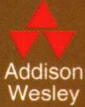


(英文版)



软件工程系列教材

# 个体软件过程

## Introduction to the Personal Software Process<sup>SM</sup>

Watts S. Humphrey



SEI SERIES IN SOFTWARE ENGINEERING

人民邮电出版社  
POSTS & TELECOMMUNICATIONS PRESS

软件工程系列教材

# 个体软件过程 (英文版)

Introduction to the Personal Software Process

Watts S. Humphrey

人民邮电出版社

软件工程系列教材  
个体软件过程（英文版）

---

- ◆ 著 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>  
读者热线 010-67132705  
北京汉魂图文设计有限公司制作  
北京朝阳展望印刷厂印刷  
新华书店总店北京发行所经销
- ◆ 开本: 800×1000 1/16  
印张: 19.25  
字数: 319 千字 2002 年 10 月第 1 版  
印数: 1-4 050 册 2002 年 10 月北京第 1 次印刷

著作权合同登记 图字: 01 - 2002 - 3748 号

ISBN 7-115-10349-6/TP · 2908

---

定价: 32.00 元

本书如有印装质量问题, 请与本社联系 电话: (010) 67129223

---

## 内容提要

---

《个体软件过程引论》(简称“PSPi”),是 Watts S. Humphrey 于 1997 年特为美国大学一年级学生编写的教科书。书中描述了很多资深的软件工程师解决软件工程问题的方法,特别是有关软件项目计划和软件质量控制方面的先进方法,并提供了很多练习来帮助读者掌握这些方法。

美国 Embry-Riddle 航空大学计算机科学系以本书初稿为教材,为该系一年级学生讲授了这门课程,经验证明,这对造就学生成为合格的软件专业人员很有帮助。每章之后还附有习题,以帮助读者复习与掌握该章的主要内容。由于本书是在两个学期的计算机科学或软件工导论课程中使用,所以本书的内容分成两部分,在第一学期中讲授时间管理(前 10 章),在第二学期中讲授质量问题(后 10 章)。

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,是进行软件过程改进和能力成熟度模型 SW-CMM 评估的重要参考资料。本书同样适用于软件过程改进人员、软件开发项目经理、程序员和一般编程爱好者在开发软件时参考。

---

# 序

---

*Introduction to the Personal Software Process* (个体软件过程引论) 这本著作是 Watts S. Humphrey 特地为大学一年级学生编写的教科书。该书描述了很多资深的软件工程师解决软件工程问题的方法, 其中包括 Humphrey 本人的许多宝贵经验, 并提供了很多练习来帮助学生掌握这些方法。根据美国 Carnegie-Mellon (卡内基梅隆) 大学软件工程研究所 Humphrey 等人的实践数据, 在应用了 PSP 之后, 软件系统中总的缺陷减少了 58.0%, 在测试阶段发现的缺陷减少了 71.9%, 生产效率提高了 20.8%。在美国, Embry-Riddle 航空大学计算机科学系率先以本书初稿为教材, 为该系一年级学生讲授了这门课程, 取得了很好的效果, 接着几十所大学相继开设了这门课程。经验证明, 这对造就学生成为合格的软件专业人员很有帮助。

为了在企业中推广 PSP, 需要高层经理的大力支持, 需要有合格的教员和合适的教材, 而且还需要所有参与人员的积极参与。然而, 遵循个体软件过程开发软件并不是一件轻而易举的事, 需要人们改变自己的工作方式。但是, 人们通常很难改变自己的工作习惯, 他们虽然愿意做些改变, 但往往离不开原有的习惯。只有在用他们自己的事实证明新的方法确实有效之后, 才愿意真正进行改变。因此, 推广个体软件过程需要以认真的实践为基础, 并努力总结亲身经历的经验和教训。

个体软件过程是一门实践性很强的学科, 需要通过编写程序才能真正掌握 PSP 的精髓。为了精简讲授内容, 减少重复实践, 可以将 PSP 课程的程序设计练习与其它课程的作业结合起来。在最近与 Humphrey 的一次通信中, 他认为如果在讲授个体软件过程 (PSP) 之前, 先讲一些统计过程控制原理, 对理解 PSP 的方法和理念非常有利。

诚然, PSP 方法的成功与否, 在很大程度上取决于教师激发学生学习和实践这些概念的能力。这不仅需要教师本人对 PSP 持积极态度, 而且需

## 2 序

要努力实践，认真积累数据，用自己的经验向学员展示 PSP 方法的威力。可以认为，教员是否按 PSP 原理进行认真的实践，是讲授好 PSP 课程的重要前提。

在 Embry-Riddle 航空大学计算机科学系的教员和学员为本书撰写的前言以及 Humphrey 本人在本书正文和他为本书撰写的前言中，都特别强调采集数据的重要性，并特别指出千万不要粗制滥造数据，否则所收集的数据就没有什么用处。因此，我们在学习 PSP 课程时，要按正确的方法去做，要努力理解 PSP 的整体框架和所蕴涵的概念，积极采集真实的数据，在自己的工作中坚持贯彻 PSP 的原则。诚然，要掌握 PSP 的方法，不仅需要认真参与有关 PSP 的培训，更需要在后续课程和实际工作中进一步提高、巩固和扩展。可以认为，这是掌握 PSP 方法的基本保证。

自从 1994 年以来，Humphrey 一直大力倡导这种方法，他在美国很多著名公司推行 PSP 方法，获得了很好的结果。而且还从 1996 年开始，大力倡导群组软件过程（也称小组软件开发过程）TSP 方法。应该指出，如果一个组织正在按照 CMM 改进过程，则 PSP 和 TSP 是与 CMM 完全相容的。如果一个组织还没有按照 CMM 改进过程，则有关 PSP 和 TSP 的训练，可以为未来的 CMM 实践奠定坚实的基础。他在最近还进一步指出，可以把 PSP 看作 CMM 五级的个体过程，而把 TSP 看作 CMM 五级的群组过程。因此要有效地进行软件过程改进，若能将软件能力成熟度模型 SW-CMM、个体软件过程 PSP 和群组软件过程 TSP 这三者紧密结合起来，并以统计过程控制理论为基础，相互配合、各有侧重，形成一个相互支持、不可分割的整体，一定能取得很好的效果。

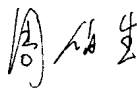
在美国 Carnegie-Mellon 大学软件工程研究所对外开设的课程中，对个体软件过程 PSP 给予了特别的重视。其中“个体软件过程引论”是一门历时两天的课程，讨论了 PSP 的关键概念以及相关的术语和理念，并为软件工程师人员参加群组软件过程作好准备。其中为软件工程师开设的 PSP 课程有两门，第一门讨论软件项目规划，第二门讨论软件产品质量，每门课都是五天。在第一门课中，除了讨论个体过程引论，还讨论了规模测量、规模估计、如何基于历史数据来估计规模和资源以及如何进行过程测量等问题。第二门课主要讨论缺陷管理、设计过程、设计验证、如何将 PSP 应用于大型项目以及如何进行过程开发等问题。这两门课程都是以 Humphrey 所著的另一本书《软件工程规范》（*The Discipline for Software Engineering*）为教本。

北京航空航天大学软件工程研究所从 1997 年开始在硕士研究生中开

设《个体软件过程引论》课程，最近两年改在本科生的软件工程课程中讲授 PSP 的内容，今年还对一些公司的软件工程师进行了 PSP 培训，这些都取得了一定的效果。但是由于实践时间不长，数据采集不全，需要进一步坚持改善与总结提高。

从 2000 年下半年以来，在我国软件企业界掀起能力成熟度模型 (CMM) 评估的热潮。需要着重指出的是，在软件能力成熟度模型 SW-CMM 的 18 个关键过程域中，有 12 个与个体软件过程 PSP 紧密相关，有 16 个与群组软件过程 TSP 紧密相关。因此，如果能熟悉个体软件过程 PSP 和群组软件过程 TSP，不仅有助于工程师改善工作效率，而且也非常有利于组织的过程改善。可以预料，在大学中讲授 PSP 和 TSP 课程，对培养合格的软件专业人员和加速我国软件产业的发展将起积极的推动作用。

我们应该看到，在软件领域我国在总体上离开世界先进水平还有相当大的差距。但是，我们不能跟随他国的脚印，走他人的老路。我们应该抓住机遇，直接针对未来的目标，在软件工程技术和软件工程管理两个方面，注意研究和倡导经过实践证明行之有效的软件工程的新原理和新方法。联系实际，认真实践，并充分利用我国丰富的人力资源和尊重教育的优良传统，大力培养各个层次的高质量的软件工程人员，使其具有开发各类大型、复杂软件系统的能力。我相信，人民邮电出版社把 *Introduction to the Personal Software Process* 这本著作影印出版，在把我国建设成为一个真正现代化的软件产业大国的历史任务中，特别是在培养和造就初中级软件工程技术人才的历史任务中，一定能起到积极的推动作用。



2002 年 9 月 20 日于北京

---

FOR BARBARA

**My love, my life, my wife**



---

# FACULTY FOREWORD

We used a draft version of this book to teach process principles in the first year Computer Science program at Embry-Riddle Aeronautical University. The book provides a subset of the Personal Software Process (PSP)<sup>SM</sup> elements and activities that a freshman can easily assimilate with the more traditional first-year programming topics. The book also provides the motivation and a structure for introducing students to disciplined personal practices. We have enjoyed using this book and feel it is helping our students to become competent software professionals.

For some years we have been trying to provide our students with realistic software engineering experiences. We have had moderate success with introducing software engineering theory and practice early in the curriculum and adding team projects to some upper division courses. Unfortunately, we have found that when students work on these projects, they do not understand time management, scheduling, and quality management. As industry has found, the ability of engineering teams to develop quality software products efficiently and effectively greatly depends on the ability of the individual engineers. On reflection, the students' problems with time and quality management are not surprising since we had not provided courses that show students how to plan and manage their work. We thus decided to try introducing process concepts at the beginning of the undergraduate curriculum.

We felt that beginning college students could best learn and benefit from time management practices, so we started by introducing this book's Chapter 1 through Chapter 10 materials in CS1. Although all students entering our CS1 course had

<sup>SM</sup>Personal Software Process and PSP are service marks of Carnegie Mellon University.

some programming experience, they were not yet ready for a formally defined software development process. They first needed exposure to the problems of modern software development before they could truly comprehend the roles and practices of software engineers.

After finishing CS1 and completing their first college semester, students were ready for a more disciplined way to develop programs. We then introduced the PSP process in CS2, using the materials in Chapters 11 through 20. Here, the students planned each of their programming projects. Following the defined PSP practices, they used their own historical data to estimate size, effort, and quality (defect projection). They also collected and recorded actual data for each project on a summary report form.

After a year of experience, we have found that the approach of introducing process activities to beginning computer science students can work. What we mean by “can work” is that students can learn how to use the process outlined in the book. They do, eventually, see the value of recording effort, size, and quality data, and they can use these data in planning projects and analyzing their personal effectiveness. Collecting data on their own work gives them a quantitative basis for estimating. They regularly perform structured reviews and they learn to follow defined development phases in their work (e.g., planning, design, coding, compiling, testing, and postmortem). We also feel that delaying the introduction of PSP for another semester (or year) would allow sloppy and undisciplined programming practices to become more entrenched and would make the students more resistant to change.

The PSP has helped the students understand the importance of a disciplined approach to developing software. It also provides a more rigorous foundation for later introducing more advanced individual and team topics. For the most part, student data are accurate, but one must be careful to analyze and reject suspicious data. Unfortunately, the students did not become better at scheduling their work. Many still put off assignments until near their due date—a perennial beginning programmer problem.

Not surprisingly, we discovered that the success of the PSP approach was highly dependent upon our ability to motivate students to learn and practice these concepts. We used the ideas and arguments in this book to encourage a positive view of process methods. We found that providing the class with regular feedback and analysis of class data stimulated students’ interest in looking more closely at their personal data. Inviting industry professionals to discuss their process experiences with the class was also very helpful.

There were some problems at first in teaching the new courses. Initially, we did not sufficiently integrate the PSP materials with the rest of the CS1 and CS2 courses. The students thus had trouble relating the time management activities to their programming work. We also failed to provide sufficient feedback of class aggregate data.

An interesting and beneficial side effect of the PSP is the large supply of data available to the teacher. In CS1 we get weekly activity reports on how students spend their time in the course. In CS2 we get a PSP summary on each programming project that provides size, effort, and defect data. These data often provoke discussion about the methods taught in the course and how they affect programmer productivity and program quality. The PSP provides a quantitative basis for detailed analysis and discussion of such questions.

We are continuing to teach the PSP in the freshman year of our program. We also require students who have completed CS1 and CS2 to use the PSP in the Data Structures and Algorithms course that follows CS2. We believe this will better prepare them for the more complex team projects they will face in their junior and senior years. We also plan to guide students in extending and enhancing the PSP for their subsequent courses.

We have found this book helpful in introducing our students to professional software disciplines and hope that other students and teachers using this book will experience similar benefits.

Thomas B. Hilburn, Aboalfazl Salimi, Massood Towhidnejad  
*Embry-Riddle Aeronautical University*

---

# STUDENT FOREWORD

After we finished the PSP course in our freshman year, a couple of the faculty at Embry-Riddle Aeronautical University asked if we would like to collaborate on writing a foreword to the finished textbook. We agreed. Since we were not sure how to write a foreword, they suggested that we merely answer some questions. Here are the questions and our answers:

**1. What type of tasks did you do in the PSP course?**

We kept track of all the time we spent on programming assignments and projects. There was a lot of paperwork to keep track of. We also kept track of program size and defects and used the data we collected to estimate time, size, and defects on future projects.

**2. How did it work? How did this material fit with the material in the other freshman courses?**

It fit well into the course work, and having an estimate helped provide confidence in what you were doing.

At the beginning it looks like the PSP work is a hindrance to the other course work, but once you get to the end of the course, you realize that these activities actually help you complete your work. You go through the course and keep asking yourself, "Why am I doing this?," but later you start to see that having an estimate of what it is going to take you to complete a program actually helps you.

It is very important not to fudge the data (times) because then the data you collect are not as helpful.

**3. What did you learn?**

In addition to what we have already talked about, you learn how you can use your time more efficiently and do some work before you get to the computer. You end up doing a lot of work on paper before you get to the computer.

You also learn about your mistakes and other people's mistakes (through examples and discussions). It also helps you keep your programming organized (since you do work on paper before you get to the computer). The PSP is also something that can be used in other activities (not only in software development), although the forms need to be modified.

**4. What would you recommend to other students who will use the PSP in the future?**

Do it right. Don't fudge it. Follow instructions. Try to understand the big picture and understand the concepts. Don't let the paperwork get to you; it will pay off.

Ben Bishop, Andrew Henderson, Michael Patrick  
*Embry-Riddle Aeronautical University*

---

# PREFACE

If you are studying to be a software engineer, this book is designed for you. It describes the methods many experienced engineers use to do competent work and it provides exercises to help you learn these methods. Each chapter describes a single topic in which you will become skilled as you practice the homework exercises. Completed examples of each exercise will help you to check your work.

---

## Why I Wrote This Book

Developing software products involves more than just stringing programming instructions together and getting them to run on a computer. It requires meeting customer requirements at an agreed cost and schedule. To be successful, software engineers need to consistently produce high-quality programs on schedule and at their planned costs. This book shows you how to do this. It introduces the Personal Software Process (PSP), which is a guide to using disciplined personal practices to do superior software engineering.

The PSP will show you how to plan and track your work and how to consistently produce high-quality software. Using PSP will also give you data that show the effectiveness of your work and identify your strengths and weaknesses. This tool is like the stop-watch and distance measures you need to test yourself on joining a track team and deciding which events to try for. To make an intelligent decision, you would need such measures to know where you excel and where you need to improve. Like a track team, software engineering has many specialties, and en-

engineers have widely varying skills and talents. To have a successful and rewarding career, you need to know your skills and abilities, strive to improve them, and capitalize on your unique talents in the work you do. The PSP will help you do this.

---

## **Using the PSP**

By using the PSP, you will be practicing the skills and methods professional software engineers have developed through many years of trial and error. Building on the experiences of your predecessors will help you to learn more quickly and avoid repeating their errors. The essence of being a professional is understanding what others have done before you and building on their experiences.

## **How Students Will Benefit**

While the PSP is now generally introduced in graduate software engineering programs, its principles can be learned and practiced by beginning students. This book is designed to introduce the PSP methods in gradual steps as you do your other course work. As you read each chapter, do the exercises at the end. These show how to manage your time, how to plan and track your work, and how to consistently produce high-quality programs.

Since it takes time to develop effective skills and habits, you should practice the PSP methods with every software assignment. If you do this, you will have learned, practiced, and perfected these skills before you need them in software engineering work.

## **How Working Engineers Can Use This Book**

Practicing software engineers can also use this book to learn the rudiments of the PSP. I suggest that you work through the exercises from the beginning of the book to the end, using them as guides for improving the way you do your regular work. Practice each exercise until it feels natural, then read the next chapter and add its methods. Again, practice both the new and the already learned methods before advancing to the next step. The key is to take the time to master one method before progressing to the next.

With some dedication and discipline, you should have no trouble mastering this material by yourself. Success is more likely, however, if you do this work in a class or with a group of co-workers with whom you can exchange experiences and share ideas. In any case, plan to spend about an hour or two each week studying

the textbook, recording and analyzing your PSP data, and adapting the PSP methods to your work. Although the time you need to learn the PSP will depend on your current habits and practices, once you have completed this material, you will have a sound foundation for continued professional development. Note, however, that the key to learning the PSP is to look at and think about the data on your work and what these data tell you about your personal performance.

---

## Some Suggestions for Instructors

This book is designed as a companion text for traditional two-semester computer science or software engineering courses. It makes no assumptions beyond a college preparatory education. The book presents the PSP in steps that students can use with their regular course work. The exercises in the first ten chapters are quite general and can be used with programming or nonprogramming work. The exercises in the final ten chapters are designed for use with about six to eight or more small programming exercises.

Although some students first learn to program in college, many now learn basic programming in high school. This material is thus designed for use in either a first programming course or a more advanced course. Whether the students already know how to program or are just learning, they should readily understand the material and find it immediately helpful.

This material is an introduction to the PSP and not a replacement for it. The book does not, for instance, cover the statistical techniques needed for accurate estimating or data analysis. It also does not cover the methods for scaling up the PSP for larger projects or the process definition and improvement techniques used in applying the PSP to tasks other than writing small programs. The full PSP should thus be taught at a later point in the student's educational program.<sup>1</sup>

As you lead students through the material of this book and they complete the assignments, they will learn to track and monitor their work, to manage their time, and to make plans. In the second semester, they will learn about program quality and ways to do reviews and use various quality measurement and management methods. They will also learn about defects, their causes, and the engineers' personal responsibility for the quality of the products they produce. At the end of the two-semester course, the students will have learned the rudiments of the PSP. To build on this foundation and to give them added experience with

<sup>1</sup>The PSP and the PSP course are described in more detail in my textbook *A Discipline for Software Engineering* (Reading, MA: Addison-Wesley, 1995). The textbook support materials include an instructor's guide and an instructor's diskette with lecture overheads and assignment materials.



these methods, later courses should require the students to continue using the PSP.

## **A Teaching Strategy**

Since this book is meant to be used in conjunction with a two-semester introductory computer science or software engineering course, its material is divided into a first semester that covers time management (10 chapters) and a second semester that covers quality. Teaching this material takes about six lecture hours spread over the two semesters. Since the students use PSP methods while doing their currently required course work, this material does not add significantly to student workload. Any additional time students spend learning these methods is very likely compensated for by the efficiency they gain.

As you progress through the textbook, assign the exercises at the back of each chapter. Experience has shown that it is best to cover the first 10 chapters in the first few weeks of the first semester. The students then have the rest of the semester to practice the methods introduced. The second semester should follow the same strategy of introducing the PSP topics in the first few semester weeks and then using these methods during the balance of the semester.

It is crucial to present this material as an integral part of the course. Explain that these are essential software engineering methods that the students must learn and practice to satisfactorily complete the course. As you assign each exercise, explain that the students' grades will depend both on the quality of their work and on how well they apply the PSP methods. They must do each PSP exercise and then continue to use each PSP method after it is first introduced. The course strategy, suggested lecture contents, and assignment kits are included in the instructor's guide and support materials described on the last page of this text.

## **Instructor Preparation**

In teaching this course, you will find it helpful to have used these methods yourself. You could, for example, use the planning and time management methods to prepare the class lectures or grade homework. After you have personally used the PSP, you will better appreciate the personal discipline required. This background will help you explain the PSP to the students and guide them in its use. When they find that you have used the PSP, they are more likely to use it themselves.