

## Developing Your Hiring Strategy

by Johanna Rothman

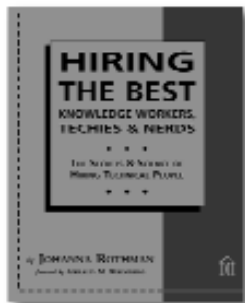
Adapted from the Preface of

### Hiring the Best

#### Knowledge Workers, Techies & Nerds: The Secrets & Science of Hiring Technical People

ISBN: 0-932633-59-5 © 2004 352 pages softcover \$43.95 postpaid

I've had the opportunity to hire or participate in the hiring of hundreds of technical people over the years, including developers, testers, technical writers, technical support staff, pre- and post-sales applications engineers, consultants, leads, and their managers. I've been part of interview teams charged with hiring product managers, electrical engineers, mechanical engineers, in-house teaching staff, and information systems staff.



Hiring technical people has never been easy. Many organizations persist in a near-constant state of having too few *qualified* technical workers. When the economy is strong, we attribute the shortage to too few qualified candidates to fill the many openings. When the economy is weak, we attribute the shortage to too many poorly trained applicants with unsuitable backgrounds.

Too often, we apply the same hiring techniques to knowledge workers that we use to hire skill-based staff. Skill-based staff members possess a set of tools and techniques that can be applied in the same way in almost all situations. Technical people—in particular, knowledge workers—must adapt their knowledge to the specific situation. Such workers are not just the sum of their technical knowledge; they are the sum of both *what they know* and *how they apply that knowledge* to the product. In particular, how they use their technical skills to benefit the product, how they manage their work, and how they manage their relationships with other people all must be assessed when hiring and evaluating a knowledge worker.

While there are some similarities in the hiring process, hiring technical people—knowledge workers—is vastly different from hiring purely skill-based staff. Knowledge workers have unique qualities, preferences, and skills—such workers are not fungible assets. The ability to adapt knowledge and to innovate makes one developer, tester, project manager, or technical manager different from another. That difference among people is key to making good hiring decisions.



## Greetings

Last December, Dorset House celebrated its twentieth anniversary. For an independent publisher, this is no small feat. We could not have done it without you, our loyal readers. We **thank you** for proving that people issues and editorial quality count in the world of computer books.

### Hiring the Best Launches to Rave Reviews

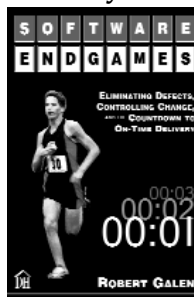
*Hiring the Best Knowledge Workers, Techies & Nerds*, by Johanna Rothman, has been very well received. First, *The Journal of Object Technology* named it “one of the best books of 2004.” Then Joel Spolsky, author of the popular blog “Joel on Software,” wrote that it “joins *Peopleware* and *The Mythical Man-Month* as must-reads for technical managers.” In February, it was named a finalist for the 15th Annual *Software Development JOLT Awards*, for products that have “jolted” the software industry. Read more at [www.dorsethouse.com/books/hire.html](http://www.dorsethouse.com/books/hire.html).

### Tom DeMarco Gives 2 Thumbs Up to Al Davis's New Book

NOW IN STOCK: *Peopleware* coauthor Tom DeMarco offered this early review of Alan M. Davis's new book, *Just Enough Requirements Management*: “Al Davis takes for his subject the largely unexplored middle ground between the requirements purists and the requirements cowboys. Since it's this middle ground where real work gets done, his guidance is both useful and welcome.” Excerpt on page 4.



### Timothy Lister Recommends Software Endgames



NOW IN STOCK: *Peopleware* coauthor Timothy Lister offered this early review of Robert Galen's new book, *Software Endgames*: “Before entering the endgame on your next software project, you don't need theory—you need proven, practical advice from an endgame veteran.” We interview Robert Galen on page 3 of this issue.

### David Harel Recommends Lavi/Kudish on ECSAM

HOT OFF THE PRESSES: *Systems Modeling & Requirements Specification Using ECSAM*, by Jonah Z. Lavi and Joseph Kudish, has received positive early reviews, including this one from Prof. David Harel: “I like this book a lot. It is one of very few books on systems and software engineering that introduces a solid and comprehensive methodology, which has been carefully worked out, has been extensively used, and has also been meticulously taught to a large number of students and engineers. . . . A truly valuable contribution to the field!”

Prof. David Harel: “I like this book a lot. It is one of very few books on systems and software engineering that introduces a solid and comprehensive methodology, which has been carefully worked out, has been extensively used, and has also been meticulously taught to a large number of students and engineers. . . . A truly valuable contribution to the field!”

### Write Like Jerry Weinberg

See page 7 to reserve your advance copy of *Weinberg on Writing*, in which prolific and popular author Gerald M. Weinberg helps readers finish their writing projects.

### Get the Right Attitude

Our new release has just arrived—*System Testing with an Attitude*, by Nathan Petschenik. See page 6 for an excerpt and page 8 to order.

#### INSIDE DHQ

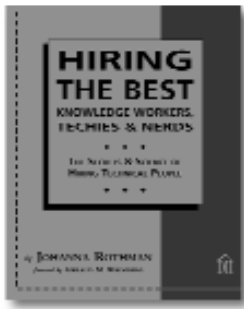
Greetings . . . . .	1
Hiring Excerpt . . . . .	1
Galen Interview . . . . .	3
Just Enough Excerpt . . . . .	4
System Testing Excerpt . . . . .	6
Forthcoming Books . . . . .	7
Recent Reviews . . . . .	7
Order Form . . . . .	8

#### Is this your first DHQ?

Send your postal address for a free subscription, but also send your e-mail address for e-DHQ, our monthly e-mail newsletter. Since we only mail DHQ to subsets of our list, e-DHQ is the most reliable way to get DHQ's news and features. We maintain our list in-house, and we do not rent, sell, or share information about our customers with any third-parties. Call 1-800-342-6657 or (212) 620-4053, fax (212) 727-1044, e-mail [info@dorsethouse.com](mailto:info@dorsethouse.com), or visit us at [www.dorsethouse.com](http://www.dorsethouse.com)  
Editor: David McClintock  
Asst. Eds.: Nuno Andrade, Claire Pixley  
Copyright © 2005 by Dorset House Publishing Co., Inc.  
ALL RIGHTS RESERVED.

(continued on page 2)

**NOW IN STOCK**



**Hiring the Best Knowledge Workers, Techies & Nerds: The Secrets & Science of Hiring Technical People**

by Johanna Rothman

ISBN: 0-932633-59-5 © 2004  
352 pages softcover \$43.95 postpaid

When you decide to hire someone, you, your team, and your organization will live with the long-term consequences of that decision. Developing a hiring strategy will improve your decisions, shorten your decision time, and decrease the ramp-up time needed for each new hire.

Unlike skill-based workers, technical people typically do not have access to cookie-cutter solutions to their problems. They need to adapt to any situation that arises, using their knowledge in new and creative ways to solve the problem at hand. As a result, one developer, tester, or technical manager is not interchangeable with another. This makes hiring technical people one of the most difficult and critical processes a technical manager can undertake.

*Hiring the Best Knowledge Workers, Techies & Nerds* 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 concise, targeted job description
- source candidates
- develop ads for mixed media
- review résumés quickly to determine Yes, No, or Maybe candidates
- develop intelligent, nondiscriminatory, interview techniques
- create fool-proof phone-screens
- check references with a view to reading between the lines
- extend an offer that will attract a win-win acceptance or tender a gentle-but-decisive rejection
- and more.

**Order Today!**  
**Call 1-800-342-6657**

[www.dorsethouse.com/books/hire.html](http://www.dorsethouse.com/books/hire.html)

**DHQ Excerpt** Hiring the Best (continued from page 1)

You want your organization to succeed, and so you need to know how to define and assess a technical candidate's qualities, preferences, and skills, but you also need to be able to predict a technical person's chance at succeeding in *your organization*. The techniques and recommendations set forth in this book are designed to make hiring a streamlined, efficient, and satisfying experience.

**Why Read *Hiring the Best*?**

*"The whole interviewing thing takes forever."*

*"How do I know this candidate will work out?"*

*"I can't seem to find candidates who meet the job's specifications."*

I hear comments like the preceding every day. If you're like most of the technical managers I've worked with, you may not be sure how to define the job's requirements, how to find suitable candidates, what skills you need to interview well, or how to make an offer that the candidate will accept. Or, possibly, you know how to do all of that, but the hiring process consumes more of your time than you comfortably can allocate.

Here's what I hope my book can teach to every hiring manager who uses it as I've intended it:

**Save time and money.** The more you streamline your hiring approach, the faster you will be able to evaluate suitable candidates, and the better the hiring decisions you'll make. An effective hiring process is especially important when you consider the toll a bad hire can take on your organization. Add up the direct monetary costs of recruiting, the cost of the time you and your staff spend on hiring the person, and the actual cost of the person's salary and benefits while he or she works for you, and you'll quickly see that the cost of a bad hire can be enormous.

**Hire people who can perform the work.** Few things are worse than feeling that a new employee somehow misled you on his or her résumé or in the interview. You thought you hired Dr. Jekyll, but Mr. Hyde showed up to work.

**Screen, evaluate, and hire the right staff for your specific organization.** You can help yourself hire well

by first defining a standard for what "a good employee" is for your specific organization, and then translating that standard into precise job requirements, a sound job description, and a comprehensive listing of information needed for successful interviewing.

**Fire fewer people.** Most managers dislike the firing process: the warnings, the get-well plan, the actual firing. Many ignore the problem altogether or shunt the non-performing employee to other projects or other managers.

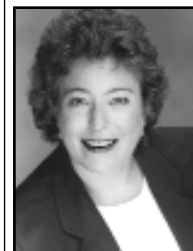
**Develop and demonstrate your own management competence.** Simply put, managers who can tell the Jekylls from the Hydes will be more successful than managers who cannot. Likewise, a manager with a staff whose members can work with others in the company will be able to complete his or her assignments faster and with greater success.

The bottom line: Once you've defined what a "good employee" means for your needs and your culture, you can quickly review résumés, conduct interviews, make offers, and hire the right technical person for the job. I hope the numerous tips, suggestions, and recommendations in *Hiring the Best* will help you expedite the hiring process as well as make it a more pleasurable experience.

**Read More About This Book.**

Visit [www.dorsethouse.com/books/hire.html](http://www.dorsethouse.com/books/hire.html).

**About the Author**



Johanna Rothman is a highly regarded speaker, author, and consultant; she is known for her pragmatic approach to the problems of managing high technology product development and

workers. During the past twenty years, she has been influential in the hiring of hundreds of technical people, including developers, testers, technical editors, technical support staff, and their managers. Based in Arlington, Massachusetts, she is the president of Rothman Consulting Group.

Visit [www.jrothman.com](http://www.jrothman.com).

**DHQ Interview****ABOUT THE BOOK**

with

**Robert Galen**

author of

**Software Endgames**

ISBN: 0-932633-62-5 © 2005

328 pages softcover \$39.95 postpaid



**DHQ:** *Peopleware* coauthor Tim Lister calls you an “endgame veteran” and hails your “proven, practical advice.” What exactly is the endgame?

**GALEN:** To me, the endgame is the period starting when a software project is first released for external testing, and ending when it’s available to your customer. External testing in this case probably implies release to a non-development testing group. It’s a period primarily focused on requirement validation, testing, defect repairs, software change negotiation, and iterative releases.

**DHQ:** What are the worst things people do during the endgame?

**GALEN:** I’d break “people” up in this case into two categories—the team members proper and then the management team. From a team point of view, I think the worst thing you can do is too much. Try to deliver too much, repair too much, work too much. I use the term “youthful enthusiasm” to indicate this tendency I’ve seen within teams. It’s much better to drive toward realistic goals that map to the historical capabilities of the team.

From a management perspective, I focus on two potentially terrible patterns. First is driving blaming or other dysfunctional behaviors within the team if things aren’t going right. I consider it management’s job, or a key leadership responsibility, to keep a level head no matter what is going on. The other major mistake is driving too much overtime. Overtime is considered unavoidable in the endgame, yet it’s probably the single biggest cause of failure.

**DHQ:** What’s the best way to turn down change requests—to say no?

**GALEN:** Actually, I really emphasize not having predefined or binary responses to change requests within the

endgame. I believe the more flexibility you have, the better. So, promising to fix something in a subsequent release becomes a viable option for handling change requests. However, you should create as many options as possible.

For example, you can implement only part of a request now and defer bits until later releases. Some requests should never be met, so an absolute no is viable as well. I always try to get to the “true need” behind any change request. If it’s truly required for a product to be successful, then I try to accommodate the request in full or at least partially. Flexibility and breadth in your response is the key.

**DHQ:** What’s unique or different about the endgame in an agile software development process?

**GALEN:** Some of my recommended processes would need to be scaled to support much shorter release points or iterations. That’s probably the most obvious change. Another is focusing more on self-directing teams when applying the book’s endgame guidance techniques. I’ll give you an example:

In the book, I propose that the development manager assign defects in a “work queue” fashion for each engineer on the team. This is one of the more controversial recommendations I make within the book. Within an agile team, I think the same approach applies, but the team members should probably select their defects with some simple guidelines to influence balance. As long as the spirit of my recommendation is maintained, I think these sorts of adjustments are perfectly fine within an agile context.

**DHQ:** What one lesson from *Software Endgames* would you most like the reader to take to heart?

**GALEN:** The key is to set up structures that illuminate the goal and guide the team. Then, get out of the way and simply guide the team toward success. In many ways, the agile methodologies have come to the same conclusion—with the notion of self-directing teams. My best endgames have been all about the team directing itself toward a clear set of goals and objectives. Inevitably, I’m always amazed at the power of technical teams to meet a challenge and succeed. A good endgame leader’s job is to set the compass and guide the way, but not get in the way.

**DHQ:** Tell us about your current work and activities.

**GALEN:** In the last few years, I’ve focused on the endgame from a testing point of view. It’s interesting because I’ve learned that the testing role—at least in certain contexts—seems to have a limited capacity to properly influence the endgame. I believe that within the software development management and project management functions, there is more natural capacity to have influence, so I encourage those managers to engage their testing teams during their endgames. As I mention in the book, testing teams are the *secret weapons* of the past few years has only underscored that view.

That being said, I’m still learning and growing from my endgame pursuits and from sharing lessons and experiences. The book is opening more doors in this area. I’m also looking forward to ongoing endgame collaboration and lessons-sharing via our discussion Yahoo group. For information on how to join, check [www.rgalen.com](http://www.rgalen.com).

**DHQ:** Thank you, Bob!

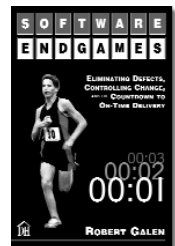
**There’s More to This Interview.**  
Visit [www.dorsethouse.com/books/send.html](http://www.dorsethouse.com/books/send.html).

**NOW IN STOCK****Software Endgames:**

Eliminating Defects,  
Controlling Change, and the  
Countdown to  
On-Time Delivery

by Robert Galen

ISBN: 0-932633-62-5 © 2005  
328 pages softcover \$39.95 postpaid



In *Software Endgames*, experienced project manager and consultant Robert Galen shows readers how to conduct effective defect triage—analyzing, understanding, and categorizing defects—in preparation for scheduling repairs.

Readers learn how to transform the endgame from a time of rampant defects and utter chaos into one of focused repairs, effective teamwork, and change management.

**Order Today!**  
**Call 1-800-342-6657**

[www.dorsethouse.com/books/send.html](http://www.dorsethouse.com/books/send.html)

## DHQ Excerpt

# Eight Attributes Are Just Enough for Requirements

by Alan M. Davis

Adapted from Chapter 4 of  
**Just Enough Requirements Management: Where Software Development Meets Marketing**

ISBN: 0-932633-64-1 © 2005  
 256 pages softcover \$39.95 postpaid

The days of large, word-processed requirements documents are over. These days, there are too many questions that a manager needs answered quickly—quicker than a word-processed document can manage. For example, a manager needs to know the following:

- How many requirements are there?
- How many high priority requirements have been delayed to a release after Release 2.0?
- What percent of the requirements for Release 2.0 are low priority?
- Which requirements in Release 2.0 are high priority, are being built for Customer X, and are the responsibility of Sally?

The need for quick answers to questions like these convinces me that the *only* way to record requirements when pressed for time is as a *list* of discrete requirements, each annotated with multiple attributes.

In a 1993 article, I put forth a long list of attributes that a well-written set of requirements should exhibit. I must have been feeling obsessive-compulsive at the time. Although it was an interesting academic discourse, the article left one wondering what a practitioner could possibly do with such an enormous list. After much analysis and

experience, I now have to admit that only eight attributes are truly important, and of these, some are not even possible to achieve. Here is the list of attributes that I think are important for today's *just enough* world (in decreasing order of importance):

## 1 Correct

A requirement is correct if it helps to satisfy a stakeholder need. This is by far the most important attribute because it gets right to the essence of why we are documenting requirements: to help satisfy stakeholder needs. In Figure 1, the circular region represents all stakeholder needs. If a requirement helps to support these needs, it is correct; otherwise, it is incorrect. For example, consider the following requirement:

*The system shall provide a red button labeled stop.*

This is correct if one or more stakeholders need such a button. If no stakeholder has this need, it is incorrect. Only the stakeholders can determine the correctness of a requirement. A project cannot hire a consultant to come in and find correctness errors unless that consultant is intimately familiar with the application and the needs of every stakeholder.

The reason this attribute is so important is simple. If a requirement is incorrect, the product will *fail* to meet stakeholder expectations.

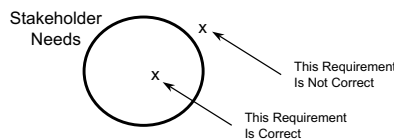


Figure 1: Diagram Showing Correctness.

## 2 Consistent

A requirement is consistent if its satisfaction does not conflict with the satisfaction of another requirement, a set of requirements, or some other previously approved document.

A developer choosing to satisfy requirement *A* in a manner that is inconsistent with another requirement does not make requirement *A* inconsistent. Instead, inconsistency is only an inherent attribute of a requirement in relation to other requirements. If there is some way to satisfy requirement *A* that does not conflict with other requirements, then it is the developer's

responsibility to discover and use that approach.

The reason this attribute is rated so highly is simple. If a requirement is inconsistent with other expectations, development will find it impossible to satisfy without failing to satisfy other stakeholder needs.

## 3 Achievable

A set of requirements is achievable if it is possible to construct some system that satisfies all requirements in that set.

Achievability is meaningful only within a context defined by the capabilities of team members, the state of technology, the amount of available resources, and the desired delivery date.

This attribute is very important because a lack of achievability makes it impossible to satisfy stakeholder needs.

## 4 Annotated

A requirement is annotated if it is easy for us to find characteristics of the requirement, including its relationships with other requirements.

Requirements should be annotated by their relative priority (to facilitate triage), estimated cost (to facilitate both triage and project management), and necessity, effort, subset, and cover dependency (to further facilitate triage and project management). Other annotations include origin, advocate, responsible party, primary customer, details, and tasks.

For example, the *origin* (also called a *backward trace* or *upstream trace*) of a requirement is a pointer to or notation concerning *why* it exists. If the requirement is derived from an earlier system-level requirement, this should be a pointer to the specific system requirement. If the requirement is being satisfied because the customer said, "Read my lips—I want the system to be green," then the origin is a notation to that effect. The primary benefit of recording the origin is that it provides the team with a place of referral whenever somebody suggests changing the requirement at a later date.

## 5 Traceable

A requirement is traceable if it has a unique identifier. Although the creation of such traces is not part of requirements activities, it is important to give

(continued on page 5)

(continued from page 4)

every requirement a unique identifier so that later tracing is possible.

## 6 Unambiguous

A requirement is unambiguous if it has only one possible interpretation. If you attempt to squeeze out all remnants of ambiguity, you will be forced to use formal methods, your costs will skyrocket, and you will run the risk of alienating customers. The alternative

**“... I put forth a long list of attributes that a well-written set of requirements should exhibit. ... I must have been feeling obsessive-compulsive at the time.”**

—AMD

is to remain in natural language and accept some degree of ambiguity. Yes, you should endeavor to remove glaring ambiguities. No, you should not accept an ambiguous requirement that resulted from a compromise among multiple parties who clearly do not agree. And no, do not go on a crazy ambiguity-hunt just so you can say you did.

## 7 Complete

Of what value is a discussion of the completeness of a requirements document when we all know it is impossible to achieve? The needs of the customer are in constant flux. Time changes needs. The best definition of completeness I can come up with is this: “A requirements document is complete if it includes all the requirements that the stakeholders expect to have satisfied in the corresponding release.”

## 8 Verifiable

A requirement is verifiable if there is a finite, cost-effective method to check that the system, as built, meets the requirement as stated. If the requirement is ambiguous (see discussion above), then by its very nature, it cannot be verified, for it possesses multiple interpretations. The best way to define a verifiable requirement, then, is to state that “A requirement is verifiable if there exists some finite, cost-effective method to check that the system as built meets the requirement as stated, to a degree sufficient to convince all relevant parties.”

## Other Attributes

As mentioned above, my 1993 paper included many other attributes. The attribute “understandable by customers” is still important, but I have combined its essence into the discussion of unam-

biguous requirements. It is of course important for a requirements document to be “modifiable”; however, I’m no longer sure what a non-modifiable requirements document would look like. It is still important for requirements to be “traced” to their origins and “organized,” but I now treat those concepts as annotations. The concept of “design independent” is important as well, but once we have agreed that a requirement is only a requirement if it maintains an external

perspective and that every requirement by its nature limits design alternatives, it seems unnecessary to insist that a requirement should be design independent. As far as “conciseness” goes, yes, it would be nice if the requirements document was concise, but if it isn’t, then what? I would not waste any time trying to make a requirements document any smaller than it is.

**N**o matter how hard you try, you will not be able to squeeze out every last drop of requirements defect. You should strive to make sure that the document is correct, consistent, achievable, and at least annotated by effort and relative priority. The remaining attributes are nice to have, but in attempting to achieve them, be careful not to lose your ability to do requirements in a just enough manner.

**“If you repeatedly find yourself having troubles managing requirements for your information system development projects, or if you have a hard time communicating with your marketing or business departments or even with your customers, this book will undoubtedly make your day.”**

—Valentin Crettaz

Javaranch.com, Val’s Blog

### About the Author



Alan M. Davis is a prolific author with nearly 30 years’ experience consulting for more than 100 major corporations worldwide including Boeing, Federal Express, and General Electric. He is currently a professor of information systems at the University of Colorado at Colorado Springs.

Visit [web.uccs.edu/adavis/](http://web.uccs.edu/adavis/).

## NOW IN STOCK



### Just Enough Requirements Management: Where Software Development Meets Marketing

by Alan M. Davis

ISBN: 0-932633-64-1 © 2005

256 pages softcover \$39.95 postpaid

If you develop software without understanding the requirements, you’re wasting your time.

On the other hand, if a project spends too much time trying to understand the requirements, it will end up late and/or over-budget. And products that are created by such projects can be just as unsuccessful as those that fail to meet the basic requirements.

Instead, every company must make a reasonable trade-off between what’s required and what time and resources are available.

Finding the right balance for your project may depend on many factors, including the corporate culture, the time-to-market pressure, and the criticality of the application. That is why requirements management—gathering requirements, identifying the “right” ones to satisfy, and documenting them—is essential.

Just Enough Requirements Management shows you how to discover, prune, and document requirements when you are subjected to tight schedule constraints. You’ll apply just enough process to minimize risks while still achieving desired outcomes. You’ll determine how many requirements are just enough to satisfy your customers while still meeting your goals for schedule, budget, and resources.

If your project has insufficient resources to satisfy all the requirements of your customers, you must read *Just Enough Requirements Management*.

**Order Today!**

**Call 1-800-342-6657**

[www.dorsethouse.com/books/jerm.html](http://www.dorsethouse.com/books/jerm.html)

**DHQ Excerpt**

# The Formula for Success in System Testing

by Nathan Petschenik

Adapted from the Introduction to  
**System Testing with an Attitude:  
An Approach That Nurtures  
Front-Loaded Software Quality**

ISBN: 0-932633-46-3 © 2005  
368 pages softcover \$45.95 postpaid

Professional project relationships are like contracts. Your role in a software project is defined by an agreement between you and your colleagues, one which requires you to live up to your end of the bargain if project goals are to be achieved. In order to establish and maintain such contracts, we need to understand each other's roles. If I am charged with fulfilling an important project responsibility, I want you to understand my role so that you will know what to expect from me. Conversely, if I need something from you to fulfill my role, I want you to have a clear understanding of my expectations.

In many software projects, the relationship between developers and testers is clouded by misunderstanding and disappointment on both sides. Figure 1 characterizes the typical situation.

Some developers believe that the existence of testers on a project relieves them of some or all of the responsibility for testing their work. This can result in the developers expecting more from the testers than they wind up getting. On the other side of the relationship, if the developers are delivering software that they have not tested to the extent the testers expect, the testers will continually gripe about the quality of the software that they receive from the developers.

Not only does this tension exist during the development and testing phases of the software life cycle, it tends to continue after the software is deployed. In particular, as users report problems with the software, it is not unusual for project management, the users, and even the developers to point

to the testers and ask the question, "How could you have missed that problem?" as if to imply that somehow the testers are responsible for the quality problem. On the other hand, questions of how the problem got there in the first place and why the developers didn't find it are hardly ever asked at all (or at least that's how the testers perceive it).

The underlying issue in the developer/tester relationship is that most project members have a vague notion of the role of testers on their projects. This is particularly true in projects where there is a distinct team charged with system testing. Developers, project managers, and even the testers themselves tend to have unrealistically high expectations about the amount of testing that can go on during the system test, about the level of coverage that can be achieved, and about the types of problems that can be routinely uncovered. This has the effect of de-emphasizing the developers' responsibility for software quality and imposing a definition of success on testers that is impossible to achieve.

**S**ystem Testing with an Attitude is about the role of system testing in a software project and is written from the point of view of the system-test team. Two main questions are addressed:

1. How do we create a clear "contract" between the system-test team and the rest of the organization?
2. How do we fulfill the system-testing end of the contract?

These two questions are interrelated (and, in fact, this is a key theme of the book). The relationship can be shown in a formula:

**System-Testing Success =  
(Technical Excellence) +  
(Nurturing Front-Loaded  
Quality)**

Part I of the book focuses on the issues that shape the role of system testing. We need to understand these issues to answer both of the above questions. The technique that we will use in Part I to get to the underlying issues is based on a series of workshops that I have run over the years called "Role-Awareness Seminars." You may decide to use this technique to establish and reach agreement

on the role of system testing in your project as part of addressing the first question. This book provides everything you will need to run role-awareness seminars for your project. In Part I, you will be a participant.

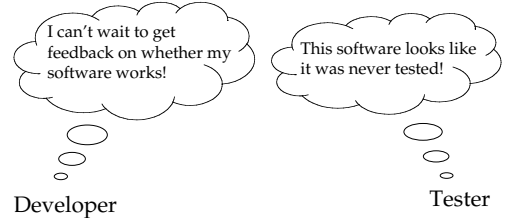


Figure 1: Misunderstanding Between Developers and Testers.

Part II of the book focuses primarily on answering the second question. This includes both technical advice for fulfilling the system-test contract and procedural advice that will influence the behavior of others throughout the project.

The technical material is intended to increase the practical advice available in the literature on system testing. Fulfilling the system-test contract requires technical excellence in the science of system testing.

As for the procedural advice, it may be possible to fulfill aspects of the system-test contract strictly by influencing behavior. However, you won't be totally successful unless you change attitudes as well. In other words, in order to answer the first question completely, everyone on the project should "buy in" to the contract. At the end of Part II, we will put the final touches on the answer to the first question by describing how to run a role-awareness seminar using the material from Part I.

So, again, the underlying theme is that you need to answer both questions in order to answer either.

**Order Today!  
Call 1-800-342-6657**

[www.dorsethouse.com/books/sta.html](http://www.dorsethouse.com/books/sta.html)

## About the Author



Nathan Petschenik is an internationally known consultant on software testing. He is currently Vice President of Software Testing Services, Inc., an IT consulting firm specializing in all aspects of software testing and quality assurance.

Visit [www.stsv.com](http://www.stsv.com).

## Author News

**Robert Galen:** In Minneapolis (9/13-9/19), presents a featured talk related to his book, *Software Endgames: "Learning to Finish What You've Started"*; teaches a full-day Software QA Leadership workshop. In San Francisco (9/21-9/22), presents "Negotiating the Defect Minefield for a Successful Project Release" at the 2005 Better Software Conference.

Visit [www.rgalen.com](http://www.rgalen.com).

**Jutta Eckstein:** In Aarhus, Denmark (9/25-9/30), presents "Agility In Perspective" at the JA00 2005 conference ([www.jaoo.dk](http://www.jaoo.dk)).

Visit [www.jeckstein.com](http://www.jeckstein.com)

**Naomi Karten:** In Phoenix (11/6-11/9), teaches at the AYE Conference ([www.ayeconference.com](http://www.ayeconference.com)).

Visit [www.nkarten.com](http://www.nkarten.com).

**Timothy Lister:** In San Francisco (9/20, 9/22), teaches and keynotes at the 2005 Better Software Conference.

Visit [www.systemsguild.com](http://www.systemsguild.com).

**Johanna Rothman:** In San Francisco (9/19), teaches at the 2005 Better Software Conference. In Phoenix (11/6-11/9), hosts the AYE Conference ([www.ayeconference.com](http://www.ayeconference.com)).

Visit [www.jrothman.com](http://www.jrothman.com).

**Gerald Weinberg:** In Boston (9/27), keynotes at the 2005 SD East Best Practices Conference. In Phoenix (11/6-11/9), hosts the AYE Conference ([www.ayeconference.com](http://www.ayeconference.com)).

Visit [www.geraldmweinberg.com](http://www.geraldmweinberg.com).

# FORTHCOMING\*

## Weinberg on Writing: The Fieldstone Method

**Est. Nov. 2005—in Time for AYE!**

by GERALD M. WEINBERG ISBN: 0-932633-65-X © 2006 est. 200 pages softcover \$30.95 ppd.



Gerald M. Weinberg, author of nearly 400 articles and some forty books—including eighteen published by Dorset House—reveals his secrets for collecting and organizing his ideas for writing projects. Drawing an analogy to the stone-by-stone method of building fieldstone walls, Weinberg shows writers how to construct fiction and nonfiction manuscripts from key insights, stories, and quotes. The elements, or stones, are collected nonsequentially, over time, and eventually find logical places in larger pieces. The method renders writer's block irrelevant and has proved effective for scores of Weinberg's writing class students. If you've ever wanted to write a book or an article—or need to revitalize your writing career—don't miss this intimate glimpse into the mind behind some of the computer industry's best books.

## Object-Oriented Computation in C++ and Java

**Est. Nov. 2005**

by CONRAD WEISERT ISBN: 0-932633-63-3 © 2005 est. 200 pages softcover \$39.95 ppd.

Virtually all business, scientific, and engineering applications are heavily reliant on numeric data items. However, relatively few programmers are trained to exploit the full computational power of C++ and Java. Professor and consultant Conrad Weisert offers new techniques for manipulating numeric data types, reinforcing the concepts with helpful exercises throughout. The text is designed for self-study or use in an undergraduate course.

## Testing Dirty Systems

**Est. Dec. 2005**

by WILLIAM E. PERRY and RANDALL W. RICE ISBN: 0-932633-56-0 © 2006 est. 368 pages softcover \$41.95 ppd.

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 • and dirty system repair.

\***Save 20% When You Order Before Publication—See Page 8 for Details.**

## Recent Reviews

### Best Practices for the Formal Software Testing Process

by **Rodger D. Drabick**

ISBN: 0-932633-58-7 © 2004 312 pages  
softcover \$41.95 (includes \$6 for UPS in US)

"The book offers two major benefits to the reader. The first is when to test at points during the development life cycle, and the second is how to test at those points in the development life cycle. The book even addresses an important issue of testing the test plan. Testers cannot assume that their work is defect free. Drabick provides an approach for testers to use to remove defects from their test process.

"Drabick's book contains literally hundreds of ideas. . . . If you only picked one good idea from the book, it would be more than worth the price."

—**William E. Perry**

Exec. Director, Quality Assurance Institute



### Agile Software Development in the Large

by **Jutta Eckstein**

ISBN: 0-932633-57-9 © 2004 248 pages  
softcover \$39.95 (includes \$6 for UPS in US)

"Overall, this book works with one premise: communication is the key to using Agile development processes for the creation, revision, management activities, etc., associated with successful end-products. This is refreshing, since communication is crucial to any IT effort's success. . . . This book needs to be read by testers for a 'reality check' on the realized gains of the promotion of testing as both an art and a science as well as how much of our journey still remains."

—**Meredith Otto**

Stickyminds.com



### Systems Modeling & Requirements Specification Using ECSAM

by **Jonah Z. Lavi** and **Joseph Kudish**  
ISBN: 0-932633-45-5 © 2005 400 pages softcover \$53.95 (includes \$6 for UPS in US)

"This is a comprehensive resource for computer-based systems designers, researchers, and students. It provides a well-structured treatment of the underlying techniques that place the Embedded Computer Systems Analysis and Modeling method in well-grounded concepts of object-oriented design and state-space behavioral specifications. This unique reference draws extensively from the authors' experience on large design projects. It will serve as a fundamental reference text for embedded systems practitioners."

—**Jerzy W. Rozenblit**

Professor, Electrical and Computer Engineering, The University of Arizona



Dorset House Faxable Order Form • Fax (212) 727-1044 • Call (800) 342-6657 or (212) 620-4053

Visit www.dorsethouse.com/savings.html for special offers and updates.

† Discounted price valid on orders received before the indicated expiration date. You will not be charged until we're ready to ship. International customers: We'll contact you with shipping options. ✓ Call for details: Save 20% when you order with a full-priced title. Qty. limited

Table with columns: TITLE, #, PRICE, TOTAL. Lists various software development and management books with their respective prices and special offers.

SHIPPING AND HANDLING POLICY: First book or video, add \$6.00 for UPS shipping (first class, air, and all shipments outside continental USA are additional: please call, fax, or e-mail with quantities desired, shipping address, and your contact information (phone, fax, e-mail); we will provide shipping costs estimate). 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; NYS residents, please add sales tax. Prices are effective March 1, 2002 and are subject to change without notice. For estimates, e-mail info@dorsethouse.com.

Summary table with rows: SUBTOTAL: SHIPPING: NYS TAX: TOTAL:

Enclosed is my check or money order. Charge the total to my credit card:

VISA MC AMEX Card # Exp. date Signature

Very important -> Daytime phone fax or e-mail

Please e-mail a shipping confirmation. Please e-mail me the e-DHQ newsletter.

NAME TITLE COMPANY STREET (NO P.O. BOXES FOR UPS DELIVERY) CITY STATE ZIP COUNTRY

DORSET HOUSE PUBLISHING 353 W. 12TH ST. NEW YORK, NY 10014 USA Callers, Please Mention "DHQ V15N1"