Systems Design in the Fourth Generation

OBJECT-BASED DEVELOPMENT USING d®BASE

Systems Design in the Fourth Generation

OBJECT-BASED DEVELOPMENT USING d®BASE

John A. Lehman
UNIVERSITY OF ALASKA

John Wiley & Sons
NEW YORK / CHICHESTER / BRISBANE / TORONTO / SINGAPORE

Acquisitions Editor: Joe Dougherty
Designer: Laura Nicholls
Production Supervisor: Gay Nichols
Digital Production Supervisor: Ann Louise Stevens

Library of Congress Cataloging in Publication Data:

Lehman, John A
Systems design in the fourth generation using dBASE III and dBASE IV / John A Lehman

p cm —(Wiley series in business computing and information processing)
Includes index
ISBN 0-471-52752-1
1 System design 2 dBase III (Computer program) 3 dBase IV (Computer program) I Title II Series

QA76 9 S88L45 1991
005 75'65—dc20 90-48154
CIP

Printed in the United States of America

10 9 8 7 6 5 4 3 2 1

Wiley Series in Computing and Information Processing

Hugh J. Watson, University of Georgia-Athens, Series Editor

Blissmer. Introducing Computers: Concepts, Systems, and Applications, 1990-1991 Edition
Burch and Grudnitski. Information Systems: Theory and Practice, 5th
FitzGerald. Business Data Processing: Basic Concepts, Security, and Design, 3rd
FitzGerald. and FitzGerald. Fundamental Systems Analysis: Using Structured Analysis and Design Techniques, 3rd
Lehman. Systems Design in the Fourth Generation: Object-Based Programming Using dBASE
Nelson. End-User Computing: Concepts, Issues, and Applications
Panko. End-User Computing: Management, Applications, and Technology
Stern and Stern. Computing with End-User Applications
Stern and Stern. Computing with End-User Applications and BASIC
Vazsonyi. Information Systems for Management
Watson and Blackstone. Computer Simulation, 2nd
Wysocki and Young. Information Systems: Management Practices in Action
Wysocki and Young. Information systems: Management Principles in Action

ABOUT THE AUTHOR

John Lehman has worked as a user manager, an applications programmer, a systems programmer, and a software project manager. He is the author of the first article on programming languages for personal computers, as well as the first articles on business use of personal computers, and the first personal computer programs for financial analysis and statistical analysis. His doctoral dissertation was the first integrated software system, combining financial analysis, statistics, database management, and presentation graphics. He has consulted and/or taught for IBM, UNISYS (formerly Sperry and Burroughs), Control Data, Honeywell, and several noncomputer Fortune 500 companies. He has taught at the University level since 1973 and beginning in 1983 has taught program design in the American Assembly of Collegiate Schools of Business summer MIS Faculty Development Institute. He is also a Certified Management Accountant (CMA).

ACKNOWLEDGMENTS

I am grateful to many people for their contributions to the development of this book. I owe particular thanks to the following:

At the AACSB MIS Faculty Development Institute, to Gary Dickson and Chuck Hickman for giving me the chance to refine these materials on 350 faculty guinea pigs, and to the guinea pigs themselves.

At the University of Alaska, to Peter Biesiot, and Mike Rice for providing an environment where I could write 50 pages a week, and to my students for a great deal of useful feedback.

At the University of Michigan, to Jim Fry, Tom Schrieber, Dan Teichrow, and especially to Dennis Severance for inflicting me on the academic world.

At the University of Minnesota, to Gordon Davis and Sal March for extensive advice, encouragement, and assistance.

In the Real World, to Tim Fargo at UNISYS, and especially to the two successful MIS practitioners in my family: my sister Lee whose systems from London to San Francisco and from Miami to Valdez come in on time and under budget, and my wife Lisa who is a Real Programmer on Real Computers (mainframes), and only occasionally believes that IBM stands for "In Bleakest Mordor."

And finally to Don Barker, College of Business Administration, Gonzago University, Spokane, Washington; Jennifer L. Wagner, MSIS Program Director, Roosevelt University, Chicago, Illinois; Mark A. Schlesinger, College of Management, University of Massachusetts, Boston, Massachusetts; Hugh Watson, Department of Management, University of Georgia, Athens, Georgia; and Cyrus Azarbod, Computer Science Department, Mankata State University, Mankato, Minnesota for their efforts as reviewers in helping turn a raw manuscript into a much more useful text!

John A. Lehman

PREFACE

This is a book on modern programming practices as they are adapted to fourth generation languages (described in Chapter 4). They are here applied to a specific fourth generation language: dBASE-III+ and IV. This is not a book on DBASE programming. Rather, it is a book on analysis and design in a fourth generation environment using DBASE examples. Although this book may be used as a programming textbook in conjunction with the dBASE manuals, its main contribution is language-independent.

There are two audiences for this book: students of programming (primarily in business schools) and people who do programming and wish to learn more about it.

For students (and their instructors!) the book has several advantages over competing textbooks. First, it teaches modern techniques of systems analysis and program design in the context of fourth generation languages. In so doing, it combines instruction in modern systems analysis and programming practices with instruction in what is probably the most popular fourth generation language, and with instruction in business systems. Additional advantages are the use of a consistent case study (which teaches about business programming), and the testing that the book (and the teaching method that it represents) has undergone. An additional and unique aspect of this book is that it avoids the use of acronyms except where their non-use would be in blatant conflict with industry terminology.

One of the problems encountered in teaching programming is the need for potential managers and programmers to learn three things:

- How Business Information Systems Work,
- How to Design Programs,
- How to Implement Programs.

Most of us do not have the luxury of three separate courses in these areas. However, this text covers all three. The case study, which is integrated with the discussion throughout, teaches about common transaction-driven systems in business—customer orders, purchase orders, inventory management, billing, and so forth. This provides the student with a knowledge of what such information flows are like and practice in creating programs that are of immediate use. Thus, an instructor who wishes to teach programming, program design, and business systems in the same course need use only one text.

A second advantage of this text is that, unlike existing books on program design, this one modifies the use of analysis and design methods to fit in a fourth generation, database-oriented environment. Many of the methodologies developed for COBOL and file-processing environments are irrelevant, or require modification when they are used with higher-level languages. Specifically, this text integrates data modeling techniques from database design and event-driven analysis techniques. The techniques taught in this book have been proved in both personal computer and mainframe programming environments, using a variety of fourth generation languages.

The course on which this book is based has been taught at both large and small schools, at both the graduate and undergraduate level. It has also been taught to programmers and managers at a number of Fortune 500 companies. In addition, for seven years it has been taught to faculty from around North America at the American Assembly of Collegiate Schools of Business (AACSB) MIS Faculty Development Institute in Minneapolis. This text incorporates suggestions from all of these students (thank you all!)

For the other audience, that of people who are not in school but who wish to develop additional skills, probably on a personal computer, the advantages are much the same. The text covers both how to design programs and how to program in dBASE III and dBASE IV. Most of the content of the book will continue to be valid no matter what computer system one uses. In addition, the case study provides a ready-to-use system for much business programming.

John A. Lehman

CONTENTS

Section 1

INTRODUCTION AND BACKGROUND

Chapter 1

INTRODUCTION

HOW THIS BOOK CAME TO BE

This is a book about modern programming practices as applied to a specific fourth generation language—dBASE (versions III and IV). By way of introduction to the need both for modern programming practices and for fourth generation languages, let me explain how I got involved in this field.

Most readers, whether students or practitioners, are familiar with the process of registering students at a university—a somewhat specialized version of the general task of entering customer work orders. Most major universities put this operation onto punched cards in the 1930s or 1940s, and until the late 1960s, registration was generally done in some sort of batch mode.

In 1968, the university in question decided to produce an on-line registration system. Since everything in data processing needs an acronym, let us call this CRISP (Computerized Registration In Spite of Problems). The estimate in 1968 was that the system would take three years to develop, require four programmers, one manager, and cost approximately $300,000. It was also estimated that 20 operators would be required to run the system during registration.

In the early 1970s, my second job out of college was as registration supervisor. CRISP was brought on-line in 1975—four years late. Total cost to this time was around $1,500,000. A major hardware upgrade had been required, and 30 operators were now needed. Under the old batch system, registration had required about an hour. Under the new on-line system, the average wait to register was approximately nine hours. When high-level administrators showed up to see what was happening, students spat on them. The director of registration had a nervous breakdown and spent the rest of the week at a terminal checking registering students for library fines. I took his job. The system development life-cycle for CRISP is illustrated in Figure 1-1. Ironically, within 10 years the data processing department, which had developed CRISP, was using it as an example of a successful system design effort—marred only by a minor database path specification error that had caused problems at first.

- Wild Enthusiasm
- Disillusionment
- Total Confusion
- Search for the guilty
- Punishment of the innocent
- Promotion of the nonparticipants

FIGURE 1-1 Real-world system development lifecycle.

The next year I went back to school to get an MBA and learn what went wrong. At the time, I thought my experience was uniquely terrible. I soon learned it was normal for large computer projects. About the same time, personal computers and fourth generation languages appeared. Hence, this book.

GOALS

This book has two goals:

- To describe modern programming techniques as applied to database-oriented fourth generation languages.
- To describe how to use DBASE III and IV to create successful programs using those techniques.

A major problem in computer use over the years has been that rapid changes in technology make knowledge obsolete. A given piece of hardware becomes obsolete in a few years, and new versions of software mandate significant relearning every year or two. Much of the reason for this problem is that we traditionally learn details rather than principles. A course devoted only to the syntax of a particular programming language becomes obsolete when that language changes. A widespread example of this problem is the large number of applications programmers in industry who have learned COBOL rather than computer programming, and who therefore find that a change to more modern languages would make most of their knowledge obsolete.

This book presents an alternative approach. Although it is intended to help you learn to develop complex systems in dBASE III and IV, 90% of the contents of the book involve principles of program design that are valid for any database-oriented language. Thus, as the dBASE standard changes over time, most of what you learn in this book will continue to be valid. What is important is not the syntax of a particular language but rather the principles of program design which that language instantiates. These are important for several reasons.

PROGRAM DESIGN

Why this different approach to programming? After all, programming textbooks since the 1950s have concentrated on language syntax and have left design issues for advanced courses. The

Year	Nominal Price	Price in 1988 $
1965	$1.50	$5.13
1975	$0.04	$0.085
1985	$0.0001	$0.0001
1987	$0.00009	$0.00009
1990	$0.00007	$0.00007

FIGURE 1-2 Price per byte of memory.

reasons why it is important to concentrate on design rather than syntax arise from both technical and organizational problems.

The first problem (or more correctly, opportunity) is the precipitous drop in computer hardware costs. You have probably seen the advertisements that boast that if cars had evolved like computers, a Rolls Royce would cost a nickle and get two million miles per gallon. Figure 1-2 shows some comparative prices for computer hardware, adjusted for inflation.

This rapid decline in hardware prices has two impacts on program design. First, the goals of programming are changed. Many programming techniques were developed at a time when computer memory and computer speed were far more expensive than they are today. As a result, these techniques save hardware costs at the expense of people costs. As we shall see in the next few paragraphs, this is backwards with current technology.

An additional impact of declining hardware cost comes from increased demand. As an example, in the twelve years between 1961 and 1973, Digital Equipment Corporation produced about 20,000 of their popular PDP-8 minicomputers—the most popular machine of its day [DEC 1973]. In 1987, IBM produced more than 1,000,000 PS/2 personal computers in six months—and this represented less than 40% of the market for personal computers alone during that period [PC Week 1987]. All of these computers being used by more and more people means that the demand for programs is far higher than it used to be, and the demand is growing exponentially each year.

Additionally, these new computers are increasingly powerful. This means that they require more complex software. In 1974, for example, an IBM 370/158 mainframe provided service for over 100 users and was catered to by a staff of more than a dozen professionals. Today, the same computing power sits on my desk.

```
//FORT      EXEC   PGM=IEYFORT,REGION=100K
//SYSPRINT  DD     SYSOUT=A
//SYSPUNCH  DDS    YSOUT=B
//SYSLIN    DD     SNAME=&LOADSET
//                 DISP=(MOD,PASS),UNIT=SYSSQ,
//                 PACE=(80,(200,100),RLSE),DCB=BLKSIZE=80
//SYSIN     DD     *
```

FIGURE 1-3 IBM 360 JCL (circa 1968)

As hardware costs have fallen, software development costs have increased. This is due to two reasons: the cost of programmers is higher, and programs are far more complex than was true in the past. One of the reasons for this complexity is that as computers become more integrated into everyday work, they must become easier to use. As a result, modern programs devote a very large portion of their code to implementing a more-or-less friendly user interface. Fifteen years ago this would have been considered a waste of time; now even user-group freeware is user friendly. Consider, for example, the difference between the Job Control Language (JCL) required to run a program on an IBM 360 (Figure 1-3) and doing the same task by moving the cursor over the file with the mouse and clinking the mouse button twice on a Macintosh—and consider that the IBM 360 probably served several hundred people, whereas the Macintosh serves one.

Software development costs do not tell the whole story, however. Depending on the organization, between 70% and 90% of software cost is incurred in program maintenance—the modification of existing programs. Some of this maintenance is required because bugs are discovered in the programs, but most is required because user needs change, the business environment changes, or the technical environment changes. To make the problem worse, traditional program development methods produce programs that are

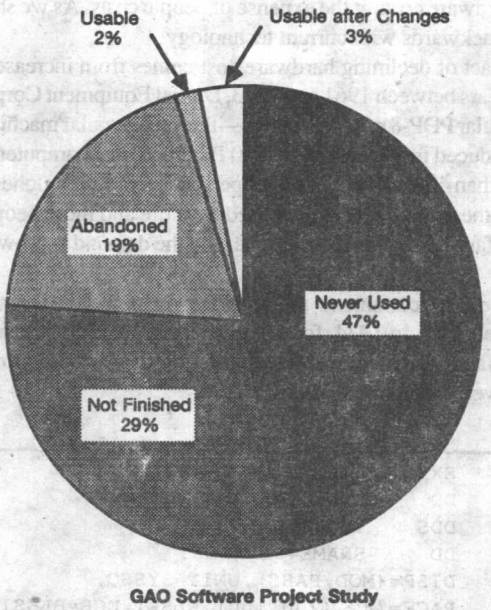

FIGURE 1-4 GAO Software project study.

difficult to maintain. One of the major goals of modern program design, therefore, is to produce programs that are easy to maintain and to modify.

All of these changes in the computer environment are a prelude to the basic problem—software design is not a reliable process. The registration system described earlier is not unique: almost all large software projects are

- Over budget
- Late
- Not what the user expected.

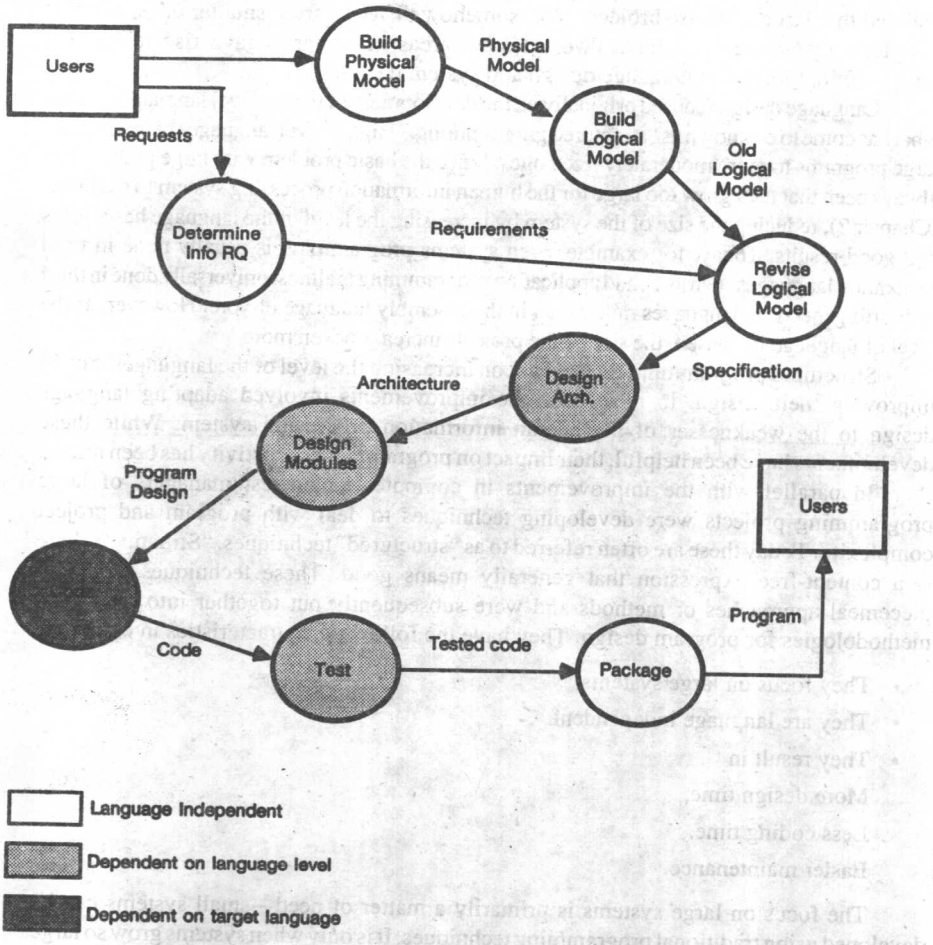

FIGURE 1-5 The systems development life-cycle.