# The Personal Software Process℠ (PSP℠)

CMU/SEI-2000-TR-022
ESC-TR-2000-022

Watts S. Humphrey

November 2000

# 个人软件过程

台 林泰龙 译
钟华 校对国内术语并翻译图

# 目 录

致谢 ..... 4
摘要 ..... 4
1.软件质量 ..... 5
2.PSP 的发展过程 ..... 6
3.PSP 的原理 ..... 7
4.PSP 过程的架构 ..... 8
4.1 PSP 计划 ..... 13
4.2 使用 PROBE 规模估算 ..... 14
4.3 计算 ..... 15
4.4 使用 PROBE 资源估算 ..... 15
5.PSP 数据收集 ..... 16
5.1 时间测量 ..... 16
5.2 规模测量 ..... 17
5.2.1 代码行 ..... 17
5.2.2 规模类别 ..... 17
5.2.3 规模纪录 ..... 18
5.3 质量测量 ..... 19
6.PSP 质量管理 ..... 28
6.1 缺陷和质量 ..... 29
6.2 工程师的职责 ..... 29
6.3 早期缺陷移除 ..... 30
6.4 缺陷预防 ..... 30
7.PSP 设计 ..... 31
8.PSP 规程 ..... 33
9.导入 PSP ..... 33
10.PSP 成果 ..... 35
11.PSP 和过程改进 ..... 38
12.PSP 状态与未来趋势 ..... 39
参考 ..... 40

## 图表索引

图 1: PSP 过程流
图 2: PSP 过程要素
图 3: 项目计划过程
图 4: 开发 vs.使用缺陷；IBM 发行版 1(r2=.9267)
图 5: PSP/TSP 课程结构
图 6:工作量估算精确度
图 7: 缺陷水平改善
图 8: 生产率的成果

## 表格索引

表 1:项目计划摘要
表 2: PSP1 过程脚本
表 3: C++ 对象规模数据

表 4: PSP 缺陷注入与移除的数据
表 5: 移除缺陷的范例
表 6: 缺陷注入与移除的范例
表 7: 合格率计算
表 8: 对象规格结构
表 9: PSP 模板结构

# 致谢

# Acknowledgements

Contributing reviewers for this report were Noopur Davis, Frank Gmeindl, Marsha Pomeroy Huff, Alan Koch, Don McAndrews, Jim McHale, Julia Mullaney, Jim Over, and Bill Peterson.
为本报告做出贡献的审阅/评论者包括 Noopur Davis, Frank Gmeindl, Marsha Pomeroy Huff, Alan Koch, Don McAndrews, Jim McHale, Julia Mullaney, Jim Over, and Bill Peterson.

# 摘要

# Abstract

The Personal Software Process$^{SM}$ (PSP$^{SM}$) provides engineers with a disciplined personal framework for doing software work. The PSP process consists of a set of methods, forms, and scripts that show software engineers how to plan, measure, and manage their work. It is introduced with a textbook and a course that are designed for both industrial and academic use. The PSP is designed for use with any programming language or design methodology and it can be used for most aspects of software work, including writing requirements, running tests, defining processes, and repairing defects. When engineers use the PSP, the recommended process goal is to produce zero-defect products on schedule and within planned costs. When used with the Team Software Process$^{SM}$ (TSP$^{SM}$), the PSP has been effective in helping engineers achieve these objectives.

个人软件过程(PSP$^{SM}$)给工程师提供一个有规律开展软件工作的个人框架。PSP 过程由一套指导工程师如何计划、测量和管理他们工作的理论、格式和程序组成。PSP 是通过为工业和高等院校使用而设计的教科书和课程来引入的。PSP 被设计以任何编程语言或者设计方法来使用，并且可以用于大多数软件工作方面，包括编写需求、执行测试、定义过程以及修复缺陷。当工程师使用 PSP 的时候，推荐的过程目标是：按照进度并在计划成本之内，生产零缺陷产品。当和团队软件过程(TSP$^{SM}$)一起使用的时候，PSP 在帮助工程师达到这些目标方面非常有效。

This report describes in detail what the PSP is and how it works. Starting with a brief discussion of the relationship of the PSP to general quality principles, the report describes how the PSP was developed, its principles, and its methods. Next is a summary of the PSP courses, the strategy used for teaching the PSP, selected data on PSP experience, PSP adoption in university curricula, and the status of PSP introduction into industry. The report concludes with comments on likely future trends involving the PSP.

本报告详细描述了什么是 PSP 以及 PSP 是如何工作的。首先从简单探讨 PSP 与一般质量原则之间的关系开始，接着 描述 PSP 是怎样被开发出来的，它的原理和方法是什么。接下来是 PSP 课程、讲授 PSP 使用的策略、在 PSP 经验上选择数据、PSP 调整为大学课程、以及 PSP 引入工业的状态等等摘要说明。本报告以评论涉及 PSP 的可能未来趋势作为总结。

SM Personal Software Process, PSP, Team Software Process, and TSP are service marks of Carnegie Mellon University.

® Capability Maturity Model and CMM are registered in the U.S. Patent and Trademark Office.

# 1.软件质量

# Software Quality

Until shortly after World War II, the quality strategy in most industrial organizations was based almost entirely on testing. Groups typically established special quality departments to find and fix problems after products had been produced. It was not until the 1970s and 1980s that W. Edwards Deming and J.M. Juran convinced U.S. industry to focus on improving the way people did their jobs [Deming 82, Juran 88]. In the succeeding years, this focus on working processes has been responsible for major improvements in the quality of automobiles, electronics, or almost any other kind of product. The traditional test-and-fix strategy is now recognized as expensive, time-consuming, and ineffective for engineering and manufacturing work.

直到在第二次世界大战之后不久，大多数工业组织的质量策略差不多完全以测试为基础。团队典型做法是建立特定的质量部门来查找和修复产品生产出来后的问题。直到 1970 和 1980 年代 Deming 和 Juran 才说服美国工业界，要集中焦点于改进人们的工作方法上。随后几年，关注工作流程的做法才承担起包括汽车、电子或者几乎任何类型产品主要的质量改进责任。现在已经意识到，工程和制造工作传统的测试并修复(test-and-fix)策略是昂贵、耗时而且没有效率的。

Even though most industrial organizations have now adopted modern quality principles, the software community has continued to rely on testing as the principal quality management method. For software, the first major step in the direction pioneered by Deming and Juran was taken by Michael Fagan when in 1976 he introduced software inspections [Fagan 76, Fagan 86]. By using inspections, organizations have substantially improved software quality. Another significant step in software quality improvement was taken with the initial introduction of the Capability Maturity Model®. (CMM®.) for software in 1987 [Humphrey 89, Paulk 95]. The CMM's principal focus was on the management system and the support and assistance provided to the development engineers. The CMM has had a substantial positive effect on the performance of software organizations [Herbsleb 97].

尽管大部分工业组织目前已经采取了现代质量原理，软件业界仍旧依赖于测试作为主要的质量管理方法。对软件而言，在 Deming 和 Juran 开辟的方向上前进的重要一步，是 Michael Fagan 做出的，当他在 1976 年引入软件审查(software inspections)时候。通过使用软件审查，组织充分改进了软件质量。在软件质量改进上重要的另外一步，是在 1978 年软件能力成熟度模型(CMM®)的初步导入。CMM 的原则焦点放在管理体系上以及给开发工程师提供辅助和支持。CMM 在软件组织的效能方面已经产生了实质性的正面效果。

A further significant step in software quality improvement was taken with the Personal Software Process (PSP) [Humphrey 95]. The PSP extends the improvement process to the people who actually do the work—the practicing engineers. The PSP concentrates on the work practices of the individual engineers. The principle behind the PSP is that to produce quality software systems, every engineer who works on the system must do quality work.

在软件质量改进方面进一步重要步骤是采取个人软件过程(PSP)。PSP 将改进过程拓展到实际执行工作的人员－从事工作的工程师－身上。PSP 专心于个体工程师的实践工作。PSP

背后的原理：是要产生具备质量的软件系统，在系统中工作的每个工程师必须参与质量工作。
The PSP is designed to help software professionals consistently use sound engineering practices. It shows them how to plan and track their work, use a defined and measured process, establish measurable goals, and track performance against these goals. The PSP shows engineers how to manage quality from the beginning of the job, how to analyze the results of each job, and how to use the results to improve the process for the next project.

PSP 设计用于帮助软件专业人士前后一致地使用充分的工程实践，指导工程师如何计划和跟踪其工作、使用一个定义好的和可测量的过程、建立度量目标，并且对这些目标跟踪执行。PSP 指导工程师在工作一开始如何管理质量，如何分析每个工作的结果，并且如何运用这些结果改进下次项目的过程。

## 2.PSP 的发展过程

## How the PSP Was Developed

After he led the initial development of the CMM for software, Watts Humphrey decided to apply CMM principles to writing small programs. Many people had been asking how to apply the CMM to small organizations or to the work of small software teams. While the CMM principles applied to such groups, more guidance was needed on precisely what to do. Humphrey decided to personally use CMM principles to develop module-sized programs both to see if the approach would work and to figure out how to convince software engineers to adopt such practices.

在领导对软件的 CMM 初始开发之后，Watts Humphrey 决定将 CMM 原理应用到编写小型程序上面。有许多人在问，要如何把 CMM 应用到小型组织或者小型软件团队的工作中。由于当 CMM 原理应用在这类的团体中时，需要更多怎么做的明确指导。于是 Humphrey 决定亲自使用 CMM 原则来开发模块规模的程序，一来看看是否这种途径能够行的通，二来要研究出如何说服软件工程师采用这样的实践。

In developing module-sized programs, Humphrey personally used all of the software CMM practices up through Level 5. Shortly after he started this project in April 1989, the Software Engineering Institute (SEI) made Humphrey an SEI fellow, enabling him to spend full time on the PSP research. Over the next three years, he developed a total of 62 programs and defined about 15 PSP process versions. He used the Pascal, Object Pascal, and C++ programming languages to develop about 25,000 lines of code. From this experience, he concluded that the Deming and Juran process management principles were just as applicable to the work of individual software engineers as they were to other fields of technology.

在开发模块规模程序期间，Humphrey 亲自使用所有的软件 CMM 实践直到第 5 级。在 1989 年 4 月开始这个项目不久以后，软件工程研究所(SEI)让 Humphrey 成为其一个研究员，使他可以将全部的时间投入在 PSP 的研究上。在接下来 3 年，他开发了总数为 62 个程序和定义了大概 15 个 PSP 过程版本。Humphrey 使用 Pascal, Object Pascal 和 C++编程语言来开发大约 25,000 行代码。从这些经验中，他得出结论：Deming 和 Juran 过程管理原理，和在其它科技领域中一样，都可以应用在个体软件工程师的工作上。

Humphrey next wrote a textbook manuscript that he provided to several associates who planned to teach PSP courses. In September 1993, Howie Dow taught the first PSP course to four graduate students at the University of Massachusetts (Lowell). Humphrey also taught the PSP course during the winter semester of 1993-1994 at Carnegie Mellon

University, as did Nazim Madhavji at McGill University and Soheil Khajanoori at Embry Riddle Aeronautical University. Based on the experiences and data from these initial courses, Humphrey revised the PSP textbook manuscript and published the final version in late 1994 [Humphrey 95]. At about the same time, Jim Over and Neil Reizer of the SEI and Robert Powels of Advanced Information Services (AIS) developed the first course to train instructors to teach the PSP in industry. Watts Humphrey and the SEI have continued working on PSP development and introduction by applying the same principles to the work of engineering teams. This work is called the Team Software Process.

Humphrey 随后撰写了一本教科书原稿，提供给一些计划教授 PSP 课程的相关人员。在 1993 年 9 月，Howie Dow 在马萨诸塞州(洛厄尔)大学给 4 个研究生上了第一堂 PSP 课。Humphrey 在 1993－1994 上学期在卡梅隆大学教授 PSP 课程，同时 Nazim Madhavji 在马克吉尔大学以及 Soheil Khajanoori 在安布里里得航空大学也在教授 PSP 课程。基于来自这些最初课程的经验和数据，Humphrey 修订了 PSP 教科书原稿并且在 1994 年晚期发行了最终版本[Humphrey 95]。几乎在同一时间里面，SEI 的 Jim Over 和 Neil Reizer 以及先进信息服务的 Robert Powels 研制出第一套用于在工业中教授 PSP 的培训教师课。Watts Humphrey 与 SEI 继续钻研 PSP 发展和推广－通过在工程团队工作中应用相同原理，这一工作称之为团队软件过程(Team Software Process) 。

## 3.PSP 的原理

## The Principles of the PSP

The PSP design is based on the following planning and quality principles:

PSP 设计基于以下计划的质量原理：

- Every engineer is different; to be most effective, engineers must plan their work and they must base their plans on their own personal data.
- 每个工程师都有差异，要获得最大的效果，工程师必须计划他们的工作并且其计划必须以自己的个人数据为基础。
- To consistently improve their performance, engineers must personally use well-defined and measured processes.
- 要持续改进他们的绩效，工程师必须亲自运用定义良好和处于测量状况下的过程
- To produce quality products, engineers must feel personally responsible for the quality of their products. Superior products are not produced by mistake; engineers must strive to do quality work.
- 要生产质量产品，工程师必须能够自觉其对产品质量的负责。优异的产品无法通过错误生产出来的，工程师必须努力开展质量工作。
- It costs less to find and fix defects earlier in a process than later.
- 在过程的早期发现和修复缺陷比晚期花费的少
- It is more efficient to prevent defects than to find and fix them.

- 预防缺陷比查找和修复缺陷更有效
- The right way is always the fastest and cheapest way to do a job.
- 正确方法是永远采用最快和成本最少的方法来工作。

To do a software engineering job in the right way, engineers must plan their work before committing to or starting on a job, and they must use a defined process to plan the work.

To understand their personal performance, they must measure the time that they spend on each job step, the defects that they inject and remove, and the sizes of the products they produce.

To consistently produce quality products, engineers must plan, measure, and track product quality, and they must focus on quality from the beginning of a job. Finally, they must analyze the results of each job and use these findings to improve their personal processes.

要在正确的方法上开展软件工程工作，工程师在承诺或者开始工作之前，必须对工作进行计划，并且必须使用定义好的过程来计划工作。

要理解他们个人的绩效。工程师必须测量其花费在每个工作步骤上的时间、他们注入和移除的缺陷以及他们所生产产品的规模。

要持续生产具备质量的产品，工程师必须对产品的质量进行计划、测量和跟踪，而且必须在工作开始的时候，就将焦点放在质量上。最后，工程人员必须对每一项工作的结果进行分析，并且运用分析的结果来改善其个人的过程(personal processes)。

## 4.PSP 过程的架构

## The PSP Process Structure

The structure of the PSP process is shown conceptually in Figure 1. Starting with a requirements statement, the first step in the PSP process is planning. There is a planning script that guides this work and a plan summary for recording the planning data. While the engineers are following the script to do the work, they record their time and defect data on the time and defect logs. At the end of the job, during the postmortem phase (PM), they summarize the time and defect data from the logs, measure the program size, and enter these data in the plan summary form. When done, they deliver the finished product along with the completed plan summary form. A copy of the PSP1 plan summary is shown in Table 1.

PSP 过程的架构，概念上如图 1 所示。从需求说明开始，PSP 过程的第一步是计划。有

指导工作的计划脚本以及用于记录计划数据的计划摘要。当工程师遵循脚本工作时，他们在时间和缺陷日志中记录他们的时间和缺陷数据。在工作结束后，在事后检查阶段，工程师从日志中汇总时间和缺陷数据，测量程序规模并却将这些数据填入到计划摘要表格中。在这些工作完成后，他们和完整的计划摘要表格一起交付完成产品。PSP 计划摘要表格的副本如表 1 中显示。

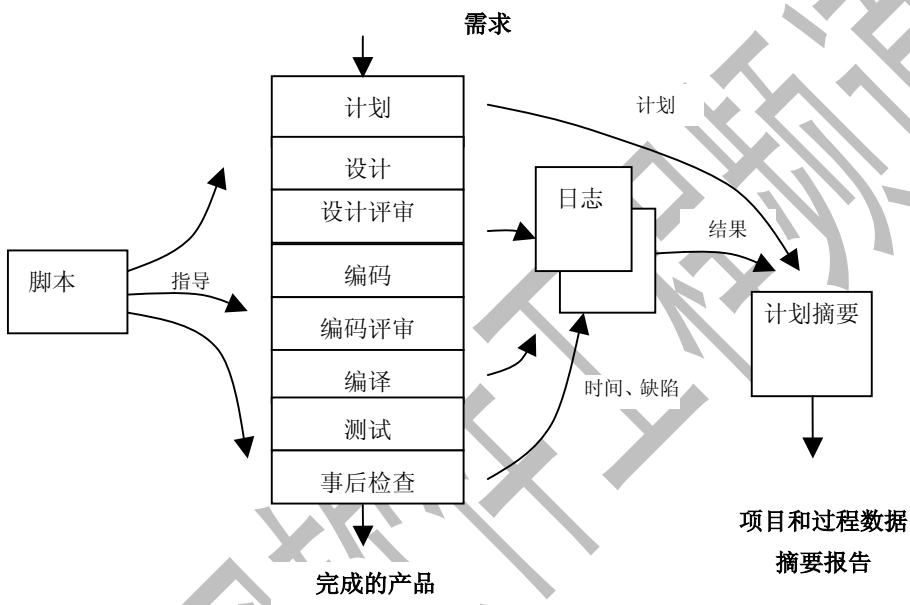



*Figure1: PSP Process Flow*

*图 1：PSP 过程流*

*Table1:PSP1 Project Plan Summary*

*表 1：PSP1 项目计划摘要*

| | | | |
|---|---|---|---|
| 学生 | | 日期 | |
| 程序 | | 程序 # | |
| 指导者 | | 语言 | |
| 摘要 | ***计划*** | ***实际*** | ***截止目前*** |
| ***LOC/小时*** | | | |
| **程序规模(LOC)**: | ***计划*** | ***实际*** | ***截止目前*** |
| 基数 (B) | | | |
| 删除 (D) | （测量） | （测量） | |

| | | | | |
|---|---|---|---|---|
| 修改 (M) | （估算） | （计算） | | |
| 增加 (A) | （估算） | （计算） | | |
| 重用 (R) | (N-M) | (T-D+D-R) | | |
| 新增&变更总计(N) | （估算） | （计算） | | |
| LOC 总计 (T) | （估算） | (A+M) | | |
| 新重用总计 | (N+B-M-D+R) | （测量） | | |
| 对象 LOC 总计 (E) | | | | |
| **阶段中的时间 (分)** | **计划** | **实际** | **截止目前** | **截止目前%** |
| 计划 | | | | |
| 设计 | | | | |
| 代码 | | | | |
| 编译 | | | | |
| 测试 | | | | |
| 事后检查 | | | | |
| 总计 | | | | |
| **缺陷注入** | | **实际** | **截止目前** | **截止目前%** |
| 计划 | | | | |
| 设计 | | | | |
| 代码 | | | | |
| 编译 | | | | |
| 测试 | | | | |
| 开发总计 | | | | |
| **缺陷移除** | | **实际** | **截止目前** | **截止目前%** |
| 计划 | | | | |
| 设计 | | | | |
| 代码 | | | | |
| 编译 | | | | |
| 测试 | | | | |
| 开发总计 | | | | |
| 开发之后 | | | | |

Since the PSP process has a number of methods that are not generally practiced by engineers, the PSP methods are introduced in a series of seven process versions. These versions are labeled PSP0 through PSP3, and each version has a similar set of logs, forms, scripts, and standards, as shown in Figure 2. The process scripts define the steps for each part of the process, the logs and forms provide templates for recording and storing data, and the standards guide the engineers as they do the work.

PSP 过程有许多的方法工程师并未普遍实践，所以 PSP 方法通过一系列的 7 个过程版本来进行介绍。这些版本是以 PSP0 到 PSP3 来标示的，并且每个版本有一组类似的日志、

表格、脚本以及标准，如图 2 所示。过程脚本定义过程每一部分的步骤，日志和表格提供记录与存储数据的模板，标准用于工程师进行工作时提供指导。

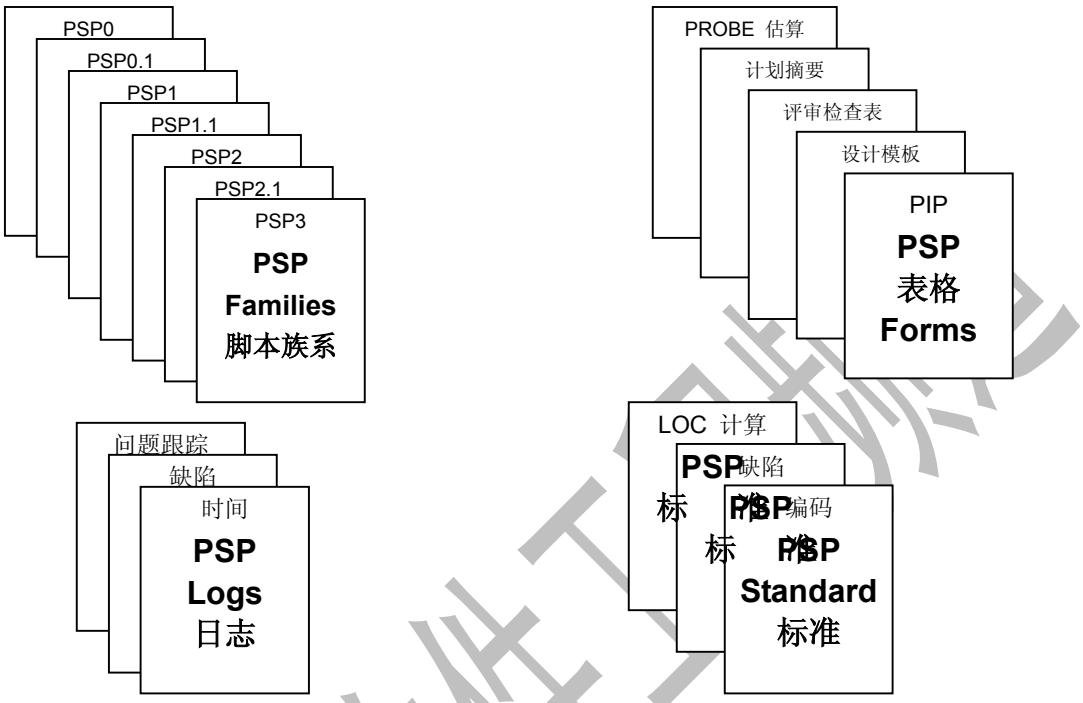



*Figure2: PSP process Element*

*图 2：PSP 过程要素*

An example PSP script is shown in Table 2. A PSP script is what Deming called an operational process [Deming 82]. In other words, it is a process that is designed to be used. It is constructed in a simple-to-use format with short and precise instructions. While scripts describe what to do, they are more like checklists than tutorials. They do not include the instructional materials that would be needed by untrained users. The purpose of the script is to guide engineers in consistently using a process that they understand. The next several sections of this report describe the various methods that the PSP uses for planning, estimating, data gathering, quality management, and design.

PSP 脚本的范例如表 2 所示。PSP 脚本就是 Deming 所称的作业过程(operational process)。换句话说，就是一个设计来使用的流程。它是采用简单易用的格式建构的，具有简短而精准的(工作)指示。当脚本描述做什么的时候，其比起指南来更象检查表，脚本不包括未受训用户应当需要的指导材料。脚本的目的是指导工程师持续使用他们了解的过程，本报告书后续的数个章节将描述 PSP 为计划、估算、数据收集、质量管理和设计使用的不同的方法。

*Table 2: PSP1 Process Script*

*表 2：PSP1 过程脚本*

| Phase Number<br>阶段编号 | Purpose<br>目的 | To guide you in developing module-level programs<br>在开发模块级别程序中的指导 |
|---|---|---|
| | Entry Criteria<br>进入标准 | . Problem description 问题描述<br>. PSP1 Project Plan Summary form<br>PSP1 项目计划摘要表格<br>. Size Estimating Template 规模估算模板<br>. Historical estimate and actual size data<br>历史估算和实际规模数据<br>. Time and Defect Recording Logs 时间和缺陷记录日志<br>. Defect Type Standard 缺陷类型标准<br>. Stop watch (optional) 停止观察（可选） |
| 1 | Planning<br>计划 | . Produce or obtain a requirements statement.<br>产品或者获取需求描述<br>. Use the PROBE method to estimate the total new and changed LOC required.<br>使用 PROBE 方法估算总的新和变更的 LOC 需求<br>. Complete the Size Estimate Template. 完全的规模估算模板<br>. Estimate the required development time. 估算需求开发时间<br>. Enter the plan data in the Project Plan Summary form.<br>数据计划数据到项目计划摘要表格<br>. Complete the Time Recording Log. 完全的时间记录日志 |
| 2 | Development<br>开发 | . Design the program. 设计程序<br>. Implement the design. 执行设计<br>. Compile the program and fix and log all defects found.<br>编译程序并且修复和归档所有发现的缺陷<br>. Test the program and fix and log all defects found.<br>测试程序并且修复和归档所有发现的缺陷<br>. Complete the Time Recording Log.<br>完全的时间记录日志 |
| 3 | Postmortem<br>事后检查 | Complete the Project Plan Summary form with actual time, defect, and size data.<br>完全的项目计划摘要表格，有实际时间、缺陷和规模数据 |
| | Exit Criteria<br>退出标准 | . A thoroughly tested program<br>充分的测试程序<br>. Completed Project Plan Summary form with estimated and actual data<br>完全的项目计划摘要表格，有估算和实际数据 |

| | | |
|---|---|---|
| | | . Completed Size Estimating Template 完全的规模估算表板<br>. Completed Test Report Template 完全的测试报告模板<br>. Completed PIP forms 完全的 PIP 表格<br>. Completed Defect and Time Recording Logs<br>完全的缺陷和时间记录日志 |

## 4.1 PSP 计划

## PSP Planning

The PSP planning process is shown in Figure 3. The following paragraphs describe the tasks in this figure.

PSP 的计划过程如图 3 所示，以下各段一一描述图表中的任务。

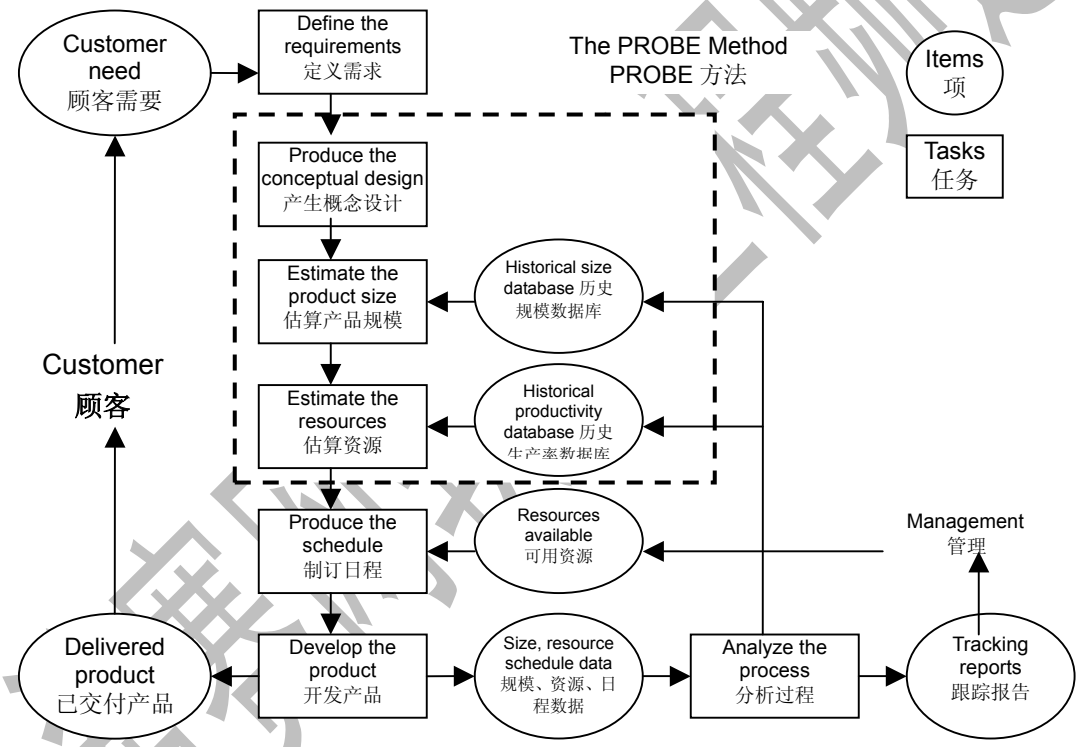



*Figure 3: Project Planning Process*

*图 3：项目计划过程*

**Requirements.** Engineers start planning by defining the work that needs to be done in as much detail as possible. If all they have is a one-sentence requirements statement, then that statement must be the basis for the plan. Of course, the accuracy of the estimate and plan is heavily influenced by how much the engineers know about the work to be done.

**需求** 工程师计划的第一步，是通过尽可能详细定义需要完成的工作。如果他们所拥有的，是一句简单的需求说明，则该项说明就必然是计划的基础。当然，工程师对于应完成之工作事项的了解程度，会对估算与计划的精准度，产生重大的影响。

**Conceptual Design.** To make an estimate and a plan, engineers first define how the product is to be designed and built. However, since the planning phase is too early to produce a complete product design, engineers produce what is called a conceptual design. This is a first, rough guess at what the product would look like if the engineers had to build it based on what they currently know. Later, during the design phase, the engineers examine design alternatives and produce a complete product design.

**概念设计** 要进行估算和计划，工程师首先定义如何设计和构建产品。然而，由于计划阶段太早，无法提出一个完整的产品设计。因此，工程师会提出一种称为概念设计的东西。若是要求工程师一定要依据他们目前所知来建构产品，那么，这会是一个产品看上去会象什么的初步粗糙的猜测。其后，在设计阶段，工程师检查设计备选方案并且提出一个完整的产品设计。

**Estimate Product Size and Resources.** The correlation of program size with development time is only moderately good for engineering teams and organizations. However, for individual engineers, the correlation is generally quite high. Therefore, the PSP starts with engineers estimating the sizes of the products they will personally develop. Then, based on their personal size and productivity data, the engineers estimate the time required to do the work. In the PSP, these size and resource estimates are made with the PROBE method.

**估算产品规模和资源** 程序规模与开发时间的关联，对于工程团队及组织来说仅属适中。然而，对个体工程师，相关程度一般来说相当高，因此，PSP 从工程师估算自己开发的产品规模开始，接着，依据个人规模和生产率数据，工程师估算施工所需时间，在 PSP 中，这些规模和资源估算使用 PROBE 方法进行。

## 4.2 使用 PROBE 规模估算

## Size Estimating with PROBE

PROBE stands for ***PRO***xy ***B***ased ***E***stimating and it uses proxies or objects as the basis for estimating the likely size of a product [Humphrey 95]. With PROBE, engineers first determine the objects required to build the product described by the conceptual design. Then they determine the likely type and number of methods for each object. They refer to historical data on the sizes of similar objects they have previously developed and use linear regression to determine the likely overall size of the finished product. The example object size data in Table 3 show the five size ranges the PSP uses for objects. Since object size is a function of programming style, the PROBE method shows engineers how to use the data on the programs they have personally developed to generate size ranges for their personal use. Once they have estimated the sizes of the objects, they used linear regression to estimate the total amount of code they plan to develop. To use linear regression, the engineers must have historical data on estimated versus actual program size for at least three prior programs.

PROB 代表以委托方式估算，是使用代理或者对象作为类似产品规模估算的基础[Humphrey 95]。使用 PROBE，工程师首先决定以概念设计描述的被构建产品所需的对象。然后决定每个对象可能采用的类型和方法的数量。工程师参考其先前开发过的、相似对象的历史规模数据，并以线性回归法计算出完工之产品整个可能的规模。在表 3 的，对象规模范

例数据中，列出了 PSP 用于对象的可能 5 种规模范围。因为对象规模是一个程序设计风格的功能/函数，PROBE 方法指导工程师如何使用数据－在其亲自开发来产生供他个人使用的规模范围的程序上。一旦工程师估算了对象的规模，就可以使用线性回归来估算其计划开发的代码的总数量。要使用线性回归，工程师必须至少要有 3 个先前程序估算 VS.实际程序规模的历史数据。

## 4.3 计算

## Calculation

*Table 3: C++ Object Size Data*[1]

*表 3：C++ 对象规模数据*

| C++ Object Sizes in LOC per Method 每种方法中 C++以 LOC 计算的对象规模 | | | | | |
|---|---|---|---|---|---|
| **Category** | **Very Small** | **Small** | **Medium** | **Large** | **Very Large** |
| **类别** | **极小** | **小** | **中等** | **大** | **很大** |
| Calculation 计算 | 2.34 | 5.13 | 11.25 | 24.66 | 54.04 |
| Data 数据 | 2.60 | 4.79 | 8.84 | 16.31 | 30.09 |
| I/O | 9.01 | 12.06 | 16.15 | 21.62 | 28.93 |
| Logic 逻辑 | 7.55 | 10.98 | 15.98 | 23.25 | 33.83 |
| Set-up 安装 | 3.88 | 5.04 | 6.56 | 8.53 | 11.09 |
| Text 文本 | 3.75 | 8.00 | 17.07 | 36.41 | 77.66 |

[1] Reprinted from A Discipline for Software Engineering, page 117, by Watts S. Humphrey, Addison Wesley Publishing Co., 1995, Permission Required.

## 4.4 使用 PROBE 资源估算

## Resource Estimating with PROBE

The PROBE method also uses linear regression to estimate development resources. Again, this estimate is based on estimated size versus actual effort data from at least three prior projects. The data must demonstrate a reasonable correlation between program size and development time. The PSP requires that the $r^2$ for the correlation be at least 0.5.

PROBE 方法也使用线性回归来估算开发所需的资源。同样，估算依据至少 3 个先前项目得来的估算 VS.实际工作量数据。数据必须能够说明程序规模与开发时间之间合理的相关性。PSP 要求 $r^2$ 相关性至少为 0.5。

Once they have estimated the total time for the job, engineers use their historical data to estimate the time needed for each phase of the job. This is where the column To Date% in the project plan summary in Table 1 is used. To Date% keeps a running tally of the percentage of total development time that an engineer has spent in each development phase. Using these percentages as a guide, engineers allocate their estimated total development time to the planning, design, design review, code, code review, compile, unit test, and postmortem phases. When done, they have an estimate for the size of the program, the total development time, and the time required for each development phase.

一旦估算出工作的总时间，工程师要使用其历史数据来估算每个工作阶段所需要的时间。这就是表 1 项目计划摘要中使用的列「截至目前%」所填的数据。「截至目前%」维持工程师花费在每个开发阶段的总开发时间百分比连续记录。使用这些百分比作为指导，工程师分配其估算全部开发时间到计划、设计、设计评审、编码、编码评审、编译、单元测试和事后检查阶段。当完成后，就可以获得程序规模，总开发时间以及在每个阶段需要的时间的估算值。

**Produce the Schedule.** Once engineers know the time required for each process step,

they estimate the time they will spend on the job each day or week. With that information, they spread the task time over the available scheduled hours to produce the planned time for completing each task. For larger projects, the PSP also introduces the earned-value method for scheduling and tracking the work [Boehm 81, Humphrey 95].

**产生日程** 一旦工程师知道每个过程阶段需要的时间，他们估算其将要花费在工作上的每天或者每周时间。根据这些信息，工程师在可用日程小时中配置任务时间，以产生完成每项任务的预计时间。对大型的项目，PSP 也引入挣值分析法来调度和跟踪工作。

**Develop the Product.** In the step called Develop the Product, the engineers do the actual Programming work. While this work is not normally considered part of the planning process, the engineers must use the data from this process to make future plans.

**开发产品** 在名为开发产品的步骤中，工程师执行实际的编程工作，这一个工作通常不被考虑为计划过程的一部分，工程师必须使用此过程的数据来做将来的计划。

**Analyze the Process.** After completing a job, the engineers do a postmortem analysis of the work. In the postmortem, they update the project plan summary with actual data, calculate any required quality or other performance data, and review how well they performed against the plan. As a final planning step, the engineers update their historical size and productivity databases. At this time, they also examine any process improvement proposals (PIPs) and make process adjustments. They also review the defects found in compiling and testing, and update their personal review checklists to help them find and fix similar defects in the future.

**分析过程** 在完成一个工作后，工程师进行事后检查分析工作。在事后检查中，工程师要用实际数据更新项目计划摘要，计算任何必要的质量或者其它执行数据，并且评审他们按照计划执行得有多好。作为计划的最后一步，工程师更新其历史数据和生产率数据库，此时，工程师也检查任何过程改建建议(PIP)并且做过程调整,也评审在执行和测试时发现的缺陷，并且更新他们个人评审检查表以便将来发现和修复类似的缺陷。

# 5.PSP 数据收集

# PSP Data Gathering

In the PSP, engineers use data to monitor their work and to help them make better plans. To do this, they gather data on the time that they spend in each process phase, the sizes of the products they produce, and the quality of these products. These topics are discussed in the following sections.

在 PSP 中。工程师利用数据监控他们的工作并帮助他们做更好的计划。要做到这一点，工程师搜集自己在每个过程阶段花费的时间，生产产品的规模以及这些产品的质量的数据。以下几个章节讨论这些主题。

## 5.1 时间测量

## Time Measures

In the PSP, engineers use the time recording log to measure the time spent in each process phase. In this log, they note the time they started working on a task, the time when they stopped the task, and any interruption time. For example, an interruption would be a phone call, a brief break, or someone interrupting to ask a question. By tracking time precisely,

engineers track the effort actually spent on the project tasks. Since interruption time is essentially random, ignoring these times would add a large random error into the time data and reduce estimating accuracy.

在 PSP 中，工程师使用时间记录日志来测量在每个过程阶段花费的时间。在这个日志中，工程师记下任务开始时间，停止任务的时间以及任何中断时间。例如，中断可能因为打电话。短暂的休息或者被打断来回答某人问题。通过精确跟踪时间，工程师跟踪实际花费在项目任务上的工作量。由于中断时间基本上是很随机的，忽略这些时间，会在时间数据中增加大量的随机的误差，并且削弱估算的精确性。

## 5.2 规模测量

## Size Measures

Since the time it takes to develop a product is largely determined by the size of that product, when using the PSP, engineers first estimate the sizes of the products they plan to develop. Then, when they are done, they measure the sizes of the products they produced. This provides the engineers with the size data they need to make accurate size estimates. However, for these data to be useful, the size measure must correlate with the development time for the product. While lines of code (LOC) is the principal PSP size measure, any size measure can be used that provides a reasonable correlation between development time and product size. It should also permit automated measurement of actual product size.

由于用在开发产品上的时间很大程度上由产品的规模决定，当使用 PSP 的时候，工程师首先估算计划要开发的产品的规模。然后，当这些完成以后，他们测量他们生产的产品规模。这提供了工程师需要做准确规模估算的规模数据。然而，要让这些成为有用的数据，产品规模必须与开发时间相关联。虽然程序代码行(LOC)是 PSP 主要的规模测量方法，然而，只要能提供在开发时间和产品规模上的合理关联，任何规模测量都可以使用。此方法应当能自动测量产品的实际规模。

### 5.2.1 代码行

**Lines of Code (LOC)**

The PSP uses the term “logical LOC” to refer to a logical construct of the programming language being used. Since there are many ways to define logical LOC, engineers must precisely define how they intend to measure LOC [Park 92]. When engineers work on a team or in a larger software organization, they should use the team’s or organization’s LOC standard. If there is no such standard, the PSP guides the engineers in defining their own. Since the PSP requires that engineers measure the sizes of the programs they produce, and since manually counting program size is both time consuming and inaccurate, the PSP also guides engineers in writing two automated LOC counters for use with the PSP course.

PSP 使用术语“逻辑 LOC”是指所使用编程语言的逻辑构造。由于定义逻辑构想的方法有许多种，工程师必须精确定义出如何测量他们想要测量的 LOC。当参与项目团队或在大型软件组织中工作时，工程师应遵循项目团队或组织的 LOC 标准。如果没有这类的标准，PSP 会指导工程师定出自己的 LOC 标准。由于 PSP 要求工程师测量自己生产程序的规模，又因为以人工方式来数程序的规模既耗时又不精确，PSP 也在为使用 PSP 课程编写两个自动化的 LOC 计数器方面指导工程师。

### 5.2.2 规模类别

**Size Categories**

To track how the size of a program is changed during development, it is important to consider various categories of product LOC. These categories are

为了追踪开发期间程序规模的变更，考虑产品 LOC 的种类别相当的重要。这些类别包括：

- **Base.** When an existing product is enhanced, base LOC is the size of the original product version before any modifications are made.
- 基础 当现有产品被增强后，基础 LOC 为修改前源产品版本的规模。
- **Added.** The added code is that code written for a new program or added to an existing base program.
- 新增 新增程序代码，是指为程序所编写或加入现存基础程序中的程序代码。
- **Modified.** The modified LOC is that base code in an existing program that is changed.
- 修改 修改后之 LOC 是已变更现存程序中的基础程序代码。
- **Deleted.** The deleted LOC is that base code in an existing program that is deleted.
- 删除 删除后之 LOC 是已删除现存程序中的基础程序代码。
- **New and Changed.** When engineers develop software, it takes them much more time to add or modify a LOC than it does to delete or reuse one. Thus, in the PSP, engineers use only the added or modified code to make size and resource estimates. This code is called the New and Changed LOC.
- 新建与变更 在开发软件的时候，工程师新建或修改一个代码行会比删除或重用一个代码行花费更多的时间。因此， 在 PSP 中， 工程师会只采用新建或修改的程序代码来做规模及资源的估算。此代码称为新建与变更 LOC。
- **Reused.** In the PSP, the reused LOC is the code that is taken from a reuse library and used, without modification, in a new program or program version. Reuse does not count the unmodified base code retained from a prior program version and it does not count any code that is reused with modifications.
- 重用 在 PSP 中， 重用 LOC 是把代码从重用库中取出并用于新程序或者新程序版本中，不经过修代。重用不计算从先前程序版本保留的未经修改的基础代码，而且不计算任何经过修改重用的代码。
- **New reuse.** The new reuse measure counts the LOC that an engineer develops and contributes to the reuse library.
- 新重用 新重用测量计算工程师开发并提供给重用库的 LOC。
- **Total.** The total LOC is the total size of a program, regardless of the source of the code.
- 总计 LOC 总数是是忽略代码来源的程序总规模。

### 5.2.3 规模纪录

**Size Accounting**

When modifying programs, it is often necessary to track the changes made to the original program. These data are used to determine the volume of product developed, the

engineer's productivity, and product quality. To provide these data, the PSP uses the size accounting method to track all the additions, deletions, and changes made to a program [Humphrey 95].

在修改程序期间，常常需要跟踪对原始程序所做的变更。这些数据用于决定所开发产品的产量、工程师的生产率、以及产品质量。要提供这些数据，PSP 采用规模纪录法来追踪所有对程序所做的增添、删除、以及变更。

To use size accounting, engineers need data on the amount of code in each size category. For example, if a product of 100,000 LOC were used to develop a new version, and there were 12,000 LOC of deleted code, 23,000 LOC of added code, 5,000 LOC of modified code, and 3,000 LOC of reused code, the New and changed LOC would be

N&C LOC = Added + Modified

28,000 = 23,000 + 5,000

要使用规模纪录法，工程师需要每种规模类别的代码数量的数据。例如，100,000LOC 的产品被用于开发一个新版本，同时删除 12,000LOC 的代码、增添 23,000LOC 代码、修改 5,000LOC 代码、以及重用 3,000LOC 代码，则新建与变更 LOC 将会是：

N&C LOC = Added + Modified

N&C LOC = 新增 + 修改

28,000 = 23,000 + 5,000

When measuring the total size of a product, the calculations are as follows:

Total LOC = Base – Deleted + Added + Reused

在测量产品总规模时，计算如下：

总 LOC = 基础 – 删除 + 新增 + 重用

Neither the modified nor the "new reuse" LOC are included in the total. This is because a modified LOC can be represented by a deleted and an added LOC, and the "new reuse" LOC are already accounted for in the added LOC. Using this formula, the total LOC for the above example would be

Total = 100,000 – 12,000 + 23,000 + 3,000 = 114,000 LOC

修改与"新重用"LOC 两者都不包含在总计之内。这是因为修改 LOC 可以用删除与新增 LOC 来表示，而"新重用"LOC 已经计算在新增 LOC 之中，使用此公式，上例中的总 LOC，将会是：

总计 = 100,000 – 12,000 + 23,000 + 3,000 = 114,000 LOC

This is 114 KLOC, where KLOC stands for 1,000 LOC.

即 114 KLOC ，其中 KLOC 表示 1,000 LOC 。

## 5.3 质量测量

## Quality Measures

The principal quality focus of the PSP is on defects. To manage defects, engineers need data on the defects they inject, the phases in which they injected them, the phases in which

they found and fixed them, and how long it took to fix them. With the PSP, engineers record data on every defect found in every phase, including reviews, inspections, compiling, and testing.

PSP 的质量焦点主要置于缺陷之上。为了管理缺陷，工程师需要有关自己所注入的缺陷的数据，自己注入缺陷的阶段、自己发现及修正这些缺陷的阶段、以及修正缺陷所需要的时间。使用 PSP，工程师记录在每个阶段包括审查、检视、编译、与测试中每个所发现缺陷的数据。

These data are recorded in the defect recording log.

这些数据要记在缺陷记录日志上。

The term *defect* refers to something that is wrong with a program. It could be a misspelling, a punctuation mistake, or an incorrect program statement. Defects can be in programs, in designs, or even in the requirements, specifications, or other documentation. Defects can be found in any process phase, and they can be redundant or extra statements, incorrect statements, or omitted program sections. A defect, in fact, is anything that detracts from the program's ability to completely and effectively meet the user's needs. Thus a defect is an objective thing. It is something that engineers can identify, describe, and count.

术语*缺陷*指程序中的错误。可能是拼写错误、标点符号错误、或者不正确的程序声明。缺陷可能在程序中、设计上、或者甚至是在需求、规格说明书或者其它文档中。缺陷在过程的任何阶段都可以发现，并且缺陷可能是冗余或额外的程序声明、错误声明、或是忽略的程序区段。事实上，缺陷就是任何降低程序完全和有效的满足用户需要的能力。因此，缺陷是客观的东西，是工程师可以辨识、描述并计数的东西。

With size, time, and defect data, there are many ways to measure, evaluate, and manage the quality of a program. The PSP provides a set of quality measures that helps engineers examine the quality of their programs from several perspectives. While no single measure can adequately indicate the overall quality of a program, the aggregate picture provided by the full set of PSP measures is a generally reliable quality indicator. The principal PSP quality measures are

有了规模、时间与缺陷数据，就有很多方法对程序质量进行测量、评估和管理。PSP 提供了一套质量测量用于协助工程师从几个角度来检查其程序的质量。尽管没有单一的测量项目可以充分指出整个程序的质量，整套 PSP 测量提供的总体视图是一般可信赖的质量指数。PSP 的主要质量测量如下：

- Defect density 缺陷密度
- Review rate 评审率
- Development time ratios 开发时间比率
- Defect ratios 缺陷比率
- Yield 合格率
- Defects per hour 每小时的缺陷数
- Defect removal leverage 缺陷移除有效率

●Appraisal to failure ratio (A/FR) 失效比率估计

Each of these measures is described in the following paragraphs.

每一个测量将在以下的段落中描述。

**Defect Density.** Defect density refers to the defects per new and changed KLOC found in a program. Thus, if a 150 LOC program had 18 defects, the defect density would be

**缺陷密度** 缺陷密度是指在程序中每新增与变更 KLOC 发现的缺陷数。因此，如果一个有 150LOC 的程序有 18 个缺陷，缺陷密度将会是

1000*18/150 = 120 defects/KLOC

Defect density is measured for the entire development process and for specific process phases. Since testing only removes a fraction of the defects in a product, when there are more defects that enter a test phase, there will be more remaining after the test phase is completed.

缺陷密度会在整个开发过程与指定的过程阶段中测量。由于测试仅能移除产品中一部分的缺陷，其间，会有更多的缺陷进到测试阶段中，在测试阶段完成后，会留下更多的缺陷。

Therefore, the number of defects found in a test phase is a good indication of the number that remains in the product after that test phase is completed.

因此，测试阶段所发现的缺陷数，是测试阶段完成后，残留在产品中缺陷数极好的指标。

Figure 4 shows IBM data on a major product that demonstrates this relationship [Kaplan 94]. These data imply that when engineers find relatively fewer defects in unit test, assuming that they have performed a competent test, their programs are of relatively higher quality. In the PSP, a program with five or fewer defects/KLOC in unit test is considered to be of good quality. For engineers who have not been PSP trained typical unit test defect levels range from 20 to 40 or more defects/KLOC.

图 4 所示的 IBM 在主要产品上的数据展示了这种关系。这些数据意味着在单元测试中工程师发现的缺陷数相对较少，则可以假定其已完成充分的测试，他们的程序具有相对较高的质量。在 PSP 中，在单元测试时程序中每 KLOC 有少于五个缺陷，则可以被认为有良好的质量。对于未接受过 PSP 训练的工程师来说，在单元测试中缺陷的典型水平会介于每 KLOC 中 20 至 40 个或更多缺陷。

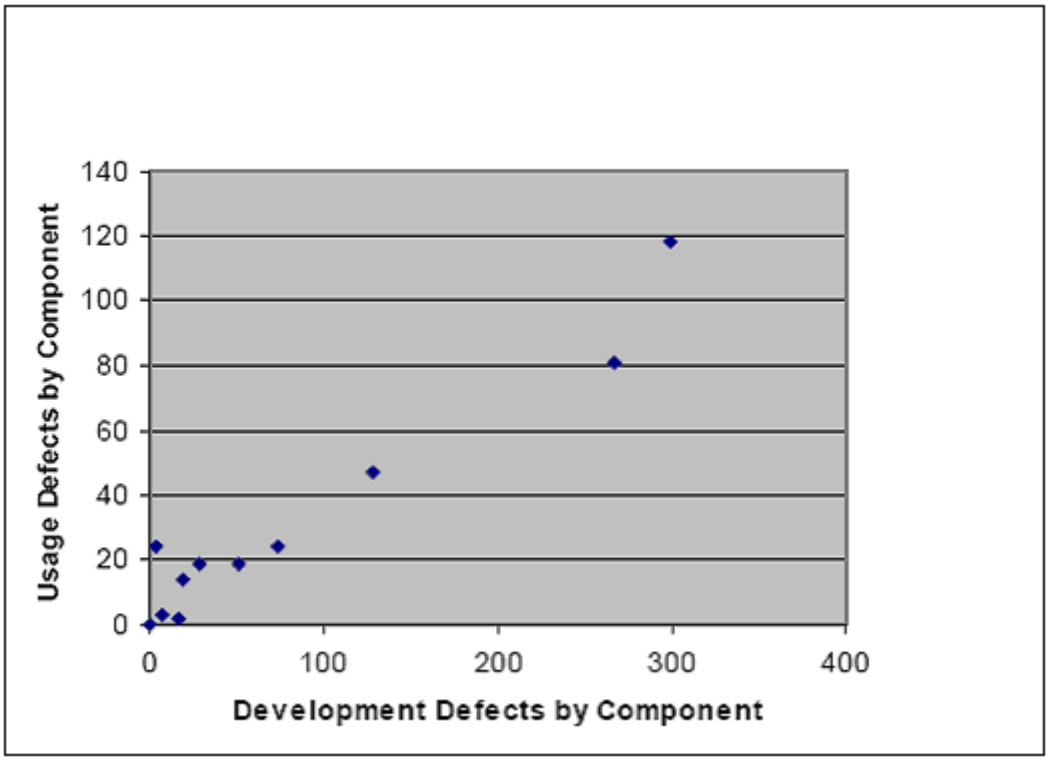



*Figure 4: Development vs. Usage Defects; IBM Release 1 ($r^2$=.9267)*

*图 4：开发 vs.使用缺陷；IBM 发行版 1(r2=.9267)*

**Review Rate.** In the PSP design and code reviews, engineers personally review their programs. The PSP data show that when engineers review designs or code faster than about 150 to 200 new and changed LOC per hour, they miss many defects. With the PSP, engineers gather data on their reviews and determine how fast they should personally review programs to find all or most of the defects.

**评审率** 在 PSP 的设计与程序代码评审中，工程师个人要评审自己的程序。PSP 的数据显示，当工程师评审设计或代码的速度高于每小时 150 至 200 个新建与变更 LOC 时，他们会遗漏许多缺陷。采用 PSP，工程师可以采集个人评审的数据并决定个人评审程序的速度以找出所有或大部分的缺陷。

**Development Time Ratios.** Development time ratios refer to the ratio of the time spent by an engineer in any two development phases. In the PSP, the three development time ratios used in process evaluation are design time to coding time, design review time to design time, and code review time to coding time.

**开发时间比率** 开发时间比率是指一工程师在任何两个开发阶段中所花费时间的比率。在 PSP 中，过程评估所使用的三项开发时间比率为：设计时间比编码时间、设计评审时间比设

计时间、及代码评审时间比编码时间。

The design time to coding time measure is most useful in determining the quality of an engineer's design work. When engineers spend less time in designing than in coding, they are producing most of the design while coding. This means that the design is not documented, the design cannot be reviewed or inspected, and design quality is probably poor. The PSP guideline is that engineers should spend at least as much time producing a detailed design as they do in producing the code from that design.

设计时间对编码时间测量对判断工程师设计工作质量特别有用。如果设计所花费的时间少于编码时间，意味着工程师在程序编码时绝大部分是在做设计的工作。也就是说，设计并未文档化，设计无法被评审或者审查，而且设计的质量可能很差。PSP 的指导原则是工程师进行详细设计所花费的时间，至少应和从设计进行编码所花费的时间一样多。

The suggested ratio for design review time to design time is based on the PSP data shown in Table 4. In the PSP courses, engineers injected an average of 1.76 defects per hour in detailed design and found an average of 2.96 defects per hour in design reviews. Thus, to find all the defects that they injected during the design phase, engineers should spend 59% as much time in the design review as they did in the design. The PSP guideline is that engineers should spend at least half as much time reviewing a design as they did producing it.

设计评审时间对设计时间的建议比率基于表 4 所示的 PSP 数据。在 PSP 的课程中，工程师在详细设计时平均每小时注入的缺陷为 1.76 个，在设计评审时平均每小时发现的缺陷为 2.96 个。因此，要想找出在设计阶段注入的所有缺陷，工程师应花比在设计上多 59%的时间从事设计评审工作。PSP 的指导原则是工程师应至少花比设计多一半的时间用于设计评审。

*Table 4: PSP Defect Injection and Removal Data*

*表 4：PSP 缺陷注入与移除的数据*

| Phase 阶段 | Injected/Hour 注入/小时 | Removed/Hour 移除/小时 |
|---|---|---|
| Design 设计 | 1.76 | 0.10 |
| Design Review 设计评审 | 0.11 | 2.96 |
| Coding 程序编码 | 4.20 | 0.51 |
| Code Review 代码评审 | 0.11 | 6.52 |
| Compile 编译 | 0.60 | 15.84 |
| Unit Test 单元测试 | 0.38 | 2.21 |

Note: These data are from PSP courses where the exercises produced 2,386 programs with 308,023 LOC, 15,534 development hours, and 22,644 defects.

注：这些数据来自 PSP 课程，课程中练习共产生 2,386 个程序，共有 308,023LOC、15,534 个开发小时，以及 22,644 个缺陷。

The code review time to coding time ratio is also a useful measure of process quality. During coding, engineers injected an average of 4.20 defects per hour and they found an

average of 6.52 defects per hour in code reviews. Therefore, based on the PSP data in Table 4, engineers should spend about 65% as much time in code reviews as they do in coding. The PSP uses 50% as an approximate guideline for the code review time to coding time ratio.

程序代码评审时间对编码时间比率也是对过程质量有用的测量。在编码期间，工程师平均每小时注入 4.20 个缺陷并且在代码评审的期间，平均每小时发现 6.52 个缺陷。因此，基于表 4 所列的 PSP 数据，工程师在代码评审期间应比编码期间多花 65%的时间。PSP 使用 50%作为代码评审对编码时间比率的近似指导。

**Defect Ratios.** The PSP defect ratios compare the defects found in one phase to those found in another. The principal defect ratios are defects found in code review divided by defects found in compile, and defects found in design review divided by defects found in unit test. A reasonable rule of thumb is that engineers should find at least twice as many defects when reviewing the code as they find in compiling it. The number of defects found while compiling is an objective measure of code quality. When engineers find more than twice as many defects in the code review as in compiling, it generally means that they have done a competent code review or that they did not record all the compile defects. The PSP data also suggest that the design review to unit test defect ratio be two or greater. If engineers find twice as many defects during design review as in unit test, they have probably done acceptable design reviews.

**缺陷比率** PSP 缺陷比率是将某阶段所发现的缺陷与另一个阶段所发现的缺陷做比较。主要缺陷比率是将代码评审所发现缺陷除以编译时所发现的缺陷，以及设计评审所发现的缺陷除以单元测试所发现的缺陷。一个合理的经验法则是工程师在代码评审时所发现缺陷应当至少为编译时所发现缺陷的两倍。编译期间所发现的缺陷数是代码质量的客观测量值。当工程师在代码中所发现的缺陷数超过编译期间所发现缺陷数的两倍以上时，通常意味着工程师已充分完成的代码评审或者没有记录全部编译缺陷。PSP 数据也建议设计评审对单元测试缺陷比率为两倍或更大。如果工程师发现比单元测试时多两倍的设计评审缺陷，那么他们也许做了能被接受的设计评审。

The PSP uses these two measures in combination with time ratios and defect density measures to indicate whether or not a program is ready for integration and system testing. The code review to compile defect ratio indicates code quality, and the design review to unit test defect ratio indicates design quality. If either measure is poor, the PSP quality criteria suggest that the program is likely to have problems in test or later use.

PSP 使用这两个测量，以及和时间比率与缺陷密度测量一起，来指出程序进行集成与系统测试是否就绪。代码评审对编译缺陷比率显示代码的质量，而设计评审对单元测试缺陷比率显示设计的质量。如果其中任一项测量很差，PSP 质量标准认为程序在测试中或其后的使用期间可能会存在问题。

**Yield.** In the PSP, yield is measured in two ways. Phase yield measures the percentage of the total defects that are found and removed in a phase. For example, if a program entered unit test with 20 defects and unit testing found 9, the unit test phase yield would be 45%. Similarly, if a program entered code review with 50 defects and the review found 28, the code review phase yield would be 56%. Process yield refers to the percentage of the defects removed before the first compile and unit test. Since the PSP objective is to produce high quality programs, the suggested guideline for process yield is 70% better.

**合格率** 在 PSP 中，合格率可以用两种方法来测量。阶段合格率测量一个阶段中所发现与移除之总缺陷数的百分比。例如，假设程序进入单元测试前有 20 个缺陷并且进行单元测试时发现了 9 个，则单元测试阶段的合格率会是 45%。同样地，假设程序进入代码评审前有 50 个缺陷并且经过评审发现了 28 个，则代码评审阶段的合格率为 56%。过程合格率是指在首次编译与单元测试前所移除缺陷数的百分比。由于 PSP 的目标为生产高质量的程序，因此，过程合格率的建议指标在 70%更好。

The yield measure cannot be precisely calculated until the end of a program's useful life. By then, presumably, all the defects would have been found and reported. When programs are of high quality, however, reasonably good early yield estimates can usually be made. For example, if a program had the defect data shown in Table 5, the PSP guideline for estimating the yield is to assume that the number of defects remaining in the program is equal to the number found in the last testing phase. Thus, in Table 5, it is likely that seven defects remain after unit testing.

合格率测量值不到程序使用生命结束是无法精确计算的。到了那个时候，大概所有的缺陷已经被发现并且报告了。然而，如果程序是高质量的，则合理而且好的早期合格率估算是可以做出的。例如，如果程序的缺陷数据如表 5 中所示，PSP 对于合格率估算的指导原则是，假设残留在程序中的缺陷数等于在最后测试阶段发现的缺陷数。因此，在表 5 中，在单元测试后还有 7 个缺陷残留在程序中。

*Table 5: Example Defects Removed*

*表 5：移除缺陷的范例*

| Phase 阶段 | Defects Removed 缺陷移除 |
|---|---|
| Design Review 设计评审 | 11 |
| Code Review 代码评审 | 28 |
| Compile 编译 | 12 |
| Unit Test 单元测试 | 7 |
| Total 总计 | 58 |

To calculate yield, engineers use the data on the phases in which the defects were injected. They also need to assume that the defects remaining after unit test were injected in the same phases as those found during unit test. The data in Table 6 include one such defect in coding and six in detailed design.

要计算合格率，工程师使用被注入缺陷的阶段的数据。他们也必须假设在单元测试后残留的缺陷和单元测试时发现的缺陷一样是在同一个阶段被注入的。表 6 中所列的数据包括编码阶段有 1 个详细设计阶段有 6 个这类的缺陷。

*Table 6: Example Defects Injected and Removed*

*表 6：缺陷注入与移除的范例*

| Phase<br>阶段 | Defects Injected<br>缺陷注入 | Defects Removed 缺陷移除 |
|---|---|---|
| Detailed Design 详细设计 | 26 | 0 |
| Design Review 设计评审 | 0 | 11 |
| Code 编码 | 39 | 0 |
| Code Review 代码评审 | 0 | 28 |
| Compile 编译 | 0 | 12 |
| Unit Test 单元测试 | 0 | 7 |
| After Unit Test 单元测试后 | 0 | 7 |
| Total 总计 | 65 | 65 |

As shown in Table 7, the total defects injected, including those estimated to remain, is 65. At design review entry, the defects present were 26, so the yield of the design review was 100*11/26 = 42.3%. At code review entry, the defects present were the total of 65 less those removed in design review, or 65 – 11 = 54, so the code review yield was 100*28/54 = 51.9%.

如表 7 所示，注入总缺陷数包括估计的残留数为 65。在设计评审入口，出现的缺陷数是 26，因此，设计评审的合格率就是 100*11/26=42.3% 。在代码评审入口，出现的缺陷数总计为 65 减去设计评审移除的缺陷数，或者 65-11=54 ，所以代码评审的合格率为 100*28/54=51.9% 。

Similarly, the compile yield was 100*12/(65-11-28) = 46.2%. Process yield is the best overall measure of process quality. It is the percentage of defects injected before compile that are removed before compile. For the data in Table 7, the process yield is 100*39/65 = 60.0%

.同样地，编译合格率是 100*12/(65-11-28)=46.2% 。过程合格率是过程质量最佳的整体测量，它是编译前移除缺陷除于编译前注入的缺陷的百分比。对于表 7 中的数据，过程合格率为 100*39/65=60.0%。

*Table 7: Yield Calculations*

*表 7：合格率计算*

| Phase<br>阶段 | Defects Injected<br>缺陷数注入 | Defects Removed<br>缺陷移除 | Defects at Phase Entry 阶段入口缺陷 | Phase Yield 阶段合格率 |
|---|---|---|---|---|
| Detailed Design 详细设计 | 26 | 0 | 0 | |

| | | | | |
|---|---|---|---|---|
| Design Review 设计评审 | 0 | 11 | 26 | 42.3% |
| Code 编码 | 39 | 0 | 15 | |
| Code Review 代码评审 | 0 | 28 | 54 | 51.9% |
| Compile 编译 | 0 | 12 | 26 | 46.2% |
| Unit Test 单元测试 | 0 | 7 | 14 | 50.0% |
| After Unit Test 单元测试后 | 0 | 7 | 7 | |
| Total 总计 | 65 | 65 | | |

**Defects per Hour.** With the PSP data, engineers can calculate the numbers of defects they inject and remove per hour. They can then use this measure to guide their personal planning. For example, if an engineer injected four defects per hour in coding and fixed eight defects per hour in code reviews, he or she would need to spend about 30 minutes in code reviews for every hour spent in coding. It would take this long to find and fix the defects that were most likely injected. The defects per hour rates can be calculated from the data in the project plan summary form.

**每小时的缺陷数** 使用 PSP 数据，工程师可以计算出其每个小时注入与移除的缺陷数。然后，他们可以使用此测量指导其个人计划。例如，如果在编码阶段工程师每小时注入 4 个缺陷，在代码评审时每小时修正 8 个缺陷，他/她每花费在编码上 1 小时需要花费大约 30 分钟代码评审。这将花费这么长的时间去找出并修复可能注入的缺陷。每小时缺陷率可从项目计划摘要表格数据中计算出来。

**Defect Removal Leverage (DRL).** Defect removal leverage measures the relative effectiveness of two defect removal phases. For instance, from the previous examples, the defect removal leverage for design reviews over unit test is 3.06/1.71 = 1.79. This means that the engineer will be 1.79 times more effective at finding defects in design reviews as in unit testing.

**缺陷移除有效率** 缺陷移除有效率测量两个缺陷移除阶段的相对有效性。例如，依前例，设计评审对单元测试缺陷移除有效率为 3.06/1.71=1.79。这意味着工程师在设计评审时发现缺陷的有效性比单元测试时发现缺陷的有效性多 1.79 倍。

The DRL measure helps engineers design the most effective defect removal plan.

DRL 测量帮助工程师设计出更有效的缺陷移除计划。

A/FR. The appraisal to failure ratio (A/FR) measures the quality of the engineering process, using cost-of-quality parameters [Juran 88]. The *A* stands for the appraisal quality cost, or the percentage of development time spent in quality appraisal activities. In PSP, the appraisal cost is the time spent in design and code reviews, including the time spent repairing the defects found in those reviews.

A/FR 失效比率估计使用质量成本参数测量工程过程的质量。*A* 代表估计质量成本，或是

开发时间用于质量评估活动的百分比。在 PSP 中，评估成本是花费在设计与代码评审上的时间，包括修复在这些评审上所发现缺陷的时间。

The *F* in A/FR stands for the failure quality cost, which is the time spent in failure recovery and repair. The failure cost is the time spent in compile and unit test, including the time spent finding, fixing, recompiling, and retesting the defects found in compiling and testing.

A/FR 中的 F 代表失效质量成本，是故障排除与修复所花费的时间。失效成本为编译与单元测试所花费的时间，包括在编译与测试时所发现，修复、重新编译与重新测试缺陷所花费的时间。

The A/FR measure provides a useful way to assess quality, both for individual programs and to compare the quality of the development processes used for several programs. It also indicates the degree to which the engineer attempted to find and fix defects early in the development process. In the PSP course, engineers are told to plan for A/FR values of 2.0 or higher. This ensures that they plan adequate time for design and code reviews.

A/FR 测量提供了一个相当有用的方法来评估质量，不管对个别的程序，还是为几个程序使用的开发过程质量比较都一样。A/FR 也指出在开发过程早期工程师试图查找和修复缺陷的程度。在 PSP 课程中，工程师被告知计划的 A/FR 值为 2.0 或更高。这样可以确保其计划充分的时间用于设计和代码评审。

## 6.PSP 质量管理

## PSP Quality Management

Software quality is becoming increasingly important and defective software is an increasing problem [Leveson 95]. Any defect in a small part of a large program could potentially cause serious problems. As systems become faster, increasingly complex, and more automatic, catastrophic failures are increasingly likely and potentially more damaging [Perrow 84]. The problem is that the quality of large programs depends on the quality of the smaller parts of which they are built. Thus, to produce high-quality large programs, every software engineer who develops one or more of the system's parts must do high-quality work. This means that all engineers must manage the quality of their personal work. To help them do this, the PSP guides engineers in tracking and managing every defect.

软件质量变得愈来愈重要而有缺陷的软件也是一个日益增长的问题。大型程序中的一小块有任何的缺陷都很有可能导致严重的问题。当系统变得愈来愈快、愈来愈复杂、愈来愈自动化，灾难性故障愈来愈可能发生而潜在的破坏性也愈来愈大。问题在于大型程序的质量与构成它的较小部分的质量息息相关。因此，要生产高质量的大型程序，每个开发一个或更多系统子部分的软件工程师必须进行高质量工作。这意味着，所有的工程师必须管理好其个人工作的质量。为帮助工程师们完成这些，PSP 指导工程师跟踪和管理每一个缺陷。

## 6.1 缺陷和质量

## Defects and Quality
## 缺陷与质量

Quality software products must meet the users' functional needs and perform reliably and consistently. While software functionality is most important to the program's users, the functionality is not usable unless the software runs. To get the software to run, however, engineers must remove almost all of its defects. Thus, while there are many aspects to software quality, the engineer's first quality concern must necessarily be on finding and fixing defects. PSP data show that even experienced programmers inject a defect in every seven to ten lines of code [Hayes 97]. Although engineers typically find and fix most of these defects when compiling and unit testing programs, traditional software methods leave many defects in the finished product.

具有质量的软件产品必须满足用户在功能性需求而且可靠并持续运行。当软件功能性对于程序的使用者来说是最重要的时候，除非软件在运行否则功能性是没有用的。然而，为了使软件能够运行，工程师必须移除软件几乎所有的缺陷。因此，尽管软件质量有许多个方面，工程师关心的首要问题必定是查找和修复缺陷。PSP 的数据显示即便是富有经验的工程师也会每 7 到 10 行代码注入一个缺陷。虽然，在程序编译与单元测试时工程师通常会找出并修复大部分的这类缺陷，但是传统的软件方法仍然在已完成的产品中留下了许多的缺陷。

Simple coding mistakes can produce very destructive or hard-to-find defects. Conversely, many sophisticated design defects are often easy to find. The mistake and its consequences are largely independent. Even trivial implementation errors can cause serious system problems. This is particularly important since the source of most software defects is simple programmer oversights and mistakes. While design issues are always important, newly developed programs typically have few design defects compared to the large number of simple mistakes. To improve program quality, PSP training shows engineers how to track and manage all of the defects they find in their programs.

简单的编码错误会产生非常具有破坏性或难以发现的缺陷。相反的，许多复杂的设计缺陷常常可以很容易地找出来。错误及其后果很大程度上是相互独立的，即使是微不足道的执行错误也可能引发严重的系统问题。这点特别重要因为绝大多数的软件缺陷来源于单纯的程序员监控与错误。虽然设计问题一直很重要，但最近开发的程序与大量的单纯错误相比设计缺陷少很多。为了改善程序的质量，PSP 训练显示工程师如何跟踪与管理所有在其程序中找到的缺陷。

## 6.2 工程师的职责

## The Engineer's Responsibility

The first PSP quality principle is that engineers are personally responsible for the quality of the programs they produce. Because the software engineer who writes a program is most

familiar with it, that engineer can most efficiently and effectively find, fix, and prevent its defects. The PSP provides a series of practices and measures to help engineers assess the quality of the programs they produce and to guide them in finding and fixing all program defects as quickly as possible. In addition to quality measurement and tracking, the PSP quality management methods are early defect removal and defect prevention.

PSP 首项质量原则是工程师个人对其所生产的程序的质量亲自负责。因为编写程序的软件工程师对该程序非常熟悉，所以，工程师可以最有效率和有效地发现、修复及预防程序的缺陷。PSP 提供了一系列的实践与测量以协助工程师评估自己所生产的程序的质量，并且指导工程师尽快地查找并修复程序的所有缺陷。除了质量的测量和跟踪外，PSP 质量管理方法是早期移除和预防缺陷。

## 6.3 早期缺陷移除

## Early Defect Removal

The principal PSP quality objective is to find and fix defects before the first compile or unit test. The PSP process includes design and code review steps in which engineers personally review their work products before they are inspected, compiled, or tested. The principle behind the PSP review process is that people tend to make the same mistakes repeatedly. Therefore, by analyzing data on the defects they have made and constructing a checklist of the actions needed to find those mistakes, engineers can find and fix defects most efficiently.

PSP 主要的质量目标是在第一次编译或单元测试前找出并修复缺陷。PSP 的过程包括：在工作产品审查、编译或测试之前工程师亲自对其进行评审的设计与代码评审步骤。PSP 评审过程背后的原理是人们有重复地犯同样错误的倾向。因此，分析他们造成的缺陷上的数据，并且构建一个查找这些错误所需采取的行动的检查表，工程师可以更有效的查找和修复缺陷。PSP-trained engineers find an average of 6.52 defects per hour in personal code reviews and 2.96 defects per hour in personal design reviews. This compares with 2.21 defects found per hour in unit testing [Humphrey 98]. By using PSP data, engineers can both save time and improve product quality. For example, from average PSP data, the time to remove 100 defects in unit testing is 45 hours while the time to find that number of defects in code reviews is only 15 hours.

受过 PSP 训练的工程师在个人的代码评审中平均每小时发现 6.52 个缺陷，在个人设计评审中每小时 2.96 个缺陷。比较一下单元测试中每小时发现 2.21 个缺陷。通过使用 PSP 的数据，工程师既省时又可改善产品质量。例如，根据 PSP 的平均数据，在单元测试中移除 100 个缺陷的时间为 45 个小时，而在代码评审时找到 100 个缺陷的时间只需要 15 个小时。

## 6.4 缺陷预防

## Defect Prevention

The most effective way to manage defects is to prevent their initial introduction. In the PSP,

there are three different but mutually supportive ways to prevent defects. The first is to have engineers record data on each defect they find and fix. Then they review these data to determine what caused the defects and to make process changes to eliminate these causes. By measuring their defects, engineers are more aware of their mistakes, they are more sensitive to their consequences, and they have the data needed to avoid making the same mistakes in the future. The rapid initial decline in total defects during the first few PSP course programs indicates the effectiveness of this prevention method.

最有效的管理缺陷方法就是预防缺陷的初始导入。在 PSP 中，共有三种不同但相互支持的方法预防缺陷。第一个方法是工程师记录其发现和修复的每个缺陷的数据。然后，工程师评审这些数据以判断引起这些缺陷的原因是什么以及变更过程来根除这些原因。通过测量他们的缺陷，工程师更了解他们的错误，他们对缺陷造成的后果更加敏感，并且，他们拥有了在将来避免犯同样错误的所需数据。在首次少数 PSP 课程程序期间指出这种预防方法的有效性后，总体缺陷数首次快速下降。

The second prevention approach is to use an effective design method and notation to produce complete designs. To completely record a design, engineers must thoroughly understand it. This not only produces better designs; it results in fewer design mistakes.

第二个预防方法是使用有效的设计方法及标志来产生完整的设计。为了纪录完整的设计，工程师必须对此充分了解。这样不但能产出更好的设计；也引起更少的设计错误产生。

The third defect prevention method is a direct consequence of the second: with a more thorough design, coding time is reduced, thus reducing defect injection. PSP data for 298 experienced engineers show the potential quality impact of a good design. These data show that during design, engineers inject an average of 1.76 defects per hour, while during coding they inject 4.20 defects per hour. Since it takes less time to code a completely documented design, by producing a thorough design, engineers will correspondingly reduce their coding time. So, by producing thorough designs, engineers actually inject fewer coding defects [Humphrey 98].

第三个缺陷预防方法是第二个方法的直接结果：由于更彻底的设计，编码的时间会缩短，因此减少了缺陷的注入。对 298 个经验丰富工程师的 PSP 数据显示好的设计影响潜在的质量。这些数据表明在设计期间，工程数注入平均每小时 1.76 个缺陷，同时在编码期间每小时注入 4.20 个缺陷。由于将一个完全文档化的设计进行编码所花费的时间比少，通过产出完整的设计，工程师可以相对减少他们的编码时间。所以，通过产完整的设计，工程师实际上注入更少的编码缺陷。

## 7.PSP 设计

## PSP Design

While the PSP can be used with any design method, it requires that the design be complete.

The PSP considers a design to be complete when it defines all four of the dimensions shown in the object specification structure shown in Table 8 [De Champeaux 93]. The way that these four templates correspond to the object specification structure is shown in Table 9. The PSP has four design templates that address these dimensions.

虽然在 PSP 中被任何的设计方法使用，但还需要设计是完整的。PSP 认为只有当设计定义了表 8 中对象规格结构显示的 4 个维度时才算设计完成。与对象规格结构的相符合的 4 个模板如表 9 所示。PSP 有 4 个设计模板来说明这些维度。

*Table 8: Object Specification Structure*

*表 8：对象规格结构*

| **Object Specification 对象规格** | **Internal 内部** | **External 外部** |
|---|---|---|
| Static 静态 | Attributes Constraints 属性限制 | Inheritance Class Structure 继承类结构 |
| Dynamic 动态 | State Machine 状态机 | Services Messages 服务信息 |

*Table 9: PSP Template Structure*

*表 9：PSP 模板结构*

| **Object Specification Templates 对象规格模板** | **Internal 内部** | **External 外部** |
|---|---|---|
| Static 静态 | Logic Specification Template 逻辑规格模板 | Function Specification Template (Inheritance Class Structure) 功能规格模板(继承类结构) |
| Dynamic 动态 | State Machine Template 状态机模板 | Functional Specification Template (User Interaction) 功能规格模板(用户交互)<br>Operational Scenario Template 操作场景模板 |

- The *operational scenario template* defines how users interact with the system.
- *操作场景模板*定义用户如何与系统相互影响。
- The *function specification template* specifies the call-return behavior of the program's objects and classes as well as the inheritance class structure.
- *功能规格模板*规定程序对象和类以及继承类结构的调用返回状态。
- The *state specification template* defines the program's state machine behavior.
- *状态规格模板*定义程序的状态机行为。
- The *logic specification template* specifies the program's internal logic, normally in a pseudo code language.
- *逻辑规格模板*规定程序的内部逻辑，通常采用伪码语言。

For a complete description of these templates and how to use them, consult the reference [Humphrey 95].

有关于这些模板的完整说明和使用方法，请参阅参考。

# 8.PSP 规程

## PSP Discipline

A principal problem in any engineering field is getting engineers to consistently use the methods they have been taught. In the PSP, these methods are: following a defined process, planning the work, gathering data, and using these data to analyze and improve the process. While this sounds simple in concept, it is not easy in practice. This is why a principal focus of PSP introduction is on providing a working environment that supports PSP practices. This is the main reason that the SEI developed the Team Software Process: to provide the disciplined environment that engineers need to consistently use the PSP methods in practice.

在任何工程领域中主要的问题是使工程师持续贯彻使用他们被传授的理论。在 PSP 中，这些理论包括：遵循已定义的过程、计划工作、搜集数据并使用数据分析及改进过程。尽管在观念上虽然听起来简单，但执行起来不容易。这就是为什么 PSP 导入的焦点主要集中提供一个支持 PSP 实践的环境。这也是 SEI 开发团队软件过程的主要原因：提供一个工程师在实践中需要持续贯彻使用 PSP 理论的受训的环境。

# 9.导入 PSP

## Introducing the PSP

PSP introduction in universities starts with a PSP course in which students write 10 programs and complete 5 data analysis reports [Humphrey 95]. The university course typically takes a full semester, during which the students follow the PSP process shown in Figure 5 to complete 10 exercise programs. Students start with the PSP0 process, where they use their current programming practices, record the time they spend in each process phase, and log all the defects they find. The PSP process is enhanced through seven process versions, with students writing one or two programs with each PSP version. For each program, they use the process methods just introduced, as well as all of the methods introduced with the previous process versions. By the end of the course, the students have written 10 programs and have learned to use all of the PSP process methods. This course is designed for graduate and advanced undergraduate students.

在大学中导入 PSP 从 PSP 课程开始，在课程中，学生们编写 10 个程序并完成 5 个数据分析报告。大学课程一般需要一整个学期的时间，在这期间，学生遵循图 5 所示的 PSP 过程来完成 10 个实习程序。学生们从 PSP0 过程开始，他们使用其现有的程序实践，记录他们在每个过程阶段花费的时间，并记载他们所有发现的缺陷。PSP 过程通过 7 个过程版本、学生使用每个 PSP 版本编写一到两个程序来进行提高。对于每个程序，学生们利用刚才导入的过程方法以及先前过程版本导入的所有方法来编写。在课程最后，学生们编写完 10 个

程序并学习所有 PSP 过程方法的用法。本课程是为研究生与大学高年级学生而设计的。

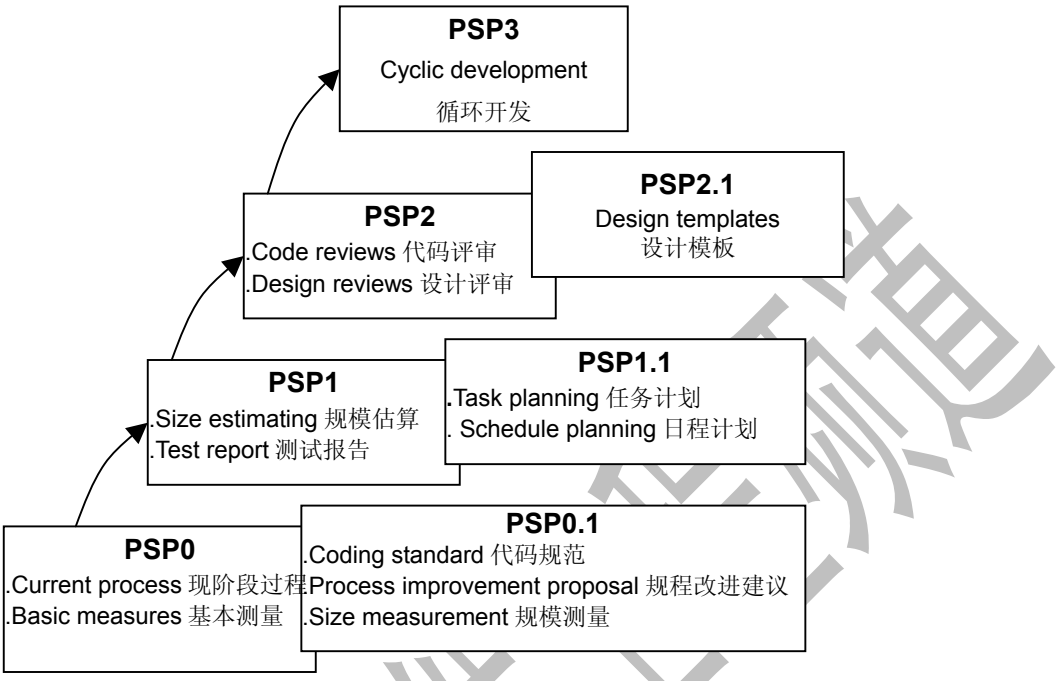



*Figure 5: PSP/TSP Course Structure*

*图 5：PSP/TSP 课程结构*

There is also a PSP course for beginning university students [Humphrey 97]. While this introductory course teaches the same basic PSP principles as the advanced course, it is not as rigorous and students typically do not learn how to practice the full set of PSP methods. However, they do start using sound engineering practices with their first programming assignments. Then they are more likely to develop sound personal practices, and it is easier to teach them rigorous engineering practices when they later take the full PSP course.

另外，也有为大学新生所设计的 PSP 课程。虽然，此项入门课程与进阶课程一样教授相同的基本 PSP 原理，但是并不严格，学生通常也不需要学习如何实践 PSP 全套的方法。然而，通过其第一个编程任务分配，他们真正开始使用可靠的工程实践了。然后，学生们更有可能发展健全的个人时间，并且当他们最后接受完整的 PSP 课程的时候，更容易教他们严格的工程实践。

When introducing the PSP in industry, engineers complete the full PSP course and write all 10 of the A-series of programming exercises in the PSP textbook [Humphrey 95]. This course takes about 120 to 150 engineering hours, and it is generally spread over 14 working days. After engineers are PSP trained, experience has shown that they must be properly managed and supported to consistently use the PSP in their work [Ferguson 97]. To help working engineers, teams, and managers consistently use the PSP methods, the

SEI has developed the Team Software Process (TSP).

在行业引入 PSP 的时候，工程师要完成全部的 PSP 课程并且编写 PSP 教科书上全部 10 个 A 系列程序习题。这个课程大概需要 120 到 150 个工程小时，并且通常延续 14 个工作日。工程师在经过 PSP 的培训后，经验显示，他们肯定被适当的管理和支持以在其工作中贯彻使用 PSP。要协助工作中的工程师、团队和管理者持续使用 PSP 方法。SEI 已经开发出团队软件过程 (Team Software Process, TSP) 。

## 10.PSP 成果

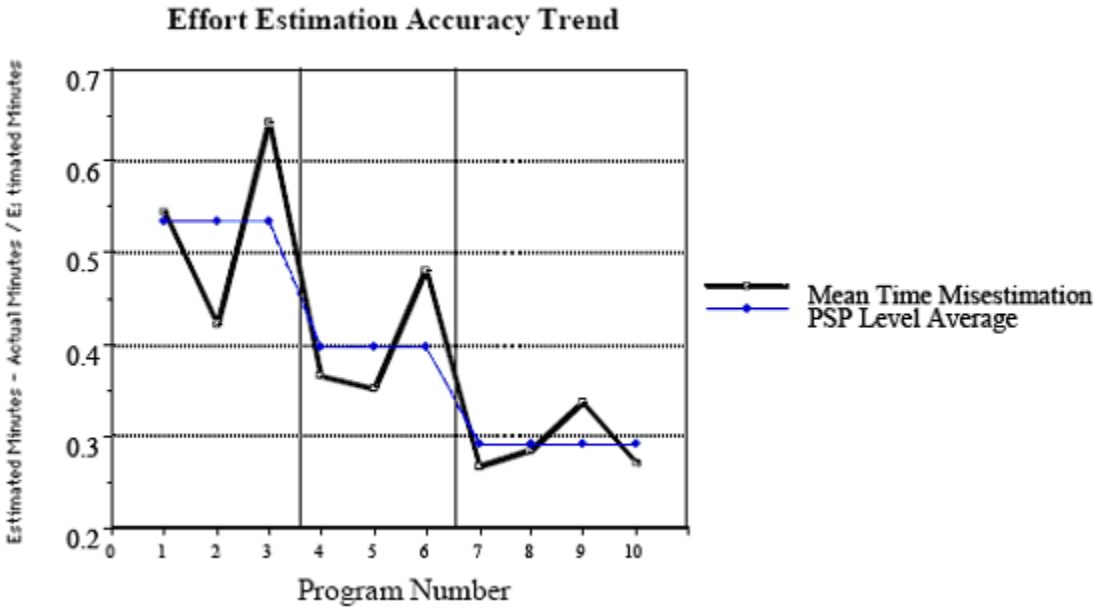
## PSP Results

Hayes and Over have shown that the PSP course substantially improves engineering performance in estimating accuracy and early defect removal while not significantly affecting productivity [Hayes 97]. Typical results for estimating accuracy are shown in Figure 6. When engineers take the PSP course, they write 10 programs and gather data on their work. In Figure 6 and the following charts, performance on program 1 is shown at the left and the results achieved with program 10 are on the right. Since the students are using their prior programming practices with program 1 and the full set of PSP methods with program 10, the presumption is that the improvements result from the PSP methods. In writing these 10 exercise programs, defect levels also improved as shown in Figure 7. The productivity results are shown in Figure 8. All these data are averages for 298 students in SEI industrial courses on the PSP.

Hayes 与 Over 已经显示虽然没有有效的影响生产率，但是 PSP 过程实质上在估算精确度和早期缺陷移除方面改进了工程师绩效。估算精确度的典型成果在图 6 中显示。当工程师接受 PSP 课程时，他们编写 10 个程序并收集自己工作的数据。在表 6 以及随后的图表中，程序 1 的效率显示在左边，程序 10 达到的成果在右边。因为学生们在程序 1 上使用其先前的程序实践，而且使用全套 PSP 方法在程序 10 上，可以假定改进的原因来自于 PSP 方法。在编写那些 10 个程序联系时，缺陷水平也改进了，如图 7 所示。生产率成果显示在图 8 中。所有这些数据为 SEI 的 PSP 行业课程 298 个学生的平均值。

估算精确度成果趋势

估算记录－实际记录/估算记录　平均时间误估算 PSP 级别平均



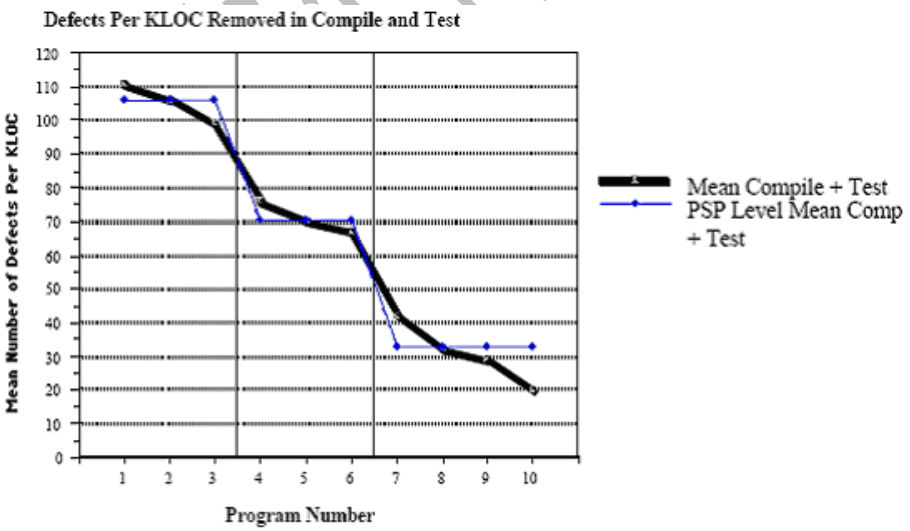

程序数

*Figure 6: Effort Estimation Accuracy*

图 6：工作量估算精确度

编译和测试中的 缺陷/KLOC

缺陷/KLOC 平均数 平均编译+测试 PSP 级别平均编译+测试




程序数

*Figure 7: Defect Level Improvement*

图 7：缺陷水平改进

代码行（新增和变更）

总开发时间 产出/小时

平均 LOC./小时 平均 LOC/小时 PSP 级别平均

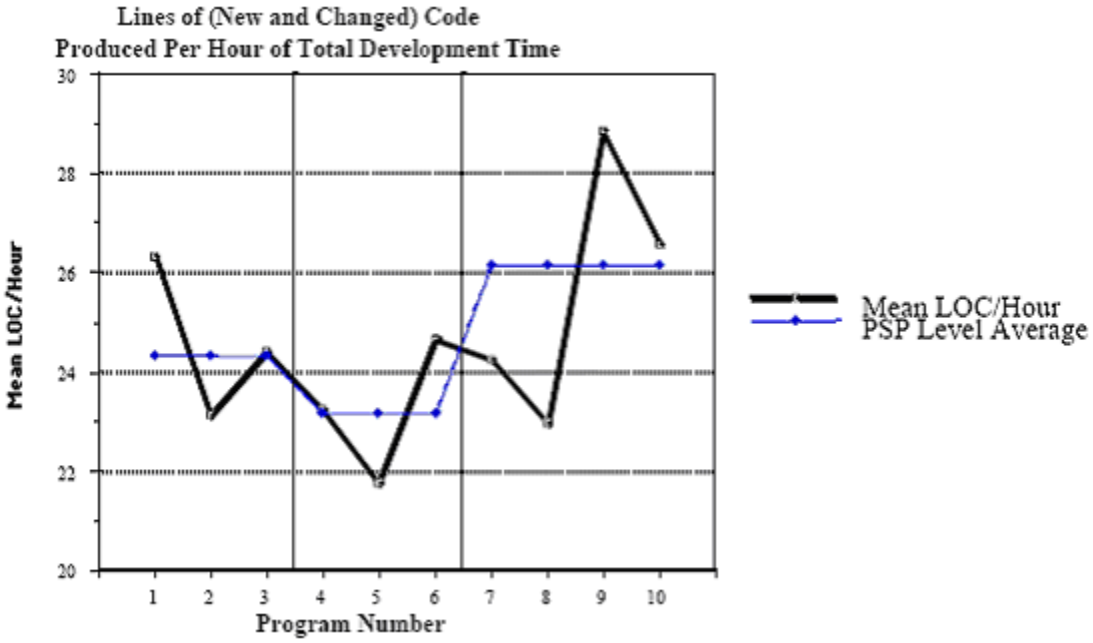



*Figure 8: Productivity Results*

图 8：生产率的成果

While the PSP is relatively new, early industry results are becoming available [Ferguson 97, Seshagiri 00].

虽然 PSP 相当的新，早期行业成果已经变为可用。

No controlled studies have yet compared the results obtained when students wrote identical programs, with an experimental group using the PSP methods and a control group not using the PSP. However, when taught by qualified instructors, student results consistently show significant improvement. Industrial data also show that engineering performance after PSP training is substantially better than before [Ferguson 97, Seshagiri 00]. It thus appears likely that PSP training was responsible for much of the improvement found in the PSP courses.

至今仍然没有受控的学生，对一个实验组使用 PSP 方法，而一个控制组不使用 PSP，在编写同一程序的时所获得的结果进行比较。然而。在合格的教师的教导下，学生们的结果一致显示出明显的改进。行业数据也显示工程师绩效在 PSP 训练后比以前有实质性的好转。因而，看起来很可能 PSP 训练对在 PSP 课程中发现的改进负更多的责任。

# 11.PSP 和过程改进

## The PSP and Process Improvement

The PSP is one of a series of three complementary process-improvement approaches. These are

PSP 是一系列三种互补过程改善方法之一。这些方法为：

- The Capability Maturity Model (CMM), which addresses software management issues [Humphrey 89, Paulk 95].
- 能力成熟度模型(Capability Maturity Model, CMM) ，对应软件管理的问题。
- The PSP, which is described in this report.
- PSP，本报告所描述的。
- The Team Software Process (TSP), which guides PSP-trained engineers, teams, and managers in developing software-intensive products. While these methods are all related, they address different aspects of organizational capability.
- 团队软件过程(Team Software Process, TSP)，在开发软件密集产品时指导受过 PSP 训练的工程师、团队、与管理者。尽管这些方法都有关联，他们描述组织能力的不同方面。

First, to produce superior products, the organization's management must have a sound business strategy and plan as well as a set of products that meet a market need. Without these, organizations cannot be successful, regardless of their other characteristics.

第一，要生产优质的产品，组织的管理阶层必须要有一个健全的商业策略与计划，以及一组满足市场需要的产品。没有这些，不管有什么其他特性，组织不可能成功。

Second, to produce superior products, management must obtain superior performance from their engineers. This requires that they hire capable people and provide them with suitable leadership, processes, and support.

第二，要生产优质的产品，管理层必须获得来自其工程师的优秀绩效。这需要雇佣有能力的人员，并且给他们提供适当的领导、过程、以及支持。

Third, the engineers must be able to work together effectively in a team environment and know how to consistently produce quality products.

第三，工程师们必须能够在团队环境中有效的协同工作，并且知道如何才能持续生产优质产品。

Fourth, the engineering teams must be properly trained and capable of doing disciplined engineering work.

第四，工程团队一定要进行适当的训练并且有能力进行遵守纪律的工程工作。

Without any one of these conditions, organizations are not likely to do superior work. In the above list, the CMM is designed to provide the second capability, the TSP the third, and the PSP the fourth. Unless engineers have the capabilities provided by PSP training, they cannot properly support their teams or consistently and reliably produce quality products.

Unless teams are properly formed and led, they cannot manage and track their work or meet schedules. Finally, without proper management and executive leadership, organizations cannot succeed, regardless of their other capabilities.

没有这些条件任何一条，组织不可能展开优质的工作。如上所列，CMM 设计用于提供第二项能力的，TSP 是第三项、PSP 则是第四项。除非工程师有 PSP 训练所提供的能力，否则，他们无法适当地支持其团队，或者持续而可靠地生产优质产品。除非团队能适当地编制与领导，否则，他们无法管理或追踪其工作，或者赶上进度。最后，没有适当的管理与高层的领导力，尽管具备了其它的能力，组织仍旧无法成功。

## 12.PSP 状态与未来趋势

## PSP Status and Future Trends

In the future, software engineering groups will increasingly be required to deliver quality products, on time, and for their planned costs. Guaranteed reliability and service levels, warranted security, and contract performance penalties will be normal and engineering staffs that cannot meet such commitments will not survive.

在将来，越来越要求软件工程组交付有质量的产品、准时并且符合其计划的成本。有保障的可靠度与服务水平，有保证的安全性，以及履行合同的惩罚将变得很平常，并且工程人员如果不能符合这样的承诺将无法生存。

While superior technical work will continue to be required, the performance of each individual engineer will be recognized as important. Quality systems require quality parts, and unless every engineer strives to produce quality work, the team cannot do so. Quality management will be an integral part of software engineering training. Engineers will have to learn how to measure the quality of their work and how to use these measures to produce essentially defect free work.

尽管不断需要优良的技术工作，每个独立工程师的绩效也被认为是很重要的。质量体系需要有质量的部分，并且，除非每个工程师努力产出有质量的工作，否则团队无法产出有质量的工作。质量管理将成为软件工程培训的一个主要部分，工程师必须学会如何测量其工作的质量以及如何使用这些测量来产生本质上无缺陷的产出。

The PSP is designed to provide the disciplined practices software professionals will need in the future. While some industrial organizations are introducing these methods, broader introduction of disciplined methods must start in universities. Academic introduction of the PSP is currently supported with courses at both introductory and advanced levels [Humphrey 95, Humphrey 97]. Several universities in the U.S., Europe, and Australia now offer the PSP, and several institutions in Asia are considering its introduction. The SEI, in conjunction with various universities, also supports a one-week summer workshop for faculty who teach or wish to teach the PSP.

PSP 设计用于提供将来需要的受训的实践软件专业人才。尽管一些行业组织已经引入这些方法，更广泛的受训的方法的引入必须从大学开始。PSP 学术性介绍目前由入门级和进阶级两种课程提供支持。现在，美国、欧洲以及澳大利亚的几所大学提供 PSP，并且亚洲的几个机构正考虑导入 PSP。SEI 协同不同大学，也通过一周夏季专题研讨来支持讲授和希望教授 PSP 的教职人员。

Industrial PSP introduction is also supported by the SEI with a PSP course for engineers and instructors. This course is offered several times a year in a condensed format tailored to industrial needs. The SEI also qualifies PSP instructors to assist organizations in introducing the PSP and it maintains a registry of qualified PSP instructors [SEI 00].

SEI 也支持行业 PSP 导入，通过使用针对工程师和讲师的 PSP 课程。此课程一年数次，以浓缩格式并根据行业需要而裁剪。SEI 也对 PSP 讲师提供资质认证以协助组织引入 PSP，并且维护 PSP 讲师认证注册名单。

While the PSP is relatively new, the early results are promising. Both industrial use and academic adoption are increasing. Assuming that these trends continue, the future should see a closer integration of the PSP, TSP, and CMM methods and a closer coupling of the PSP academic courses with the broader computer science and software engineering curricula.

尽管 PSP 是相当新颖的，但早期的成果显示非常有前途。无论是行业使用还是学术采纳都日益增长。假如这些趋势持续下去，将来应当可以看见更紧密集成的 PSP、TSP 和 CMM 理论以及更紧密耦合的 PSP 的学术课程与更广泛的计算机科学和软件工程课程。

## 参考

## References


[Boehm 81] Boehm, B. *Software Engineering Economics.* Englewood Cliffs, NJ: Prentice-Hall, 1981.

[De Champeaux 93] De Champeaux, D., Lea, D., and Faure, P. *Object-Oriented System Development,* Reading MA: Addison-Wesley, 1993.

[Deming 82] Deming, W. E. *Out of the Crisis.* MIT Center for Advanced Engineering Study, Cambridge, MA, 1982.

[Fagan 76] Fagan, M. “Design and Code Inspections to Reduce Errors in Program Development.” *IBM Systems Journal*, 15, 3 (1976).

[Fagan 86] Fagan, M. “Advances in Software Inspections.” *IEEE Transactions on Software Engineering,* SE-12, 7, (July 1986).

[Ferguson 97] Ferguson, P., Humphrey, W., Khajenoori, S., Macke, S., and Matvya, A. “Introducing the Personal Software Process: Three Industry Case Studies,” *IEEE Computer,* 30, 5 (May 1997): 24-31.

[Hayes 97] Hayes, W. and Over, J. *The Personal Software Process: An Empirical Study of the Impact of PSP on Individual Engineers* (CMU/SEI97-TR-001), Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 1997 <http://www.sei.cmu.edu

/pub/documents/97.reports/pdf/97tr001.pdf>.

[Herbsleb 97] Herbsleb, J., Zubrow, D., Goldenson, D., Hayes, W., and Paulk, M. “Software Quality and the Capability Maturity Model,” *Communications of the ACM,* 40, 6 (June 1997): 30-40.

[Humphrey 89] Humphrey, W. *Managing the Software Process.* Reading, MA: Addison-Wesley, 1989.

[Humphrey 95] Humphrey, W. A *Discipline for Software Engineering.* Reading, MA: Addison-Wesley, 1995.

[Humphrey 97] Humphrey, W. *Introduction to the Personal Software Process.*Reading MA: Addison-Wesley, 1997.

[Humphrey 98] Humphrey, W. “The Software Quality Index,” S*oftware Quality Professional,* 1, 1 (December 1998):. 8-18.

[Juran 88] Juran, J. and Gryna, F. Juran's *Quality Control Handbook, Fourth Edition.* New York: McGraw-Hill Book Company, 1988.

[Kaplan 94] Kaplan, C., Clark, R., and Tang, V. *Secrets of Software Quality, 40 Innovations from IBM.* New York, N.Y.: McGraw-Hill, Inc., 1994.

[Leveson 95] Leveson, N. *Safeware, System Safety and Computers.* Reading, MA: Addison Wesley, 1995.

[Park 92] Park, R. *Software Size Measurement: A Framework for Counting Source Statements* (CMU/SEI-92-TR-20, ADA258304). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University (Sept. 1992) <http://www.sei.cmu.edu/pub/documents/92.reports/pdf/tr20.92.pdf>.

[Paulk 95] Paulk, M., et al. *The Capability Maturity Model: Guidelines for Improving the Software Process.* Reading, MA: Addison Wesley, 1995.

[Perrow 84] Perrow, C. *Normal Accidents, Living with High-Risk Technologies.* New York, NY: Basic Books, Inc., 1984.

[SEI 00] “Building High Performance Teams Using Team Software Process (TSP) and Personal Software Process (PSP).” Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University <http://www.sei.cmu.edu/tsp>.

[Seshagiri 00] Seshagiri, G. “Making Quality Happen: The Managers’ Role, AIS Case Study,” *Crosstalk* (June 2000).

台湾 林泰龙 繁体中文版本

钟华 增加图像翻译和中英文对照，根据国内术语习惯更改。