

■ PETER T. ITTIG, Feature Editor, College of Management, University of Massachusetts, Boston

The BASIC programming language (originally the “Beginner’s All-purpose Symbolic Instruction Code”) evolved over the years into more powerful tools including the later Microsoft versions “QBASIC” (Q for Quick) in the DOS environment and more recently “VBasic” (V for Visual) in the Windows environment. An early advantage of BASIC was that it was often included in PC’s. Today, a version of Visual Basic is included in Excel as “VBA” (Visual Basic for Applications). In the following guest article, Dr. Brett McKenzie compares three texts for the latest incarnation of this evolving programming tool, Visual Basic.NET.

BASIC Update—Approaches to Teaching VB.NET

by W. Brett McKenzie, Roger Williams University

It has been said that a discipline derives from a set of laws and theories as does physics, which builds on Newtonian mechanics and incorporates theories of relativity. While the debate about the grounding theory rages in CIS/MIS, it remains a field of study, like education, rather than a discipline. This gives CIS/MIS the flexibility to borrow from many fields and the ability to adapt quickly to changes without the disruption of a paradigm shift. This advantage, however, can make it challenging to select texts and develop an agreed upon instructional model because the absence of theory validates many approaches. Consequently, texts in the field reflect the biases of the discipline from which it was drawn. The texts selected for this review represent a range of views from a computing science heritage, through a software engineering perspective, to a business case model.

The release of VB.NET by Microsoft and its cessation of support for Visual Basic 6.0 has complicated teaching introductory computing courses. Visual Basic, an outgrowth of Quick Basic and a cousin of True Basic, can trace its heritage to BASIC, the language developed originally at Dartmouth College when it was believed that all students in all disciplines should have exposure to computing. Through the variants of BASIC, Visual Basic has developed into one of the world’s most popular computer languages and introduced a number of people to the frustrations, joys, and

power of producing software instead of merely consuming it.

The .NET initiative, which includes VB and C#, as well as many other languages, seems to be Microsoft’s response to the radical transformation of computing in the 1980’s with the deployment of graphical user interfaces and the demonstration of the power of object-oriented programming coupled with the subsequent deployment in the 1990’s of “write once” code with Java and the meteoric rise of computing enabled by the Internet. These changes in software development and deployment served as a wakeup call to Microsoft, which has retooled its languages and their deployment through the .NET framework. In briefest terms, the .NET framework creates a common base to allow a programmer to mix languages within the same program while it integrates database with the World-Wide Web technologies through open standards, improves the packaging of programs to ease their deployment and installation, and allows for easy extension to alternate computing platforms such as personal digital assistants and smart phones. Along with the development of the new language has been a new version of Visual Studio.NET, the Integrated Development Environment (IDE), which allows for the coexistence of multiple language references within the same programming project and manages language sensitive interactive editors and help sessions.



W. Brett McKenzie

serves on the faculty of the Gabelli School of Business at Roger Williams University in Bristol, Rhode Island, where he specializes in introductory computing courses. Among his research interests are portable computing, the social dimensions of technology, and the uses of computers as communicative tools. Prior to his faculty appointment, he worked in academic computing at Bryant College and Brown University, where he focused on technology integration across the curriculum. He holds a bachelor’s degree from the U.S. Naval Academy, a master’s degree from Harvard University, and a PhD from Clark University.

wmckenzie@rwu.edu

The changes to Visual Basic to make it VB.NET are significant. First, the language has become fully object oriented. It is now more rigorous and similar to other higher level languages such as C++ in that many of the "shortcuts" or "forgiveness" have been eliminated. For example, controls no longer have default properties just as variables must be properly declared, eliminating the Variant type. Secondly, error handling has been improved with structured error handling through "Try...Catch" blocks. Finally, all the code generated by the application can now be examined in the editor. This is invaluable in teaching as one can demonstrate how setting properties in the graphical development environment changes the code associated with the programmable objects. Similarly, a console application has been added so that pure code may be run and tested without developing a graphical interface. In short, an instructor can now show students how to build a complete Windows application in Visual Basic using only a text editor. The release of VB.NET presents new challenges to an introductory computing course as shown in the reviews of the texts below.



**Visual Basic.NET
How to Program,
Second Edition**
Harvey M. Deitel, Paul
J. Deitel, Tem R. Nieto
Prentice Hall, 2002,
1517 pages, \$85
www.prenticehall.com

THIS IS THE MOST COMPREHENSIVE TEXT from a company with extensive experience developing texts specifically for computing education. The text follows the model developed in the How to Program series, which includes C++, Java, and other popular languages. Similar to other Deitel texts, this one can serve as a reference for the language with clear, concise code examples of about 15 lines to illustrate the function being explicated. The text covers Visual Basic itself and the new developments of the .NET environment including XML integration, ADO.NET for database connectivity, ASP.NET for interactivity via the Web, and Web Services, the distributed application architecture for the Web.

All texts in the Deitel series begin with a similar overview of computing, placing the target language in the larger context of computing. It then introduces the fundamentals of structured programming, implementing the features of the particular language, followed by a lengthy discussion of the Object-Oriented programming approach. This includes chapters specifically devoted to the foundations of object-based programming, inheritance, and polymorphism. The approach to demonstrating the language is to introduce one feature and model it with an example, then build upon that example with variations. For example, a simple averaging of student grades builds from computing a numerical average to creating a pass/fail grade as an output based on computed scores. Reflecting its computer science heritage, the text begins with a console application and the classic "Hello World" program where the computer responds with "Hello World" when the program executes. Similarly, the computing heritage relegates graphical controls, a signature feature of Windows programs, to a later introduction in Chapter 12.

The text is richly produced with excellent graphics and numerous screen shots. These are particularly valuable because VB.NET uses VS.NET, the Visual Studio graphical integrated development environment that can be confusing to novices. The code samples are also colored to correspond to the colors used in VS.NET, which makes comparing code with the text samples easy. To supplement the examples and discussions, Deitel and Deitel use brief call outs clustered in groups such as Good Programming Practice, Common Programming Errors, and similar observations. These hints show that programming is both an art and a science.

The strength of the text lies in its complete treatment of Visual Basic. This strength, however, is also its weakness as it is too much for a single semester with novice programmers. Additionally, many of the problems are more related to computer science and applied mathematics, such as demonstrations of Fibonacci series or mathematical algorithms for encryption and decryptions that business students may find too esoteric.



**An Introduction to
Programming with
Visual Basic.NET,
Fifth Edition**
David I. Schneider

Prentice Hall, 2002,
736 pages, \$70

www.prenticehall.com

THIS IS THE MOST FOCUSED OF THE TEXTS reviewed. Consequently, it lends itself easily to a single semester introduction. Schneider even offers guidelines for a short course that skips all of Chapter 2 (Problem Solving), Chapter 8 (Sequential Files), Chapter 10 (Database Management), and Chapter 11 (Object-Oriented Programming). He alone, however, includes an appendix, with the misleading title, "Converting From Visual Basic 6.0 to VB.NET." This appendix does not explain conversion or migration of legacy VB 6.0 code, a problem within the VB community, but compares the differences between the two languages as they appear in his text. This appendix is helpful for an instructor who is familiar with VB 6.0, but may not help a student new to programming, which seems to be the target audience for the text.

Schneider's approach differs from Deitel in being a textbook that presumes less prior exposure to programming, little knowledge of computers, and less comfort with the mathematical foundations of computing. The text can be used with true novices and with students who are interested in exploring programming. For example, the text begins with an introduction to Windows and quickly introduces features that appeal to students, such as list boxes and buttons. These basic controls give students the sense of writing "real programs" rather than completing mere exercises.

The chapters are structured into brief, focused segments followed by a series of short word problems and exercises that ask for brief answers or short programming examples. The text includes answers to the odd problems at the end of each section, which is a familiar model for students to encourage self-study and review. The chapters conclude with programming projects that incorporate each of the chapter aspects into a more complex program. The solutions to these projects are not included in the text and their descriptions, like Deitel's,

are terse, single-paragraph statements, sometimes with a screen shot of the user interface. As the student skill level develops, Schneider introduces case studies in the middle chapters for a payroll, a loan, and a checking account. These more elaborated and well-discussed cases make up for the very brief descriptions of the end of chapter programming projects. Similar to Deitel, Schneider presents code snippets in a full color outline, following the conventions within Visual Studio. He, however, has many fewer examples and directions for navigating the development environment.

Given the defined scope of the text, one does not get the sense of the breadth of VB.NET. For example, Schneider ignores the console application, which I find a particularly valuable starting point as many new students have never seen a command prompt because their experience with computers is only through a graphical interface. As one of the first texts available for teaching VB.NET, being released shortly after Microsoft released .NET, the object-oriented and .NET chapters seem to be "bolted" onto an existing text. While there is support from the publisher's Web site, there is little connection to supporting material that may be at the Microsoft site and few references to programming practice in the larger programming world.



**Programming with
Microsoft Visual
Basic .NET: An
Object-Oriented
Approach—
Comprehensive**
Michael V. Ekedahl,
William Newman

Thompson/Course
Technology, 2003

664 pages, \$53

www.course.com

SIMILAR TO DEITEL &

DEITEL, this text addresses the major components of the .NET environment, which includes the programming environment of VB.NET, database programming with ADO.NET, and Web integration with ASP.NET. The final chapter focuses on application deployment. Interestingly, of all the texts, this is the only one to address this

area, although easing deployment of Windows applications was a major reason for development of the .NET framework. While the text includes these components, it seems to be built more upon the VB6 text than being a rewrite for the new environment. There are some teaching notes which would be helpful to include, especially in the Web services area, which can be highly dependent on local setup for development. The author, however, is accessible through the publisher, Course Technologies, to provide clarification and support for these questions.

This text has a different organization from the other two. Each chapter begins with the presentation of a completed solution that introduces the components of the chapter in a working model. The chapter then steps through the process of building the presented model. Each block of code, and sometimes each line of code, is dissected and commented on. Unlike the other two texts, all the code snippets and screen shots are in black and white.

Following the chapter model and analysis, there are review exercises asking for short answers, programming questions that focus on the programming environment, and hands-on projects. These projects provide good practice for the students and usually present a meaningful context for the elements of the chapter. Many are associated with business problems, such as depreciation and amortization, cash registers and billing systems, and bank or checkbook transaction systems. These problems are richer and more detailed than the other texts and include significant hints or scaffolding to complete the problem. For tentative programmers, this detail can be helpful, while, for those with experience or those who take effortlessly the thinking behind programming, the hints can be confining.

The biggest limitation of the text is its difficulty in serving as a reference because the presentation is so tightly coupled to the cases under review. Some students gain the impression that the elements are used in a restrictive sense only as in the text examples. Some faculty in upper division courses note that students become bound by the text, being able to complete the assignments, but becoming confused when confronting novel situations. The text also treats the object-oriented environment very lightly

and is highly pragmatic, being more focused on providing solutions than explicating the theory.

Conclusion

As expected from computing texts, each of these texts has extensive material on an included CD-ROM, with Deitel providing the most supplementary information as well as the most comprehensive support via the Web. Interestingly, each text relegates debugging to an appendix. Given the strength of the debugging tools in Visual Studio and the high ranking of debugging skills in a local survey of employers, it is surprising none of these texts has integrated debugging into the discussions and problems more fully.

School populations and faculty differ so the approaches taken in each of these texts may work in one environment and fail in another. For example, in a program with a strong undergraduate emphasis in mathematics and students with programming experience in high school, Deitel & Deitel might be most appropriate. In another setting with extensive lab time, such as a course taught completely in a computer lab, Schneider might be best as the course structure makes up for the lack of images in the text. Finally, in a program where students will be introduced to programming but may not enroll in other programming courses and need a sense of the scope rather than the depth, then Ekedahl will provide that. These three texts represent only a small range of the available instructional materials and every program should be able to find texts and an approach to match their student needs and instructional goals. ■

Peter T. Ittig
College of Management
University of Massachusetts
Boston, MA 02125-3393
http://www.faculty.umb.edu/peter_ittig/