Promotional information about McGraw-Hill book

Visual Basic Developers' Toolkit:
Performance Optimization, Debugging, Rapid Application Development, and Distribution

Call 1 (800) 2McGraw (McGraw-Hill) for more information, pricing and availability.

Visual Basic is a real development environment used to create well-known and best-selling Solomon Software financial modules, Quicken 4.0 and Quick Books 3.0, MicroHelp Uninstaller. It is also a serious rival to PowerBuilder, CA Visual Tools, IBM Visual Code, and 4GL toolkits. Many small-distribution applications have been created with VB, for both internal-corporate or external sales. The integration of Microsoft Office with VBA and the office productivity toolkit will expand the possibilities for rapid customized application development and the need for knowledge and skills to do so.

The important points, and these are perceived as hard topics or as too tough to write, include the four key terms in the book's subtitle, namely performance optimization, rapid application development, debugging, and distribution.

People in the VB development environment, which I want to write to, want to know how to manage the whole process, ensure its success, handle design from prototyping to application rollout and those corresponding upscaling problems. VB is suitable for heavy-duty client/server applications, perhaps both better and worse than PowerBuilder, for many reasons. In the same way, Borland Delphi provides a means to respond to upsizing and performance issues perceived with VB (time code created with each). There are many tools available to supplement VB, yet most developers do not have the time to evaluate or integrate them into their development process. Many do not think how they can justify the cost for these tools and create effective budgets and process planning. Additionally, there are important packaging issues related to platform, DOS and GUI releases and versions, .DLL and .VBX versions, and compatibility which can create problems for those inexperienced developers and even experienced ones.

Additionally, one of the serious limitations of RAD or application prototyping is that the sample is unfortunately rolled out as a production solution—this creates problems with performance, upgrading, maintenance, and stability. Source code control and automatic testing tools become very important. Also, there are a few tools which convert VB resources into C and C++ or Delphi for multiplatform distribution, and there will be more as VB grows in popularity. Furthermore, VB version 4.0 is forthcoming in third or fourth quarter in 1995, and there is growing body of third-party tools for developing TAPI, Lotus Notes, SQL front ends, spreadsheets, and graphics. It is possible to convert VB into C by recycling objects and resources, or convert VB into Borland OWL and Delphi95. Developers will be looking at converting 16-bit applications with VBXs into 32-bit versions with OCX code. These represent optimization topics important to many corporate and professional developers alike.

Conversion and distribution is important when one considers the desire to take a primitive prototype and expand it into a working product or take what is a single-user application and make it multi-user or at least reentrant within a multitasking environment. It is nice if a demo (the skeleton) can be converted into the complete application without starting over. Other issues include memory management, record locking, platform issues, version and upgrades, and performance tuning that are critical for client/server applications in the extended WAN environment.

I want to address application development process from the standpoint of using VB to get applications up and running quickly, Very few books, even C books relate how to:

- create a packaged product
- deal with deadlines and production pressures
- develop in a team environment (with code control)
- get the most out of the tools
- use CASE tools and prototypers to qualify a design
- design a referential database with design tools
- using VB for testing interface concepts
- using VB for improving testing workflow and utility
- how to benefit from RAD and avoid its pitfalls
- how to design an application for performance (and what to avoid)
- how to debug code, APIs, and user-interface errors
- how to solve EXE and DLL problems
- how to use VB debugger, breakpoints, Debug.Print, code spy
- how to trace triggers, cascades, excess baggage, and eliminate them
- maintain code libraries for object and code reuse

Although there are a number of good books aimed either at new programmers, programmers new to VB, or to experts who want to learn the newest tricks, they still generally cater to the lowest common denominator; they present either a lot of basic material, present API calls and bells and whistles which do not really relate to the mainstream corporate development process where development is focused to customer applications rather than mass-market ones, or are imbued with arcane tips in an encyclopedic-style without thesis or focus. After acquiring a first basic Visual Basic book, I think everyone seems to want a real book for "experts" like themselves. The books by Mitch Waite represent the best resources, but I think they leave a lot of gaps. The few other high-end books are very disappointing, such as Visual Basic Database Programming, by Karen Waterson, a promising book which seems to me to lack real utilitarian content. It is rises above the code with the result that it does not address coding or design issues.

Of note though because it seems contrary to my assertions, the more frequent downloads from Internet VB FAQ sites (where tracked) and CompuServe are the VBXs, DLLs, and VB extensions. While this speaks forthright to developers' interests in bells and whistles, it also addresses the need to ram through a application quickly with previously debugged modules. Such modules are supposedly time-savers, and the implication is that the programmers are desperately seeking to find faster ways to achieve their ends within resource limits.

RAD means different things in VB (and other languages, including rapid application decay and rapid application death with some good reason), I want to address as many of these as possible. Specifically, there are the issues of getting a skeleton up quickly, of replacing complex code (say with MCA API and many state conditions) with a simpler VBX. What are the trade-offs and how does the typical developer evaluate the tools? This is particularly relevant as networking and telecommunications merge to create viable CTI needs with phone, e-mail, pagers, fax, scanning, IVR, Lotus Notes, and Internet distribution. These issues do tie together because the MS Back Office is providing the means to automate many workflows and production processes with VBA and standalone VB with OLE.

This is book for readers with problems to provide problem resolution and the techniques for it. Every developer runs into problems. I do, and there is a limited place to turn for help. CompuServe and Microsoft VB support is insufficient, particularly for debugging or optimization. The 3 month free support service is very limiting. The 900 and 800 fee-based Mastercard services do not debug and they are very expensive. (My solution has been to keep buying new copies of VB and Access because it is cheaper than the priority service fees.) When a programmer has a problem and a deadline, this is the books they will order Federal Express Urgent. Production and performance problems are incredibly expensive and critical to many organizations and no current books really tackle the professional needs of the programmer or their organizations. According to Microsoft at VBITS 94, they have sold 1,783,xxx copies of the VB language to date; they will not reveal if this is current version of VB WIN 3.0, or the sum total

of all outstanding VB WIN, VB DOS, and VB Office APP development kits. Nevertheless, that is a huge market that dwarfs even C language tools, and quite a market for developers with problems.

For example, one of the problems which typically faces me is the tradeoff between trying a new VBX or toolkit and doing the task the same old way. Not only is there an expense to the toolkit, more importantly, there are time and space constraints to each new tool. Specifically, I have to learn how to use the tool, evaluate it for effectiveness and balance the need for it against disk space and distribution disk space. I may review Internet news groups or CompuServe forums for samples, demos, shareware, or information about a commercial product. Yet, this takes a few hours, time which is very precious and the results are often open-ended; I may not meet my needs. Even commercial products create conflicts in terms of memory management, speed, performance, integration, and packaging. I learned this one the hard way when I converted an VB/MS Access 1.1 application and database to Access 2.0 Jet in order to provide referential integrity. Unfortunately, I realized only after I tried to distribute it that I needed the Access Distribution Developers kit and about 2.6 MB more for distribution. This killed the upgrade and forced me to revert to earlier code (via tape archives... thank god for that).

The disk or CD-ROM can include ScreenCAM or V4W demos to demonstrate how to do the tricky things which most people are adverse to reading or trying. VB is very useful for prototyping applications quickly as a matter of course, for demo-ing concepts and technical ideas, and for creating fully-functioning client/server applications. Because this development environment is so rich and has been extended with so many third-party tools, the typical person new to this environment or even quite an old hand at VB does not know all the methods and new tools to speed product development.

Chapter 1—Overview
An overview of the book explaining the purpose, audience, book content, and structure of book.

Chapter 2 —Visual Basic and Tools
A hands-on how to tour of Visual Basic in terms of what it can and cannot do, its limitations, how it can be extended, and what represents doable tasks. Reference here should be made to PowerBuilder, Delphi, C/C++, source code control, version control, Bricklen's Demo, project planning, CASE, and automated testing tools which integrate into various 4 GL and OODBMSs. This includes Lotus Notes development, ODBC, Progress/Crescent Stingray workgroup tools, CTI toolkits, graphics routines, and others. I will also include discussions on PinPoint, VB Compress, and various toolkits and add-on VBXs and OCXs. A reader will want to find out what works, what the alternatives are, and why they need the toolkits. Groupware issues are also important because of performance optimization, record or device acceSs, rollout, and replication.

Chapter 3—Visual Basic Performance and Design Errors
A hands-on view of VB performance flaws, including concept and design errors, reinvention of solutions which are built-into VB or libraries, and bad code. This Chapter will also detail common types of code errors and code bloating and what to do about them.

Chapter 4—Performance and Debugging Tools
This chapter details tools, including the built-in debugger, Windows heap tools, and profilers. There are at least five new and good profilers of VB performance and this Chapter will reference them. I hope to include several of these tools on the CD-ROM. I will profile other tools which help teams develop applications in tandem including source code control, X-ref, and third-party libraries of common functions. A discussion of various metrics, which to use, how to implement them, and how to get information from them.

Chapter 5—Design Issues
Designs that will work, won't work, won't scale, will create implementation, memory, runtime, disk

space, file space, access time, and user problems... and how to avoid the traps. User interface issues, including reference to Win 95/Win NT interface, tabs, popup help, tips, status bars, context sensitive help, order. I may also include issues relating to graphics, sound, and control of device drivers. WinG, WinMM, and Win32s may also be relevant, depending upon the state of the new VB 4.0 and Win 95 platforms.

Chapter 6—Multi-developer/Multi-user/Multiplatform
This chapter explains source code control, code libraries, function libraries, modularization, and the requirements for creating applications that run as multiple copies of the same application, as multi-user applications, as multi-user, shared database applications, and applications which can be ported across platforms. Reentrancy, heaps, record locking, table locking, transactions, compiled SQL statements, are also important. Similarly, some of the issues required for successful group design and coding will be addressed here. Another interesting approach is conversion of VB and VBA applications into DLL and C/C++ standalone applications through resource and form object conversions.

Chapter 7—Application Tuning
A detailed Chapter showing the reader how to tune code, solve problems in better ways, resolve event cascades, stack errors, and program so that everything in VB is not a trigger. I will show samples of code with elapsed runtimes, several versions of code that trade more memory for faster processing, and uses of API's for improved performance.

Chapter 8—Limited Distribution Products
A hands-on view of using VB for corporate projects, from the simple, to complex implementations with DDE and OLE. I also want to include issues related to client/server front end development, access to host data, special tools for CTI and other real world communication.

Chapter 9—Rapid Application Development
The pros and cons of this philosophy, how it differs from waterfall design, and conflicts with the best intentions of object-oriented technology. What RAD really means, how you can utilize it, and what tools integrate into the development process. How the previously-defined planning, control, CASE, DB design, toolkits, report writers, and automation tools improve the accuracy of development, help a team build a good product, maintain deadlines, and automatically generate skeletons for modules, data sets, and help files.

Chapter 10—Debugging
Code debugging with debugger, spy tools, message capture tools, PinPoint, X-ref, Compress, and other similar tools. Once these critical basics are explained, I will show how to automate testing with various explorative and iterative macros-type tools.

Chapter 11—Distribution
Testing to conformance of concept. Builds. VB installation wizard and the templates, and how to alter them as needed. Other disk build tools and installation routines. Some thoughts on shareware, freeware, registrationware, and application locks. Distribution and installation of VBXs, OCXs, DLLs and common problems, and how to get around them.

Chapter 12-Help Desk Support
Tracking and tracing bugs, Internet Support, on-line help, on-line Help desk, bug issues, data backup, data conversion, and installation support. Possible video driver, chip, memory, GDI, or USER resource issues; solutions for them.

Chapter 13—Code Examples
Examples of applications and modules which I will develop for various common functions. One of the serious limitations I find with the wealth of options is that I do not have the time to evaluate what works and how it affects the overall product when I am in the midst of development. Clearly, this is true for every other developer or prototyper. In so far as most books show how to read

the .INI file, or create forms, or perform some bell and whistle, I want to show the reader how to get as much accomplished within the shortest possible time. This chapter will show how some tools create instant forms from a database, create referential code, and automatically generate the code to open, create new, backup, and copy a database structure. Some of the tools for example, generating dialog boxes are primitive time-savers, and not really within the scope of ensuring project success. If at all possible, I will use AVI or ScreenCam movies to actually demonstrate the techniques; seeing is believing and also sometimes the showing a person how to actually create a referential link is better than trying to explain with enumerated steps.

Chapter 14—The CD-ROM
CD-ROM or disk will include: (I have not worked out the sizes, but this looks to be at least 60 MB)

- Skeletons
- code samples
- small specialized applications
- ScreenCam movies of various techniques, toll usages, and methods
- hypertext versions of book
- demos of various tools
- selected shareware for various VBXs, OCXs, and tools

Author's Bio
Marty Nemzow has written a number of best-selling books for McGraw-Hill, including the Ethernet Management Guide in its third edition, Implementing Wireless Networks, and a series on tuning computing environments: LAN Performance Optimization, Computer Performance Optimization, and the newest, Enterprise Network Performance Optimization. His company, Network Performance Institute (Miami Beach, FL) provides enterprise network design and improvement consulting services, and markets capacity planning and network configuration software tools.

