HZ BOOKS

软 件 工 程 技 术 丛 书

质量管理系列

PRENTICE HALL PTR

面向对象的软件缺陷管理

Object-Oriented Defect Management of Software

（美） Houman Younessi 著

赵文耘 沈钺 等译

机械工业出版社
China Machine Press

软 件 工 程 技 术 丛 书

质量管理系列

面向对象的软件缺陷管理

Object-Oriented Defect Management of Software

（美） Houman Younessi 著

赵文耘 沈钺 等译

机械工业出版社
China Machine Press

随着软件的增加，需求越来越复杂，维护成本越来越高，如何提高软件的质量、进行有效的缺陷管理就变得越来越重要。本书针对这个问题，提出面向对象的软件缺陷管理的概念。本书主要介绍面向对象技术在缺陷管理方面的特殊性、缺陷预防和缺陷标识，以及如何在软件生命周期的各个阶段进行缺陷管理。书中还给出大量的模板和检查列表，方便读者使用。本书概念清晰，讲述透彻，适合软件工程师、从事面向对象技术与缺陷管理领域工作的技术人员、研究人员阅读，也可作为高校软件工程专业本科生、研究生的教材，或面向对象缺陷管理领域的培训教材。

本书版权登记号：图字：01-2002-5454

图书在版编目（CIP）数据

面向对象的软件缺陷管理/（美）尤尼西（Younessi，H.）著；赵文耘等译．-北京：机械工业出版社，2004.1
（软件工程技术丛书　质量管理系列）
书名原文：Object-Oriented Defect Management of Software
ISBN 7-111-13105-3

Ⅰ.面…　Ⅱ.①尤…②赵…　Ⅲ.软件质量-质量管理　Ⅳ.TP311.5

中国版本图书馆 CIP 数据核字（2003）第 087878 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：朱　劼
北京牛山世兴印刷厂印刷·新华书店北京发行所发行
2004 年 1 月第 1 版第 1 次印刷
787mm×1092mm 1/16·14.75 印张
印数：0 001-5 000 册
定价：30.00 元

凡购本书，如有倒页、脱页、缺页，由本社发行部调换
本社购书热线电话：（010）68326294

前　言

可能你选择本书的初衷只是想了解更多面向对象程序测试的知识，但是实际上你的收获将远不止于此。正如书名所示，本书主要讨论面向对象软件系统的缺陷管理，但本书所介绍的内容已经远远超出了面向对象代码测试的范围。

首先，我们讨论只有一种测试类型的缺陷管理。软件开发过程的任何阶段都可能引入缺陷。如果未经检测并发现这些缺陷，它们将在产品交付执行时成为错误或故障（bug）。测试的主要功能是执行程序并以程序执行失败为目标，这一活动可以发现许多缺陷。另外，还有些技术只需要检查过程中的各种制品，而不检查可执行的代码。我们将前者称为静态技术，后者称为动态技术。在本书中，将详细讨论缺陷管理的静态和动态方法。

其次，测试是在产品投入使用前清除缺陷的最后的机会。因此，采用基于代码测试的方法（一种严格的纠错性方法）与软件工程的原则产生抵触。我们还同时采用预防性方法。从整体上来看，软件工程可以看做是软件产品的缺陷管理过程。这个断言听起来难以置信，其实不然。换一种角度来看，我们在软件过程中进行的活动的目的都是避免将缺陷引入产品或将已引入的缺陷识别出来并将其排除。也就是说，缺陷管理技术不仅应用在代码层次，还可以应用于软件工程过程的所有相关活动中。本书涵盖了软件工程过程各阶段的相关缺陷管理技术，而不仅介绍处理代码的技术。而且，纠正性和预防性方法都会在本书中加以讨论。

再次，本书针对的是面向对象环境中的缺陷管理。但这并不是说，本书介绍的所有缺陷管理技术是面向对象独有的。恰恰相反，本书提到的每个在面向对象环境下的有用的技术，可能原来并不是为该环境开发的。

最后，考虑到近来统一建模语言（UML）和 Rational 统一过程（RUP）已分别发展为面向对象过程的建模与管理的标准方法，所以我们将在逻辑上尽可能与其保持兼容。

除了以上特点之外，本书还可以以多种方式使用。本书适合专业软件工程师、对象技术与缺陷管理领域的研究人员和软件工程专业的研究生阅读，可作为面向对象缺陷管理领域高级课程的教材或面向对象软件工程领域高级课程的阅读材料，也可以作为面向对象缺陷管理领域的行业培训材料。

在本书面世之际，我要向许多人表达感激之情，尤其是：

- 所有帮助我完成此书的老师、学生和同事，有了你们的极力帮助，我才能完成此书。
- 我的妻子和最好的朋友 Sheyda，由于你的支持和理解才能使这本书出版。
- 我的儿子 Z. Daniel Younessi，他在文本处理与美工方面给予我极大的帮助。

- Prentice-Hall 的专家们，尤其是 Victoria Jones 女士、Carol Wheelan 女士和 Mike Meehan 先生，你们的努力使本书顺利出版。
- 我的读者，给了我写这本书的原动力。

衷心地希望这本书能对你有用。

Houman Younessi

South Windsor，Connecticut

2001 年 10 月

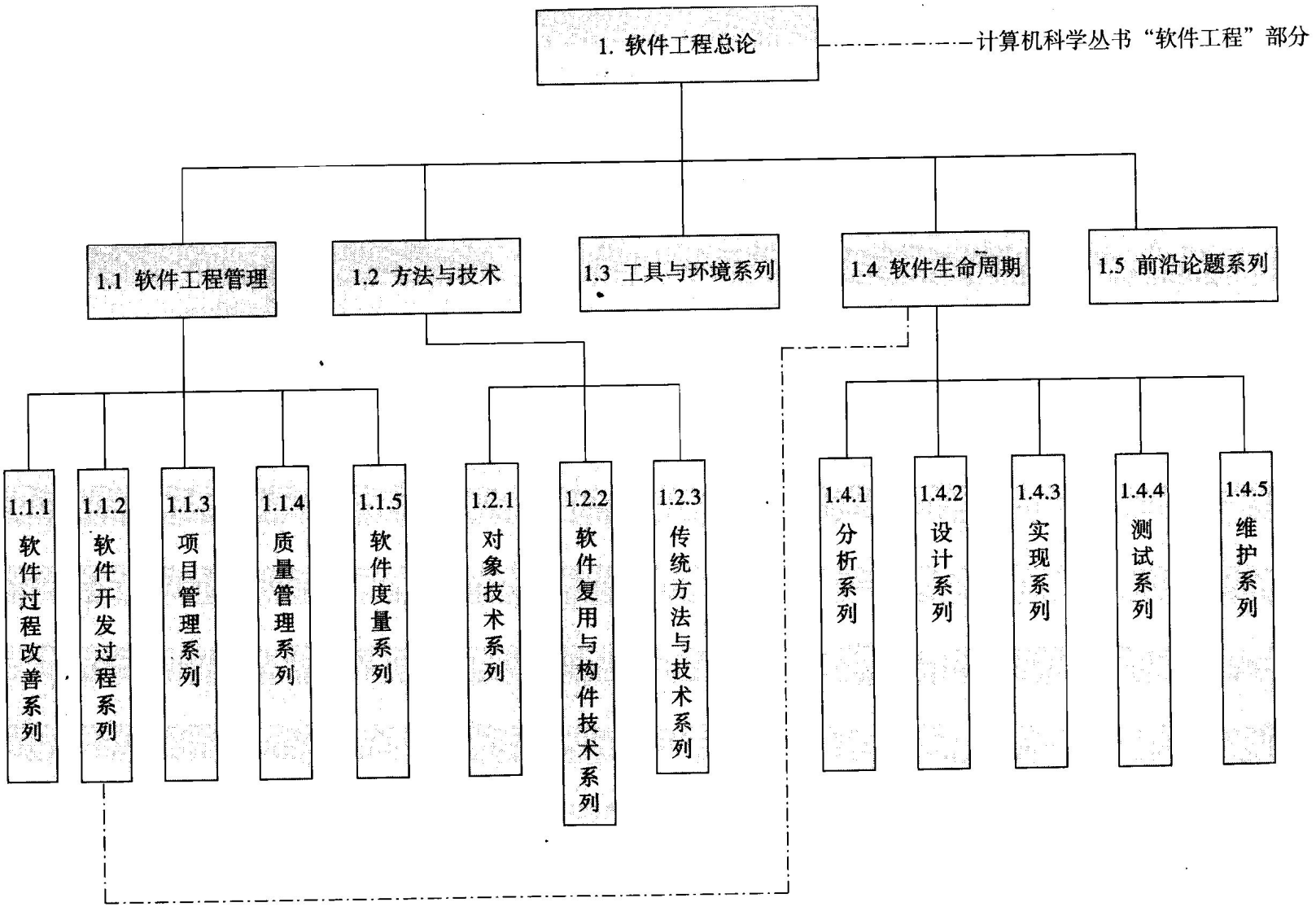
软件工程技术丛书书目

丛书编号	英文书名	中文书名	作　者
1	Object-Oriented and Classical Software Engineering, 5E	面向对象与传统软件工程(原书第5版)	Stephen R. Schach
1	Object-Oriented Software Engineering	面向对象的软件工程	Ian Sommerville
1	Software Engineering: A Practitioner's Approach, 5E	软件工程:实践者的研究方法(原书第5版)	Roger S. Pressman
1	Software Engineering, 6E	软件工程(原书第6版)	Ian Sommerville
1	Software Engineering with Java	软件工程:Java语言实现	Stephen R. Schach
1	Project-Based Software Engineering:An Object-Oriented Approach	基于项目的软件工程:面向对象研究方法	Evelyn Stiller
1.1.1	Software Process Improvement: Practical Guidelines for Business Success	软件过程改进	Sami Zahran
1.1.1	Making Process Improvement Work	软件过程改进简明实践	Neil S. Potter
1.1.2	The Road to the Unified Software Development Process	统一软件开发过程之路	Ivar Jacobson
1.1.2	The Unified Software Development Process	统一软件开发过程	Jacobson/Booch/Rumbaugh
1.1.2	The Rational Unified Process: An Introduction , 2E	Rational统一过程引论(原书第2版)	Philippe Kruchten
1.1.2	UML and The Unified Process: Practical Object-Oriented Analysis & Design	UML和统一过程: 实用面向对象的分析和设计	Jim Arlow
1.1.3	Managing Global Software Projects: How toLead Geographically Distributed Teams,Manage Processes and Use Quality Models	全球化软件项目管理	Gopalaswamy Ramesh
1.1.3	Software Project Management: A Unified Framework	软件项目管理:一个统一的框架	Walker Royce
1.1.3	How to Run Successful Projects III: The Silver Bullet	成功的软件项目管理:银弹方案(原书第3版)	Fergus O'Connell
1.1.3	Successful Software Development, 2E	成功的软件开发(原书第2版)	Scott E. Donaldson
1.1.3	Six Sigma Software Development	6σ软件开发	Christine B. Tayntor
1.1.3	IT Project Management: On Track from Start to Finish	IT项目管理: 从开始到结束的历程	Joseph Phillips
1.1.3	Successful IT Project Delivery: Learning the lessons of Project Failure	IT项目成功交付的秘诀	David Yardley
1.1.3	Software Project Management, 3E	软件项目管理(原书第3版)	Bob Hughes
1.1.3	Architecture-Centric Software Project Management	软件项目管理实用指南: 以体系结构为中心	Daniel J. Paulish
1.1.4	Handbook of Software Quality Assurance, 3E	软件质量保证(原书第3版)	Gordon G. Schulmeyer
1.1.4	Software Reliability Engineering	软件可靠性工程	John Musa
1.1.4	Implementing ISO 9001:2000 The Journey from Conformance to Performance	ISO 9001:2000 实施指南	Tom Tairnubg
1.1.4	CMMI Distilled: A Practical Introduction to Integrated Process Improvement	CMMI精粹:集成化过程改进实用导论	Dennis M. Ahern
1.1.4	CMM Implementation Guide	CMM实施与软件过程改进	Kim Caputo
1.1.4	Implementing the Capability Maturity Model	CMM实施指南	James R.Persse
1.1.4	Object-Oriented Defect Management of Software	面向对象的软件缺限管理	Houman Younessi
1.1.4	Metrics and Models in Software Quality Engineering	软件质量工程:度量与模型	Stephen H. Kan
1.1.4	Performance Solutions: A Practical Guide to Creating Responsive, Scalable Software	软件性能工程	Connie U. Smith

丛书编号	英文书名	中文书名	作　者
1.4.2	Object-Oriented Software Construction, 2E	面向对象的软件结构(原书第2版)	Bertrand Meyer
1.4.2	Pattern-Oriented Software Architecture, Vol I: A System of Patterns	面向模式的软件体系结构 卷1:模式系统	Frank Buschmann
1.4.2	Pattern-Oriented Software Architecture, Vol II: Patterns for Concurrent and Networked Objects	面向模式的软件体系结构 卷2:用于并发和网络化对象的模式	Douglas Schmidt
1.4.2	Server Component Patterns	面向模式的软件体系结构 卷3:服务器组件模式	Markus Volter
1.4.2	DesignPatterns: Elements of Reusable Object-Oriented Software	设计模式:可复用面向对象软件的基础	Gamma/Helm/Johnson /Vlissides
1.4.2	Patterns of Enterprise Application Architecture	企业级应用体系结构模式	Martin Fowler
1.4.2	AntiPatterns and Patterns in Software Configuration Management	软件配置管理中的模式与反模式	William J. Brown
1.4.2	Software for Use: A Practical Guide to The Models and Methods of Usage-Centered Design	面向使用的软件设计	Larry L. Constantine
1.4.2	Software Architecture: Organizational Principles and Patterns	软件架构: 组织原则与模式	David Dikel
1.4.2	The UML Profile for Framework Architectures	框架体系结构的UML档案	Marcus Fontoura
1.4.2	Software Architect's Profession: An Introduction	软件架构师入门必读	Marc Sewell
1.4.2	Business Modeling with UML: Business Patterns at work	UML业务建模: 实用业务模式	Hans-Erik Eriksson
1.4.3	Software Fabrication:Automating Application Development	软件构造:自动化软件开发	Jack Greenfield
1.4.3	Building J2EE Applications With The Rational Unified Process	用RUP构建J2EE 应用程序	Peter Eeles
1.4.3	Programming from Specifications	从规范出发的程序设计	Carroll Morgan
1.4.4	Testing IT: An Off-the-Shelf SoftwareTesting Process Handbook	实用软件测试过程之路	John Watkins
1.4.4	Lessons Learned in Software Testing	软件测试经验与教训	Cem Kaner
1.4.4	Testing Computer Software: The Bestselling Software Testing Book Of All Time, 2E	计算机软件测试(原书第2版)	Cem Kaner
1.4.4	Software Testing in the Real World: Improving the Process	软件测试过程改进	Edward Kit
1.4.4	Effective Methods for Software Testing, 2E	有效的软件测试方法(原书第2版)	William E. Perry
1.4.4	Beta Testing for Better Software	软件Beta测试	Michael R. Fine
1.4.4	A Practical Guide to Testing Object-Oriented Software	面向对象的软件测试	John D. McGregor
1.4.4	Managing the Testing Process, 2E	测试过程管理(原书第2版)	Rex Black
1.4.4	Software Testing: A Craftsman's Approach, 2E	软件测试(原书第2版)	Paul C. Jorgensen
1.4.4	Just Enough Software Test Automation	软件测试自动化	Daniel J. Mosley
1.4.4	The Craft of Software Testing: Subsystem Testing Including Object-Based and Object-Oriented Testing	软件测试实用技术	Brian Marick
1.5	JAVA Tools for Extreme Programming: Mastering Open Source Tools, Including Ant,JUnit, and Cactus	极限编程的JAVA工具	Richard Hightower
1.5	Agile Software Development Ecosystems	敏捷软件开发生态系统	Tom DeMarco
1.5	Agile Modeling: Effective Practices For eXtreme Programming and The Unified Process	敏捷建模:极限编程和统一过程的有效实践	Scott W. Ambler
1.5	Model-Driven Development: Automating Component Design, Implementation, and Assembly	模式驱动的软件开发	David Frankel
1.5	A Practical Guide to Feature-Driven Development	特征驱动开发方法:原理与实践	Steve R. Palmer

丛书编号	英文书名	中文书名	作　者
1.1.4	Solid Software	Solid Software	Shari Pfleeger
1.1.4	Peer Reviews in Software: A Practical Guide	同级评审	Karl E. Wiegers
1.1.5	Practical Software Measurement	实用软件度量	John McGarry
1.1.5	Software Metrics: A Rigorous and Practical Approach, 2E	软件度量(原书第2版)	Norman E. Fenton
1.1.5	Software Assessments, Benchmarks, and Best Practices	软件评估、基准测试与最佳实践	Capers Jones
1.2.1	The Object Primer: The Application Developer's Guide to Object Orientation and the UML, 2E	面向对象软件开发教程(原书第2版)	Scott W. Ambler
1.2.1	UML and C++: A Practical Guide to Object-Oriented Development, 2E	C++面向对象开发(原书第2版)	Richard C.Lee
1.2.1	Object-Oriented Methods: Principles & Practices, 3E	面向对象方法:原理与实践(原书第3版)	Ian Graham
1.2.1	Principles of Object-Oriented Software Development, 2E	面向对象软件开发原理(原书第2版)	Anton Eliëns
1.2.1	Object Solutions: Managing the Object-Oriented Project	面向对象项目的解决方案	Grady Booch
1.2.1	An Introduction To Object-Oriented Programming, 3E	面向对象程序设计导引(原书第3版)	Timothy Budd
1.2.1	The Object Advantage: Business Process Reengineering with Object Technology	对象优势:采用对象技术的业务过程再工程	Ivar Jacobson
1.2.1	The Unified Modeling Language User Guide	UML用户指南	Booch/Rumbaugh/Jacobson
1.2.1	The Unified Modeling Language Reference Manual	UML参考手册	Rumbaugh/Jacobson/Booch
1.2.1	Applying UML and patterns: An Introduction to Object-Oriented Analysis and Design, 1E	UML和模式应用(原书第1版)	Craig Larman
1.2.1	Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and the Unified Process , 2E	UML与模式应用(原书第2版)	Craig Larman
1.2.1	Object-Oriented Analysis and Design with Applications, 2E	面向对象分析与设计(原书第2版)	Grady Booch
1.2.2	Software Reuse: Architecture, Process and Organization for Business Success	软件复用: 结构、过程和组成	Ivar Jacobson
1.2.2	Software Reuse Techniques: Adding Reuse to the Systems Development Process	软件复用技术: 在系统开发过程中考虑复用	Carma McClure
1.2.2	Practical Software Reuse: Strategies for Introducing Reuse Concepts in Your Organization	实用软件复用方法	Donald J. Reifer
1.2.2	Large-Scale Component-Based Development	大规模基于构件的软件开发	Alan W. Brown
1.2.2	Component-based Product Line Engineering with UML	基于构件的产品线工程:UML方法	Colin Atkinson
1.4.1	Object-Oriented Analysis & Design	面向对象的分析与设计	Andrew Haigh
1.4.1	Analysis Patterns: Reusable Object Models	分析模式: 可复用的对象模型	Martin Fowler
1.4.1	Requirements Analysis and System Design: Developing Information Systems with UML	需求分析与系统设计	Leeszek A. Maciaszek
1.4.1	Systems Analysis and Design in a Changing World	系统分析与设计	John W. Satzinger
1.4.1	Advanced Use Case Modeling, Vol I: Software Systems	高级用例建模 卷1: 软件系统	Frank Armour
1.4.1	Requirements Engineering: A Good Practice Guide	需求工程	Ian Sommerville
1.4.1	Software Requirements and Estimation	软件需求与估算	Swapna Kishore
1.4.1	Effective Requirements Practices	有效需求实践	Ralph R.Young
1.4.1	Applying Use Cases: A Practical Guide, 2E	用例分析技术(原书第2版)	Geri Schneider
1.4.1	Managing Software Requirements	软件需求管理:统一方法	Dean Leffingwell
1.4.1	Writing Effective Use Cases	编写有效用例	Alistair Cockburn
1.4.1	Problem Frames Analyzing and Structuring Software Development Problems	Problem Frames Analyzing and Structuring Software Development Problems	Michael Jackson

软件工程技术丛书结构图



目　录

第1章 预备知识

1.1 软件、软件工程和软件工程过程

和其他任何工程专业一样，软件工程专业同样关注制品的建造，只不过它关注的是软件这种制品的建造。在这个过程中，人们应该使用尽可能少的资源来建造制品并生产出质量最好的制品，否则就是浪费。为此，我们有必要关注做什么、怎么做、由谁做、在什么条件下做以及使用什么工具做。换言之，过程非常重要，因为过程的质量是产品质量的保证。

软件工程（software engineering）通常包括以下含义：尽可能生产高质量的产品；通常是针对复杂而大型的软件系统；在可持续和可重复的合理基础上使用最小的资源成本。为达到这一目的，软件工程使用了三种主要方法和其他一些原则。这三种主要方法是：

- 提高软件开发过程的管理水平。
- 建立严格的软件开发基础。
- 技术支持。

在讨论产品质量与产生该产品的过程间的关系时，可假定产品的外部质量属性就是过程属性的职能。这意味着软件过程一定有方法学因素和技术层面（Haas 等，1994）。而且不可否认，组织性因素（Humphrey，1989）也蕴含在所有具体的软件过程实例中。因此针对我们的目的而言，可将软件过程定义为：一个实体为了生产一种软件产品在一定环境中使用的技术与方法的组织。

这个定义反映出大多数人对软件过程这一术语的理解，而且区别于将软件过程等同于开发方法学的理解，开发方法学认为仅靠采用软件开发方法就足以保证软件质量的提高（Sauer 和 Lau，1994）。但开发方法学无法解释为什么一种形式化方法适用于一个环境或组织却不适用于另一个环境或组织。

尽管我相信前述的定义是可行的，但是我必须承认，我提到的因素难免仍有遗漏，各种因素之间也可能有重复。

其实，当前大多数对过程的定义（即使未明确指出）也都包含了上述定义中三个因素。已有人断言（Humphrey，1989；Younessi 和 Henderson-Sellers，1997），前面列举的三个主要方法同样支持这个定义，即软件开发过程依赖于如下三个要素：

- 人与组织的影响（提高管理水平的需要）。
- 方法学（严格与形式化的需要）。
- 技术（工具和技术支持的需要）。

因此，我提出了上述定义。在同一篇文章中，Younessi 和 Henderson-Sellers（1997）还断言，这三个要素的平衡就是过程，同时使过程质量达到某个级别。

1.1.1 软件工程的构造性[⊖]方法

构造软件的过程并不是完全一成不变的，事实上也不应该这样认为。可以将构造软件的过程看做集成一组精心挑选的技术来生产软件制品以及生产软件必需的中间产品的过程。有一种软件构造的“Lego”[⊖]方法，允许我们将注意力集中于特定的技术——甚至多种技术——只要该技术适于完成特定的任务。然后，我们可以定义这种技术是什么、如何使用这种技术、产生什么产品、需要什么资源、与其他技术的接口如何，甚至可以定义该技术的有效性以及在何种条件下可以应用。这种方法允许在各种技术之间进行一定程度上的比较分析。本书也采用了该方法。

其实，我们可以更进一步使用此方法，将软件工程过程视为一个活动（activity）的集合，其最终产物就是软件产品。可以将活动视为由任务（task）组成的集合，任务可以看做一个工作单元，在一个任务中我们使用一种或多种技术，以产生一个或多个工作产品。这种构造方法恰恰是许多现代软件过程的精髓，如 Object-Oriented Process Environments and Notation（OPEN; Graham 等，1997），如图 1-1 所示。

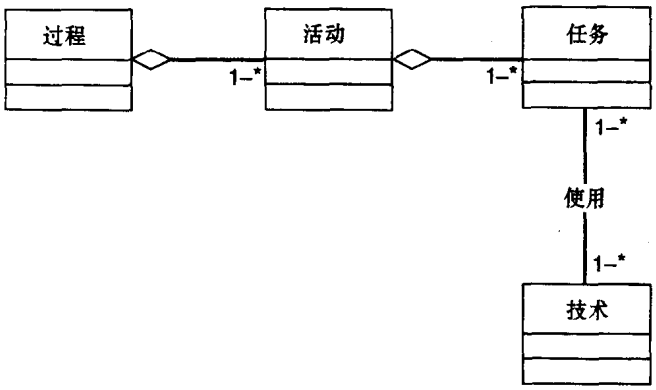


图 1-1 OPEN 的核心关系——一个构造方法的例子

1.1.2 构造软件的过程

在任何构造性的项目中（无论构造棉被、飞机引擎、房屋还是软件），都有一些必不可少的活动，比如：

- 找出构造工作中需要解决的问题。
- 提出一种或几种备选的解决方案。
- 实际构造产品。
- 评估产品的适用性及解决问题的能力。
- 交付产品。

⊖ “构造性”一词是指技术被有目的地组织起来并应用某种规则形成一个过程的方式。

⊖ Lego 是 The LEGO Group A/S 的注册商标。

- 对适用性的持续评估及产品维护。

对软件工程来说，这些必不可少的活动包括：

- 需求的抽取、分析和规约（有时称为需求工程）。
- 设计和设计规约。
- 实现。
- 验证与确认。
- 交付与安装。
- 维护。

目前的软件过程框架（如 OPEN（Graham 等，1997）和 Rational Unified Process（RUP；Jacobson 等，1999））已经意识到这些重要因素，并允许用户设计一个包含若干任务的软件过程实例，以实现这些活动。前面提到，各种任务可归到各个活动中。例如，在某项目中，测试可以是验证与确认活动的一个任务，而变异测试（Budd，1980）可以作为完成此任务的技术。另外，我们可以将审查作为验证与确认活动的另一项任务，Strauss and Ebenau（1995）的技术可用于完成这一审查任务。

在后面的内容中，我们将看到，从这种角度看待软件工程过程有助于更好地理解本书所要说明的问题。这使我们可将每种活动、任务和具体的技术看做是管理缺陷的一种手段，这里所说的缺陷就是可交付的软件产品中的任何缺点。

软件过程的这种定义也是基于范型的。也就是说，用于完成每个软件工程活动的任务和技术都是根据通盘的考虑、哲学的观点或某一个范型而选择的。如，选择一个基于转换的范型（假定问题的语境、问题的解决方案以及解决问题的软件系统都能被建模，建模过程通过对某些输入进行一些转换得到某些输出而完成）将需要选择某种建模技术（如流程图），而如果我们采用因果范型（假定一个系统能在状态转换的基础上进行建模），那么这种建模技术就不合适了。因此，在选择任务和技术时，范型扮演了核心角色，而这些任务和技术也定义了我们缺陷管理的各项工作。本书中，我们的范型是面向对象。

1.2 面向对象

近十年来，出现了许多有关对象范型的著作。显而易见，面向对象业已成为软件工程中的主流范型。面向对象起源于程序构造领域，近来更普及到软件过程中。其已有的技术包括面向对象的分析技术、设计方法、规约、形式化技术、验证与确认、度量等等。现在甚至出现了采用面向对象观点的完整的软件过程定义和框架，如 OPEN（Graham 等，1997）和 RUP（Jacobson 等，1999）。但到底面向对象是什么？它为何如此重要？

理解对象范型

当被问及面向对象的本质时，许多人会提到类、继承、多态行为和复用这些概念。尽管这些概念有趣且重要，但它们其实只是面向对象的副产品。面向对象的本质并不在于此，而是与我们如何理解系统和为系统建模有关，不论是系统的问题、解决方案，还是解决方案的实现。

就面向对象系统的构造而言，创建模型是一个中心问题。简言之，模型（model）是从一个领域到另一个领域的映射（Ross 和 Wright，1992）。这就是说，在一定目标和抽象层次下，我们有意识地将从一个环境得到的信息翻译到另一个环境中。比如，如果我们需要确定房间里有多少把椅子，那么房间内观察到的一把椅子可以映射为一个正整数；如果我们想知道房间内家具的颜色，那么可以将家具映射到一个颜色的构造集（Allen 和 Yen，1979）。

从认知心理学角度来说，模型是这样一种事物：它使用某种通信语言或建模语言（Eysenck 和 Keane，1990），将我们对一个领域（通常是真实世界）的某种情况或事物的理解映射到另一个领域。在此基础上，大多数模型（试图映射通过我们的能量激励感官从外界感知的知识）都至少被描述成一个双重映射。第一重映射是从外部环境（真实世界）映射为我们的认知，即形成了精神上的模型（所谓的感知）；第二重映射利用我们的感知建立另一种模型，以说明利用感知所理解的知识，其目的是将我们的认知传递给其他人（如一个蓝图或作为设计者和实现者之间媒介的对象图）或以备自己将来使用（如，草稿上的注释，或一张在与客户会面时用来让客户提出要求的图；Eysenck 和 Keane，1990；Martin 和 Odell，1995）。有时，第一重映射可以省略（如，当我们要建立 Gorgon 的模型时）。

因此，建模与感知和交流活动密切相关。与此以及大量其他定义（Bruner，1957；Checkland，1981；Gibson，1966；Gregory，1980；Neisser，1976；Preece 等，1995）相一致，建模可以定义为，获取、保留并交流与观察者相关的当前情形的视图。

与前面讨论的各种“现实”视图相关，几个世纪以来（Aristotle，1924；Descartes，1911；Plato，1974，1989），人们一直关注着下面的问题：有关结构、过程和事件发生顺序的知识是理解真实世界的前提，而真实世界又是对其某一方面进行建模的前提。在对所创建模型的未来用途还不确定的时候，这一点尤其重要。下面的例子可以很好地说明这一点。

假设我们用上述方式创建一个模型来回答这类问题：X 是由什么组成的？将此作为该模型将要使用的先验知识，大概足以建立一个关注系统中的事物（对象）及其相互关系的模型。这被称为结构模型（structural model）或对象模型（object model）。如果我们对系统中的 Y 是如何完成的这一问题感兴趣，那么可以设计一个关注输入、处理和输出的模型，这种模型称为过程模型（process model）或更恰当地说是转换模型（transformational model）；如果对做事的时间和顺序更有兴趣，那么就可以设计一个状态、事件和时序的模型，通常称为动态（dynamic）或时序（sequential）模型。

然而，如果缺少使用模型的先验知识，建模者只好提供所有这三种模型及其相互关系。有时我们正是这么做的。我们的认知不仅包含对象，也包含改变和时序。

传统的软件工程建模语言与范型通常仅关注某一方面，而且仅限于产生概括的模型。例如，数据模型主要获取情境中的结构信息而缺少转换和因果细节；流程图获取的是转换的细节而缺少重要的结构和因果细节；而状态图或状态机获取的是时序信息却缺少结构和转换细节。

对象技术结合了这三个方面。其实，这种提供不依赖于任何一种建模视图的模型的能力是面向对象的功能最强大之处。通过从所有三个方面获取并结合情境的相关细节，面向对象把现实“封装”成了模块，就如我们希望的那样。换言之，面向对象范型允许我们将系统看成很多最适

合我们理解的组成部分，就如我们的认知使用相同的抽象对情境进行分解、封装并建模一样。这也说明，面向对象模型与我们对世界（系统）的感知最接近。这就是封装（encapsulation）的概念。封装使我们以适当的角度认识世界上的对象，并能识别所有具有相同特征的对象。这就产生了类型的概念。从某种角度看来具有相同或相似特征的封装物可被看成属于同一类型。一个类型的每一个成员称为对象（object）。

对于一个运转的系统（世界）来说，其内部的对象之间必定存在一定的关系和交流。这在现实世界的任何面向对象模型中都很重要。就是说，作为因果模型的一部分，我们识别并确定了给我们的系统以特别含义的相关交互。即，我们从对象间无限多个可能的交互中抽取出一部分并对其建模，就如同对待对象一样。也就是说，一个对象会对向其发出服务请求的另一个对象做出反应。这就是消息传递模型，同封装一样，这也是面向对象的本质特征。

其实，我们可将面向对象看做是客户－服务器模型在微观（单个对象）层次上的实现。也就是说，提供服务 x 的对象 A 可被需要服务 x 的对象 B 调用。对象 B 是客户，对象 A 是服务器。

作为一种功能强大的范型，面向对象为软件工程从业人员带来机遇的同时也提出了挑战。下面是对机遇的讨论。

（1）综合模型生成

一个定义良好的面向对象模型是多维的。它描述需要建模的情境的结构、转换和因果－时序方面的封装和组合情况。鉴于这种多维性，有人认为，某一情境的面向对象模型要比只根据前面提到的三维之一而开发的模型包含更多信息。因此有理由相信，在相同级别的粒度和精确度下，对于所建模的情境，与实体－联系（ER）图、数据流图（DFD）、流程图和状态机中的任何一种相比，面向对象模型能够回答更多的问题。一般来说，尤其对本书的目标而言，这个特性更具优势。因为如果要尽最大可能对系统的功能建模，遗漏三维之中任何一维都会造成模型所包含信息内容的缺损。后面我们将会看到，这种遗漏可归为一种缺陷而与缺陷管理相关。

然而，这并不是说在综合性藉由面向对象过程得到改善的同时，面向对象产品的质量必定会高于以传统方式开发的产品；而是说，采用面向对象方法可以提供更多的可用信息并进行更方便的信息传递。这一点使项目更清晰且更易观察，有利于产品质量的提高。

（2）无缝性

面向对象的另一个优点是具有无缝性。前面提到，软件系统开发过程必须经历三个相关的活动。观察员或分析师必须开发一个待完善的对问题情境的理解文档，而且该理解在必要时可被记录并传递，因此首先必须创建问题情境的模型。一旦软件开发过程中的专家对问题情境有了足够理解（通常是利用得到的模型），就可以对问题提出解决方案，他们必须将头脑中的可行方案进行建模以将其传递给产品制造人员，这个过程就称为设计模型（design model）。最终，设计被构造为程序代码，也就是第三个模型，这是一个可执行的解决方案的模型。在将面向对象范型应用到软件系统开发的过程中时，我们可以使用相同的建模要素开发所有这三种模型。换言之，我们使用同样的抽象、同样的词汇和同样的语法讲了这三个故事。这种语言讲述对