国外优秀信息科学与技术系列教学用书

软件评估、度量与最佳方法

（影印版）

SOFTWARE ASSESSMENTS, BENCHMARKS, AND BEST PRACTICES

■ Capers Jones

高等教育出版社
Higher Education Press
Pearson Education
出 版 集 团

国外优秀信息科学与技术系列教学用书

软件评估、度量与最佳方法

（影印版）

SOFTWARE ASSESSMENTS, BENCHMARKS, AND BEST PRACTICES

Capers Jones

高 等 教 育 出 版 社

Pearson Education 出版集团

图字：01-2002-6600 号

Software Assessments, Benchmarks, and Best Practices, First Edition
Capers Jones

图书在版编目（CIP）数据

软件评估、度量与最佳方法/（美）琼斯（Jones，C,）编. —影印本. —北京：高等教育出版社，2003.1
ISBN 7-04-011770-3

Ⅰ.软... Ⅱ.琼... Ⅲ.软件开发—评估—高等学校—教材—英文 Ⅳ.TP311.52

中国版本图书馆 CIP 数据核字（2003）第 001134 号

出版发行	高等教育出版社	购书热线	010-64054588
社址	北京市东城区沙滩后街 55 号	免费咨询	800-810-0598
邮政编码	100009	网址	http://www.hep.edu.cn http://www.hep.com.cn
传真	010-64014048		
经销	新华书店北京发行所		
印刷	北京中科印刷有限公司		
开本	787×1092　1/16	版次	2003 年 1 月第 1 版
印张	42.75	印次	2003 年 1 月第 1 次印刷
字数	850 000	定价	49.50 元

本书如有缺页、倒页、脱页等质量问题，请到所购图书销售部门联系调换。

出版说明

20世纪末，以计算机和通信技术为代表的信息科学和技术对世界经济、科技、军事、教育和文化等产生了深刻影响。信息科学技术的迅速普及和应用，带动了世界范围信息产业的蓬勃发展，为许多国家带来了丰厚的回报。

进入21世纪，尤其随着我国加入WTO，信息产业的国际竞争将更加激烈。我国信息产业虽然在20世纪末取得了迅猛发展，但与发达国家相比，甚至与印度、爱尔兰等国家相比，还有很大差距。国家信息化的发展速度和信息产业的国际竞争能力，最终都将取决于信息科学技术人才的质量和数量。引进国外信息科学和技术优秀教材，在有条件的学校推动开展英语授课或双语教学，是教育部为加快培养大批高质量的信息技术人才采取的一项重要举措。

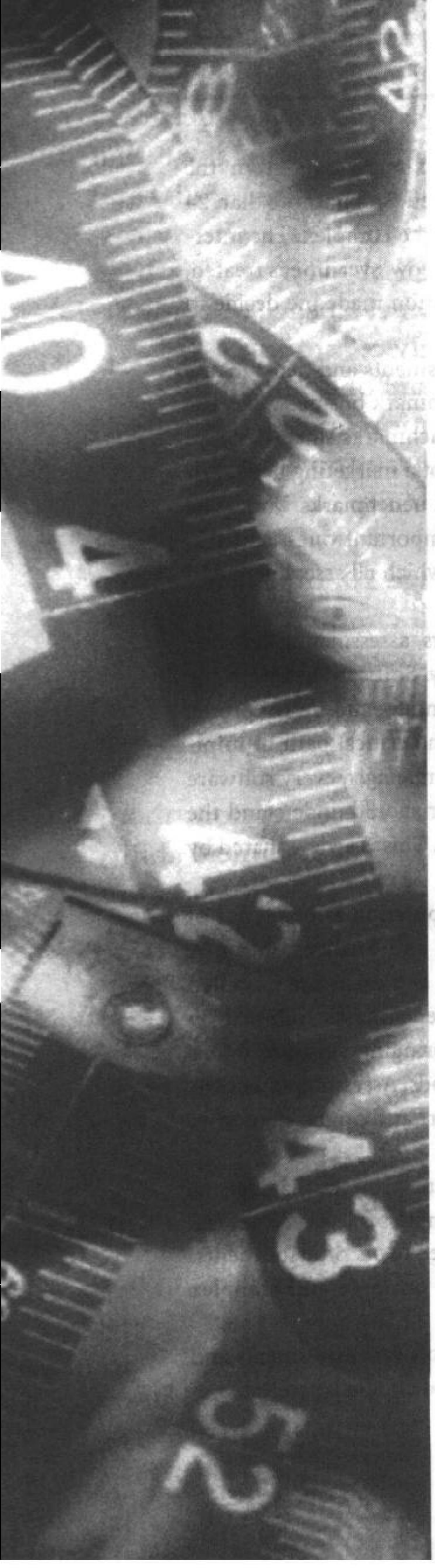
为此，教育部要求由高等教育出版社首先开展信息科学和技术教材的引进试点工作。同时提出了两点要求，一是要高水平，二是要低价格。在高等教育出版社和信息科学技术引进教材专家组的努力下，经过比较短的时间，第一批由教育部高等教育司推荐的20多种引进教材已经陆续出版。这套教材出版后受到了广泛的好评，其中有不少是世界信息科学技术领域著名专家、教授的经典之作和反映信息科学技术最新进展的优秀作品，代表了目前世界信息科学技术教育的一流水平，而且价格也是最优惠的，与国内同类自编教材相当。这套教材基本覆盖了计算机科学与技术专业的课程体系，体现了权威性、系统性、先进性和经济性等特点。

目前，教育部正在全国35所高校推动示范性软件学院的建设，这也是加快培养信息科学技术人才的重要举措之一。为配合软件学院的教学工作，结合各软件学院的教学计划和课程设置，高等教育出版社近期聘请有关专家和软件学院的教师遴选推荐了一批相应的原版教学用书，正陆续组织出版，以方便各软件学院开展双语教学。

我们希望这些教学用书的引进出版，对于提高我国高等学校信息科学技术的教学水平，缩小与国际先进水平的差距，加快培养一大批具有国际竞争力的高质量信息技术人才，起到积极的推动作用。同时我们也欢迎广大教师和专家们对我们的教材引进工作提出宝贵的意见和建议。联系方式：hep.cs@263.net。

高等教育出版社
二〇〇二年九月

To Eileen—for making all things possible



Preface

During my writing, this book evolved considerably from the first plan. Originally I intended to divide the book into two major sections. The first section was to discuss a number of assessment and benchmark methods used in the United States and Europe. The second section was to present an overview of software productivity and quality benchmarks, and associated "best practices" derived from benchmark studies. The benchmarks and best practices in this book cover six major kinds of software project: (1) management information system (MIS) projects, (2) outsource projects, (3) systems and embedded software projects, (4) commercial software projects, (5) military software projects, and (6) personal software projects developed by end users.

However, as the writing commenced, the focus of the book began to change. It soon became clear that a complete discussion of benchmarks and best practices for each of the six kinds of software would be about twice as large as initially planned. I had planned to devote approximately 30 pages to the benchmark and best-practice information for each type of software. But to do justice to the available data, almost 60 pages were needed for five of the six forms of software. Furthermore, a discussion of how assessment and benchmark studies operate and their technical differences may be of interest to those of us in the assessment and benchmark business, but it is not necessarily of great interest to those outside the limited circle of benchmark consultants.

As a result, the discussion of assessment and benchmark methods was cut back, and the sections devoted to information gathered during assessments and benchmark studies was expanded. Instead of a book with two sections of roughly equal size, the book now has a briefer introductory section and greatly expanded discussions of each type of software and the issues that confront each type.

This book also emphasizes assessments and benchmark data from the United States. Although my colleagues and I have gathered data in more than 24 countries, the issues of international benchmarks are quite complex. The international variations in working years and working days, how overtime is treated, and European restrictions on some kinds of data collection made me decide to concentrate on U.S. data.

Readers should note that this is a book about assessments and benchmarks written by someone who is in the assessment and benchmark business. Because my company has been performing assessments and benchmarks since 1985, we have an obvious interest in the topic. However, this is not a marketing book, nor is it a book about how my company's assessments and benchmarks work. The topics of software assessments and benchmarks are important ones, and this book attempts to include the general principles under which all assessment and benchmark consulting groups operate.

In my view, and also in the view of my competitors, assessments and software benchmarks are important to the global economy. Software has been the most labor-intensive product of the twentieth century, and the most error prone. Assessments, benchmarks, and empirical data are on the critical path to minimizing software project failures. Every software project manager, every software quality assurance specialist, and every software engineer should understand the basic concepts of software assessments and benchmarks. This is a view shared by all of the assessment and benchmark consulting groups.

The software industry has achieved a notorious reputation as being out of control in terms of schedule accuracy, cost accuracy, and quality control. A majority of large systems run late, exceed their budgets, and many are cancelled without ever reaching completion. Assessment and benchmarks followed by planned process improvement programs can aid in bringing software under management control. These are not "silver bullet" methods. Assessments, benchmarks, and process improvement programs require effort and can be expensive, but project failures are far more expensive.

This book discusses the kinds of complex software projects that benefit from assessments and benchmark studies. Small and simple projects are not the main focus of assessments and benchmarks. The proper focus of assessments, benchmarks, process improvements, and this book is on large and complex applications.

Chapter 1 provides an introduction to the topic of software assessments and benchmarks. This chapter discusses the kinds of data that should be collected. It also cautions against some common problems, such as depending on data without validating it, and using hazardous metrics such as *lines of code.* This chapter also discusses the need to keep client data protected, and suggests some coding methods that can be used to perform benchmarks without revealing proprietary client information.

Chapter 2 deals with the history of software process assessments and discusses some of the kinds of information that are gathered during software process assessments. Although more than a dozen forms of assessment exist, the form made popular by the Software Engineering Institute is the best known. Some recent and more specialized forms of assessment, such as those performed for the Y2K problem, have also been widely used since about 1998.

Chapter 3 deals with the related topics of software benchmarks and software baselines. *Benchmarks* collect and compare quantitative data against industry norms. *Baselines* measure the rate at which a company can improve productivity and quality when compared with an initial starting point. Of course, sometimes productivity and quality can get worse instead of better.

Chapter 4 discusses 36 key factors that should be recorded during assessment and benchmark studies. If these 36 key factors are recorded, the data gathered by almost any benchmark and assessment consulting group, or by any company or government agency, could be compared meaningfully.

Chapter 5 addresses an important topic that is somewhat ambiguous in the software literature. When we speak of "best practices" what exactly do we mean? Chapter 5 discusses some criteria for including or excluding tools and technologies from best-practice status. It is suggested that any technology considered a potential best practice needs empirical results from at least ten companies and 50 projects.

Chapter 6 discusses an important follow-on activity to assessment and benchmark studies. Both assessments and benchmark studies are diagnostic in nature, rather than therapeutic. These studies can identify problems, but they cannot cure them. Therefore, a natural follow-on activity to either an assessment or a benchmark analysis, or both, would be to implement a process improvement program.

Chapter 7 deals with benchmarks and best practices for MIS projects. These are software applications that companies and government agencies build for their own internal use. MIS applications are often keyed to large corporate database access, and their main purpose is to convert raw data into useful information. Although MIS projects are often fairly high in productivity for small projects, large MIS projects tend to experience more than average failure rates. Quality at the large end is often poor too.

Chapter 8 deals with benchmarks and best practices for outsource software projects. The emphasis in this chapter is on projects under contract for MIS, rather than for military or systems software outsourcing. The major outsource vendors such as Andersen, Electronic Data Systems, and IBM concentrate on the MIS market because it is the largest market for their services. In general, outsource projects have higher productivity and quality levels than in-house MIS projects; however, litigation between clients and outsourcers does occur from time to time.

Chapter 9 deals with benchmarks and best practices for systems and embedded software, which are applications that control physical devices such as computers, telephone switching systems, aircraft flight controls, or automobile fuel injection systems. The close coupling of systems software to physical hardware devices has led to very sophisticated quality control methods. The systems software community has the best track record for applications larger than 10,000 function points.

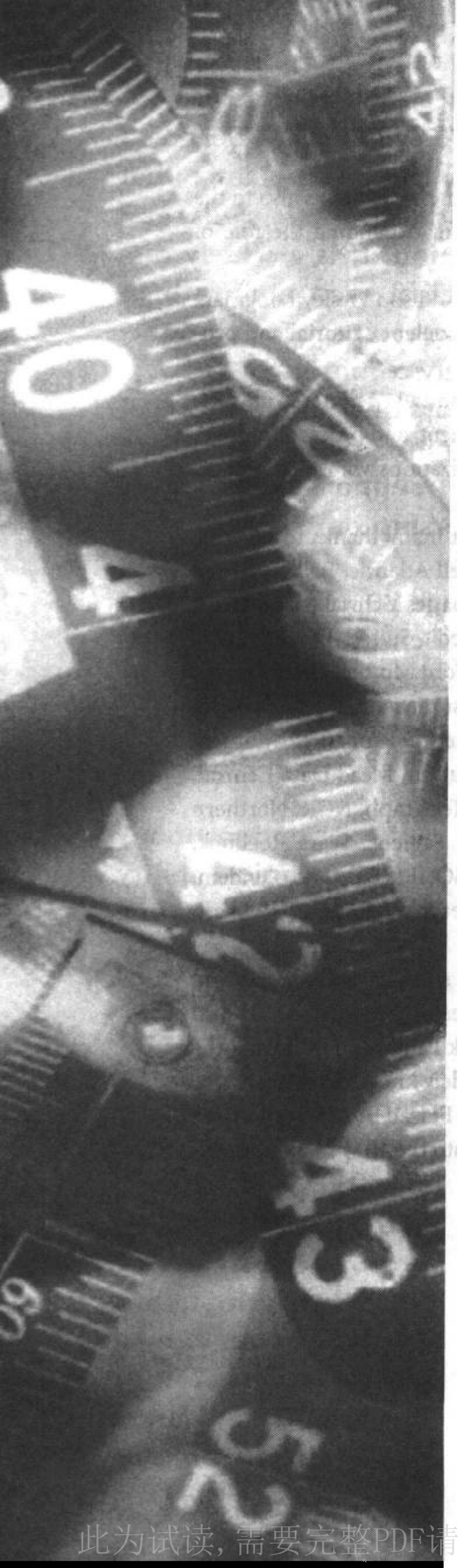
Chapter 10 deals with benchmarks and best practices for commercial software. Commercial software applications are intended for the mass market, and some of these applications are used by millions of customers on a global basis. The commercial and systems software domains overlap in the arena of operating systems because commercial products such as Windows 98 are both systems and commercial software. The commercial world needs to deal with special issues such as translation and nationalization of packages, piracy, and very extensive safeguards against viruses.

Chapter 11 deals with benchmarks and best practices for military software, with special emphasis on the U.S. armed services and the Department of Defense. The military software domain is fairly good at building large and complex applications, although military software productivity is lower than any other domain. The legacy of U.S. military standards has left the defense community with some very cumbersome practices. Plans and specifications in the military domain are approximately three times larger than equivalent civilian projects. The bulk is due primarily to military oversight requirements, rather than to the technical needs of the project.

Chapter 12 deals with benchmarks and best practices for end user software development. As the new century begins, there are more than 12,000,000 U.S. office workers who know how to write computer programs if they wish to do so.

By the middle of this century, the number of computer-literate workers in the United States will top 125,000,000. Indeed, there are some signs that computer literacy will actually pull ahead of conventional literacy in the sense of being able to read and write. End user applications are currently in a gray area outside the scope of normal assessments and benchmarks. More importantly, end user applications are also in a gray area in terms of intellectual property law. As end user applications become more and more numerous, it is important to set policies and guidelines for these ambiguous applications.

As this book is written, benchmarks based on function point metrics are dominant in the software world, except for military software, in which benchmarks based on lines of code still prevail. This book utilizes function point metrics and cautions against lines-of-code metrics for benchmarks involving multiple programming languages. Version 4.1 of the function point rules defined by the International Function Point Users Group is the standard metric used throughout.



Acknowledgments

As always, special thanks to my wife, Eileen Jones, for her help. She makes all of my books possible. Eileen handles all of our publishing contracts, and by now knows the details of these contracts as well as some attorneys. Thanks also for her patience when I get involved in writing, even on holidays.

This manuscript is being finished approximately two years after the merger of Software Productivity Research (SPR) with Artemis Management Systems. Many thanks to Steve Yager, the president of Artemis, for his interest and support of SPR's assessment and benchmark studies. Thanks also to the many Artemis colleagues and distributors around the world.

Thanks also to Alec Gores and Vance Diggins of the Gores Group for including SPR in such an excellent organization.

Special thanks to Charles Douglis, SPR's president, for many years of leadership and friendship. Great appreciation is due to all of my colleagues at SPR for their aid in gathering data. Special thanks to the families and friends of the SPR staff, who have had to put up with lots of travel overtime. Thanks to Ed Begley, Chuck Berlin, Amy Bowers, Julie LeBaron, Michael Bragen, Doug Brindley, Jack Boyle, Tom Cagley, Sudip Charkraboty, Michael Cunnane, Gail Flaherty, Richard Gazoorian, Mike Griffin, David Gustafson, Bill Harmon, Bob Haven, Steve Hone, Jan Huffman, Peter Katsoulas, John Mulcahy, Joy Mohr, Donna O'Donnell, Mark Pinis, Mickie Prault, Tom Riesmeyer, Evelyn Rooney, Kathy Russell, Todd Santora, Keith Savage, Pam Simonvich, Bill Walsh, and John Zimmerman.

Special thanks to Allan Albrecht, the inventor of function points, for his invaluable contribution to the industry and for his outstanding work with SPR. Without Allan's pioneering work in function points, the ability to create

accurate baselines and benchmarks would probably not exist. Thanks also to the International Function Point Users Group (IFPUG) for expanding the role of function points.

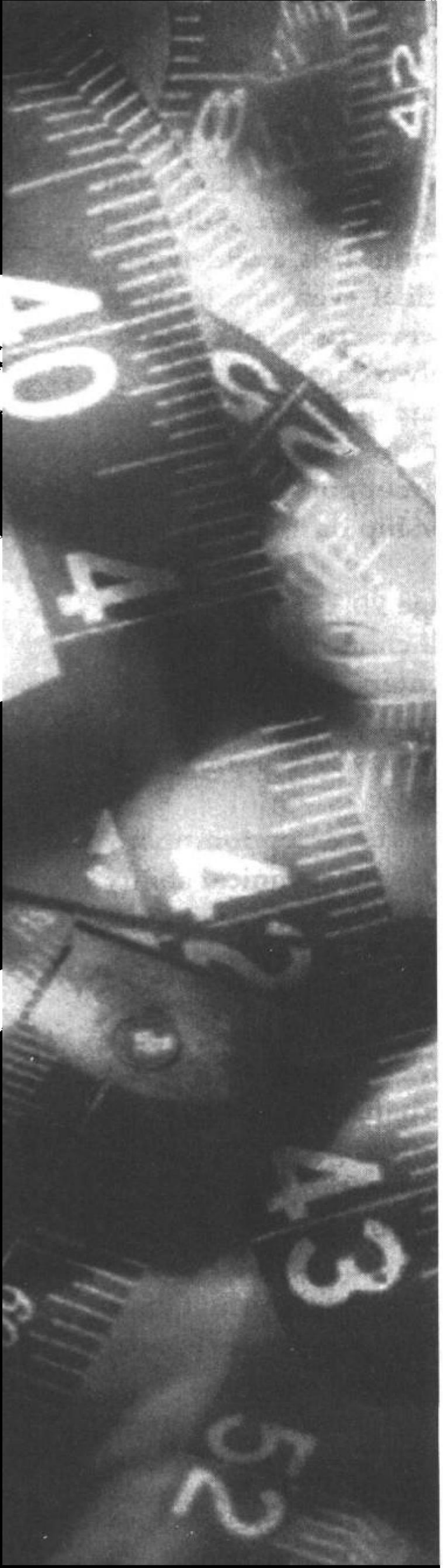
Thanks also to Mary O'Brien and her colleagues at Addison Wesley Longman. It is always enjoyable for an author to work with an excellent editorial and production team.

Much appreciation is due to the client organizations whose interest in software assessments, benchmarks and baselines, measurement, and process improvements have let us work together. These are the organizations whose data make estimation tools possible.

There are too many groups to name them all, but many thanks to our colleagues and clients at Andersen Consulting, AT&T, Bell Atlantic, Bell Northern Research, Bell Sygma, Bendix, British Air, CBIS, Charles Schwab, Church of the Latter Day Saints, Cincinnati Bell, CODEX, Credit Suisse, DEC, Dun & Bradstreet, DuPont, Electronic Data Systems, Finsiel, Ford Motors, Fortis Group, General Electric, General Motors, GTE, Hartford Insurance, Hewlett-Packard, IBM, Informix, Inland Steel, Internal Revenue Service, JC Penney, JP Morgan, Kozo Keikaku, Language Technology, Litton, Lotus, Lucent, Mead Data Central, McKinsey Consulting, Microsoft, Motorola, Nippon Telegraph, NCR, Northern Telecom, Bell Atlantic, Pacific Bell, Ralston Purina, Sapiens, Sears Roebuck, Siemens-Nixdorf, Software Publishing Corporation, SOGEI, Sun Life, Tandem, TRW, Unisys, U.S. Air Force, U.S. Navy surface weapons groups, US West, Westinghouse, and many others.

Thanks also to my colleagues and competitors in software assessments and benchmark studies. Without the pioneering work of researchers such as Howard Rubin, Watts Humphrey, Bill Curtis, Ron Radice, Mark Paulk, and Bob Park, the assessment and benchmark domains would not be widely known in the software world. Special thanks to the late Ted Climis and Jim Frame of IBM, who were pivotal in introducing assessments and benchmarks into the IBM software community.

Capers Jones



Abstract

Benchmarks are collections of quantitative data that compare an enterprise against other similar enterprises in the same industry. Software benchmarks are often used for productivity and quality comparisons, and also for comparisons of salaries and spending levels.

Baselines are collections of quantitative data used to mark the starting point of a process improvement program. Baselines and benchmarks are similar in that they both collect quantitative data. Baselines are often used with outsource agreements that serve as the starting point for contractual obligations to improve productivity and quality.

Software assessments are on-site reviews of the methods, tools, and processes used to develop software applications. Assessment data is qualitative in nature. Assessments are used to place organizations on a comparative scale and show relative levels of capability.

When assessment data, benchmark data, and baseline data are combined, a powerful synergy results. The combination of assessments, benchmarks, and baselines can identify best practices, average practices, and worst practices. The term *best practices* refers to methodologies, tool suites, and organizational structures that raise software quality and productivity levels above average by at least 15% in repeated trials within at least ten companies and 50 projects.

Contents

CHAPTER 4 THIRTY-SIX KEY FACTORS FOR SOFTWARE ASSESSMENT AND BENCHMARK STUDIES 115

CHAPTER 5 IDENTIFYING SOFTWARE BEST AND WORST PRACTICES 127

CHAPTER 6 SOFTWARE PROCESS IMPROVEMENTS 143