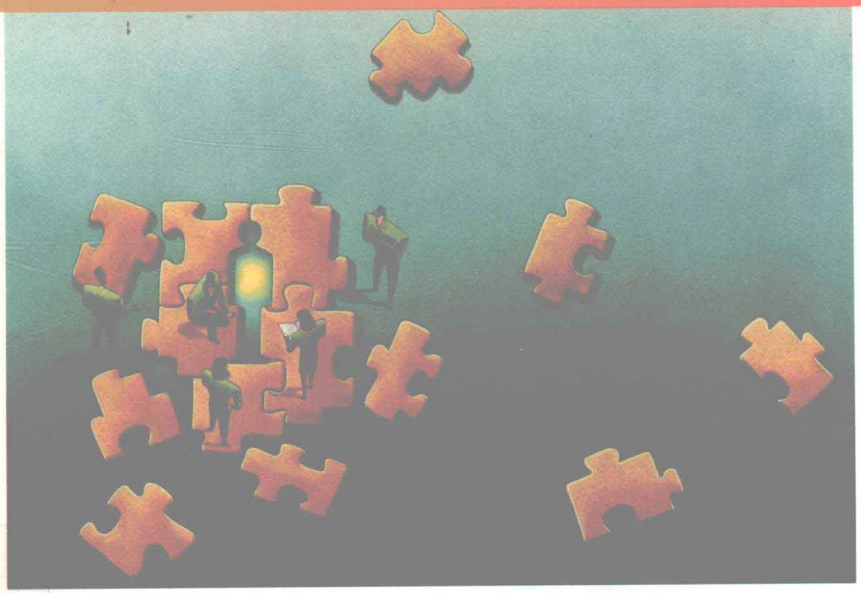
VISUAL BASIC®:

A Programmer's Guide To Managing Component Based Development

How Visual Basic can be used to drastically improve the development life cycle



ERAN MAROM

VISUAL BASIC

A Programmer's Guide to Managing Component Based Development

Eran Marom

To join a Prentice Hall PTR mailing list, point to:
http://www.prenhall.com/register

Prentice Hall PTR, Upper Saddle River, NJ 07458
http://www.prenhall.com

Library of Congress Cataloging-in-Publication Data

Marom, Eran.
Visual Basic : a programmer's guide to managing component based development / Eran Maron
p. cm.
Includes index.
ISBN 0-13-591504-X
1. Microsoft Visual BASIC. 2. Computer software--Development.
I. Title.
QA76.73.B3M3744 1996
005.265--dc20 96-36077
CIP

Acquisitions editor: *Paul Becker*
Editorial/production supervision and interior design: *bookworks*
Marketing manager: *Dan Rush*
Manufacturing manager: *Alexis Heydt*
Cover design: *Design Source*
Cover illustration: *David Chen/Stock Illustration Source, Inc.*
Cover design director: *Jerry Votta*

© 1997 by M&D Advanced Systems, Inc.
Published by Prentice Hall PTR
Prentice-Hall, Inc.
A Simon & Schuster Company
Upper Saddle River, New Jersey 07458

The publisher offers discounts on this book when ordered in bulk quantities.
For more information, contact:
Corporate Sales Department, PTR Prentice Hall
One Lake St.
Upper Saddle River, NJ 07458
Phone: 800-382-3419 FAX: 201-236-7141 E-mail: corpsales@prenhall.com

For information:
Eran Marom, M&D Advanced Systems, Inc.
161 Hilburn Road, Scarsdale, NY 10583
(914) 722-4241
EMarom@AOL.COM

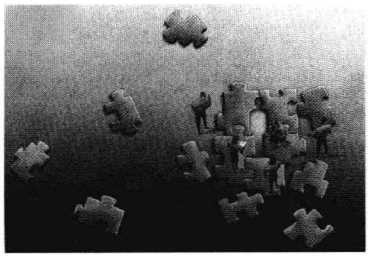
Printed in the United States of America

10 9 8 7 6 5 4 3 2 1

ISBN 0-13-591504-X

Prentice-Hall International (UK) Limited, *London*
Prentice-Hall of Australia Pty. Limited, *Sydney*
Prentice-Hall Canada, Inc., *Toronto*
Prentice-Hall Hispanoamericana, S.A., *Mexico*
Prentice-Hall of India Private Limited, *New Dehli*
Prentice-Hall of Japan, Inc., *Tokyo*
Simon & Schuster Asia Pte. Ltd., *Singapore*

In Memory of My Father



Preface

This book is a practical guide on how to fully exploit the features of Visual Basic in order to achieve drastic improvements in the project life cycle. Such efficiency is possible with Visual Basic since it is a component-based language. Thus, developing in VB better resembles an assembly project than the more traditional craft-work of third generation languages. But in order for this paradigm shift to materialize into real cost and time improvements, the method by which one goes about managing a software development project must change. Following the real-life, pragmatic recommendations in this book will provide the reader with the tools to carry through such a transformation.

The first few chapters of the book introduce the Visual Basic language and development environment, and contrast them with those of some of the more traditional languages. Special attention is paid to the difference between the component-based technology displayed by VB and the object-oriented approach of languages such as C++. This discussion is combined with everyday analogies to other industries and situations where assembly techniques had proven very successful.

Next, the book turns to some more elementary aspects of project management, such as scheduling, budgeting, and staffing. For each item, the discussion first demonstrates how Visual Basic affects the function and then suggests ways that best take advantage of the language to reap the most benefits. Specifically, this section answers questions such as "How to justify a VB project?," "How to decide which project to pursue?," "How to schedule a VB

project?," "How much to budget for?," "How many people to hire?," and "How to best organize the team?"

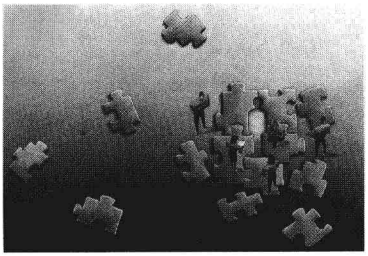
Moving slightly away from the nuts and bolts issues and toward the more touchy–feely aspect of management, the book next deals with management style, motivational techniques, and productivity measurements. As with the previous items, these aspects of project development are also affected by the rapid pace at which Visual Basic projects are carried out and, thus, must be modified accordingly. This part of the book is concluded by summing up the development approach through the definition of a new development model.

For the remainder of the book, the discussion moves back and forth between the logical steps of the development life cycle—specifications, design, development, testing, and implementation—and the building blocks that are pertinent to each step. The latter is covered on a technical level, but with a strong emphasis on managerial implications. Among such technical issues is a chapter dedicated to the selection and usage of client–server databases as a component in the Visual Basic environment. Another chapter covers the process by which reports are generated with VB. Optimizing applications is the subject of yet another technical discussion. Finally, client–server application security is looked at extensively.

No book on component-based technology would be complete without defining Open Database Connectivity (ODBC), Object Linking and Embedding (OLE), and Dynamic Link Libraries (DLL). Sure enough, these three subjects each have their own chapter that specifically points out their usage in the VB project.

Two standards guidebooks are included: One instructs on the Visual Basic coding conventions, the other is a sample of a graphical user interface manual.

The material in this book represents precisely the type of information I wish I had access to when I first encountered Visual Basic and, later on, as I learned to fully exploit its advantages. It is sure to help you jump start the learning curve if you are a Visual Basic novice, or climb it more easily if you are a veteran.



Contents

Let's face it: Managing large systems development in today's *give-it-to-me-by-yesterday-but-don't-ask-me-to-pay* environment is just no fun anymore.

Alas, help is here! Thanks to the ingenuity of the Microsoft development team and the foresight of its founder, Bill Gates, a specific component-based language known as Visual Basic can and will, if used correctly, return the fun to your job. So promising is this product that on every package of Visual Basic the following familiar inscription should be prominently displayed:

> Give me your poor, your tired, your huddled masses. . . .

Or maybe it should read:

> Give me your cost-conscious companies, your frustrated MIS directors, your army of programmers. . . .

Indeed, it has long been kept a secret that Visual Basic increases programmers' productivity by an order of magnitude never before experienced in data processing. This is because Visual Basic virtually eliminates the need for substantial chunks of specifications, coding, documentation, and testing. So powerful is this new development environment that a project developed using Visual Basic and one that deploys the language's full capabilities can cut a project's costs by as much as a half compared with conventional development (see financial analysis at the end of this chapter).

But to gain these enormous benefits data processing managers must change the way their groups develop applications. After all, one cannot develop tomorrow's systems using yesterday's methodologies. Take, for instance, the hardware selection process. Visual Basic is a language developed for the personal computer and PC networks. Therefore, in order to take advantage of Visual Basic you must migrate a substantial portion of the design, development, and installation from the mainframe world to the personal computer network world and from UNIX to Windows.

It is not my intention to devote much space to argue that PCs and Windows are more suited than mainframes and UNIX for meeting the current and future needs of the business community. Doing so would be beyond the scope of this book. I will, however, point out that while the old guard is struggling to develop ever more complicated systems on hardware and operating systems that are ill fitted to meet users' needs, their competitors are using Visual Basic (VB) to develop PC systems in a fraction of the cost and time. At current trends, those who refuse to change and join the PC world may end up arguing the merits of a mainframe system or a UNIX operating system from outside the data processing field.

Don't believe me? Listen to this one statistic: "CFOs surveyed by Deloitte & Touche and Hyperion Software Corp. said that 82% of the new accounting systems that they are installing will be on 'client-server platforms'. . . . James Perakis, president of Hyperion, based in Stamford, Connecticut, said that the annual spending for client-server networks will grow from today's $1.5 billion to $5 billion in 1999. 'Client-server accounting net-

works are much more flexible and easy to change and maintain than mainframes,' he added."[1] This is an amazing statistic considering that we can hardly get 82% of any population to agree on what day it is, let alone on what technology to use!

And if you think PCs do not provide the same level of performance as your mainframe, or that Windows is not as robust as UNIX, think again. Even the smallest of today's PCs are faster and better than the mainframe on which many older business systems were originally developed.

But if you are one of those who is still not convinced—and surely I could not sway you from your faith in just two paragraphs—note that I am not at all advocating disposing of the mainframe. I am merely suggesting that your trusted machine may better serve you as, say, a data warehouse than it would as the full-blown information system it was originally designed to be. So you are hereby invited to take your user interface, electronic mail system, spreadsheets, data processors, accounting systems, and payroll systems off of that old contraption and move on down to the PC. Welcome to the late twentieth century.

Rule #1: Don't let yesterday's technology stand in the way of today's decision-making process or you will be unable to benefit from component-based languages.

The second change that you must make requires you to forget everything you know about programming and the management of systems development. This coerced amnesia is necessary since, if you attempt to manage a VB project the way traditional software projects have been developed, you risk losing all of the productivity improvements that could otherwise be gained. Instead you must learn a new methodology that better fits the rapid application development (RAD) paradigm.

Rule #2: To capture the benefits of a component-based system development, you must change your management methodology.

The rest of this book details this new management methodology starting with justification for using VB on through the development cycle and right up to system deployment.

WHO IS THIS BOOK FOR?

By now, some of you are probably reaching for a firearm and dialing frantically to inquire my whereabouts. After all, my contention that (a) Visual Basic is the greatest thing in computers since, well, computers, and that (b) mainframes are, at best, peripheral technology, is hardly popular among computer professionals. Yet, before you attempt to inflict any bodily harm on

[1] *The Wall Street Journal*, Dow Jones, Inc., March 21, 1996, p. B6.

yours truly, please understand that this book is limited in scope and by no means attempts to address the entire data processing world. Some of you, for example, will find both Visual Basic and the material in this book so helpful that you will not know how you got along without such wonders. Others might as well return this book to the counter and ask for their money back. Still others may fall somewhere in between the two groups, receiving some benefit from the methodology this book advocates yet finding no practical use for Visual Basic, or vice versa.

To save you the aggravation of reading the first 200 pages to discover which group you may belong to, I have prepared a chart of how different users may benefit from the material of this book:

If you are . . .	You will . . .	Because . . .
A corporate data processing manager or an aspiring manager who is building database-intensive systems to support such business functions as accounting, inventory, sales, finance, payroll, personnel, and so on,	love Visual Basic and benefit from using the component-based methodology of this book	you could save buko bucks, deliver your systems in no time flat, and become a folk hero among your peers.
A consultant or systems house manger who specialize in data-intensive systems,	love Visual Basic, benefit from the methodology, but become very frustrated when your clients insist on using other technology	Visual Basic could allow you to earn the same amount of money you make now but spend much more time on the beach or golf course.
	Also you may find a lucrative niche market in providing custom software that does not fit within the component-based paradigm	there is always somebody who wants a very specific design and is willing to pay any price to get it. Remember, Rolls Royce is thriving by building cars the way they were built 50 years ago.
Component builders (the ones producing the objects used by VB),	agree with everything said in this book, but probably find the material useless. Booch's analysis and design methodology, and such languages as C++ may fit your needs better than VB	your work is more of a craft than an assembly. Although your products are being used in the assembly process you may have to resort to older technology to manufacture such parts.

If you are . . .	You will . . .	Because . . .
A data processing professional developing exotic systems such as artificial intelligent applications, space exploration modules, and so on,	find this book not very interesting	you are currently working on the cutting edge of software development. The components this book talks about will not be developed for your application for a year or two, by which time you will most likely be working on something even more advanced.
The average third-generation language programmer,	hate VB, hate this book, hate me, and hate Microsoft	VB and the component-based paradigm threaten to take away your craftsperson status and turn you into an assembly worker.
	nonetheless, you must continue to read,	you do not want to become unemployed.
The average "reformed" (read-VB) programmer,	benefit from the methodology and the management training	you probably already know about VB.

WHY DO WE NEED COMPONENT-BASED DEVELOPMENT TOOLS?

For as long as I have been involved in data processing, I have been hearing and reading about the ultimate data processing vision. I even started to believe that development managers dream only one dream: that someday, they would be able to build big systems by gluing together program objects, each answering a different functional need. In this dream, the programmer's job is transformed from that of a creative artist who crafts software systems to that of an assembly worker, slapping together interchangeable building blocks that are easily reused.

The desire to achieve this dream is easily understood: Software development is a labor-intensive, expensive, and time-consuming effort. Whether you are building an investment management system for your company's treasurer, an inventory system for the factory, or a payroll system for the accounting department, you must first spend thousands of person-hours specifying the needs of the system in view of the available hardware. You then have to hire some very highly paid prima donnas—read, software engineers—to program the system, expending a fortune on pay and benefits in the process. As a consequence, over the next few years, your job transforms into that of a wrestling match referee who must keep the users who "absolutely, positively must have a certain feature if you want us to use your damn system" from your program-