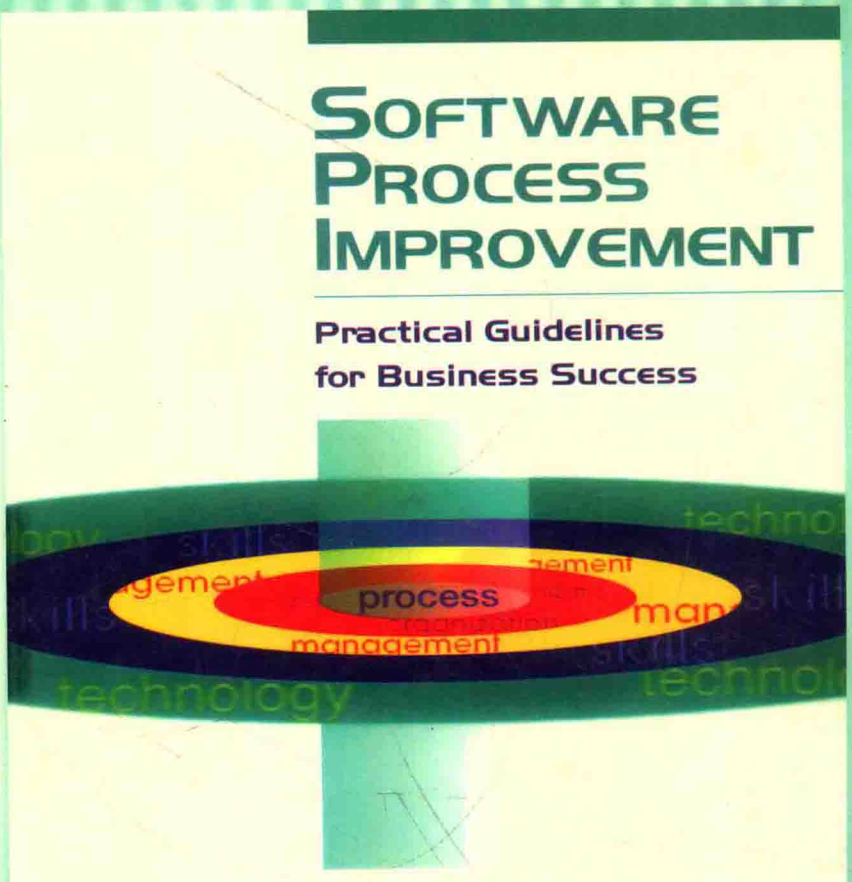
经 典 原 版 书 库

# 软件过程改进

（英文版）




Sami Zahran 著

机械工业出版社
China Machine Press

经 典 原 版 书 库

# 软件过程改进

（英文版）

# Software Process Improvement

## Practical Guidelines for Business Success

Sami Zahran 著

机械工业出版社
China Machine Press

Sami Zahran: Software Process Improvement: Practical Guidelines for Business Success (ISBN: 0-201-17782-X).



**本书版权登记号：图字：01-2003-1009**

**图书在版编目（CIP）数据**

软件过程改进（英文版）/扎罕（Zahran, S.）著. -北京：机械工业出版社，2003.4
（经典原版书库）
书名原文：Software Process Improvement: Practical Guidelines for Business Success
ISBN 7-111-11809-X

Ⅰ. 软… Ⅱ. 扎 Ⅲ. 软件开发 - 英文 Ⅳ. TP311.52

中国版本图书馆CIP数据核字（2003）第017228号

机械工业出版社（北京市西城区百万庄大街22号 邮政编码 100037）
责任编辑：杨海玲
北京瑞德印刷有限公司印刷·新华书店北京发行所发行
2003年4月第1版·2003年6月第2次印刷
787mm×1092mm 1/16·30.25印张
印数：3 001-5 000册
定价：49.00元

# 出版者的话

文艺复兴以降，源远流长的科学精神和逐步形成的学术规范，使西方国家在自然科学的各个领域取得了垄断性的优势；也正是这样的传统，使美国在信息技术发展的六十多年间名家辈出、独领风骚。在商业化的进程中，美国的产业界与教育界越来越紧密地结合，计算机学科中的许多泰山北斗同时身处科研和教学的最前线，由此而产生的经典科学著作，不仅擘划了研究的范畴，还揭橥了学术的源变，既遵循学术规范，又自有学者个性，其价值并不会因年月的流逝而减退。

近年，在全球信息化大潮的推动下，我国的计算机产业发展迅猛，对专业人才的需求日益迫切。这对计算机教育界和出版界都既是机遇，也是挑战；而专业教材的建设在教育战略上显得举足轻重。在我国信息技术发展时间较短、从业人员较少的现状下，美国等发达国家在其计算机科学发展的几十年间积淀的经典教材仍有许多值得借鉴之处。因此，引进一批国外优秀计算机教材将对我国计算机教育事业的发展起积极的推动作用，也是与世界接轨、建设真正的世界一流大学的必由之路。

机械工业出版社华章图文信息有限公司较早意识到“出版要为教育服务”。自1998年开始，华章公司就将工作重点放在了遴选、移译国外优秀教材上。经过几年的不懈努力，我们与Prentice Hall，Addison-Wesley，McGraw-Hill，Morgan Kaufmann等世界著名出版公司建立了良好的合作关系，从它们现有的数百种教材中甄选出Tanenbaum，Stroustrup，Kernighan，Jim Gray等大师名家的一批经典作品，以“计算机科学丛书”为总称出版，供读者学习、研究及庋藏。大理石纹理的封面，也正体现了这套丛书的品位和格调。

“计算机科学丛书”的出版工作得到了国内外学者的鼎力襄助，国内的专家不仅提供了中肯的选题指导，还不辞劳苦地担任了翻译和审校的工作；而原书的作者也相当关注其作品在中国的传播，有的还专诚为其书的中译本作序。迄今，“计算机科学丛书”已经出版了近百个品种，这些书籍在读者中树立了良好的口碑，并被许多高校采用为正式教材和参考书籍，为进一步推广与发展打下了坚实的基础。

随着学科建设的初步完善和教材改革的逐渐深化，教育界对国外计算机教材的需求和应用都步入一个新的阶段。为此，华章公司将加大引进教材的力度，在“华章教育”的总规划之下出版三个系列的计算机教材：除“计算机科学丛书”之外，对影印版的教材，则单独开辟出“经典原版书库”；同时，引进全美通行的教学辅导书“Schaum's Outlines”系列组成“全美经典学习指导系列”。为了保证这三套丛书的权威性，同时也为了更好地为学校和老师们服务，华章公司聘请了中国科学院、北京大学、清华大学、国防科技大学、复旦大学、上海交通大学、南京大学、浙江大学、中国科技大学、哈尔滨工业大学、西安交通大学、中国人民大学、北京航空航天大学、北京邮电大学、中山大学、解放军理工大学、郑州大学、湖北工学院、中国国

家信息安全测评认证中心等国内重点大学和科研机构在计算机的各个领域的著名学者组成“专家指导委员会”，为我们提供选题意见和出版监督。

这三套丛书是响应教育部提出的使用外版教材的号召，为国内高校的计算机及相关专业的教学度身订造的。其中许多教材均已为M. I. T.，Stanford，U.C. Berkeley，C. M. U. 等世界名牌大学所采用。不仅涵盖了程序设计、数据结构、操作系统、计算机体系结构、数据库、编译原理、软件工程、图形学、通信与网络、离散数学等国内大学计算机专业普遍开设的核心课程，而且各具特色——有的出自语言设计者之手、有的历经三十年而不衰、有的已被全世界的几百所高校采用。在这些圆熟通博的名师大作的指引之下，读者必将在计算机科学的宫殿中由登堂而入室。

权威的作者、经典的教材、一流的译者、严格的审校、精细的编辑，这些因素使我们的图书有了质量的保证，但我们的目标是尽善尽美，而反馈的意见正是我们达到这一终极目标的重要帮助。教材的出版只是我们的后续服务的起点。华章公司欢迎老师和读者对我们的工作提出建议或给予指正，我们的联系方法如下：

电子邮件：hzedu@hzbook.com
联系电话：（010）68995264
联系地址：北京市西城区百万庄南街1号
邮政编码：100037

# 专家指导委员会

（按姓氏笔画顺序）

| | | | | |
|---|---|---|---|---|
| 尤晋元 | 王　珊 | 冯博琴 | 史忠植 | 史美林 |
| 石教英 | 吕　建 | 孙玉芳 | 吴世忠 | 吴时霖 |
| 张立昂 | 李伟琴 | 李师贤 | 李建中 | 杨冬青 |
| 邵维忠 | 陆丽娜 | 陆鑫达 | 陈向群 | 周伯生 |
| 周克定 | 周傲英 | 孟小峰 | 岳丽华 | 范　明 |
| 郑国梁 | 施伯乐 | 钟玉琢 | 唐世渭 | 袁崇义 |
| 高传善 | 梅　宏 | 程　旭 | 程时端 | 谢希仁 |
| 裘宗燕 | 戴　葵 | | | |

*I dedicate this book to Watts Humphrey – the 'prophet' of the Software Process.*

*This book is a contribution towards the Software Engineering Institute's vision:*
*'Bringing engineering discipline to the development and maintenance of software'.*

# Foreword

## by Watts S. Humphrey

It is a pleasure to introduce this book by Sami Zahran. He covers an important topic in an interesting way. I enjoyed reading the book and I am sure you will as well. In addition to talking about process improvement, Sami provides useful guidance from a practitioner's perspective. He clearly explains the purposes and methods of process improvement and he compares the leading methods, their principal features and their characteristics. Most importantly, he discusses the issues that you, the user, will face as you pursue process improvement on your own.

As you read this book and think about what Sami says, I suggest you keep some special topics in mind. They should provide a useful perspective and help ensure that your process improvement efforts are most effective.

In the change business, there is something called unfreezing. People are naturally resistant to change and unfreezing breaks through their resistance. It shows the engineers and their managers what is wrong today. It makes them even more unhappy with the current situation, and convinces them that there are better and more effective ways to do their jobs. When the engineers and their managers see all the key problems in one big pile, they realize they really must do something about them.

A properly done assessment walks through the issues in great detail, breaks the ice, and starts unfreezing. Then you can begin to talk about what to improve. Until you get to that point, all improvement talk will be just that – talk. And nothing much will happen. This is why the assessment process was invented. By involving the engineers and managers, and asking them what is wrong and what can be improved, the assessment becomes a way of learning from the organization. When you get their ideas on what should be improved, the people are most likely to support and participate in the change process.

Another area of misunderstanding is the role of Capability Maturity Model (CMM)[SM] goals. In first developing the CMM, we did not want to block good ideas. The software community is full of creative people. Once they are involved, they will see many ways to improve things. Also, software engineering is a relatively new and rapidly changing field. What seems appropriate today could be hopelessly out of date in just a few years. A CMM that is too specific could easily inhibit change instead of encouraging it.

This is why the real CMM objective is the goals; everything else is examples of how the goals could be met. I thus suggest that you keep your eye on the goals, and use all the detail for guidance on possible ways to achieve them.

Another difficult issue is maturity ratings. These can be very helpful in focusing attention on immediate priorities. While they are an important means of communication, they can also cause the wrong behaviour. People can easily lose sight of the principal objective of improving the process. They begin to see the next CMM level as the target.

When people focus on the level, they think of assessment as a way to measure the level. Then they become concerned with the accuracy of the CMM as a measurement tool. Some even seek sophisticated measures that will precisely determine the maturity level. This is nonsense. Software processes are far too complex to measure with one or even a few numbers. The critical need is for a framework engineers and managers can use to examine and talk about their processes. Then they can use their detailed knowledge of the organization to identify the key problems, and to decide what and how to improve.

The level measure provides so much value that we decided to keep it. The need, however, is to focus on those few improvements that will make a difference right now. The CMM can guide you in identifying these opportunities. Other than that, don't worry about the maturity level. If you continue working on process improvements, the level will follow. If you focus on the level, however, improvement is much less likely.

To keep this perspective, I suggest you do the following:

1 Use maturity levels to do assessments and evaluations, and to set priorities
2 Keep your objectives focused on making specific improvements
3 Make improvement the job of every manager and track his or her performance against these goals
4 Plan to get the highest-priority key process areas (KPAs) in place as soon as possible
5 When those plans are well along, plan and implement the next most important KPAs.

If you do this, when you next do an assessment, you will see significant improvements. And the levels will take care of themselves.

Watts S. Humphrey
Sarasota, Florida, USA
April 21, 1997

# Foreword

## by Mark C. Paulk

Anyone familiar with computers is familiar, often painfully so, with the 'software crisis'. Our ability to build software-intensive systems is orders of magnitude greater today than it was five decades ago, but our appetite for software has grown even faster, and the software industry is still evolving from a craft to an engineering discipline. Historically, the result has been the chronic software crisis: software is (almost) always later than expected, more expensive than planned, and with less functionality than hoped. There is hope, however, that we have turned the corner on the software crisis.

If we are overcoming the software crisis, one of the major reasons is the topic of Dr Sami Zahran's book: software process improvement. Peter Freeman stated in the foreword to Watts Humphrey's *Managing the Software Process* that 'The "software crisis" is dead!' and that Humphrey's book was one of the best signs of that change.

Eight years later, the increasing ability of mature software organizations to deliver high-quality software products on budget and on schedule shows that Freeman was correct – at least for that part of the software community that has adopted a systematic approach to software process improvement.

Unfortunately only a minority of software organizations have chosen to pursue systematic improvement. The reasons are manifold, but perhaps the crux of the problem is that disciplined software engineering is easy to describe but devilishly hard to do.

Much of the problem lies in the fact that 'changing the way we do things around here' requires behavioural change across the board. True software process improvement requires that management, especially senior management, take an active role in process improvement. It also requires that the workers in the trenches participate in defining and implementing usable and

effective processes. This means a diversion from the 'real work' of shipping product. If software process improvement is considered a 'silver bullet' rather than an investment in the future, then it will wind up being another 'flavour of the month' fad, and its value will never be attained.

Improvement also implies facing a sometimes unpleasant reality. Some of the pain of the software crisis is caused by human nature. In response to the question 'Why does software cost so much?', Jerry Weinberg replies 'Compared to what?'. Tom DeMarco suggests that this assertion is a negotiating position; people complain because they know we work harder when they complain. In one survey, most of the responding professional software managers reported that their estimates were dismal, but they weren't on the whole dissatisfied with the estimating process! All too many software professionals would agree with DeMarco, but many software managers and customers are vitally interested in understanding how to manage software projects more effectively.

Customers and managers who use schedule pressure and overtime as motivational tools have to deal with the resulting quality trade-off. Customers and managers who are interested in truly managing software projects – and facing up to a sometimes unpleasant reality – have available a number of approaches for systematically improving the process for developing and maintaining software. The results of successfully applying these approaches give us hope that the software crisis is finally coming to an end.

Perhaps the best-known approaches to software process improvement are the International Organization for Standardization's ISO 9001 standard for quality management systems, the Software Engineering Institute's Capability Maturity Model for Software, and the ISO 15504 (frequently referred to as SPICE) standard for software process assessment. These approaches, among others, apply Total Quality Management principles to the software process and are described by Dr Zahran in this book. Hopefully the comments in this book will help the reader navigate the quagmire of alternative approaches!

As the product manager for the Software CMM, I have a biased view of the various approaches to software process improvement. We are incorporating a number of process implementation and management ideas from the various standards and models described in this book in the next version of the Software CMM. While I believe that the Software CMM is the best foundation for software process improvement, and we are actively working to maintain this position, any systematic approach to improvement can help an organization succeed. Regardless of the approach chosen, process improvement is becoming essential to survival in today's highly competitive world.

The importance of high-quality software products cannot be overemphasized. Recent UK court decisions and proposed changes to the US Uniform Commercial Code foreshadow a potential for legal action by dissatisfied customers. The concept that software should be sold free of major bugs and should work as intended, like other commercial goods, may be a major paradigm shift for many software developers!

To survive, much less thrive, modern organizations must continually improve all aspects of their business. Improvement in software-intensive products and services is crucial and difficult. The challenge is to implement good software engineering and management practices in the high-pressure environment software organizations face. A disciplined and systematic approach to software process and quality improvement, such as these models and standards support, is necessary to survive and thrive.

Process improvement is not, however, sufficient for success. Other issues are also fundamental, such as:

- building the right product – one that customers want to buy;
- hiring, selecting, and retaining competent staff;
- overcoming organizational barriers (for example, between systems engineering and software staff).

An effective software process improvement programme should be aligned with other organizational initiatives, perhaps under a Total Quality Management umbrella, to address the totality of business issues that are related to process improvement.

Regardless of the approach selected, building competitive advantage should be focused on improvement, not on achieving a score, whether the score is a maturity level, a certificate or a process profile. Dr Zahran's book should help the reader understand the trade-offs and issues associated with effective software process improvement.

Mark C. Paulk
Software Engineering Institute
Pittsburgh, Pennsylvania, USA
22 September, 1997

# Preface

## Software development is a challenging endeavour

Developing reliable software on time and within budget represents a difficult endeavour for many organizations. As the role of software becomes increasingly critical for business as well as for human lives, the problems caused by software products that are late or over budget, or that do not work, become magnified. Loss of life or widespread inconvenience caused by unreliable software makes big headlines in the news media. It is estimated that in the last few years around 4000 people have died as a result of software defects. In a modern aircraft, if software stops functioning for more than 200 milliseconds, the aircraft is irrecoverable. In June 1996 a European Space Agency rocket carrying a number of European satellites exploded seconds after its launch. The accident was attributed to software failure. A few years ago, unreliable software made big news in the UK, from emergency services disasters to social security payment blunders, let alone the failure of a large project for the London Stock Exchange. Improved software quality is essential to ensure reliable products and services, and to gain customer satisfaction. The US Government Accounting Office (GAO) reported recently on 'cost rising by millions of dollars, schedule delays of not months but years, and multi-billion dollar systems that do not perform as envisioned' (Paulk *et al.*, 1994).

## CASE tools are not enough

Stories about failure of software projects that still excite the press are in sharp contrast to the inflated promises of CASE tools that filled the same press back in the mid and late 1980s. The industry has realized that tools are not enough.

One fact that the software industry has established is that 'a fool with a tool is still a fool!'. Usually business solutions have three main aspects: people, process, and technology. It is evident from industry experience that, when implementing a business solution or introducing a change, the least problematic aspect is usually technology, while processes and people are the critical factors that could make the difference between success and failure. People are an integral part of the process, since they are the enablers of the process activities, process monitoring and process management.

## Competent individuals are not enough

The software industry's experience with CASE tools has proved that the main reason for failing software projects has little to do with technology and tools, and much to do with lack of process discipline. Software development is a team effort. In the absence of process discipline, a team may follow different processes, or more commonly use no defined process at all. In such a case it will be 'like a ball team with some team members playing soccer, some baseball, and others football. Under these conditions, even the best individual players will form a poor team' (Humphrey, 1995). In contrast, a team that follows consistent process definitions can better coordinate the work of individual members, direct the efforts of the team members towards the common goal and more precisely track the progress.

## Process focus offers better chances for success

Software development as a discipline has existed for more than four decades, but we have not yet turned the software industry into an engineering discipline. The recent focus on the software development process is a step in the right direction. Only by creating a disciplined process for software development can we manage and control the quality of software products. Organizations are realizing that their fundamental problem is the immaturity of their software development process. All the evidence is that investing in software process improvement promises to offer better hope for the software industry, as it has done for other industries such as manufacturing. Also, there is a difference in motivation between a software movement based on tools and one based on process improvement. Process improvement is the responsibility of the organization developing the software and there are no tool vendors with vested interest. The use of tools to automate a chaotic process will lead to more (automated) chaos. Examples are abundant, but one striking example outside the software industry is the shipping of sophisticated armoury and destructive weapons to a chaotic war between two tribal factions in a primitive country. These 'technological tools' are not likely to result in stability, but will probably increase human suffering. Experience has shown that introducing new technology and tools in an immature or undisciplined environment is likely to increase the chaos. A software project without defined processes for control and management (for example quality assurance, configuration management

and project management) will not benefit from tools. Dumping tools into such a project is likely to increase the chaos, to speed up the production of faulty software and to multiply user dissatisfaction. Such projects could ultimately end up as a disappointment to all concerned. Such disappointments take various forms, such as wasted effort, time, money and resources, and possibly unavoidable disasters.

## Software process movement: the second wave in the software industry

Structured methods were developed in the 1970s to cater for the increasing demands and complexity of software, and consequently the increasing size of development teams. That was the first wave of the software industry. It came as a response to the growing need to build complex interactive commercial applications using shared systems and to make such systems maintainable. Structured methods focus on ways to formalize the definition of requirements and on the traceability of requirements through design and build into finished systems. Some of these transformations have been assisted to varying degrees by automated tools. Although this was the beginning of transforming software development from a 'cottage industry' to mass production, it was not quite enough. Real issues that make or break software projects, such as project management and requirements management, were not a mainstream focus. The software process movement came as a response to the increasing rate of failure of software projects. Focus on process started through sponsorship by the US Department of Defense (DoD) which funded the Software Engineering Institute (SEI) to come up with a method for assessing the capability of the Deparment's software subcontractors. Watts Humphrey first joined the SEI in an undefined position and in a couple of months was named Director of the Process Program. Since that time the process message coming out of the SEI has gone from strength to strength to influence the whole software industry worldwide.

One can easily trace the roots of the software process to the quality movement that started in the 1930s and prevailed throughout the 1970s, 1980s and 1990s. The concepts of quality gurus such as Edwards Deming and Philip Crosby gained popularity across manufacturing industry all over the world. Watts Humphrey applied those same quality principles to software development. The process maturity movement prepares the way for the third wave of the software industry: 'software industrialization'. In the third wave software development will become like an assembly and manufacturing process. Enabling technologies for the third wave include object-oriented technology and reusable component libraries. It will then be possible to assemble software from standard reusable components. A critical enabling factor for the third wave is a disciplined software engineering process with predictable quality, schedule and functionality.

## Aims of this book

This book offers a pragmatic approach to the effective implementation of software process improvement. It provides guidelines for creating process support infrastructure, and makes the case for adopting a process view to software development. It outlines a practical approach for setting up a disciplined and continuously improving software process environment. In summary, the book presents a framework for establishing an effective environment for continuous software process improvement.

The arguments in the book put emphasis on the people aspect. Understanding and following the process is as important as the process definition. Another equally important emphasis is on the impact of process discipline on team performance and business goals and objectives.

The concepts offered in this book are the result of more than thirty years of the author's practical work in the software industry. Most of those years were spent in practical experience and research on different facets of software engineering. This included experiencing the pains and pleasures associated with developing software and managing software projects. The projects covered business and industrial applications for a variety of industry sectors ranging from oil, banking and government, to defence, manufacturing and aerospace. They also included involvement in the development of operating systems, database management systems, data dictionary systems, transaction processing systems and a large number of commercial applications. Having lived through both successes and failures of software projects, I readily identified with the process message and teachings which I first received at the Software Engineering Institute, Carnegie Mellon University, in February 1992. Since that time my dedication to the software process has been uninterrupted.

## Intended audience

This book is relevant to and readable by a wide audience, including those who already have some knowledge of software process assessment and improvement and those who have little knowledge beyond knowing that the subject is significant for them. In other words it contains new ideas and approaches that will interest those who have prior knowledge, and is simple and readable enough to interest those who do not. In particular, it is aimed at the following special interest groups.

*Software engineering managers and professionals*
The whole book should be of interest to everyone involved in software engineering activities including management, coordination, development and control. This includes business managers with interest in software, project managers, team leaders, software engineers, and software support functions, such as configuration management, quality assurance and process improvement teams. Also the book is suitable for inclusion in graduate software engineering degrees within a unit on software process improvement.

*Process improvement teams*
The first part of the book discusses process thinking in generic terms. It should be beneficial to all those interested in process improvement activities including business process re-engineering, business process redesign and business process improvement.

*Process research scientists*
The book offers a holistic approach to a process improvement environment. The concepts and discussions in the book are intended to provide inspiration for further research effort on process modelling and quality concepts.

## Structure of the book

The book is structured in five parts, followed by a glossary and list of references.

*Part 1: Process thinking*
This part lays the intellectual foundation for the rest of the book. It defines and explains process thinking, relates the concepts discussed to process discipline, and describes the characteristics of an effective process environment. It also relates these concepts to the software process environment. Part 1 contains four chapters:

Chapter 1. Process thinking
Chapter 2. Process discipline
Chapter 3. Effective process environment
Chapter 4. Process maturity: the second wave of the software industry

*Part 2: A framework for software process improvement*
This part describes the framework proposed for the software process environment. It describes the components of the framework, the process infrastructure, process improvement roadmaps, process assessment methods, and process improvement plans. This framework should lead to a continuous process improvement environment. Part 2 contains five chapters:

Chapter 5. A framework for software process improvement
Chapter 6. Software process infrastructure
Chapter 7. Process improvement roadmaps
Chapter 8. Fundamentals of software process assessment
Chapter 9. Software process improvement action plan

*Part 3: Making software process improvement happen*
Part 3 describes strategies and plans for planning and launching a software process improvement programme in your organization. It discusses steps for converting the assessment results into an improvement plan and highlights the need for measuring the benefits of software process improvement. It discusses