A Name Parki BOOK

AN INFORMATION SYSTEMS MANIFESTO

Library of Congress Cataloging in Publication Data

Martin, James
An information systems manifesto.

Includes bibliographical references and index.
1. Electronic data processing.
2. Management information systems.
I. Title.
QA76.M327
1984
658.4'0388
83-17730
ISBN 0-13-464769-6

An Information Systems Manifesto James Martin

© 1984 by James Martin

All rights reserved. No part of this book may be reproduced, in any form or by any means, without permission in writing from the publisher.

Editorial/production supervision by Linda Mihatov Front matter design by Judith A. Matz Jacket design by Diane Saxe Manufacturing buyer: Gordon Osbourne

Printed in the United States of America

10 9 8 7 6 5 4

J-PJ7444767 UBZI

PRENTICE-HALL INTERNATIONAL, INC., London
PRENTICE-HALL OF AUSTRALIA PTY. LIMITED, Sydney
EDITORA PRENTICE-HALL DO BRASIL, LTDA., Rio de Janeiro
PRENTICE-HALL CANADA INC., Toronto
PRENTICE-HALL OF INDIA PRIVATE LIMITED, New Delhi
PRENTICE-HALL OF JAPAN, INC., Tokyo
PRENTICE-HALL OF SOUTHEAST ASIA PTE. LTD., Singapore
WHITEHALL BOOKS LIMITED, Wellington, New Zealand

PREFACE

An Information Systems Manifesto is designed to provide end users, DP staff, and senior and top management with a strategy and direction on how to change and manage the dramatically changing environment of information systems and data processing.

The revolution described in information systems is already in progress. Yet many corporations ignore it or fail to recognize the impact it will have on their organizations. The intent of this book is to provide people with the information they need to direct and implement these changes successfully.

The first part of the book describes the actions that should be taken to improve the use of computers, for example, by placing the end user firmly in the driver's seat and automating the process of automation.

The second part of the book examines the future, particularly the Japanese and the essential need for entrepreneurs.

This book was designed for a wide cross-section of people and is uncluttered with technical details.

It should be mandatory reading for *everyone* concerned with information systems and data processing.

James Martin

ACKNOWLEDGMENTS

The author wishes to thank the consultants of James Martin Associates for their contributions to this book and their ongoing work in implementing the methodologies it describes.

CONTENTS

Table of Boxes xv

Preface xvii

PART

THE PRESENT

1 The Crisis in Data Processing 3

Problems with Data Processing 3; Application Backlog 5; Invisible Backlog 5; The Need for Vastly Improved Productivity 6; Maintenance 6; Strategic Information 7; Decision Support Systems 9; Types of Management Systems 10;

The Fallacy of Progressive Summarization 11; Solutions 15; Cultural Changes 17; A Manifesto for Top Management 17

2 The Automation of Automation 19

Introduction 19; Meat Machines 19; Quantum Leaps 20; Insufficient Improvements from Structured Techniques 22; Avoidance of Manual Techniques 24; Rigorous Specification 24; Categorization of Development Software 25; Integration of Features 29; Procedural and Nonprocedural Languages 30; Fourth-Generation Languages 31; Solving the Wrong Problem 33; Caution 35;

x Contents

Obsolete Standards Manuals 35; On Engineering 36; Software Factories 37; Clinging to the Past 38

3 The User in the Driver's Seat 39

Introduction 39; Alternatives to Programmers 39; Varieties of End Users 40; Development That Does Not Work 41; More Rigorous Specifications 42; User-Driven Versus Prespecified Computing 43; The Uncertainty Principle 43; Management Information 45; A New Type of DP Implementation 46; Prototyping 46; The Change Needed in DP Management 48; A Wall Between User and Programmer 48; Spectacular End-User Systems 50; The Information Center Concept 50; Personal Computers for Top Management 53; Caution 54

4 Problems with Specifications 55

Problems with the Specification Process 55; User Sign-Off 56; Bugs 56; Failures 57; Solutions 59; System Sketches 59; Computable Designs 60; Immediate Error Feedback 63; The Structured Revolution 64

5 Specification Languages 65

Introduction 65; Successive Refinement 65;
Two Types of Languages 66; Computable Specifications 67;
Automation of Design 68;
Standing on the Shoulders of Others 69; Data Models 70;
Integration of Definition Levels 70;
A Common Communication Vehicle 71;
Up-Front Detection of Errors 71;
Integrated Top-Down and Bottom-Up Design 73;
Mathematically Rigorous Languages 73; User-Friendliness 74;
Spectrum of Specification Languages 77; Summary 79

6 Information Engineering 83

The Mess in Data Libraries 83; The Failure of Data Administration 83; The Costs of Bad Data Administration 85; Separate Developments with Incompatible Data 86; Stable Foundation Stone 90; Stable Data Bases 91; Logical Design of Data Bases 91; Contents

What Is Information Engineering? 92; The Building Blocks of Information Engineering 94; Computerization of Information Engineering 97; A House Upon the Sand 97; Two Images 98

7 The Information Center Concept 101

The Payback of an Information Center 101;
The Need for Management 104; Delivery Vehicles 104;
Electronic Filing Cabinets 105;
Information Center Support 105;
Connection to Information Engineering 106;
Data Coordinator 107; Conflict in Speed of Results 108;
Information Center Organization 108;
Differences in Scope 109; Techniques Analyst 111;
Functions of IC Staff 112; Selling 112; Rivalry 113;
Languages Supported 116; Communication Skills 117;
Early Success 117; Spreading the Success 117; Auditors 118;
Who Pays? 119; Businesspeople in Control 120

8 An Ideal DP Development Facility 127

Introduction 127; Uses of Data 127; Basic Facilities 128; User Friendliness 131; Default Assumptions 131; Subsetting 133; Human Usage Aids 134; Specification Tools 135; Data Base 136; Intelligent Data-Base Functions 137; Data Administration Aids 137; Semantic Disintegrity 137; System Controls 138; Code Generation and Checking 138; Portable Code 139; Code Checking 139; Provably Correct Code 139; Library 139; Theoretical Principles 140; Infrastructure 140; Multisystem Infrastructure 140

9 Mathematically Provable System Design 143

Mathematical Proofs of Correctness 143; Impossibility of Conventional Debugging 146; Higher-Order Software 146; HOS Binary Trees 147; Functions 148; From Requirements Statements to Detailed Program Design 152; Three Primitive Control Structures 152; JOIN 153; INCLUDE 154; OR 154; Control Maps 155; Generation of Code 155; Four Types of Leaf Nodes 155; Static and Dynamic Testing of Programs 157; Embellishments 158; HOS Software 159; A Way to Think About Systems 160 xii Contents

10 The Impact of Computer-Aided Software Design 163

The Revolution 163; Effect on Programming 163; Effect on Specifications 164; What Does "Provably Correct" Mean? 165; Syntax and Semantics 165; Internal and External Semantics 166; A Way to Think About Systems 167; Standards 167; Verification and Testing 168; Building Higher Levels of Trust 168; Improvements in Productivity 169; Cost Savings 169; Effect of Program Size 170; Error Statistics 172; Human System Components 173; Software Factories 174

11 The Change in the Development Life Cycle 177

Introduction 177; The Methodology Zoo 182; Life-Cycle Staffing 184; Elimination of Unnecessary Tools and Procedures 185; Use of Other Front-End Methodologies 186; Data Modeling and the Life Cycle 187; A Continuum of Information Resource Building 188; Minimizing Interactions Among Developers 190; One-Person Projects 192; Prototype Cycles 192; Multiple Types of Life Cycle 193

12 Strategic Planning 195

A Cautionary Tale 195; Future Planning 196; The Need for Top-Level Planning 196; The Danger of Integrated Systems Planning 197; Too Much Government 197; Three Types of Strategic Planning 198; Strategic Information Technology Planning 201; Aspects of the Technical Strategic Plan 201; Design of Distributed Systems 203; Networks 203; Migration Path 204; System Kluges 204; Top Management Involvement 205; Summary 206

13 Managing a Revolution 207

Resistance to Change 207; Argument Against Change 207; New Development Channels 209; Information Centers 209; Three Channels 210; Difficulties of Unlearning the Old Methods 210; New Graduates 211; Revolutions Come from the Outside 212; Case Study of a DP Revolution 212; Creativity 215; Contents xiii

Creative Partnership 215; Early Adapters 216; Motivation 217; Directives to Change Need to Be Specific 217; Winding Down Low-Productivity Development 218

PART

THE FUTURE

14 Entrepreneurs Who Will Get Rich 223

The Opportunity 223; New Directions 223; Entrepreneurial Success 224; Development Divorced from Reality 225; Bureaucracy 225; Solutions to Big-Money Problems 226; New Software 226; Hardware Changes 226, Non-Von-Neumann Languages 227; Microcode 228; Tandem-Like Architectures 228; Relational Machines 229; An HOS Machine 230; Graphics Processors 231; Knowledge-Based Processing 231; Human Language 231; Operating Systems and Software 232; Software-Hardware Combinations 233

15 Will the Next Generation Be Japanese? 235

A Discontinuity in Ideas 235; Traps for the West 236; Knowledge-Based Technology 236; Expert Systems 238; Expert Systems for Systems Experts 238; The Human Interface 240; Internal Knowledge Bases 240; Knowledge Acquisition 241; Consumer Systems 242; Effects on the Economy 243

16 Entrepreneurs and Government 247

The Threat from Japan 247; The Alvey Report 251; Four Enabling Technologies 251; Excitement 252; Entrepreneurs 253; Skunk Works 254; What Should Be Government's Role? 255; Research Inappropriate for Private Industry 255; Conclusion 257

17 Chain Reaction 259

The Automation of Automation 259; Pyramiding 261; Changing the Computer Industry 261; The Knowledge-Based Chain Reaction 262; Complexity 263; Social Effects 263; Automation of Work 264; Technical Potential/Human Potential 265 xiv Contents

Appendix

How to Improve DP Productivity 269

Appendix

II Manifestos 271

II-A: Manifesto for Senior Management 275

II-B: Manifesto for End Users 276
II-C: Manifesto for DP Executives 277
II-D: Manifesto for System Analysts 278
II-E: Manifesto for Programmers 279

II-F: Manifesto for Computing Professors 280 II-G: Manifesto for Software Houses 281

II-H: Manifesto for Computer Manufacturers 282

II-I: Manifesto for Entrepreneurs 283

References 284

Index 294

Table of boxes

Box No.	Title	Page No.
1.1	Problems of computing as seen by users and management	4
1.2	Examples of corporations achieving major financial	
	advantage from strategic (as opposed to routine) uses of	
	computerized data	8
1.3	Characteristics of decision-support systems	12
1.4	A manifesto for senior management	18
3.1	The distinction between prespecified computing and	
	user-driven computing. Much of what has been prespecified	
	ought to be user-driven with today's software.	44
3.2	Reasons why end users should "do their own thing" with	
	computers	51
3.3	Facilities needed for user-driven computing	52
5.1	Desirable properties of a specification [11]	80
5.2	Desirable properties of a specification language	80
6.1	Reasons for failure of corporate data administration	84
6.2	Essentials for the overall control of data in an enterprise	84
7.1	Quotations from executives at companies that employ	
	information centers	102
7.2	Functions that should be carried out by an information	
_	center	114
7.3	How to succeed with information centers	123
10.1	Problems with traditional development and the effects of	
	rigorous specification languages with code generators	174
11.1	What do we need in DP methodologies?	182
12.1	Examples of system problems that lead to expensive	
	maintenance and can be avoided with top-down	
	planning	205
15.1	Some existing expert systems using knowledge-based	
	technology	239
16.1	Targets of the Japanese Fifth Generation project	
	for delivery by 1990 [3]	248
17.1	Aspects of work that will be automated and	
	aspects of work that will not be automated	265

PART THE PRESENT

THE CRISIS IN DATA PROCESSING

PROBLEMS WITH DATA PROCESSING

The power of today's computing technology is not being used as it should be in most enterprises. Data processing is bogged down in problems. Managers

are not receiving the information they require from their systems. Many decisions that should be made with the aid of computers are in fact being made with hand methods or inadequate information. Systems are so difficult to change that they often inhibit the implementation of new and important procedures that managers require. Computer users are increasingly hostile to DP but feel powerless to prevent the problems.

Among top-level managers there is often a sense of anger that they are spending so much on computing and yet seem unable to change procedures or obtain the information they need. In one corporation with an expensive and elegant worldwide computer network the chief executive complained bitterly to the author that for years he had been asking for daily or even weekly figures and cash balances but seemed no closer to obtaining these or other information he needed.

Box 1.1 lists the problems of computing as seen by users and management.

Failure to use computers effectively is serious on a national scale because it affects most aspects of productivity. This affects inflation and the ability to compete overseas.

The computer is the most flexible machine invented, capable of a staggering diversity of applications. It is rapidly dropping in cost and its power needs to be used as fully as possible for improving the efficiency of organizations. The problem lies not in the machine itself, but in the methods we use for creating applications. The traditional "application development life cycle" is slow and rigid. Its methods have been cast into concrete in many organizations with standards and procedures. But in many ways the procedures are not working.

BOX 1.1 Problems of computing as seen by users and management

- Users cannot obtain applications when they want them. There is often a delay of years.
- It is difficult or impossible to obtain changes that managers need in a reasonable amount of time.
- The programs have errors in them or sometimes do not work.
- Systems delivered often do not match the true user requirements.
- It is difficult to understand DP and communicate precise requirements.
- Specifications, on which users have to sign off, are difficult to check and are usually full of inconsistencies, omissions, and errors.
- Systems cost much more to develop and to maintain than anticipated.
- Because of the long time required to obtain results, the most important decision-support systems are never implemented. The support is needed more quickly than the time needed to create the programs.

This inability to use computers effectively should be regarded as a major organizational problem to which solutions *must* be found. The problem has reached crisis proportions. The solutions now exist but many organizations are not using them except in isolated pockets. They require new software, new procedures, and new management structures.

As we look at the history of technology we can observe certain times when a major break with the past methods had to occur. In computing, a set of application development methods has been accepted and slowly refined for more than two decades. We have now reached a point when these are inhibiting the most effective uses of computers. Fundamentally different methods are needed and are coming into use. Unfortunately, many data-processing (DP) organizations are not adopting the new methods rapidly enough.

Steam engines made earlier this century were beautifully intricate machines. They had many polished brass sliding rods, levers, and cams. Engineers had invented elaborate mechanisms for extracting a fraction of a percent of extra performance. They lovingly tuned the mechanisms and held technical symposia on steam engines long after the electric motor was in use.

Comparing relational data-base facilities with some of the old data-base management systems, or comparing new-architecture small computers with the old large operating systems, seems like comparing an electric motor with steam engines. The architects and maintainers of the old software are making

elaborate additions which give only minor improvements, when a switch to fundamentally better mechanisms is needed.

The difference between COBOL development and the creation of applications with the new application generators is even more dramatic. Many powerful new techniques exist today for creating software and building applications, but a million programmers and analysts around the world have not heard about them. Amazingly, in Europe steam engine designers were still refining these machines 50 years after the electric motor had been invented.

In Victorian factories with steam engines there were overhead shafts 100 feet long with large pulleys and belts going down to each machine tool. With electric power each machine tool could have its own motor. But the shafts remained in many factories long after their usefulness had ended. New tools often need fundamentally new methodologies. The procedures manuals in DP are often referred to as the "Bible" of DP development and it is heresy to disobey the Bible even when new software and techniques render it hopelessly obsolete. The old DP procedures, like the steam-driven shaft and pulleys, can preclude the freedom that is necessary to move flexibly with the new methods.

The Random House Dictionary defines a manifesto as "a public declaration of intentions, objectives, opinions or motives." In organizations everywhere a manifesto is needed from top management giving their policy for information systems. This can be translated into manifestos concerning the fundamental changes in DP techniques and productivity and how they should be handled by DP executives, systems analysts, and programmers.

APPLICATION BACKLOG

In most well-managed corporations the demand for new applications is rising faster than DP can supply them. The imbalance between demand and supply

is steadily becoming worse.

Because of this, the backlog of needed applications is growing. Most corporations now have a backlog of from two to four years. One bank executive informed us that the bank's backlog was seven years. This situation is likely to become worse as machines drop in cost, unless better methods of creating applications are found.

The long backlog and inability of computer departments to respond to end users' needs quickly is very frustrating for the users. In many cases the users have felt that they cannot wait and have obtained their own departmental minicomputer or desktop computers.

INVISIBLE BACKLOG

Even though today's application backlogs are so long, they reveal only part of the story. When the documented backlog is several years (as it is in

most installations), the end users do not even consider making requests for many of the applications they need. There is thus an *invisible backlog*.