SYSTEMS DEVELOPMENT USING STRUCTURED TECHNIQUES

Malcolm Bull

First published in 1989 by
Chapman and Hall Ltd
11 New Fetter Lane, London EC4P 4EE
Published in the USA by
Chapman and Hall
29 West 35th Street, New York NY 10001

Typeset in 10/12 Photina by Thomson Press (India) Limited, New Delhi

Printed in Great Britain by St. Edmunsbury Press,
Bury St. Edmunds, Suffolk

ISBN 0 412 31010 4 (hardback)
0 412 31020 1 (paperback)

British Library Cataloguing in Publication Data

Bull, Malcolm, 1941-
Systems development using structured techniques.
1. Computer systems. Structured systems analysis
I. Title
004.2'1

ISBN 0-412-31010-4

Library of Congress Cataloging in Publication Data

Bull, Malcolm, 1941-
Systems development using structured techniques / Malcolm Bull.
p. cm.
Includes index.
ISBN 0-412-31010-4.—ISBN 0-412-31020-1 (pbk.)
1. System design. 2. Electronic data processing—Structured techniques. I. Title.
QA76.9.S88B85 1989
004.2'1—dc20 89-15883
CIP

Acknowledgements

A number of people have contributed directly or indirectly to the conception, content and final presentation of this book.

Those who have attended my training courses on any of a number of subjects – systems analysis, systems design, file design, programming or project management – will recognize much of my arguments and lecture material here. Teaching is never a one-way flow, and I must thank all my students, past and present, for their questions and their questioning. Their needs led to the conception of the book.

Special thanks are due to my editor, Elizabeth Johnston of Chapman and Hall, for the unfailing encouragement with which she spurred me on to produce the book you see here. Her ideas and suggestions were helpful throughout and her, albeit daunting, list of changes, and her persistence in implementing them, reduced my original manuscript from a wordy tome on system design theory down to the manageable and, let us hope, practical form which you see here.

My thanks also to Kay Birtles of McDonnell Douglas for her help with PRO IV in the section on fourth generation languages.

Finally, I should like to thank Paul Anstee for his tireless reading, re-reading and proof-reading of the text in its various manifestations.

If, after all this support, there are any omissions or any errors remaining in the book, then the responsibility and blame for these can only be laid at my own door.

Malcolm Bull
Lusaka

Preface

The principle aim of this book is to introduce and illustrate the structured techniques which are available for use during the development of a computer system.

When the book was conceived, it was the author's intention to direct the discussion at those readers who are experienced project leaders, systems analysts and programmers and who, being familiar with traditional methods, now wish to move on to more structured techniques. However, discussion with potential readers – and especially those who have come into systems development via other disciplines – suggested that this scope be widened. As a result, the principles of systems investigation are also considered. The book is suitable for anyone who may be new to systems analysis, including programmers who are making the transition to analysis. It is especially suitable for end-users who are increasingly having to work with systems anaysts practising structured techniques, and who, with the growth in application-oriented tools, and the increasing accessibility of computer systems, now need to play a greater role in the design and development of the computer systems.

Structured techniques were developed for use in the commercial environment, and the book is particularly appropriate in such areas, whether it be used for private study or as a part of a formal training course. It is also suitable for use in the academic setting as supplementary reading for students following Computer Studies and Computer Science courses at schools and colleges, and for those studying first year Information Technology and other computer subjects at university. It will adequately prepare anyone leaving such a course for a career in the commercial computing environment.

Since structured techniques were introduced, the choice of available techniques and methodologies has widened. The present book does not aim to increase this range. Instead, we have tried to strip off much of the jargon and cant which may be encountered, revealing the reasons and the reasoning behind the methodologies. This produces a systems development approach which is valid as an independent methodology and which will also prepare the reader for any more detailed methodology which may be implemented in his or her own working environment.

The book is divided into eight chapters with two interludes:

Chapter 1 introduces the concepts of structured techniques.

Chapter 2 looks at the techniques involved during the preliminary stages of systems investigation.

Chapter 3 looks at the tools and techniques which are available for systems analysis.

Interlude 1 considers the intermediate stage before the systems design phase.

Chapter 4 looks at the tools and techniques which are available for the data analysis stage of the systems design phase.

Chapter 5 looks at the tools and techniques which are available for the functional analysis stage of the systems design phase.

Chapter 6 discusses structured programming and modular programming, and looks at the ways in which these interact with the systems design phase.

Interlude 2 discusses the final stages and we consider the future life of a computer system.

Chapter 7 looks at some of the ways in which systems development is supported by automated tools and techniques. Within this chapter, the reader will find an introduction to a number of the newer aspects of data processing, such as prototyping, database management systems, query languages and fourth generation languages.

Chapter 8 comprises a number of practical projects. The first one is worked and illustrates the points discussed in the main text. The remaining case studies are provided to allow the readers to develop their own solutions.

Finally, there are appendices comprising a short bibliography and a glossary of some of the terms that the reader may encounter in the present book and in the literature of the subject.

Within the audience mentioned above, it is anticipated that:

- project managers will find all sections of interest
- trainee systems analysts and trainee programmers will find all sections of interest
- systems analysts could probably omit Chapter 2 and Interlude 2, but those who are unfamiliar with structured techniques may find the remaining sections of interest
- programmers will find Chapters 4, 5 and 6 of interest
- experienced programmers may wish to omit Chapter 6
- end-users will find all sections of interest if they omit the practical work and the detailed examples, and if they skim those areas which they feel are too technical for them.

For each of the major elements of the analysis and design phases, there is a separate section with a title such as:

Producing the data flow diagram

These sections provide a summary of the guidelines for the preparation of the various tools. Examples of all the documents which are covered in these sections

can found in Chapter 8 where we produce the solution to our Case Study.

The book is intended to be practical, and for this reason, many of the arguments for and against the use of structured techniques have been left to more critical authors. With this same intention, the various topics are illustrated by their application to a simple case study. The reader should bear in mind that there is no definitive answer – no right or wrong solution – to systems development problems. In offering a worked case study, we are not implying that ours are the best possible answers – for in data processing, there are only solutions and better solutions and only our fictitious user could arbitrate here. But if we can stimulate the reader to find a better solution, then we have achieved our objective – that of getting the analyst and programmer to use the structured techniques.

Malcolm Bull

List of acronyms

CAD	Computed Aided Design
CAP	Computer Aided Programming
CASA	Computer Aided Systems Analysis
CASD	Computer Aided Systems Design
DBA	Database Administrator
DBMS	Database Management System
DBTG	Data Base Task Group
DDL	Data Description Language
DFD	Data Flow Diagram
DML	Data Manipulation Language
DocFD	Document Flow Diagram
DocFM	Document Flow Matrix
DSD	Data Structure Diagram
1GL	First Generation Language
2GL	Second Generation Language
3GL	Third Generation Language
4GL	Fourth Generation Language
5GL	Fifth Generation Language
1NF	First Normal Form (also FNF)
2NF	Second Normal Form (also SNF)
3NF	Third Normal Form (also TNF)
SSA	Structured Systems Analysis

Contents

1

Introduction to structured techniques

General systems theory (GST) applies scientific principles and methods to the study of systems of all kinds, and is able to predict and determine the short-term and the long-term behaviour of systems. The scope of this behaviour includes its day-to-day and its year-to-year operation, its reactions to the outside world and to other systems, and the ways in which it changes, dies and grows. We can define a system as follows:

> **System:** an organized collection of parts, the sub-systems, which are linked together so as to function as a unit with a common purpose or objective.

In the present context, we are concerned with business systems, management information systems and computer systems. These, like any other, comprise a number of smaller sub-systems, each of which performs a specific purpose. The operation of the system as a whole is determined and affected by the interaction, the inter-relation, the co-ordination and the co-operation of these various sub-systems. If we take a steel manufacturing company as an example, then systems theory concerns itself with the company's short-term and long-term behaviour, and addresses questions such as:

1. How does the company utilize its resources and how does it react to the outside world – what happens when the world price of steel falls or when the national economy goes up or down?
2. How does its own sub-systems interact – how is the production affected by a shortage of workers, and how is the price affected by the production?

All systems – and sub-systems – are based upon five fundamental activities: input, storage, processing, control and output. Within a system, the output from one sub-system will often be the input to another. The amount of its resources which a system expends on reacting to internal and external stimuli depends upon the particular system. A **closed system** may interact little with the outside world, if at all, so there will be minimal input and output in such a case. This may be because the system is self-sufficient, such as a large country which is able to support itself completely and independently of all other nations.

A business system interacts with the social, commercial and technical facets of its environment and is said to be an **open system.**

It is the systems analyst's task to study a business system or sub-system. The result of the study will be a model, a representation of the current system – with all its good and bad features. This model will then be used as a basis for the design of a new system. This new system may well be implemented as a computer system, and for the purposes of the present book, we shall assume that this is so.

1.1 SYSTEMS DEVELOPMENT

For many years, systems analysis, systems design and programming have been taught in what we might call the traditional manner, with the trainee learning the principles and skills required to perform the various tasks in the development of a computer project. Fig. 1.1 shows the traditional stages of systems development.

In the real world of computers and data processing, it has long been apparent that being able to perform this sequence of operations was not sufficient. The process was not flexible enough. If the users change their minds during the investigation and analysis stages, then the analyst can accommodate these alterations fairly easily. However, when a project is just beginning, the business ideas are new to the analyst and the computer ideas are new to the user. The future system is quite likely to be based upon the ideas established at this primitive stage, and for this reason these ideas will not be as grand or as comprehensive as they might be.

Furthermore, the existing methods of systems analysis were proving to be too slow, and the time-scales involved meant that, whilst the analyst was producing and correcting the system specification, the users were quite likely to have changed their minds about what they wanted from the system. This may have come about as a result of incomplete understanding about what the system could have done for them, or there may have been organizational and other changes which must be reflected in the computer system.

Frequently, this came about as a result of information and possibilities which the user discovered during his discussions with the analyst, or as a result of pursuing topics raised during such discussions. How many times will the analyst hear the user's cry:

'I didn't know that you could do that.'
'You didn't tell me that I could have had that in my system.'
'Smith-Watsons have got a nice facility ... and I wondered if ...'

The user was then faced with the dilemma of either not having the new system do what was really required, or of expending still more money and time in order for the system design to be modified and the changes to be implemented.

It has been estimated that if users change their mind whilst the systems analyst is carrying out the investigation and it costs £1 to make that change, then the same change will cost £10 if it is made when the systems analysis is being performed, £100 if it is made when the systems design phase is being carried out, and so on, rising to a cost of £100 000 to make exactly the same change after the

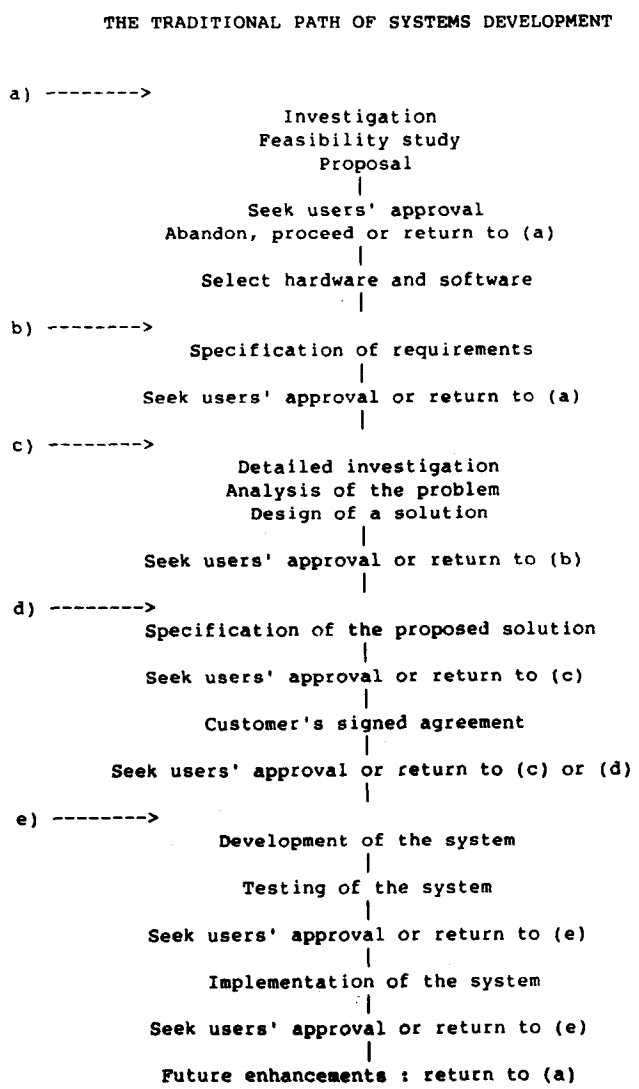

Figure 1.1 The traditional path of systems development.

system has been finally implemented. Obviously, such figures are very approximate, but they do emphasize the escalating costs of making a change later – rather than earlier – in the development cycle.

Even if it is designed and implemented with few modifications, almost every system will be subject to some changes after implementation. We have seen that changes are expensive, and they also have the disadvantage that the associated

documentation must be changed to incorporate these changes: the analyst must change the specifications, the programmer must amend the programs, the user documentation must be corrected, and the users may have to be re-trained. In practice, these were not always done adequately.

What was needed was a methodology – a way of analysing, designing and producing systems – a methodology which:

1. would be more responsive to the users' needs
2. would enable systems to be developed more quickly

and if that methodology would also enable systems to be developed more cheaply, and ease the task of system and program maintenance and be easy to learn and apply, then so much the better.

The solutions to some of the problems came in the guise of **structured techniques**. The techniques include **structured systems analysis** and **design** for the analyst, and **structured programming** for the programmer. In this book, we shall present these topics and show how they achieve the goals of modern systems analysis and design. We shall also see what impact these and other techniques have upon the programmer.

The interest in structured techniques is such that several methods and styles have been produced, each with its own advantages and disadvantages. The trainee analyst will meet, amongst others, those of: Codd, DeMarco, Dijkstra, Jackson, Parnas, Warnier and Orr, and Wirth. There are also a large number of acronyms, e.g.:

- HIPO – Hierarchy plus Input Processing and Output
- SASD – Yourdon's Structured Analysis and Structured Design
- SMDM – Systems Modelling Development Method
- SSADM – the structured systems analysis and design methodology which is recommended for use on administrative computer projects undertaken by the British Government
- STRADIS – Structured Analysis, Design and Implementation of Information Systems, a McDonnell Douglas product developed with Gane and Sarson.

The concepts of **software engineering** and **information engineering** have emerged. These are concerned with the application of structured techniques to the production of reliable and maintainable software. A very close parallel can be recognized with the principles which we shall consider here and the processes of design, development, prototyping, production, quality assurance and maintenance which are practised in the field of mechanical engineering.

A number of software packages and computer aided software engineering (CASE) tools will be encountered, such as the PROKIT software and STRADIS/DRAW, the McDonnell Douglas Information Systems graphics software for the Gane and Sarson STRADIS methodology, and the DATAMATE, AUTOMATE, AUTOMATE PLUS and SUPER-MATE software from Learmonth and Burchett Management Systems plc (LBMS).

1.2 OVERVIEW OF STRUCTURED SYSTEMS ANALYSIS

The techniques of systems analysis and design break the analyst's work down into a number of discrete phases. The first phase – systems analysis – requires us to look at the existing system which is to be replaced or modified, or to derive the requirements for a completely new system. A number of tools are available for this:

1. **Document flow diagrams** – these depict the key documents which are associated with the existing system, their source and destination, the flow and the information which they carry;
2. **Data flow diagrams** – these depict the flow of data around the system, the sources and destination of data, the places in which data are stored, and the processes which handle those data;
3. **Data structure diagrams** – these depict the entities about which data are to be held and processed, and show the relationship between these entities;
4. **Problems and requirement list** – this records the shortcomings of the present system, including problems recognized by the user and the analyst and requirements stated by the user. The ultimate solution should endeavour to address all these.

These documents will represent a model of the features of the existing system, whether it be a manual system or a computer-based system.

Like the other documents which we shall produce, these are quite suitable for presentation to the users and allow the results of the analysis to be continually reviewed to ensure that they reflect the true situation.

1.3 OVERVIEW OF STRUCTURED SYSTEMS DESIGN

Before proceeding to the systems design phase, the analyst must stand back and appraise the current system. The first step is to identify the ways in which the existing system could be reorganized so as to achieve some, or all, of the goals established in the problems and requirements list. Up to this point, no decision has been made as to the nature of the future system – indeed, it may not be necessary to implement a computer solution at all if we can reorganize the current working methods and practices.

The analyst will then identify a number of possible solutions which may be provided to achieve the required system. These are presented to the user in a manner which will emphasize the cost effectiveness and the benefits to be gained by the proposed solutions.

If a decision is made to go ahead with a computer-based solution, then the user will choose one – or a combination – of the possible solutions, and the analyst will proceed to design the proposed system around these solutions.

The tools which are available for the systems design phase will describe the features of the proposed system.

1. **Data flow diagrams** – for the proposed system
2. **Data normalization** – this enables the analyst to determine the nature and contents of the data and the physical files which will be used by the system, such that they serve the system efficiently
3. **Data structure diagrams** – for the proposed system based upon the results of the data normalization task
4. **Entity descriptions** – these list the data items contained within each entity or physical file
5. **Entity/function matrix** – this is a chart in which each function in the data flow diagram is cross-referenced to each entity on the data structure diagram, to produce an entity/function matrix. In this way, the analyst can ensure the complete integrity of the data and the processing
6. **Function catalogue** – this accompanies the data flow diagrams and provides a free text description of the processing requirements
7. **Function descriptions** – these specify the detail of the processing requirements of the system and ultimately they will serve as the program specifications
8. **Function maps** – these depict the routes by which the users will access the functions of the system
9. **Module catalogue** – this combines and summarizes the various low-level routines which are shared and used by the functions of the system
10. **Module descriptions** – these give a detailed description of the processing which each module performs
11. **Module maps** – these depict the modules and the functions which use them.

These documents will represent a model of the proposed system. As before, the documents can be reviewed with the user to confirm that they reflect the required situation.

1.4 OVERVIEW OF STRUCTURED PROGRAMMING

It is in the field of programming that structured techniques have had most popular impact. Most programmers will be familiar with the concepts of **structured programming** and **modular programming**. For this reason, we shall only touch lightly upon these topics in the book in order to see how they interact with the products of the systems design phase.

We shall also look at the concept of **prototyping** as this, too, may affect the programmer.

1.5 THE TOP-DOWN APPROACH

One of the valuable features of a structured methodology is the **top-down** approach. This begins by taking a top-level view of a system. When this has been comprehended, the picture can be expanded to get a second-level, more detailed view. The new information at this level is then studied and assimilated. This

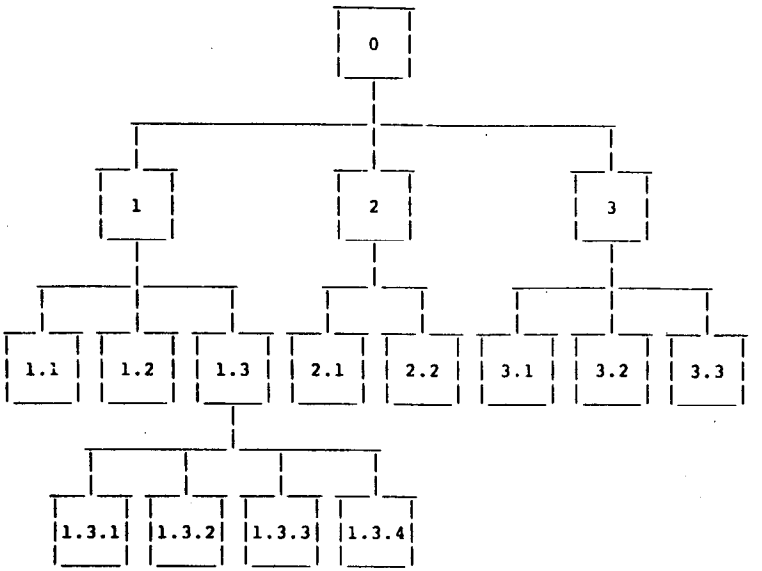

Figure 1.2 A Jackson diagram.

process of refinement is repeated until the complete system has been broken down and the view represents all the details of the system.

A useful device for depicting this process is the tree structure, or Jackson Diagram, shown in Fig. 1.2. Box 0 represents the broad picture of the system with hardly any detail, and might be the detail which you would give if your grandmother were to ask what the system does:

'Oh, it's a computer system to calculate the wages for the people at the carpet factory.'

We might then expand this to a more detailed level – Boxes 1, 2 and 3 – such as you might explain to a fellow analyst:

'They want the system to read in the clock cards, calculate the pay and make up the wages packets.'

Going down to an even-lower level – Boxes 1.1, 1.2 and so on – we go into the detail which would be needed for a programmer who is going to have to write the programs which perform the processing:

'You read in the clock card and calculate the number of hours worked that week. Then you get the personnel number from the card and read the staff record from the Staff file. Then...'

Some of the more complex parts of this – as with Box 1.3, in this illustration – may have to be expanded into even more detail.

We can represent the information in other ways: a Warnier – Orr Diagram would depict it as shown in Fig. 1.3.