Book Reviews

Richard F. Ferraro

Programmer's Guide to the EGA, VGA, and Super VGA Cards,
Third edition

Addison–Wesley Publishing Company Inc., Reading, Massachusetts, 1994, pp. xxix, 1600, ISBN 0-201-62490-7.

This book deals with the programming of graphics cards, one of the most important peripherals for personal computers. Graphics cards enable users to generate sharp, effective graphics presentations on the video monitor via images created by dots, lines, curves, arbitrary shapes, and alphanumeric characters. First display formats standardized for the PC family of computers were based on *Monochrome Display Adapter (MDA), Color Graphics Adapter (CGA)*, and *Hercules Graphics Adapter*, all of which exhibited poor resolution or a limited palette of colors. Specifically to solve the problem associated with CGA, IBM developed its *Enhanced Graphics Array (EGA)* standard and the pertinent adapter board in 1984. Since then, EGA has become the third most popular color graphics standard for the PC. In 1987 the *EGA* was replaced with the *Video Graphics Array (VGA)*, the principal advantage over EGA being the addition of a 320-by-200 256-color mode and the ability to drive an analog monitor. Although the VGA standard still reign supreme in color graphics systems for the PC, it has grown beyond its original scope so that all modern VGA cards exceed the standard in some significant way. Such cards have more memory, additional display modes, increased color resolution, a hardware cursor, and additional hardware supported graphics functions. These new and improved VGAs have been labelled *Extended VGAs, Advanced VGAs*, and *Super VGAs*, while Ferrara calls them *Super VGAs*. The additional features of Super VGAs have unfortunately not yet been fully standardized, in spite of efforts of VESA (*Video Electronics Standards Association*), making life difficult for graphics applications programmers. This book is meant to alleviate the situation by offering a thorough description of Super VGA details. This is also the main difference with respect to the previous editions, which focused on EGA and VGA.

The book consists of 30 chapters and has a well-compiled index, all-in-all over 1500 pages of excellent text on programming graphics cards. The chapters could be divided into three groups: Chapters 1 to 5 offering the basics of PC graphics, Chapters 6 to 12 giving an exhaustive description of EGA and VGA, and Chapters 13 to 30, making the bulk of the book in corroborating on Super VGA.

After the introductory chapter, Chapters 2 through 5 initiate the reader to EGA/VGA and Super VGA card features including display, and graphics read and write modes, spatial and color resolutions, and display memory. Definitions and basic explanations on graphics accelerators are also provided. *Principle of Computer Graphics* acquaints the reader with coordinate systems and transformations, characters, point, line, circle, image and color theory. Special attention is given to the important line drawing (incremental and Bresenham line algorithm), circle drawing and clipping algorithms.

Chapters 6 through 11 offer comprehensive information on display modes, display memory, graphics processing, downloadable fonts, color processing, control registers, and BIOS calls for EGA/VGA cards. *Alphanumeric Processing* furnishes the reader with the necessary tools to program character shapes, sizes, and attributes as well as downloadable character sets, here including the organization of the fonts and tech-

niques for reading, writing and selecting them. *Graphics Processing* supplies the reader with the necessary tools for programming graphics on EGA/VGA. Character, character attributes, and character fonts are described as they apply to the graphics display modes, which are also presented. The organization of the display memory for each of these graphics modes is provided. *Color Palette and Color Registers* illustrates the color processing capabilities of the EGA/VGA. The palette registers and the color registers are described in detail, with special attention paid to techniques for reading data from and writing data to these registers. Techniques for converting data to colors as well as from one color scheme to another are presented, too. *Reading the State of the EGA and VGA* focuses on determining the state of the graphics adapters. The register default values for the EGA/VGA are presented in table form. These values are useful to the programmer when trying to understand the operation of display modes and the interaction of various display registers. *The EGA/VGA Registers* presents in detail the entire set of registers resident in the EGA/VGA cards. *The EGA/VGA BIOS* describes all of the BIOS calls on the EGA/VGA cards, providing also input and output parameters as well as the effects that each call has on display memory, host registers, EGA/VGA registers, and the BIOS data area. *Programming Examples* gives implementations of many standard functions for EGA/VGA cards written in both C and assembly language. Many of these functions address directly the EGA/VGA, while others rely on BIOS calls. These functions span the text, font graphics, and color sections of the adapters. Each function is well documented and simple to understand.

After this deep insight in EGA/VGA card organization and operation the rest of the book is devoted to Super VGA. The performance of these cards depends on the particular VGA controller chip used. Although the cards differ in many ways, performance features of each of them are very similar. The chapter entitled *The Super VGA* equips the reader with the basics of their operation. The user is exposed to the terminology, identifying the VGA card and its capabilities, including the access to extended registers, display modes, and display memory. *Super VGA BIOS, VESA Video Bios Extensions, Memory Addressing Techniques, Working with Color, BitBlts, Color Expansion, Transparency, Masking, Raster Operation, Hardware Cursor, Line Drawing* are some of the important sections and subsections describing the main features of Super VGA cards. Chapter 14 provides insight into graphics accelerators found on many of the newer generation Super VGA chips. It highlights their power and capabilities as well as their drawbacks. *Super VGA Code Basics* describes the theory behind the common features in the Super VGAs and illustrates some of the code examples. Details and program examples for particular implementations for different Super VGAs can be found in Chapters 16 through 30.

Chapters 16 through 18 offer a detailed look at the IBM Applications (AI) interface, and IBM adapters 8514/A and XGA. The AI interface is fully supported with the XGA, 8154, and several of the Super VGAs. It typically comes as a TSR (terminate-and-stay-resident) device driver and provides easy access to several popular functions including filling regions, BitBlts, scrolling, state initialization, setting modes, setting patterns, text processing, palette control, and line drawing.

The rest of the chapters provide an in-depth look at the Super VGA chips that make up the vast majority of all VGA cards on the market. However, the cards themselves are not covered. The Super VGA chips discussed in these chapters are the ones coming from well known manufacturers like ATI, Chips&Technology, Cirrus, Headland/Video7, Integrated Info Tech (IIT), NCR, Oak, S3, Trident, Tseng Labs, Western Digital, and Weitek. Functional descriptions of chips, the way to identify the manufacturer, chip and version number, the way to access display memory, bank swap, control of the start address and cursor, descriptions of BIOS extensions, functional descriptions of graphics accelerators when present, and special features of concern to the programmer are some of the topics covered.

In conclusion, *Programmer's Guide to the EGA, VGA, and Super VGA Cards* is a brilliant reference book that has a lot of details otherwise to be found in a number of technical manuals, but contrary to them it also offers a great deal of instructive examples, helping the reader to more easily master graphics programming. The examples are written so as to show most of the advanced features of EGA/VGA and Super

VGAs cards. In my opinion this book deserves to be on the desk of every graphics applications programmer (or better, beside his video monitor).

Goran Zelić, Krunoslav Martinčić
Faculty of Electrical Engineering and Computing
University of Zagreb
Zagreb, Croatia

P. Smith

Frame Relay. Principles and Applications

Addison–Wesley Publishing Company Inc., Wokingham, England, 1993, pp. xi, 268, ISBN 0-201-6240-1

Although not formally divided, this book could be logically split into two parts. The first part, from the beginning to inclusively Chapter 5, recapitulates the evolution of the frame relay protocol, its implementations, and the process of protocol standardization. Particular emphasis is placed upon the pitfalls and benefits of other WAN communication methods, and the reason why frame relay fits in as an alternative. It is also explained how frame relay works, with special attention pointed to those situations which are ideally suited for frame relay as well as those areas which are not suited to the protocol. The rest of the book, from Chapter 6 to the end, covers the detailed technical aspects of frame relay and describes the frame relay protocol in detail. While covering an in-depth technical description of the protocol implementation, this part also covers the enhancements to frame relay being considered and describes the implementation of the congestion control procedures and the network management requirements.

Chapter 1 introduces the reader to the evolution of information technology applications and the different wide area networking solutions. This leads then to a description of the rationale and philosophy behind frame relay. The second chapter explains the principles behind frame relay in a simple, not too much technical manner. The chapter offers an introduction to the concept of committed information rates and explains the situations where frame relay will work well, as well as those applications that are not suited to frame relay. In Chapter 3 many different implementations of frame relay are examined. Comparisons are given between frame relay as a public network service and as a private network solution, between packet switching and circuit switching vendor's implementations, and between different implementations of LAN routing networks. Finally, the chapter provides some simple questions for the network manager to consider when implementing a frame relay network. As frame relay is not suited to all data networking requirements, Chapter 4 discusses several protocol issues which need to be taken into consideration. This chapter also covers issues such as network performance, data delivery and congestion management. Chapter 5 offers a perspective on the evolution of frame relay from the ISDN standards and explains the work of the Frame Relay Forum in the development of new frame relay standards.

A thorough description of the frame relay protocol, its features and its options, is included in Chapter 6. The local management protocol is discussed in depth. An interesting part in this chapter is a comparison between the frame relay specifications established by ITU-TS (previously CCITT) and ANSI. Subtle differences between two standards, such as a different PVC (Permanent Virtual Circuit) status information element identifier within the LMI (Local Management Interface) specification can result in lack of interoperability between devices built around different standards. Chapter 7 details the enhancements available within the frame relay protocol and describes in detail the network-to-network interface, switched virtual circuits and multiprotocol encapsulation methods. For completeness this chapter also describes a proposal for carrying voice circuits across a frame relay network. One of the most important aspect of frame relay networks is their method for congestion handling, a topic covered in Chapter 8. This chapter describes several available congestion control mechanisms explaining the recommendations laid down by the standards committees and the ways in which they can be supplemented by vendors' equipment. The management of

any network is an important topic, and chapter 9 discusses some of the issues which need to be considered when managing a frame relay network. This chapter also discusses several network design considerations which may serve to improve the overall network performance. Frame relay is not likely to be implemented as a single protocol network, consequently the last chapter describes some of the approaches being taken to mix different traffic types across frame relay backbone networks. The operation of frame relay assembler/disassemblers (FRADs) and frame relay concentrators is described as well.

Reading would have been more enjoyable if there were not some errors spread throughout the text. There are some errors which could be accounted as typing errors; e.g. there is a formula to calculate the number of circuits in a fully interconnected network which should have 2 in the denominator, instead of N as indicated (page 15). There are some other errors which are more of logical nature, such as the statement "*The input lines can and frequently will, exceed the sum of the output line*". Some errors are not obvious as in the above mentioned examples, so forcing the reader to stop and think over to distinguish whether it is the case of an awkwardly assembled sentence or he just does not understand the matter. Such places are not abundant, but are anyway causing annoyance while reading the book.

By reading this book someone will benefit from getting a good insight in frame relay technology. However, the book lacks some deeper theoretical foundations to be considered a textbook. Maybe the most valuable thing to learn is the proper placement of frame relay on the datacom technology ladder, somewhat relieving the *deus ex machina* hype accredited to it through popular magazine articles. I would recommend this book to anyone who has some basic knowledge on data communications and wants to understand how frame relay works, as well as where, how and especially why it should be implemented. Students of computer data communication and similar courses could use it as a supplementary reading.

Dragutin Vuković
MicroLAB Computer Consulting & Engineering
Zagreb, Croatia

R. Cheswick, S. M. Bellowin

Firewalls and Internet Security: Repelling the Wily Hacker

Addison–Wesley Publishing Company, Reading, 1994, Massachusetts, pp. xiv, 306, ISBN 0-201-63357-4.

As a user of the Internet, you are fortunate to be tied into the world's greatest communication and information exchange — but not without a price. As a result of this connection, your computer, your organization's network, and everything the network reaches are all vulnerable to potentially disastrous infiltrations by hackers. With this book in hand, you will be well equipped to deal with such threats.

The authors, who are the senior researchers at AT&T Bell Laboratories, where they have designed and are maintaining AT&T's Internet gateway, have published numerous papers in the field, which have established them as experts on the subject. In the book reviewed they show a step-by-step plan for setting up a "firewall" gateway — a dedicated computer equipped with safeguards that acts as a single, more easily defended, Internet connection. The book consists of 14 chapters divided in four parts, three appendices, a large bibliography, a list of possible vulnerabilities and a well-compiled index. Many examples, figures and tables complement this easy readable book.

The book begins by introducing the problem of security in Chapter 1. The authors describe various security policies and strategies for secure network such as: host security, gateways and firewalls, protecting passwords and encryption. In Chapter 2, *An Overview of TCP/IP*, they survey the important parts of TCP/IP protocol suite, with particular attention to security issues. The authors discuss security holes in different TCP/IP layers, routers and routing protocols and in the Domain Naming System. Standard services (SMTP used for electronic mail transfer, telnet, the Network Time Protocol, *finger*

and *whois*) are also analysed with focus on security. RPC based and file transfer protocols, information services and the X11 System conclude this chapter.

The second part of the book describes firewall construction in detail. In Chapter 3 the authors describe several sorts of firewall gateways that have been built. They focus on firewall philosophy and placement, and discuss packet-filtering gateways in much detail. This chapter completes with some considerations on what firewalls can't do. In the next chapter, *How to Build an Application-Level Gateway*, the authors present a comprehensive description of the construction of their newest gateway. The main topics of discussion in this chapter are hardware configuration options, initial installation, gateway tools and services. Gateway administration, safety analysis, performance and evaluation are also analysed. In Chapter 5 the variety of authentication strategies to choose from are presented. The subject matter is divided into user authentication and host-to-host authentication. Some other gateway tools they used such as *proxylib* and *syslog* are shown in Chapter 6. Chapter 7, *Traps, Lures, and Honey Pots*, presents the sorts of monitors the authors have installed, and is complemented by a description of necessary logs and how to monitor them. In Chapter 8, *The Hacker's Workbench*, the authors describe the hacking tools they have built to test security. Information on how to find hosts and networks, probe hosts, monitor networks and break-ins concludes the second part of this book.

Chapter 9, *Classes of Attacks*, which opens the third part of the book, is an attempt at defining a taxonomy of hacking, and an analysis of different categories of attacks. The main categories discussed are: stealing passwords, social engineering, bugs and backdoors, authentication failures, protocol failures, information leakage and denial-of-service. Chapter 10, *An Evening with Berferd*, gives the description of the single most determined attempt to hack the author's system, the so-called "Berferd" incident. Every detail of the break-in and the measures the authors used against it is presented. Chapter 11, *Where the Wild Things Are: A Look at the Logs*, summarizes the log data the authors and others have collected over the years. Proxy use, attack sources and line noise are also analyzed and discussed.

The last part of the book begins with Chapter 12, *Legal Considerations*. The authors discuss the legal implications of computer security such as computer crime statutes, log files as evidence, legality of monitoring and tort liability considerations. In Chapter 13, *Secure Communications over Insecure Networks*, they show how encryption can be used in high-threat environments. From an introduction to cryptography, over the Kerberos Authentication System and link, network and transport level encryption to application level encryption, topics are presented in detail. The final chapter of the fourth part, Chapter 14, *Where Do We Go from Here?*, is the discussion on further developments and targets in the Internet security.

The book *Firewalls and Internet Security* gives the reader invaluable advice and practical tools for protecting his organization's computers from the very real threat of a hacker attack through the Internet. This is the book every network and system administrator would like to have in his/her book collection, and I recommend it.

Darin Cihlar
Faculty of Electrical Engineering and Computing
University of Zagreb
Zagreb, Croatia

Thomas A. Standish

Data Structures, Algorithms & Software Principles in C

Addison–Wesley Publishing Company, Inc., Reading, Massachusetts, 1994, pp. xi, 748, ISBN 0-201-59118-9.

This book covers material recommanded for a second course in computer science. Also, it covers material recommended by ACM's Curriculum '78 (and revised in 1984), and Computing Curriculum 1991.

The book is appropriate for students who are already familiar with C's statements, the C's block

structure and control flow. It covers traditional software engineering concepts and principles, but also introduces new material in the software engineering (risk-based software lifecycle models, rapid prototyping, and reusable software components).

The final part of the book is devoted to object-oriented programming based on C++.

The text has sixteen chapters, an appendix and an index.

In the introducing chapter, *Preparing for the Journey*, the author explains computer science as being slightly different from other contemporary sciences, as a blend of mathematics, science and engineering. Also, he stresses some enduring principles in computer science (basic machine organization principles, fundamental arithmetic, logical, and data movement operations, representation, abstraction, information-hiding, interfaces etc.).

The second chapter, *Linked Data Representations*, deals with linked representations. After an intuitive discussion of pointers, the technical details of managing pointers in C are introduced by the simplest example of pointers — pointers to integers. The tricky issues, such as dangling pointers and aliases are described. The topic of the chapter are linear linked lists. How to declare the pointer data types for list nodes in C, how to create and delete list nodes and how to link them together is shown in the chapter. The last section gives brief examples of linked representations that can be constructed from nodes having two links per nodes (two-way linked lists and linked binary trees).

The third chapter introduces recursion as an important concept in computer science. An author's approach to recursion is based on a graduated sequence of examples of recursive programs that aim at helping a reader learn to "think recursively". On the experience based on a sequence of examples of recursive programs, T. A. Standish generalizes the recursion as a divide and conquer strategy. At the and of the chapter the typical examples of recursion are considered.

The goal of the chapter *Modularity and Data Abstraction* is to learn about modularity, information hiding, and data abstraction. The definitions of a module as a unit of organization of a software system, abstract data type and interface are given. The main objectives of this chapter are: to learn about the parts of a C module, to understand the purpose of the separate interface and implementation files in a C module, to learn what an abstract data type is,and to learn how to use the information-hiding features of C modules to hide the implementation details of an ADT's operations. Also, the chapter deals with concepts for representing abstract data types and how to implement representation-independent notations.

The fifth chapter, *Introduction to the Software Engineering Concepts*, describes software principles concerning small programs (i.e. programs consisting of a few lines or a few pages). The chapter deals with topics such as top-down programming stepwise, proving programs correctness, transforming and optimizating programs, testing and bottom-up programming. The last section describes programming structuring and documentation.

In the chapter *Introduction to Analysis of Algorithms*, the elementary and simple mechanisms for algorithm analysis are introduced. The formal definition of O-notation and some useful shortcuts for manipulating O-notation are given.

The examples of analyzing simple algorithms (Sequential search, Selection sort, Towers of Hanoi, MergeSort and Binary searching) are presented.

Linear data structures as collections of components arranged in a straight line are described in the seventh chapter *Linear Data Structures — Stacks and Queues*.

Chapter 8. is devoted to two special cases of linear data structures: lists and strings. This chapter covers dynamic memory allocation techniques and generalized lists.

Chapter 9. deals with trees as one of the most important data structures in computer science. After introducing some basic concepts and terminology about trees, the binary trees and their representations are discussed. Some techniques for traversing binary trees are also described. The final sections explore binary search trees, AVL tree, and a tree called 2–3 tree. An application of binary trees is illustrated for the process of establishing a minimal-length encoding for messages spelled with letters in an alphabet

where the frequency of the use of the letters is known.

Graphs, as collections of nodes in which various pairs of nodes are connected by the line segments, are described in Chapter 10. The graph representations, graph searching algorithms, topological ordering, the shortest path algorithms and task networks are topics of this chapter.

A table as an abstract storage device that contains table entries is described in Chapter 11. Numerous ways to represent table operations (retrieve, update, delete, insert and enumerate) are introduced in this chapter. The author also investigates another class of representations for abstract tables relying on a technique called hashing. In the final part of the chapter, the algorithms for hashing by open addressing are described.

Chapter 12 is devoted to processing external collections of data. After describing characteristics of the external storage devices, the techniques to search and sort large collections of records are explained.

The problem of sorting is investigated in Chapter 13. The sorting methods are grouped in various clusters that share common themes. These clusters are the base of the organization of the sections in this chapter. Priority queue methods, divide-and-conquer methods, insertion-based methods, address-calculation methods are the main topics described in the chapter.

In Chapter 14 the author shows how to use recursion to specify the grammar of arithmetic expressions and how to use a parser to recognize the structure of arithmetic expressions.

Chapter 15, called *Object-Oriented Programming*, introduces the basic concepts of object-oriented programming. The origins of object-oriented programming, objects, object references, C++ as an extension of C, classes and inheritance, and virtual functions are the topics of this chapter. The basic concepts are illustrated by a small object-oriented drawing program. The advantages and disadvantages of object-oriented programming are discussed in the final part of this chapter.

The last chapter, called *Advanced Software Engineering Concepts*, presents the programming-in-the-large, i.e. building big software systems. The chapter deals with the software lifecycle, software productivity and software process models.

Each chapter has a well-organized structure consisting of the following parts: introduction and motivation, plan for the chapter, central theme, review questions, exercises, pitfalls tips and techniques, references for further study, and chapter summary.

It is my pleasure to recommand this excellent book to all students and postgraduate students of computer science. Also, it is a very useful book for scientists, experts, programmers and software engineers.

Slobodan Ribarić
Faculty of Electrical Engineering and Computing
University of Zagreb
Zagreb, Croatia

Bjarne Stroustrup

The Design and Evolution of C++

Addison–Wesley Publishing Company, Reading, Massachusetts, 1994, pp. x, 461, ISBN 0-201-54330-3.

In the last decade C++ has become the most popular object-oriented programming language. Although it was originally designed for operating systems programming, today it is increasingly used in a large variety of applications, even in numerical and scientific work. C++ is often mistakenly described as a superset of a plain C language. It is an evolution of the C programming, which extends it in at least two very important ways: it supports object-oriented programming and it supports data abstraction. Therefore, the way of thinking in C++ is completely different than in other classical languages.

Bjarne Stroustrup is the designer and original implementor of C++. He is a distinguished member of the Computer Science Research Center at AT&T Bell Laboratories. His research includes distributed systems, operating

systems, simulation, and design. Dr. Stroustrup is the author of the reference book *The C++ Programming Language* (first and second edition), co-author of The *Annotated C++ Reference Manual*, and the author of a long list of articles published in journals and presented on conferences. He is also an active member of the ANSI and ISO C++ standards committees.

This book consists of two parts divided in 18 chapters. The first part presents a roughly chronological retrospective of C++ development, starting from the early days of C with Classes till the first ANSI/ISO standards efforts and commercial widespread of C++ compilers. Part II describes the post-Release-1.0 development of C++. This part does not follow the chronological order, but is organized around major language features. The chapters, sections and subsections are organized in such a way that the book can be read in different ways. If you are not interested in details, then you may skip all subsections in which concrete details are dealt with; after getting acquainted with principles and generalizations, you can look at the details. To make the reader's life easier, cross-references to these tiny subsections are extensively used.

The introductory chapter *Notes to the Reader* presents the organization of the book, gives a guide how to read the book and gives a concise evolution of object-oriented programming languages. It also contains an extensive list of references quoted throughout the book.

In Chapter 1 *The Prehistory of C++* the author reveals the initial problems he has faced while writing system software during his doctoral studies in England and after getting a job at Bell Labs. This experience initiated the idea of a new language for system programming, based on C. In addition, the author presents his general background, including his European education and interest in philosophy, which had a strong influence on the development of C++.

The following chapter describes the birth of C with Classes, the forefather of C++. It explains in detail the main features provided in the initial 1980 implementation, like classes, derived classes, access control, constructors and destructors etc. The author also gives the arguments why has he chosen C as a base for a new language ("As close to C as possible — but no closer").

The Birth of C++ chapter describes what happened after the "medium" (in the author's own words) success of C with Classes. Major additions to C with Classes were virtual functions, operator overloading and improved type checking. This chapter also presents the evolution of Cfront compiler, which enabled C++ to become widely spread on different platforms, including even PCs.

I found the fourth chapter *C++ Language Design Rules* particularly instructive. It exhibits the basic principles and general rules which led the author during the language development (e.g. "Don't get involved in a sterile quest for perfection", "Always provide a transition path", "Don't try to force people").

A short Chapter 5 (*Chronology 1985-1993*) is focused on three main events of that period: Release 2.0, *The Annotated C++ Reference Manual*, and the start on ANSI and ISO standardization. The problem of standardization is particularly analyzed in the sixth chapter, which includes a very intriguing section *How does the (ANSI) Committee Operate?*, and an interesting subsection dealing with international character sets.

While most of the book deals with events "under the hood", not seen by public, the seventh chapter *Interest and Use* is dedicated to the growing public interest — conferences, journals and commercial C++ compiler releases. There is a very interesting section in this chapter dealing with the problem of teaching and learning C++. This section could be attractive to all of those who doubt if C++ is the language they could learn to use. The author provides the answers to common questions like: "I don't know C or C++, should I learn C first?" or "How long does it take to learn C++?".

The *Libraries* chapter describes one of the most powerful features of C++ (as well as C). As the author states, people more then often realize that designing a library is better than adding a language feature. They can be coded in languages other than C++, to provide high performance operations. Standard and some specialized libraries (e.g. numeric libraries) are illustrated throughout the chapter.

Chapter 9 *Looking Ahead* is more speculative and relies on author's personal opinions and

generalizations. He is searching for answers to some of his doubts like: "did C++ succeeded at what it was designed for", "what should and could have been different", "what should have been left out" and "what should have been added", "what was the biggest mistake".

Chapter 10 *Memory Management* opens Part II of the book. The problem of memory allocation and deallocation is studied. A special emphasis is put on the problem of garbage collection. Stroutstrup reveals that C++ was deliberately designed not to rely on automatic garbage collection in order not to gain time and space overhead. Therefore, the programmer has to implement some memory management scheme himself.

Operator and function overloading (described in chapter 11) is one of the most popular features implemented in C++. It allows multiple function instances that provide common operation on different argument types to share a common name. Problems involved in the use of overloading, like type-safe linkage, are briefly discussed in this chapter.

Following chapters describe features like multiple inheritance, abstract classes and static member functions. The closing four chapters are dealing with "major"extensions of C++: run-type information templates, exceptions and namespaces; what makes them major is that they affect the way programs can be organized. The lack of these features usually lead to difficulties in maintaining a consistent high level of abstraction. In particular, templates allow efficient, compact, and type-safe C++ container classes to be designed and conveniently used.

The *Design and Evolution of C++* is a unique book in many aspects (not to mention the symbolic cover photograph of a little oak sprouting in the shade of a fallen old tree). It is the definitive insider's guide to the design and development of the C++ programming language. As the author himself reveals, this is not a book to learn programming in C++ from. The primary aim of this book is to give C++ programmers a better idea of the background and fundamental concepts of the language. It should also be read by experienced programmers and students of programming languages to help them decide whether C++ might be worth their while. And, of course, this is a must-book for everyone involved in teaching object-oriented programming languages. The recognizable Stroustrup-style, with instructive and often amusing sentences or citations in front of each chapter, and lots unrevealed details yet will engage any reader's attention.

Julijan Šribar
Faculty of Electrical Engineering and Computing
University of Zagreb
Zagreb, Croatia

Hassan Gomaa

Software Design Methods for Concurrent and Real-Time Systems,

(SEI Series in Software Engineering)

Addison–Wesley Publishing Company, Reading, Massachusetts, 1993, pp. 447, ISBN 0-201-52577-1.

With the massive reduction in the cost of microprocessors and companion microelectronic devices over the last decade and the immense increase in their performance, concurrent, real-time, and, in the most general case, distributed real-time microcomputer-based systems have became a cost-effective solution to many control problems. There are few books oriented toward the design and development of large-scale concurrent and real-time software systems that are the crucial components of any solution. This book is one of them. It outlines the characteristics of concurrent and real-time systems, describes important concepts, and surveys and compares a number of software design methods. It then describes in more detail two related software design methods and gives an illustration with a number of case studies.

The book is divided in four parts. First part describes overall design concepts and focuses on particular topics relevant to real-time systems such as concurrent tasks, information hiding, object-oriented concepts, and finite state machines.

Second part surveys several software design methods (Real-time Structured Analysis and Design, Design Approach for Real-Time Systems — DARTS, Jackson Systems Development, the Naval Research Laboratory Software Cost Reduction Method, Object-Oriented Design). The concepts on which each method is based are described first. After a brief description of the notation, the steps involved in using the method are explained. Since the methods are best understood by studying an example, the same example, namely an automobile cruise control and monitoring system, is used for explanation of all methods. The methods are compared from the perspective of how they address the important design concepts.

Third part provides a detailed description of two software design methods: Ada-based Design Approach for Real-Time Systems — ADARTS, and COncurrent Design Approach for Real-Time Systems — CODARTS. Finally, the fourth part presents a number of case studies. These are an Automobile Cruise Control and Monitoring Systems, Robot Controller, an Elevator Control System, and a distributed Factory Automation System.

The author's writing style is clear, straightforward, and supported by many figures and charts. He has over 25 years of experience in software engineering and he was personally involved in the development or modification of some of the described methods. Therefore, his recommendations can be very valuable for the designers and developers faced with the challenge to specify and design interactive, concurrent, real-time systems. Anyone with a serious interest in this field will find this book very useful.

Leo Budin
Faculty of Electrical Engineering and Computing
University of Zagreb
Zagreb, Croatia