大学计算机教育国外著名教材系列（影印版）

PEARSON Addison Wesley

PROCESS QUALITY ASSURANCE FOR UML-BASED PROJECTS

UML项目管理的过程质量保证

Bhuvan Unhelkar 著

清华大学出版社

大学计算机教育国外著名教材系列(影印版)

Process Quality Assurance for UML-Based Projects

UML 项目管理的过程质量保证

Bhuvan Unhelkar

清华大学出版社
北京

北京市版权局著作权合同登记号 图字：01-2003-8450

图书在版编目(CIP)数据

UML项目管理的过程质量保证＝Process Quality Assurance for UML-Based Projects/昂黑尔纳(Unhelkar, B.)著. —影印本. —北京：清华大学出版社，2004.9
(大学计算机教育国外著名教材系列)
ISBN 7-302-09215-X

Ⅰ. U… Ⅱ. 昂… Ⅲ. ①面向对象语言，UML－程序设计－高等学校－教材－英文 ②软件开发－项目管理－高等学校－教材－英文 Ⅳ. ①TP312 ②TP311.52

中国版本图书馆 CIP 数据核字(2004)第 082085 号

出 版 者：清华大学出版社 地 址：北京清华大学学研大厦
http://www.tup.com.cn 邮 编：100084
社 总 机：010-62770175 客户服务：010-62776969

责任编辑：龙啟铭
印 刷 者：北京季蜂印刷有限公司
装 订 者：三河市召亮装订有限公司
发 行 者：新华书店总店北京发行所
开 本：185×230 印张：27.25
版 次：2004 年 9 月第 1 版 2004 年 9 月第1次印刷
书 号：ISBN 7-302-09215-X/TP・6477
印 数：1～3000
定 价：49.00 元（含光盘）

出版说明

进入21世纪，世界各国的经济、科技以及综合国力的竞争将更加激烈。竞争的中心无疑是对人才的竞争。谁拥有大量高素质的人才，谁就能在竞争中取得优势。高等教育，作为培养高素质人才的事业，必然受到高度重视。目前我国高等教育的教材更新较慢，为了加快教材的更新频率，教育部正在大力促进我国高校采用国外原版教材。

清华大学出版社从1996年开始，与国外著名出版公司合作，影印出版了“大学计算机教育丛书(影印版)”等一系列引进图书，受到国内读者的欢迎和支持。跨入21世纪，我们本着为我国高等教育教材建设服务的初衷，在已有的基础上，进一步扩大选题内容，改变图书开本尺寸，一如既往地请有关专家挑选适用于我国高校本科及研究生计算机教育的国外经典教材或著名教材，组成本套“大学计算机教育国外著名教材系列(影印版)”，以飨读者。深切期盼读者及时将使用本系列教材的效果和意见反馈给我们。更希望国内专家、教授积极向我们推荐国外计算机教育的优秀教材，以利我们把“大学计算机教育国外著名教材系列(影印版)”做得更好，更适合高校师生的需要。

清华大学出版社

影印版序

UML以其多视图、模型化、面向对象特征，被软件工程行业广泛采用作为一种通用的开发方法，基于这种方法之上的RUP从过程的角度提出了用例驱动、以体系结构为中心、迭代增量的统一过程模型，这种模型详细规定了初始阶段、细化阶段、构造阶段和交付阶段的交付项以及迭代增量原则。总体来说，无论UML和RUP都只侧重技术工具和方法学。对于一个成功的项目开发，迭代增量有效对质量管理至关重要。

本书从软件质量保证的原则、质量管理过程的组织与建立以及质量控制等三个部分阐述UML典型开发项目的质量管理活动、模型、过程、度量、控制技术和环境，具有较强的指导作用，可以作为高等院校的软件工程系列教材，也可作为软件工程技术管理人员的工作手册，是一本开发和管理高质量软件项目的重要参考书。

姚淑珍　教授
北京航空航天大学软件学院

Praise for Bhuvan Unhelkar's
Process Quality Assurance for UML-Based Projects

Although Unhelkar's approach to UML and Process Quality appears in this book, it is far from "bookish." With some support from the author, the ideas outlined in this book have been of immense value to us in our large-scale CRMS implementation. This work is direct and practical.

Murray Heke
RMS Programme Director
Aon Risk Services

Using the UML in a project without due consideration to the various roles such as Business Analysts, System Designers, and Architects can be a major source of confusion, frustration, and, at worst, rejection. Unhelkar's practical exposition of modeling in the problem, solution, and background spaces addresses these issues. Worth applying in any project.

Andrew Powell
Chief Technology Officer
Merryl-Lynch Australia

In this extremely well-articulated view on software quality, Unhelkar has also proved the crucial role of UML in facilitating communications in project teams. The judicious combination of modeling, quality process, and quality management takes this book beyond just UML, and makes it applicable to all aspects of a software project. This work will be of tremendous interest to practitioners.

Prabhat (S D) Pradhan
CEO
iOpsis Software
San Francisco

We have run a series of successful two-day seminars in Mumbai, Bangalore, and Delhi, India, based on this exceptionally well-researched and well-presented work by Dr. Unhelkar. Our clients have found the ideas presented in this book extremely relevant to their effort in adopting process and modeling discipline, enabling them to climb up the CMM process maturity ladder. Well worth more than one read!!

Navyug Mohnot
CEO
QAIindia
Delhi, India

We started using UML along with the Quality Software Process for our consulting projects, as outlined here by Unhelkar, resulting in vastly improved communication with our clients and superior control of our projects. This book is a must read for practitioners.

Ashish Kumar
President and CEO
Jagat Technology, Inc.

Once in a long while does a manuscript come the reviewer's way that seems adequate in every respect. This is one of those instances. One does not have to be devoured by a tiger to realize one is present in the thicket; foot-prints shall suffice. The sample chapters were ample evidence that this book has every potential to succeed. This book is definitely for the professional, although I can see it used by academics and even as a

supplemental textbook. The presence of end-of-chapter exercises certainly assists in this latter potential. To reach this wide array of audiences, again, one might add that the author has done well in striking a balance between depth and accessibility.

Prof. Houman Younessi
Rensselaer at Hartford
Hartford, Connecticut

Teaching and learning ìsoftware qualityî is always a challenge because of the elusive nature of quality. This book simplifies the task of teaching quality considerably. Each chapter is neatly outlined for a quality subtopic, followed by relevant FAQs and exercises. This work is an excellent blend of theory with practical examples that run consistently throughout the book.

Prof. Murat M. Tanik
University of Alabama at Birmingham
Birmingham, Alabama

Understanding and teaching quality requires a two-pronged approach. In the context of UML-based software development, first the UML-based models need to be subjected to rigorous quality checks of syntax, semantics, and aesthetics. Second, it is necessary to put in place the management and process aspect of development that would ensure creation of appropriate and good quality models in the first place. This second aspect is the focus of this book, as discussed in Chapters 2 through 4. While the introduction to the elusive nature of quality in Chapter 1 is extremely well-researched, Unhelkar has also demonstrated his practical experience in quality and UML in Chapter 5, "Estimates and Metrics." This is an excellent "accompanying" textbook for any course in quality, process, modeling, and management.

Prof. Athula Ginige
Head of School of Computing and Information Technology
University of Western Sydney
Australia

As the author of a commercial software process myself, I am pleased to see a book focus on process and quality rather than pure UML modeling. The focus on the process aspect of quality is the hallmark of this work. Certainly a process practitioner's material suitably interspersed with theory and anecdotes.

Dr Julian Edwards
Principal author of Process Mentor
Object Oriented Pty. Ltd.
Sydney, Australia

This work judiciously blends the crucial people aspect with software quality. Very practical, entertaining, and readable.

Ramin Marzbani
CEO
ACNielsen.consult
Australia

Foreword

When I agreed to write the foreword for this book, I expected to battle my way through a morass of technical jargon. I was willing to do this because the Unified Modeling Language (UML) is an approach widely used at the Raytheon Garland, Texas, site and I wanted to expand my knowledge of it. I must honestly admit that I started reading the text with dread and with a rather basic (and somewhat cynical) question: Other than a general understanding of what UML is, why should I care about it?

In my position as Director of Software Engineering, I rarely need detailed knowledge of development languages and tools, not to mention the fact that these development aids are constantly changing. UML, however, seems to have caught the attention of my software engineers, so I marched forward into *Process Quality Assurance for UML-Based Projects*. I convinced myself that quality assurance and process were at least familiar words and an integral part of my job.

Then something amazing happened: I couldn't put the book down. The introduction to the elusive nature of quality is consistent with my own views, and these views are well expressed (and frequently humorous). The historical facts behind UML (Booch, Jacobson, and Rumbaugh creating UML), the definition of UML (a means of visualizing, constructing, and documenting object-oriented software artifacts through a standard set of notations, diagrams, and specifications), and the relevance of UML to practical modeling are stated succinctly. The text deepens the reader's understanding of the direct and practical aspects of modeling with UML through the discussion of UML-based CASE tools and processes, which can contribute to

requirements modeling and testing as well as system, database, and architectural design. As I read through the text, I condensed the information into the following major concepts:

- UML standardization is a good thing for large programs.
- UML supports major program activities associated with requirements, testing, design, and implementation.
- UML is effective in reducing the complexity of and enhancing the ability to test reality.
- UML addresses some of the most serious problems in system development, especially satisfaction of user expectations.
- UML helps mitigate the long-standing problems related to the intangibility of software.
- UML helps clarify interdependencies, which are common problem points in system integration.
- UML is not a monolithic construct—it has a different relevance and application in each of the three modeling spaces of problem, solution, and background (architecture).
- UML transition should be planned and handled carefully, particularly in separating UML techniques from its CASE tools and processes.

By the end of the first reading session, I became a fan of UML and Bhuvan Unhelkar's writing style. Instead of seeing UML as a software mystery understood by only the most esoteric guru or hard-core programmer, the reader begins to see UML as a value-added tool. In the quest for quality, UML supports system development through reduction of complexity, clarification of requirements, and addition of structure to the design and integration process.

One of the strong points of the text is the "big-picture" view of an entire program-development cycle. Unhelkar approaches the discussion of UML in the context of each modeling space and the process support required in each of these modeling spaces. The text is distinguished through the emphasis on definition of roles required to perform each process activity.

Many technical texts focus on the *what*, *when*, and *how*, but neglect the *who*. In this work, Unhelkar recognizes the importance of the sociological aspect of system development. In my own experience, I have seen system-integration program managers begin to recognize the psychological traits

and attitudes of the program developers as valid parameters in determining and managing the program constraints of cost, schedule, and quality.

Traditionally, system development has been organized around technology and methodology. Nonquantifiable aspects of program success such as communication have been overlooked, despite the fact that poor communication accounts for a large percentage of project failures. Program managers focus on the triple constraints—quality, schedule, and cost. Unhelkar inspires the reader to also look at the second tier beneath each of these constraints—people, technology, and process. While making it clear that UML is not a process, the text addresses process-components with their respective roles, responsibilities, activities, tasks, and deliverables that produce UML-based artifacts.

Unhelkar has validly recognized that people are the key to building quality systems, and that training, organizing, empowering, and managing people are essential to producing a quality product. Emphasis on the human factors related to system developers and system users is found throughout the text. For example, Unhelkar states, "Proper study of quality would have to delve into other branches of science such as psychology, sociology, and social sciences."

To develop a system, many competencies—both engineering and nonengineering—are required. Some of the less-technical competencies addressed include understanding the user, establishing trust relationships, evaluating and selecting component suppliers, developing cost estimations, preparing test plans, balancing cost and quality, balancing reuse against customer requirements and system quality/cost/schedule, creating and managing schedules, managing system configuration, developing flexible designs, and selecting and using appropriate metrics.

Obviously, finding individual engineers capable of fulfilling all the required roles in system development is a difficult task, if not an impossible one. The solution lies in the ability to form teams composed of members whose skills are complementary and cover the gamut of necessary roles. Issues that affect the ability to form such complementary and effective teams include project size, geographical distribution of team members, and inherent communication issues arising from the possible lack of face-to-face interactions.

The text addresses ways to assemble teams, the value of diversity, and issues arising in a virtual environment. Unhelkar also touches on profiling team members to help each one determine roles best aligned with their

skills and interests. Domain expertise and mentoring are addressed and presented as tools to aid in team coalescence.

Unhelkar's belief in the value of mentoring is demonstrated in his approach to writing this text. He does not seem interested in impressing the reader with his technical jargon, but rather, he sets the stage for learning about the UML and its relation to quality by virtue of its application within the framework of a process. His analogies and examples are entertaining as well as instructive as he steps logically through the text material.

As a mentor to system-development professionals, he clarifies who should develop or review UML models of the problem space, the solution space, and the background space (architecture) of the system. He provides commonsense suggestions for determining the applicability of UML to a project, and for planning UML training and mentoring. His instruction is encapsulated in FAQs and exercises at the end of each chapter. As a result, the text can serve as an excellent seminal reference for practitioners, academics, and researchers. The text might also be selected as an excellent basis for a class or a self-study group as well as an ideal textbook used in a transdisciplinary or interdisciplinary engineering curriculum.

Unhelkar's practical understanding of issues that confront those who build systems adds a dimension to the text that is lacking in many technical writings. I applaud this unique blend of theory and practice and highly recommend this text as a reference for software-system projects, software engineering courses, and research in the areas of quality, modeling, and process. My best wishes are with the author for this influential work, and I offer encouragement for him to pursue future projects, especially in the areas of product integration, metrics, cost estimation, and sociological/psychological factors affecting project success. I believe that my enthusiasm for *Process Quality Assurance for UML-Based Projects* will be shared by all who read it.

Vicki P. Rainey, Ph.D.
Director, Software Engineering
Raytheon—Garland Texas Site

Preface

Quality is subjective[1]

Purpose of This Book

This book is about the *process* aspect of quality assurance in UML-based projects. Process is one of the two major aspects of software quality assurance—the other being *modeling*. This book is written with an aim of directly addressing the paucity of literature in the area of quality assurance for UML-based projects—with a specific focus on process.

This is because despite its popularity, the UML literature needs discussion on and application of quality *with* UML. While we have some excellent literature on the processes of software development (most notably *The Unified Process* by Jacobson et al. and *The OPEN Process Specification* by Ian Graham et al.) it seems to fall short of separate and detailed discussions on quality.

On the other hand, works like Binder's *Testing Object Oriented Software* focus on the technical aspects of testing using the UML notations, but they do not provide the process aspect of improving the quality of software development. Indeed, none of this literature deserves any criticism for the lack of "quality" discussion—because these literary works do not purport to be discussing quality. The focus of these respectable and popular works

[1] This is such a universal statement, it must have come from you!

is either development or testing. This book in your hand complements the current literature in the UML arena.

Good quality is all about satisfying the needs of the user. However, "good" is a highly subjective term. The reference point against which quality is judged depends on time, place, and situation—all of which can change! Hence, the essential ingredients in producing good quality are:

- A product that satisfies the changing needs of the user
- A process that enables the creation, verification, and validation of such a product
- A common mechanism to establish communication
- Continuous improvement of the process of producing product

When applied to software development, these quality requirements translate into producing a software product that evolves, scales, and changes according to the needs of its users—primarily the business. We not only need a process for developing such a software product, but we also need significant checking and cross checking of the models and processes that have been used to construct the software product. There is a need to create, follow, and check all necessary process steps in order to achieve maturity of processes that result in good quality software products. These process steps must be executed iteratively, incrementally, and sufficiently.

Process steps should also be malleable enough to suit various development environments and various types and sizes of projects. These are some of the specific and significant areas of process-quality-related work required in a project incorporating the UML that are addressed in this book. Some of this quality work includes how to organize the overall quality function, the process steps to be followed in the creation of UML diagrams, the steps in the verification and validation of these diagrams, when to conduct such verification, how to interpret the results of quality activities, who should create and validate the UML diagrams, and how to create a quality control (testing) strategy.

These process steps eventually result in good quality models. Quality, however, is further enhanced by applying quality checks to the software models—ensuring their syntactical correctness, semantic consistency, and aesthetic symmetry. For detailed analysis and discussion of the model quality of UML diagrams, readers are encouraged to peruse *Model Quality Assurance of UML-Based Projects* (due 2003).

Summary of the Book

This book is divided into six chapters as summarized in the table below.

Chapter	Description
1. The Quality Game	Builds the background theory and arguments for quality
2. Quality Environment: Managing the Quality Function	Quality management, team formation, sociology, and psychology of quality teams; importance of process
3. The Quality Process Architecture	Process-components encompassing activities, tasks, deliverables, and roles that make up a Quality Software Process
4. Enacting the Quality Software Process	Quality process in practice; iterations, increments, and parallel development
5. Estimates and Metrics for UML-Based Projects	Some suggestions on practical estimation of time, budgets, and people for UML-based projects
6. Quality Control of Software Products	Detailed discussion on strategies for quality control and testing

Chapter 1: The Quality Game

In this background chapter on quality assurance we discuss the elusive nature of quality in the context of software. Modeling, particularly with the UML, is shown as a means to improve communication and quality and is conducted in the three distinct yet related modeling spaces of problem, solution, and background. We discuss process in the context of its three dimensions of technology (what), methodology (how), and sociology (who). This is followed by a discussion on the various checks (syntax, semantics, and aesthetics) needed to validate and verify UML-based models and the checks of necessity, sufficiency, and malleability needed for a good quality process. Chapter 1 also discusses the organization of the quality function and its application to various types of projects (development, integration, package implementation, outsourcing, data warehousing, and educational) as well as various sizes of projects (small, medium, and large).

Chapter 2: Quality Environment: Managing the Quality Function

The process aspect of quality encompasses the management functions of creating and managing a quality environment. This is because software quality is not just about verifying and validating what has been produced; it's also about sustaining an effort to follow the discipline of producing models and software. This discipline encompasses the process or the steps involved to produce good models and good software. This part of the book comprehensively considers the organization and execution of the quality function, with a detailed emphasis on the process of developing UML-based software. In other words, we discuss "how" the quality function is organized and carried out in UML-based projects. The people issues ("who") are also given due relevance in Chapter 2.

Chapter 3: The Quality Process Architecture

This chapter discusses what constitutes such a process and how it is helpful in enhancing quality in a UML-based project. This chapter does not propose a new process, but discusses a most generic process including the technological, methodological, and sociological dimensions—what constitutes a process, and its major dimensions. The technological dimension of a process deals with the "what," the methodological dimension with the "how," and the sociological dimension with the "who" of an overall process. These dimensions are described with common workday examples. Furthermore, the generic process also describes the most commonly used activities and tasks that should be present in any process. These activities and tasks and their related roles and deliverables are described with the aim of improving the discipline in a process, resulting in the enhanced quality of UML-based deliverables, and eventually the software product.

Chapter 4: Enacting the Quality Software Process

In this chapter we discuss the enactment of an example process including the practical issues of configuring an Iterative, Incremental, and Parallel (IIP) project plan, based on the process-components discussed in the previous chapter. We also discuss practical issues of tracking the progress

of a project as well as modifying the project plan based on that tracking. An iterative and incremental project plan facilitates better absorption of changes than a sequential project plan. The creation and management of such a changing plan, derived from the malleability aspect of the process, is also discussed. This chapter discusses what happens when the "rubber hits the road" in terms of application of a process.

Chapter 5: Estimates and Metrics for UML-Based Projects

This chapter discusses the important issues of measurements and estimates in UML-based software projects. Starting with an argument for the need to make good estimates and how good metrics help to make good estimates, this chapter delves into the importance of these measures and estimates in improving the quality of models and processes in the project. Technical measures related to sizes and complexities of the UML artifacts and diagrams are also discussed. Estimates for an example implementation project using the UML are shown with a view to demonstrate the application and significance of metrics in a practical project.

Chapter 6: Quality Control of Software Products

This chapter discusses in detail the quality control and testing aspect of a quality lifecycle. While we discuss process quality in the previous chapters, quality control (testing) is a major process-component dedicated to verifying and validating the results of our efforts thus far in creating models and following a process. Good quality control is inherently negative, as it is aimed at breaking everything in a system—its logic, its execution, and its performance. Thus, although quality control is an integral part of quality assurance, it is not synonymous with it. This separation is given its due importance in this separate part of the book.

CD-ROM and Potential Web Support

The CD-ROM contains details of the chapters, diagrams, and a set of templates (deliverables, project plan, and so forth) that can be customized for use in projects. Suggested metrics for improving quality (for example, size

of use cases and effort in creating classes) are also incorporated on the CD-ROM. With permission, evaluation copies of relevant process tools that deal with quality process are also provided.

Literary Audience

There are numerous books written on the UML and on processes. Their scope encompasses both academic research and practical applications. This book attempts to synergize the application of quality processes in UML-based projects. With the process focus, the reader is expected to be familiar with the UML and its modeling technique, as the book does not purport to discuss the modeling techniques of the UML.[2] However, a person responsible for quality assurance will find this work self-sufficient and may even be encouraged after reading this material to extend their

Chapters	Quality Manager	Project Manager	Tester	Process Engineer	System Designer	Developer	Business Analyst	System Architect	Academic	Director
1. The Quality Game	**	**		**					**	**
2. Quality Environment: Managing the Quality Function	**	*		*	*		*	*	**	*
3. The Quality Process Architecture	**		*	**		*	*	*	**	
4. Enacting the Quality Software Process	**		*	**	*	*	*	*	*	
5. Estimates and Metrics for UML-Based Projects	**	*		*					**	*
6. Quality Control of Software Products	**		**		*	*	*	*	*	**

[2] This is discussed in my forthcoming book *Model Quality Assurance of UML-Based Projects*, to be published by Addison-Wesley.