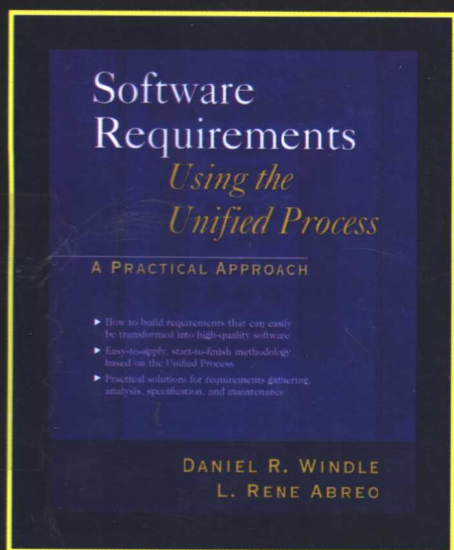


# Software Requirements

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 ■

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 Abroad 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

Copyright © 2003 Pearson Education, Inc.

Original English Language Edition Published by Prentice Hall PTR, Inc.

All rights reserved.

Reprinting edition published by PEARSON EDUCATION NORTH ASIA LTD and CHINA ELECTRIC POWER PRESS, Copyright © 2003.

本书影印版由 Pearson Education 授权中国电力出版社在中国境内（香港、澳门特别行政区和台湾地区除外）独家出版、发行。

未经出版者书面许可，不得以任何方式复制或抄袭本书的任何部分。

本书封面贴有 Pearson Education 防伪标签，无标签者不得销售。

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. —影印本. —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—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

**Preface**    *xiii*

**Acknowledgments**    *xv*

## **PART 1**

**Introducing Good Requirements**    *1*

## **CHAPTER 1**

**The Importance of Good Requirements**    *3*

Rationale for Good Requirements    *3*

Approach to Good Requirements    *4*

Benefits of Good Requirements    *5*

Impact of Good Requirements on Development and Productivity    *5*

Impact of Good Requirements on Testing and Quality    *6*

Impact of Good Requirements on the Organization    *6*

Identification of a Good Requirement    *7*

## **CHAPTER 2**

**Characteristics of Good Requirements**    *9*

Characteristics of a Good Requirement    *9*

Characteristics of a Good Set of Requirements    *16*

Language of Good Requirements    *16*

Communicability of Good Requirements    *17*

**CHAPTER 3****Overview of the Artifacts and the Process 19**

Introduction to the Artifacts 20

Introduction to the Process 22

Moving from Requirements to Design 27

**PART 2****Building the User Requirements 29****CHAPTER 4****Getting to Know the Problem Domain 31**

Researching the Area 32

Reading, Reading, Reading 33

Interviewing the Customer and Users 34

Building the Use Cases 36

**CHAPTER 5****Actors and Use Cases 39**

Defining the Boundaries of the System 41

Moving from Steady State to Steady State 44

Identifying Use Cases 45

Introducing the Change Management System Example 46

**CHAPTER 6****Modeling Use Cases 49**

Diagramming Use Cases 50

Generalizing Use Cases 51

Generalizing Use Cases for the Change Management System	52
Relationships Among Use Cases	52
Diagramming Change Management System Use Cases	52
Packaging Use Cases	54
Packaging the Change Management System Use Cases	54

## **CHAPTER 7**

### **Using Activity Diagrams to Represent Use Cases 57**

Elements of Activity Diagrams	58
Change Management System Activity Diagram	65

## **CHAPTER 8**

### **Writing Use Cases 67**

Template Use	68
Step-by-Step Description	69
Alternative Courses	72
Exceptions and Issues	73
Change Management Use Case	74

## **CHAPTER 9**

### **Using Storyboards to Validate the Use Cases 77**

Presentation of Storyboards to the User	78
Evolutionary Graphical User Interface Presentations	78
Other Diagrams and Pictures	80
Presentation of Use Cases to the User	82
Change Management Systems Storyboards	82



**PART 3****Building the Requirements Architecture 85****CHAPTER 10****Entities and Events as Objects 87**

Classes and Objects 88

Remembered Events as Classes 92

Change Management System Classes 92

**CHAPTER 11****Building a Class Diagram 95**

Generalization 96

Polymorphism 98

Aggregation 99

Association 101

Packaging Classes 102

Change Management System Examples 102

**CHAPTER 12****Using State Transition Diagrams 107**

Introduction to State Transition Diagrams 108

Class-Level State Transition Diagrams 109

Use-Case-Level State Transition Diagrams 111

System-Level State Transition Diagrams 112

Change Management System State Transition Diagram 113

**CHAPTER 13****Use Case Realization****by Means of Sequence Diagrams 115**

Introduction to Sequence Diagrams 116

Realizing Use Cases in Sequence Diagrams 118

Example Sequence Diagram for the Change Management System 121

**PART 4****Building the Specifications 123****CHAPTER 14****Developing a Software Requirements Specification 125**

Tailoring the Standard to Meet Your Needs 127

Specifying Functional Requirements from the Class Diagram 127

Specifying Nonfunctional Requirements 130

Identifying Dependencies Between Requirements 130

**CHAPTER 15****Developing an Interface Requirement Specification 133**

The IRS Template 134

Front Matter of the IRS 134

Reference Section of the IRS 134

Requirement Section of the IRS 135

Protocol and Messages Section of the IRS 137

**CHAPTER 16****Verifying the Software Requirements Specification 139**

Using the Process to Ensure Good Requirements 140

Using Peer Reviews to Ensure Good Requirements 141

Specifying a Test Tree 142

Analyzing the Test Tree 145

Requirements Verification Checklist 147

**PART 5****Using the Requirements Architecture 149****CHAPTER 17****Maintaining the System 151**

Receiving the Change Request 152

Analyzing the Change 153

Analyzing User Needs 154

Identifying New and Changed Use Cases 154

Identifying New and Changed Requirements 154

Implementing a Release 156

**CHAPTER 18****Ensuring Maximum Benefits  
from the Requirements 157**

Benefits to Development and Productivity Revisited 157

Benefits to Testing and Quality Revisited 158

Benefits to the Organization Revisited 159

**PART 6****Appendices 161****APPENDIX A****Planning Model for Requirements Development 163**

Model Work Breakdown Structure 163

Model Effort Allocation 165

Model Schedule Allocation 166

Standard Major Milestones 168

**APPENDIX B****Change Management System Artifacts 169**

- Change Management System Use Cases 170
- Change Management System Class Diagrams 214
- Change Management System State Diagrams 216
- Change Management Software Requirements Specification 219

**Bibliography 249****Index 251**

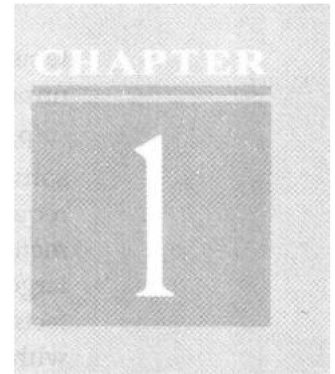
PART

# Introducing Good Requirements

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-