Visual Studio Team System

软件工程实践

Software Engineering with
Microsoft Visual Studio Team System

Microsoft Visual Studio Team System

(美) Sam Guckenheimer
Juan J. Perez 著

苏南 贺洁 等译

附赠DVD

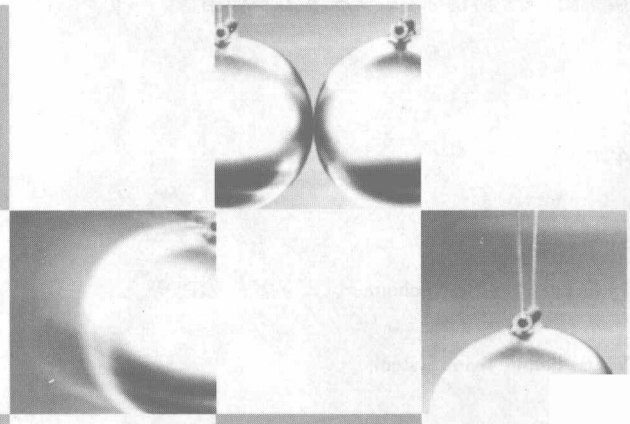
机械工业出版社
China Machine Press

Ivar Jacobson 作序

Visual Studio Team System

软件工程实践

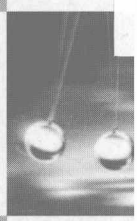
Software Engineering with
Microsoft Visual Studio Team System

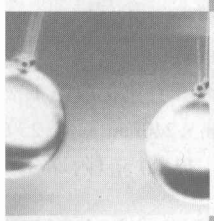


Microsoft
Visual Studio
Team System

(美) Sam Guckenheimer
Juan J. Perez 著

苏南 贺洁 等译





机械工业出版社
China Machine Press

本书讲述 VSTS 的基本概念及其实践理念。本书不仅包括最新的软件工程领域的思想和概念，还为软件开发提出了一种崭新的思维方式——价值增加。价值增加是本书的核心思想，同时也是 VSTS 的核心设计理念。

本书理论与实例并重，图文并茂，运用大量实例详实地论述了如何将最现代的软件工程思想和价值增加的思想应用到需求、项目管理、架构设计、开发和测试等软件开发生命周期中。本书适合软件项目管理人员、开发人员参考。

本书版权登记号：图字：01-2006-5654

图书在版编目(CIP)数据

Visual Studio Team System 软件工程实践/(美)古肯海默(Guckenheimer, S.)著；苏南等译. -北京：机械工业出版社，2007.2

书名原文：Software Engineering with Microsoft Visual Studio Team System

ISBN 978-7-111-20758-0

Ⅰ.V…　Ⅱ.①古…　②苏…　Ⅲ.计算机网络—程序设计　Ⅳ.TP393

中国版本图书馆 CIP 数据核字(2007)第 005881 号

机械工业出版社(北京市西城区百万庄大街 22 号　邮政编码　100037)

责任编辑：杨庆燕

北京瑞德印刷有限公司印刷 · 新华书店北京发行所发行

2007 年 3 月第 1 版第 1 次印刷

186mm × 240mm · 15.25 印张

定价：39.00 元(附DVD光盘)

凡购本书，如有倒页、脱页、缺页，由本社发行部调换

本社购书热线：(010)68326294

对本书的赞誉

“太迷人了！这本书中包括了 VSTS 功能的许多细节，还有这些功能之所以会包括在 VSTS 产品中的背后原因，这是只有 VSTS 团队内部成员才能提供的信息。可能更重要的是，本书在解释每个功能为什么对你很关键的同时，也包含了对其技术能力或 how-to指令的介绍。对于过去那些过程，本书避开了其中的隐患，详细论述了其中的“最佳击球点”。在做这些时，它定义了未来方法的指导，确定了用在你自己的项目中精细化和定制方法的度量元。”

——Mark Michaelis，《Essential C# 2.0》的作者

“对于那些希望拥抱 Visual Studio Team System 和 Microsoft Solutions Framework 4.0 的人来说，怎么样做才能符合开发者的本意呢？本书是他们的必读之作。本书的关键主题之一是‘带有问责制的敏捷’。本书既解释了价值增加的项目方法带来的思维方式的迁升，又描述了 Team System 如何支持这种迁升。本书中包含了很多如何将这种方法应用到 VSTS 开发中的实例，它们将这种方法的思想传达到了各种规模的实际软件开发中。”

——Aaron Kowall，EDS Applications Portfolio Development，创新工程组

“Sam Guckenheimer 把我们引入到了一个可信任的透明度的年代，可信任的透明度将对我们管理软件开发项目的方式进行一场革命。不要仅仅只是购买 Visual Studio Team System，还要学习如何用它来驾驭变更并获得回报。Sam 向你展示了应该如何去做。”

——David J. Anderson，《Agile Management for Software Engineering》的作者

“在本书有限的篇幅中，Sam 抓住了 Visual Studio Team System 的本质。如果你作为开发人员、测试人员、项目经理、架构师或者 CIO 参与了制作软件或管理软件项目的过程，你会想给你的团队中的每个人买一本。本书不仅使当代软件工程的实践变得可

以企及，还使用了很多清楚的例子来介绍如何用 Team System 工具来实现这些软件工程实践。与以往关于软件方法学的书籍不同，本书没有回避将理论运用到实践之中。无论你是否已经有了 VSTS 而在考虑如何用它，或者你仅仅想提高软件生产力和业务一致性，你都会发现本书充满了具有洞察力的内容。本书令人愉快、平易近人、适于在周末轻松地阅读。”

——Rick LaPlante，Microsoft，Visual Studio Team System，总经理

“Sam Guckenheimer 多年以来一直都是软件测试社区中思想的发电站和导师。很荣幸终于看到了他的一本著作，尤其是这样一本体现了他的思想的著作。”

——Cem Kaner，J. D.，Ph. D.，软件工程教授，Florida Institute of Technology；
《Lessons Learned in Software Testing》和《Testing Computer Software》的主要作者

“在本书中，Sam Guckenheimer 讲述了 Team System 与新兴的价值增加的软件过程思想的完整形态。度量所交付的价值，而非像长期以来所坚持的那样度量已完成的工作，这一思维方式才是 Team System 设计和实现的核心。因此，你会发现 Team System 提供了空前的项目透明度，这改进了团队的交互和项目的可预见能力。除此之外，它这样做也不会带来消耗时间的管理开销从而增加团队成员的负担。为了完全达到 Team System 背后的愿景，为了使价值增加的软件开发的良性循环成为可能，你必须阅读此书。”

——Rob Caron，Microsoft 内容架构师，Team System Nexus 的作者

“Sam Guckenheimer 是个技术外交家。在这个采用敏捷方法的游击队列队迎战身披 CMMI 铠甲的罗马军团的世界里，Sam 提出了一条共存的路径。这是一本关于软件工程的第一流的著作。在对计划、文档、管制、审计能力和组织等闪光点的讨论中，Sam 分别展示了敏捷和较正式的实践两种情形，并且还描述了每种情形的理想条件。虽然本书内容展示的是使用 VSTS 的情境，但是其指导性是普遍适用的。Sam 写到了项目中的每个角色，并向他们提供了合理的建议，无论他们所选择的实践是轻量级的还是重量级的。本书中的资料很新也很及时，它讨论了面向服务的架构、测试驱动的开发以及在用户界面社区中所开发的设计技术。Sam 的书是一部非常出众的软件教科书。”

——Dr. Bill Curtis，Borland 软件集团首席过程官，
《People Capability Maturity Model》的主要作者

“Sam Guckenheimer 是一个真正的用户代言人。Team System 是一个平台，它以使用工具来自动化、使用度量元来管理的方式提供过程，而这些对于用户几乎是透明的。在 Team System 的支撑下，Sam 展示了一种用于软件工程的、实用的、可做到的方法，同时正视我们还有很多难以解决的问题这一事实。”

——James Behling，Accenture 公司 Accenture 交付方法主任架构师

“Sam Guckenheimer 和我曾经一直走在一条共同的路上，我们的目标是改进开发团队与运行团队之间的支持。对于软件开发的最佳实践来说，Sam 的书给出了一种容易理解的、以过程为中心的方法，它们都收录在 MSF 中并通过 Visual Studio Team System 来交付。瀑布模型是一个失败，但是 Sam 的书可以指引你通过使用 Visual Studio Team System，采用刚好足够完成任务的过程来进行快速开发。”

——Brian White，iConclude 公司产品管理的资深总监，《Software Configuration Management Strategies and Rational ClearCase：A Practical Introduction》的作者

“在今天的敏捷环境中，透明度是一个关键的元素。Sam 过去和现在都在帮助创建总体架构，总体架构为 Team System 提供了完整性和透明度的级别，这对于从敏捷项目到大型团队都是必要的。如果将这一透明度用于一个孕育了信任和个人安全感的环境中，就能够随着敏捷方法的纪律的繁衍而创造一个更有生产力的开发团队。报告速度之类的信息也变得毫不费力。现在，整个软件开发团队（包括业务分析师、架构师和测试人员）都能够加入到敏捷过程之中。”

——Granville “Randy” Miller，《A Practical Guide to eXtreme Programming and Advanced Use Case Modeling》的作者之一

“你能想像一下将业务流程再造（BPR）的工具应用于软件工程（SE）吗？有一个能实际帮助 IT 产业变得更精益的工具吗？这正是这本书全部所讲的内容！本书让我们开阔了眼界：它打开了通往软件工程新时代的大门。本书中的危险问题很简单：MSFT VSTS 是否能够让我们的 IT 产业变得更像一门科学，而不像一门艺术呢？Sam Guckenheimer 不仅解释了为什么会有这种情况，而且给出了很多提示，告诉我们如何能让整个 SE 团队在没有手工开销的情况下也能演变得更加有生产力和有效率。”

——Francis T. Delgado，资深规划经理，Avanade，Inc.

译 者 序

对于软件过程的实践，存在着敏捷过程和以 CMM/CMMI 为代表的正式过程这两大黑白分明的阵营。本书从一个崭新的、客观的高度来看待这两大阵营之间的争论，使你了解到这黑白两色不过是五彩光谱的两端；没有正确的过程，只有适合的过程。

作为 VSTS 的主要设计者，作者一直站在软件工程领域的最前沿。本书不仅包括了最新的软件工程领域的思想和概念，还为软件开发提出了一种崭新的思维方式，这便是价值增加。价值增加是本书的核心思想，同时也是 VSTS 的核心设计理念。

本书不是一本讲述如何具体操作 VSTS 的书，它所讲述的重点是 VSTS 的思想及其实践。在理解了 VSTS 的思想之后，你才能够根据自己项目本身的特点正确地使用 VSTS，发挥出 VSTS 最大的潜力，使你的软件项目顺利地进行，为客户提供可感知的价值。

本书理论与实践并重，图文并茂，运用大量实例详实地论述了如何将最现代的软件工程思想和价值增加思想应用到需求、项目管理、架构设计、开发和测试等软件开发生命周期中。

如果你正在使用 VSTS 或计划使用 VSTS，本书能够帮助你了解如何在软件开发过程中运用 VSTS。如果你并不使用 VSTS，书中丰富的具体实例也能帮助你对现代的软件工程思想有一个透彻的认识，并且对于管理软件开发也有指导意义。

感谢作者 Sam Guckenheimer 先生。在本书的翻译过程中，他回答了我们很多关于对原文的理解和文化背景方面的问题。感谢本书繁体版的译者蔡焕麟先生，他在网上分享了一些本书的章节和翻译心得。这些都给本书的翻译工作提供了参考和启发。

参加本书翻译的人员除封面署名外还有尚计升、史红伟、祝国志、张峰峰、张文军、张艳军、王小财和周宝华。本书翻译力求忠于原著，但由于时间仓促，译者水平有限，翻译的错误和不妥之处在所难免，欢迎广大读者批评指正。

苏 南

2006 年 11 月于北京

序　言

近10年中，我一直鼓励Sam Guckenheimer写一部关于软件工程的书。“噢不，我还没有准备好。”他始终这样回答我。

随着Visual Studio Team System的发布，Sam再也没有借口了：他真的不得不解释一下他的那些软件工程的观点了，这样才能帮助人们弄懂这个体现了这些观点的产品。本书的优点在于实验课程和理论占据了相同的分量，既不是整本的产品广告，也不是对于软件工程的哲学进行的一次含糊的讨论。我喜欢书中那些具体的例子：它们使得概念变得有生命了。

本书中的一个关键概念是价值增加的过程。Sam相信我们在与软件打交道的方式上面临着一个思维方式的变迁，这看来是正确的。工作消减的思维方式已经给软件开发过程带来了很多问题，并最终导致项目失败的比例很高。价值增加的思维方式是否能够解决这些问题而又不会造成新的问题呢？当然，这还需要我们拭目以待。

过去，软件度量的实践没有充分发挥其潜力，很大程度上是因为数据收集的成本很高。正如Sam在本书中所解释的，使用工具进行日常活动使得数据收集不再痛苦，这也为有意义的度量开创了一组新的机会。Sam所做的还不止这些；他还运用一些来自精益(lean)项目管理的更有趣的技术，来演示如何以日报为基础对软件项目进行问题解析。这也使我们能够可靠地应用价值增加的过程。

近10年来，很多思想都渗透到了软件工程的不同领域，包括编程、用户体验、测试和架构方面。Sam把这些思想中最好的内容聚集在一起，应用到整个的软件生命周期中。

我相信你会像我一样喜欢这些内容。

Ivar Jacobson 博士

Ivar Jacobson Consulting LLC

前　言

为什么要写这本书

在2003年加入微软后，我负责Visual Studio Team System（VSTS）项目，这一新产品线刚刚于2005年底发布。作为这一组产品的规划师，我担任了首席客户代言人这一我很喜爱的角色。我已经在IT行业工作了20多年，在职业生涯的大部分时间里，我做过测试员、项目经理、分析师和开发员。

作为一名测试人员，对于那些高级开发者的实践（比如单元测试、代码覆盖度、静态分析以及内存和性能的特征分析），我始终都能够理解其理论价值。同时，我不能理解一个人怎样才能有耐性去学习那些晦涩难懂工具，而这些工具又是你在遵循正确的实践时所需要的。

作为一名项目经理，我经常烦恼的是：我们所能得到的体面的数据仅仅是那些与缺陷有关的数据。单靠缺陷数据来驾驭一个项目，就仿佛是闭着眼睛开车，只有在你碰到什么东西时才转动方向盘。你真正想要看的是能指引正确方向的正确指示器，而不是仅仅在偏离路线时能感到的颠簸。在这一岗位上，也是类似的情况，对于诸如代码覆盖度和项目速度之类的度量，我始终都能够理解它们的价值，但是我不能理解一个人怎样才能实际地收集到所有这些数据。

作为一名分析员，我爱上了建模。我边看边思考，发现图形化的模型控制了文档和沟通的方式。但是一旦进入到实现阶段，模型就总会过期。而且模型还正好没有处理开发者、测试者和运行中的关键顾虑。

在所有这些情形中，由于把整个团队的各开发环节连起来太困难了，我受到了挫折。我喜爱SCRUM（一个敏捷过程）中“单一产品待处理队列”的创意：你能在一个地方看到所有工作，但是人们实际使用的工具却都以它们各自的方法来分割工作。这些需求与那些工作有什么关系，与这边的模型元素，与那边的测试又有什么关系？还有，在这样的混合中，源代码又处于一个什么样的位置呢？

从历史的观点来看，当IT停止了对手工过程的自动化尝试时，我认为，已经“柳暗花明又一村”了。现在IT所询问的问题是“使用了自动化之后，我们怎么样才能再造

核心业务过程?”这是IT开始交付真正的业务价值的时刻了。

人们常说“鞋匠的孩子没鞋穿”。这对于IT也很贴切。我们在忙于其他业务过程的自动化的同时，却在很大程度上忽略了我们自己。实际上，所有的以IT专业人士和团队为目标的工具仿佛仍然是对老的手动过程的自动化。那些过程在自动化之前需要很高的开销，在有了自动化以后，它们仍然有很高的开销。有多少次你参加的是一个1小时的项目会议，而会议的前90分钟却都在讨论谁的数字才正确呢?

现在，有了Visual Studio Team System，我们会很严肃地问“有了自动化，我们怎样才能再造我们的核心IT过程呢?我们怎样能消除这些优秀过程中的额外开销呢?我们怎样才能既让这些不同角色的每个人都能有更高的生产力，同时还能把他们整合成一个高效的团队呢?”

谁应该读这本书

本书为那些考虑在软件项目中使用VSTS的软件团队而写。这是一本关于为什么的书，而不是一本关于怎样做的书[1]。VSTS身后的指导思想是什么?为什么这些观点以这些方式来表现?例如，为什么把这么多东西都叫作工作项?度量元仓库度量了些什么?你为什么要使用这些特定的报表?

我一次又一次地体验过：人们在知识、技能和经验上的不同使得软件项目启动时的条件也不一致。有的事情对于有的人来说是一个自证明的真理，而对于另一个人则是一个神话；有些事情对于有的人来说是常识，而对于另一个人则是个发现。各种功能角色往往已被固定在职业阶梯中，而这种对功能角色的自然强调使这一问题更加恶化了。我当然相信是有专家级的开发人员、专家级的测试人员、专家级的架构师、专家级的业务分析师和专家级的项目经理的，但是交付的客户价值要求的是各学科之间的协作。在与其他角色隔离的情况下，去努力优化某一个角色，并不一定会提升客户所能看到的结果的交付。

解决这一矛盾的一种方法是找一个现场教练(onsite coach)来领导团队通过一组一致的过程。教练很棒，但不是每个人都能享受这种奢侈能和教练一起工作。因为我不能发给你一个随需应变的教练，所以我就写了这本书。

本书不是用户手册那样的教程，不会一步一步地告诉你应该以怎样的顺序点击哪里。VSTS中已经带有大量关于这些主题的优秀文档，我会在适当的地方列出这些文档的引用。确切地说，本书提供了一个框架，让我们按照一种可以直接利用VSTS工具的方式来思考软件项目。实际上，我们开发VSTS正是为了能够以这种方式来管理软件项目。

本书也不是一个软件工程文献的纵览。在最近的40年中，已经有了几十种甚至上

百种关于软件工程的著作。这里我就不对它们重述了，并且也不再全部包括那些其他著作已经包括了的材料。我预料到很多专家会批评说我的某些观点在今天是不言而喻的。不幸的是，正如 Freud 所指出的，不言而喻的东西往往根本没有人说。其结果就是：只有当发生了错误的争论时，团队成员之间在假设上的不同才能暴露出来。因此，如果你责怪我陈述了过多显而易见的东西，我承认的确如此。

我展示了足够的 Team System 理论和实践例子，目的是向大多数主流的 IT 项目和团队描述一个实际的过程。对于必须符合 FAA(美国联邦航空管理局)的要求的航空软件来说，这一过程可能不够正式；对于驻扎在车库中的 3 个人的团队来说，这一过程可能又不够松散。

如何读这本书

VSTS 所包含的过程指南被称作微软解决方案框架(Microsoft Solutions Framework，MSF)，其中所包含的核心概念就是一个基于团队同行的团队模型。团队模型还考虑到了不同尺度的专门化。MSF 定义了在成功的项目中必须扮演的 7 类选民(选区)，或者说视点，如图 P-1 所示。MSF 还包括了对于上调和下调伸缩性的建议。在本书中，我始终会强调这些视点，并使用下面这样的一个图标：

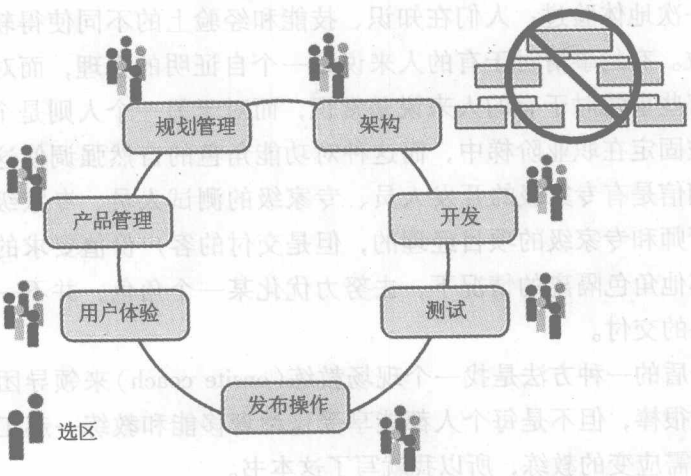


图 P-1　微软方案框架引入了一个团队同行的模型，锚定在一个成功项目所需要表现出的 7 个视点之上。用于 CMMI 过程改进的 MSF 把这 7 个视点细化为 18 个，而用于敏捷软件开发的 MSF 把产品管理和用户体验合并在了一起，从而使用 6 个角色实现了这 7 视点，它同时还提供了将它们减至为 3 个的指南

这本书是写给整个团队的。它展示信息的风格是为了能让所有团队成员感知彼此的视点。不过，本书还是针对角色来划分章节，这样就使读者能够根据自己对特定角色的需要而关注某些章节或略过某些章节。我已经努力将这些主题保持在这样一个级别上：既对团队的所有成员都有吸引力，又对任何人都不晦涩难懂(我的这一选择可能更会使某些人要批评我讲的过于简化)。在当今这个专业化的年代中，你与同事的专长各有不同，而你们之间的约定和对他们的预期也应该至少保持在这样一个级别上，我认为这很重要。如果你很忙，可以根据这些图标阅读那些与你最感兴趣的角色相关的主题。

文档提示

正如我所说的，这不是一本讲如何做的书。在那些需要 VSTS 的细节或它的文档的地方，你会看到一个文档提示，就像下面这个例子：

> **Microsoft Developer Network(MSDN)订阅**
>
> 每套 VSTS 中都包括了一个微软开发者网络(MSDN)的订阅。请从链接 http://msdn.microsoft.com/teamsystem 开始，按照 Reference → Product Documentation 的链接向下走。对于有些术语，你可能想检查一下它们的用法。想要查询它们，请查阅 MSDN 的主题：
>
> Development Tools and Technologies
>
> Visual Studio Team System
>
> Visual Studio Team System Glossary[⊖]

我做出这样的选择是因为我假设在你读本书的大多数时间里，你没有坐在一台电脑前，你只是偶尔才会想要回到电脑桌前去做一些动手操作。当你只是阅读时，你可以跳过这些提示。

⊖ 译者注：若要访问中文 MSDN 中 Visual Studio Team System，请用：
http://www.microsoft.com/china/msdn/vstudio/teamsystem/default.aspx。
VSTS 词汇表在中文 MSDN 中相应的主题为：
Visual Studio 2005
Visual Studio Team System 文档
Visual Studio Team System 词汇表
(http://msdn2.microsoft.com/zh-cn/library/ms242882(VS.80).aspx)

其他人的思想

我在本书中的目标是介绍 VSTS 背后的思想，但不是原样宣读这些思想。重新设计 VSTS 就是为了使不同的过程能够合适地应用到不同的组织和项目上。VSTS 和本书都大量使用了软件社区已制定出的优秀实践。我已经在尾注中尽可能地包括了相应的资源。如果你对于这些参考书目没有兴趣，就不必去读这些尾注了。

在你开始之前需要对 VSTS 了解的内容

本书初稿的评审者曾经抱怨我没有把 VSTS 中有什么解释得足够清楚，因此我就把这个直接来自微软网站的产品介绍放到了下面的板块中。现在已经有了 4 个 VSTS 的客户端产品，并且以后可能还会更多，但是我并没有区分它们，因为价值是在 Team Suite 中的。当然，微软让你按菜单点菜的方式来购买功能，但我想保持简单。

因此，在我写到“VSTS”或“Team System”的时候，我所写的就是指 Team Suite。

微软解决方案框架(Microsoft Solutions Framework，MSF)是 VSTS 的一部分，如图 P-2 中的“过程指导”框所示。MSF 在框外分为两种形式，并能定制成无数的变种。这两个标准形式是：

- 用于敏捷软件开发的 MSF(MSF for Agile Software Development)
- 用于 CMMI 过程改进的 MSF(MSF for CMMI Process Improvement)

稍后我会对这两种 MSF 描述得略微深入一些，但是基本上，如果你的组织对软件过程比较陌生，请从用于敏捷软件开发的 MSF 开始。如果出于地理上的分布、过程改进计划、一致性审计或期望通过 CMMI 评估等原因，你需要更为正式的过程的话，那么你应该考虑用于 CMMI 过程改进的 MSF。

只有在有必要的时候，我才会区别这些产品，除此之外，我将保持关注于它们的公共概念。

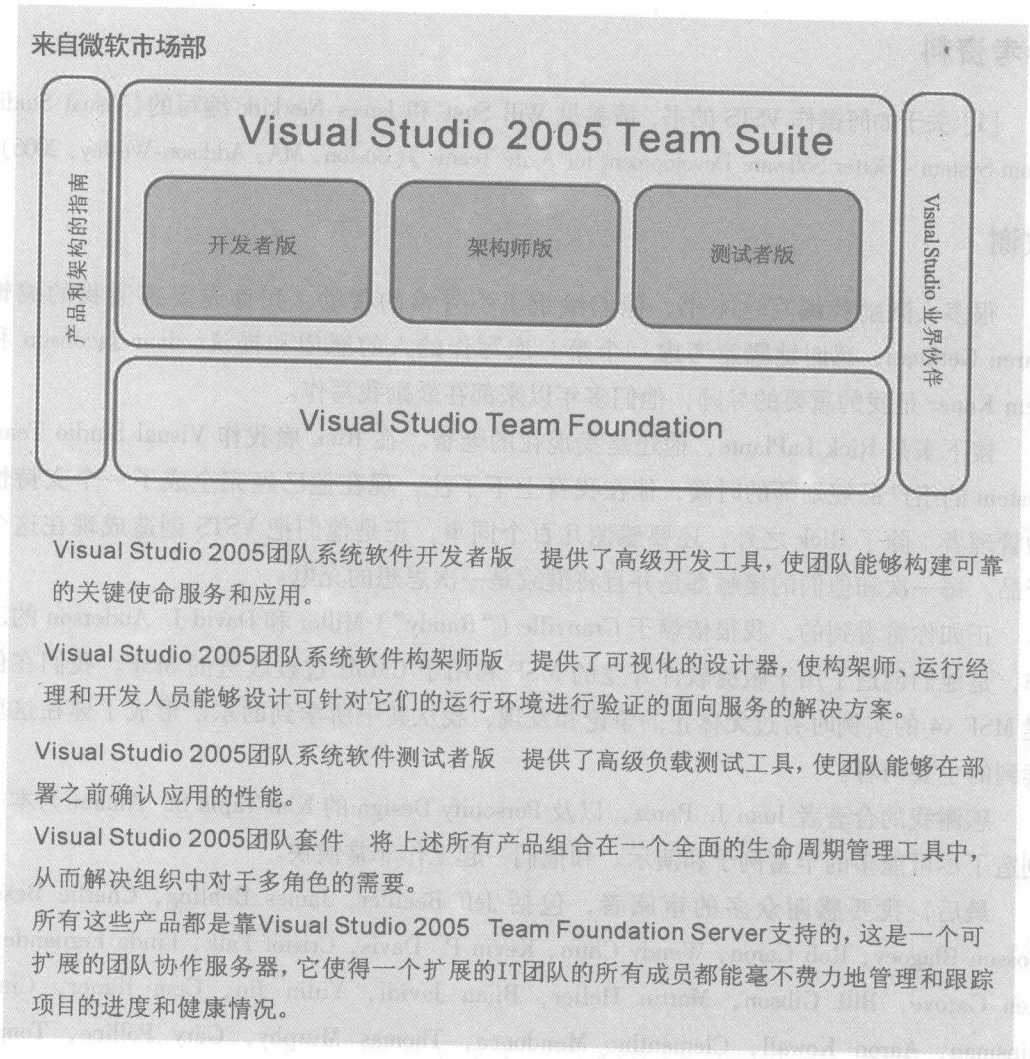


图 P-2 Visual Studio 2005 Team System 由 4 个客户端产品和一个服务器产品组成

声明

最后，我需要澄清一下，本书中的观点仅是我个人的观点，并不一定是微软公司的观点。虽然我是微软的一名雇员，但是我的写作是我的个人行为，不是为了公司而写作。不要为了我在这里所表达的观点和所犯的错误而责备微软(除非你想告诉他们雇佣我是一个错误)，请来向我抱怨。你可以直接来我的博客 http：//blogs. mircosoft. com/sam/与我讨论。

参考资料

[1] 关于如何操作 VSTS 的书，请参见 Will Stott 和 James Newkirk 编写的《Visual Studio Team System - Better Software Development for Agile Teams》(Boston, MA: Addison-Wesley, 2006).

致谢

很多人激励我编写这本书，他们给予了不寻常的帮助。我首先要感谢我的编辑 Karen Gettman，感谢她愿意考虑一个第一次写作的人的愿望和提议。Ivar Jacobson 和 Cem Kaner 是我的重要的导师，他们多年以来都在鼓励我写作。

接下来是 Rick LaPlante，他还是我现在的老板。在 Rick 雇我作 Visual Studio Team System 的组产品规划师的时候，他在我身上下了注，现在他已经完全成了一个支持性的管理者。除了 Rick 之外，还要感谢几百个同事，正是他们把 VSTS 创造成现在这个产品。每一次和他们的接触都是并且将继续是一次思想的充电。

正如你将看到的，我很依赖于 Granville（“Randy”）Miller 和 David J. Anderson 的工作，是他们创造了用于敏捷软件开发的 MSF 和用于 CMMI 过程改进的 MSF。我们在创建 MSF v4 的实例时有过无休止的争论和发现，我从其中所学到的东西形成了你在这里读到的主要内容。

感谢我的合著者 Juan J. Perez，以及 Personify Design 的 Kim Tapia St. Amant 为本书创造了尽可能多的丰富例子和演示。和他们一起工作非常愉快。

最后，我要感谢众多的审阅者，包括 Jeff Beehler, James Behling, Charlie Bess, Rossen Blagoev, Rob Caron, Wendy Chun, Kevin P. Davis, Cristof Falk, Linda Fernandez, Ken Garove, Bill Gibson, Martin Heller, Bijan Javidi, Yulin Jin, Cem Kaner, Chris Kinsman, Aaron Kowall, Clementino Mendonca, Thomas Murphy, Gary Pollice, Tomas Restrepo, Johanna Rothman, Joel Semeniuk, Will Stott, Dan Sullivan, David Trowbridge, Mike Turner, Kumar Vadaparty 和 Peter Williams。Addison-Wesley 的 Kim Boedigheimer, Ben Lawson 和 Michael Thurston 在最后阶段给了我极大的帮助。如果没有所有这些审阅者的建议和意见，本书可能只有现在篇幅的一小部分。

Sam Guckenheimer

华盛顿州雷德蒙德

2006 年 1 月

目　录