Practical Software Requirements: Engineering and Management

Karl E. Wiegers
August 15, 1998

Proposal

Despite some 50 years of collective experience, many software development organizations continue to struggle with the collection, validation, and management of requirements for their products. The difficult and often frustrating requirements process frequently gets short shrift from practitioners who would prefer to write code than deal with elusive and inconsistent customers. The requirements process underscores the fact that the software profession involves roughly equal portions of computing and communication. We seem to be better at the former than the latter.

The author has written several articles about software requirements, and some of his experiences with applying improved requirements techniques were related in two chapters of his first book, *Creating a Software Engineering Culture* (Dorset House, 1996). In 1997 and 1998, the author presented more than 30 full- and half-day seminars on software requirements engineering and management at many companies, conferences, and public seminars. There is clearly a great deal of interest in this topic, because many companies are still having problems with their requirements practices. Requirements process improvement is also being motivated by the Software Capability Maturity Model, which identifies requirements management as one of the key process areas that must be mastered to achieve CMM maturity level 2.

The volume of published literature on software requirements is huge, with well over one thousand books, articles, and conference proceedings papers available. However, most practitioners do not appear to be well acquainted with this body of literature. The available books range from turgid texts, to thin collections of essays, to useful "good practices" references. Practical application of the research approaches presented in many of the more academic papers and proceedings seem to be rare. There is no shortage of effective techniques available for gathering, analyzing, documenting, and managing software requirements. However, a substantial gap remains between the available body of knowledge and current industrial and commercial practice.

Reader Objectives

The proposed book is intended to address this gap. The purpose of the book is to concisely present effective techniques for the requirements engineering and management processes from a practical perspective that can benefit any software organization. The book is intended to be relatively short (approximately 250 pages), easy to read, and highly pragmatic. Guidance on the realities of implementing improved requirements engineering and management techniques in practice will be included.

Readers will be able to accomplish the following by applying the contents of this book:

- Achieve higher customer satisfaction, and reduce maintenance and support costs.
- Educate both development and customer organizations about the risks and costs they face from inadequate requirements work, so the stakeholders can agree to devote appropriate effort to this critical set of tasks.
- Determine who really represents the voice(s) of the customer for their product, and use structured processes for gathering and analyzing input from those sources.

- Differentiate between high-level business or user requirements, and low-level developer, or functional, requirements.
- Document both functional and non-functional requirements in formats and levels of detail appropriate for each project.
- Reduce project rework, thereby improving productivity, by improving the quality of their requirements early in the project's development cycle.
- Develop more realistic, accurate, and achievable schedules and plans by more thoroughly understanding the problem being solved prior to making commitments.
- Better manage the scope creep that affects many software projects by implementing a structured process for collecting, evaluating, and incorporating proposed changes in product requirements.
- Actually implement improved requirements engineering and requirements management processes in their organizations.

The author's experience suggests that most practitioners dealing with software requirements use informal methods for gathering requirements, and they document them using natural language. Few practitioners seem to use formal logic, mathematical representations, or rigorous modeling as routine requirements practices. The emphasis of this book is therefore on techniques to improve the requirements engineering and management processes without demanding unrealistic current knowledge or a commitment to learning complex new approaches. Formal requirements representation techniques and methodologies are not addressed. Nor does this book attempt to comprehensively summarize the entire domain of software requirements practice and research. Instead, the reader is presented with a palette of established good requirements development practices, together with examples of their application on various projects.

Synopsis of Topics Covered

Nowhere more so than in the requirements gathering portions of the project do all of the stakeholders share common interests. Inadequate customer involvement leads to an "expectation gap" between what is expected and what is delivered. Collaboration of customers and developers is explored, emphasizing a "product champion" model for achieving effective customer participation on requirements engineering. Effective methods for conducting dialogs between customers and developers around requirements are explored, including a chapter that provides guidance for those sitting on the customer side of the table. The use case method is emphasized as a preferred technique for eliciting requirements, but from a non-dogmatic and highly accessible perspective. The need for two levels of requirements detail, the higher-level user view (represented in use cases) and the lower-level developer view (represented in detailed functional requirements) is addressed.

No matter how they are gathered, requirements must be documented, analyzed, and reviewed from a variety of perspectives. No single view of the requirements provides an adequate representation. Suggested contents and organization of a structured software requirements specification are described. A variety of requirements modeling techniques are illustrated. In addition to the more obvious functional requirements, the book also examines the importance of non-functional requirements, including product quality attributes. Case studies and examples are provided from the author's personal experience to illustrate the effective application of all these techniques on a variety of projects in different organizations.

Techniques for verifying the correctness and completeness of requirements through technical reviews and requirements-based testing methods are emphasized. While most software engineers have been taught they should do test planning in parallel with development, this happens infrequently in practice. The use case method provides a powerful technique for developing functional test cases very early in the project. Examples are provided of combining use cases, conceptual test cases, dialog maps (models of user interface architecture), and functional re-

quirements to improve the ultimate quality of the product long before a line of code has been written.

Once gathered, documented, reviewed, and baselined, changes to requirements must be managed in a structured fashion to minimize the adverse impact of those changes on the project's quality and chances of success. Concepts and practices for requirements management are adapted from the Software Capability Maturity Model, including the requirements traceability matrix, version control, and change control processes. The use of impact analysis of proposed requirements changes is presented as one way to help manage the scope creep that plagues so many software projects. Examples of effective requirements change control processes and procedures are provided, and the benefits are elaborated.

Distinctive Features

Case Studies. To help readers see how to apply the principles presented in practice, two major case studies will be threaded throughout the book. Both are adapted from the author's experiences with using these requirements techniques on real projects. The first case study is a medium-sized information system called the Chemical Tracking System. The second is a web development project. Each case study will provide examples of requirements-related deliverables (e.g., software requirements specification, use case descriptions, statement of scope, models, quality attributes statements, change request). Sample dialogs among project stakeholders in various requirement-related situations will also be drawn from the case studies.

Requirements Practices Self-Assessment. A self-assessment questionnaire will be included, to allow readers to evaluate their current requirements engineering and requirements management practices. Pointers will be provided to specific parts of the book that can help readers address their problem areas.

Annotated Bibliography. Each chapter will include an annotated bibliography of references cited and other pertinent readings, rather than a simple citation list. If the size of the book creeps up beyond 250 pages, this feature may be dropped in favor of the additional content.

Requirements Problems and Solutions Matrix. An appendix will be included that lists many typical problems practitioners encounter with gathering, documenting, or managing requirements. These problems are collected from small group discussions held as part of the one-day seminar called "In Search of Excellent Requirements" that the author has presented many times to diverse groups. The problems, their impacts, and possible root causes are described, and specific techniques from the book are suggested as possible remedies.

Market

This book has a potentially large readership drawn from any software development organization (commercial, management information systems, or contract). Anyone involved in gathering, analyzing, or documenting requirements, and anyone involved in dealing with changes to product requirements after development has begun, will find the contents to be valuable. User representatives who wish to collaborate more effectively with development organizations on requirements development will also find this useful. The book will also help both development and customer managers understand the software requirements process better.

Much of the material in this book has been presented in the author's seminars at conferences, companies, and in public forums. In addition to the predominant audience of software developers, audiences have included managers, requirements engineers, business analysts, process improvement specialists, marketing staff, and end users, All audience groups have found the ma-

terial to be easy to understand and valuable. The appeal of this book will be in its emphasis on immediately applicable, contemporary, practical approaches (as opposed to theory or methodology-specific focus), and a conversational, easy-to-understand writing style.

About the Competition

Andriole, Stephen J. ***Managing Systems Requirements: Methods, Tools, and Cases.*** New York: McGraw-Hill, 1996. 318 pages, hardcover, ISBN 0-07-001974-6, $45.00. *Summary: A fairly readable treatment of requirements topics, with emphasis on the CMM's approach and on requirements management tools.*

Andriole describes the requirements management process, requirements analysis and modeling, and requirements change management. The approach taken is adapted and extended from the Software Capability Maturity Model. Each chapter includes a rather clumsy executive briefing in bullet list format. 90 pages are devoted to summaries of more than 70 requirements management tools, although some of the more recent and well-established tools are not mentioned. Unfortunately, tool information rapidly becomes out of date, including version numbers, prices, and even vendors. Some case studies are presented, including some on process re-engineering that appear unrelated to the rest of the book. The book is quite readable and fairly practical, but the large section on tools is more of a distraction than an aid.

Davis, Alan M. ***Software Requirements: Objects, Functions, and States.*** Englewood Cliffs, N.J.: PTR Prentice-Hall, 1993. 521 pages, hardcover, ISBN 0-13-805763-X, $70.00. *Summary: The most comprehensive and coherent textbook presently available.*

Davis has written the current definitive text on software requirements. It includes an amazing bibliography of 772 references, many with descriptive annotations. Davis addresses the breadth of techniques used for requirements analysis. In fact, just one part of one chapter is 123 pages long, called "Survey of Techniques" for problem analysis. Nice chapters addresses the software requirements specification, behavioral requirements, nonbehavioral requirements, and prototyping. The book's main drawbacks are it's very comprehensiveness and its size. I believe most practitioners are not attracted to the prospect of having to sort through such an abundance of choices for practices (10 approaches for behavioral requirements techniques alone). Short shrift is also given to customer interactions and requirements process improvement, topics the proposed book would address in some detail.

Ferdinandi, Pat. ***Clarifying the Software Requirement Mystique.*** (manuscript in preparation). *Summary: Similar material to the current proposal, but more conceptual, with much less content on guiding the reader in the practical application of improved requirements practices.*

This proposed book identifies the requirements problem as a misunderstanding of the definition, process, practices, and management of requirements. It is intended to address the requirements process, requirements engineering, requirements standards, and requirements management. It will include sections on what a requirement is, the requirements evolutionary process, the requirements engineer's skill set, and future directions in requirements. The author's proposal states, "This book is a generic book on requirements for Information Technology Solutions. Though techniques and methods may be introduced, this book does not include a tutorial of current techniques and methods. It is the author's opinion that in order to learn how to do something, you must understand what it is you are to do. Other sources for "how-to" are recommended within the text." It therefore differs significantly from my proposed book, which emphasizes the "how to do it" and practical application of requirements techniques.

Gause, Donald C., and Gerald M. Weinberg. ***Exploring Requirements: Quality Before Design*****.** New York: Dorset House Publishing, 1989. 300 pages, hardcover, ISBN 0-932633-13-7, $44.95. *Summary: A good, readable book on important general product requirements topics using an anecdotal approach, but no direct treatment of software issues.*

Gause and Weinberg address important issues in the requirements of any product, including the problems caused by the ambiguity of language. However, the problems of software requirements are not specifically addressed; the words "software" and "specification" do not even appear in the index. There are good sections on how to talk with users to understand the problem and reduce ambiguity, brainstorming, and the use of context-free questions. A nice section addresses devising test cases to test requirements. There is nothing on what should go into a good software requirements document or how to structure it. The authors include a section on general ideas about doing quality reviews, but not much specific about how to review requirements documents (who should participate, what to look for, and so on). There's a useful discussion of product attributes that go beyond functionality, but again not as it pertains to software.

Jackson, Michael. ***Software Requirements & Specifications: A Lexicon of Practice, Principles and Prejudices*****.** Reading, Mass.: Addison-Wesley, 1995. 228 pages, paperback, ISBN 0-201-87712-0, $26.95. *Summary: An alphabetical collection of thoughts and stories on many topics, some of which seem only tangentially related to the requirements process. Neither a tutorial nor a reference on how the reader improve current requirements practices.*

In this collection of essays, Jackson presents much valuable food for thought, but little pragmatic guidance for the reader who wishes to better understand the software requirements process and improve his own. Many unfamiliar concepts and terms are presented, which I have never before seen in writings on software development. Useful examples are provided of shortcomings in several widely used requirements analysis practices. Historical and anecdotal stories are often used to illustrate the points. They are amusing and interesting, but not that helpful. Formal notations, such as predicate logic, are sprinkled throughout the book. I have never seen a requirements practitioner use such notations, and I don't think they'll be understood by most readers (including myself).

Sommerville, Ian, and Pete Sawyer. ***Requirements Engineering: A Good Practice Guide*****.** New York: John Wiley & Sons, 1997. 404 pages, paperback, ISBN 0-471-97444-7, $49.99. *Summary: An excellent reference compilation of sound requirements practices, well-organized and well-presented, with many good guidelines for effective implementation.*

This useful book addresses some 66 guidelines for effective requirements engineering and requirements management. Each guideline includes a statement of benefits, ideas for implementation, costs, and potential problems. The book is clearly written and easy to understand. The full breadth of the requirements development discipline is addressed. It differs from the proposed book in that it is more of a reference than a tutorial. Case studies that illustrate application of the techniques are not included.

Wieringa, Roel. ***Requirements Engineering: Frameworks for Understanding*****.** New York: John Wiley and Sons, 1996. 453 pages, paperback, ISBN 0-471-95884-0, $80.00. *Summary: Heavy on methodology and modeling, written in a an academic fashion that is not easy to read.*

This textbook emphasizes the use of models and methods for requirements engineering, including entity-relationship models (58 pages), structured analysis (52 pages), Jackson System Development (52 pages), and model integration (30 pages). Another 50 pages are devoted to development strategies and life cycles and how to select one. The requirements specification itself receives a scant 10 pages of treatment. Despite the rigorous treatment of modeling methods, object-oriented approaches are not addressed. Practical approaches to gathering functional and non-

functional requirements from users and managing the requirements over time are lacking. The book has a ponderous and academic tone that makes reading it a chore, not a pleasure.

Publishing Details

This book is intended to be approximately 250 pages, or about 75,000 words, long. Illustrations will be relatively simple line drawings or models imported from software graphical design tools (no photographs). There will be approximately 30-40 such illustrations. It may be desirable to obtain permission to reuse up to four figures from the author's previous book, *Creating a Software Engineering Culture*.

The publication format should be intended to make the book as accessible and appealing as possible; a reasonable price point might be $30. Draft chapters will be posted on the author's web site at www.processimpact.com as they become available. Approximately 20 reviewers from many different companies and backgrounds have volunteered to participate in chapter reviews. The final manuscript will be completed by approximately July 1, 1999.

Author Background

Karl E. Wiegers is Principal Consultant with Process Impact in Rochester, New York. Previously, he spent 18 years at Eastman Kodak Company, including experience as a photographic research scientist (4 years), software developer (9 years), software manager (3 years), and software process and quality improvement leader (4 years). Karl received a B.S. degree in chemistry from Boise State College, and M.S. and Ph.D. degrees in organic chemistry from the University of Illinois. He is a member of the IEEE, IEEE Computer Society, American Society for Quality, and Association for Computing Machinery.

Karl is the author of the Jolt Productivity award-winning book *Creating a Software Engineering Culture* (Dorset House, 1996), as well as more than 110 articles on many aspects of computing, chemistry, and military history. He is a frequent speaker at software conferences, public seminars, and professional society meetings. As an independent consultant, Karl presents training seminars and consulting engagements at a variety of companies on topics including requirements engineering and management, software process improvement, software technical reviews, software measurement, and risk management.

Recent Publications

1. "Know Your Enemy: Software Risk Management," *Software Development* (October, 1998).
2. "Read My Lips: No New Models!", *IEEE Software* (September/October, 1998).
3. "Software Process Improvement: Eight Traps to Avoid," *Crosstalk*, (September 1998).
4. "A Project Management Primer," *Software Development,* vol. 6, no. 6 (June 1998).
5. "Molding the CMM to Your Organization," *Software Development*, vol. 6, no. 5 (May 1998).
6. "The Seven Deadly Sins of Software Reviews," *Software Development*, vol. 6, no. 3 (March 1998).
7. "Metrics: Ten Traps to Avoid," *Software Development,* vol. 5, no. 10 (October 1997).
8. "Standing on Principle," *Journal of the Quality Assurance Institute*, vol. 11, no. 3 (July 1997).
9. "Recognizing Achievements Great and Small," *American Programmer*, vol. 10, no. 5 (May 1997).
10. "Listening to the Customer's Voice," *Software Development,* vol. 5, no. 3 (March 1997).
11. "Misconceptions of the CMM," *Software Development,* vol. 4, no. 11 (November 1996).
12. "Software Process Improvement: 10 Traps to Avoid," *Software Development*, vol. 4, no. 5 (May 1996).

13. "Reducing Maintenance with Design Abstraction," *Software Development*, vol. 4, no. 4 (April 1996).
14. "Improving Quality through Software Inspections," *Software Development*, vol. 3, no. 4 (April 1995).
15. "Creative Client/Server for Evolving Enterprises" (with Bruce Thompson), *Software Development*, vol. 3, no. 2 (February 1995).
16. "In Search of Excellent Requirements," *Journal of the Quality Assurance Institute*, vol. 9, no. 1 (January 1995) [won 1995 Best Article award from QAI].

Recent Conferences

Borland Developers Conference (1997, 1998)
Software Development East and West (1991-1992, 1994-1998)
Developing Strategic I/T Metrics Conference (1998)
Software Engineering Process Group Conference (1997)
Pacific Northwest Software Quality Conference (1997)
International Conference on Software Quality (1997)
Summit '98 (1998)

Recent Client Companies for Requirements Classes or Consulting

West Group, Intel Corporation, The Foxboro Company, Merrill Lynch, Bell Atlantic, NYNEX Science and Technology, Caterpillar, Advanced Information Services