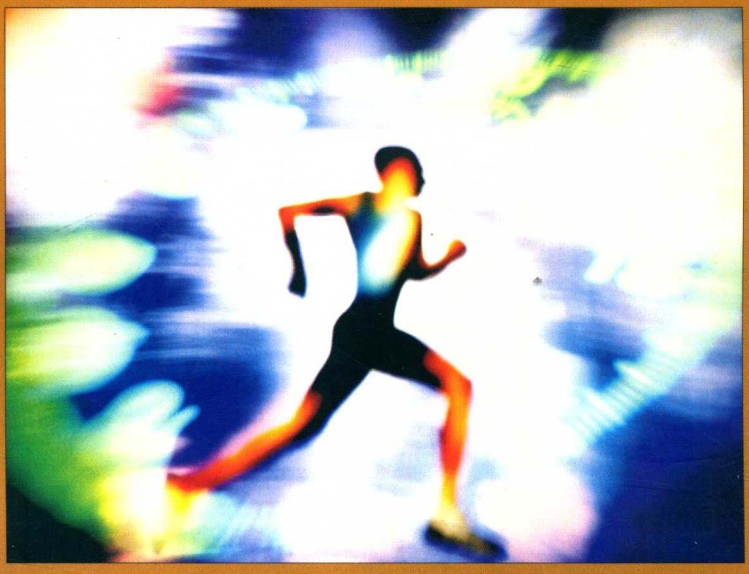
SEI SERIES IN SOFTWARE ENGINEERING

软件工程经典系列

PSP[SM]软件工程师的自我改进过程（英文版）

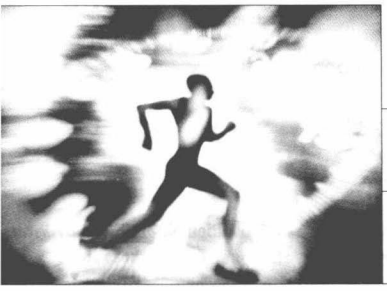
PSP[SM]: A Self-Improvement Process for Software Engineers



〔美〕Watts S. Humphrey 著

人民邮电出版社
POSTS & TELECOM PRESS

软件工程经典系列



PSP[SM]

软件工程师的自我改进过程（英文版）

PSP[SM]: A Self-Improvement Process for Software Engineers

〔美〕Watts S. Humphrey 著

人民邮电出版社

图书在版编目（CIP）数据

PSP 软件工程师的自我改进过程. 英文/（美）汉弗莱（Humphrey, W. S.）著. —北京：人民邮电出版社，2006.4
（软件工程经典系列）
ISBN 7-115-14597-0

Ⅰ. P… Ⅱ. 汉… Ⅲ. 软件工程—英文 Ⅳ. TP311.5

中国版本图书馆 CIP 数据核字（2006）第 019069 号

软件工程经典系列

PSP[SM] 软件工程师的自我改进过程（英文版）

◆ 著 [美] Watts S. Humphrey
责任编辑 俞 彬

◆ 人民邮电出版社出版发行 北京市崇文区夕照寺街 14 号
邮编 100061 电子函件 315@ptpress.com.cn
网址 http://www.ptpress.com.cn
北京铭成印刷有限公司印刷
新华书店总店北京发行所经销

◆ 开本：700×1000 1/16
印张：23
字数：415 千字 2006 年 4 月第 1 版
印数：1 – 4 000 册 2006 年 4 月北京第 1 次印刷

著作权合同登记 图字：01-2006-1567 号
ISBN 7-115-14597-0/TP・5297

定价：42.00 元

读者服务热线：(010)67132687 印装质量热线：(010)67129223

内容提要

随着软件工程专业的发展和成熟，个体软件过程（PSP）得到了广泛认可和应用实践。本书系统描述了个体软件过程（PSP），并且解释了按可预测的进度交付优良产品的实践和方法。读者将会了解一个规范的软件工程过程所包括的具体内容。

本书通过 14 章分步介绍了 PSP 方法。第 1 章描述了 PSP 整体原理及介绍策略。第 2 章和第 3 章解释了如何遵循一个定义的过程和怎样收集和使用用于管理程序开发作业的数据。第 4 至第 7 章介绍了估算和计划。第 8 章至第 12 章阐述了质量管理和设计。第 13 章描述了如何将 PSP 方法用于各种工作。第 14 章描述如何将 PSP 方法用于 TSP 过程，以及 TSP 指导经过 PSP 训练的软件工程师怎样在一个项目中使用这些方法。

本书实用性与可读性较强，可作为高等学校计算机软件工程课程的教材，也可作为工程技术人员自学个体软件过程的教材，同时也是进行软件过程改善和能力成熟度模型 SW-CMM 评估的重要参考资料。此书作为 SEI 的 PSP Body of Knowledge（BOK）的学习指导教材，为掌握 PSP 知识体提供必要的帮助。本书还可供软件过程改进人员、软件开发项目经理、程序员阅读参考。

IN MEMORY OF DR. JOHN E. YARNELLE

an inspiring teacher

前　言

大多数开发组的记录是不尽如人意的，而软件组的记录尤其差。Standish 组报告，一半以上的软件项目严重拖期且超过预算，而且近四分之一的项目没有完成就被取消了。[1]只有低于 30%的项目是成功的。据我了解，大多数的软件开发人员非常清楚这些问题，甚至可以解释它们的原因，如：不切实际的进度，不充分的资源和不稳定的需求。尽管这些问题是广为人知的，也不难解决，可是很少有开发人员知道如何解决它们。

常见的推辞是面对自己的困难来责备其他人，而一副受害者的态度是不能解决问题的。当以适当的方式着手解决这些问题时，通常可以解决它们。然而，这大概需要一些你从来没有学过的技能和实践，也需要用管理的观点来处理管理问题。我们可以用个体软件过程（PSP）[2]获得需要的实践。本书描述了 PSP，并且解释了按可预测的进度交付优良产品的实践和方法。当学习了这些技能后，就可以成为一个有资格的团队软件过程（TSP）的小组的一员。

这样的小组称为“自指导”小组，因为自己能够定义工作实践并能

---

[1] The Standish Group International, Inc., 586 Olde King's Highway, Dennis, MA 02638; www.shandishgroup.com

[2] 个体软件过程（PSP）和小组软件过程（PSP）是卡内基梅隆大学的注册标记。

够与管理人员沟通他们的计划和进度。本书的最后一章描述了 TSP 及它是如何使你管理个人的工作的。

作为一名软件工程师

作为一个工程师应该知道怎样一致地且可预测地完成优良的工作。在美国许多州有管理工程实践的规章，那些自称工程师的人必须展示他们的专业能力，否则不被承认。大多数工程领域的建立，最初是因为公众要求防止不合格的操作，特别在建筑结构、蒸汽发电厂等领域。虽然颁发执照制度没有奇迹般地解决所有这些问题，但还是起到了很大作用。

有执照的工程师使用已知的且证明了的方法，经过测试来确保他们能够自始至终完成优良的工作。他们需要展示出来他们在生产安全产品方面的能力。有执照的工程师与其他技术工人间的差别就是工程师知道用适当的方法来执行他们的工作，这是法律的需要，而不管管理、客户或其他压力。

如果我们称我们自己为工程师，我们必须学会在可预测的进度下生产优良的产品。这需要我们学会如何始终满足我们的承诺，并且我们知道如何处理日常创造性的开发工作的挑战。软件开发是具有挑战的专业化职业，我们必须自始至终使用最好的、有用的方法来满足我们管理方面和客户的需要。

质量问题

缺乏质量管理引发许多现今的软件问题。大多数的软件专业人员在开发和最终测试期间花费近一半的时间来测试并修复他们的产品。不良质量也会导致进度问题，有缺陷的产品交付期会较长。尽管修复一些缺陷看起来好像较麻烦，可是连一些小的程序都会有上百个缺陷的话，发现和修复它们则需要花几周甚至几个月。软件质量始于开发人员本身，如果任何程序模块都有大量的缺陷的话，就很难去测试，也会花很多时间将他们集成到较大的系统中，并且还会给用户带来许多麻烦。

在编写一些小程序时，我们中间的大多数人都有很高的生产率。然而，当我们开发较大程序时，我们的生产率急剧下降。虽然开发较大的系统还涉及到一些额外的构架和设计工作，大多数增加的工作量是由缺陷引起的。随着程序的增大，平均花在发现和修复每个缺陷上的时间以指数倍数增加。然而，如果我们能够自始至终地编写高质量的模块化的程序，我们就会生产出更好的产品并且提高我们的生产率及组织的生产率。

一个有规范的软件工程过程包括有效地缺陷管理、全面的计划、精确的项目跟踪和报告。本书指出作为一个个体或一个 TSP 组成员如何使用这些规范来执行更好的开发工作。它还说明了如果想要管理自己的工作这些实践就是基础。

软件工程师的收益

在我们的生活中越来越依赖软件，安全、保密及质量要求不断增加。这意味着要求有能力的软件专业人员也随之增加。不幸的是，没有任何方法可以将软件开发人员与许多草草编出劣质代码的程序员区分开。经过了 PSP 训练，你可以向软件工程研究所申请一个软件专业人员正式的资格证书。这是唯一的资格，这是你与其他开发人员的区别。PSP 训练也使你有资格参与到一个 TSP 团队，为了达到可预测的成本和承诺的进度的目的，PSP 证书是确保你成为有能力生产高质量软件的潜在的雇员。还有一些 PSP 证书的好处，它增加了作为一个有技能的软件专业人员的认可度，同时容易获得更多信任和高薪的岗位。目前具有这样资格的开发人员可以获得更广泛选择机会，而且将来这样的需要将会更多。

哪些人应该学习 PSP

现代技术工作涉及到许多专业，而且参与到开发现代产品和系统的人员来自广泛的学科。为了在可预测的进度下生产出优良的产品，这些人们的所有工作必须要计划、管理及质量控制。这意味着与系统开发相关的每个人必须知道如何执行有规范的工程工作，这也意味着做这个工

作的任何一个人都会从学习 PSP 中获益。

虽然在这本书中列举的例子和练习是针对开发小程序的，正是因为即使是小程序，软件开发也会有丰富的可以度量和分析的过程。这使软件过程特别适合讲授有规范的工程实践。几乎任何技术领域中的大多数现代的专业人员在他们受教育期间都学过写程序，因此 PSP 课程适合几乎任何一个从事计划工程或技术生涯的人，而且特别适合任何一个计划在产品和系统开发中工作的人们。

本书采用的方法

随着软件及软件产品的重要性的提高，组织对软件工程师的需要也随之增加，要求这些工程师要坚持使用规范的个人实践。为了满足这些需要，必须在我们写的每个程序中学习和坚持实践这些规范。如果我们在编写模块程序时不使用合理的开发实践，我们就几乎没有机会在写大程序时使用这些实践。

当学生开始编写程序时，通常从学习编程语言开始。他们学着解决很初级的问题，提高个人处理初级问题的技能。随着课程的增多，完善了个人的技能，并且能较快地开发模块级别较大的程序了。然而，开发小程序的技能本身就受到限制，尽管他们能够足以胜任小模块的个人任务，可是很难提供足够的基础来解决在一个项目组内的大规模、多人多任务问题。

本书遵循着一个根本上就不同的策略而展开。它按比例调整工业界软件实践来满足模块规模程序开发的需要，然后按照一系列软件过程提供对大级别模块软件开发的合适的基础指导。通过做练习并使用本书介绍的方法，可以学会如何在自身实践中使用这些方法。一旦你掌握了这些技能并且在模块级别程序开发上应用了这些实践，就可以使用这些技能来开发大项目。虽然我们在开发大程序时还需要另外的需求、设计、实现及测试方法，但是本书中讲授一些基本的软件工程规范可以直接应用到大规模系统开发。这就是为什么一个大系统能够通过对程序模块的集成而建立起来的原因，而这些模块很像你在 PSP 课程中开发的程序。

本书的主要目标是指导软件开发人员培养个人软件工程技能，以便更出色地完成任务。我们要考虑到改进个人性能会遇到挑战，例如，在运动时，跑步运动员要知道跑道的长度、个人的时间、最快的时间、每次记录的时间。经过适当的训练和指导，他们研究个人的优势和弱点并寻求如何改进。在软件中，没有很明确的性能度量方法，很少有人能够了解个人的优势和弱点或看看怎样改进。本书中的方法将能帮助你评定自己的性能数据，识别改进的方法并指导你的改进。

为了改进个人的性能，本书还帮助建立起开发大型软件所需要的工程技能的基础，学习如何制定一个准确的计划、如何估计计划的准确性，以及如何依据计划跟踪个人的性能。还将学会使用缺陷管理、设计和代码评审、设计模板及过程分析方法，使用一个已定义的和经过度量的过程完成它们。这些度量和分析规范可以帮助你评价个人的性能，了解优势及找到待改进的地方。从而你就可以为你职业生涯中持续地进行个人改进而开发工具了。

在学习 PSP 中所涉及到的内容

在学习 PSP 中，我们所获得的收益取决于个人的投入。虽然有许多方法可以组织 PSP 课程，基本的需求是通读全书的 14 章内容，并且完成程序练习和报告练习，这些要求在 SEI 网站上可以获得（www.sei.cmu.edu/tsp/psp）。在这个网站中也提供了各种课程计划。原来的 PSP 课程共 10 个程序练习和 5 个报告练习。另外，补充了多方面的程序练习，除基本的需求是在 6 个 PSP 过程等级中每一个等级上写一个程序以外，还补充 2～4 个程序来掌握这些方法，并为继续下一步工作建立数据。

读这些章节不会花太长时间，可编程序大概所花的时间就不一样了。这些 PSP 练习程序已经写过几千次了，我从 8100 套 PSP 程序数据采样中选择一些作为书中的实例，在写这些程序时约二分之一的开发人员每个程序花了不足 4 小时，三分之一的人员平均在 3 小时以内。为了缩短时间，使用最熟悉的编程语言和环境，还要用简单易懂的方法进行设计。

由于 PSP 课程涉及写程序，人们常常认为它是个编程的课程。其实不是，即使写了所有要求的程序，但不遵循预先描述的 PSP 过程并收集、分析和使用所有特定的数据，就等于没有学习 PSP。这是一个过程课程，为了避免浪费时间，就要遵循这个预描述的过程来写每个程序。如果你在做时遇到问题，请向教员寻求帮助。

为了从 PSP 课程获得更多的收获，下面有两个建议。第一，不要用你工作中的编程任务替代本书的程序。在成千上万个完成了 PSP 训练的开发人员中，没有一个像那样做而成功的。课程培训涉及到学习新的和陌生的方法。课程中的练习像实验，在项目中工作时，程序员用陌生的方法普遍难于进行。第二，不要只读书，不做练习就试着应用方法。至少到今天为止没有一个人这样做就掌握了。只有完成了课程和练习后，才能够在工作中应用它们。

内容概览

本书通过 14 章分步介绍了 PSP 方法。第 1 章，描述了 PSP 整体原理及策略。第 2 章和第 3 章解释了如何遵循一个定义的过程和怎样收集和使用用于管理程序开发作业的数据。第 4 章至第 7 章包含了估算和计划，从第 8 章至第 12 章阐述了质量管理和设计。第 13 章描述了如何将 PSP 方法用于各种工作。第 14 章描述如何将 PSP 方法用于 TSP 过程，以及 TSP 指导经过 PSP 训练的软件工程师怎样在一个项目中使用这些方法。本书还讨论了在学习和使用 PSP 过程中的个人问题。

支持材料

这本书有大量的支持材料，可以从 www.sei.cmu.edu/tsp/psp 获得。主要的支持材料是收集数据和计划工具，PSP 练习任务集以及参考材料。还有能链接到 SEI 的 PSP 课程描述及 SEI 授权的组织的地址，他们帮助介绍 PSP 和 TSP。PSP 也可以作为大学的课程来讲授。为此联机支持材料包括教员指导、建议的 PSP 课程讲义及课程练习。这些内容

也公布在 SEI 网站上。

本书的背景和历史

这是我写的第 4 本关于 PSP 的书。我在 1995 年写了 *A Discipline for Software Engineering*[3]，在 1997 年出版了 *Introduction to the Personal Software Process*[4]。接着在 2000 年，我发表了 *Introduction to the Team Software Process*[5]。“Discipline”一书是作为计算机科学专业研究生课程教材，两本“引论”作为大学本科生教材。在这几年中，成千上万的软件开发人员接受了 PSP 课程，成百上千的 TSP 团队已经在他们的项目中使用了 PSP 方法。这个结果已经远远超出我所预料的。

这段经历展示出业界软件开发人员是非常需要一本 PSP 教材的。虽然 *Discipline for Software Engineering* 包括了所有需要的材料，但是它更多为学术性主题，对于开发人员来讲，其中一些内容不是必要的。由于课程周期也是业界组织主要关心的方面，在本书中则省略掉了这些更理论化的内容。

致谢

和原来开发 PSP 和 TSP 的软件开发人员和团队一起工作给了我一生中最有益的经验。没有那些敢于将这些理念用于他们工作中的人，这本书是根本不可能的。他们具有了解新的潜在价值的先见之明和智慧，并在实践中敢于尝试这些概念。我由衷地感谢他们。

在软件工程研究所（SEI），我的工作组在一起工作得很愉快。他们中的所有人都经过了 PSP 培训，并且我们都使用 TSP 计划和管理自己的工作。这个非常惊人和伟大的支持团队成员有 Dan Burton, Anita Carleton, Noopur Davis, Caroline Graettinger, Jim McHale, Julia Mullaney, Jim Over, Marsha Pomeroy-Huff, Dan Wall 和 Alan Willett。在需要时，我都会得到不

[3] Watts Humphrey, *A Discipline for Software Engineering*, Addison-Wesley, 1995.
[4] Watts Humphrey, *Introduction to the Personal Software Process*, Addison-Wesley, 1997.
[5] Watts Humphrey, *Introduction to the Team Software Process*, Addison-Wesley, 2000.

断的建议和支持。另外，Kim Campbell, Marlene MacDonald 和 Jodie Spielvogle 为我们提供真诚和大力的支持。我也感谢 SEI 的管理人员为我的工作提供鼓励和指导，这里有一个非常广阔的工作空间。

在写本书的过程中，我还得到了许多人的帮助和支持。有几千名软件工程师接受了 PSP 课程，他们提供了用于细化和解释方法的数据。我非常感谢许多 PSP 教员的努力、承诺和奉献，他们的支持是使这个材料得到更广泛使用和理解的基本保证。我也要感谢支持原来 PSP 工作的人们和参与评审以及使用早期 *A Discipline for Software Engineering* 草稿授课的人员。早期的工作为这本书提供了基础和经验。另外，特别要感谢早期帮助我评审此书草稿的人们，他们是 Jim Alsted, Dan Burton, Anita Carleton, David Carrington, Rod Chapman, Noopur Davis, Caroline Graettinger, Carol Grojean, Capers Jones, Jim McHale, Julia Mullaney, Bob Musson, Paul Newson, Jim Over, Marsha Pomeroy-Huff, Ron Radice, Mark Sebern, Victor Shtern, Yeun-Chung Tan, Dan Wall 和 Alan Willett.

最后还要感谢 Dr. John E. Yarnelle。在我的 *A Discipline for Software Engineering* 出版时，我就将它献给了 Dr. John E. Yarnelle。他是我的第一个数学老师，我有幸得到他的指导，这给了我毕生的激励和灵感。他具有超凡的能力使数学和科学不枯燥而且有趣味。我很幸运在早年就有这样一位令人奋进的老师。尽管是在许多年后我才有机会将书献给他，而他给予了我一生巨大的贡献。在发行那本 Discipline 书之后，我找到他并恢复了早期的不寻常的关系，因此我再奉献这本书来纪念 Dr. John E. Yarnelle。

Watts S.Humphrey

佛罗里达州，萨拉索塔市

CONTENTS