SEI软件工程丛书·影印版

软件过程管理

Managing the Software Process

瓦茨·S·汉弗莱 [Watts S. Humphrey] 著

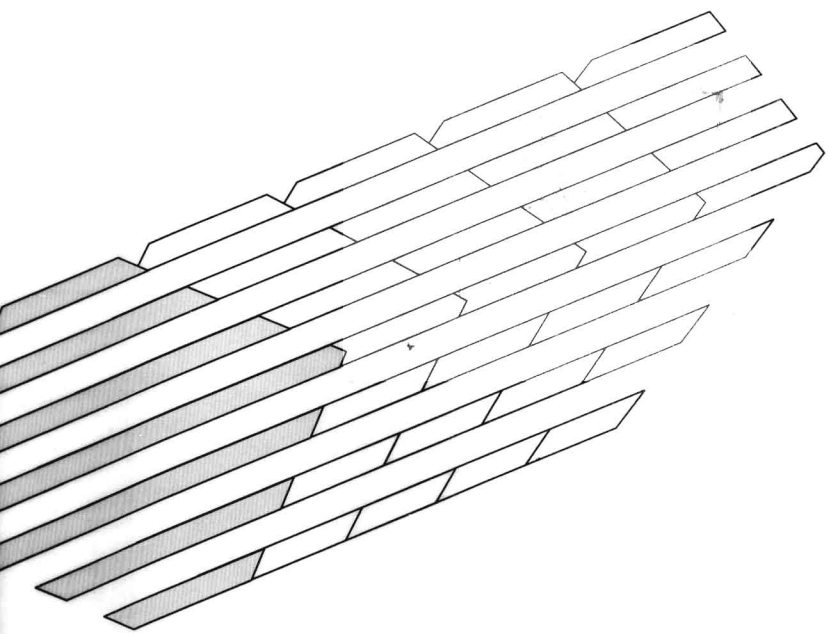
清华大学出版社

SEI软件工程丛书·影印版

软件过程管理

Managing the Software Process

瓦茨·S·汉弗莱 [Watts S. Humphrey] 著



清华大学出版社

（京）新登字 158 号

内 容 简 介

本书是软件工程发展史上里程碑性的著作，是学习软件工程的必读书目。内容覆盖软件过程管理和改进的 5 个不同侧面。编排的顺序和书中阐述的软件过程改进模型一致，该模型已经为一大批领先的软件组织提供了适用的过程改进框架。

本书读者对象为学习和研究软件的所有人员，并可作为大学相关课程教材使用。

北京市版权局著作权合同登记号：图字 01-2002-3161 号

书　　名：软件过程管理
作　　者：[美] 瓦茨·S·汉弗莱
责任编辑：尤晓东
出 版 者：清华大学出版社（北京清华大学学研大厦，邮编 100084）
　　http://www.tup.tsinghua.edu.cn
印 刷 者：世界知识印刷厂
发 行 者：新华书店总店北京发行所
开　　本：787×960 1/16　　印张：33.25　　插页：1
版　　次：2002 年 8 月第 1 版　2002 年 8 月第 1 次印刷
书　　号：ISBN 7-302-05758-3/TP·3405
印　　数：0001～5000
定　　价：59.00 元
本书仅限在中华人民共和国大陆地区发行销售。

出 版 说 明

1984 年，美国国防部出资在卡内基·梅隆大学设立软件工程研究所(Software Engineering Institute，简称 SEI)。SEI 于 1986 年开始研究软件过程能力成熟度模型(Capacity Maturity Model，CMM)，1991 年正式推出了 CMM 1.0 版，1993 年推出 CMM 1.1 版。此后，SEI 还完成了能力成熟度模型集成(Capability Maturity Model Integration，简称 CMMI)。目前，CMM 2.0 版已经推出。

CMM 自问世以来备受关注，在一些发达国家和地区得到了广泛应用，成为衡量软件公司软件开发管理水平的重要参考因素，并成为软件过程改进的事实标准。CMM 目前代表着软件发展的一种思路，一种提高软件过程能力的途径。它为软件行业的发展提供了一个良好的框架，是软件过程能力提高的有用工具。

SEI 十几年的研究过程和成果，都浓缩在由 SEI 参与研究工作的资深专家亲自撰写的 SEI 软件工程丛书（SEI Series In Software Engineering）中。

为增强我国软件企业的竞争力，提高国产软件的水平，经三联四方工作室和清华大学出版社共同策划，全面引进了这套丛书，分批影印和翻译出版，这套丛书采取开放式出版，不断改进，不断出版，旨在满足国内软件界人士学习原版软件工程高级教程的愿望。

前　言

如果你不知道要去哪里，任何一条路都可以走。

——中国谚语

如果你不知道身在何处，任何地图都与事无补。

——瓦茨 S. 汉弗莱

软件工程学的发展为软件开发带来收益的同时，在许多方面仍然不尽人意。面对软件开发领域的巨大挑战，我们仍然举步维艰，许多软件开发过程常常跌入可怕的深渊。从全国来说，软件开发的业绩亟待改进，分清轻重缓急，制定一个改进计划也迫在眉睫。改进的规划要求我们认清现状，明确目标，这些都是本书的主题。

确定软件过程的现状看起来是一件微不足道的小事，其实不然。准确定义软件过程现状，要求我们有一套评价标准、一个测量框架，还对其他许多工作提出了要求。认识问题是正确解决问题的前提。衡量软件组织能力的方法之一是观察该组织在危急时刻的表现，而危机时刻往往就是最需要优秀的实践方式的时候，也是软件人员最缺乏指导的时候。本书将帮助读者回答如下问题：

1. 我们当前的软件过程处于什么样的水平上？

2. 必须做什么才能改进过程？

3. 从哪里开始？

本书取材于卡内基·梅隆大学软件工程研究所一个美国空军项目的研究成果。这项研究的目的是为军方选择合格的软件供应商提供指导。事实证明，评价软件供应商优劣的方法同样适用于对其他软件组织的评估。本书阐述评估中发现的对于改进软件过程十分重要的技术和管理话题。书中的各个话题都和软件过程管理的基本原则相关。本书的目的在于为读者提供一

个软件开发过程评估和改进的框架及相关技术，而非单纯地提供一套特定的解决方案，这与学习阅读技能和学习阅读一个故事之间的区别相似。软件领域是一个新的领域，很多新的工具和方法将会相继产生。本书阐述的技术，以几个世纪以来，一直推动科学和工程技术进步的持续有效的原则为基础。这些原则为学习和改进软件工程过程构筑了一个强大的知识框架。

个人、团队和群体

软件开发的历史是一个规模不断扩大的历史。初期少数几个人可以手工编写一些小程序，但这种生产能力很快就难以满足发展的要求。由一、两打专业人员组成的团队应运而生，但软件开发的成功却变得含糊了。当很多机构终于解决了小型系统的问题时，软件开发的规模继续在扩大。今天，大型的项目需要很多团队协同工作。软件开发的复杂性已经超出了我们靠直觉及时解决问题的能力，急需的是更加结构化的软件过程管理方法。

我们发现过程改进对策既适用于大型项目，也适用于较小的项目。事实上稍作简化，这套方法对个人编程也十分有益。这很关键，如果不能为独立的专业人士服务，它就失去了生命力。

人才和软件过程

在任何软件组织中，人才都是最重要的因素。找到优秀的人才是至关紧要的第一步。软件开发人员水平越高、经验越丰富，开发出一流产品的可能性越大。

我们竭尽所能找到优秀人才之后，下一步该做什么呢？如果每个人都使用不同的编程语言，使用不同的约定，或者更改设计和编码时都不与伙伴协调，结果只能是一团糟。成功的软件组织已经认识到，即使最优秀的专家也需要一个结构化、制度化的环境，才能完成合作任务。

在没有建立这些规则的软件组织中，员工经常无休止地重复处理技术细节问题。也许挑战性的工作正等待处理，可是人

们已经被堆积如山的失控的细节问题压得无暇顾及。除非这些细节问题得到严格的管理，否则最优秀的人也无法有所作为。一流的人才是基础，一流的人才需要有序的过程支持才能完成一流的工作。

超级程序员的神话

人们通常认为，几个软件高手远比典型的软件团队工作更有成效。这就是说，这些高手凭直觉就知道如何完成一流的工作，任何有序的过程框架都是多余的。果真如此的话，拥有优秀人才的软件组织将不会遭遇软件质量和生产能力方面的常见问题。但事实并非如此，一些领先的软件组织一直在最好的计算机院校招聘高材生，这些组织的员工都是当时最好的人才，但是他们的软件开发组面临着许多和其他软件组织同样的问题。“超级程序员方法”需要的人才在社会上和顶尖大学里也是找不到的。很显然，这种靠超级高手就能解决软件问题的想法，充其量只能是一个理论上的解决方案，没有任何现实意义。吸引优秀人才很重要，通过对软件过程的有效管理为优秀人才提供支持也必不可少。

过程和技术

另一个神话是一种广泛传播的观点，认为某些技术先进的工具和方法可以魔术般地消除软件危机。这种观点不仅错误，而且危险。长远的改进离不开技术，但是不加思考地依赖虚无飘渺的“银弹”，会将人们的注意力从追求更好的过程管理上错误地转移开。

当被问及面临的关键问题时，部分软件专业人士仍然会提到技术。他们的问题主要涉及到需求的可修改性、变更失控、进度表武断、测试时间不足、缺乏培训、以及未经管理的系统标准等。他们还会提到机时有限、终端不足、或者工具质量低劣等问题，但是这些问题更多的是关于管理的而非单纯的高技术问题。

以下几方面的因素制约了软件技术的高效运用：过程定义

失误、技术实施不协调、以及过程管理的缺乏等。在这些问题得到充分的重视之前，软件技术不可能充分发挥作用。

本书主要阐述软件过程管理。软件过程是将用户的需求转化为有效的软件解决方案的一系列活动，许多软件组织无法正确定义和控制这一过程，这是组织改进的巨大潜力所在，也是本书关注的焦点。

本书内容简介

本书共分 5 个部分，分别涵盖软件过程改进的 5 个不同的方面。编排的顺序和书中阐述的软件过程改进模型一致，这一模型已经为一批领先的软件组织提供了适用的过程改进框架。每个部分中的章节和相应改进阶段的关键因素对应。根据第 1 章中提出的成熟度框架，软件组织可以确定自身目前所处的阶段，从而明确其改进活动应从何处开始，然后根据后面相关章节内容确定过程改进的步骤。例如处于 1 级的组织，应当重点关注第 II 部分的内容；处于 2 级的组织，应当重视本书第 III 部分的内容。

值得注意的是，很多软件问题是相互关联的，一些方面的内容几乎贯穿软件活动的所有阶段。过程管理的关键问题是优先顺序，不仅要回答改进应该关注哪些问题，还要搞清楚谁先谁后。本书的逻辑结构也是按照管理的优先次序编排的，例如，在第 III 部分强调标准，并不代表标准问题在已定义过程阶段之前或之后的阶段中不重要，而是表明在第 III 部分，标准需要优先得到管理者的关注。

在本书中，每个部分的开头都有一个简洁的概述，总结了该部分的主要话题，以及这些话题在本阶段的过程改进中的重要性。本书的 5 个部分分别是：

第 I 部分　软件过程成熟度。描述了软件过程管理的框架及其在软件过程评估中的作用，阐述了开始有效过程改进的步骤。

第 II 部分　可重复的过程。概述了建立基本的软件过程控

制所需要的活动，这些活动可以保证过程持续有序改进所需的稳定性。

第Ⅲ部分　已定义的过程。阐述如何确定开发过程，以及控制过程所需明确的技术和管理方面的概念。软件开发是一个难以预料的过程，一个已定义的过程有助于有序地应付突发事件。

第Ⅳ部分　已管理的过程。导入了软件过程数量控制，在本阶段，要求收集和分析数据，以便支持质量和过程的数量管理。数量管理是指导我们准确把握工作任务的工具，是准确控制我们活动的方法，是持续改进的正确基础。

第Ⅴ部分　优化的过程。代表了软件过程演化的最高级别，将关注的焦点从解决问题转移到预防问题上。在这一水平上，软件经理和专家学会利用数量化的过程方法，发挥技术作用，达到质量和生产能力的持续改进。

建议本书阅读方式

本书的编排适合按从头至尾的顺序阅读。部分读者希望先概要了解本书的内容，可以先阅读第1章、第19章和第20章，然后再读其他各章后面的小结，这样可以对当前的软件问题和未来的过程目标心中有数。通常情况下，最好能够配合实践中软件过程改进的进程，从头开始研读本书。

……

瓦茨·S·汉弗莱

Managing the Software Process

Watts S. Humphrey
SOFTWARE ENGINEERING INSTITUTE

ADDISON-WESLEY

Boston • San Francisco • New York • Toronto • Montreal
London • Munich • Paris • Madrid
Capetown • Sidney • Tokyo • Singapore • Mexico City

Software Engineering Institute

The SEI Series in Software Engineering

Library of Congress Cataloging-in-Publication Data
Humphrey, Watts S., 1927–
Managing the software process / by Watts S. Humphrey.
p. cm.
Includes bibliographical references and index.
ISBN 0-201-18095-2
1. Software engineering—Management. I. Title.
QA76.758.H86 1989
005.1'068—dc19 88–34453

Reprinted with corrections August 1990

Text printed on recycled and acid-free paper.

ISBN 0201180952

28 2930313233 CRW 05 04 03 02

28th Printing March 2002

Software Engineering Institute

The SEI Series in Software Engineering

TO MY CHILDREN

Katharine, Lisa, Sarah, Watts Jr., Peter,
Erica, and Christopher

FOREWORD

The "software crisis" is dead!

"When did it die?" you might ask. Many may not have noticed, and, indeed, we cannot pinpoint the exact moment of demise; nonetheless, it is clear that things are different today than they were a few years ago in the realm of software development.

This book is one of the best signs of that change that I have seen.

Although for two decades we have heard about, read about, and lived the "software crisis" in the popular sense of a set of terrible conditions that beset us, I am using the term "crisis" here more in its preferred dictionary sense of a decisive point or turning point. While many of those conditions (missed budgets and schedules, poor quality, unreasonable expectations) are still with us, we have changed from feeling that software is some kind of totally unmanageable beast to believing that under the right conditions we can manage it just as we have learned to manage other problematical situations in our universe.

Watts Humphrey's considerable experience as a software manager, and now as a student of the development process, has given him the insight to understand that building software is not just a monolithic process, always the same. As the central theme of this book so clearly indicates, we can classify those differences in what we actually observe in development organizations in a way that permits us to see that there is a sequence of maturity levels in the software process.

Understanding that there are different stages of maturity and understanding something of the conditions that determine where one is and where one can hope to be is often the key to growth—to turning the corner from chaotic software development to a more controlled and manageable process. That is the theme of this

highly readable, well-grounded, and pragmatic book. It will help you move beyond the turning point (or crisis) of feeling overwhelmed by the task of managing the software process to understanding what is essential in software management and what you can do about it.

Long live the crisis!

Peter Freeman

LAGUNA BEACH, CALIFORNIA

PREFACE

If you don't know where you're going, any road will do.
CHINESE PROVERB

If you don't know where you are, a map won't help.
WATTS S. HUMPHREY

Software engineering can be both rewarding and disappointing. The intellectual challenge of software is unsurpassed, but our business performance has all too often been abysmal. There is an urgent national need to improve this performance, and to do so we need an improvement plan and a set of priorities. Such planning calls for a vision of our goals and a clear understanding of where we are. These are the themes of this book.

While it may seem trivial to define the state of our current software process, it is not. The definition task requires an evaluation standard, a measurement framework, and much work. To intelligently attack our problems, we must know what they are. One way to measure the capability of a software organization is to observe what it does in a crisis. That is when good practices are most important, and that is when software people often have the least guidance. This book will help you deal with the following questions:

1. How good is my current software process?
2. What must I do to improve it?
3. Where do I start?

This book grew out of work at the Software Engineering Institute at Carnegie Mellon University on a U.S. Air Force project. The objective was to provide guidance to the military services in selecting capable software contractors.[1] The resulting method for evaluating their strengths and weaknesses has proved valuable for assessing other software organizations. This book describes the techni-

[1]Humphrey, W. S., and W. L. Sweet. "A method for assessing the software engineering capability of contractors," SEI Technical Report SEI-87-TR-23, September 1987.

cal and managerial topics these assessments have found most critical for improvement.

The book's individual topics are presented in relation to the basic principles of software process management. The approach is to provide a framework and some techniques for evaluating and improving the process of doing software rather than presenting a specific set of solutions. It is like the distinction between learning a story and learning to read. The software field is so new that many new tools and methods will surely be developed. The techniques outlined in these pages, however, are grounded in the durable principles that have fueled several centuries of scientific and engineering advancement. These principles provide a powerful conceptual framework for learning about and improving the software engineering process.

Individuals, Teams, and Armies

The history of software development is one of increasing scale. Initially a few individuals could hand craft small programs; the work soon grew beyond them. Teams of one or two dozen professionals were then used, but success was mixed. While many organizations have solved these small-system problems, the scale of our work continues to grow. Today large projects require the coordinated work of many teams. Complexity is outpacing our ability to solve problems intuitively as they appear. What is required is a more structured approach to software process management.

In addition to working with very large projects, however, we have found that the same methods are effective for smaller groups. In fact, in simplified form, they are even helpful for individual programmers. That, of course, is the key. If our methods do not serve the individual professionals, they will not endure.

People and the Software Process

Talented people are the most important element in any software organization. The crucial initial step is thus to get the best people available. The better and more experienced they are, the better the chance of producing first-class results.[2]

Once you have the best people you can get, however, what next? If everyone wrote in different programming languages, used special conventions, or didn't coordinate their design and code changes with their peers, the results would be chaos. Successful software organizations have learned that even the best professionals need a structured and disciplined environment in which to do cooperative work.

Software organizations that do not establish these disciplines condemn their

[2]Boehm, B. W. *Software Engineering Economics*. Englewood Cliffs, NJ: Prentice-Hall, 1981.