ABOUT THE BOOK

with Lawrence H. Putnam and Ware Myers

authors of

Five Core Metrics:
The Intelligence Behind Successful Software Management

328 pages softcover $49.95 ppd.

DHQ: What are the “five core metrics” featured in your new book? When in the development process do you take those measurements?

LHP: The five core metrics are size, schedule time, effort (cost), defects, and process productivity. You use them from the start of and then throughout the life of the development process, whenever necessary and appropriate.

WM: In the first two phases of the Unified Process, Inception and Elaboration, one of the purposes is to figure out sufficiently what is to be done, in order to attach a size estimate to the project. So, thinking about what is to be done (especially in terms of size) goes all the way back to the beginning.

DHQ: Who on the development team is responsible for recording the metrics? Does it require additional staff?

LHP: The project manager is responsible overall. He or she usually has a quality assurance team as part of the development team. This is often a very good place to put the metrics collection and analysis function. It can be done as a part-time effort. It usually does not require a full-time dedicated person. Tools are useful for recording and processing the metrics data. The tools can be homegrown using spreadsheets. Also, there are excellent commercial tools that have been specially developed for this purpose. Such tools aid in collecting the metrics data and also provide analytical capabilities to interpret the data and tell the story the metrics are designed to convey.

DHQ: How often do you record the metrics?

LHP: Monthly or weekly during development, so you have a good history of what went on during the project.

DHQ: How often and when do you do analysis work with the metrics?

LHP: Do analysis at the start of the project, using historic data and estimates of the key parameters for the new project. This is related to the estimate for the new project. Update this estimate whenever the scope or other key factors of the project change. Use the monthly or weekly metrics data to do comparison analysis as soon as the project is under way. By the time the project is

(continued on page 2)
About the Book (continued from page 1)

To succeed in the software industry, software developers need to cultivate a reliable development process. By measuring what teams have achieved on previous projects, developers can more accurately set goals, make bids, and ensure the successful completion of new projects.

Known long-time collaborators Lawrence H. Putnam and Ware Myers present simple but powerful measurement techniques to help software developers allocate limited resources and track project progress. Drawing new findings from an extensive database of software project metrics, the authors demonstrate how readers can control projects with just five metrics, the authors demonstrate how readers can control projects with just five metrics—Time, Effort, Size, Reliability, and Process Productivity. With these metrics, developers can adjust ongoing projects to changing conditions—surprises that would otherwise cause project failure.

ABOUT THE AUTHORS

Larry Putnam, Sr., and Ware Myers have written three previous books and numerous articles together over many years. Mr. Putnam, a leading expert in the software estimation and management field, is the president of Quantitative Software Management, a software management consulting firm based in McLean, Virginia.

Ware Myers is an independent consultant and a long-time contributing editor to IEEE Computer and IEEE Software. His current interests include the application of metrics to software planning, estimating, bidding, and project control.

DHQ: Won’t keeping track of metrics make it more difficult to follow a formal software development process?

LHP: Keeping track of metrics during development is not an onerous task. It can be handled part-time by project support people (usually in quality assurance). The volume of data collected is very modest when focused on the five core metrics we discuss in our book. There are good commercial tools available that collect data, help with analysis, and plan and control the development process, so the time and effort spent servicing the metrics work is very modest and not costly.

WM: It is perfectly true that collecting metric data takes a little time and making use of it takes a little more time. But it is customary on a planet of limited resources—limited daytime, limited lifespans, and so on—to try to get work done within some limits. In fact, we have a name for it: competition. And those who don’t come close to competition-set limits are removed from business (and even government). In software development, the name for this set of activities is metrics. To the extent that software development is an economic activity, not a weekend pastime, it has to base itself on metrics, and most organizations do use some kind of metrics. The issue, really, is whether the metrics are good or “seat of the pants.” Part of making them good is the realization that they are tied into the metrics are good or “seat of the pants.” Part of making them good is the realization that they are tied into the management of development because they allow resource requirements for development to be integrated into the planning and control process that is at the heart of management. This permits you to put the right number of staff on the project at the right time; it permits the prediction of when defects will occur and how many, so defect elimination is timely; it allows you to project ahead to determine when the defect discovery rate will be low enough that the product can be released for use by the end user. These metrics functions are important to the planning and control of any development process. There is a significant void in development when metrics are not used.

LHP: Metrics can be a part of any development methodology and should be. The originators of the Unified Process did not make metrics a part of their original creation. However, the company they are affiliated with, Rational, has been integrating metrics with a number of its products and is beginning to infuse its thinking with metrics concepts. Metrics ideas are important to the management of development because they allow resource requirements for development to be integrated into the planning and control process that is at the heart of management. This permits you to put the right number of staff on the project at the right time; it permits the prediction of when defects will occur and how many, so defect elimination is timely; it allows you to project ahead to determine when the defect discovery rate will be low enough that the product can be released for use by the end user. These metrics functions are important to the planning and control of any development process. There is a significant void in development when metrics are not used.

WM: One addition that I would make to what Larry said is something about the psychology of data collection. Management should make it clear that this data is being collected for metric purposes, not for personnel use. People are willing to collect data honestly for estimating and control purposes, but if they suspect it is to have something to do with personnel appraising, they begin to fudge the data and consequently to distort the data for metric purposes.

DHQ: What’s new about relating metrics to the Unified Process, and why isn’t it enough to follow the process without metrics?

LHP: Metrics should be a part of any development methodology and should be. The originators of the Unified Process did not make metrics a part of their original creation. However, the company they are affiliated with, Rational, has been integrating metrics with a number of its products and is beginning to infuse its thinking with metrics concepts. Metrics ideas are important to the management of development because they allow resource requirements for development to be integrated into the planning and control process that is at the heart of management. This permits you to put the right number of staff on the project at the right time; it permits the prediction of when defects will occur and how many, so defect elimination is timely; it allows you to project ahead to determine when the defect discovery rate will be low enough that the product can be released for use by the end user. These metrics functions are important to the planning and control of any development process. There is a significant void in development when metrics are not used.

LHP: Metrics can be a part of any development methodology and should be. The originators of the Unified Process did not make metrics a part of their original creation. However, the company they are affiliated with, Rational, has been integrating metrics with a number of its products and is beginning to infuse its thinking with metrics concepts. Metrics ideas are important to the management of development because they allow resource requirements for development to be integrated into the planning and control process that is at the heart of management. This permits you to put the right number of staff on the project at the right time; it permits the prediction of when defects will occur and how many, so defect elimination is timely; it allows you to project ahead to determine when the defect discovery rate will be low enough that the product can be released for use by the end user. These metrics functions are important to the planning and control of any development process. There is a significant void in development when metrics are not used.
Efficiency and/or Effectiveness
by Tom DeMarco

Why Achieving Both Is Not Easy
Let’s face it, the implicit goal in all organizations is to be both: to make effective choices about what to do and then carry those choices out efficiently. That presumption is so strongly built into organizational cultures everywhere that their executives sometimes can’t see when it isn’t happening. It’s absolutely supposed to be happening, so it must be.

The fact that the organization is moving in a given direction is strong a priori evidence that it must be the right direction. Executives are annoyed when anyone in the organization challenges direction. “We wouldn’t be doing this stuff at all if it weren’t right; now what we need is for everyone to get on board to help us do it as efficiently as possible.”

Unfortunately, momentum in some direction or other does not necessarily imply carefully thought-out strategic thinking. A company can begin to move (or be moved) by a process that is more or less drift. The Brownian motion within the company asserts a net force in some direction and “By God we’re moving.” The difference between strategic thinking and drift is a matter of whether the key choices are made mindfully or mindlessly.

It may sound like a harsh charge that organizations are setting directions mindlessly, that they’re prone to get their tactics right but not their strategy. But tactics are a lot easier than strategy. Tactics can be handled in isolation. You as head of a single department in your company can optimize that department to make it more efficient in what it does, but you can’t unilaterally redirect it to do something different. That change would have to be effected above you, where the issues are an order of magnitude more complex. And it would have to be done in such a way as to build wide consensus among disparate interest groups. This requires both powerful vision and charismatic leadership. The idea that drift is often substituted for strategic direction-setting is no more surprising than the observation that visionary, charismatic leaders are few and far between.

All this suggests that a lot of companies are not really led at all. If that’s true, why isn’t it more apparent? Why don’t they seem leaderless? That is the direct result of what I call the Easy Executive Option:

Directing an entire organization is hard. Seeming to direct it, on the other hand, is easy. All you have to do is note which way the drift is moving and instruct the organization to go that way.

It was the Easy Executive Option at work, for example, that caused General Motors to cede the small-vehicle sector to foreign competition and to lag behind during the Eighties and Nineties in energy-efficient engines and nontraditional fuels.

In addition to being flat-out hard to do, building effectiveness into an organization often comes into direct conflict with increasing efficiency. This is an unfortunate side effect of optimization, first noted by the geneticist R.A. Fisher, and now referred to as Fisher’s fundamental theorem: “The more highly adapted an organism becomes, the less adaptable it is to any new change.” Fisher’s example was the giraffe. It is highly adapted to food found up among the tree branches, but so unadaptable to a new situation that it can not even pick up a peanut from the ground at the zoo.

The more efficient your organization has become, the more it’s going to need the steps laid out in the next few chapters to improve effectiveness. Taking those steps is not going to be trivial, but the alternative is to proceed more and more efficiently away from your real goals.

“Reading this book clears up the trade-offs between efficiency and effectiveness, between doing and planning, between switching and concentration, and how squeezing excess capacity out of your company can sometimes leave it terminally unresponsive.”

—Bob Metcalfe
inventor of the Ethernet, founder of 3Com author of Internet Collapses

About the Author
Tom DeMarco is a principal of the Atlantic Systems Guild and author or coauthor of four bestselling Dorset House books (Peopleware, Software State-of-the-Art, Waltzing with Bears, and Why Does Software Cost So Much?) and a ground-breaking training video (Productive Teams, with Timothy Lister).

Order Slack Today!
Call (800) 342-6657 or Visit www.dorsethouse.com
Why We Need the Formal Testing Process Model

by Rodger D. Drabick

About the Author

Rodger D. Drabick is a nationally recognized quality engineering and systems testing expert with extensive experience in the Capability Maturity Model for software and process improvement. With nearly three decades in software quality assurance and testing, he has been responsible for SQA initiatives and developing process improvement action plans at companies such as Amtrak, Bell-Atlantic, the Federal Aviation Authority, Kodak, and Lockheed Martin.

One of the reasons for reviewing requirements and addressing testability of those requirements early in the life cycle is to make testing easier, but testable requirements will also make design and coding easier for software development. If requirements are so ambiguous that test engineers don’t know what to test, how can developers be expected to design and code to those requirements?

In contrast to some of my contemporaries, I have always tried to encourage complete and open communication between a test team and a development team. Software development personnel should be reviewing test documentation, in the same way that test engineers review requirements, design, and code. Early and frequent feedback between the various groups on a program will contribute to lower costs and faster time-to-market. There should be no surprises in the test documentation for development staff members, since they should have thoroughly reviewed these products.

How to Use the Formal Testing Process Model

There is no “one true way” to test software, so the goal of this book is to provide you with a structure and set of best practices that you can use to develop a formal testing process that fits your needs, and those of your customers, your management, and your organization. You should consider this book as a menu of things you could do, and then establish a set of priorities for what you’ll do first, what you’ll do second, and so on.

As an example, suppose you are working in a test engineering role in a company in which the current practice is that when executive management gets an idea for a new application, they create a proposal including definition of what you’ll do next, and then establish a set of priorities for what you’ll do first, what you’ll do second, and so on.

In brief, what I am providing in this book is a list of tasks and processes for a program of formal testing. If you are just starting to implement a testing process, you will not want to try to implement all parts of this model in one fell swoop; it would be too overwhelming and too costly. In later chapters, I suggest ways to implement this process in a prioritized, piecewise fashion.
the “requirements”), the developer hands the code to the test engineer he or she is most comfortable working with, or to the team leader of testing. This may well be the first time that the folks involved in testing have heard about this new or upgraded application.

If you’re in the role of testing team leader or test engineer, what good does the model in this book do you? In this specific instance, I’d advise you to look carefully at the Perform Formal Test process (see figure below). Consider which of the five subprocesses (Hold Pretest Meeting, Execute Test, Determine Disposition of Incidents, Hold Posttest Meeting, or Write Test Report) you are currently performing. I suspect that in the environment I’ve postulated, your group is only performing the Execute Test and Determine Disposition of Incidents subprocesses. It would not be difficult to start holding pretest and posttest meetings, and these could markedly improve your process. At the pretest meeting, you should ask the developer to describe the unit and integration testing performed and to identify areas of risk. The personnel assigned to testing can then concentrate on these risk areas. In such an environment, there probably hasn’t been time to develop a test plan or test design, but what would you base the plan or design on, anyway? Test cases or test procedures will have been developed based on discussions with the developer and the development team lead or manager. These can also be reviewed at the pretest meeting for adequacy. This could turn out to be a long meeting for a complex application. You might want to schedule a test-case and test-procedure review meeting prior to the pretest meeting. Peer reviews of test documentation are always a good idea, and the earlier they can be held, the better the return.

As part of the Perform Formal Test process, you should also make sure you have some sort of tool or process in place to record defects discovered in testing, a process to present them to the developers, a process to incorporate the fixes, and a process to regression-test the application to make sure that the fixes work (and, perhaps equally important, that they don’t break something else).

The Need for the Formal Testing Process Model (continued from page 4)

**MODELING THE TESTING PROCESS**

*(Perform Formal Test)*

- **Hold Pretest Meeting**
  - 1.4.1
  - Test Documentation
  - Source Code, Software Doc., Test Doc., Resources, Test Tools
  - Requirements, Design, Code Issues
  - Tested Code, Incidents, Test Output Data

- **Execute Test**
  - 1.4.2
  - Test Documentation
  - Environment
  - Retest
  - CM System
  - Tested Code
  - Minutes
  - Test Document

- **Hold Posttest Meeting**
  - 1.4.4
  - Test Documentation
  - Reqs., Design, Code Issues
  - Incident-Tracking System

- **Determine Disposition of Incidents**
  - 1.4.5
  - Test Report
  - CM System
  - Tested Code

**Perform Formal Test**: The steps engineers must take to prepare for testing, execute the test steps, handle incidents, determine success or failure of the test, and write the formal test report.

**NOW IN STOCK**


by Rodger D. Drabick


312 pages softcover $41.95 ppd.

Testing is not a phase. Software developers should not simply throw software over the wall to test engineers when the code is considered complete. A good testing life cycle begins during the requirements elucidation phase of software development, and it concludes when the product is ready to install or ship after a successful system test.

Nevertheless, there is no “one true way” to test software. The most reliable, reasonable approach is to implement a formal testing process that fits the needs of the testers, the organization, and the users. Read *Best Practices for the Formal Software Testing Process* and discover the benefits of testing throughout the software development cycle.

A formal test plan is a vital part of your software development life cycle. This book presents a series of tasks to help you develop a formal testing process model, as well as the inputs and outputs associated with each task. These tasks include: review of program plans, development of the formal test plan, creation of test documentation (test design, test cases, test software, and test procedures), acquisition of automated testing tools, test execution, updating the test documentation, tailor the model for projects of all sizes.

[Read More About This Book at www.dorsethouse.com/books/bpf.html](http://www.dorsethouse.com/books/bpf.html)

Order Today! Call 1-800-342-6657 or Visit www.dorsethouse.com
**Recent Reviews**

**Five Core Metrics**  
by Lawrence H. Putnam and Waru Myers  
softcover  $49.95 (includes $6 for UPS in US)  
“...an invaluable resource for project managers and project leaders. Those managers bombarbed with software measurements will find what Putnam and Meyers refer to as the ‘five core metrics’ to be a welcome solution to their metric woes.”  
— Michelle Giles, StickyMinds.com  
“...a must-read for managers who want to bring development under control.  
“The book is intended to be used by software development managers, and their bosses, and provides a comprehensive approach to achieving predictability in the software development process.”  
— Joe Saur  
ACM Software Engineering Notes

**Waltzing with Bears**  
by Tom DeMarco and Timothy Lister  
softcover  $33.95 (includes $6 for UPS in US)  
“The authors invest a good deal of energy into fighting with general practice. Statements like ‘Risk management is project management for adults’ are meant to needle people into changing the status quo.  
“if you have anything to do with software, you really need the information in this book.”  
— Richard Mateosian  
IEEE Micro  
“i liked this book. it caused me to reexamine the way i budget software as well as other deeper assumptions about what i ‘choose to believe’ or rather why i choose to believe it. if you get a chance to read the book, you will probably come to the same conclusion, ‘wow, i never thought about things that way, and i really think i should!’”  
— Will Tracz  
ACM Software Engineering Notes

**Book Proposal Guidelines**

Manuscripts are welcome on topics in systems and software development. We seek unique, usable insights on analysis, design, testing, and management, ideas that serve to inspire and improve the efforts of professional IS developers and managers.

Send us your big, a topic discussion, an assessment of competing books, a detailed table of contents, and as many sample chapters as are available. Let us know your ideas for promoting the book. Text-only, Word, PDF, or other files are welcome by e-mail to info@dorsethouse.com.

Please note that due to the rigor and selectivity of our review process—and the busy schedules of our expert reviewers—our response time varies from six to ten weeks, sometimes longer. Simultaneous submissions are welcome, but please notify us of competing offers.

**Book Proposals**

Dorset House Publishing  
353 West 12th Street  
New York, NY 10014

---

**ORDER EARLY & SAVE**

**Agile Software Development in the Large: Diving Into the Deep**  
by Jutta Eckstein  

Most agile processes have been developed to support small to mid-sized software development projects. This book shows how to apply the principles and values of agile processes to large teams. Topics include: the effect of the principles and value system of agile processes on large teams; the impact on a team of a switch to an agile process; the characteristics of the process that will allow you to coordinate several sub-teams; how the size of the project and of the team influence the underlying architecture; and more.

**Hiring Technical People:**  
**The Artful Science of Getting the Right Person for the Job**  
by Johanna Rothman  

Hiring technical people is one of the most critical and difficult processes a manager can undertake. This book takes the guesswork out of hiring, and diminishes the risk of costly hiring mistakes. With the aid of step-by-step descriptions and detailed examples, you’ll learn how to: write a job description; source candidates; develop ads; review résumés; develop interview techniques; create phone-screening; check references; extend an offer; and more.

**System Testing with an Attitude**  
by Nathan Petschenik  

Developers striving for high quality and rapid time-to-market need to adopt an attitude: it is unacceptable for software that does not meet requirements to even reach the system test phase. Instituting this attitude requires not only the involvement of the system testers, but also that of the development team. System Testing with an Attitude explains how to cultivate productive relationships between developers and system testers and stresses the importance of identifying and delineating the responsibilities of each group, which can prevent problems in the system before system testing even begins.

**Systems Modeling & Requirements Specification Using ECSAM: Embedded Computer-Based Systems Analysis**  
by Jonah Z. Lavi and Joseph Kudish  

Discover ECSAM, a method for requirements engineering and the modeling of computer-based systems (CBS). Practiced since 1980 in evolving versions by large numbers of systems and software engineers worldwide, ECSAM was developed in part at Israel Aircraft Industries for the analysis and design of complex reactive embedded systems and software. The method guides engineers in modeling operational, functional, and design requirements, considering both static and dynamic aspects of systems.

**Testing Dirty Systems**  
by William E. Perry and Randall W. Rice  

Some systems are more difficult to test than others. Software testers contend with undefined or partially defined requirements; outdated, incomplete, or nonexistent documentation; complex logic; a mixture of languages; or worse. All of these factors make a system dirty, or virtually untestable. In Testing Dirty Systems, William Perry and Randall Rice—authors of Surviving the Top Ten Challenges of Software Testing—teach testers a six-step process for approaching such systems: system diagnosis; test planning; test execution; test analysis; report development; dirty system repair. Project leaders, independent testers, quality assurance personnel, and IS auditors will benefit from this book, as well as end-users and customers with a vested interest in the success of their systems.

*Save 20% On These and Other Titles—See Page 7 for Details.*
Order Form

Questions? Call or fax, e-mail info@dorsethouse.com, or visit www.dorsethouse.com.

Title

Adaptive Software Development (Highsmith) FORTCOMING
Agile Software Development in the Large (Eckstein) FORTCOMING
Amplifying Your Effectiveness (Weinberg, Bach & Karten)
Are Your Lights On? (Gaue & Weinberg)
Best Practices in the Formal Software Testing Process (Daback) NEW
Communication Gaps and How to Close Them (Karten)
Complete Systems Analysis (J. & S. Robertson)
Creating a Software Engineering Culture (Wiegears)
Data Model Patterns (Hay)
The Deadline: A Novel About Project Management (DeMarco)
Designing Quality Databases with IDEFIX Information Models (Bruce)
Dr. Peeling’s Principles of Management (Peeling) NEW
Everyday Heroes of the Quality Movement (Gluckman & Roomke)
Exploring Requirements (Gaue & Weinberg)
Five Core Metrics (Putnam & Myers) NEW
Fundamentals of Object-Oriented Design in UML (Page-Jones)
General Principles of Systems Design (D. & G. Weinberg)
Handbook of Walkthroughs, Inspections, and Technical Review (Freedman & Weinberg)
Hiring Technical People (Rothman) FORTCOMING
How to Plan, Develop and Use Information Systems (Van Steenis)
An Intro. to General Systems Thinking: Silver Anniversary Edition (Weinberg)
Managing Expectations (Karten)
Measuring and Managing Performance in Organizations (Austin)
More Secrets of Consulting: The Consultant’s Tool Kit (Weinberg)
The One Minute Methodology (OIR)
Personnel, 2nd ed. (DeMarco & Lister)
Practical Guide to Business Process Reengineering Using IDEF0 (Feldmann)
Practical Project Management (Page-Jones)
Process for System Arch. and Requirements Eng. (Hatley, Hruschka & Prabhaj)
Productive Teams: A Video (DeMarco & Lister)
Project Retrospectives (Kerth)
The Psychology of Computer Programming: Silver Anniversary Edition (Weinberg)
Quality Software Management, Vol. 1: Systems Thinking (Weinberg)
Quality Software Management, Vol. 2: First-Order Measurement (Weinberg)
Quality Software Management, Vol. 3: Congruent Action (Weinberg)
Quality Software Management, Vol. 4: Anticipating Change (Weinberg)
Rethinking Systems Analysis & Design (Weinberg)
Roundtable on Project Management (Bullock, Weinberg & Benesh)
Roundtable on Technical Leadership (Weinberg, Benesh & Bullock)
To Satisfy & Delight Your Customer (Rothman)
The Secrets of Consulting (Weinberg)
Slack (DeMarco) NEW
Software Productivity (Mills)
Software State-of-the-Art (DeMarco & Lister)
Strategies for Real-Time System Specification (Hatley & Prabhaj)
Surviving the Top Ten Challenges of Software Testing (Perry & Rice)
System Testing with an Attitude (Putnain & Kudish) FORTCOMING
Systems Modeling & Req. Spec. Using ECAMS (Lav & Kudish) FORTCOMING
Test Driven Systems (Perry & Rice) FORTCOMING
Understanding the Professional Project Manager (Weinberg)
Waltzing with Bears (DeMarco & Lister) NEW
What Every Programmer Should Know About Object-Oriented Design (Page-Jones)
Why Does Software Cost So Much? (DeMarco)

Add $1.25 per item. For 6 or more items, call us for actual charges. Please allow 1 to 4 weeks for delivery. We ship UPS unless otherwise requested. For UPS, please give complete street address, no postal boxes. PAYMENT MUST ACCOMPANY ORDER in US funds. NY State residents, please add NYS sales tax as effective Jan. 1, 2002 and are subject to change without notice.

SHIPPING AND HANDLING POLICY: First book or video, add $6.00 for UPS shipping (first class, air, and all shipments outside continental US are additional; please call or fax for estimates). Each additional book, volume, or video up to 5, add $1.25 per item. For 6 or more items, call us for actual charges. Please allow 1 to 4 weeks for delivery. We ship UPS unless otherwise requested. For UPS, please give complete street address, no postal boxes. PAYMENT MUST ACCOMPANY ORDER in US funds. NY State residents, please add NYS sales tax as effective Jan. 1, 2002 and are subject to change without notice.

Order Online

Now U.S. residents can use the PayPal shopping cart to order by credit card from Dorsethouse.com. PayPal accounts are free, and you can set one up when you order.

Call 1-800-342-6657 or (212) 620-4053. Mention “DHQ V13N2.”
Fax to (212) 727-1044 or Mail (see address below)

Questions? Call or fax, e-mail info@dorsethouse.com, or visit www.dorsethouse.com.
Greetings

This issue of The Dorset House Quarterly offers an interview, excerpts, news about forthcoming books, reviews, and Author News.

If this is your first issue of DHQ, or if you’d like to receive e-DHQ, the e-mail edition, call us at 1-800-342-6657 to sign up for a free subscription.

Or, send us your address and phone number via our Website, www.dorsethouse.com, or e-mail info@dorsethouse.com.

We welcome your comments, requests, and questions.

See Page 7 to Save 20% On Selected Titles

Inside, read an interview with Lawrence H. Putnam and Ware Myers, authors of Five Core Metrics: The Intelligence Behind Successful Software Management. ISBN: 0-932633-55-2 ©2003 328 pages $33.95 postpaid See page 1.