软 件 工 程 技 术 丛 书

HZ BOOKS

统一过程最佳实践
构造阶段

软件开发过程系列

（加） Scott W. Ambler
（澳） Larry L. Constantine 著
兰雨晴 高静 等译

The Unified Process
Construction Phase
Best Practices
in Implementing the UP

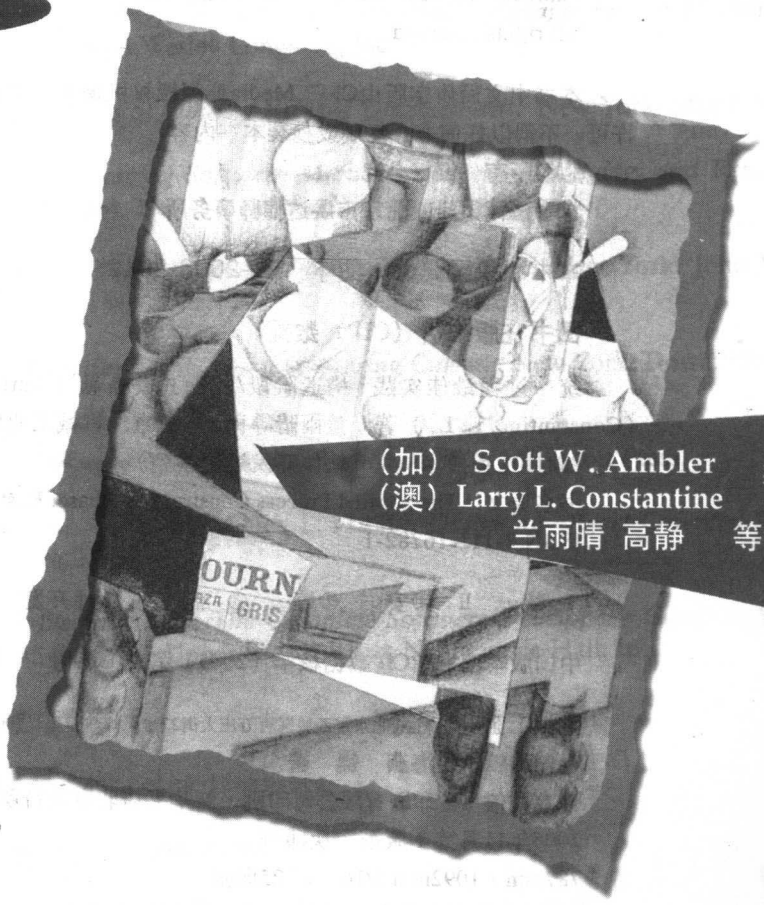
机械工业出版社
China Machine Press

软 件 工 程 技 术 丛 书

统一过程最佳实践

构造阶段

软件开发过程系列



（加） Scott W. Ambler
（澳） Larry L. Constantine 著
兰雨晴 高静 等译

The Unified Process
Construction Phase
Best Practices
in Implementing the UP

机械工业出版社
China Machine Press

本套书汇集了两位作者丰富的软件过程经验、10余位业界杰出人士的亲身体会以及《软件开发》及其前身《计算机语言》杂志中的精彩论文，提出了软件开发过程中的最佳实践方法，指导读者有效而且高效地执行这些过程。同时，作者还综合了统一过程和其他软件过程形成了一个处理真实世界软件开发和产品需要的更完整、更健壮的统一过程。

本套书共有四本，其中介绍的最佳实践方法分别对应统一软件过程的四个阶段：初始阶段、细化阶段、构造阶段、移交和产品化阶段。本书是这套书的第3本，重点介绍与统一软件过程构造阶段有关的最佳实践。

本书可以作为软件项目管理人员、软件开发工程师、过程工程师、系统工程师等专业人员的指导用书，也可作为高等院校计算机及相关专业学生的参考书。

Scott W. Ambler, Larry L. Constantine: The Unified Process Construction Phase: Best Practices in Implementing the UP (ISBN 1-929629-01-X).

本书版权登记号：图字：01-2003-3912

图书在版编目（CIP）数据

统一过程最佳实践　构造阶段/（加）安布勒（Ambler, S. W.），（澳）康斯坦丁（Constantine, L. L.）著；兰雨晴等译. -北京：机械工业出版社，2005.11

（软件工程技术丛书　软件开发过程系列）

书名原文：The Unified Process Construction Phase: Best Practices in Implementing the UP

ISBN 7-111-16782-1

Ⅰ.统…　Ⅱ.①安…　②康…　③兰…　Ⅲ.软件开发　Ⅳ.TP311.52

中国版本图书馆CIP数据核字（2005）第070359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：盛海燕　姚　蕾

北京诚信伟业印刷有限公司印刷·新华书店北京发行所发行

2005年11月第1版第1次印刷

787mm × 1092mm 1/16 · 15.25印张

印数：0 001- 3 000册

定价：35.00元

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

译者序

软件产业的不断发展，使人们越来越认识到优化的过程在软件开发和生产过程中的重要作用，遵循良好的过程是高效率、高质量、低成本地开发和生产软件的必由之路。

近年来，我国软件业发展迅猛，在众多的软件项目和软件产品的开发过程中，人们也越来越意识到优化的软件过程对于保证这些软件项目和产品质量的重要性，围绕软件过程改进进行的专题培训、企业内部培训等越来越多，业界的一些专家、学者围绕这一领域编著、翻译的书籍也很多。然而略感遗憾的是，人们在认识到软件过程重要性的时候，对如何有效且高效地实践这些过程却还未找到最佳方法，这方面的论著目前也比较匮乏。

笔者翻译的本书及本套丛书的其他书中，详细介绍了原作者在他们大量的软件过程经验和收集的来自于十余位业界杰出人物以及十多年的《软件开发》和《计算机语言》杂志的更广泛的经验的基础上提出的软件过程最佳实践方法，并将这些方法和统一过程的各阶段对应起来。

在本套丛书中，作者综合Rational 统一过程（RUP）、OPEN联盟的OPEN过程、面向对象软件过程（OOSP）、极限编程（XP）等软件过程形成了一个处理真实世界开发和产品需要的更完整、更健壮的统一过程。在详细阐述一个具有更完整的增强生命周期的统一过程之后，每一卷书都介绍了当前实现统一过程各个阶段（初始、细化、构造、移交和产品化）最佳实践的大师的经验智慧集合。通过本书作者提供的资源链接，读者可获得更广泛的“在线知识库”。

本书介绍的最佳实践方法与统一软件过程的构造阶段相对应，介绍了项目管理工作流、需求工作流、基础设施工作流、业务建模工作流、分析和设计工作流、实现工作流、测试工作流、部署工作流、配置和变更管理工作流与环境工作流的最佳实践。构造阶段——统一过程的4个阶段里的第3个阶段——致力于把在细化阶段开发的技术原型演化成较为完备的系统，是为应用程序的详细设计进行相应编码的阶段。此阶段的目标是生产交付给用户的软件及相应的支持文档。在本书中，一系列经过精心挑选的文章讲述了项目组可以遵循的最佳实践，以及掌握这个阶段的关键要素。主题包括：如何尽早达到并维持足够的质量，如何开发软件模型来指导实现，如何与用户一起工作以确认需求履行，如何实现并测试不同的系统构件，如何尽早在实践中开发有用的系统版本，如何为确认的构件建立基线，以及如何有效地管理包括风险在内的项目资源。

笔者翻译本书的目的是为实践这一过程提供最佳实践参考。本书可作为软件项目管理人员、软件开发工程师、过程工程师、质量工程师、系统工程师、系统分析员等

的软件过程实践指导用书，也可作为高等院校计算机及相关专业学生的软件工程实践参考书。

参加本书翻译工作的有兰雨晴、高静、王力光、黄勇，全书由高静统稿，兰雨晴进行了审定。由于水平和时间有限，本书翻译中的缺点和错误在所难免，个别用词还有待商榷，真诚欢迎读者批评指正。

联系：Lanyuqing@buaa.edu.cn，Gaojing@cse.buaa.edu.cn

北京航空航天大学软件工程研究所

兰雨晴　高　静

2005年夏

序

构造是令人激动的——它把一系列人类的活动集中爆发出来，声势浩大，成效显著，给人印象深刻。你可能正为即将开始阅读来自《软件开发》杂志中关于构造主题的经典文章而兴奋不已，而且准备好汲取里面有用的知识马上应用到你的项目中去。然而，当你开始之前，请你先用一点时间回顾一下你的计划和准备，问问自己它们是不是真的充分了。你定义你的业务用例了吗？最重要的一点是，你确定项目的范围了吗？你的系统构架是否足够强壮和优异以支持你的需求？如果没有，在本系列丛书的前两卷（《统一过程最佳实践·初始阶段》和《统一过程最佳实践·细化阶段》）里，Scott Ambler和Larry Constantine将帮助你做好这些准备。

我们可以理解，许多软件项目由于强调直接跳跃到编码阶段，随后又在产品开发陷入困境时再来修改弥补，而陷入了崩溃的境地。确实，许多方法都以编码为核心活动。但不幸的是，许多开发人员都误解了Kent Beck在《极限编程》（*Extreme Programming*）（AddisonWesley Longman，2000年）一书中的建议，并因此认为理所当然可以忽略，软件过程中的准备方面。无论遵循的是Beck的方法、统一过程、OPEN过程，还是Scott Ambler自己的方法——面向对象软件过程，软件工程师必须认识到每个过程都用某种形式来处理完整的生命周期，不应该忽视这些过程中的任何一个阶段。要顾及到生命周期中的其他方面，适度是最好的解决方案。

我要提醒大家注意只有40%的开发人员真正具有计算机科学的学位，而其中只有少数人具有软件工程的学位。尽管随着软件产业的成熟，以及软件产业对于公众和关键安全系统的广泛影响终于得到世人的承认，上述情况有所改观，但大多数的开发人员还是通过自学来学习专业知识。那些仅仅专注于新技术的人将会失败，原因是他们重复着前人的错误，留下了安全漏洞，会因为进度超期而窘迫不堪，或者系统实现了功能但设计不佳——换句话说，直接变成遗留的系统。而那些哪怕只花一点时间来学习管理整个生命周期的人将构建出通行世界的软件。

Alexandra Weber Morales

《软件开发》（Software Development）主编

前　言

在《软件开发》杂志和它的前身《计算机语言》中已经刊登了大量关于如何成功开发软件的文章。为这一杂志撰稿的人包括许多业界最著名的专家，比如Karl Wiegers、Steve McConnell、Ellen Gottesdiener、James Bach、Jim Highsmith、Warren Keuffel和Lucy Lockwood。简而言之，信息产业的大师们在这些年里一直在这本值得尊敬的杂志中与我们分享他们的智慧成果。

近来，在几乎所有的组织中，对软件过程改进的关注越来越多了。这有一部分是因为千年虫（Y2K）问题、大规模软件项目的高失败率以及人们渐渐意识到遵循成熟的软件过程是软件项目成功的关键因素。从20世纪90年代中期开始，Rational公司控股和合并了其他一些软件工具公司；随着公司的发展，这些工具所支持的各种过程也被合并成一种开发方法，称为“统一过程”（Unified Process）。是否有可能让整个软件过程自动化？如果有可能，那么Rational公司是否拥有一套完整的工具集？对上述问题我们并不确定。但幸运的是，其他人也在定义软件过程，所以我们还可以从多个角度来看事物应怎样运作。这些过程包括：OPEN联盟的OPEN过程、面向对象软件过程（OOSP）的过程模式以及极限编程（XP）。这些不同的视角可以用来推动统一过程观点，使其更加健壮，结果就产生了一个更能准确反映你所在组织现实需要的增强的统一过程生命周期。因为我们相信《软件开发》中包含的多年收集的智慧能够用来充实统一过程——真正将我们产业的最佳实践统一起来，所以我们编写了本系列丛书。

为什么软件过程如此重要呢？让我们先设想一下。假如你想请人给你建造一间房子，让两位承包商来竞标。第一位承包商告诉你，通过使用一项最新的建筑技术给你盖房，如果从明天就开始的话，他能在两个星期内就把房子建好，造价只有10万美元。这个承包商手下有一流的木匠和水管工，他们以前用这项新技术建造过一个花园凉棚，他们愿意日夜加班以按期交付你的新屋。而第二位承包商告诉你，她需要先和你讨论你想要建一间什么类型的房子。然后，一旦她确定明白你的需要，她将在一个星期内提供一整套设计蓝图供你审阅和提意见。这个初始阶段只会花你1万美元，当你决定了最终方案，对于其余的工作她将给出详细计划和成本估算。

你会觉得选哪个承包商更放心呢？是想马上开始建房的那个，还是先搞清楚要建什么样的房子，再建模型，再详细计划，最后动工修建的那个？显然，后者更有可能成功地交付给你一间符合你实际需要的房子。现在，设想你要构建的是软件——这通

常是复杂得多而且远比房子更昂贵的项目，再设想你还是面对两个与前面采取相同方法的承包商。选择哪个你会更放心呢？希望你的回答仍是第二个；她有一个更明智的过程。但不幸的是，实践显示：在大多数时间里，组织似乎喜欢选择第一个承包商的方法；热衷于新技术而忽视过程。当然，实践也显示：在我们的产业里，建造大型的、具有关键任务的系统的失败率在85%以上。也许这两种现象有一定的关联。

构造阶段

构造阶段是软件开发5个阶段（初始、细化、构造、移交和产品化阶段）中的第3个阶段，每个软件的发布版本在其生命周期内都将遍历这些阶段。构造阶段的目标是：

- 描述剩下的需求。
- 充实你的系统设计。
- 保证你的系统达到用户的要求，并与你所在组织的系统整体设计相适合。
- 完成构件开发和测试，包括软件产品和它的文档。
- 依靠资源优化来把开发成本降到最低。
- 尽快地获得足够高的质量。
- 为你的系统开发有用的版本（α版、β版等）。

本书向读者呈现了业界专家所撰写的描述软件领域最佳实践的文章。本书乃至本系列丛书的一个目标是提供已证实的统一过程所包含技术的可替代方案。另一个目标是弥补统一过程中的一些缺陷。因为统一过程是一个开发过程，而不是软件过程，它不可避免地遗漏或缺少了一些对软件专业人员来说非常重要的概念。幸运的是，《软件开发》杂志的作者们已经对过程范围有了更广泛的了解，并已经为我们弥补了许多缺陷。

关于本套丛书

本套丛书由四卷组成：第1卷介绍初始阶段，第2卷介绍细化阶段，第3卷介绍构造阶段，第4卷介绍移交和产品化阶段。每卷都可独立成书，但是如果想对整个软件过程有一个完整的认识，你需要通读全套丛书。本套丛书的文章覆盖了整个过程，在每卷之间没有重复。

我们在为本书选择材料时确实费了一番心思，有大量可以选择的材料，但是篇幅有限，缩小选择范围并不总是那么容易。如果时间和篇幅没有限制，那么每一本书都可能会有现在的两倍那么厚。通过缩小选择范围，我们相信留下来的文章一定都是精华中的精华。

关于编者

Scott W. Ambler

这是我最喜欢的话题了！作为《计算机语言》和后来的《软件开发》杂志的热心读者，我于1995年开始为杂志写稿，并在1997年终于成为对象专栏的作家。我从20世纪80年代早期开始从事软件开发，用Fortran和Basic等语言编写代码；到20世纪80年代中期，我开始使用

Turing、C、Prolog和Lisp语言编码。在20世纪80年代末期，我发现生活中除了编程之外还有更多的东西，于是在用COBOL和几种第四代语言为IBM大型机编程的同时，我又开始学习用户界面设计、数据建模、过程建模以及测试等多方面的技巧。到了20世纪90年代，在我对结构化/过程化技术大失所望之后，我发现了对象技术，并跳跃到Smalltalk开发，然后转为C++开发，最后又转回Smalltalk。我曾先后在多个组织中担任顾问和架构师，之后我决定结合这些经验，并运用我在多伦多大学当助教时所获得的技巧，并且在20世纪90年代中期开始做职业培训。我很快学到了几件事情。第一，尽管我喜欢教培训课程（直到现在我还在做这项工作），但是我自己并不想全职做这件事。第二，也是更重要的，我学会了如何用简单易懂的方式交流复杂的概念，比如如何开发面向对象软件。这就促使我写下了我最初的两本书：《The Object Primer》（Cambridge University Press，1995/2000）和《Building Object Applications That Work》（Cambridge University Press，1997/1998），这两本书以开发人员的观点描述了对象技术的基本原理。然后我决定再写两本书来描述面向对象的软件过程（OOSP），这两本书是《Process Patterns》（Cambridge University Press，1998）和《More Process Patterns》（Cambridge University Press，1999），其中讲述了我在加拿大一家顶尖的对象技术咨询公司工作时所获得的来之不易的经验。从那时起，我帮助过业界的几个组织机构（大的和小的，新成立的和历史悠久的），帮助它们改善其内部的软件过程。我最新的作品包括本系列丛书以及与别人合著的《The Elements of Java Style 》（Cambridge University Press，2000）。我现在是Ronin International公司（www.ronin-intl.com）的总裁，这是一家提供软件过程和软件构架指导和咨询的公司，总部设在丹佛；同时我还是我自己的网站（www.ambysoft.com）的自由作家，在这个网站上我张贴了许多白皮书。我想，我终于找到适合自己的小天地了。

Larry L. Constantine

我与《软件开发》及其前身《计算机语言》的联系是经久而富有成果的，但比起我与软件开发以及计算机语言的联系，又是小巫见大巫了。从我在计算机的摸索时代所写的第一个Fortran程序开始，我就一直热衷于寻找能把事情做得更好并能帮助其他人做得更好的方法——正是这种兴趣让我不久就超越了技术领域而进入了管理和过程，以及我们所设计和构建软件的可用性的本质问题。我在将近40年间在这一领域内摸爬滚打，一直在与把人与技术相分离的现象抗争。在我看来，软件开发的成功取决于对这两方面的理解和掌握，这一点也同样反映在我为杂志和其他刊物所写的文章中。我已经写了150多篇文章和论文以及14本书，其中包括现在这本与Scott Ambler合作编辑出版的书。经过Scott的同意，我自己在杂志中的一些专栏和文章也被收录到这套书中。其他的文章收录在《The Peopleware Paper》（Prentice Hall，2000）和《Managing Chaos: The Expert Edge in Software Development》（Addison-Wesley，2000）中。前者重印了我的长期专栏“人件”中的内容；后者集中了广受欢迎的“软件开发管理论坛”每期印在最后的精华。在最近几年里，我的专业兴趣集中在增加软件的可用性上，这导致了我在以使用为中心的设计方面的发展，也促使我与Lucy Lockwood合著的《Software for Use: A

Practical Guide to the Models and Methods of Usage-Centered Design》(Addison-Wesley, 1999)一书的诞生。《软件开发》杂志评选该书为1999年最佳书籍，授予我们Jolt大奖。最近，我频繁地跨洋旅行，因为虽然我住在美国，但是我同时还要到澳大利亚悉尼理工大学授课，我是这所大学计算机科学系的兼职教授。尽管题目是计算机科学，但我讲授的都是管理和设计问题的混合主题。我还是一位勤奋的职业培训师、设计师以及咨询顾问，帮助世界各地的客户建造更易使用的软件。我与Lucy Lockwood一起创立了Constantine & Lockwood有限公司(www.forUse.com)，由我担任研发主管。现在除了原有的工作，我们正致力于将以使用为中心的设计与统一过程和统一建模语言（UML）结合起来。

目　录

译者序
序
前言
第1章　介绍 ……1
1.1　统一过程 ……1
1.2　统一过程的增强生命周期 ……5
1.3　构造阶段的目标 ……7
1.4　在构造阶段工作一般怎样进行 ……8
1.4.1　项目管理工作流 ……8
1.4.2　业务建模工作流 ……9
1.4.3　需求工作流 ……9
1.4.4　基础设施管理工作流 ……10
1.4.5　分析和设计工作流 ……10
1.4.6　实现工作流 ……11
1.4.7　部署工作流 ……11
1.4.8　测试工作流 ……12
1.4.9　配置和变更管理工作流 ……12
1.4.10　环境工作流 ……12
1.5　本书的组织 ……13
第2章　项目管理工作流 ……15
2.1　项目管理最佳实践 ……15
2.2　极限编程 ……17
2.3　在死亡行军中生存 ……17
2.4　文章 ……17
2.4.1　“领导课程” ……18
2.4.2　“成功项目管理的秘密” ……20
2.4.3　“针对最佳团队绩效的时间定量” …23
2.4.4　“极限编程” ……27
2.4.5　“在‘死亡行军’项目中生存” ……29
第3章　基础设施管理工作流 ……33
3.1　战略性复用管理 ……34
3.2　通过框架进行企业复用 ……36
3.3　通过构件进行企业复用 ……36
3.4　文章 ……37
3.4.1　“复用发生什么情况了” ……38
3.4.2　“复用诱惑” ……41
3.4.3　“对面向对象复用的现实观察” ……45
3.4.4　“复用模式和反模式” ……53
3.4.5　“把复用变为现实” ……55
3.4.6　“常见的复用者距离” ……60
3.4.7　“改进框架可用性“ ……63
3.4.8　“使框架有价值” ……68
3.4.9　“构件构建者的规则” ……72
3.4.10　“构件与Catalysis/UML” ……77
3.4.11　“构件：逻辑的、物理的模型” ……86
第4章　分析和设计工作流 ……91
4.1　学习面向对象分析和设计的基础知识 …91
4.2　分析和设计基本原理 ……92
4.3　建模最佳实践 ……93
4.4　给用户界面设计定案 ……94
4.5　面向对象持久建模 ……94
4.6　文章 ……95
4.6.1　“建模心灵鸡汤” ……96
4.6.2　“构件的重要性” ……100
4.6.3　“把类标准化” ……103
4.6.4　“以接口为中心的设计” ……106
4.6.5　“模拟继承” ……113
4.6.6　“注重细节” ……114

4.6.7 “界面多元化” ……119
4.6.8 “从对象到关系型数据库的映射的实现” ……121
4.6.9 “穿过数据和对象之间的划分，第一部分” ……123
4.6.10 “穿过数据和对象之间的划分，第二部分” ……126
4.6.11 “映射对象到关系数据库” ……130
4.6.12 “在UML中持久性建模” ……136
4.6.13 “企业级对象标识” ……140
第5章 实现工作流 ……145
5.1 重新编写代码 ……147
5.2 编程最佳实践 ……147
5.3 文章 ……148
5.3.1 “抛弃软件” ……148
5.3.2 “编写可维护的面向对象应用程序” ……152
5.3.3 “当用代码无法表达的时候，就采用注释” ……160
5.3.4 “可移植的代码是更好的代码” ……161
5.3.5 “编写严密的代码” ……162
第6章 测试工作流 ……167
6.1 测试最佳实践 ……168
6.2 协同工作 ……169
6.3 文章 ……169
6.3.1 “编写健壮的回归测试” ……169
6.3.2 “一个UML测试框架” ……174
6.3.3 “恢复” ……179
6.3.4 “调和差异” ……181
第7章 配置和变更管理工作流 ……187
7.1 配置管理 ……187
7.2 变更管理 ……188
7.3 可跟踪性 ……188
7.4 文章 ……189
7.4.1 “创建配置管理文化” ……189
7.4.2 “来自DLL炼狱的问候” ……193
7.4.3 “实现版本描述文档” ……196
7.4.4 “软件变更管理” ……200
7.4.5 “跟踪你的设计” ……205
7.4.6 “演化类图” ……211
第8章 结束语 ……217
附录A 参考书目 ……219
附录B 有贡献的作者 ……223
附录C 参考资料和推荐读物 ……227

第1章 介 绍

什么是软件过程呢？一个软件过程就是人们用来开发和维护软件及其相关制品（如项目计划、文档、模型、源码、测试案例、手册等）的一套项目阶段、状态、方法、技术，以及实践。关键的是，你需要的不仅仅是一个软件过程，而且是一个在实践中被证明是可行的，可以裁减以适合你的实际需要的软件过程。

为什么你需要一个软件过程呢？一个有效的软件过程能使你所在组织提高软件开发的生产率。首先，通过了解软件怎样开发的基础知识，你能够做出明智的决策，比如知道要远离SnakeOilv 2.0——这一工具声称可以自动化软件过程基础部分。其次，它使你能够规范化自己的工作，还能促进项目组之间的复用和一致性。最后，它还提供机会让你把诸如代码审查、配置管理、变更控制和构架建模这些业界最佳实践引入你所在的软件组织。

一个有效的软件过程同样还能改进你所在组织的维护和支持工作，从一些方面看，这些工作也被归入到产品化工作。首先，它应该定义如何管理变更并恰当地分配维护变更到软件将来的发布中，以便使得变更过程更有效率。其次，它应该定义如何将软件平滑地转变为操作和支持，以及如何实际执行操作和支持工作。如果没有有效的操作和支持过程，软件很快就会变成搁置品。

> 一个有效的软件过程既考虑开发的需要也考虑产品化的需要。

为什么要应用现有的软件过程，或使用新技术来改进你的现有过程呢？现实在于：软件正变得越来越复杂，如果没有一个有效的方法开发和维护这些软件，会降低成功的可能性。不光是软件正变得越来越复杂，你还会被要求同时开发多个软件。大多数组织都普遍有多个项目同时在开发并且同时另有多个处于产品化阶段——项目因此需要有效的管理。此外，我们的产业正处于危机之中；我们仍在千年虫这个简单的把年份从两位表示转换为四位表示的问题上步履蹒跚，这号称是一个“较小的”问题，却让全世界付出了6000亿美元的代价。我们正在构建的软件的性质同样也在改变——从20世纪70年代使用结构化技术的简单批处理系统，到以面向对象和基于构件技术为目标的交互的、国际化的、用户友好的、24×7的，高事务率的、高可用性的在线系统。当开发软件的时候，被要求增加你将交付的系统的质量，并且尽可能的复用以便软件开发更迅速和更廉价。如果不能有效组织和管理你的团队，对于一个高要求的订单，要想做到这点几乎是不可能的。一个软件过程恰好为这些要求提供了一个基础。

> 软件正变得更加复杂，而不是相反。

1.1 统一过程

统一过程是Rational公司（Kruchten，1999）成员的最新努力成果，同样是这些人还引进了

已经成为行业标准建模符号的统一建模语言（UML）。统一过程的核心是对象工厂过程，这项技术是几年前Rational公司和Ivar Jacobson的对象工厂组织合并时获得的产品和服务中的一个。Rational用他们自己的过程（还包括其他被Rational收购或合作伙伴的软件工具公司的过程）增强了对象工厂，并于1998年12月形成并发布了统一过程的最初正式发布版本（5.0）。

图1-1显示的是统一过程的初始生命周期，沿着图表的底部可以看出贯穿整个统一过程的所有的开发周期都应该以迭代的形式组织。基本的思路是：工作组以迭代的方式工作在恰当的工作流中，以便各次迭代结束时都可产生和用户方共同工作的内部可执行产品。这样通过增加与用户的交互降低了项目的风险。另一个内置于统一过程中降低风险的技术是应该在各阶段末尾做出“继续/中止”的决策。如果项目将会失败，你一定想尽早中止它——对于大型关键项目的失败率达到80%～90%以上的行业，具备这一思路是非常重要的（Jones, 1996）。

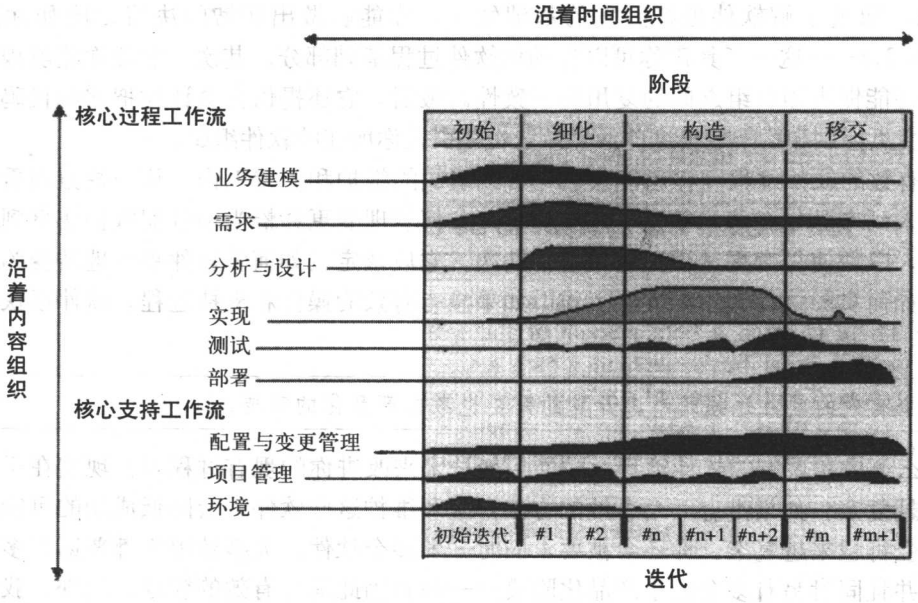


图1-1 统一过程的初始生命周期

初始阶段主要用于定义系统的项目范围及业务用例。在本阶段要确定软件的最初用例并简要描述关键的用例。用例是业界定义系统功能需求的一项标准技术。由于用例所关注的是系统对于用户的价值而不是对于产品特征的，因此，它们比传统的需求文档显著地提高了生产率。基本的项目管理文档是在初始阶段开始的，包括最初的风险评估、估算及项目进度等。正如人们所预想的，本阶段的关键任务包括业务建模和需求工程以及环境的最初定义，包括工具的选择和过程的定制等。

初始阶段定义项目范围和业务用例。

细化阶段关注于对问题域的详细分析和对项目构架基础的定义。因为用例还不足以定义所有需求，还需要定义一份称为补充说明的可交付品，用来描述你的系统的所有非功能性需求。

在这个阶段里，还应在初始阶段开始的初始管理文档的基础上开发出一份详细项目计划，以供构造阶段使用。

> 细化阶段定义系统构架基础。

构造阶段，也是本卷的主题，是为应用程序的详细设计进行相应编码的阶段。此阶段的目标是生产交付给用户的软件及相应的支持文档。项目组所犯的一个最常见的错误就是将重点放在这个阶段，由于组织没有为前两个阶段提供足够的资源投入，缺乏成功开发满足用户需求的软件的基础，因此通常这个错误对项目有害。

> 在构造阶段完成最后将部署的系统。

移交阶段的主要目标是将系统交付给用户。通常会先给用户软件的一个β版本——在大多数企业中称为引导版本（pilot release）——即在将软件发布给所有用户之前先由一小部分用户使用系统。在此阶段识别主要的缺陷并逐步进行修复。最后对投入进行评估，确定是否需要下一个开发周期以便进一步改进系统。

> 在移交阶段交付系统。

统一过程有以下几个优点。首先，统一过程基于合理的软件工程原理，例如在开发中采用迭代、需求驱动、基于构架的方法，在开发过程中进行增量发布。其次，统一过程提供了几种机制，如：每次迭代末期的工作原型，在各阶段末期的“继续/中止”决策等，这些提供了对开发过程的管理可见度。第三，Rational已经并且将继续对其统一过程（RUP）产品进行大力投资（http://www.rational.com/products/rup）。这是一个基于HTML的统一过程的描述，可以用来进行裁剪以适合不同的需要。

统一过程也有以下几个缺点。首先，它仅仅是一个开发过程。统一过程的最初版本并没有覆盖整个软件过程。正如在图1-1所看到的，很明显它遗漏了当软件发布为产品之后的操作和支持的概念。其二，统一过程并没有明确地支持多项目基础设施开发，例如组织、企业级架构建模，该部分在第5章讨论（该章描述基础管理流程），错过了组织内进行大范围复用的机会。第三，生命周期的迭代特性对于许多有经验的开发者来说是全新的，因此接受它们有一定难度，而且图1-1所示的生命周期图对此并没有提供帮助。

在统一过程的细化阶段，也就是本系列丛书的第二卷，展示了可以轻易地增强统一过程来满足现实世界开发的需要。我认为你需要从对过程的需求开始，能力成熟度模型（CMM）是一个非常好的开始。其次，你应当看看其他竞争者，即OPEN过程（Graham，Henderson-Sellers，和Younessi，1997）（http://www.open.org.au）和面向对象软件过程的过程模式（Ambler 1998b，Ambler 1999），看看从这些过程中你可以复用哪些特性。图1-2描述了OPEN过程的合同驱动的生命周期，图1-3描述的是面向对象软件过程（OOSP）的生命周期，由一系列过程模式组成。最后，应该基于所学的给出一个增强的生命周期并且用经过验证的最佳实践来支持它。

统一过程是一个好的开端，但它似乎还需要作一些裁剪和增强来满足组织的特殊需要。

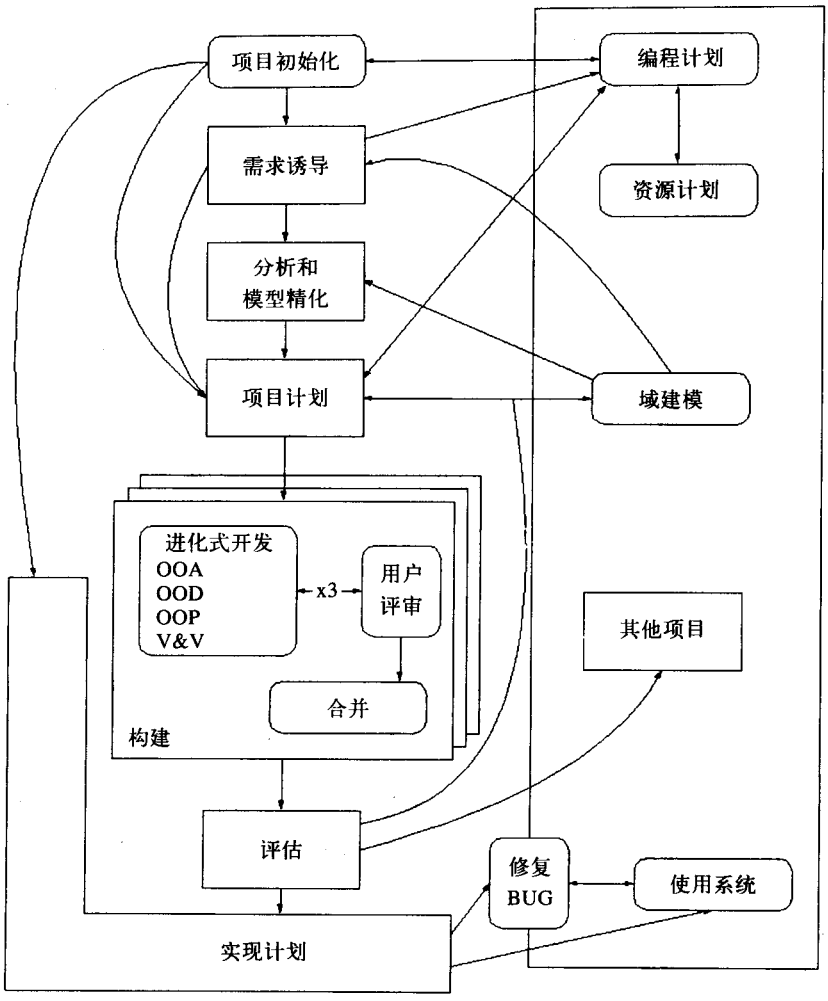


图1-2 OPEN过程的合同驱动的生命周期

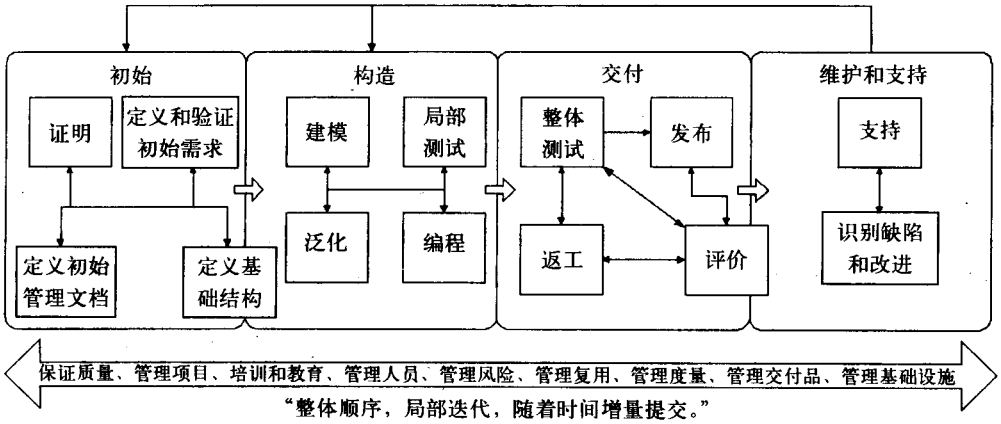


图1-3 面向对象软件过程（OOSP）的生命周期

1.2 统一过程的增强生命周期

我们已经看到了一个成熟软件过程的需求概述以及软件过程的两个竞争类型。了解了这些之后，该如何完成统一过程呢？首先，重新定义软件过程的范围使其包括完整的软件过程而不仅仅是开发过程。这意味着需要加入操作、支持及维护的软件过程。其次，为了满足当前组织的充分需要，统一过程同样需要支持项目的组合管理——类似于OPEN过程中所说的“程序管理”和OOSP中所说的基础设施管理。以上两步带来了图1-4所描述的生命周期增强版。最后，需要用已经证明是良好的实践来充实统一过程，关于这点，可以在《软件开发》中找到相应的文章。

增强的生命周期包括第5阶段——产品化阶段，描绘了系统被部署后软件生命周期的部分。正如此阶段其名字所暗示的那样，该阶段目的就是让软件一直保持在产品状态，直到它被更新的版本替代（从类似于修复bug这类的小的新版本发布到重要的新的版本发布），或者完成其使命退出产品状态。注意在这个阶段并没有迭代（或者说仅有一次迭代，这取决于你如何看待它），这是因为这个阶段应用于软件某一发布版本的整个生命过程。要开发和部署软件的新版本，需要再重新编历4个开发阶段。

> 产品化阶段包括生命周期的部署之后的部分。

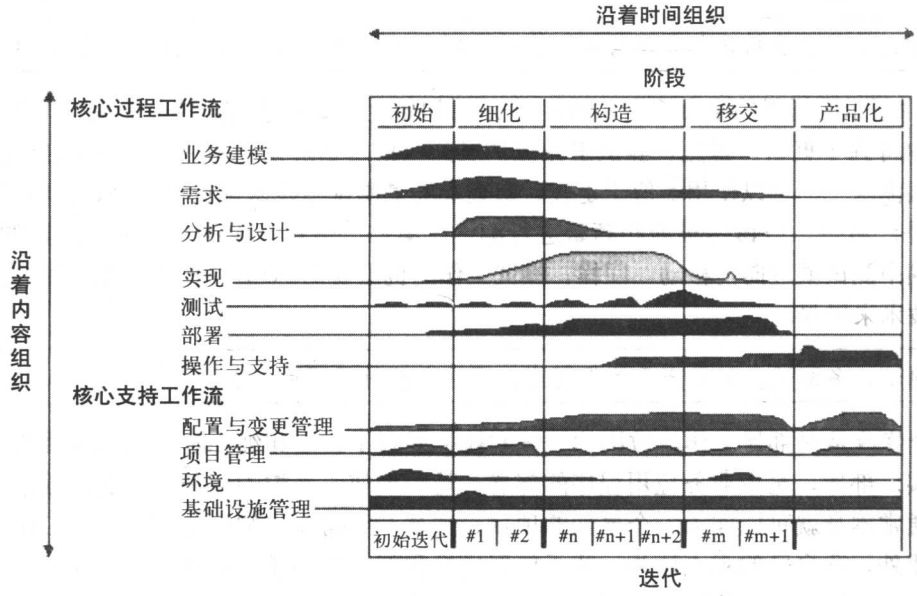


图1-4　统一过程的增强生命周期

图1-4同时还显示出，有两个新的工作流：一个过程工作流称为操作和支持，一个支持工作流称为基础设施管理。“操作和支持”工作流的目的正如其名字所暗示的，即操作和支持软件。操作和支持都是复杂工作，是需要为它们来定义过程的工作。这个工作流，就像其他所有工作流那样，跨越了多个阶段。在构造阶段，需要开发出操作和支持计划、文档、培训手册。在移交阶段，将继续开发这些工件，基于测试的结果修改它们，培训操作和支持人员来有效的保持