

<<团队软件过程>>

图书基本信息

书名：<<团队软件过程>>

13位ISBN编号：9787115251893

10位ISBN编号：7115251894

出版时间：2011-6

出版时间：人民邮电出版社

作者：Watts S.Humphrey

页数：382

版权说明：本站所提供下载的PDF图书仅提供预览和简介，请支持正版图书。

更多资源请访问：<http://www.tushu007.com>

<<团队软件过程>>

前言

前言 这本书是为已经学习并使用过个体软件过程（PSP）的学生和工程师准备的。你可能在研究生课程、高级培训课程或者初级导论课程中学过PSP。或者，你可能是工程师，正在探索如何在实际的团队工作环境中应用PSP。不论是哪种情况，只要你学过PSP，你就具备了使用本书中的方法和实践的基础。

学习过PSP之后，你可能需要指导，以便弄清如何将它应用于软件过程的诸多任务上。这就是团队软件过程（TSP）的主旨：为在软件开发领域中应用成熟的工程学方法提供一个框架。

关于团队协作有很多东西可讲，本书只涵盖了最基本的内容。TSPi（团队软件过程导论）介绍了团队的概念，组建团队的基本步骤，以及在团队中工作的方法。但是要注意，本书是为导论课程设计的，并没有涵盖在大规模工业项目中使用TSP的全部资料。

TSPi是如何帮助工程师的 本书向工程师介绍有关软件开发团队协作的内容。TSPi提供了一系列结构化的步骤，告诉工程师每一步该做什么，并且详细阐述了如何将这些步骤连接起来以开发完整的产品。

TSPi还提供了两个饶有趣味且相对具有挑战性的项目练习，每个练习都具有适当的规模。既不过大，以保证在几周之内可以完成；也不过小，以保证可以模拟典型的小规模项目。如果有能力的工程师遵循本书的指导，那么他们一定能开发出完整的工作产品。

在本书建议的TSPi策略中，团队在两到三个周期内开发一个产品。

在第一个周期内，团队构建工作产品内核。

在后续每个周期内，逐渐向内核加入新功能。

这种策略体现了使用历史项目数据来制定新项目计划的好处。

另外，由于在每个周期中都有新的角色，工程师们可以在一个项目中就经历两到三种不同类型的工作。

经过几个开发周期后，工程师们就会对团队协作方法有一个广泛的了解，这样一来，他们就更有可能在将来工作中继续使用TSPi方法。

为何要学习TSPi课程 经实践证明，在培养软件专业的学生方面，项目课程是行之有效的，因此，越来越多的大学开始开设相关课程。

这些课程很受学生欢迎，修习人数一般都会超过预期。

学生想学习可应用于将来工作的知识，而团队训练课程正好满足了他们的需要。

毕业后的学生及其雇主普遍反映，软件项目课程为实际工作打下了良好的基础。

现在，在团队项目课程方面，已经有很多有益的实践。

尽管大多数此类课程都是成功的，但普遍有三个问题：第一，学生经常去尝试过大的项目；第二，他们大多只强调产品而忽略了过程；第三，总有一些成员会破坏团队气氛。

虽然TSPi不能完全避免这些问题，但它提供了有关避免这些问题或减少这些问题影响的有益指导。

为了高效利用课程时间，团队软件课程应该精心组织，并且要基于经实践证明的项目经验。

如果没有明确定义的过程或结构化的团队框架，工程师就必须自己决定该如何完成项目。

如果没有过程和框架，工作组就必须自己学习团队建设和团队协作的基本要素，而这一过程通常都会伴随着痛苦的尝试和失败。

显然，这样做代价高昂并且没有必要，因为团队协作的原则通常众所周知、简单明了。

TSPi以行之有效的团队协作方法训练工程师，首先帮助他们熟悉团队建设的过程，然后指导他们使用经过明确定义和度量的框架来开发产品。

在经过PSP训练的前提下，工程师一般都能遵循TSPi脚本步骤，并使用TSPi支持工具来计划和管理自己的工作。

遵循TSPi的指导，项目工作会变得更有效率，工程师也可以更加专注于学习软件工程，而不是在团队建设和团队管理方面花费大量的时间。

TSPi明确定义了团队角色，每个团队成员都要以某种角色进行工作。

每个角色都有详细说明，指出这个角色应该做什么，何时以及如何去完成任务。

<<团队软件过程>>

在每个团队成员都知道他们自己和其他人应该做什么的情况下，他们就能更好地作为一个团队高效工作。

如果一个团队成员没有完成工作，其他团队成员就会知道相关情况，从而及时地采取措施处理相关问题。

如果团队不能独立解决人际关系问题，他们可以求助于教师或管理人员。

本书的教师手册给出了处理很多常见的团队协作问题的有效方法。

如果学生团队成员担任明确的角色和职责，并使大家都了解，教师就能给出更加公平和详细的业绩。

除了给整个团队打分，还可以给每个人的表现打分。

这不仅激励学生表现得更好，同时也是一种更公平的给团队训练课程打分的方式。

本书组织结构 本书是为引导团队学习TSPi过程而设计的。

前两章（第一部分）是简介，第二部分阐述了完整的团队开发周期。

书中详细解释了过程脚本，并且给出了TSPi表格的完整示例。

第三部分给出了TSPi团队成员角色的详细说明：团队领导、开发经理、计划经理、质量和过程经理，以及支持经理。

当阅读有关每个角色的章节后，你可以将TSPi角色脚本作为工作参考。

TSPi课程伊始，每个学生都要填写一个INFO表格（见附录F），这个表格记录了有关学生兴趣和背景的信息。

教师根据这些信息将整个班级划分为5人小组，再给每个小组成员分配初始角色。

如果个别小组有4或6个人，教师就必须对角色进行适当调整。

每个角色都要有人担任，每个工程师必须担任至少一个角色。

对于一个4人的小组，支持经理角色的任务要在所有小组成员间分摊。

对于一个6人小组，质量和过程经理角色要分割为两个角色：质量经理和过程经理。

选择团队成员并进行角色分配之后，各个团队就启动各自的项目，并定期汇报项目进展。

每个开发周期结束时，工程师要评估团队的整体表现和每个角色的个人表现。

基于这些信息，教师就能评估每个团队和每个人的工作，并且在后续的开发周期中更好地分配角色。

如果有必要，教师可以对团队的人员组成进行调整，但是，除非出现了严重问题，否则团队的人员组成应该在整个课程中保持稳定。

使用标准及预定义的问题 尽管TSPi可以应用于各类项目，但本书只提供了两个标准和预定义的问题，它们是为满足课程多样性的要求而专门设计的。

虽然使用真实的客户需求也有好处，但是，因为以下三个原因，我们不推荐这样做。

第一，课程有严格的进度要求。

虽然多数客户一开始都同意按照固定的时间表推进项目，但是很少有客户真正知道开发软件需要多长时间。

另外，因为入门工程师一般不知道如何按照严格的进度来管理项目，所以项目失败的几率是很高的。

这个问题的根源在于，实际的客户需求大多既不够清晰也不稳定，导致频繁变更和大量延期。

第二，团队协作课程应该为特定主题而设计。

虽然开发工作产品是项目的目标之一，但是本课程的主要目的是展示使用成熟的软件工程方法的好处。

对于真实的客户需求，满足客户要求永远都是最高优先级。

一旦需求变更或客户插进来回答问题，工作就会延期，进度就会压缩，导致团队经常把精力集中在完成产品上，而忽略了过程。

从结果来看，事与愿违，我们从这类课程中得到的主要经验就是如何避免开发软件。

第三，使用标准及预定义的问题，有助于比较各团队的表现。

同样的需求有不同的实现方式，所有的团队都可以加入课程评价。

每个团队都可以介绍自己的方案，解释有关设计、实现和测试的问题。

这个过程能够充分体现出各种开发方案的实际效果，同时也为评价将来的团队提供了参考数据。

<<团队软件过程>>

虽然使用预定义的标准练习具有很多好处，但是这也让学生没有机会接触某些重要问题。例如，如果没有实际经验，就很难察觉用户需求描述的混乱和模糊。应付模糊和多变的需求是一种重要的经历，但是这可以在专门讲授需求过程的课程中进行详细学习。本书采取这种方案，首先是要讲授高效的团队协作和过程方法，然后，在后续课程中集中研究大规模开发项目中的复杂问题。

给教师的建议 本书可以多种方式使用。最主要的用法就是作为一学期或两学期的团队训练课程的教材。这种情况下，TSPi可用来开发一个单独的产品，例如附录A中介绍的两个产品之一。一学期的课程大概有2-3个周期，而两学期的课程大概有3个或者更多的周期，以开发更大规模的产品或者附录A中产品的完整版本。过程步骤可根据项目工作的规模进行适当增加和减少。表P1、P2、P3给出了3种课程方案。

在表P1所示的每个开发周期内，团队计划并跟踪每一步工作，最终完成一个完整的小型项目，包括需求、设计、编码和测试。在每个开发周期结束时，团队评估整个团队和每个角色的表现，之后，教师重新分配团队角色。在一个包含3个开发周期的项目中，工程师们实质上获得了3个完整项目和3种不同团队角色的经验。同时，他们也得到了每个开发周期的数据，在每个开发周期中，他们都可以学习如何使用从前面的开发周期中获得的经验。

本书还可在其他课程中作为团队协作练习来使用。小型项目可以在3~7周的单个开发周期中完成，例如，简单的需求开发周期大概需要3~4周，设计开发周期大概需要四周或五周，而最短的完整开发项目可能需要6周或7周。表P2展示了一个时间跨度为几周的，开发一系列需求的团队项目。类似地，表P3展示了一个设计项目。本书还可用于半个学期的课程教学，这样一来，完整的3个周期的课程需要15周，两个周期的课程需要11周，单个周期的开发项目只需要7周。

不管采用哪种课程方案，标准TSPi脚本都将指导学生成立团队、计划并实施项目。除非团队已经有过TSPi课程经历，否则任何团队都不太可能在少于3周或4周的时间内完成任何一个项目周期。原因在于，新的团队成员需要时间去学习团队过程，还要学会如何作为一个团队一起工作。这也是第一个TSPi周期需要7周时间，而后续周期只需要4周时间的原因所在。

学习本课程的基础 本课程的主要先修课是PSP课程，无论在研究生课程中或在PSP导论课程学习过都可以。如果学生是在几个学期以前学习的PSP，则要求他们在最近的课程中使用过PSP，否则，他们就需要一到两节课来回顾一下PSP计划、数据收集和质量管理的知识。如果学生使用PSP的经验不足或者根本没有用过PSP，那么就需要在整个课程期间对他们进行仔细的指导和帮助。

在尝试团队项目以前，学生应该具有软件设计和软件需求的知识背景，配置管理、项目管理和软件测试的相关知识和经验也会很有帮助。另外，学生还必须熟练掌握一门编程语言，熟练使用编程工具。

Watts S. Humphrey Sarasota, Florida

<<团队软件过程>>

内容概要

《团队软件过程(第2版)》(简称“tspi”),是美国embry-riddle aeronautical大学为计算机科学系研究生和高年级本科生开设的一门软件工程课的教科书。这本书系统地论述了如何以开发团队的形式来进行软件的开发,并对开发过程作出了具体而详尽的指导,包括团队成员之间的协调、进度的管理、质量的控制等令读者最感兴趣的方面。

《团队软件过程(第2版)》内容包括四个部分:第一部分——绪论,包括前两章,是对理论的简单介绍,介绍了什么是tspi、tspi的组织结构等内容。

第二部分——tspi过程,包括第3章到第10章,则是整个小组研究周期的详细内容,详细解释了小组软件开发的步骤,并且给出了

tspi完整形式的例子。

第三部分——小组角色,包括第11章到第15章,提供了小组成员角色的细致描述:小组领导者、开发经理、计划经理、质量/进度监督经理,以及技术支持经理。

第四部分——使用tspi,包括第16章到第18章,讲述了在使用本书的过程中需要注意的一些原则。

《团队软件过程(第2版)》实用性与可读性较强,适用于软件开发项目经理、程序员和一般编程爱好者在开发软件时参考,也可作为高等学校计算机软件工程课程的参考书使用。

作者简介

Watts S.Humphrey 是位知名作者，在软件开发过程和软件过程改进方面著有多本影响深远的图书：
Managing the Software Process(1989)、A Discipline for Software Engineering(1995)、Managing Technical People(1997)。
Humphrey曾长期在IBM公司担任高级软件开发经理，获得了大量的软件开发过程方面的经验，是卡内基梅隆大学软件工程研究所的研究员，就软件质量和软件过程方面的主题著书立说、提供咨询，并在世界各地发表这方面的演讲。

<<团队软件过程>>

书籍目录

第一部分 绪论

第1章 tspi简介

1.1 tspi是什么

工程小组为何需要过程

1.2 tspi原则

1.3 tspi的设计

1.3.1 在个体软件过程(bsp)的基础上提供一个简单的框架

1.3.2 在几个周期内开发产品

1.3.3 建立标准的质量和绩效度量

1.3.4 为团队和学生提供精确的度量

1.3.5 进行角色和团队评价

1.3.6 需要过程规范

1.3.7 提供团队问题的指导

1.4 tspi的结构和流程

周期性开发策略

1.5 tspi过程

1.6 本书结构和流程

1.7 小结

第2章 团队软件过程的基本原理

2.1 项目为何失败

处理压力

2.2 常见的团队问题

2.2.1 无效的领导力

2.2.2 不能做出妥协安排或不善于合作

2.2.3 缺少参与

2.2.4 拖拉与缺乏信心

2.2.5 质量低劣

2.2.6 功能多余

2.2.7 无效的组员互评

2.3 团队是什么

2.3.1 团队规模

2.3.2 具有凝聚力的团队(jelled team)

2.3.3 团队协作的基本条件

2.4 建设高效团队

2.4.1 团队凝聚力

2.4.2 挑战性的目标

2.4.3 反馈

2.4.4 共同的工作框架

2.5 团队如何发展

团队如何成为具有凝聚力的团队

2.6 tspi如何建设团队

2.6.1 目标

2.6.2 角色

2.6.3 计划

2.6.4 沟通

<<团队软件过程>>

2.6.5 外部沟通

2.7 小结

2.8 参考文献

第二部分 tspi过程

第3章 启动一个团队项目

3.1 为什么需要团队启动过程

3.2 团队目标

3.2.1 设定目标需要考虑的因素

3.2.2 设定团队目标

3.2.3 tspi的目标设定

3.3 团队成员目标

设定团队成员目标

3.4 角色目标

3.4.1 团队领导目标

3.4.2 开发经理目标

3.4.3 计划经理目标

3.4.4 质量和过程经理目标

3.4.5 支持经理目标

3.5 tspi启动脚本

3.5.1 学生信息

3.5.2 产品目标

3.5.3 团队分工

3.5.4 团队目标

3.5.5 团队会议与第一次团队会议

3.5.6 数据需求

3.5.7 项目开始

3.5.8 项目资料库

3.5.9 tspi支持工具

3.6 小结

第4章 开发策略

4.1 计划先行

4.1.1 承诺之前先计划

4.1.2 为本课程制定计划

4.2 策略是什么

4.3 概念设计

4.4 风险管理

管理风险

4.5 复用策略

4.6 策略脚本

4.6.1 入口准则

4.6.2 建立策略评判准则

4.6.3 完成概念设计

4.6.4 选择开发策略

4.6.5 完成初步规模估算

4.6.6 完成初步时间估算

4.6.7 评估风险

4.6.8 建立策略文档

<<团队软件过程>>

- 4.6.9 更新开发策略
- 4.6.10 制定配置管理计划
- 4.6.11 出口准则
- 4.7 小结
- 第5章 开发计划
- 5.1 计划的必要性
 - 5.1.1 为什么制定计划
 - 5.1.2 平衡的计划
 - 5.1.3 对照计划跟踪进展
 - 5.1.4 详细计划
 - 5.1.5 处理未计划任务
 - 5.1.6 估算级别
 - 5.1.7 实现计划
- 5.2 tspi计划过程
- 5.3 tspi支持工具
- 5.4 开发计划脚本
 - 5.4.1 入口准则
 - 5.4.2 项目计划步骤2.1
 - 5.4.3 项目计划步骤2.2
 - 5.4.4 项目计划步骤3.1
 - 5.4.5 项目计划步骤3.2
 - 5.4.6 项目计划步骤4.1
 - 5.4.7 项目计划步骤4.2
 - 5.4.8 项目计划步骤5
 - 5.4.9 项目计划步骤6
 - 5.4.10 项目计划步骤7
 - 5.4.11 最后的计划步骤
 - 5.4.12 出口准则
- 5.5 跟踪工作情况
 - 5.5.1 项目跟踪步骤1
 - 5.5.2 项目跟踪步骤2
 - 5.5.3 项目跟踪步骤3
 - 5.5.4 项目跟踪步骤4
 - 5.5.5 项目跟踪步骤5
 - 5.5.6 项目跟踪步骤6
 - 5.5.7 项目跟踪步骤7
 - 5.5.8 项目跟踪步骤8
- 5.6 质量计划
 - 5.6.1 概要比率
 - 5.6.2 零缺陷率(pdf)
 - 5.6.3 每页缺陷数
 - 5.6.4 缺陷数/kloc
 - 5.6.5 缺陷比率
 - 5.6.6 开发时间比率
 - 5.6.7 a/fr
 - 5.6.8 评审速率和审查速率
 - 5.6.9 缺陷注入率

<<团队软件过程>>

- 5.6.10 缺陷排除率
- 5.6.11 阶段收益
- 5.6.12 过程收益
- 5.6.13 处理低质量部件
- 5.6.14 出口准则
- 5.7 小结
- 5.8 参考文献
- 第6章 定义需求
- 6.1 需求是什么
- 6.2 为什么需要需求
- 6.3 需求变更
- 需求提取
- 6.4 软件需求规格说明书
- 6.4.1 需求可追溯性
- 6.4.2 平衡工作量
- 6.5 tspi需求脚本
- 6.5.1 入口准则
- 6.5.2 要求陈述评审
- 6.5.3 要求陈述澄清
- 6.5.4 需求任务分配
- 6.5.5 需求文档
- 6.5.6 系统测试计划
- 6.5.7 需求和系统测试计划审查
- 6.5.8 需求更新
- 6.5.9 用户srs评审
- 6.5.10 需求基线
- 6.5.11 出口准则
- 6.6 小结
- 6.7 参考文献
- 第7章 与团队一起设计
- 7.1 设计原则
- 7.2 在团队中设计
- 7.2.1 利用整个团队
- 7.2.2 设计研究
- 7.2.3 利用所有团队成员的才智
- 7.3 设计标准
- 7.3.1 设计表达标准
- 7.3.2 用例或psp操作场景
- 7.3.3 状态机分析
- 7.3.4 产生精确的设计
- 7.4 复用性设计
- 7.4.1 可复用接口标准
- 7.4.2 可复用文档标准
- 7.4.3 可复用部件质量
- 7.4.4 应用支持
- 7.5 可用性设计
- 7.6 可测试性设计

<<团队软件过程>>

黑盒测试与白盒测试

7.7 设计评审和审查

审查的其他好处

7.8 tspi设计脚本

7.8.1 入口准则

7.8.2 高层设计

7.8.3 设计标准

7.8.4 产品总体结构

7.8.5 设计任务分配

7.8.6 设计规格说明书

7.8.7 集成测试计划

7.8.8 设计审查

7.8.9 设计更新

7.8.10 设计基线

7.8.11 出口准则

7.9 小结

7.10 参考文献

第8章 产品实现

8.1 设计完成准则

8.1.1 设计级别

8.1.2 平行实现

8.2 实现标准

8.2.1 标准评审

8.2.2 编码标准

8.2.3 规模标准

8.2.4 度量其他类型产品的规模

8.2.5 缺陷标准

8.2.6 缺陷预防

8.3 实现策略

8.3.1 实现策略：评审

8.3.2 实现策略：复用

8.3.3 实现策略：测试

8.4 评审和审查

8.4.1 随机缺陷

8.4.2 对测试的影响

8.4.3 完全测试的困难

8.4.4 源程序的设计审查

8.5 imp脚本

8.5.1 入口准则

8.5.2 实现计划

8.5.3 详细设计与设计评审

8.5.4 测试开发

8.5.5 详细设计审查

8.5.6 编码及代码评审

8.5.7 代码审查

8.5.8 单元测试

<<团队软件过程>>

- 8.5.9 组件质量评审
- 8.5.10 组件发布
- 8.5.11 出口准则
- 8.6 小结
- 8.7 参考文献
- 第9章 集成与系统测试
 - 9.1 测试原则
 - 9.2 tspi测试策略
 - 9.3 构建和集成策略
 - 9.3.1 大爆炸策略
 - 9.3.2 一次一个策略
 - 9.3.3 测试群策略
 - 9.3.4 扁平系统策略
 - 9.4 系统测试策略
 - 可选系统测试策略
 - 9.5 测试计划
 - 9.6 跟踪与度量测试
 - 9.6.1 测试日志
 - 9.6.2 缺陷易发模块
 - 9.6.3 模块缺陷数据
 - 9.6.4 跟踪缺陷数据
 - 9.7 文档
 - 9.7.1 文档的重要性
 - 9.7.2 文档设计
 - 9.7.3 文档提纲
 - 9.7.4 书写风格
 - 9.7.5 文档评审
 - 9.8 tspi测试脚本
 - 9.8.1 入口准则
 - 9.8.2 测试开发
 - 9.8.3 构建
 - 9.8.4 集成
 - 9.8.5 系统测试
 - 9.8.6 回归测试
 - 9.8.7 文档
 - 9.8.8 出口准则
 - 9.9 小结
 - 9.10 参考文献
- 第10章 结项总结
 - 10.1 为什么要进行结项总结
 - 10.2 结项总结能为你做什么
 - 10.3 过程改进建议
 - 10.4 tspi结项总结脚本
 - 10.4.1 入口准则
 - 10.4.2 评审过程数据
 - 10.4.3 质量评审
 - 10.4.4 角色评估

<<团队软件过程>>

- 10.4.5 准备周期报告
- 10.4.6 周期报告
- 10.4.7 角色报告
- 10.4.8 工程师个人报告
- 10.4.9 撰写报告
- 10.4.10 角色评估
- 10.4.11 角色评估建议
- 10.4.12 出口准则
- 10.5 小结
- 10.6 参考文献
- 第三部分 团队角色
- 第11章 团队领导角色
- 11.1 团队领导的目标
 - 11.1.1 团队成员的共同目标
 - 11.1.2 团队领导的目标1
 - 11.1.3 团队领导的目标2
 - 11.1.4 团队领导的目标3
 - 11.1.5 团队领导的目标4
 - 11.1.6 团队领导的目标5
- 11.2 有用的团队领导的技能和能力
 - 11.2.1 有拥护者的领导
 - 11.2.2 领导需要表现
 - 11.2.3 领导需要面对困境
 - 11.2.4 领导处理人际关系
- 11.3 团队领导的主要活动
 - 11.3.1 团队领导的主要活动1
 - 11.3.2 团队领导的主要活动2
 - 11.3.3 团队领导的主要活动3
 - 11.3.4 团队领导的主要活动4
 - 11.3.5 团队领导的主要活动5
 - 11.3.6 团队领导的主要活动6
 - 11.3.7 团队领导的主要活动7
 - 11.3.8 团队领导的主要活动8
- 11.4 团队领导的项目工作
- 11.5 小结
- 第12章 开发经理角色
- 12.1 开发经理的目标
 - 12.1.1 团队成员的共同目标
 - 12.1.2 开发经理的目标1
 - 12.1.3 开发经理的目标2
- 12.2 对开发经理有益的技能 and 能力
- 12.3 开发经理的主要活动
 - 12.3.1 开发经理的主要活动1
 - 12.3.2 开发经理的主要活动2
 - 12.3.3 开发经理的主要活动3
 - 12.3.4 开发经理的主要活动4
 - 12.3.5 开发经理的主要活动5

<<团队软件过程>>

- 12.3.6 开发经理的主要活动6
- 12.3.7 开发经理的主要活动7
- 12.3.8 开发经理的主要活动8
- 12.3.9 开发经理的主要活动9
- 12.3.10 开发经理的主要活动10
- 12.3.11 开发经理的主要活动11
- 12.4 开发经理的项目活动
- 12.5 小结
- 第13章 计划经理角色
- 13.1 计划经理的目标
 - 13.1.1 团队成员的共同目标
 - 13.1.2 计划经理的目标1
 - 13.1.3 计划经理的目标2
- 13.2 对计划经理有益的技能 and 能力
- 13.3 计划经理的主要活动
 - 13.3.1 计划经理的主要活动1
 - 13.3.2 计划经理的主要活动2
 - 13.3.3 计划经理的主要活动3
 - 13.3.4 计划经理的主要活动4
 - 13.3.5 计划经理的主要活动5
 - 13.3.6 计划经理的主要活动6
- 13.4 计划经理的项目活动
- 13.5 小结
- 第14章 质量和过程经理角色
- 14.1 质量和过程经理的目标
 - 14.1.1 团队成员的共同目标
 - 14.1.2 质量和过程经理的目标1
 - 14.1.3 质量和过程经理的目标2
 - 14.1.4 质量和过程经理的目标3
 - 14.1.5 质量和过程经理的目标4
- 14.2 对质量和过程经理有益的技能 and 能力
- 14.3 质量和过程经理的主要活动
 - 14.3.1 质量和过程经理的主要活动1
 - 14.3.2 质量和过程经理的主要活动2
 - 14.3.3 质量和过程经理的主要活动3
 - 14.3.4 质量和过程经理的主要活动4
 - 14.3.5 质量和过程经理主要活动5
 - 14.3.6 质量和过程经理的主要活动6
 - 14.3.7 质量和过程经理的主要活动7
 - 14.3.8 质量和过程经理的主要活动8
 - 14.3.9 质量和过程经理的主要活动9
- 14.4 质量和过程经理的项目活动
- 14.5 小结
- 第15章 支持经理角色
- 15.1 支持经理的目标
 - 15.1.1 团队成员的共同目标
 - 15.1.2 支持经理的目标1

<<团队软件过程>>

- 15.1.3 支持经理的目标2
- 15.1.4 支持经理的目标3
- 15.1.5 支持经理的目标4
- 15.2 对支持经理有益的技能 and 能力
- 15.3 支持经理的主要活动
 - 15.3.1 支持经理的主要活动1
 - 15.3.2 支持经理的主要活动2
 - 15.3.3 支持经理的主要活动3
 - 15.3.4 支持经理的主要活动4
 - 15.3.5 支持经理的主要活动5
 - 15.3.6 支持经理的主要活动6
 - 15.3.7 支持经理的主要活动7
 - 15.3.8 支持经理的主要活动8
- 15.4 支持经理的项目活动
- 15.5 小结
- 第四部分 使用tspi
- 第16章 管理自我
 - 16.1 责任心
 - 16.1.1 一个失败的项目
 - 16.1.2 履行责任
 - 16.1.3 决不放弃
 - 16.1.4 面对现实
 - 16.1.5 负责任所带来的风险
 - 16.1.6 陈述事实
 - 16.1.7 事实往往是可以争议的
 - 16.2 目标导向性
 - 16.2.1 着眼于日程表
 - 16.2.2 目标提供了工作重点和优先级
 - 16.2.3 你想让我做什么？
 - 16.3 原则性
 - 16.3.1 不与团队中其他人合作
 - 16.3.2 如何遵循处事的几个原则
 - 16.4 小结
 - 16.5 参考文献
- 第17章 在团队中工作
 - 17.1 具有凝聚力的团队
 - 17.2 团队工作的责任
 - 17.3 团队成员间的沟通
 - 17.3.1 可见性
 - 17.3.2 聆听
 - 17.3.3 协商
 - 17.3.4 为什么有原则的协商是有效的
 - 17.3.5 花费足够的时间
 - 17.4 作出和履行承诺
 - 17.4.1 负责的承诺
 - 17.4.2 做出承诺

<<团队软件过程>>

- 17.5 参与团队活动
 - 17.5.1 勇于发表自己的看法
 - 17.5.2 支持坚持己见的人
 - 17.5.3 唤起别人的注意
 - 17.5.4 对他人的意见要给予关注
- 17.6 团队建设的责任
- 17.7 接受并承担团队所分配的角色
- 17.8 建立并努力完成团队目标
- 17.9 建立和维护团队
 - 17.9.1 难以相处的团队成员
 - 17.9.2 院校团队的问题
 - 17.9.3 寻求帮助
 - 17.9.4 支持
- 17.10 小结
- 17.11 参考文献
- 第18章 团队工作
- 附录a tspi采样练习的要求说明
 - a.1 目的
 - a.2 “变化计数器”功能要求说明
 - a.3 “程序分析器”功能要求说明
 - a.4 参考文献
- 附录b 软件配置管理
 - b.1 软件配置管理问题
 - b.2 软件配置管理概要
 - 不需要的项
 - b.3 scm计划
 - b.3.1 配置标识计划
 - b.3.2 配置控制规程
 - b.3.3 配置控制委员会
 - b.3.4 变更申请表
 - b.4 系统基线
 - b.4.1 基线提交
 - b.4.2 备份规程
 - b.4.3 配置状态报告
 - b.5 scm过程自动化
 - b.6 软件配置管理过程
 - b.6.1 第一步：制定scm计划
 - b.6.2 第二步：管理系统基线
 - b.6.3 第三步：管理变更
 - b.6.4 第四步：报告scm状态
- 附录c 软件审查
 - c.1 什么是审查
 - c.1.1 审查是如何进行的
 - c.1.2 评审的时机
 - c.1.3 使用规定的审查程序
 - c.2 什么使审查有效
 - c.2.1 审查整个程序

<<团队软件过程>>

- c.2.2 集思广益
- c.2.3 采取不同的视角
- c.2.4 提供发现错误的机会
- c.2.5 全面测试的重要性
- c.2.6 只审查经个人评审过的产品
- c.3 审查方法
 - c.3.1 检查单
 - c.3.2 视角
 - c.3.3 产品侧重点
 - c.3.4 审查实践
- c.4 审查数据
 - c.4.1 审查速率
 - c.4.2 评审占开发比率
 - c.4.3 审查收益
- c.5 审查报告：ins表
- c.6 估算遗留的缺陷数
 - c.6.1 估算总数
 - c.6.2 估算程序中的缺陷数
 - c.6.3 软件审查中的捕获-重捕获方法
 - c.6.4 2个工程师的估算范例
 - c.6.5 3个工程师的估算范例
 - c.6.6 注意
 - c.6.7 一些改进
- c.7 具有高个人审查收益的重要性
- c.8 安排审查时间
- c.9 tspi审查脚本
 - c.9.1 入口准则
 - c.9.2 计划审查工作
 - c.9.3 召开审查介绍会
 - c.9.4 评审产品
 - c.9.5 召开审查会议
 - c.9.6 遍历产品
 - c.9.7 估算遗留缺陷数
 - c.9.8 总结审查会议
 - c.9.9 修改产品，验证缺陷修复
 - c.9.10 出口准则
- c.10 参考文献
- 附录d tspi脚本
- 附录e 角色脚本
- 附录f tspi表格及其使用说明
- 附录g tspi标准与规格说明

版权说明

本站所提供下载的PDF图书仅提供预览和简介，请支持正版图书。

更多资源请访问:<http://www.tushu007.com>