The Programmer's Odyssey

Garth Eaglesfield

The Programmer's Odyssey

A Journey Through The Digital Age

Garth Eaglesfield

Table of Contents

Preface i

Prologue iii

The Programmer's Odyssey 1

The Basics Of Binary And Hexadecimal 2

1940s - Code Breaking Machines 4

When The War Was Over 4

The Basics Of Turing's Ideas 5

The Turing Test 6

Meanwhile In The USA 6

1959 The Day The Music Died 10

1960s - Super Number Crunchers 12

Trying To Remember The 60s 12

The Basics Of Maths In The Pre-Calculator Era 12

The 60s Really Get Going 15

Cold War Paranoia Feeds Technology Anxieties 15

Losing My (Maths) Religion 20

1971 – I Finally Get It 22

Dave And HAL Have A Disagreement 22

Everybody's Plan B 24

Transformational Ideas 25

It's A Logic Machine 27

1972 - Training To Be A Programmer 28
COBOL - The COmmon Business Oriented Language 28
The mother of COBOL, Admiral Grace Hopper 30
The Basics Of The COBOL Language 32
1973 Brighton – On The Production Line 40
The Industrial Manufacturing Process 40
Designing Your Program 42
The Monolithic Sequential Programming Style 43
Writing Your program 46
Creating The Source Code Card Deck 48
Translating Source Code Into Machine Code 50
Program Testing 54
The Batch Processing Production Environment 54
The Program Maintenance Nightmare 59
Structured Programming 61
Software Engineers Arrive On The Scene 65
Online Developers 65
AI Stalls 67
1975 London - A Whole New Ballgame 68
Minis - The PDP11/70 Is Born 69
Train Yourself 71
Dumb Terminals 71
Do It Yourself Programming 72

RSX11 – A Multi-User Operating System 72
Subroutine Components For Programs 74
The Basics Of PDP Computer Architecture 75
Message Switch Infrastructure Software 80
The Monolithic Component Programming Style 81
1977 - The Microprocessor Revolution 84
Those Crazy Guys At Micro Focus 84
The CIS COBOL project 87
Moving To The Intel 8080 90
The End Of My Microprocessor Road 91
1979 – Minicomputers Proliferate 92
Software In The Signal Box 92
The PDP Gives Birth To The VAX 93
The Device Driver Challenge 94
The 70s Draw To A Close 95
1981 New York City – Computing USA 96
Welcome to the Big Apple 96
Do Androids Dream Of Electric Sheep? 97
The IBM PC Arrives 99
MS-DOS 99
Back To Batch But On To Freedom 100
The User Becomes The Tester 100
Lipstick On A Pig 101

The Killer App! 102
Dialling In..... 102
A Niche On Wall Street 103
Unix Takes On VMS..... 106
On The Road To Bell Labs 107
Unix Development.....110
The Instinet Trading System 112
The Wall Street Shuffle 114
The Great Vampire Squid.....115
The Basics Of Some Crucial Developments.....117
Relational Databases and SQL.....118
Inter Process Communication (IPC)..... 121
Tracking The Data Flows 122
Object Oriented Design & Programming 123
The GUI Meets The Laser Printer 124
CISC versus RISC..... 124
1988 - Distributed Computing..... 126
Networks Start New Religious Wars..... 126
The Transvik Project 129
The Basics Of Tuxedo..... 133
1991 Chicago – Electric Blues 136
A Two Tier/Three Tier Hybrid At Nielsen..... 136
There's A Storm Coming In 139

1993 Back In NYC – Pushing 3-Tier 142
Selling 3-tier 142
The Distributed Application Programming Style 144
You Can't Go Home Again 146
1994 London – Bringing It All Back Home 147
GA Systems 148
LMS – The Big One 149
BEA Systems 155
The Internet Storm Comes In 159
Dot.com Hype and Dot.com Reality 161
The 21st Century 162
The View From Outside The Trenches 162
New Players And Old Players 163
What Of Tuxedo? 163
The Users Who Came In From The Cold 164
Mobile Technology 164
Smartness Everywhere 165
'The Medium Is The Message' 166
Open Source 166
Networks And SOA 167
AI 167
And The Final Stormcloud Is …The Cloud! 168
21st Century Programming (Coding) 168

Programming Languages 168
Programming Techniques 170
Devops 171
Apps 171
Postscript/O Brother Where Art Thou? 172
Appendices 174
Appendix A – The ASCII character set 175
Appendix B - Peter Brown On CIS COBOL 176
Appendix C – BEA and GA 181
Appendix D – Some Useful References 183

A few sample pages from the book follow to give an idea of it's style and general approach.

1975 London - A Whole New Ballgame

By 1975 I was a veteran of three programming jobs in a fairly short period of time and I was concerned that my CV (US=resume) was beginning to scream ‘job hopper’. But professionally and personally I was anxious to keep moving and to learn the new technologies which were rapidly emerging. In particular I wanted to get off mainframe computers and on to mini-computers, or minis as they were commonly referred to. Not only were they the latest technology but their very name carried a trendy, even sexy, hint of two major UK cultural icons of the 1960s, the Mini car and the mini-skirt or mini-dress.

A Mini and Twiggy in a mini-dress

The solution I came up with was to join an IT consultancy, which in those days were known as ‘software houses’ in the UK. By doing so I hoped I would be able to experience many different computer environments while remaining at the same employer. London offered by far the most options so I travelled back there from Brighton and joined a small, and long since disappeared, software house called CMES.

Minis - The PDP11/70 Is Born

In the 1970s Digital Equipment Corporation (DEC) in America started a computing revolution with their PDP11/70 series of minicomputers which by comparison to traditional mainframe computers were cheap and small. This class of computers would later come to be given the generic title of servers in order to distinguish them from mainframes. DEC were based close to the Massachusetts Institute of Technology (MIT) on Boston’s route 128 which had the same iconic status in the computing world that would one day be occupied by Silicon Valley.

A PDP11/70 front panel

DEC had produced earlier versions of the PDP but it was really the power of the PDP11/70 that brought PDP computers widespread acceptance in the commercial business world.

For the first time it started to make commercial sense for small and medium sized businesses to have their own computers. You could now have a fully functional mini-computing environment without a giant air-conditioned computer room packed with bulky expensive equipment and a large operations staff; without an extensive punched card production and management operation; with a separate computer devoted to program development and with the whole environment involving a less than astronomical budget. This was also of interest to software houses as it allowed them to go beyond providing temporary staff to their customers and to develop their own software in-house. Computing was beginning to get beyond the control of the large computer manufacturers and large computer users. Even so the extent of these changes was seen as being strictly limited to businesses and in 1977 Ken Olson, founder of Digital Equipment Corporation, famously said “There is no reason anyone would want a computer in their home”. But this at least was an advance on Thomas Watson of IBM’s earlier estimation of the size of the computer market being only five computers.

It was even rumored in the London programming circles in which I moved that there were some very successful freelance programmers who had actually bought their own PDPs and could create their own software. This was the Holy Grail for programmers, to finally escape from the trap described by Karl Marx, to own the means of production and to be able to create their own software independently. The programmer’s revolution was beginning to happen.

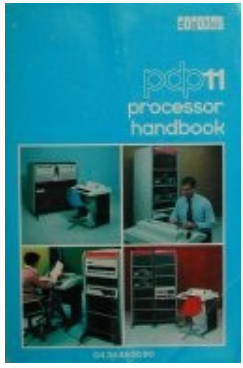
These developments meant that hardware costs as a proportion of an IT project’s budget started to fall relative to the software costs. Previously it was hardware that devoured a huge percentage of any IT project’s budget but now the balance was starting to shift towards software and would continue to do so.

For the next two years, while I worked for CMES on PDP11 projects, I was exposed to so many new programming practices and technology it is impossible to arrange them in a clear timeline as everything was happening to me simultaneously so I present them below in a more or less logical order. The overall effect of all these changes was to begin breaking down the Monolithic Sequential

Programming Style and to start replacing it with an early version of what could be called the Monolithic Modular Programming Style.

Train Yourself

At CMES, programming on a PDP11/70 involved using the low level MACRO-11 assembler language which in turn meant having a decent understanding of the actual PDP11/70 hardware architecture. DEC provided a few manuals the size of paperback books from which we at CMES were supposed to train ourselves, often during billable time at a client site.



Apart from doing the reading you could ask more experienced programmers you were working with to help you out and hope they were willing to share information with you, though not all of them were.

It was very much a sink or swim experience.

Dumb Terminals

(#UI) Something that was crucial in reducing the cost of mini-computer environments and helped popularize them were ‘dumb’ ASCII terminals with which to create relatively cheap online systems. It was their dumbness that made them so much cheaper than ‘intelligent’ IBM 3270 terminals. Unlike the 3270s dumb ASCII terminals did no pre-processing of the keyboard input and did not require expensive coaxial cabling to connect them to the PDP server. Each input keystroke was immediately transmitted to the

PDP and was processed by software running on the PDP computer itself.

A VT100 terminal

Very quickly DEC's VT100 terminal established itself as the dominant dumb terminal and as the de facto standard.

Do It Yourself Programming

My first experience of PDP11/70s was on a project at N. M. Rothchilds merchant bank where to my astonishment there were no key punch girls and no punched cards. We were expected to enter our own source code through dumb terminals using a simple line editor program and it was stored directly into a source code file on disk. For those who can remember MS-DOS, think edlin.

Never having had a typing lesson in my life, like most of my colleagues, this tended to be a rather hit and miss affair at first, but even so it was still quicker and more flexible than relying on key punch girls.

RSX11 – A Multi-User Operating System

DEC's multi-user RSX11 operating system underlay the new programming environment and it too was revolutionary. It supported large numbers of users simultaneously through a command line interface that was presented on their dumb terminals. Thus many