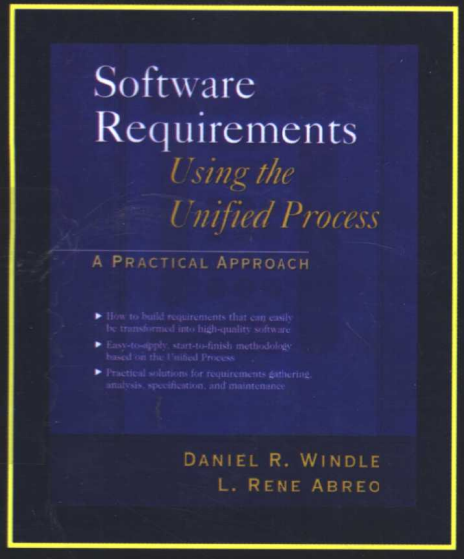
原版风暴·软件工程系列

Software Requirements
Using the Unified Process a Practical Approach

软件需求——
基于统一过程的实践方法
（影印版）

[美] Daniel R. Windle　L. Rene Abreo　著



如何建立能够更为有效的软件需求■
基于统一过程的易于使用且完整的方法■
需求收集、分析、规范及维护问题的实际解决方案■

中国电力出版社
www.infopower.com.cn

Software Requirements

Using the Unified Process a Practical Approach

软件需求——

基于统一过程的实践方法

（影印版）

[美] Daniel R. Windle　L. Rene Abreo 著

中国电力出版社

Software Requirements Using the Unified Process A Practical Approach. (ISBN 0-13-096972-9)
Daniel R.Windle &L.Rene Abreo

北京市版权局著作合同登记号：图字：01-2003-2433
For sale and distribution in the People's Republic of China exclusively (except Taiwan, Hong Kong SAR and Macao SAR).
仅限于中华人民共和国境内（不包括中国香港、澳门特别行政区和中国台湾地区）销售发行。

图书在版编目（CIP）数据

软件需求——基于统一过程的实践方法/（美）温德勒，（美）艾伯瑞奥著. 一影印本. 一北京：中国电力出版社，2003
（原版风暴·软件工程系列）
ISBN 7-5083-1506-5
Ⅰ.软... Ⅱ.①温... ②艾... Ⅲ.软件设计方法学一英文 Ⅳ.TP311.5
中国版本图书馆 CIP 数据核字（2003）第 027843 号

责任编辑：朱恩从
丛 书 名： 原版风暴·软件工程系列
书　　名： 软件需求——基于统一过程的实践方法（影印版）
编　　著： （美）Daniel R. Windle、L. Rene Abreo
出版发行：中国电力出版社
地址：北京市三里河路6号　邮政编码：100044
电话：（010）88515918　传真：（010）88423191
印　　刷： 北京地矿印刷厂
发 行 者： 新华书店总店北京发行所
开　　本： 787×1092 1/16　**印　　张：** 17.5
书　　号： ISBN 7-5083-1506-5
版　　次： 2003年7月北京第一版
印　　次： 2003年7月第一次印刷
定　　价： 35.00 元

Preface

We intend that this book provide you with a practical approach to gathering, analyzing, specifying, and managing software requirements throughout the software's life cycle.

If you are an analyst responsible for specifying requirements from which software systems are built, we think you will find a straightforward and effective approach to meeting the demands of the users you work with and the developers and testers you deliver specifications to.

If you are a developer responsible for building software systems from requirements specifications, we think you will find an effective way to communicate all software requirements in a coherent and easy-to-follow manner.

If you are a tester responsible for testing software systems from a specification, we think you'll find that the specifications described in this book provide a complete and clear control flow model of the entire system, allowing you to systematically develop tests. We also believe you will find that the requirements artifacts described in this book will allow you to learn a new system quickly and thoroughly.

If you are a manager of software development or testing, we think you will find that the establishment and maintenance of requirements artifacts will ensure that you can always quickly bring new people up to speed on your systems. We also believe you will find a practical approach to shortening the time it takes to specify software requirements while maintaining your organization's intellectual property.

Acknowledgments

First, we would like to thank our wives, Kathleen Windle and Sandi Abreo, and our children—Tasha Windle, Cynthia Windle, Michael Abreo, and Alexandra Abreo—for their understanding and patience as we embarked on this endeavor.

We would also like to thank our colleagues at the Securities Industry Automation Corporation in New York City who helped with reviews of this book, in particular, Mark Lewis and Alex Ciccotelli, who generously gave of their time. We would like to specifically thank Charles Bowman for setting us on the path to writing this book.

Contents

PART

1

Introducing Good Requirements

CHAPTER 1

The Importance of Good Requirements

Rationale for Good Requirements

Real-world systems are built every day and based on skimpy requirements. We know this because most of us have been on projects where the requirements change daily. Sometimes these changes are due to the customer changing his mind. More often, these changes are due to clarifications of the software requirements that come late in the project. Anyone responsible for the maintenance of one or more significant projects has also experienced the effects of building systems with skimpy requirements. Many of us have struggled with reliance on single individuals for knowledge about what a system does. Some of us have been that individual struggling to remember all the interdependencies among the requirements we are being asked to change to implement a particular release. The approach described in this book is a practical approach to developing software requirements. We have successfully delivered a significant number of systems by using this process and the resulting artifacts. These systems have ranged from defense command and control systems to web systems to financial transaction processing systems. We have also recovered the requirements from existing sys-

tems by using this process to alleviate the reliance on individuals and to improve the speed and quality of maintenance activities.

Requirements tell you the features a system must have as well as what the software must do to deliver those features. This is a significant aspect of requirements: they possess a dual nature. Features represent user requirements. What the software does represents software requirements. There is a large gap between user requirements and software requirements. The artifacts and process described in this book will help you fill that gap. You work with the user to determine what features the system must have. From this feature set, you must derive a set of components that can deliver those features. These components and their relationship to each other lead you to a software architecture. This architecture tells you what the software must do. The user requirements come from describing the user's interaction with the system. You derive the software requirements during analysis by mapping the user requirements onto the software architecture.

Requirements are normally the work of a requirements analyst. While the dual nature of requirements may lead an analyst to specialize in user requirements or software requirements, generally an analyst serves two customers. The first is the user of the system. The second is the developer of the system who must design a system to meet the specified requirements.

The analyst serves another customer indirectly. This customer is the organization. What a system does for which user is critical knowledge for the organization. The value a software organization brings to its customers is an understanding of how the software supports the business. Customers understand their business. Users understand how they use the software to work in the business. However, it is usually only the software professionals that understand what the software does to support the business. Therefore, good requirements are necessary to ensure that you build the right system and also to ensure that the organization safely stores the intellectual capital it possesses.

Approach to Good Requirements

How many times have you heard the phrase "the user does not know what he wants"? This phrase is most often wrong. The user often has an idea of what he wants—he just does not know the specifics. There may be many reasons for this. One reason may be that the user does not know what is possible. Another reason may be that the user is considering the software from a

single point of view. It is up to the analyst to understand what the user wants, communicate what is possible, and then specify what the user wants and exactly what the software must do.

To perform this task, the analyst must be able to move forward with the requirements work in the context of providing feedback to the user and incorporating that feedback into a growing body of knowledge. A practical approach to address these challenges is to use a process that builds knowledge incrementally. At the same time, this practical approach would also augment this process with a structure that saves, relates, and communicates this information coherently. This structure must also help the analyst keep this information consistent.

The process described in this book presents a disciplined approach. It explicitly integrates use cases with various models and the models with the requirements specification. The use cases represent the user requirements. The Software Requirements Specification represents the software requirements. The models provide the analyst with the tools to ensure that user requirements are represented by and consistent with the software requirements.

Benefits of Good Requirements

Good requirements provide many benefits. These benefits impact development and productivity, testing and quality, and the organization. You will enjoy these benefits during the development of new systems as well as during maintenance of existing systems.

Impact of Good Requirements on Development and Productivity

The most visible benefit to this process is faster and higher-quality requirements analysis. A full set of requirements allows an analyst to identify all conflicts between user requirements up front. Identifying all unanswered questions and getting answers to those questions early saves the time and effort spent in building the wrong product and the time and effort spent reworking the products that were based on the poor requirements. A good set of requirements also represents a single source of a system's require-