作者：希赛教育软考学院 来源：希赛网 2014年01月27日

CMM与软件过程改进

第12章 软件新技术简介

本章主要介绍软件的新技术。

12.1 CMM与软件过程改进

CMM是软件过程能力成熟度模型（Capacity Maturity Model）的简称，耐基梅隆大学软件工程研究所（CMU/SEI）为了满足美国联邦政府评估软件供应商能力的1986年开始研究的模型，并于1991年正式推出了CMM 1.0 版。CMM自问世以来备受关注发达国家和地区得到了广泛应用，成为衡量软件企业软件开发和管理水平的重要参考因素，以及软件过程改进事实上的工业标准。据了解，美国、印度、日本等国家已有数十家公司通过了CMM不同等级的认证。中国政府自2000年加强对软件企业的重视，大力推崇CMM以来，已经有50多家企业先后通过了CMM各种级别的认证。

1992年4月，SEI举行了一个CMM的研讨会，参加研讨会的有大约200名富有经验的软件专家。SEI在广泛听取他们的意见之后，又于1993年推出 CMM1.1版。这也是目前世界上比较流行和通用的CMM版本。

十几年来，此项工作一直在不断进行。按照SEI原来的计划，CMM的改进版本2.0应该在1997年11月完成，然后在取得版本2.0的实践反馈意见之后，在1999年完成准CMM 2.0版本。但是，美国国防部办公室要求SEI推迟发布CMM 2.0版本，而要先完成一个更为紧迫的项目CMMI。

上一节 本书简介 下一节

作者：希赛教育软考学院 来源：希赛网 2014年01月27日

CMM基本概念

12.1.1 CMM基本概念

为了行文方便，我们在本节介绍CMM中用到的有关概念和术语。

过程（Process）：为实现既定目标的一系列操作步骤。

软件过程（Software Process）：指人们用于开发和维护软件及其相关产品的一系列活动、方法、实践和革新。其中相关产品是指项目计划、设计文档、编码、测试和用户手册。当一个企业逐步走向成熟，软件过程的定义也会日趋完善，其企业内部的过程实施将更具有一致性。

软件过程能力（Software Process Capability）：描述了在遵循一个软件过程后能够得到的预期结果的界限范围。该指标是对能力的一种衡量，用它可以预测一个组织在承接下一个软件项目

时，所能期望得到的最可能的结果。

软件过程性能（Software Process Performance）：表示遵循一个软件过程后所得到的实际结果。软件过程性能与软件过程能力有区别，软件过程性能关注的是实际得到的结果，而软件过程能力关注的是期望得到的结果。由于项目要求和客观环境的差异，软件过程性能不可能充分反应软件过程整体能力，即软件过程性能受限于它的环境。

软件过程成熟度（Software Process Maturity）：是指一个具体的软件过程被明确地定义、管理、评价、控制和产生实效的程度。所谓成熟度，包含着能力的一种增长潜力，同时也表明了组织实施软件过程的实际水平。随着组织软件过程成熟度能力的不断提高，组织内部通过对过程的规范化和对成员的技术培训，软件过程也将会被使用者关注和不断修改完善。从而使软件的质量、生产率和生产周期得到改善。

关键过程（区）域（Key Process area）：是指一系列相互关联的操作活动，这些活动反映了一个软件组织改进软件过程时所必须满足的条件。也就是说，关键过程域标识了达到某个成熟程度级别时所必须满足的条件。在CMM中一共有18个关键过程域，分布在第二级至第五级中。

关键实践（Key Practices）：是指关键过程域中的一些主要实践活动。每个关键过程域最终由关键实践所组成，通过实现这些关键实践达到关键过程域的目标。一般情况下，关键实践描述了该"做什么",但没有规定"如何"去达到这些目标。

软件过程评估（Software Process Assessment）：是用来判断一个组织当前所涉及的软件过程的能力状态，判断下一个组织所面向的更高层次上的与软件过程相关的课题，以及利用组织的鼎力支持来对该组织的软件过程进行有效的改进。

软件能力评价（Software Capability Appraisal）：是用来判断有意承担某个软件项目的软件组织的软件过程能力，或是判断已进行的软件过程所处的状态是否正确或是否正常。

软件工程组（Software Engineering Group）：负责一个项目的软件开发和维护活动的团体。活动包括需求分析、设计、编码和测试等。

软件相关组（Software Related Groups）：代表一种软件工程项目的团体，它支持但不直接负责软件开发或维护工作，如软件质量保证组、软件配置管理组和软件工程过程组等。在CMM的关键实践中，软件相关组通常应该根据关键过程域和组织的上下文来理解。

软件工程过程组（Software Engineering Process Group）：是由专家组成的组，他们推进组织采用的软件过程的定义、维护和改进工作。在关键实践中，这个组织通常指"负责组织软件过程活动的组".

系统工程组（System Engineering Group）：是负责下列工作的个人或团体，分析系统需求；将系统需求分配给硬件、软件和其他成分；规定硬件、软件和其他成分的界面；监控这些成分的设计和开发以保证它们符合其规格说明。

系统测试组（System Test Group）：是一些负责策划和完成独立的软件系统测试的团体，测试的目的是为了确定软件产品是否满足对它的需求。

软件质量保证组（Software Quality Assurance Group）：是一些计划和实施项目的质量保证的团体，其工作目的是保证软件过程的步骤和标准是否得到遵守。

软件配置管理组（Software Configuration Management Group）：是一些负责策划、协调和实施软件项目的正式配置活动的团体。

培训组（Training Group）：是一些负责协调和安排组织培训活动的团体。通常这个组织负责准备和讲授大多数培训课程并协调其他培训方式的使用。

　作者：希赛教育软考学院　来源：希赛网　2014年01月27日

CMM的基本框架

12.1.2 CMM的基本框架

1.分级标准

CMM模型描述和分析了软件过程能力的发展程度，确立了一个软件过程成熟程度的分级标准，如图12-1所示。

初始级：软件过程的特点是无秩序的，有时甚至是混乱的。软件过程定义几乎处于无章法和步骤可循的状态，软件产品所取得的成功往往依赖极个别人的努力和机遇。初始级的软件过程是未加定义的随意过程，项目的执行是随意甚至是混乱的。也许，有些企业制定了一些软件工程规范，但若这些规范未能覆盖基本的关键过程要求，且执行没有政策、资源等方面的保证时，那么它仍然被视为初始级。

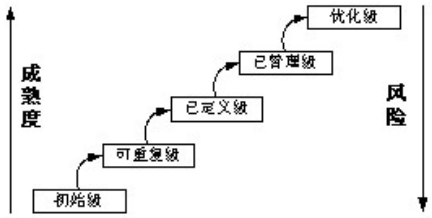

图12-1 软件过程成熟度的级别

可重复级：已经建立了基本的项目管理过程，可用于对成本、进度和功能特性进行跟踪。对类似的应用项目，有章可循并能重复以往所取得的成功。焦点集中在软件管理过程上。一个可管理的过程则是一个可重复的过程，一个可重复的过程则能逐渐演化和成熟。从管理角度可以看到一个按计划执行的、且阶段可控的软件开发过程。

已定义级：用于管理的和工程的软件过程均已文档化、标准化，并形成整个软件组织的标准软件过程。全部项目均采用与实际情况相吻合的、适当修改后的标准软件过程来进行操作。要求制定企业范围的工程化标准，而且无论是管理还是工程开发都需要一套文档化的标准，并将这些标准集成到企业软件开发标准过程中去。所有开发的项目需根据这个标准过程，剪裁出项目适宜的过程，并执行这些过程。过程的剪裁不是随意的，在使用前需经过企业有关人员的批准。

已管理级：软件过程和产品质量有详细的度量标准。软件过程和产品质量得到了定量的认识和控制。已管理级的管理是量化的管理。所有过程需建立相应的度量方式，所有产品的质量（包括工作产品和提交给用户的产品）需有明确的度量指标。这些度量应是详尽的，且可用于理解和控制软件过程和产品，量化控制将使软件开发真正变成为一个工业生产活动。

优化级：通过对来自过程、新概念和新技术等方面的各种有用信息的定量分析，能够不断地、

持续地进行过程改进。如果一个企业达到了这一级，表明该企业能够根据实际的项目性质、技术等因素，不断调整软件生产过程以求达到最佳。

除第一级外，每一级都设定了一组目标，如果达到了这组目标，则表明达到了这个成熟级别，自然可以向上一级别迈进。因为从第二级开始，每一个低级别的实现均是高级别实现的基础，所以CMM体系不主张跨级别的演化。SEI建议，从低一级别向高一级别演化的时间需要在12~30个月之间。

CMM分级标准有两个方面的用途。一方面，软件组织利用它可以评估自己当前的过程成熟度，并以此提出严格的软件质量标准和过程改进的方法和策略，通过不断的努力去达到更高的成熟程度。另一方面，该标准也可以作为用户对软件组织的一种评价标准，使之在选择软件开发商时不再是盲目的和无把握的。

2.CMM的主要内容

CMM为软件企业的过程能力提供了一个阶梯式的演化框架，它采用分层的方式来解释其组成部分。在第二至第五个成熟等级中，每个等级包含一个内部结构的概念。

每一级向上一级迈进的过程中都有其特定的改进计划，具体情况如图12-2所示。

初始级的改进方向：建立项目过程管理，实施规范化管理，保障项目的承诺；进行需求管理方面的工作，建立用户与软件项目之间的沟通，使项目真正反映用户的需求；建立各种软件项目计划，如软件开发计划、软件质量保证计划、软件配置管理计划、软件测试计划、风险管理计划及过程改进计划等；积极开展软件质量保证活动（SQA）。

可重复级的改进方向：不再按项目制定软件过程，而是总结各种项目的成功经验，使之规则化，把具体经验归纳为组织的标准软件过程，把改进软件组织的整体软件过程能力的软件过程活动，作为软件开发组织的责任；确定全组织的标准软件过程，把软件工程及管理活动集成到一个稳固确定的软件过程中，从而可以跨项目改进软件过程，也可以作为软件过程剪裁的基础；建立软件工程过程小组（SPEG），长期承担评估与调整软件过程的任务，以适应未来软件项目的要求；积累数据，建立组织的软件过程库及软件过程相关的文档；加强培训。

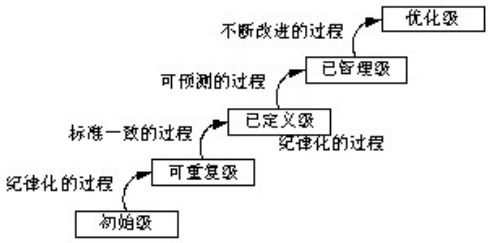

图12-2 不断改进的过程

已定义级的改进方向：着手软件过程的定量分析，已达到定量地控制软件项目过程的效果；通过软件的质量管理达到软件质量的目标。

已管理级的改进方向：防范缺陷，不仅在发现了问题能及时改进，而且应采取特定行动防止将来出现这类缺陷；主动进行技术改革管理、标识、选择和评价新技术，使有效的新技术能在开发组织中实施；进行过程变更管理，定义过程改进的目的，经常不断地进行过程改进。

优化级的改进方向：保持持续不断的软件过程改进。

3.CMM的内部结构

在CMM的五个成熟度等级中，除第一级外，每一级按完全相同的内部结构构成，如图12-3所示。

图12-3 CMM的内部结构图

成熟度等级为顶层，不同的成熟度等级反映了软件组织的软件过程能力和该组织可能实现预期结果的程度。

在CMM中，每个成熟度等级（第一级除外）规定了不同的关键过程域，一个软件组织如果希望达到某一个成熟度级别，就必须完全满足关键过程域所规定的要求，即满足关键过程域的目标。每个级别对应的关键过程域（KPA）见表12-1.

公共特性是把每个KPA的所有关键实践按照它们的属性进行分组。无论哪个KPA,它们的关键实施都统一按5个公共属性进行组织，分别是执行约定、执行能力、实施活动、度量和分析、实施验证。

过程分类 / 等级	管 理 方 面	组 织 方 面	工 程 方 面
优化级		技术改进管理 过程改进管理	缺陷预防
可管理级	定量管理过程		软件质量管理
已定义级	集成软件管理 组间协调	组织过程焦点 组织过程定义 培训程序	软件产品工程 同级评审
可重复级	需求管理 软件项目计划 软件项目跟踪与监控 软件子合同管理 软件质量保证 软件配置管理		

表12-1 关键过程域的分类

执行约定（Commitment To Perform）：也称实施保证，是企业为了建立和实施相应KPA所必须采取的行动，这些行动主要牵涉到企业范围的政策和高层管理的责任。执行约定一般与组织的方针政策和管理方式有关。

执行能力（ability to perform）：也称实施能力，描述了为使某软件过程得以始终如一地执行，必须在项目或企业中存在先决条件，是企业实施KPA的前提条件。企业必须采取措施，在满足了这些条件后，才有可能执行KPA的实践活动。执行能力关注于项目计划的实践、资源的配置、责任的布置与授权，以及各种有关的培训等。

实施活动（activities perform）：描述了执行KPA所需的必要行动、任务和步骤。在五个公共属性中，实施活动是唯一与项目执行相关的属性，其余四个属性则涉及企业CMM能力基础设施的建立。实施活动一般包括计划、执行的任务、任务执行的跟踪等。

度量和分析（measurement and analysis）：关注KPA的活动需要做的度量和度量分析要求。典型的度量和度量分析的要求是确定执行活动的状态和执行活动的有效性。度量与分析一般包括一些度量的例子，通过这些例子可以知道如何确定操作活动的状态和效果。

实施验证（verifying implementation）：实施验证是验证执行活动是否与建立的过程一致，

核实以确保所实施的过程是按照原定的计划及其要达到的目标，着眼于保证过程的实现要通过独立的个人和高级管理人员验证。验证实施涉及到管理的评审和审计及质量保证活动，包括过程执行的确保，产品要求的确保，高层管理人员进行的审核和项目经理进行的审核。

软件过程评估和软件能力评价

12.1.3 软件过程评估和软件能力评价

软件过程评估（SPE）所针对的是软件组织自身内部软件过程的改进问题，目的在于发现缺陷，提出改进方向。评估组以CMM模型为指引调查、鉴别软件过程中的问题，反过来将这些问题与CMM关键实践活动所提出的指导一起用于确定组织的软件过程改进策略。

软件能力评价（SCE）是对接受评价者在一定条件下、规定时间内能否完成特定项目的能力考核，即承担风险的系数大小。评价包括承包者是否有能力按计划开发软件产品，是否能按预算完成等。通过利用CMM模型确定评价结果后，就可以利用这些结果确定选择某一承包商的风险。也可以用来判断承包者的工作进程，推动他们改进软件过程。

CMM为评估和评价提供了一个参考框架，指出了在评估和评价中通常采用的步骤如图12-4所示。

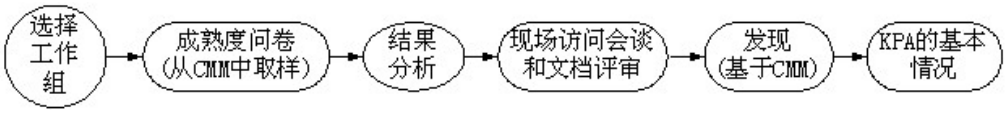

图12-4 软件过程评估和软件能力评价的步骤

具体来说，评估过程是：选择一个工作组；完成问卷调查和取样工作；结果分析；现场访问；与CMM模型对照分析；依据KPA的基本情况列出评估提纲。

尽管SPE和SCE有很多相似之处，但由于其目的和结果的不同，它们之间的差异也是必然存在的，如：

SPE和SCE在出发点和目标上的不同，使得会谈目的、调查范围、收集的信息和输出的表示方式上有着本质的不同。尤其在一些细节规范方面，SPE和SCE的方法有很大差异。

SPE和SCE的结果所起的作用不同。因为两者的侧重点不一样，即使是对同一个应用项目，运用相同的方法，也不会得出相同的结果。

被评估和评价单位的态度对SPE和SCE活动的影响。SPE在某种意义上被评估单位的态度较积极，而SCE在某种意义上被评价单位的态度可能比较慎重。SPE是在一个开放的、互相协作的环境中进行的，而SCE往往是在有较大的阻力的环境中进行的。

一般来说，CMM 的评估分为以下3个整体阶段：

计划和准备阶段。此阶段的主要工作是聘请评估公司来帮助组织分析评估需求、建立评估小组、制订评估计划、开展培训、规范和完善过程文档、开展预评估及完善。一般历时2~3个月。

正式评估阶段。此阶段的工作是评审小组进行数据的采集和记录、问卷调查、文档审查、面谈

访问、数据确认，最后作出级别评定。一般在计划和准备阶段完成后的8~12个月进行，历时1到2周。

报告结果阶段。此阶段的工作是报告评估结果、保护评估结果的机密性、？评估记录的保存，时间为一周。

作者：希赛教育软考学院　来源：希赛网　2014年01月27日

CMMI综述

12.1.4 CMMI综述

CMMI（Capability Maturity Model Integration,能力成熟度模型集成）是CMM模型的最新版本。早期的CMMI（CMMI-SE/SW/IPPD）1.02版本是应用于软件业项目的管理方法，SEI在部分国家和地区开始推广和试用。随着应用的推广与模型本身的发展，演绎成为一种被广泛应用的综合性模型。2001年12月，SEI正式发布了CMMI 1.1版本。与原有的能力成熟度相比，CMMI涉及面更广，专业领域覆盖软件工程、系统工程、集成产品开发和系统采购。据美国国防部资料显示，运用CMMI模型管理的项目，不仅降低了项目的成本，而且提高了项目的质量与按期完成率。

CMMI可以看做是把各种CMM集成到一个系列的模型中，CMMI的基础源模型包括软件CMM 2.0版（草稿C）、EIA-731系统工程，以及集成化产品和过程开发IPD CMM（IPD）0.98a版。CMMI也描述了五个不同的成熟度级别。

1.CMMI模型的表示

每一种CMMI模型都有两种表示法：阶段式和连续式。这是因为在CMMI的三个源模型中，CMM是"阶段式"模型，系统工程能力模型是"连续式"模型，而集成产品开发（IPD）CMM是一个混合模型，组合了阶段式和连续式两者的特点。两种表示法在以前的使用中各有优势，都有很多支持者，因此，CMMI产品开发群组在集成这三种模型时，为了避免由于淘汰其中任何一种表示法而失去对CMMI支持的风险，并没有选择单一的结构表示法，而是为每一个CMMI都推出了两种不同表示法的版本。

不同表示法的模型具有不同的结构。连续式表示法强调的是单个过程域的能力，从过程域的角度考查基线和度量结果的改善，其关键术语是"能力";而阶段式表示法强调的是组织的成熟度，从过程域集合的角度考查整个组织的过程成熟度阶段，其关键术语是"成熟度".

尽管两种表示法的模型在结构上有所不同，但CMMI产品开发群组仍然尽最大努力确保了两者在逻辑上的一致性，二者的需要构件和期望部件基本上都是一样的。过程域、目标在两种表示法中都一样，特定实践和共性实践在两种表示法中也不存在根本区别。因此，模型的两种表示法并不存在本质上的不同。组织在进行集成化过程改进时，可以从实用角度出发选择某一种偏爱的表示法，而不必从哲学角度考虑两种表示法之间的差异。

阶段式模型也把组织分为5个不同的级别。

级别1（初始级）代表了以不可预测结果为特征的过程成熟度。过程包括了一些特别的方法、符号、工作和反映管理，成功主要取决于团队的技能。

级别2（已管理级）代表了以可重复项目执行为特征的过程成熟度。组织使用基本纪律进行需求管理、项目计划、项目监督和控制、供应商协议管理、产品和过程质量保证、配置管理，以及度量和分析。对于级别2而言，主要的过程焦点在于项目级的活动和实践。

级别3（严格定义级）代表了以组织内改进项目执行为特征的过程成熟度。强调级别3的关键过程域前后一致的、项目级的纪律，以建立组织级的活动和实践。

级别4（定量管理级）代表了以改进组织性能为特征的过程成熟度。4级项目的历史结果可用来交替使用，在业务表现的竞争尺度（成本、质量、时间）方面的结果是可预测的。

级别5（优化级）代表了以可快速进行重新配置的组织性能，与定量的、持续的过程改进为特征的过程成熟度。

2.CMMI的目标和优点

CMMI的具体目标是：

改进组织的过程，提高对产品开发和维护的管理能力。

给出能支持将来集成其他科目CMM的公共框架。

确保所开发的全部有关产品符合将要发布的关于软件过程改进的国际标准ISO/IEC15504对软件过程评估的要求。

使用在CMMI框架内开发的模型具有下列优点：

过程改进能扩展到整个企业级；

先前各模型之间的不一致和矛盾将得到解决；

既有分级的模型表示，也有连续的模型表示，任你选用；

原先单科目过程改进的工作可与其他科目的过程改进工作结合起来；

基于CMMI的评估将与组织原先评估得分相协调，从而保护当前的投资。并与ISO/IEC15504评估结果相一致；

节省费用，特别是当要运用多科目改进时，以及进行相关的培训和评估时；

鼓励组织内各科目之间进行沟通和交流。

3.CMMI评估

CMMI产品集中的CMMI Appraisal Requirement 版本1.1给出了基于CMMI的评估中40多条需求，提供了一个综合需求集和评估方法的设计限制。根据这些需求集和设计限制可以开发出相应的基于CMMI的评估方法。

CMMI产品集中还给出了过程改进的标准方法CMMI评估方法SCAMPI,这一方法是由CMMI产品开发群组开发的，满足全部的CMMI Appraisal Requirement 版本1.1需求。组织在进行过程改进的时候可以参考该方法进行具体实施。

但是，对于一个具体的组织来说，仅仅根据CMMI产品中对评估的相应描述来照本宣科是远远不够的。特别是对于一些在过程管理方面还很不成熟的组织，在开展CMMI评估的时候更要发挥自身的创造性，灵活运用CMMI产品集中所提供的评估方法。根据CMMI Appraisal Requirement 版本1.1中的描述，评估可分为三类。

A类评估：全面综合的评估方法，要求在评估中全面覆盖评估中所使用的模型，并且在评估结果

中提供对组织的成熟度等级的评定结果。

B类评估：较少综合，花费也较少。在开始时进行部分自我评估，并集中于需要关注的过程域。不评定组织的成熟度等级。

C类评估：也称为快估。主要是检查特定的风险域，找出过程中的问题所在。该类评估花费很少，需要的培训工作也不多。

12.1.5 PSP、TSP、CMM之间的关系

个体软件过程（Personal Software Process,PSP）CMU/SEI是Watts S. Humphrey 领导开发的，于1995年公布，是一种可用于控制、管理和改进个人工作方式的自我改善过程，是一个包括软件开发表格、指南和规程的结构化框架。

小组软件过程（Personal Software Process,TSP）也是由CMU/SEI提出的，讲述了如何创建高效且具有自我管理能力的项目小组，开发人员如何才能成为合格的项目组成员，管理人员如何对小组提供指导和支持，如何保持良好的工程环境使项目组能充分发挥自己的水平等软件工程管理问题

CMM偏重的是软件组织的宏观过程规范完善，其实现最终依赖于组织中个体成员的能力、参与和创造，但CMM并未提供有关实现CMM各关键过程域所需要的具体的知识和技能；TSP侧重于组织中的各个项目团队组织，为开发软件产品的开发团队提供指导，保证单个项目和团队的成功；PSP侧重于企业中有关软件过程的微观优化，面向软件开发的个体人员。所以，CMM、TSP、PSP三者相互支持、相互配合、各有侧重，形成一个不可分割的整体，如果软件企业还没有实施CMM,可以有计划地开展PSP、TSP,为今后的CMM实施提供良好的基础。

CMM、PSP和TSP为软件产业提供了一个集成化的、三维的软件过程改进框架。要特别注意的是，如果一个组织单纯实施CMM,永远不能真正做到能力成熟度的升级，而需要将实施CMM与实施PSP和实施TSP有机地结合起来，才能达到软件过程持续改善的效果。

CMM、PSP和TSP组成的软件过程框架如图12-5所示。

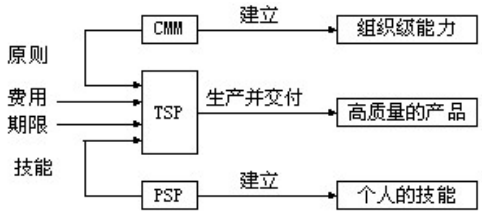

图12-5 软件过程框架图

从图12-5可以看出，CMM是过程改善的第一步，它提供了评价组织能力的方式，并为TSP提供了指导原则。企业只有开始实施CMM后，才能认识到质量的重要性，注重对员工进行培训，合理分配项目人员。PSP能够指导软件工程师如何保证自己的工作质量，估计和规划自身的工作，度量和追踪个人的表现，管理自身的软件过程和产品质量。PSP为TSP的实施提供了软件工程师的个人技能。

TSP结合了CMM的管理方法和PSP的工程技能，通过将TSP与组织相联系，向组织展示如何应用CMM的原则和PSP的技能去生产高质量的产品。

CMM的18个关键过程域与PSP和TSP的对应关系如表12-2所示。

级　别	CMM 的 18 个关键过程域	提　供　者
优化级	缺陷预防	PSP
	技术变更管理	PSP
	过程变更管理	PSP
可管理级	定量的过程管理	PSP
	软件质量管理	PSP
已定义级	组织过程焦点	PSP
	组织过程定义	PSP
	培训大纲	无
	集成软件管理	PSP
	软件产品工程	PSP
	组织协调	TSP
	同行专家评审	PSP
可重复级	需求管理	TSP
	软件项目规划	PSP
	软件项目追踪和监控	PSP
	软件子合同管理	无
	软件质量保证	TSP
	软件配置管理	TSP

表12-2　CMM的关键过程域与PSP和TSP的对应关系

上一节　本书简介　下一节

可扩展标记语言

12.2　可扩展标记语言

本节主要介绍可扩展标记语言。

上一节　本书简介　下一节

SGML、HTML与XML的比较

12.2.1　SGML、HTML与XML的比较

标准通用标记语言SGML（Standard Generalized Markup Language）是电子文档的国际化标准，其功能十分庞大，但其十分复杂，且不易掌握使用，为此出现了超文本标记语言

HTML（Hypertext Markup Language），HTML是SGML中的一个子集，其使Web页面的编写、发布、浏览十分简单，促进了Internet的高速发展。但HTML是面向描述的，而非面向对象的，且标签是固定的，不具有扩展性，存在着很大的局限性。于是可扩展标记语言XML（Extensible Markup Language）应运而生，是SGML的一个子集，融合了两者的优点。

虽然XML与HTML都是标记语言，但它们在结构和应用上有很大的区别。

HTML是一种格式化的语言，一个HTML文本可以看做一个格式化的程序。HTML（及类似的）语言定义了一套固定的标记，用来描述一定数目的元素。如果标记语言中没有所需的标记，用户也就没有办法了。这时只好等待标记语言的下一个版本，希望在新版本中能够包括所需的标记，但是这样一来就得依赖于软件开发商的选择了。另外，用该语言描述的程序或文本具有"内容+格式"的双重属性。一个HTML在不同平台、不同浏览器上的表现是一模一样的。而一段符合XML语法规范的文本则是一段"纯"的数据，它的结构由其他的称为DTD（Document Type Description）的文本来描述；而它的处理则可能是任何其他支持XML的容器或程序，例如IE浏览器依据相关的CSS或XSL文件来显示XML"数据";开发人员可以用来自Microsoft、IBM、Sun、BEA、Inprise等厂商的任何支持XML的开发工具开发自己的XML处理程序。

与HTML相比的另一个不同是，XML是一种元标记语言。它可以被用于定义其他的标记语言，甚至DTD和XSL文档也是用XML语法描述的。例如，在Peter Murray-Rust的Chemical Markup Language（化学标记语言，简写为CML）中的MOL.DTD文件描述了词汇表和分子科学的句法：其中包括chemistry（化学）、crystallography（结晶学）、solid state physics（固体物理）等词汇。它包括用于atoms（原子）、molecules（分子）、bonds（化学键）、spectra（光谱）等的标记。这个DTD可与分子科学领域中的许多不同的人共享。对于其他领域也有其他的DTD,用户还可以创建自己的DTD.

XML定义了一套元句法，与特定领域有关的标记语言（如MusicML、MathML和CML）都必须遵守。如果一个应用程序可以理解这一元句法，那么它也就自动地能够理解所有的由此元语言建立起来的语言。浏览器不必事先了解多种不同的标记语言使用的每个标记。事实上，浏览器在读入文档或是它的DTD时才了解了给定文档使用的标记。

科学家、音乐家可能还必须等待多年，才能让浏览器的开发商支持书写最基本的数学公式和乐谱所需的标记，因为Netscape Navigator和Internet Explorer还都不支持乐谱。

但有了XML就意味着不必等待浏览器的开发商来满足用户的需要了。用户可以创建自己需要的标记，当需要时，告诉浏览器如何显示这些标记就可以了。

上一节　本书简介　下一节

XML语法简介

12.2.2 XML语法简介

XML文档由标记和内容组成。XML中共有六种标记：元素（elements），属性（attributes），实体引用（entity references），注释（comments），处理指令（processing instructions）和CDATA段（CDATA sections）。其中，元素是标记的最常见的形式，由尖括号分隔，和人们熟悉的HTML中的标记看起来没什么两样。

XML声明：XML声明必须是文件中的第一项，以"<?xml"开始，以"?>"结束，其中还应包含XML的版本信息，既version ="1.0".

注释：XML文件的注释以"<!--"开始，以"-->"结束。注释的内容全都会被XML处理器所忽略。

标记：XML中的标记必须匹配使用，既每个起始标记必须和一个结束标记匹配。如果是空标记可以嵌套，但不可以交叠。

内容：就是文本，由字符内容组成。

下面就是一个简单的XML例子：

<?xml version-"1.0">

<!-- 这是一个XML的例子 -->

<通信录>

〈人员〉

<姓名> 张平 </姓名>

<地址> 南大街一号</地址>

</人员>

〈人员〉

<姓名> 王强</姓名>

<地址> 北大街二号</地址>

</人员>

</通信录>

上一节　本书简介　下一节

XML的应用

12.2.3 XML的应用

由于XML的特性，其应用是非常广泛的，现将其常用的场合总结如下。

应用于客户需要与不同的数据源进行交互时。数据可能来自不同的数据库，他们都有各自不同的复杂格式。但客户与这些数据库间只通过一种标准语言进行交互，那就是XML.由于XML的自定义性及可扩展性，它足以表达各种类型的数据。客户收到数据后可以进行处理，也可以在不同数据库间进行传递。也就是说在这种情况下，XML解决了数据的统一接口问题。

应用于将大量运算负荷分布在客户端。即客户可根据自己的需求选择和制作不同的应用程序以

处理数据，而服务器只须发出同一个XML文件。如按传统的"客户机/服务器"工作方式，客户机向服务器发出不同的请求，服务器分别予以响应，这不仅加重了服务器本身的负荷，而且网络管理者还须事先调查各种不同的用户需求以做出相应不同的程序，但假如用户的需求繁杂而多变，则仍然将所有业务逻辑集中在服务器端是不合适的，因为服务器端的编程人员可能来不及满足众多的应用需求，也来不及跟上需求的变化，双方都很被动。应用XML则将处理数据的主动权交给了客户，服务器所做的只是尽可能完善、准确地将数据封装进XML文件中，正是各取所需、各司其职。XML的自解释性使客户端在收到数据的同时也理解数据的逻辑结构与含义，从而使广泛、通用的分布式计算成为可能。

应用于将同一数据以不同的面貌展现给不现的用户。它类似于同一个剧本，我们却可以用电视剧、电影、话剧、动画片等不同形式表现出来。这一应用将会为网络用户界面个性化、风格化的发展铺平道路。

应用于网络代理对所取得的信息进行编辑、增减以适应个人用户的需要。有些客户取得数据并不是为了直接使用而是为了根据需要组织自己的数据库。比方说，教育部建立一个庞大的题库，考试时将题库中的题目取出若干组成试卷，再将试卷封装进XML文件，接下来便是最精彩的部分，在各个学校让其通过一个过滤器，滤掉所有的答案，再发送到各个考生面前，未经过滤的内容则可直接送到老师手中，当然考试过后还可以再传送一份答案汇编。此外，XML文件中还可以包含进诸如难度系数、往年错误率等其他相关信息，这样只需几个小程序，同一个XML文件便可变成多个文件传送到不同的用户手中。

上一节　本书简介　下一节

工作流

12.3 工作流

本节主要介绍工作流。

上一节　本书简介　下一节

工作流概述

12.3.1 工作流概述

在工作流方面，目前的权威性机构是"工作流管理联盟"（Workflow Management Coalition,WFMC）。它成立于1993年8月，目前已拥有130余个成员，成员包括工作流产品的供应

者、应用者，有关大学、研究机构和个人，是一个国际性的非赢利组织。在最近的投资成员（Funding members）清单中，可以看到诸如Baan、HP、IBM、Microsoft、Oracle、Peoplesoft、SAP AG、Xerox等机构。

根据WFMC的定义，工作流（Work Flow）就是自动运作的业务过程部分或整体，表现为参与者对文件、信息或任务按照规程采取行动，并令其在参与者之间传递。简单地说，工作流就是一系列相互衔接、自动进行的业务活动或任务。我们可以将整个业务过程看做是一条河，其中流过的就是工作流。

工作流管理（Workflow Management,WFM）是人与电脑共同工作的自动化协调、控制和通信，在电脑化的业务过程上，通过在网络上运行软件，使所有命令的执行都处于受控状态。在工作流管理下，工作量可以被监督，分派工作到不同的用户，以达成平衡。

根据WFMC的定义，工作流管理系统（Workflow Management System,WFMS）通过软件定义、创建工作流并管理其执行。它运行在一个或多个工作流引擎上，这些引擎解释对过程的定义，与工作流的参与者（包括人或软件）相互作用，并根据需要调用其他的IT工具或应用。

总体来说，实际企业中运作的工作流管理系统，是一个"人-电脑"结合的系统。它的基本功能体现在几个方面。

定义工作流，包括具体的活动、规则等，这些定义是同时被人及电脑所"理解"的。

遵循定义创建和运行实际的工作流。

监察、控制、管理运行中的业务（工作流），例如任务、工作量与进度的检查、平衡等。

工作流管理系统适用于电信、物流、进出口贸易、制造业、政府机关等基本业务需要众多环节流转，环环相扣的行业。除了上述业务流程的管理，当然工作流也适用于任何企业的办公自动化等内部流程的管理流域。下面举出工作流通常使用的一些场合。

企业业务流程的管理。订单管理、报价处理、采购处理、客户投诉、售后服务等。

内部管理流程的管理。财务方面的付款请求、应收款处理、差旅报销，出差、加班、请假申请原来手工流转处理的行政性申请表单。

其他应用类。贸易公司报关、物流公司货物跟踪、新产品信息跟踪处理、Bug管理等各种通过表单逐级手工流转完成的任务。

企业利用工作流管理系统可以有效地梳理和提高内部管理和外部服务流程，实现信息系统倍增器的作用，总结其对企业的贡献如下。

优化业务流程，提高工作效率。通过工作流增强业务各环节的协作能力，使业务运作更加顺畅，缩短了业务处理周期。

规范业务流程，提高服务质量。按照既定的业务规则管理和监督业务的运行，避免传统处理方式中的随意性造成业务流程混乱。并可在工作流中及时发现业务瓶颈，进行疏导或改善业务流程，实现及时有效的业务过程控制。

提高业务流程柔性，增加应变能力。通过方便灵活的流程定义工具，提高业务流程的灵活性，及时满足市场和客户的需求。

工作流系统实现

12.3.2 工作流系统实现

工作流管理联盟（WFMC）提出了一个工作流参考模型，约定了工作流系统的体系结构、应用接口及特性，主要目的是为了实现工作流技术的标准化和开放性。下面简要介绍系统中的各个部分，并对参考模型中的五类接口进行描述。

1.工作流管理系统中的各种数据

工作流控制数据（Workflow Control Data）：工作流执行服务/工作流机通过内部的工作流控制数据来辨别单个过程或活动实例的状态。这些数据由工作流执行服务/工作流机控制。用户、应用程序或其他的工作流机/工作流执行服务不能对其进行直接读写操作，它们可以通过向工作流执行服务/工作流机发送消息来获得工作流控制数据的内容。

工作流相关数据（Workflow Relevant Data）：工作流管理系统通过工作流相关数据来确定过程实例状态转换的条件，并选择下一个执行的活动。这些数据可以被工作流应用程序访问并修改。因此，工作流管理软件应该在活动实例之间传递工作流相关数据。

工作流应用数据（Workflow Application Data）：这种数据指那些由应用程序操作的数据。它们是针对应用程序的，工作流管理系统无法对它们进行访问。

2.工作流模型和工作流建模工具

工作流模型包含了工作流执行服务运行该过程的所有必需的信息，包括它启动和结束的条件、组成的活动、活动间导航的准则、参与其中的用户、需要激活的应用程序的指针、需要用到的工作流相关数据的定义等。

在工作流的建模期间需要参考组织/角色模型来获得有关组织结构和组织内角色的信息。过程定义指定完成某项活动的组织实体或角色，而不是定义具体人员。工作流执行服务负责在工作流运行环境内将组织实体或角色映射为特定的人员。

工作流建模工具主要用于分析、建模、描述并记录经营过程。它应输出一个能被工作流机动态解释的过程定义。不同的工作流产品其建模工具输出的格式是不同的，参考模型中的接口1不仅使工作流的定义阶段和运行阶段分离，使用户可以分别选择建模工具和执行产品，还可以使不同的工作流产品合作成为一个过程定义的执行提供运行服务环境。

工作流管理联盟针对工作流建模做了两方面的工作。

建立了一个元模型（process meta model）：它用于描述一个过程模型内各个对象、它们之间的关系及它们的属性，有利于多个工作流产品之间交换模型信息。

定义了一套可以在工作流管理系统之间及在管理系统与建模工具之间交互过程模型定义的API接口。

图12-6所示为工作流管理联盟定义的过程元模型。

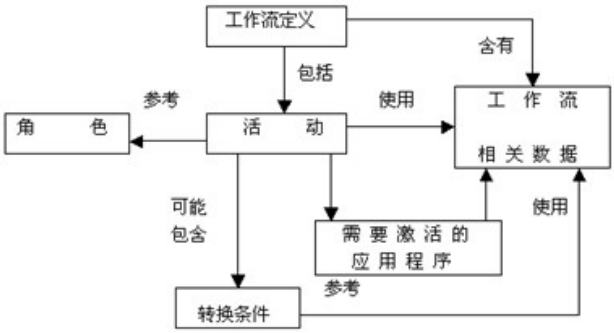

图12-6 过程元模型

3.工作流执行服务

工作流执行服务由一个或多个工作流机组成（在分布环境下，由多个工作流机组成），提供了过程实例执行的运行环境，主要完成以下功能。

解释流程定义，生成过程实例，并管理其实施过程。

依据过程定义和工作流相关数据为过程实例的导航提供进入和退出的条件、并行或串行执行活动的信息、用户信息或所需激活的应用程序的信息等。

与外部资源交互完成各项活动。

维护工作流控制数据和工作流相关数据（这些数据包括不同过程和活动实例的内部状态信息、工作流机用于协调和恢复的各种检查数据和恢复/重起信息等），并向用户传递必要的相关数据。

在分布式的工作流执行服务中，多个工作流机协调工作，推进工作流机实例的执行。每一个工作流机控制过程执行的一部分，并使用相关的资源和应用工具。这种执行服务需要共同的命名和管理范围，便于过程定义和用户/应用名称一致。分布式的工作流系统采用特定的协议来同步各工作流机，并传递相应的控制信息。在一个同构的工作流执行服务中这些协议是因厂家而异的。当选用不同的工作流系统产品时，各工作流机之间需要一个标准来进行转换。它应包括以下几个方面的内容。

一个共同的命名机制；

支持共同的过程定义对象和属性；

能够传递相应的工作流相关数据，并控制过程实例的生成；

能够在异构的工作流机间传递过程、子过程及活动；

支持共同的管理职能。

4.工作流机

工作流机是一个为工作流实例的执行提供运行环境的软件服务或"引擎",它主要提供以下功能。

对过程定义进行解释；

控制过程实例的生成、激活、挂起、终止等；

控制活动实例间的转换，包括串行或并行操作、工作流相关数据的解释等；

支持用户操作的界面；

维护工作流控制数据和工作流相关数据，在应用或用户间传递工作流相关数据；

提供用于激活外部应用程序和访问工作流相关数据的界面；

提供控制、管理和监督的功能。

工作流机的一个重要功能就是控制实例和活动实例的状态转换。工作流管理联盟的参考模型中为过程实例的运行状态和活动实例的状态进行了定义，并给出了状态转换的条件。图12-7和图12-8

分别描述了过程实例和活动实例各个状态之间的转换。

过程实例包括以下几种运行状态。

初始（Inactived）：一个过程实例已经生成，但该过程实例并没有满足开始执行的条件；

准备运行（Running）：该过程实例已经开始执行，但是还不满足开始执行第一个活动并生成一个任务项的条件；

运行中（Active）：一个或多个活动已经开始执行（也就是已经生成一个工作项并分配给了合适的活动实例）；

挂起（Suspended）：该过程实例正在运行，但处于静止状态，除非有一个"重启"的命令使该过程实例回到准备运行状态，否则所有的活动都不会执行；

结束（Completed）：该过程实例满足结束的条件，工作流管理系统将执行过程实例结束后的操作（如统计），并删除该过程实例；

终止（Terminated）：该过程实例在正常结束前被迫终止，工作流管理系统将执行补救措施，并删除该过程实例。

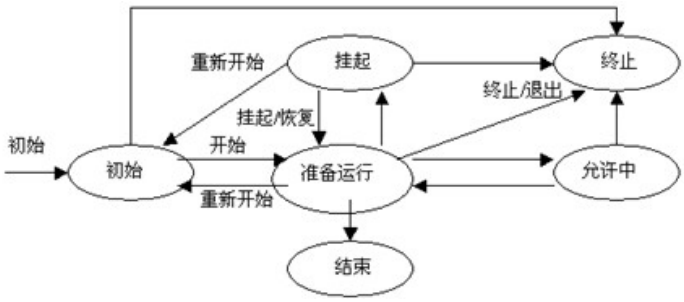

图12-7 过程实例状态转换图

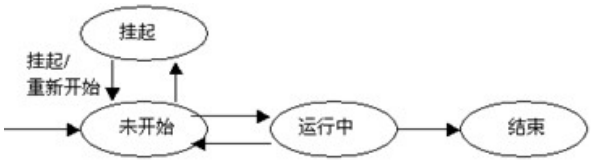

图12-8 活动实例状态转换图

活动的运行状态包括：

未开始（Inactive）：该活动实例已经生成但还没有被激活（例如活动开始条件没有满足）；

运行中（Active）：该活动实例已经被激活了；

挂起（Suspended）：该活动实例处于静止状态；

结束（Completed）：该活动已经执行完毕，工作流管理系统将进行活动结束后的导航工作，激活下一个符合启动条件的活动实例。

5.客户端应用

这种方式适合于需要人员参与的活动。在这种情况下，工作流机通过任务项列表管理器来进行控制。工作流管理联盟提供了四种可能的通过任务项列表来实现工作流客户与工作流机之间的通信方式，如图12-9所示。其中一种支持集中式的结构，另外三种适合分布情况下的工作流系统。

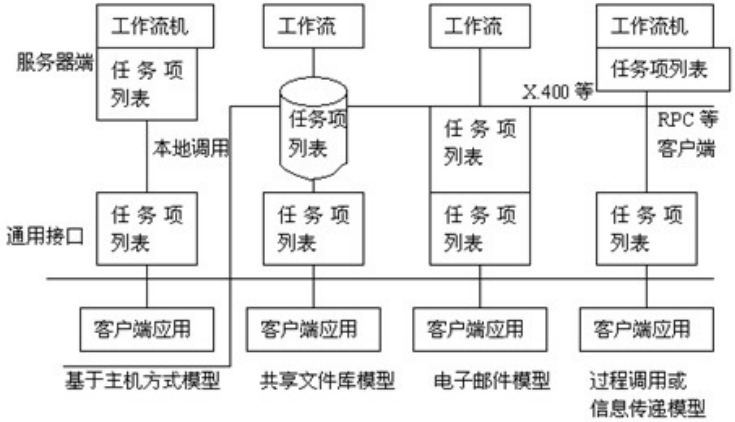

图12-9 四种任务项列表管理器的实现方法

基于主机方式的模型（Hust Based Model）：这种方式适合于集中的情况。此时，客户端应用程序、任务项列表管理器、任务项列表和工作流机都列在中央的主服务器上，用户通过模拟一个终端用户来获得任务项列表。

共享的文件库模型（Shared Filestore Model）：在这种情形下，客户应用程序和任务列表管理器位于用户的工作站上，而工作流位于中央服务器上。任务项列表位于一个客户应用和工作流机都能够达到的共享的文件系统中。

电子邮件模型（Electronic Mail Model）：客户应用和任务项列表管理器位于用户的工作站上，工作流机位于中央主机上。所有的通信都使用电子邮件。此时，任务项列表一般位于客户端。

过程调用或信息传递模型（Procedure Callor Message Passing Model）：客户应用程序和任务项列表管理器位于用户的工作站上，任务项列表和工作流机位于服务器端。用户通过RPC或者其他的消息传递机制来获得任务项列表。

6.由工作流机直接调用的应用程序

这种情况适合于不需要人员参与的活动。在简单的情况下，工作流机通过过程模型中定义的活动的信息、应用程序的类型和需要的数据来激活应用程序。被激活的应用程序可以和工作流机位于一台计算机上，可以位于相同的运行平台上，也可以位于网络可以到达的不同平台上。模型定义提供了有关应用程序的类型、地址等充分信息，便于工作流机激活该程序并执行相应的动作。

7.工作流执行服务之间的互操作性

工作流联盟的目标之一就是规定一个标准使得不同厂商提供的工作流产品能够协调工作，整个系统能够无缝地在各个产品之间传递任务项。工作流管理联盟在互操作性上的工作主要集中在提供了一系列互操作的情景，从简单的任务传递到传输整个工作流过程模型和工作流参考数据。尽管有可能考虑那些很复杂的情形（如不同厂商提供的工作流机共同协作实现工作流执行服务，这在目前还不可能实现，因为它要求所有的工作流机都能够解释过程模型，共享一套工作流控制数据，并在异构的工作流机环境下共享过程实例状态）。但就目前来说，比较切合实际的目标是在不同的工作流执行服务间传递过程的部分内容，支持其实例的运行。

8.系统管理和监控工具

该工具能够对工作流在整个组织内的流动状况进行监控，并提供一系列的管理功能，如有关安全性、对过程的控制和授权操作等方面的管理。主要功能包括以下几个方面。

建立、设置和优化组成工作流管理系统的各个软件；

对过程模型进行实例化；

将过程模型中的角色实例化；

将运行中的过程实例、活动实例和数据分发到各个工作流机中；

启动、挂起、恢复和终止过程实例；

管理正在执行的过程实例，并对正常或异常退出的过程的历史数据进行统计和分析。

9.工作流参考模型中的5类接口

工作流联盟给出了5类接口。

接口1:工作流服务和工作流建模工具；

接口2:工作流服务和客户应用之间的接口，这是最主要的接口规范，它约定所有客户方应用和工作流服务之间的功能访问方式；

接口3:工作流机和直接调用的应用程序之间的接口；

接口4:工作流管理系统之间的互操作接口；

接口5:工作流服务和工作流管理工具之间的接口。

其中，接口1为在不同物理或电子介质之间传递过程定义的信息提供了交互的形式和API调用；接口2定义了通信建立、工作流定义操作等功能；接口3激活应用程序的API函数应覆盖的几个方面的功能；接口4完成工作流执行服务之间需要提供大量的WAPI来实现互操作，可以是在两个工作流执行服务之间的直接调用或是通过网关函数；接口5主要实现对工作流的管理和监视。

极限编程

12.4 极限编程

敏捷方法论有一个共同的特点，那就是都将矛头指向了"文档",它们认为传统的软件工程方法文档量太"重"了，称为"重量级"方法，而相应的敏捷方法则是"轻量级"方法。正是因为"轻量级"感觉没有什么力量，不但不能够有效体现灵活性，反而显得是不解决问题的方法论似的。因此，就有了一次划时代的会议，创建了敏捷联盟。

在敏捷方法论领域中，最知名的、最有影响力的却不是它们，而是拥有与Microsoft最新的操作系统相同缩写语-XP的极限编程（eXtreme Programming）。极限编程方法论可以说是敏捷联盟中最鲜艳的一面旗帜，也是被研究、尝试、应用、赞扬、批判最多的一种方法论，也是相对来说最成熟的一种。

这一被誉为充满"黑客文化"的方法论的雏形最初形成于1996-1999年间，Kent Beck、Ward Cunningham、Ron Jeffery在开发C3项目（Chrysler Comprehensive Compensation）的实践中总结出了XP的基本元素。在此之后，Kent Beck和他的一些好朋友们一起在实践中完善提高，终于形成了极限编程方法论。

解析极限编程

12.4.1 解析极限编程

那么什么是XP呢？XP是一种轻量（敏捷）、高效、低风险、柔性、可预测、科学而且充满乐趣的软件开发方式。与其他方法论相比，其最大的不同在于：

在更短的周期内，更早地提供具体、持续的反馈信息。

迭代地进行计划编制，首先在最开始迅速生成一个总体计划，然后在整个项目开发过程中不断地发展它。

依赖于自动测试程序来监控开发进度，并及早地捕获缺陷。

依赖于口头交流、测试和源程序进行沟通。

倡导持续的演化式的设计。

依赖于开发团队内部的紧密协作。

尽可能达到程序员短期利益和项目长期利益的平衡。

Kent Beck曾经说过"开车"就是一个XP的范例，即使看上去进行得很顺利，也不要把视线从公路上移开，因为路况的变化，将使得你必须随时做出一些这样那样的调整。而在软件项目中，客户就是司机，他们也没有办法确切地知道软件应该做什么，因此程序员就需要向客户提供方向盘，并且告知我们现在的位置。

XP包括些什么呢？如图12-10所示，XP由价值观、原则、实践和行为四个部分组成，它们彼此相互依赖、关联，并通过行为贯穿于整个生命周期。

图12-10 XP组成示意图

四大价值观

12.4.2 四大价值观

XP的核心是其总结的沟通、简单、反馈、勇气四大价值观，它们是XP的基础，也是XP的灵魂。

1.沟通

通常程序员给人留下的印象就是"内向、不善言谈",然后项目中的许多问题就出在这些缺乏沟通的开发人员身上。经常由于某个程序员做出了一个设计决定，但是却不能够及时地通知大家，结果使得大家在协作与配合上出现了很多的麻烦。而在传统的方法论中，并不在意这种口头沟通不畅的问题，而是希望借助于完善的流程和面面俱到的文档、报表、计划来替代，但是这同时又引入了效率不高的新问题。

XP方法论认为，如果小组成员之间无法做到持续的、无间断的交流，那么协作就无从谈起，从这个角度能够发现，通过文档、报表等人工制品进行交流面临巨大的局限性。因此，XP组合了诸如对编程这样的最佳实践，鼓励大家进行口头交流，通过交流解决问题，提高效率。

2.简单

XP方法论提倡在工作中秉承"够用即好"的思路，也就是尽量地简单化，只要今天够用就行，不考虑明天会发现的新问题。这一点看上去十分容易，但是要真正做到保持简单的工作其实是很难的。因为在传统的开发方法中，都要求大家对未来做一些预先规划，以便对今后可能发生的变化预留一些扩展的空间。

正如对传统开发方法的认识一样，许多开发人员也会置疑XP,保持系统的扩展性很重要，如果都保持简单，那么如何使得系统能够有良好的扩展性呢？其实不然，保持简单的理由有两个：

开发小组在开发时所做的规划，并无法保证其是符合客户需要的，因此做的大部分工作都将落空，使得开发过程中重复的、没有必要的工作增加，导致整体效率降低。

另外，在XP中提倡时刻对代码进行重构，一直保持其良好的结构与可扩展性。也就是说，可扩展性和为明天设计并不是同一个概念，XP是反对为明天考虑而工作，并不是说代码要失去可扩展性。

而且简单和沟通之间还有一种相当微妙的互相支持关系。当一个团队之间，沟通得越多，那么就越容易明白哪些工作需要做，哪些工作不需要做。另一方面，系统越简单，需要沟通的内容也就越少，沟通也将更加全面。

3.反馈

是什么原因使得我们的客户、管理层这么不理解开发团队？为什么客户、管理层总是喜欢给我们一个死亡之旅？究其症结，就是开发的过程中缺乏必要的反馈。在许许多多项目中，当开发团队经历过了需求分析阶段之后，在相当长的一段时间内，是没有任何反馈信息的。整个开发过程对于客户和管理层而言就像一个黑盒子，进度完全不可见。

而且在项目过程中，这样的现象不仅出现在开发团队与客户、管理层之间，还包括在开发团队内部。这一切问题都需要我们更加注重反馈。反馈对于任何软件项目的成功都是至关重要的，而在XP方法论中则更进一步，通过持续、明确的反馈来暴露软件状态的问题。具体而言就是：

在开发团队内部，通过提前编写单元测试代码，时时反馈代码的问题与进展。

在开发过程中，还应该加强集成工作，做到持续集成，使得每一次增量都是一个可执行的工作版本，也就是逐渐使软件长大。整个过程中，应该通过向客户和管理层演示这些可运行的版本，以便及早地反馈，及早地发现问题。

同时，我们也会发现反馈与沟通也有着良好的配合，及时和良好的反馈有助于沟通。而简单的系统，更利于测试和反馈。

4.勇气

在应用XP方法论时，我们每时每刻都在应对变化：由于沟通良好，因此会有更多需求变更的机会；由于时刻保持系统的简单，因此新的变化会带来一些重新开发的需要；由于反馈及时，因此会有更多中间打断你的思路的新需求。

总之这一切，使得你立刻处于变化之中，因此这时就需要你有勇气来面对快速开发，面对可能的重新开发。也许你会觉得，为什么要让我们的开发变得如此零乱，但是其实这些变化若你不让它早暴露，那么它就会迟一些出现，并不会因此消亡，因此，XP方法论让它们早出现、早解决，是实现"小步快走"开发节奏的好办法。

也就是XP方法论要求开发人员穿上强大、自动测试的盔甲，勇往直前，在重构、编码规范的支持下，有目的地快速开发。

勇气可以来源于沟通，因为它使得高风险、高回报的试验成为可能；勇气可以来源于简单，因为面对简单的系统，更容易鼓起勇气；勇气可以来源于反馈，因为你可以及时获得每一步前进的状态（自动测试），会使得你更勇于重构代码。

5.四大价值观之外

在这四大价值观之下，隐藏着一个更深刻的东西，那就是尊重。因为这一切都建立在团队成员之间的相互关心、相互理解的基础之上。

5个原则

12.4.3　5个原则

1.快速反馈

及时地、快速地获取反馈，并将所学到的知识尽快地投入到系统中去。也就是指开发人员应该通过较短的反馈循环迅速地了解现在的产品是否满足了客户的需求。这也是对反馈这一价值观的进一步补充。

2.简单性假设

类似地，简单性假设原则是对简单这一价值观的进一步补充。这一原则要求开发人员将每个问题都看得十分容易解决，也就是说只为本次迭代考虑，不去想未来可能需要什么，相信具有将来必要时增加系统复杂性的能力，也就是号召大家出色地完成今天的任务。

3.逐步修改

就像开车打方向盘一样，不要一次做出很大的改变，那样将会使得可控性变差，更适合的方法是进行微调。而在软件开发中，这样的道理同样适用，任何问题都应该通过一系列能够带来差异的微小改动来解决。

4.提倡更改

在软件开发过程中，最好的办法是在解决最重要的问题时，保留最多选项的那个。也就是说，尽量为下一次修改做好准备。

5.优质工作

在实践中，经常看到许多开发人员喜欢将一些细小的问题留待后面来解决。例如，界面的按钮有一些不平整，由于不影响使用就先不管；某一两个成员函数暂时没用就先不写等。这就是一种工作拖泥带水的现象，这样的坏习惯一旦养成，必然使得代码质量大打折扣。

而在XP方法论中，贯彻的是"小步快走"的开发原则，因此工作质量决不可打折扣，通常采用测试先行的编码方式来提供支持。

12个最佳实践

12.4.4 12个最佳实践

在XP中，集成了12个最佳实践，有趣的是，它们没有一个是创新的概念，大多数概念和编程一样老。其主要的创新点在于提供一种良好的思路，将这些最佳实践结合在一起，并且确保尽可能彻底地执行它们，使得它们能够在最大程度上互相支持。紧接下来，我们就对每一种最佳实践进行一番了解。

1.计划游戏

计划游戏的主要思想就是先快速地制定一份概要的计划，然后随着项目细节的不断清晰，再逐步完善这份计划。计划游戏产生的结果是一套用户故事及后续的一两次迭代的概要计划。

"客户负责业务决策，开发团队负责技术决策"是计划游戏获得成功的前提条件。也就是说，系统的范围、下一次迭代的发布时间、用户故事的优先级应该由客户决定；而每个用户故事所需的开发时间、不同技术的成本、如何组建团队、每个用户故事的风险，以及具体的开发顺序应该由开发团队决定。

好了，明白这些就可以进行计划游戏了。首先客户和开发人员坐在同一间屋子里，每个人都准备一支笔、一些用于记录用户故事的纸片，最好再准备一个白板，就可以开始了。

客户编写故事：由客户谈论系统应该完成什么功能，然后用通俗的自然语言，使用自己的语汇，将其写在卡片上，这也就是用户故事。

开发人员进行估算：首先客户按优先级将用户故事分成必须要有、希望有、如果有更好三类，然后开发人员对每个用户故事进行估算，先从高优先级开始估算。如果在估算的时候，感到有一些故事太大，不容易进行估算，或者是估算的结果超过2人/周，那么就应该对其进行分解，拆成2个或多个小故事。

确定迭代的周期：接下来就是确定本次迭代的时间周期，这可以根据实际的情况进行确定，不过最佳的迭代周期是2~3周。有了迭代的时间之后，再结合参与的开发人数，算出可以完成的工作量

总数。然后根据估算的结果，与客户协商，挑出时间上够、优先级合适的用户故事组合，形成计划。

2.小型发布

XP方法论秉承的是"持续集成、小步快走"的哲学，也就是说每一次发布的版本应该尽可能地小，当然前提条件是每个版本有足够的商业价值，值得发布。

由于小型发布可以使得集成更频繁，客户获得的中间结果也越频繁，反馈也就越频繁，客户就能够实时地了解项目的进展情况，从而提出更多的意见，以便在下一次迭代中计划进去，以实现更高的客户满意度。

3.隐喻

相对而言，隐喻这一最佳实践是最令人费解的。什么是隐喻呢？根据词典中的解释是："一种语言的表达手段，它用来暗示字面意义不相似的事物之间的相似之处".那么这在软件开发中又有什么用呢？总结而言，常常用于四个方面。

寻求共识：也就是鼓励开发人员在寻求问题共识时，可以借用一些沟通双方都比较熟悉的事物来做类比，从而帮助大家更好地理解解决方案的关键结构，也就是更好地理解系统是什么、能做什么。

发明共享词汇：通过隐喻，有助于提出一个用来表示对象、对象间的关系的通用名称。例如，策略模式（用来表示可以实现多种不同策略的设计模式）、工厂模式（用来表示可以按需"生产"出所需类的设计模式）等。

创新的武器：有的时候，可以借助其他东西来找到解决问题的新途径。例如："我们可以将工作流看做一个生产线".

描述体系结构：体系结构是比较抽象的，引入隐喻能够大大减轻理解的复杂度。例如：管道体系结构就是指两个构件之间通过一条传递消息的"管道"进行通信。

当然，如果能够找到合适的隐喻是十分快乐的，但并不是每一种情况都可以找到恰当的隐喻，你也没有必要强求。

4.简单设计

强调简单的价值观，引出了简单性假设原则，落到实处就是"简单设计"实践。这个实践看上去似乎很容易理解，但却又经常被误解，许多批评者就指责XP忽略设计是不正确的。其实，XP的简单设计实践并不是要忽略设计，而且认为设计不应该在编码之前一次性完成，因为那样只能建立在"情况不会发生变化"或者"我们可以预见所有的变化"之类的谎言的基础上的。

Kent Beck概念中的简单设计是这样的：

能够通过所有的测试程序。

没有包括任何重复的代码。

清楚地表现出了程序员赋予的所有意图。

包括尽可能少的类和方法。

他认为要想保持设计简单的系统，需要具备简单思考的能力，拥有理解代码和修改代码的勇气，以及为了消除代码的"坏味道"而定期重构的习惯。

那么如何开始进行简单的设计呢？XP实践者们也总结出了一些具体的、可操作的思考方法。

首先写测试代码：具体将在后面详细描述。

保持每个类只负责一件事：SRP（单一职责原则）是面向对象设计的基础原则之一。

使用Demeter（迪米特）法则：迪米特法则，也称为LoD法则、最少知识原则。也就是指一个对象应当对其他对象尽可能少地了解。用隐喻的方法来解释的话就是"只与你直接的朋友们通信"、"不要和陌生人说话".

使用CRC卡片进行探索。

5.测试先行

当我第一次看到"测试先行"这个概念的时候，我的第一感觉就是不解，陷入了"程序都还没有写出来，测试什么呀？"的迷思。我开始天马行空地寻求相关的隐喻，终于找到了能够启发我的工匠。首先，我们来看看两个不同工匠是如何工作的吧。

工匠一：先拉上一根水平线，砌每一块砖时，都与这根水平线进行比较，使得每一块砖都保持水平。

工匠二：先将一排砖都砌完，然后再拉上一根水平线，看看哪些砖有问题，对有问题的砖进行适当的调整。

你会选择哪种工作方法呢？你一定会骂工匠二笨吧！这样多浪费时间呀！然而你自己想想，你平时在编写程序的时候又是怎么做的呢？我们就是按工匠二的方法在工作呀！甚至有时候比工匠二还笨，是整面墙都砌完了，直接进行"集成测试",经常让整面的墙倒塌。看到这里，你还觉得自己的方法高明吗？这个连工匠都明白的道理，自己却画地为牢呀。

不仅我们没有采用工匠一的工作方法，甚至有的时候程序员会以"开发工作太紧张"为理由，而忽略测试工作。但这样却导致了一个恶性循环，越是没空编写测试程序，代码的效率与质量越差，花在找Bug、解决Bug的时间也越来越多，实际产能大大降低。由于产能降低了，因此时间更紧张，压力更大。你想想，为什么不拉上一根水平线呢？难道，我们不能够将后面浪费的时间花在单元测试上，使得我们的程序一开始就更加健壮，更加易于修改吗？不过，编写测试程序当然要比拉一条水平线难得多，所以我们需要引入"自动化测试工具",免费的xUnit测试框架就是你最佳的选择。

为了鼓励程序员愿意甚至喜欢在编写程序之前编写测试代码，XP方法论还提供了许多有说服力的理由。

如果你已经保持了简单的设计，那么编写测试代码根本不难。

如果你在结对编程，那么如果你想出一个好的测试代码，那么你的伙伴一定行。

当所有的测试都通过的时候，你再也不会担心所写的代码今后会"暗箭伤人",那种感觉是相当棒的。

当你的客户看到所有的测试都通过的时候，会对程序充满前所未有的信心。

当你需要进行重构时，测试代码会给你带来更大的勇气，因为你要测试是否重构成功只需要一个按钮。

测试先行是XP方法论中一个十分重要的最佳实践，并且其中所蕴含的知识与方法也十分丰富。

6.重构

重构是一种对代码进行改进而不影响功能实现的技术，XP需要开发人员在闻到代码的坏味道时，有重构代码的勇气。重构的目的是降低变化引发的风险，使得代码优化更加容易。通常重构发生在两种情况之下。

实现某个特性之前：尝试改变现有的代码结构，以使得实现新的特性更加简单。

实现某个特性之后：检查刚刚写完的代码后，认真检查一下，看是否能够进行简化。

在《重构》一书中，作者Martin Fowler提示我们：在考虑重构时，应该要养成编写并经常运行测试代码的习惯；要先编写代码，再进行重构；把每一次增加功能都当做一次重构的好时机；将每一个纠正错误当做一次重构的重要时机。同时，该书中也列出大量需要重构的情况和重构的方法。

最后类似地，给还没有足够勇气进行重构的读者打上几剂强心针：

XP提倡集体代码所有制，因此你可以大胆地在任何需要修改的地方做改动。

由于在XP项目组中有完整的编码标准，因此在重构前无须重新定义格式。

在重构中遇到困难，和你结对编程的伙伴能够为你提供有效的帮助。

简单的设计，会给重构带来很大的帮助。

测试先行让你拥有了一个有效的检验器，随时运行一下就知道你重构的工作是否带来了影响。

由于XP在持续集成，因此你重构所带来的破坏很快就能够暴露，并且得以解决。

重构技术是对简单性设计的一个良好的补充，也是XP中重视"优质工作"的体现，这也是优秀的程序员必备的一项技能。

7.结对编程

"什么！两个人坐在一起写程序？那岂不是对人力的巨大浪费吗？而且我在工作时可不喜欢有一个人坐在边上当检察官。"是的，正如这里列举出来的问题一样，结对编程技术还是被很多人置疑的。

不过，自从20世纪60年代，就有类似的实践在进行，长期以来的研究结果却给出了另外一番景象，那就是结对编程的效率反而比单独编程更高。一开始虽然会牺牲一些速度，但慢慢地，开发速度会逐渐加快。究其原因，主要是结对编程大大降低了沟通的成本，提高了工作的质量，具体表现在：

所有的设计决策确保不是由一个人做出的。

系统的任何一个部分都肯定至少有2个人以上熟悉。

几乎不可能有2个人都忽略的测试项或者其他任务。

结对组合的动态性，是一个企业知识管理的好途径。

代码总是能够保障被评审过。

而且XP方法论集成的其他最佳实践也能够使得结对编程更加容易进行：

编码标准可以消除一些无谓的分歧。

隐喻可以帮助结对伙伴更好地沟通。

简单设计可以使得结对伙伴更了解他们所从事的工作。

结对编程技术被誉为XP保持工作质量、强调人文主义的一个最典型的实践，应用得当还能够使得开发团队之前的协作更加顺畅、知识交流与共享更加频繁，团队的稳定性也会更加的稳固。

8.集体代码所有制

由于XP方法论鼓励团队进行结对编程，而且认为结对编程的组合应该动态地搭配，根据任务的不同、专业技能的不同进行最优组合。由于每一个人都肯定会遇到不同的代码，所以代码的所有制就不再适合于私有，因为那样会给修改工作带来巨大的不便。

也就是说，团队中的每个成员都拥有对代码进行改进的权利，每个人都拥有全部代码，也都需要对全部代码负责。同时，XP强调代码是谁破坏的（也就是修改后发生问题），就应该由谁来修

复。

由于在XP中，有一些与之相匹配的最佳实践进行配合，因此你并无须担心采用集体代码所有制会让你的代码变得越来越乱：

由于在XP项目中，集成工作是一件经常性的工作，因此当有人修改代码而带来了集成的问题，会在很快的时间内被发现。

由于每一个类都会有一个测试代码，因此不论是谁修改了代码，都需要运行这个测试代码，这样偶然性的破坏发生的概率将很小。

由于每一个代码的修改就是通过了结对的两个程序员共同的思考，因此通常做出的修改都是对系统有益的。

由于大家都坚持了相同的编码标准，因此代码的可读性、可修改性都会比较好，而且还能够避免由于命名法、缩进等小问题引发经常性的代码修改。

集体代码所有制是XP与其他敏捷方法的一个较大不同，也是从另一个侧面体现了XP中蕴含的很深厚的编码情节。

9.持续集成

在前面谈到小型发布、重构、结对编程、集体代码所有制等最佳实践的时候，我们多次看到"持续集成"的身影，可以说持续集成是对这些最佳实践的基本支撑条件。

可能大家会对持续集成与小型发布代表的意思混淆不清，其实小型发布是指在开发周期中经常发布中间版本，而持续集成的含义则是要求XP团队每天尽可能多次地做代码集成，每次都在确保系统运行的单元测试通过之后进行。

这样，就可以及早地暴露、消除由于重构、集体代码所有制所引入的错误，从而减少解决问题的痛苦。

要在开发过程中做到持续集成并不容易，首先需要养成这个习惯。而且集成工作往往是十分枯燥、烦琐的，因此适当地引入每日集成工具是十分必要的。XP建议大家首先使用配置管理服务器将代码管理起来，然后使用Ant或Nant等XP工具，编写集成脚本，调用xUnit等测试框架，这样就可以实现每当程序员将代码Check in到配置服务器上时，Ant就会自动完成编译和集成，并调用测试代码完成相应的测试工作。

10.每周工作40小时

这是最让开发人员开心、管理者反对的一个最佳实践了，加班、再加班早已成为开发人员的家常便饭，也是管理者最常使用的一种策略。而XP方法论认为，加班最终会扼杀团队的积极性，最终导致项目的失败，这也充分体现了XP方法关注人的因素比关注过程的因素更多一些。

Kent Beck认为开发人员即使能够工作更长时间，他们也不应该这样做，因为这样做会使他们更加容易厌倦编程工作，从而产生一些影响他们效能的其他问题。因此，每周工作40小时是一种顺势行为，是一种规律。其实对于开发人员和管理者来说，违反这种规律是不值得的。

开发人员：如果不懂得休息，那么就无法将自己的节奏调整到最佳状态，那么就会带来很大的负面影响。而且在精神不集中的状态下，开发的质量也得不到保证。

管理者：也许这可以称得上"第二种人月神话",那就是你不能够通过延长每天的工作时间来获得更多的人月。这是因为，每个开发人员的工作精力是有限的，不可能无限增长，在精力不足的时候，不仅写出来的代码质量没有保障，而且还可能为项目带来退步的效果。因此，采用加班的方式

并不是一个理性的方式，是得不偿失的。

不过有一点是需要解释的，"每周工作40小时"中的40不是一个绝对数，它所代表的意思是团队应该保证按照"正常的时间"进行工作。那么如何做到这一点呢？

首先，定义符合你团队情况的"正常工作时间".

其次，逐步将工作时间调整到"正常工作时间".

再次，除非你的时间计划一团糟，否则不应该在时间上妥协。

最后，鼓起勇气，制定一个合情合理的时间表。

正如米卢说过的"享受足球"一样，同样地，每一个开发人员应该做到"享受编程",那么"每周工作40小时"就是你的起点。

11.现场客户

为了保证开发出来的结果与客户的预想接近，XP方法论认为最重要的是需要将客户请到开发现场。就像计划游戏中提到过的，在XP项目中，应该时刻保证客户负责业务决策，开发团队负责技术决策。因此，在项目中有客户在现场明确用户故事，并做出相应的业务决策，对于XP项目而言有着十分重要的意义。

也许有人会问，客户提交了用户故事之后不就完成工作了吗？其实很多尝试过用户故事的团队都会发现其太过简单，包含的信息量极少，XP方法论不会不了解，因此，不会把用户故事当做开发人员交付代码的唯一指示。用户故事只是一个起点，后面的细节还需要开发人员与客户之间建立起来的良好沟通来补充。

作为一名有经验的开发人员，绝对不会对现场客户的价值产生任何怀疑，但是都会觉得要想实现现场客户十分困难。要实现这一点，需要对客户进行沟通，让其明白，相对于开发团队，项目成功对于客户而言更为重要。而现场客户则是保障项目成功的一个重要措施，想想在你装修房子的时候，你是不是常常在充当现场客户的角色呢？其实这个隐喻就是让客户理解现场客户重要性最好的突破口。

其实现场客户在具体的实施时，也不是一定需要客户一直和开发团队在一起，而是开发团队应该和客户能够随时的沟通，可以是面谈，可以是在线聊天，可以是电话，当然面谈是必不可少。其中的关键是当开发人员需要客户做出业务决策时，需要进一步了解业务细节时能够随时找到相应的客户。

不过，也有一些项目是可以不要现场客户参与的：

当开发组织中已经有相关的领域专家时。

当做一些探索性工作，而且客户也不知道他想要什么时（例如新产品、新解决方案的研究与开发）。

去尝试吧，现场客户不仅可以争取得到，而且还能够使得团队焕然一新，与客户建立起良好的合作与信任。

12.编码标准

编码标准是一个"雅俗共享"的最佳实践，不管是代表重型方法论的RUP、PSP,还是代表敏捷方法论的XP,都认为开发团队应该拥有一个编码标准。XP方法论认为拥有编码标准可以避免团队在一些与开发进度无关的细节问题上发生争论，而且会给重构、结对编程带来很大的麻烦。试想如果有人将你上次写的代码的变量命名法做了修改，下次你需要再改这部分代码时，会是一种什么感觉呢？

不过，XP方法论的编码标准的目的不是创建一个事无巨细的规则列表，而是只要能够提供一个确保代码清晰，便于交流的指导方针。

如果你的团队已经拥有编码标准，就可以直接使用它，并在过程中进行完善。如果还没有，那么大家可以先进行编码，然后在过程中逐步总结出编码规则，边做边形成。当然除了这种文字规范以外，还可以采用一些如自动格式化代码工具之类的方法进行代码规范。事实上，你只需要很好地贯彻执行其他实践并且进行沟通，编码标准会很容易地浮现出来。

13.融合是关键

有句经典名言"1+1>2"最适合表达XP的观点，Kent Beck认为XP方法论的最大价值在于在项目中融会贯通地运用这12个最佳实践，而非单独地使用。你当然可以使用其中的一些实践，但这并不意味着你就应用了XP方法论。XP方法论真正能够发挥其效能，就必须完整地运用12个实践。

上一节　本书简介　下一节

Web Service

12.5 Web Service

本节主要介绍Web Service.

上一节　本书简介　下一节

Web Service简介

12.5.1 Web Service简介

Web Service是一套标准，它定义了应用程序如何在Web上实现互操作性。支持用不同的语言（如VB、Java等）在不同的平台上（如Windows、UNIX、Linux等）编写Web Service ,而后通过Web Service的标准对外发布服务，其他用户或应用也通过Web Service的标准来对这些服务进行查询和后续的访问调用。

Web Service由SOAP（简单对象访问协议）、WSDL（服务描述语言）、UDDI（服务注册检索访问标准）3个协议有力的支持和实现，下面分别给予简单的介绍。

SOAP:简单对象访问协议SOAP（Simple Object Access Protocol）提供了标准的RPC方法来调用Web Service协议，定义了服务请求者和服务提供者之间的消息传输规范。SOAP用XML来格式化消息，用HTTP来承载消息，其有很大的可扩展性和平台语言无关性，在各种平台上很容易实现。

WSDL:服务描述语言WSDL（Web Service Description Language）为服务提供者提供了用

XML格式描述Web Services 的标准格式，以表达一个Web Service 能提供什么功能，它的位置在哪里，如何调用它等。

UDDI:服务注册检索访问标准UDDI（Universal Discovery, Description, Integration）提供了一种机制让Web服务提供商发布他们的产品，并最终让他们的客户能定位他们所提供的Web服务。其核心组件是UDDI商业注册，它利用WSDL语言来描述企业及其提供的Web服务。

Web Service的实现

12.5.2 Web Service的实现

Web Service通过服务的建立、描述、发布、查找、调用几步来实现不同平台间服务的分布调用，具体描述如下。

Web服务的建立（Build）：可用不同的语言在不同的平台上开发Web服务。

Web服务的描述（Description）：Web服务开发出来后，用WSDL的标准来服务请求和响应的参数格式及其他协议相关的描述。

Web服务的发布（Publish）：为了使服务可访问，服务提供者需要首先将服务进行一定描述并发布到注册服务器上。

Web服务的查找（Find）：服务请求方根据注册服务器提供的规范接口发出查询请求，以获取绑定服务所需的相关信息。

Web服务的调用（Bind）：服务请求方通过分析从注册服务器中得到的服务绑定信息，包括服务的访问路径、服务调用的参数、返回结果、传输协议、安全要求等，对自己的系统进行相应配置，进而远程调用服务提供者所提供的服务。

可用图12-11所示来描述Web Service的实现过程。

总之，Web Service 体系结构基于3种角色之间的交互操作，一起作用于Web Service 构件：服务提供者使用WSDL（服务描述语言）来定义抽象的服务描述，然后把具体的服务发布到UDDI注册中心；服务请求者使用查找（Find）操作从UDDI注册中心检索服务描述，然后使用服务描述与服务提供者进行绑定（Bind），并调用Web Service实现访问。

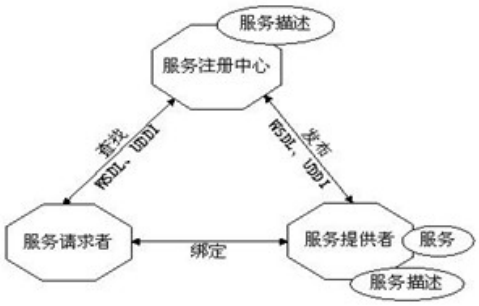

图12-11 Web Service的实现过程

　作者：希赛教育软考学院　来源：希赛网　2014年01月27日

Web Service的应用

12.5.3 Web Service的应用

Web Service的主要目标是跨平台，创建可互操作的分布式应用程序。为了达到这一目标，Web Service完全基于XML、XSD等独立于平台、独立于软件供应商的标准，主要适用于企业内部不同应用的集成、B2B集成、代码和数据重用，以及通过Web进行客户端和服务器的通信的场合，下面分别一一详细介绍。

企业内部不同应用的集成和数据交互：企业内部的不同语言写成的在不同平台上的各种系统经常要进行相互访问或数据交换，目前往往都采用临时编制的专用接口，工作复杂而且效率低下。通过Web Service,应用程序可以用标准的方法把功能和数据暴露出来，供其他的应用程序使用，克服诸多不足。

利用浏览器访问服务端的应用：目前大型企业信息系统的客户端都分布在世界各地，传统的C/S模式存在着更新复杂及客户端和服务器之间的通信和安全强度弱等诸多问题。于是，现在一般都采用浏览器作为客户端，此时使用Web Service结构，就可以轻松方便地解决以上问题。

B2B的集成：用Web Service集成应用程序，可以使公司内部的商务处理更加自动化。但当交易跨越了供应商和客户，突破了公司的界线时又会怎么样呢？跨公司的商务交易集成通常叫做B2B集成。 Web Service是B2B集成成功的关键。通过Web service,公司可以把关键的商务应用暴露给指定的供应商和客户。例如，把你的电子下单系统和电子发票系统暴露出来，你的客户就可以以电子的方式向你发送购货订单，而你的供应商则可以以电子的方式把原料采购的发票发送给你。用Web Service来实现B2B集成的最大好处在于可以轻易实现互操作性。只要把你的商务逻辑暴露出来，成为Web Service,你就可以让任何指定的合作伙伴轻松地调用你的商务逻辑，而不管他们的系统在什么平台上运行，使用的是什么开发语言。这样就大大减少了花在B2B集成上的时间和成本。

软件重用：Web Service允许你在重用代码的同时，重用代码后面的数据。使用Web Service,你不再像以前那样，要先从第三方购买、安装软件组件，再从你的应用程序中调用这些组件。你只需要直接调用远端的Web Service就可以了。举个例子，你想在你的应用程序中确认用户输入的邮件地址，那么，你只需把这个地址直接发送给相应的Web Service,这个Web Service 就会帮你查阅街道地址、城市、省区和邮政编码等信息，确认这个地址的确在相应的邮政编码区域。Web Service 的提供商可以按时间或使用次数对这项服务进行收费。这样的服务要通过组件重用来实现是不现实的，因为那样的话，你必须下载并安装好包含街道地址、城市、省区和邮政编码等信息的数据库，而且这个数据库还是不能实时更新的。

另一种软件重用的情况是，把好几个应用程序的功能集成起来。例如，你想要建立一个局域网上的门户站点应用，让用户既可以查看股市行情，又可以管理他们的日程安排，还可以在线购买电影票。现在Web上有很多应用程序供应商，都在其应用中实现了上面的这些功能。一旦他们把这些

功能都通过Web Service 暴露出来，你就可以非常轻易地把所有这些功能都集成到你的门户站点中，为用户提供一个统一的、友好的界面。

上一节　本书简介　下一节

例题分析

12.6 例题分析

例题1（2011年5月试题30）

关于过程改进，以下叙述中不正确的是 （30） .

（30）A.软件质量依赖于软件开发过程的质量，其中个人因素占主导作用

B.要使过程改进有效，需要制定过程改进目标

C.要使过程改进有效，需要进行培训

D.CMMI成熟度模型是一种过程改进模型，仅支持阶段性过程改进而不支持连续性过程改进

例题分析：

软件过程改进的实施对象是软件企业的软件过程，也就是软件产品的生产过程，其中还也包括软件维护之类的维护过程。

在本题各选项的描述中，A、B、C都是正确的，D不正确。

CMMI是Capability Maturity Model Integration的简称，即能力成熟度模型集成，它是在CMM的基础上发展起来的。CMMI是一种过程改进模型，它不仅支持阶段性过程改进，而且还支持连续性过程改进。

例题答案：（30）D

上一节　本书简介　下一节

综述

第13章 计算机专业英语

本章主要讲解计算机专业英语。

13.1 综述

英语能力是软件设计师的必备能力，因此，计算机英语是软件设计师考试的重要内容。考试大纲要求"具有工程师所要求的英语阅读水平，理解本领域的英语术语".在软件设计师上午试题中，共75分，其中英语占5分（2007年以前英语占10分）。

1.软件设计师英语考试与其他英语考试的比较

近几年的软件设计师英语考试主要有如下几方面的特点。

难度略高于大学英语四级，相当于研究生入学考试。

题材限于计算机文化读物，不如其他英语考试广泛。

题型只限于短文填空（完型填空），题型单一。

2.复习与应试要点

根据考试试题的特点，软件设计师英语复习要点如下。

找一本研究生入学考试（或四级）英语复习资料，复习相关的固定搭配、短语、语法知识，重点复习其中的完型填空，掌握完型填空的考点及要求。

注意多读计算机报刊、杂志的时文，在了解这个领域最新信息的同时积累语言知识，训练阅读能力。在复习时看一些计算机英语材料，对这一领域的表达方式和词汇进行热身。本章我们精选了一些英语材料，供大家复习参考。

用近几年的软件设计师英语考试试题进行模拟测试，本章收集了近几年的试题。

由于软件设计师英语考试题型只限于短文填空（完型填空），因此，自己可以在考前作一些专项练习，结合复习总结出一些解题的技巧。一般可采用三步法，其要点如下。

粗略地看一遍全文，了解全文的信息。

以了解的信息作为基础，对全文进行精读，并进行完型填空。

从全局的角度，对答卷进行检查。

第 13 章：计算机专业英语　作者：希赛教育软考学院　来源：希赛网　2014年01月27日

试卷分析

13.2　试卷分析

从1991年起，软件设计师计算机专业英语部分的考查方式为完型填空，延续至今。

从广度上看，考查的题材比较新颖，内容广泛，涵盖了计算机的几个主要领域。

涉及到网络的：2006年11月考网络访问控制，2006年5月考Cookies,2005年5月考DOM,2003年考网络协议，2003年考浏览器，2001年考数据包裹，2000年考VOIP（IP话音业务）解决方案；

涉及面向对象知识的：2008年12月考UML,2008年5月考面向对象分析，2007年11月考RUP;

涉及到程序语言及其基础理论的：2002年考强制型语言，1998年考字处理拼写检查，1996年考Java;

涉及到信息安全的：2005年11月考数字认证，2005年5月考邮件病毒，2002年考计算机系统攻击，2000年考防火墙；

涉及到计算机系统结构的：2001年考多指令多数据流系统，1992年考精简指令集对复杂指令集的争议；

涉及到数据库的：1997年考关系型数据库模型，1994年考面向对象的数据库管理系统；

涉及到操作系统的：1995年考微内核技术，1994年考Windows NT操作系统；

涉及到新的应用技术和领域的：2006年11月考虚拟化，1995年考可视化的开发工具，1993年考移动式计算机，1992年考人工智能硬件；

涉及到计算机文化和产业的：1999年考信息产业，1991年考计算机文化教育产业。

但从深度上看，对计算机知识考得都比较浅显，多是一些IT方面的时文摘要。考查的力度仅仅体现在"了解，知道"这个层次上，没有对考生做更深入的要求。

从考查的内容来看，不仅仅考查计算机知识。还考一些英语基础，如：查上下文和主题、词形的辨析、词义的辨析、词性的辨析、考时态、考语态和分词、介词及其固定搭配等。

因此在准备这部分考试内容的时候，不应该仅仅局限于计算机领域词汇的记忆。考试大纲要求"具有工程师所要求的英语阅读水平，理解本领域的英语术语".这要求考生在语言的基本功上应该有所积累，如大学英语四六级，研究生英语考试考纲要求的词汇，相关的固定搭配、短语，相关的语法知识，这些都是"正确阅读和理解文章"的本钱。除此之外，还应该注意多读计算机报刊、杂志的时文，在了解这个领域最新信息的同时积累语言知识，训练阅读能力。最好还能辅以一些英语短文完型填空的练习，自己总结出一些解题的技巧。从这几年来的试卷和考试大纲来看，主要考查两个方面。

考查的计算机专业词汇及相关知识，这方面，试卷既体现了"日新月异"的特点，又选择了相对稳定和主流的内容，没有刻意地求"新"和求"偏".笔者根据命题的趋势，从近年来较新的计算机文献中挑选了一些专业词汇和缩略语作为附录。既注意了"新",又兼顾了"全",剔除了过分陈旧和过分偏颇的词汇。在今后几年可能会考查到，或者作为文章的题材涉及到。亦可供平时查阅之用，免去考生自己收集整理之繁琐。希望对大家有所帮助。

考查"正确阅读和理解计算机领域的英文文献",本章中编者精选了一些最新英语素材，供考生复习时参考。