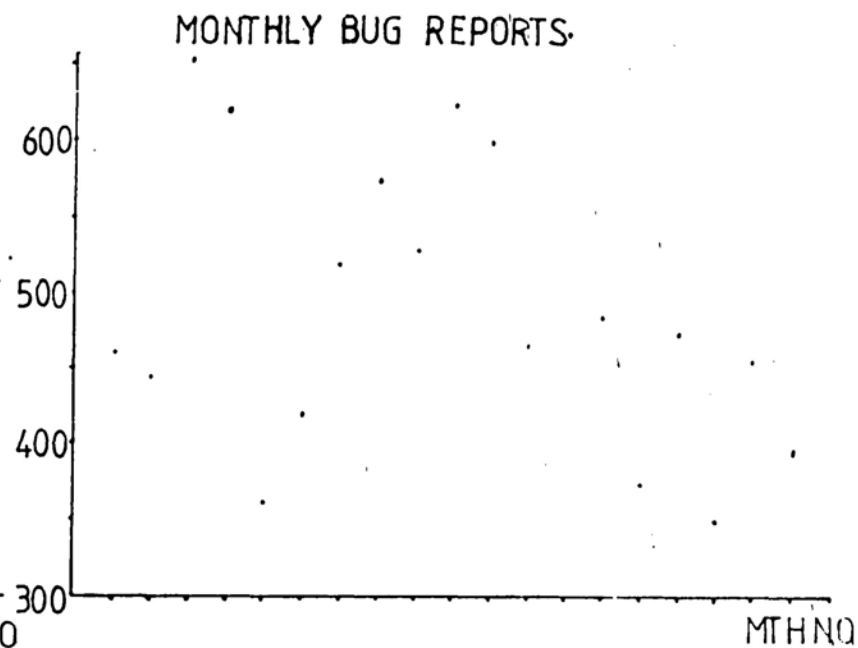
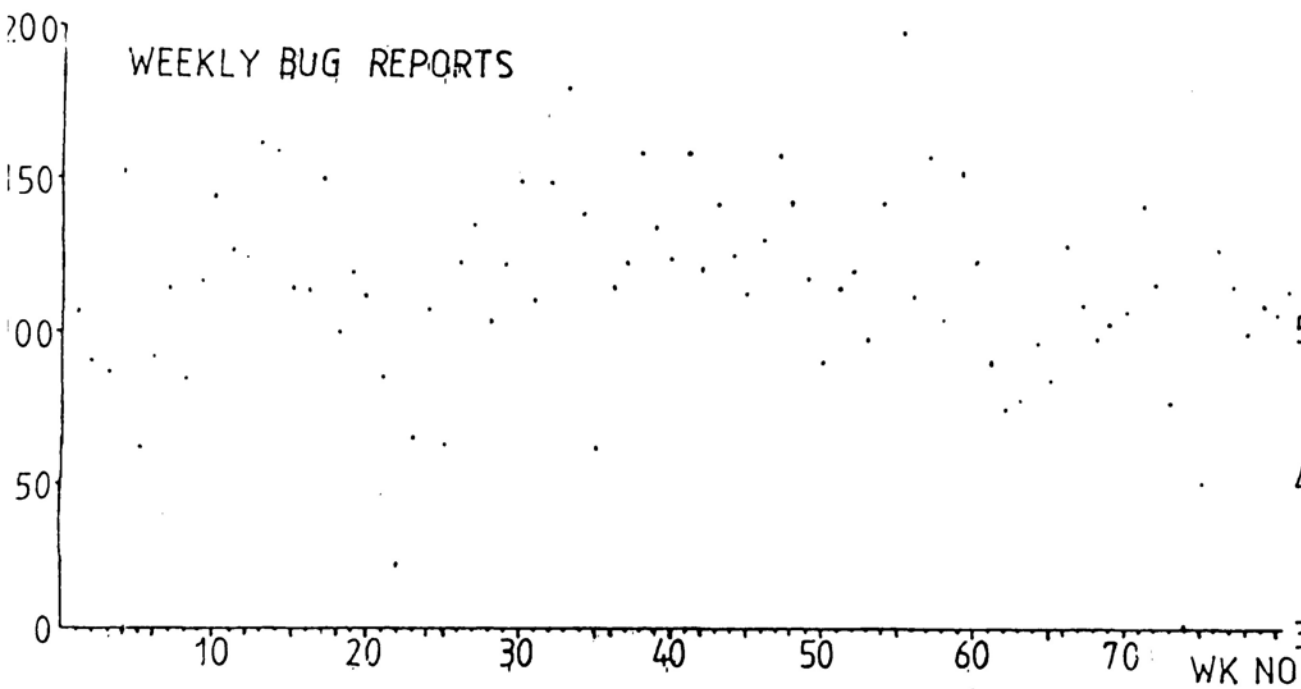
FIGURE 45 ALL SOFTWARE NOTICES





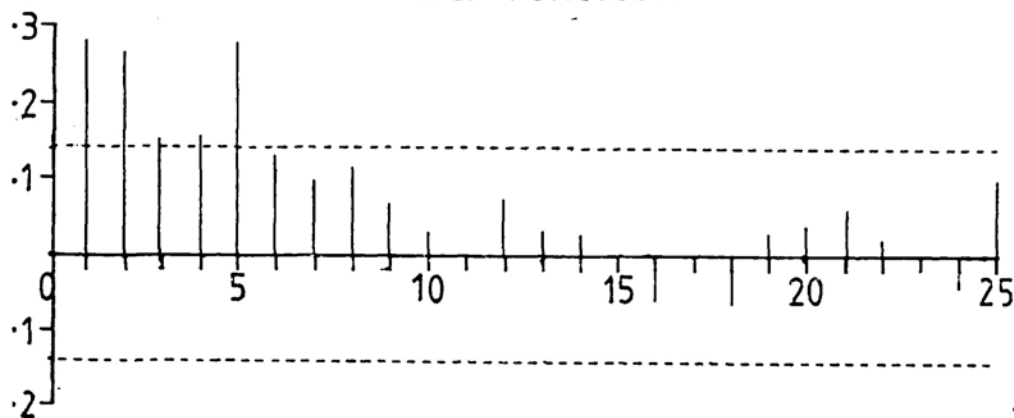
FIGURES



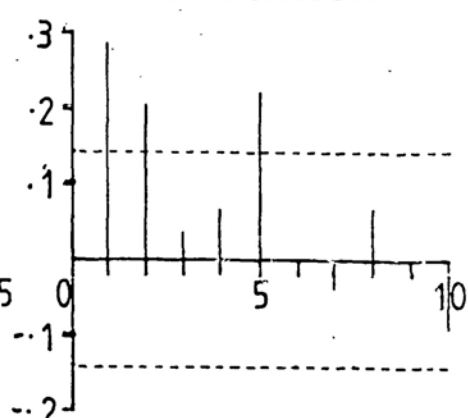


DAILY BUG REPORTS

ACF FUNCTION

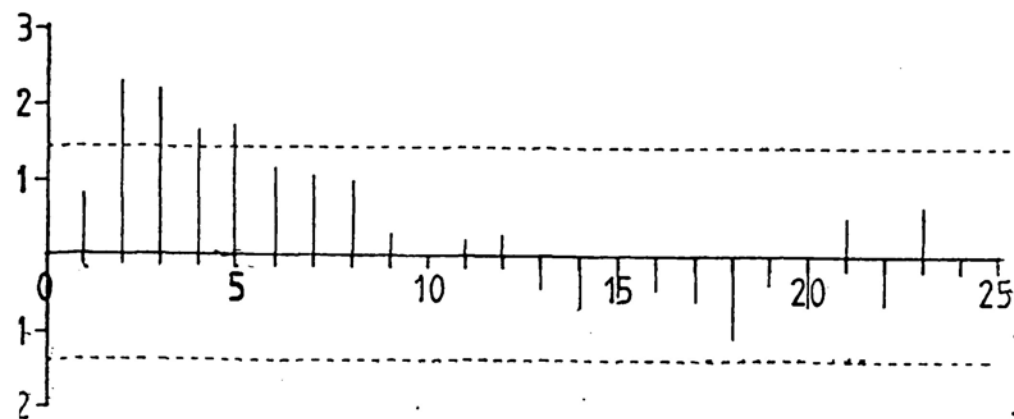


PACF FUNCTION



DAILY BUG RESPONSES

ACF FUNCTION



PACF FUNCTION

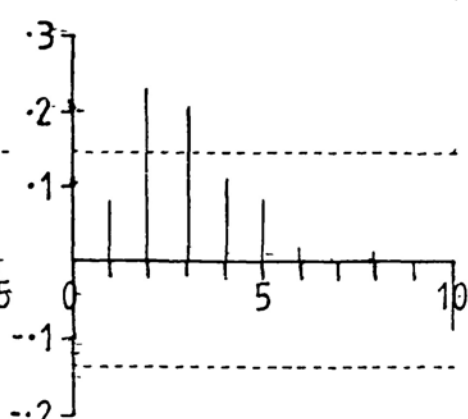
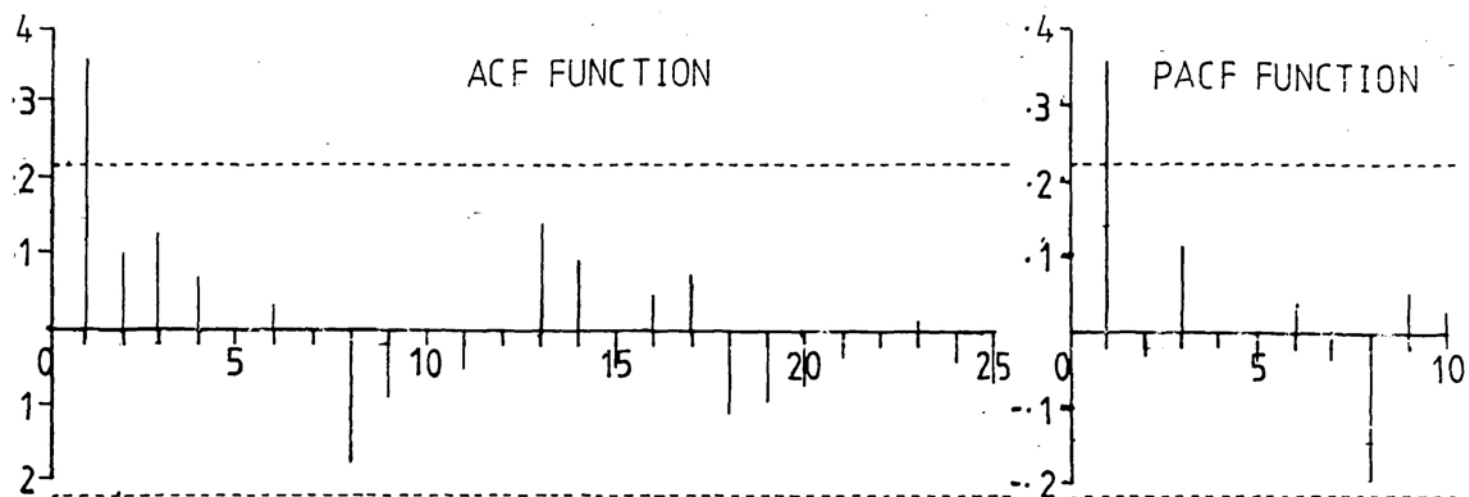


FIGURE 49

WEEKLY BUG REPORTS



WEEKLY BUG RESPONSES

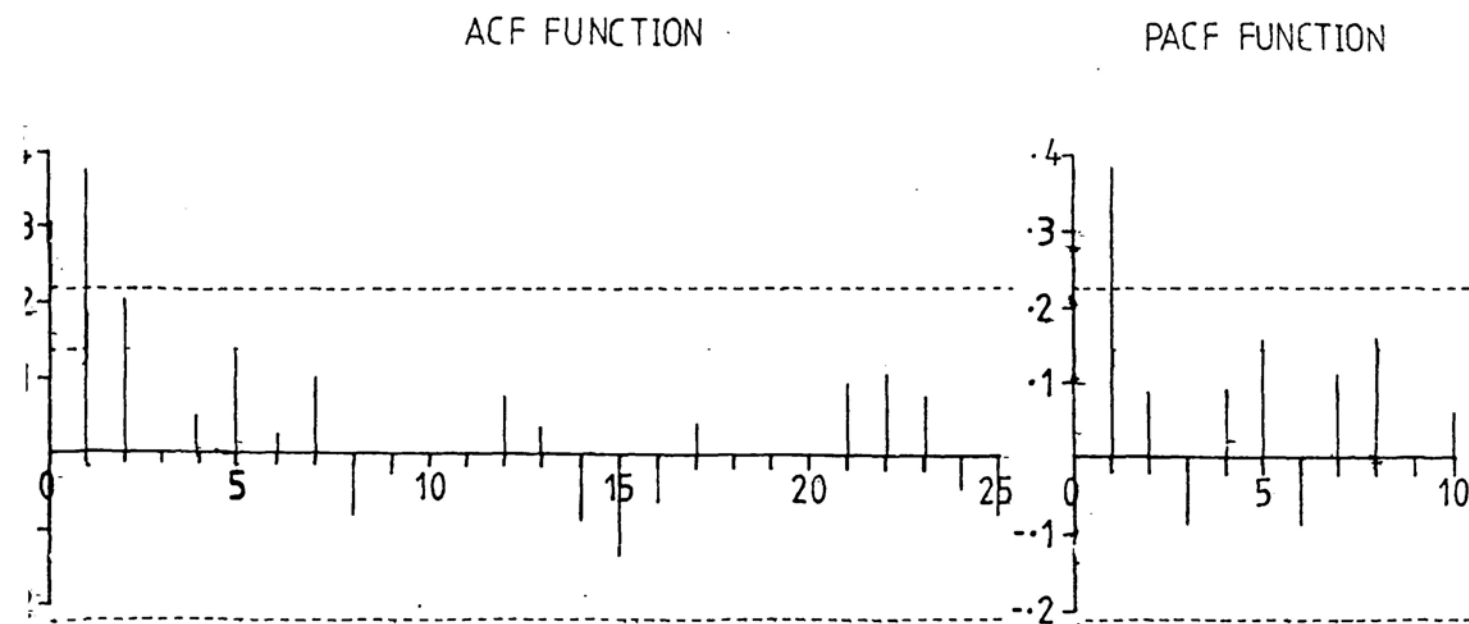
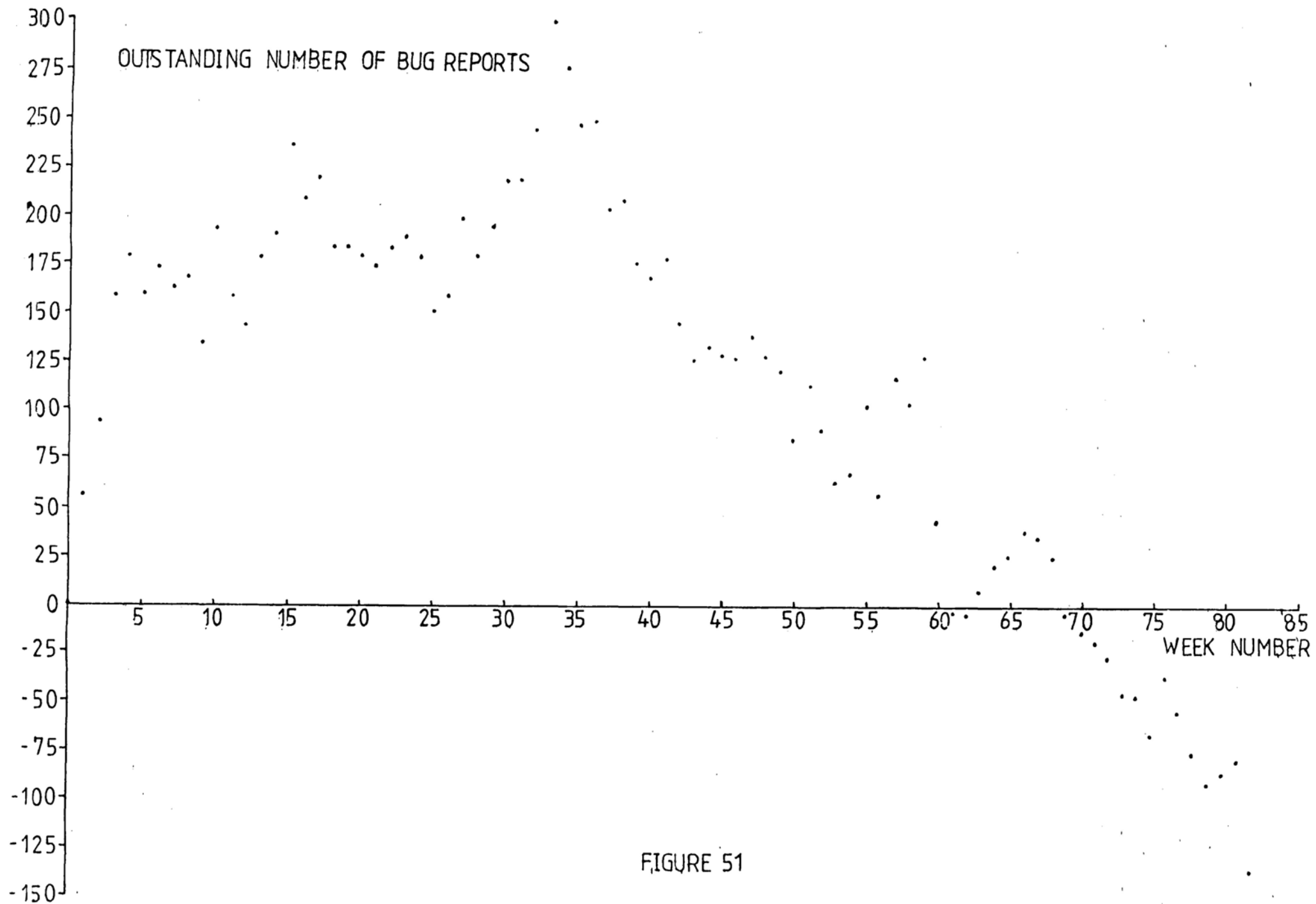


FIGURE 50



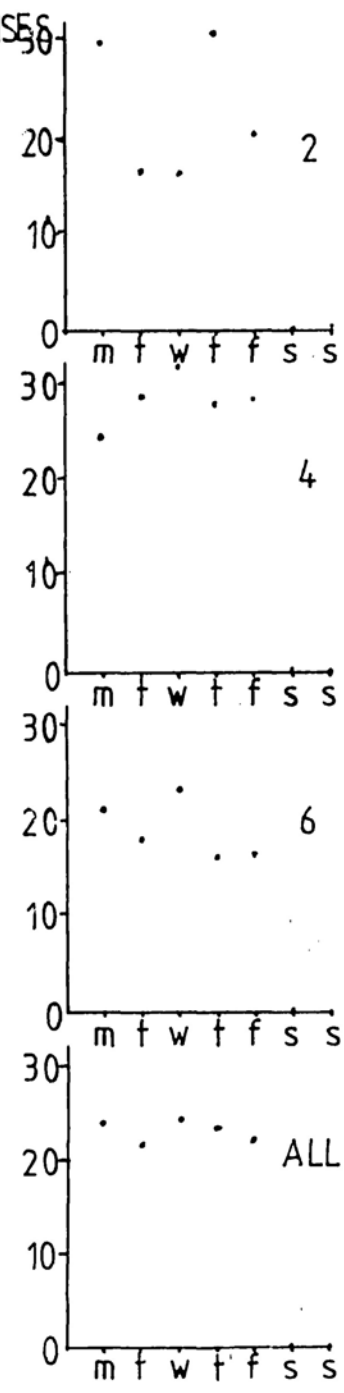
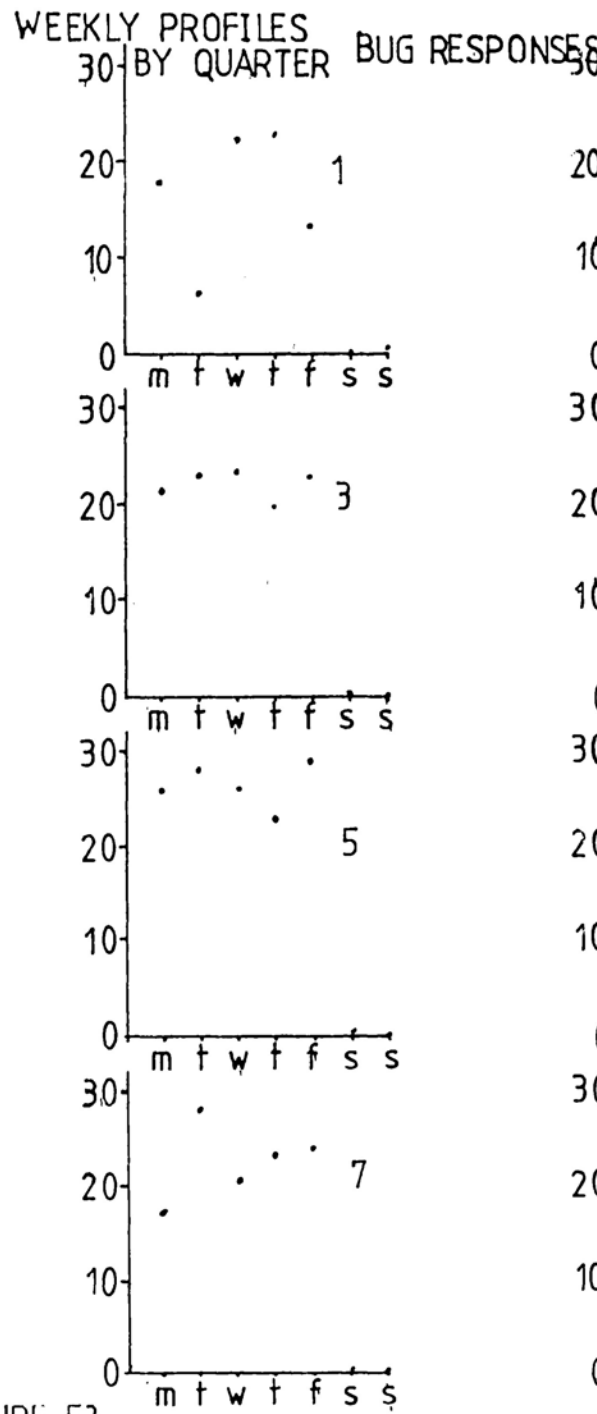
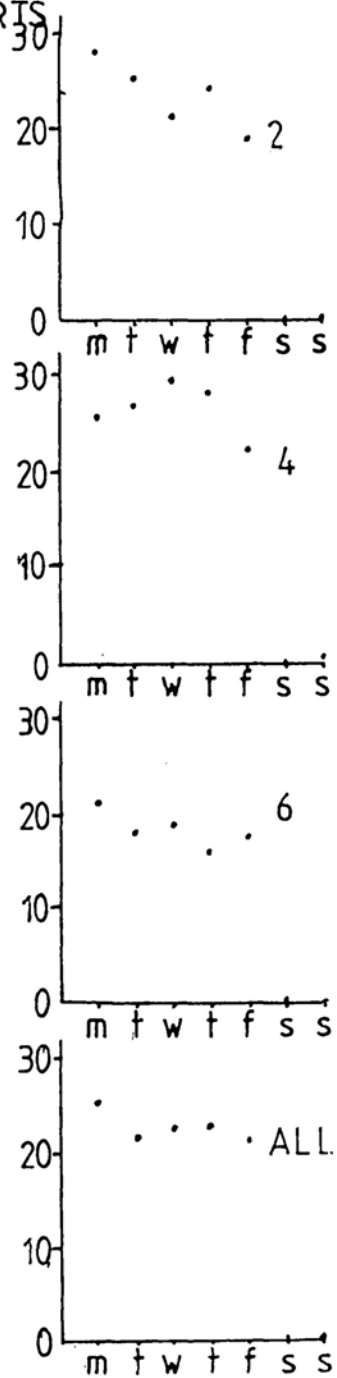
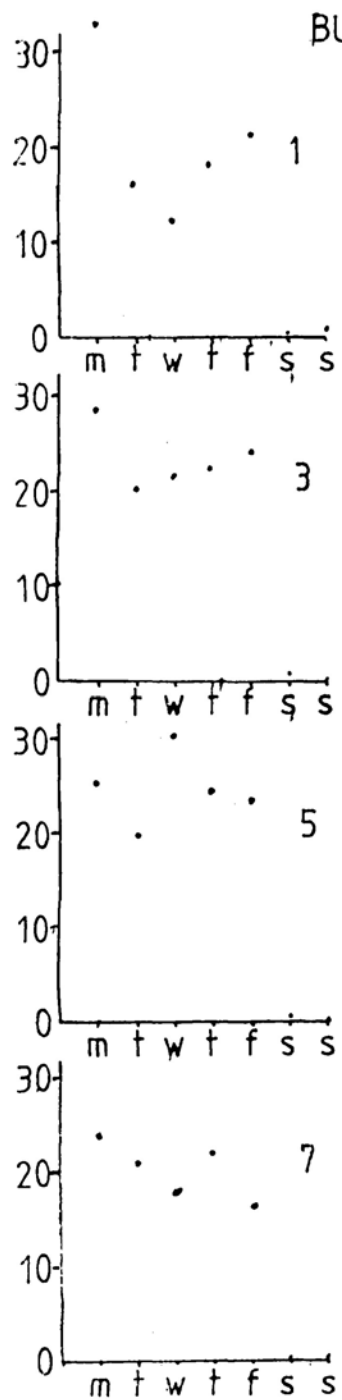


FIGURE 52

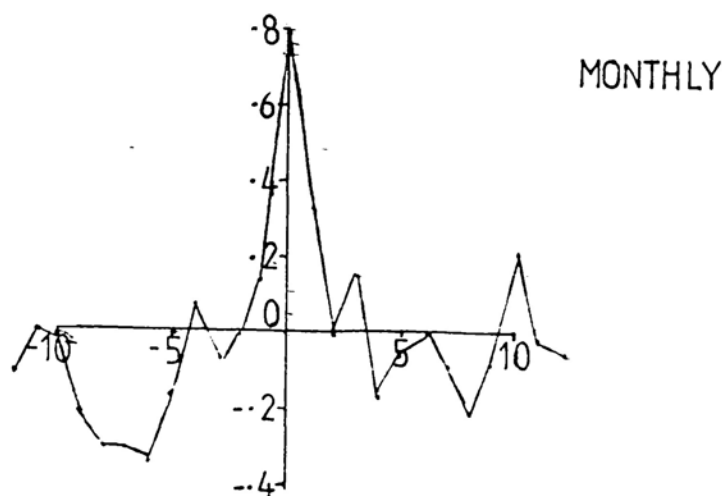
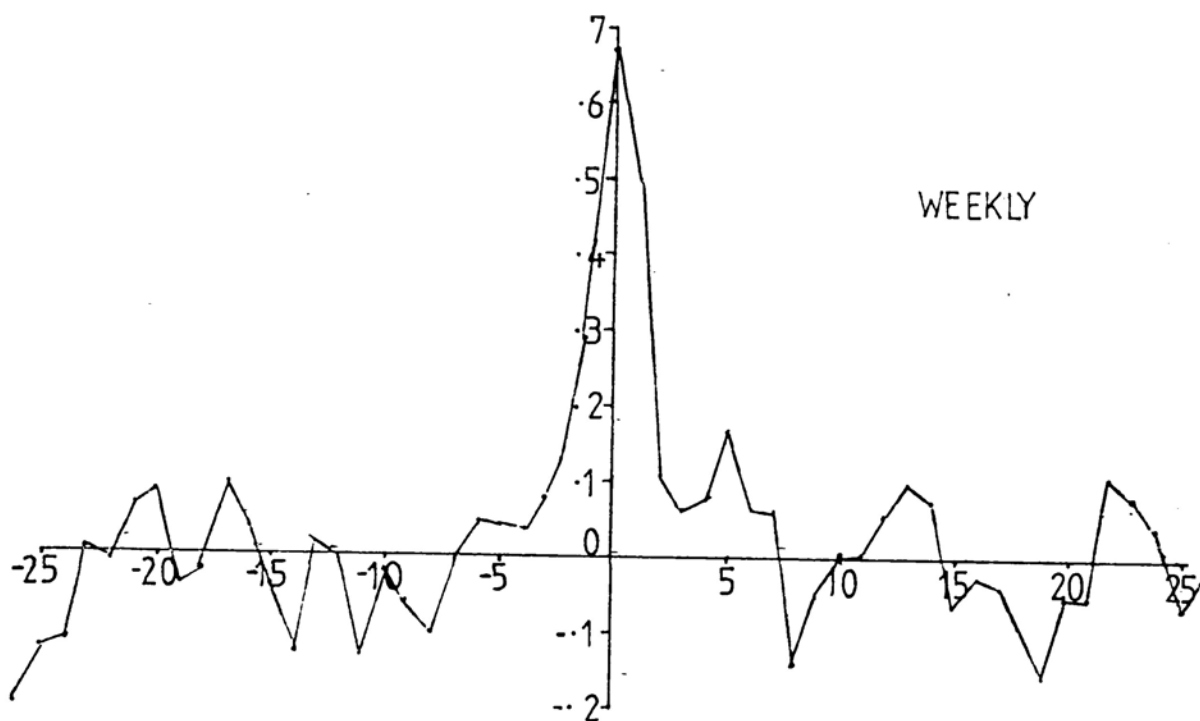
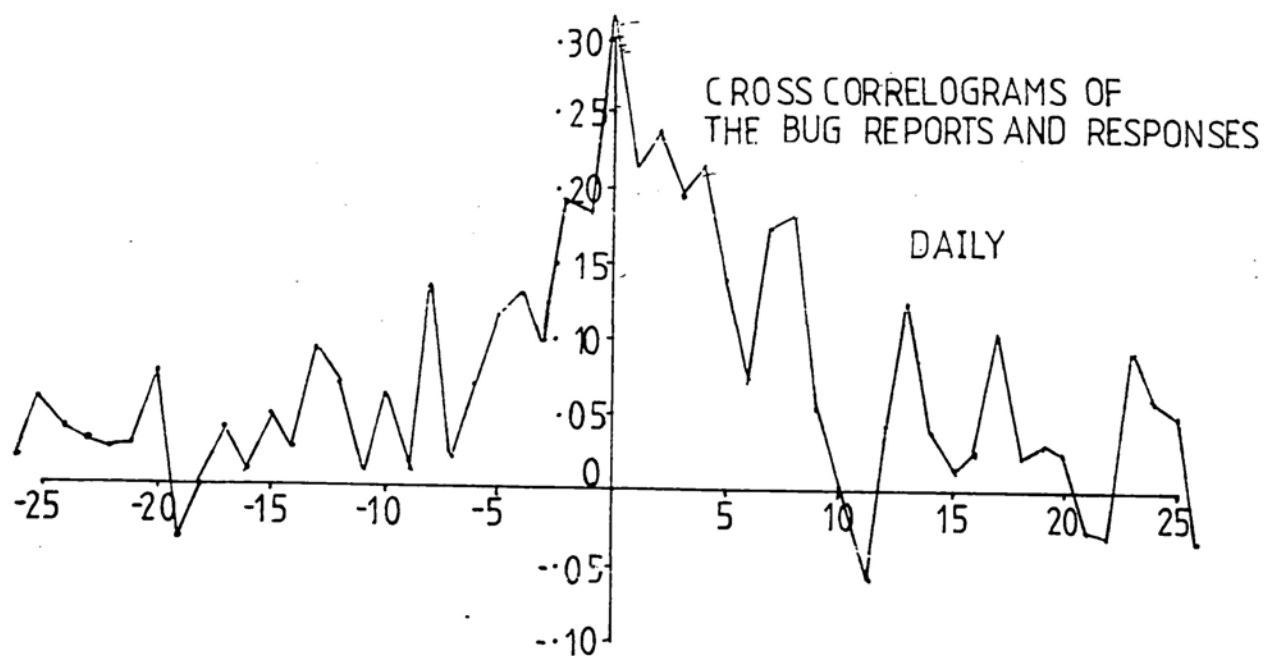


FIGURE 53



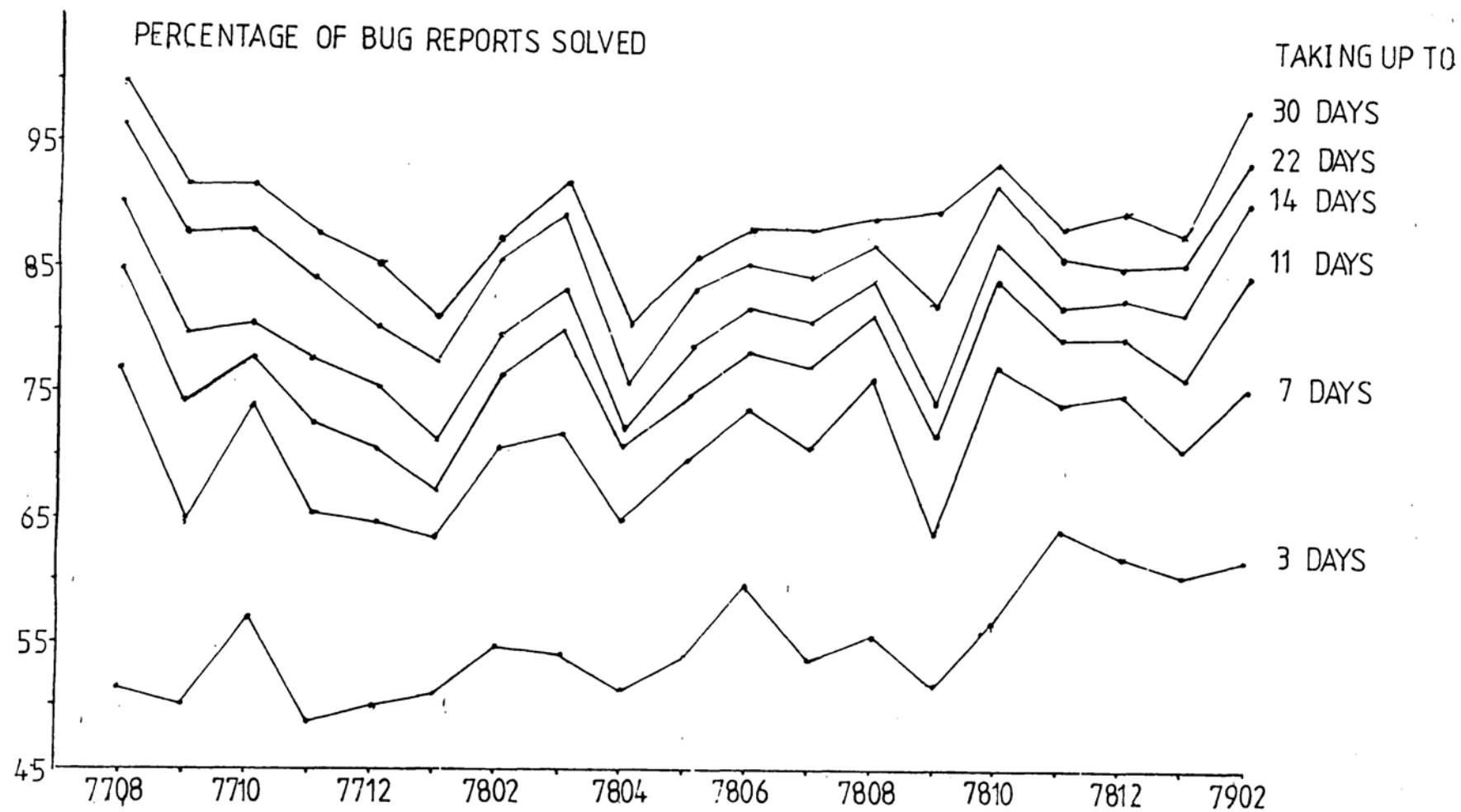
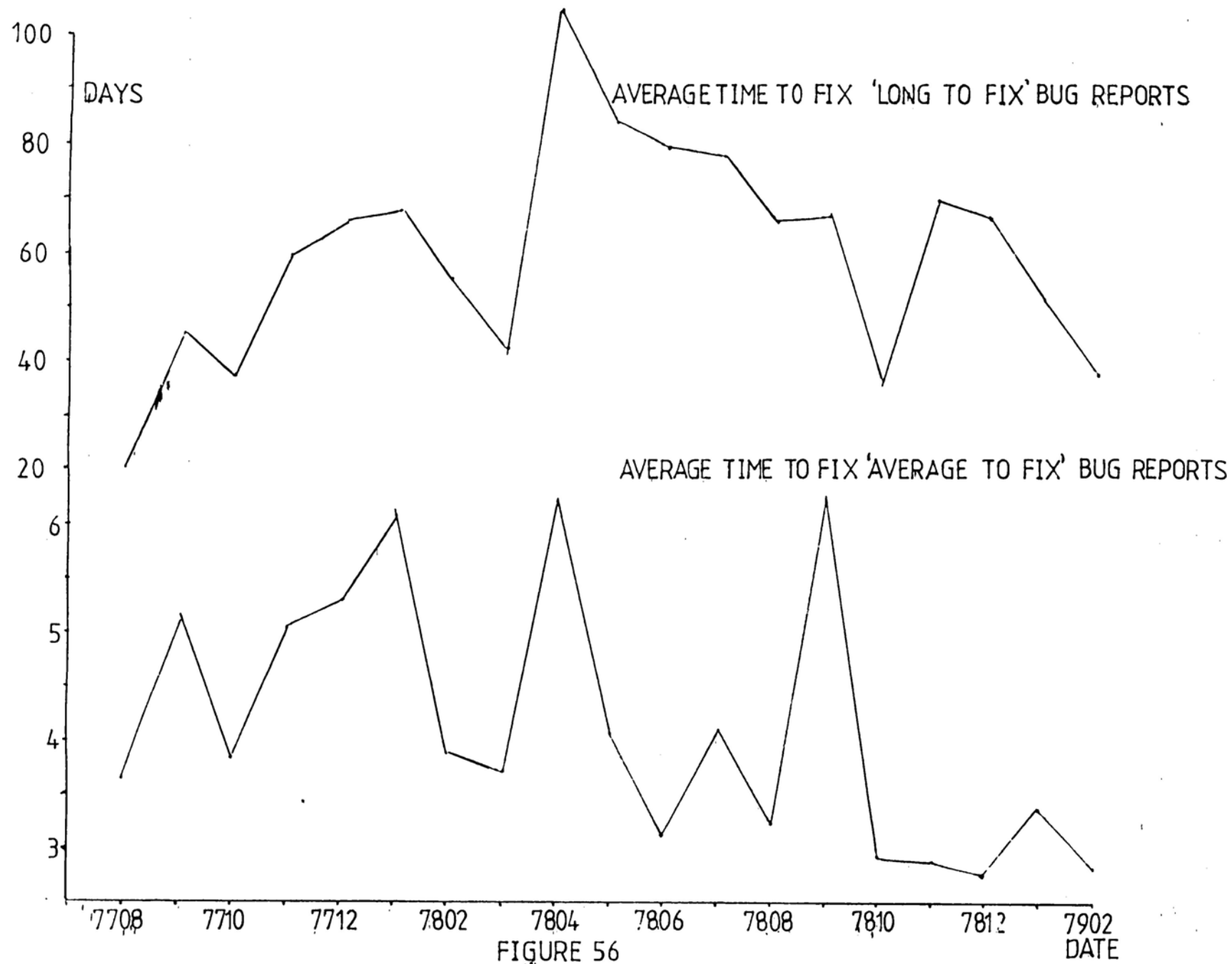
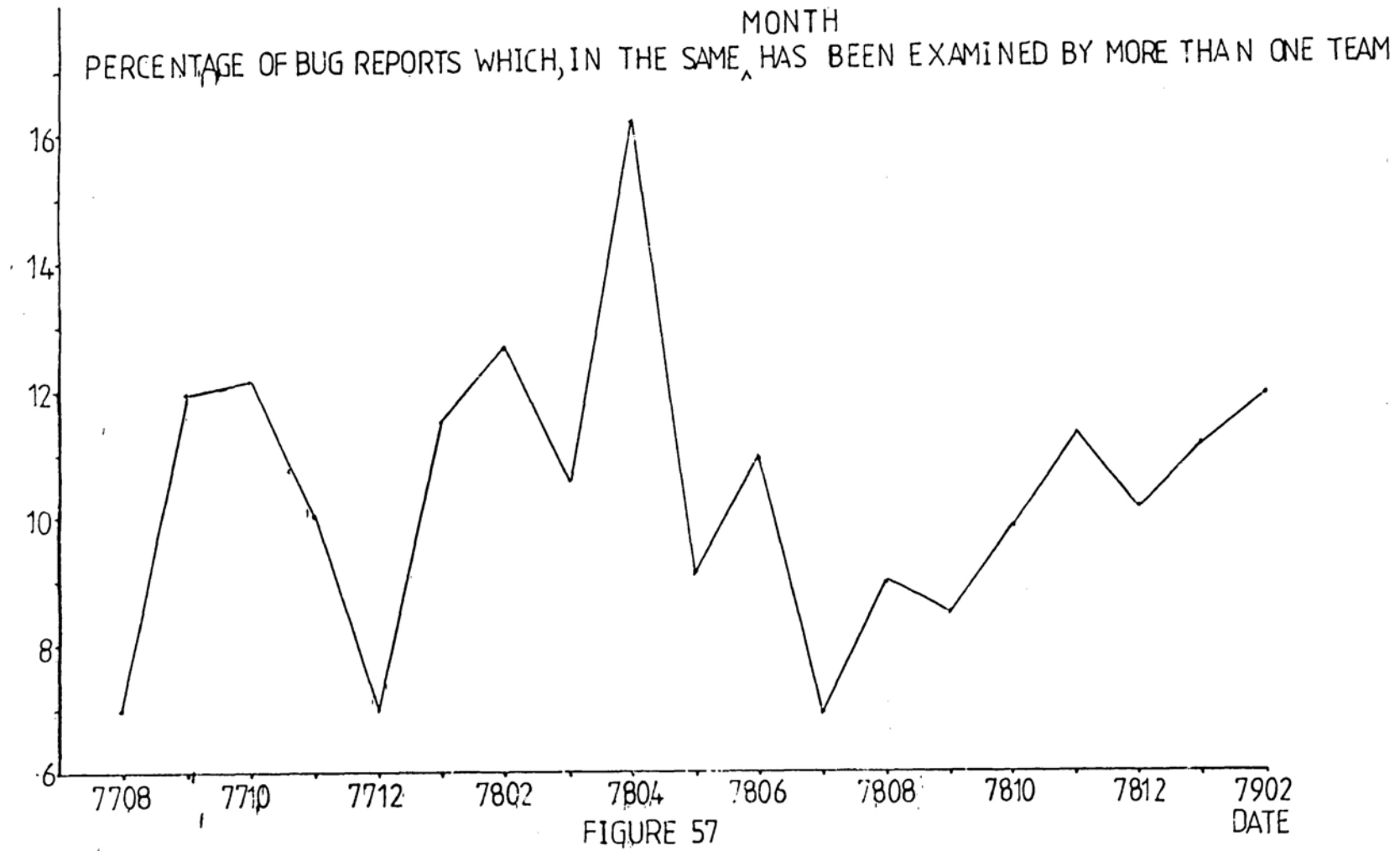
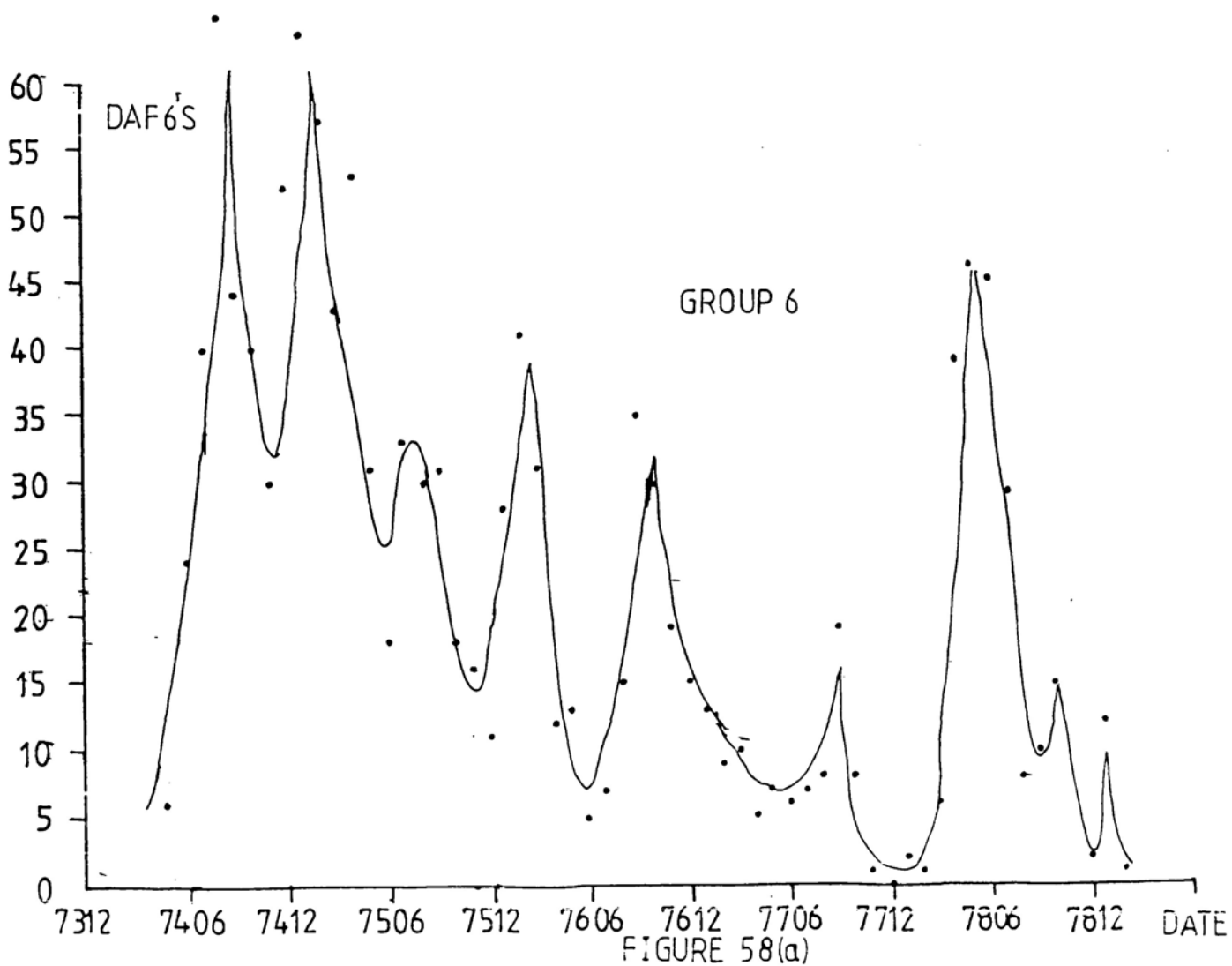
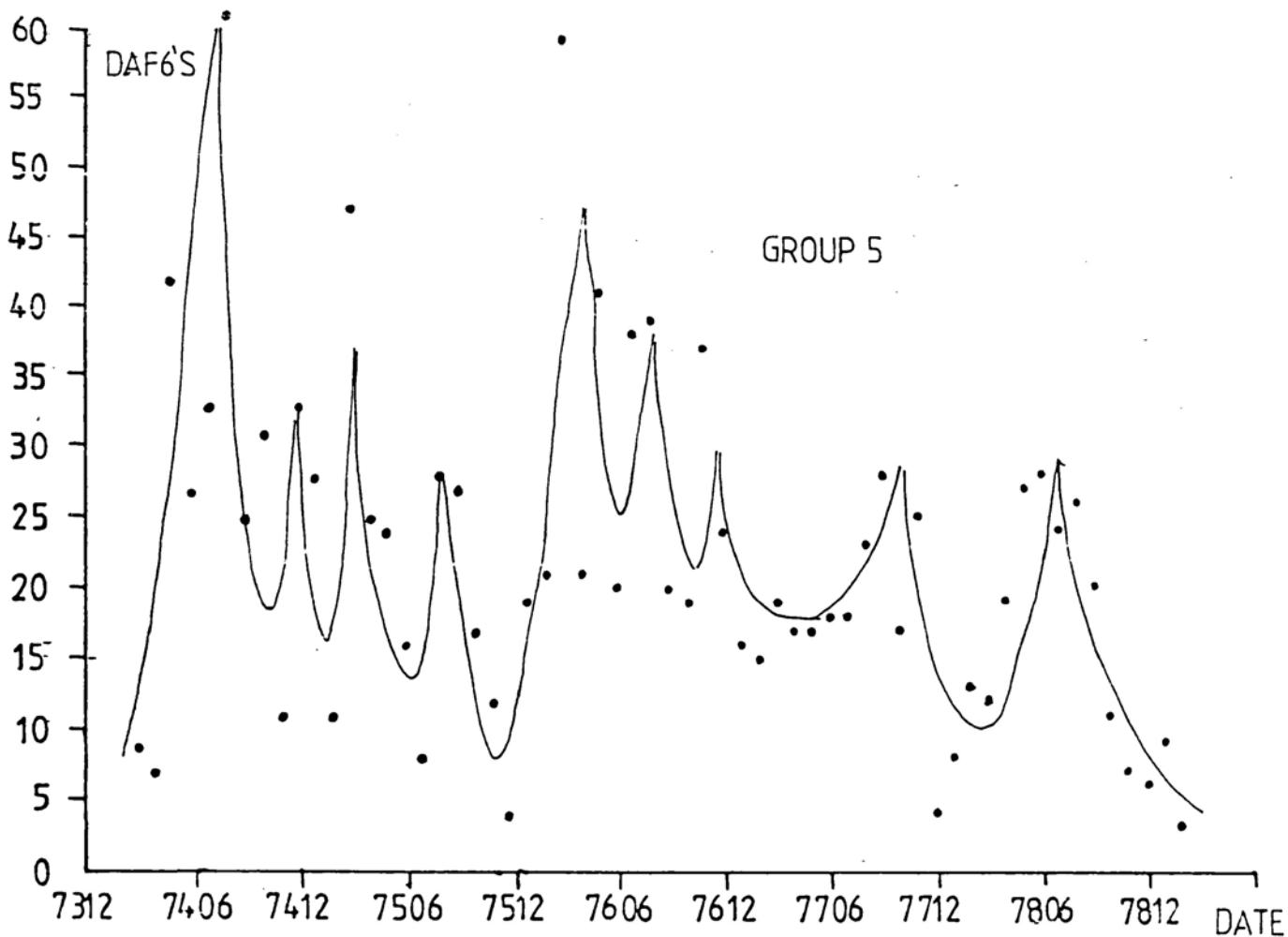
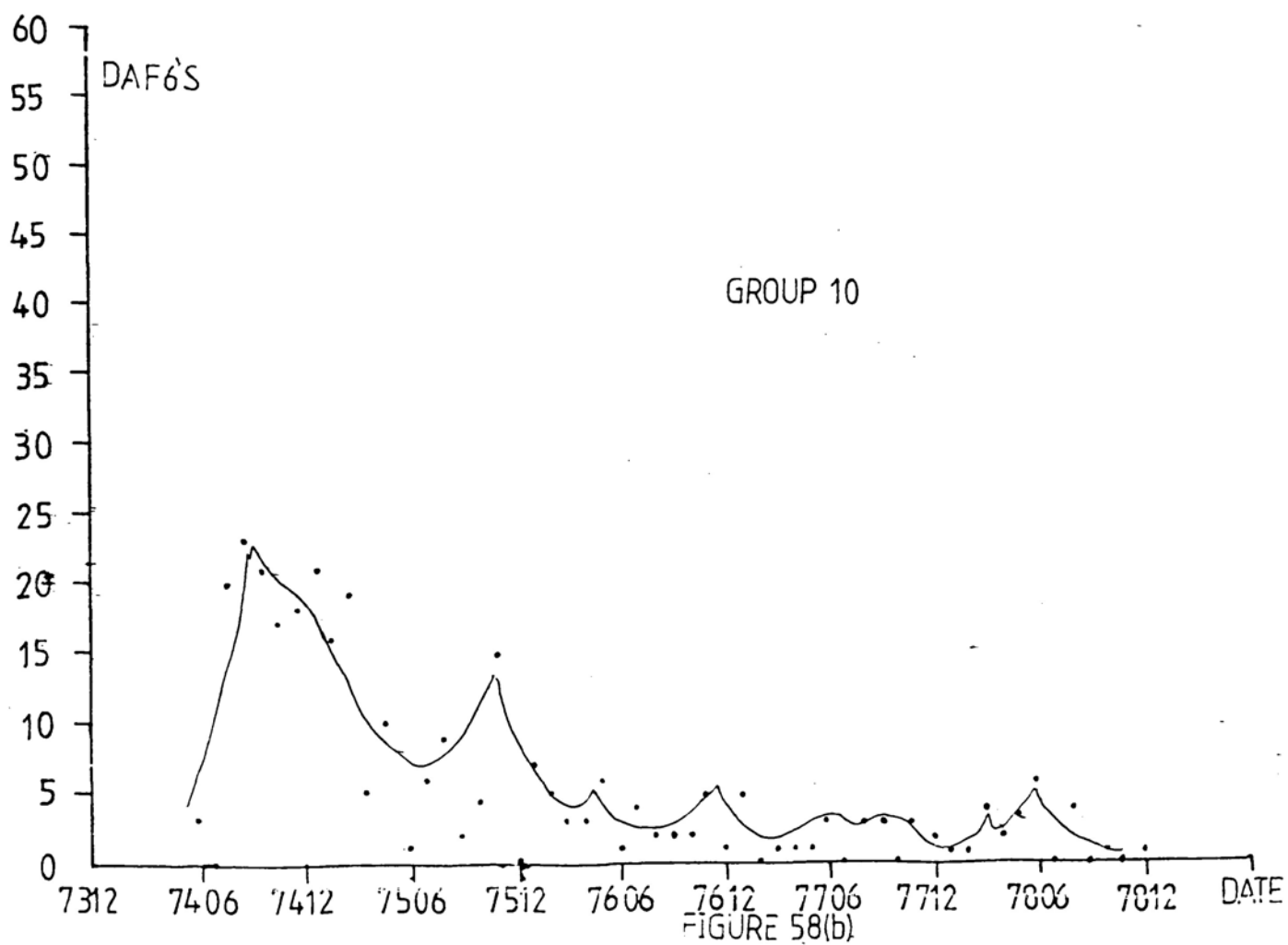
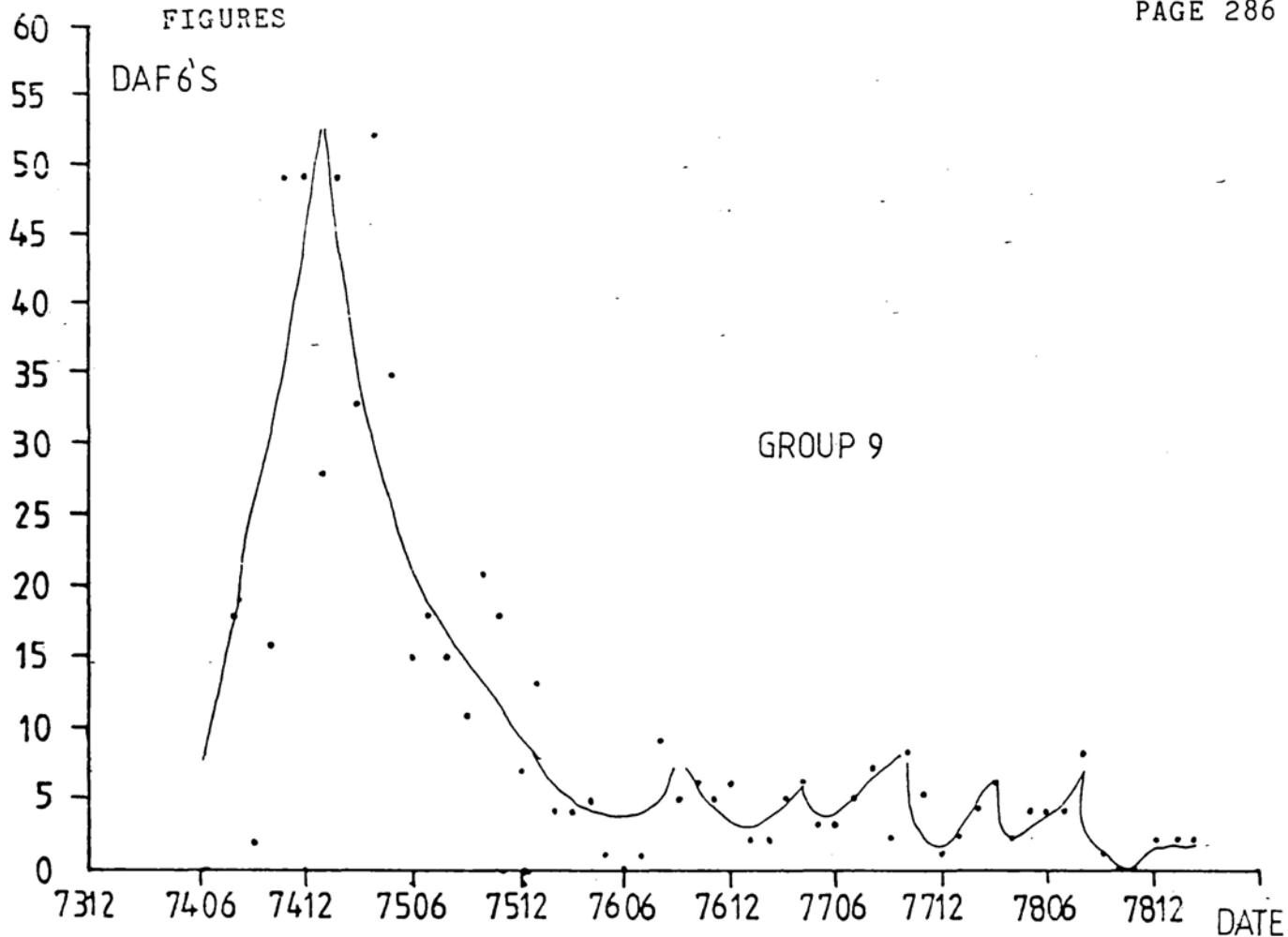


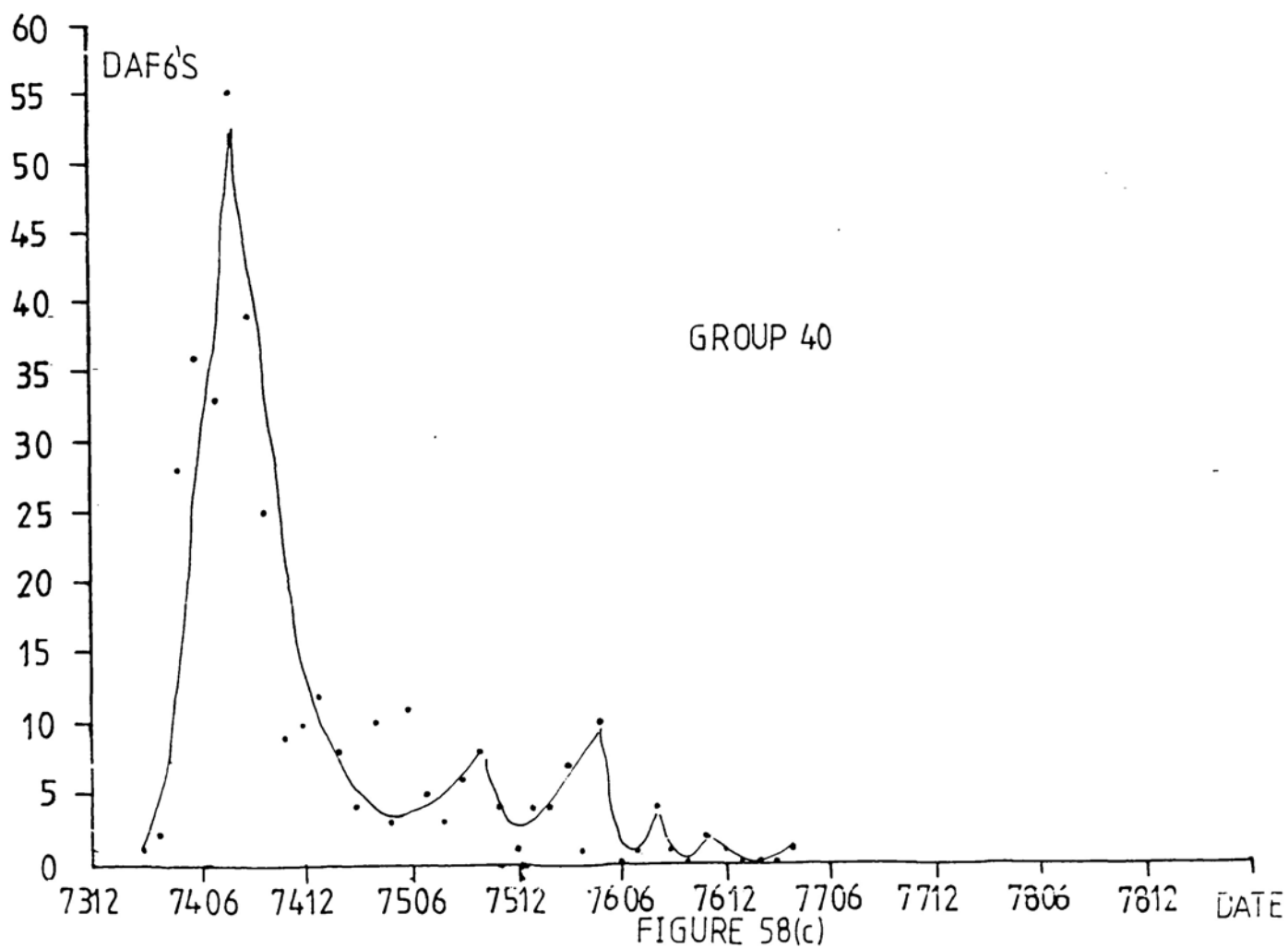
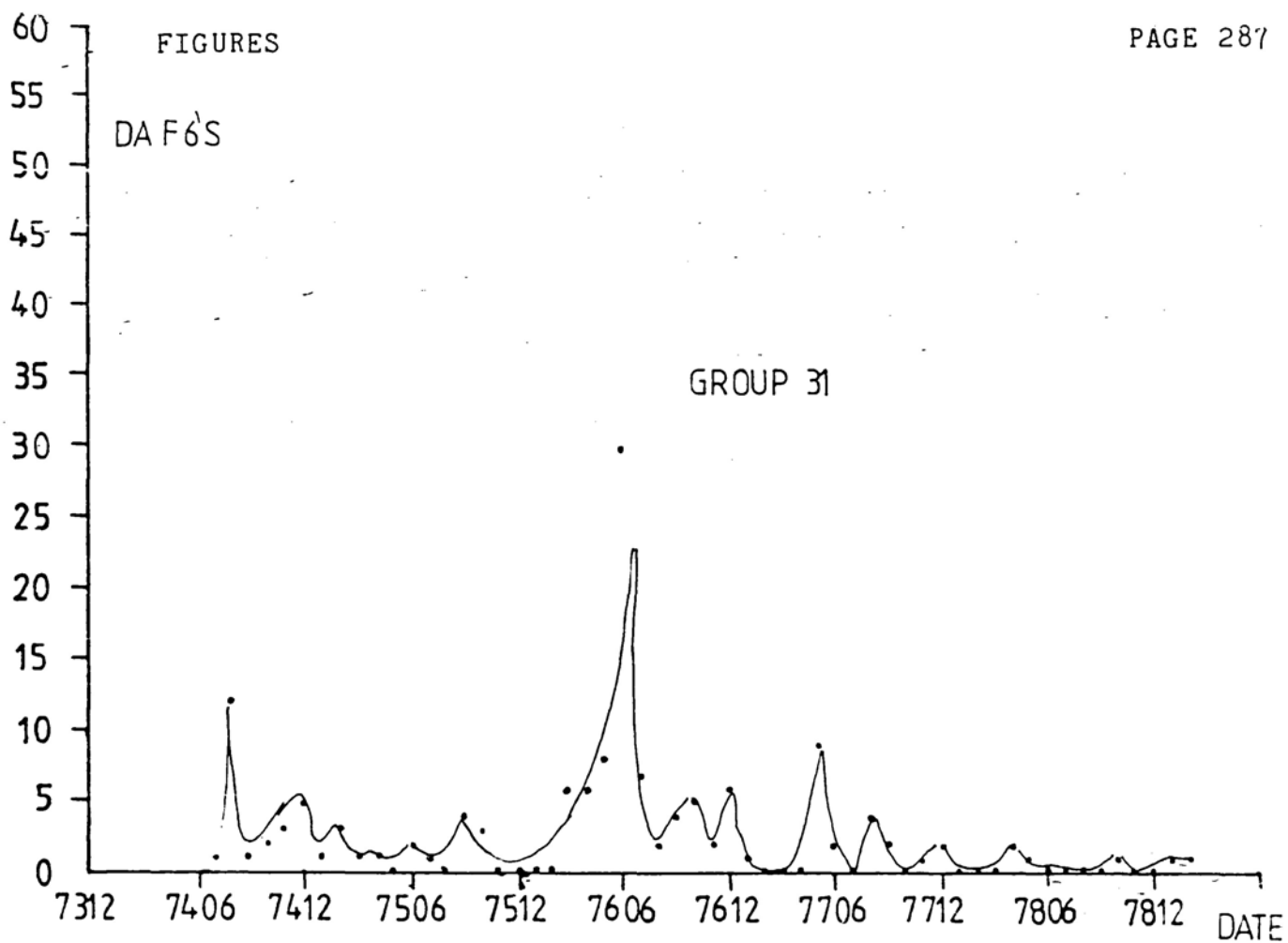
FIGURE 55











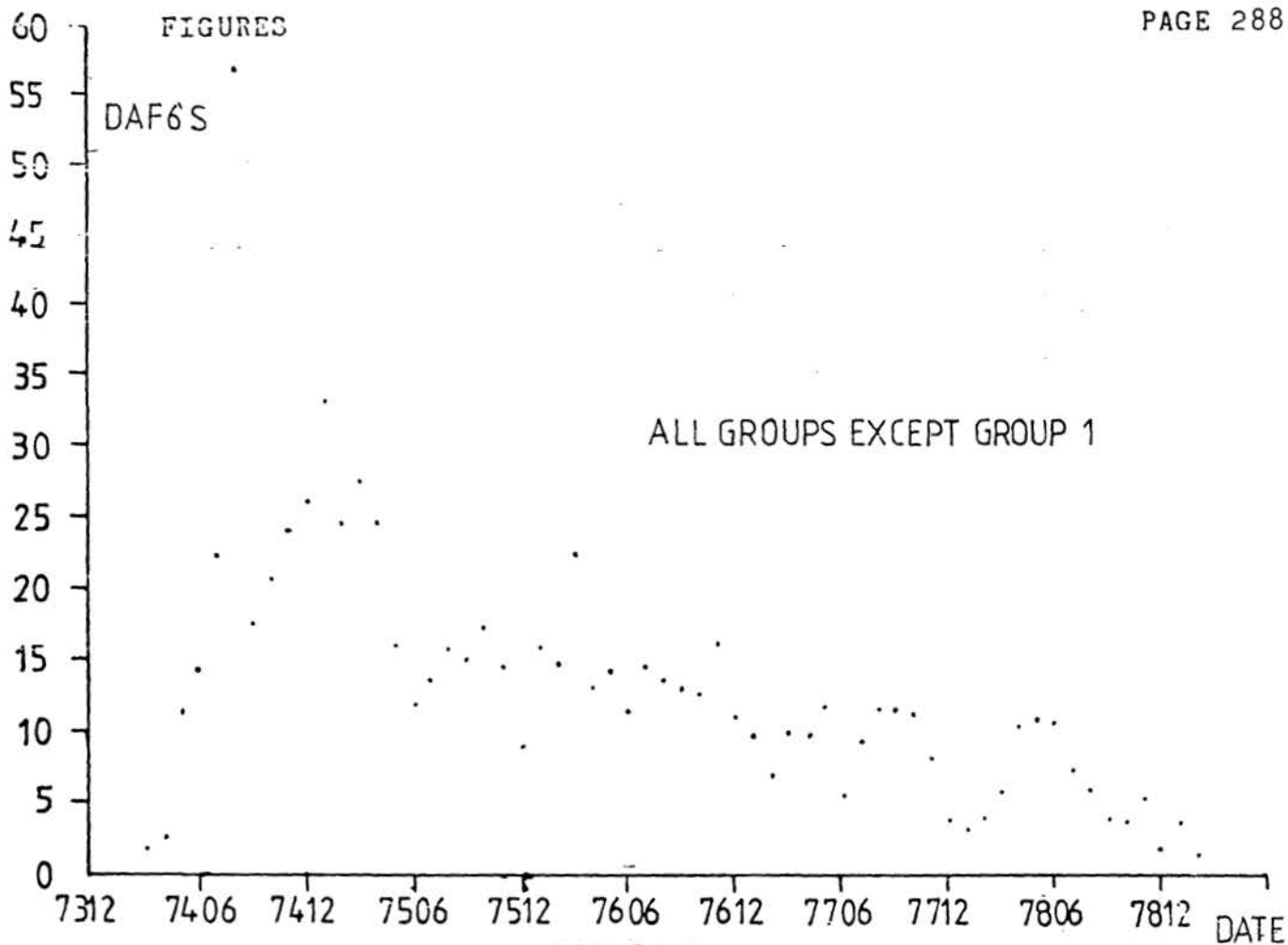


FIGURE 59

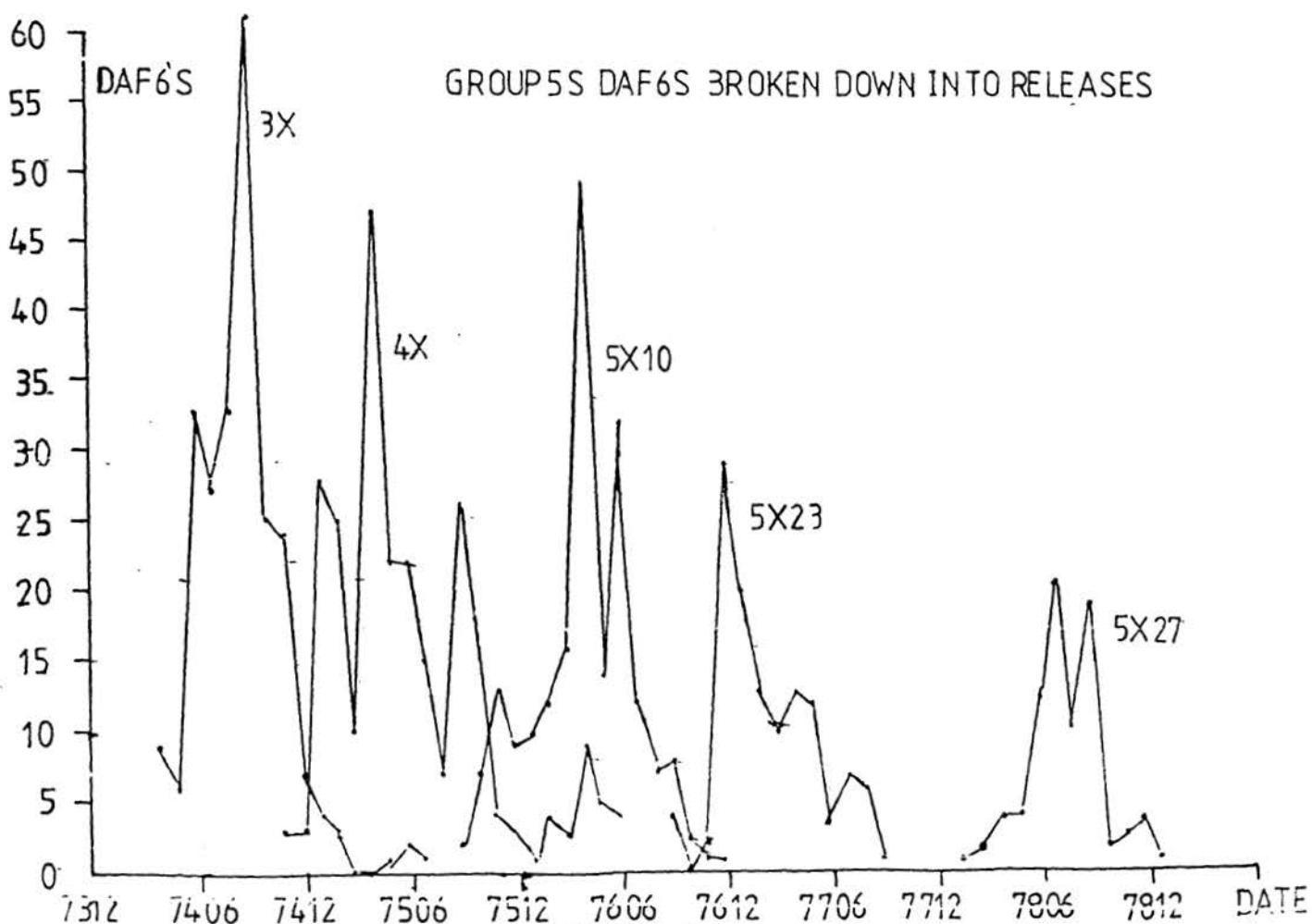


FIGURE 60

- [1] AKIYAMA, F., An Example of Software System Debugging, Proceedings of IFIP, 1971.
- [2] ALEXANDER, C., Notes on the Synthesis of Form, Harvard University Press, Cambridge, Massachusetts, 1964.
- [3] ANDERBERG, M.R., Cluster Analysis for Applications, Academic Press, New York, 1973.
- [4] ANDERSON, H.A. & SARGENT, R.G., A Statistical Evaluation of the Scheduler of an Experimental Interactive Computing System, in Freiburger, W., (Ed.), Statistical Computer Performance Evaluation, Academic Press, New York, 1972.
- [5] ARON, J.D., Estimating Resources for Large Programming Systems, In Buxton, J.N. & Randell, B., (Eds.), Software Engineering Techniques, April 1970.
- [6] ARON, J.D., The Program Development Process, Addison-Wesley, Reading, Massachusetts, 1974.
- [7] BAKER, P.T., System Quality Through Structured Programming, Proceedings FJCC, 1972.
- [8] BASILI, V.R. & TURNER, A.J., Iterative Enhancement: A Practical Technique for Software Development, IEEE Transactions on Software Engineering, Vol. SE-1, No. 4, December 1975.
- [9] BASILI, V.R. & ZELKOWITZ, M.V., Analysing Medium Scale Software Development, Proceedings of the 3rd International Conference on Software Engineering, Atlanta, Georgia, May 10-12, 1978.
- [10] BASILI, V.R. & ZELKOWITZ, M.V., Operation on the Software Engineering Laboratory, 2nd Software Life Cycle Management Workshop, Atlanta, Georgia, August 21-22, 1978.
- [11] BAUMOL, W.J., Macro-Economics of Unbalanced Growth, American Economic Review, June 1967.
- [12] BELADY, L.A., Session Summary, Measures and Empirical Studies, 2nd Software Life Cycle Management Workshop, Atlanta, Georgia, August 21-22, 1978.
- [13] BELADY, L.A. & LEHMAN, M.M., Programming Systems Dynamics or the Meta-Dynamics of Systems in Maintenance and Growth, IBM Research Report, RC 3546, September 17, 1971.
- [14] BELADY, L.A. & LEHMAN, M.M., A Case Study in Programming Systems Growth Dynamics, IBM Research Report, RC 3721, February, 1972.

- [15] BELADY, L.A. & LEHMAN, M.M., An Introduction to Growth Dynamics, in Freiburger, W., (Ed.), Statistical Computer Performance Evaluation, Academic Press, New York, 1972.
- [16] BELADY, L.A. & LEHMAN, M.M., A Systems Viewpoint of Programming Projects, Imperial College, Department of Computing and Control, Research Report 72/31, 1972. Also Published in Advances in Cybernetics and Systems, I, Gordon & Breach, London, 1975.
- [17] BELADY, L.A. & LEHMAN, M.M., The Evolution Dynamics of Large Programs, IBM Research Report, RC 5615, September 1975.
- [18] BELADY, L.A. & LEHMAN, M.M., A Model of Large Program Development, IBM Systems Journal, Vol. 15, No. 3, 1976.
- [19] BELADY, L.A. & LEHMAN, M.M., Large Programs and Large Scale Programming, Infotech State of the Art Report, 37, Software Reliability, Infotech International Ltd., Maidenhead, Berks, UK, 1977.
- [20] BELADY, L.A. & LEHMAN, M.M., The Characteristics of Large Systems, in Wegner, P., (Ed.), Research Directions in Software Technology, MIT Press, 1979.
- [21] BELFORD, P.C., BERG, R.A. & HANNAN, T.L., Central Flow Control Software Development: A Case Study of the Effectiveness of Software Engineering Techniques, Proceedings of the 4th International Conference on Software Engineering, Munich, September 17-19, 1979.
- [22] BELL, T.E. & THAYER, T.A., Software Requirements: Are They Really a Problem?, Proceedings of the 2nd International Conference on Software Engineering, October 1976.
- [23] BENYON-TINKER, G. & LEHMAN, M.M., Complexity and Complexity Change in a Large Applications Program, Research Report, Department of Computing and Control, Imperial College, April 1979.
- [24] BERSTEIN & CETRON, A Delphic Approach to Information Processing, Technological Forecasting, Vol 1, No. 1, June 1969.
- [25] BLOOMFIELD, P., Fourier Analysis of Time Series: An Introduction, Wiley, London, 1976.
- [26] BOEHM, B.W., Software and Its Impact: A Quantitative Assessment, Datamation, May 1973.
- [27] BOEHM, B.W., et al., Structured Programming: A Quantitative Assessment, IEEE Comp., June 1975.

- [28] BOEHM, B.W., BROWN, J. (Jr.), KASPAR, H., LIPOW, M., MACCLEOD, G.J. & MERRIT, M.J., Characteristics of Software Quality, TRW Series of Software Technology, North Holland, 1978.
- [29] BOX, G.E.P. & JENKINS, G.M., Time Series Analysis: Forecasting and Control, Holden-Day, San Francisco, 1970.
- [30] BROOKS, F.P., The Mythical Man-Month: Essays on Software Engineering, Addison-Wesley, Reading, Massachusetts, 1975.
- [31] BROWN, J.R. & LIPOW, M., The Quantitative Measurement of Software Safety and Reliability, TRW Report SDP-1776, 1973.
- [32] BRUGGERE, T.H., Software Engineering: Management, Personnel and Methodology, Proceedings of the 4th International Conference on Software Engineering, Munich, September 17-19, 1979.
- [33] BUCKLE, J.K., Managing Software Projects, MacDonald Computer Monographs, New York, 1977.
- [34] BUXTON, J.N. & RANDELL, B., (Eds.), Software Engineering Techniques. Report on a Conference, Rome, October 1969.
- [35] CHATFIELD, C., The Analysis of Time Series: Theory and Practice, Chapman-Hall, London, 1975.
- [36] CHONG HOK YUEN, C.K.S., Evolution Dynamics and its Application to Two Large Programs, B Sc Project Report, Department of Computing and Control, Imperial College, June 1977.
- [37] CHONG HOK YUEN, C.K.S., BD Study: Progress Report 1 (PR1), Internal Report, Department of Computing and Control, Imperial College, 1978.
- [38] CHONG HOK YUEN, C.K.S., BD Study: Progress Report 2 (PR2), Internal Report, Department of Computing and Control, Imperial College, 1978.
- [39] CHONG HOK YUEN, C.K.S., BD Study: Progress Report 3 (PR3), Internal Report, Department of Computing and Control, Imperial College, 1978.
- [40] CHONG HOK YUEN, C.K.S., BD Study: Progress Report 4 (PR4), Internal Report, Department of Computing and Control, Imperial College, 1978.
- [41] CHONG HOK YUEN, C.K.S., BD Study: Progress Report 5 (PR5), Internal Report, Department of Computing and Control, Imperial College, 1978.

- [42] CHONG HOK YUEN, C.K.S., BD Study: Progress Report 6 (PR6), Internal Report, Department of Computing and Control, Imperial College, 1978.
- [43] CHONG HOK YUEN, C.K.S., BD Study: Progress Report 7 (PR7), Internal Report, Department of Computing and Control, Imperial College, 1978.
- [44] CHONG HOK YUEN, C.K.S., BD Study: Progress Report 8 (PR8), Internal Report, Department of Computing and Control, Imperial College, 1978.
- [45] CHONG HOK YUEN, C.K.S., BD Study: Progress Report 9 (PR9), Internal Report, Department of Computing and Control, Imperial College, 1978.
- [46] CHONG HOK YUEN, C.K.S., VME/B Study: Progress Report 1 (PR1), Internal Report, Department of Computing and Control, Imperial College, 1979.
- [47] CHONG HOK YUEN, C.K.S., VME/B Study: Progress Report 2 (PR2), Internal Report, Department of Computing and Control, Imperial College, 1979.
- [48] CHONG HOK YUEN, C.K.S., VME/B Study: Progress Report 3 (PR3), Internal Report, Department of Computing and Control, Imperial College, 1979.
- [49] CHONG HOK YUEN, C.K.S., VME/B Study: Progress Report 4 (PR4), Internal Report, Department of Computing and Control, Imperial College, 1979.
- [50] CHONG HOK YUEN, C.K.S., VME/B Study: Progress Report 5 (PR5), Internal Report, Department of Computing and Control, Imperial College, 1980.
- [51] CLARE, C.P., A Case Study in Operating System Evolution Dynamics, M Sc Thesis, Department of Computing and Control, Imperial College, September, 1978.
- [52] CLIFF, A.D. & ORD, J.K., Spatial Autocorrelation, Plon, London, 1973.
- [53] CLIFFORD, H.T. & STEPHENSON, W., An Introduction to Numerical Classification, Academic Press, London, 1975.
- [54] DALKEY, N. & HELMER, O., An Experimental Application of the Delphi Method to the Use of Experts, Management Science, 1963.
- [55] DALY, E.B., The Management of Software Development, IEEE

Transactions on Software Engineering, Vol. SE-3, No. 3, May 1977.

[56] DRAPER, N. & SMITH, H., Applied Regression Analysis, John Wiley, New York, 1966.

[57] ELSHOFF, J.L., An Analysis of Some Commercial PL/I Programs, IEEE Transactions on Software Engineering, Vol. SE-2, No. 2, June 1976.

[58] EMERY, F.E. & TRIST, E.L., Socio-technical Systems, In Churchman, C.W. & Verhulst, M., (Eds.), Management Science, Models and Techniques, Volume 2, Pergamon, 1960.

[59] ENDRES, A., An Analysis of Errors and Their Causes in System Programs, IEEE Transactions on Software Engineering, Vol. SE-1, No. 2, June 1975.

[60] EVERITT, B., Cluster Analysis, London, 1974.

[61] FEUER, A.R. & FOWLKES, E.B., Some Results from an Empirical Study of Computer Software, Proceedings of the 4th International Conference on Software Engineering, Munich, September 17-19, 1979.

[62] FORRESTER, J.W., Industrial Dynamics, MIT Press, 1961.

[63] FREIBERGER, W., (Ed.), Statistical Computer Performance Evaluation, Academic Press, New York, 1972.

[64] GANNON, J.D. & HORNING, J.J., The Impact of Language Design on the Production of Reliable Software, Proceedings of the International Conference on Reliable Software, April 1975.

[65] GILB, T., Software Metrics, Winthrop, Cambridge, Massachusetts, 1977.

[66] GNEDENKO, B.V. & KOLMOGOROV, A.N., Limit Distributions for Sums of Independent Variables, Addison-Wesley, Reading, Massachusetts, 1964.

[67] GODFREY, M.D., Computational Methods for Time Series, Bult. IMA, 10, 1974.

[68] HALSTEAD, M., Elements of Software Science, MacDonald-Elsevier, 1977.

[69] HANEY, F.M., Module Connection Analysis - A Tool for Scheduling Software Debugging Activities, Proceedings of the FJCC, 41, 1972.

[70] HERNDON, M.A. & LANE, J.A., An Approach to the

Quantification of Software Errors as a Function of Module Complexity, Proceedings of 1977 Hawaii International Conference on System Sciences, Vol. 10, Western Periodicals Company, Los Angeles, CA, 1977.

[71] HERNDON, M.A. & LANE, J.A., Analysis of Software Errors for Cost Factors, Proceedings of 1977 AIAA Computers in Aerospace Conference, Los Angeles, CA.

[72] HOGAN, T., A Case Study in Evolution Dynamics, B Sc Project Report, Department of Computing and Control, Imperial College, June 1978.

[73] HOOTON, D.H., A Case Study in Evolution Dynamics, M Sc Thesis, Department of Computing and Control, Imperial College of Science and Technology, September 1975.

[74] HOROWITZ, E., (Ed.), Practical Strategies for Developing Large Software Systems, Addison-Wesley, Reading, Massachusetts, May 1975.

[75] HUTCHINGS, A.F., MCGUFFIN, R.W., ELLISTON, A.E., TRANTER, B.R. & WESTMACOTT, P.N., CADES - Software Engineering in Practice, Proceedings of the 4th International Conference on Software Engineering, Munich, September 1979.

[76] JACKSON, M.A., Principles of Program Design, Academic Press, 1975.

[77] JEFFERY, D.R. & LAURENCE, M.J., An Inter-Organisational Comparison of Programming Productivity, Proceedings of the 4th International Conference on Software Engineering, Munich, September 1979.

[78] JELINSKI, Z. & MORANDA, P.B., Software Reliability Research, In Freiburger, W., (Ed.), Statistical Computer Performance Evaluation, Academic Press, New York, 1972.

[79] JENKINS, G.M. & WATTS, D.G., Spectral Analysis and its Applications, Holden-Day, San Francisco, 1968.

[80] JEVONS, W.S., Elementary Lessons in Logic, MacMillan, London, 1870.

[81] JOHNSTON, J., Econometric Methods, 2nd ed., McGraw-Hill, New York, 1972.

[82] JONCKHEERE, A.R., A Distribution-Free K-Sample Test Against Ordered Alternatives, Biometrika, 41, 1954.

[83] JONES, J.C., Continuous Design and Redesign, 67th Infotech State of the Art Conference, Long Life Software,

London, 10-12 December, 1979.

[84] KAC, M., Some Mathematical Models in Science, Science, Volume 166, November 1969.

[85] KATZ, D. & KAHN, R.L., The Social Psychology of Organisations, Wiley, 1966.

[86] KENDALL, M.G., Time Series, Griffin, London, 1973.

[87] KENDALL, M.G. & STUART, A., The Advanced Theory of Statistics, Vol. 2, Griffin, London, 1966.

[88] KENDALL, M.G. & STUART, A., The Advanced Theory of Statistics, 2nd Ed., Vol. 3, Griffin, London, 1966.

[89] KOBAYASHI, H., Modelling and Analysis, An Introduction to System Performance Evaluation Methodology, Addison-Wesley, Reading, Massachusetts, 1978.

[90] KNUTH, D.E., An Empirical Study of Fortran Programs, Software Practice and Experience, Vol. 1, No. 2, 1971.

[91] LECHT, C.P., The Management of Computer Programming Projects, American Management Association, New York, 1967.

[92] LEE, R.C.T., SLAGLE, J.R. & MONG, C.T., Application of Clustering to Estimate Missing Data and Improve Data Integrity, Proceedings of the 2nd International Conference on Software Engineering, San Francisco, October, 1976.

[93] LEHMAN, M.M., The Programming Process, IBM Research Report, RC 2722, December 5, 1969.

[94] LEHMAN, M.M., Programming Systems Growth Dynamics, Infotech State of the Art, No. 18, Lecture on Computer Measurement and Analysis Techniques, London, June 1972, (Published in Computer Reliability, No. 20, 1974).

[95] LEHMAN, M.M., Programs, Cities, Students - Limits to Growth?, Inaugural Lecture, 14 May 1974, Published in Imperial College of Science and Technology Inaugural Lecture Series, Vol. 9, 1970-1974.

[96] LEHMAN, M.M., Human Thought and Action as an Ingredient of System Behaviour, In Duncan, R. & Weston-Smith, M., The Encyclopedia of Ignorance, Pergamon Press, London, 1977.

[97] LEHMAN, M.M., Evolution Dynamics, A Phenomenology of Software Maintenance, Software Life Cycle Management Workshop, AirTie, August 1977.

- [98] LEHMAN, M.M., Laws of Program Evolution - Rules and Tools for Programming Management, Infotech State of the Art Conference, Why Software Projects Fail, 10-12 April 1978.
- [99] LEHMAN, M.M., Systems Dynamics, Infotech State of the Art Tutorial, Why Software Projects Fail, 13-14 April 1978.
- [100] LEHMAN, M.M., The Software Engineering Environment, ISES Steering Committee, Corkstown, Ottawa, 31 August 1978.
- [101] LEHMAN, M.M., The Environment of Software Design methodology, Cambridge, 9-12 April 1979.
- [102] LEHMAN, M.M., The Software Engineering Environment, Infotech State of the Art, Beyond Structured Programming, 1978.
- [103] LEHMAN, M.M., Laws and Conservation in Large Program Evolution, 2nd Software Life Cycle Management Workshop, Atlanta, Georgia, August 1978.
- [104] LEHMAN, M.M., On Understanding Laws, Evolution and Conservation in the Large Program Life Cycle, 67th Infotech State of the Art Conference, Long Life Software, December 1979.
- [105] LEHMAN, M.M., Characteristics of Large Programs, October 1977.
- [106] LEHMAN, M.M., Measures of Systems Evolution and of the Programming Process, Infotech State of the Art Tutorial, System Dynamics, 12-13 April 1978, Lecture No. 3.
- [107] LEHMAN, M.M., Programs, Programming and the Software Life Cycle, Proc. IEEE Special Issue on Software Engineering, to be published.
- [108] LEHMAN, M.M. & PARR, F.N., Program Evolution Dynamics and its Role in Software Engineering and Project Management, Eurocomp Conference, London, September 1976.
- [109] LEHMAN, M.M. & PARR, F.N., Program Evolution and its Impact on Software Engineering, Proceedings of the 2nd International Conference on Software Engineering, San Francisco, October 1976.
- [110] LEHMAN, M.M. & PATTERSON, J.K., Preliminary CCSS Systems Analysis, Software Life Cycle Management Workshop, Airlie, August 1977.
- [111] LEINTZ, B.P. & SWANSON, E.B., Software Maintenance: A User/Management Tug-Of-War, Data Management, April 1979.
- [112] LEONTIEF, W., Theoretical Assumptions and Non Observed

Facts, American Economic Review, Vol. 61, March 1971.

[113] LIM, A.L., Preface to ED Analysis, Working Document 4, IBM, April 1975.

[114] LIPOW, M., Estimation of Software Package Residual Errors, TRW-SS-72-09, TRW Systems Group, Redondo Beach, CA, November 1972.

[115] LITECKY, C.R. & DAVIS, G.B., A Study of Errors, Error-proneness, and Error Diagnosis in COBOL, CACM, Vol. 19, No. 1, January 1976.

[116] LITTLEWOOD, B., A Reliability Model for Markov Structured Software, Proceedings ICRS, April 1975.

[117] LITTLEWOOD, B., How to Measure Software Reliability and How Not to..., Proceedings of the 3rd International Conference on Software Engineering, Atlanta, Georgia, May 1978.

[118] LITTLEWOOD, B., Validation of a Software Reliability Model, 2nd Software Life Cycle Management Workshop, Atlanta, Georgia, August 21-22, 1978.

[119] LITTLEWOOD, B. & VERRALL, J.L., A Bayesian Reliability Growth Model For Computer Software, IEEE Symposium Comp. Software Reliability, April 1973.

[120] MACCABE, T., A Program Complexity Measure, Proceedings of the 2nd International Conference on Software Engineering, October 1976.

[121] MACWILLIAMS, W.H., Reliability of Large Real-Time Control Software Systems, Proceedings of a Symposium on Software Reliability, 1973.

[122] MCGEACHIE, J.S., Reliability of the Dartmouth Time Sharing System, IEEE Symposium on Software Reliability, 1973.

[123] METZGER, P.W., Managing a Programming Project, Prentice-Hall, Englewood Cliffs, NJ, 1973.

[124] MILLER, G.A., The Magic Number 7, Plus or Minus 2: Some Limits on Our Capacity for Processing Information, Psychological Review, 63, 1956.

[125] MILLS, H., The Complexity of Programs, in Hetzel, W.C. (Ed.), Program Test Methods, Prentice-Hall, Englewood Cliffs, 1973.

[126] MIYAMOTO, I., Software Reliability in On Line Real Time Environment, Proceedings of the International Conference on

Reliable Software, April 1975.

[127] MORISSEY, J.H. & WU, L.S.Y., Software Engineering, An Economic Perspective, Proceedings of the 4th International Conference on Software Engineering, Munich, Germany, September 17-19, 1979.

[128] MORRISON, D.F., Multivariate Statistical Methods, 2nd ed., McGraw-Hill, New York, 1976.

[129] MUSA, J.D., A Theory of Software Reliability and its Application, IEEE Transactions on Software Engineering, Vol. SE-1, No. 3, 1975.

[130] MUSA, J.D., An Exploratory Experiment with "Foreign" Debugging of Programs, MRI Symposium on Computer Software Engineering, April 1976.

[131] MYERS, G.J., Reliable Software Through Composite Design, Petrocelli, New York, April 1975.

[132] NAUR, P. & RANDELL, B., (Eds.), Software Engineering, Report on a Conference, Sponsored by the NATO Science Committee, Garmisch, Germany, October 1968.

[133] NEEDHAM, R.M. & ARON, J.D., Software Engineering and Computer Science, In Naur, P. & Randell, B., (Eds.), Software Engineering, Report on a Conference, Sponsored by the NATO Science Committee, Garmisch, Germany, October 1968.

[134] NORDEN, P.V., Useful Tools for Project Management, In STARR, M.K., (Ed.), Management of Production, Penguin Books, 1970.

[135] NORDEN, P.V., Project Life Cycle Modelling: Background and Application of the Life Cycle Curves, in SLC77, 1977.

[136] PARNAS, D.L., On the Criteria to be Used in Decomposing Systems into Modules, CACM, Vol. 15, No. 12, December 1972.

[137] PARNAS, D.L., On the Design and Development of Program Families, IEEE Transactions on Software Engineering, March 1976.

[138] PARR, F.N., A Life Cycle Model Based on System Structure, 2nd Software Life Cycle Management Workshop, Atlanta, Georgia, August 21-22, 1978.

[139] PARSONS, T. & SHILS, E.A., Toward a General Theory of Action, Harvard University Press, Cambridge, Massachusetts, 1951.

- [140] PEARSON, D.J., CADES - Computer Aided Design and Evaluation System, Computer Weekly, July 26th, August 2nd, August 9th, 1973.
- [141] PLATT, J.R., Strong Inference, Science, 146, No. 3642, 351, 1964.
- [142] PRATTEN, G.D., The CADES Software Development System, IEEE Colloquium, 1978.
- [143] PRATTEN, G.D. & SNOWDON, R.A., Support for the Development of Complex Software, Proceedings of EUROCOMP, London, 1976.
- [144] PRATTEN, G.D. & SNOWDON, R.A., CADES - Methodology and Support System for the Development of Operating Systems, IFIP Congress, Toronto, 1977.
- [145] PUTNAM, L.H., A Macro-Estimating Methodology for Software Development, 13th IEEE Computer Society International Conference, COMPCON 1976, Washington, D.C., September 1976.
- [146] PUTNAM, L.H., A General Empirical Solution to the Macro Software Sizing and Estimating Problem, COMPSAC 77.
- [147] PUTNAM, L.H., The Influence of the Time Difficulty Factor in Large Scale Software Development, In SLC77, 1977.
- [148] PUTNAM, L.H., Session Summary, 2nd Software Life Cycle Management Workshop, Atlanta, Georgia, August 21-22, 1978.
- [149] RANDELL, B., Software Engineering in 1968, Proceedings of the 4th International Conference on Software Engineering, Munich, September 17-19, 1979.
- [150] RIORDON, J.S., An Evolution Dynamics Model, Department of Systems Engineering and Computing Science, Carleton, University of Ottawa, Canada, 1976, also Published in the Proceedings of the First Software Life Cycle Management Workshop, Airlie, August 1977.
- [151] ROYCE, W.W., Managing the Development of Large Software Systems: Concepts and Techniques, Proceedings of WESCON, August 1970.
- [152] RUBEY, R.J., Quantitative Aspects of Software Validation, Proceedings of the 1975 International Conference on Reliable Software, April 1975.
- [153] RUBEY, R.J. & HARTWICK, R.D., Quantitative Measurement of Program Quality, Proceedings of the 23rd National Conference, ACM, 1968.

- [154] SAMET, P.A., Measuring the Efficiency of Software, Proceedings SEAS XIV, Grenoble, France, 1969.
- [155] SCHICK, G.J. & WOLVERTON, R.W., Assessment of Software Reliability, 11th Annual Meeting, German OR Society, September 1972.
- [156] SCHNEIDEWIND, N.F., An Approach to Software Reliability Prediction and Quality Control, Proceedings of FJCC, 1972.
- [157] SCHUTZENBERGER, M.P., A Tentative Classification of Goal-Seeking Behaviours, Journal of Mental Science, Vol. 100, 1954.
- [158] SCOTT, R.F. & SIMMONS, D.B., Programmer Productivity and the Delphi Technique, Datamation, May 1974.
- [159] SHERKAT-KHAMENEH, S., A Study of Evolution Dynamics, B Sc Project Report, Department of Computing and Control, Imperial College, June 1979.
- [160] SHOOMAN, M.L., Probabilistic Models for Software Reliability Prediction, In Freiburger, W., (Ed.), Statistical Computer Performance Evaluation, Academic Press, New York, 1972.
- [161] SHOOMAN, M.L. & BOLSKY, M.I., Types, Distribution, and Test and Correction Times for Programming Errors, Proceedings of 1975 International Conference on Reliable Software, April 1975.
- [162] SHOOMAN, M.L., DICKSON, J.G., HESSE, J.L., & KIENTZ, A.C., Quantitative Analysis of Software Reliability, Annual Reliability and Maintainability Symposium, San Francisco, California, January 25-27, 1972.
- [163] SIEGEL, S., Non-parametric Statistics for the Behavioural Sciences, McGraw-Hill, 1956.
- [164] SIMON, H.A., Rational Choice and the Structure of the Environment, Psychological Review, Vol. 63, 1956.
- [165] SNOWDON, R.A., Experiences with CADES in the Development of VME/B, Second Hungarian Computer Science Conference, Budapest, 1977.
- [166] SOKAL, R.R. & SNEATH, P.H.A., Principles of Numerical Taxonomy, Freeman, San Francisco, California, 1963.
- [167] SOROKIN, P., Sociocultural Causality, Space and Time, Russell and Russell, New York, 1964.

- [168] STAUDHAMMER, J., Resources for Long-Life Software: Tools and Staffing, Long Life Software, Infotech, 1979.
- [169] STEPHENSON, W.E., An Analysis of Resources Used in the Safeguard System Software Development, Proceedings of the 2nd International Conference on Software Engineering, San Francisco, October 1976.
- [170] SUTHERLAND, J.W., Systems - Analysis, Administration and Architecture, Von Nostrand, New York, 1975.
- [171] THAYER, T.A., Understanding Software Through Empirical Reliability Analysis, AFIP, Vol. 44, 1975.
- [172] THAYER, T.A., Understanding Software Through Analysis of Empirical Data, Proceedings NCC, 1975.
- [173] THAYER, T.A., et al., Software Reliability Study, TRW System Report SS-76-03, 1976.
- [174] THIBODEAU, R. & DODSON, E.N., The Implications of Life Cycle Phase Interrelationships for Software Cost Estimating, 2nd Software Life Cycle Management Workshop, Atlanta, Georgia, August 21-22, 1978.
- [175] TURSKI, W., (Ed.), The Efficient Production Of Large Programs, Polish Academy of Sciences, 1970.
- [176] TURSKI, W., Look Ahead at Software Engineering, Proceedings of the 4th International Conference on Software Engineering, Munich, September 17-19, 1979.
- [177] VON BERTALANFFY, L., The Theory of Open Systems in Physics and Biology, Science, Vol. III. 1950.
- [178] WALDBAUM, G., Evaluating Computing System Changes by Means of Regression Models, Proceedings of 1st SIGME Symposium on Measurement and Evaluation, 1973.
- [179] WALSTON, C.E. & FELIX, C.P., A Method of Program Measurement and Estimation, IBM Systems Journal, Vol. 16, No. 1, 1977.
- [180] WARD, J.A., 20 Commandments for Managing the Development of Tactical Computer Programs, Proc. National Computer Conference, 1974.
- [181] WASSERMAN, A.I., Data Design for Evolving Systems, Long Life Software, Infotech, 1979.
- [182] WEINBERG, G.M., The Psychology of Computer Programming, Van Nostrand, Reinhold, New York, 1971.

- [183] WEINBERG, G.M., An Introduction to General Systems Thinking, Wiley, London, 1975.
- [184] WOLVERTON, R.W., The Cost of Developing Large-Scale Software, IEEE Transactions on Computers, June 1974.
- [185] WOLVERTON, R.W., Session Summary, 2nd Software Life Cycle Management Workshop, Atlanta, Georgia, August 21-22, 1978.
- [186] WOODSIDE, C.M., A Mathematical Model for the Evolution of Software, Department of Systems Engineering and Computer Science, Carleton, University of Ottawa, Canada, 1979.
- [187] YEH, A.C., An Application of Statistical Methodology in the Study of Computer System Performance, in Freiburger, W., (Ed.), Statistical Computer Performance Evaluation, Academic Press, London, 1972.