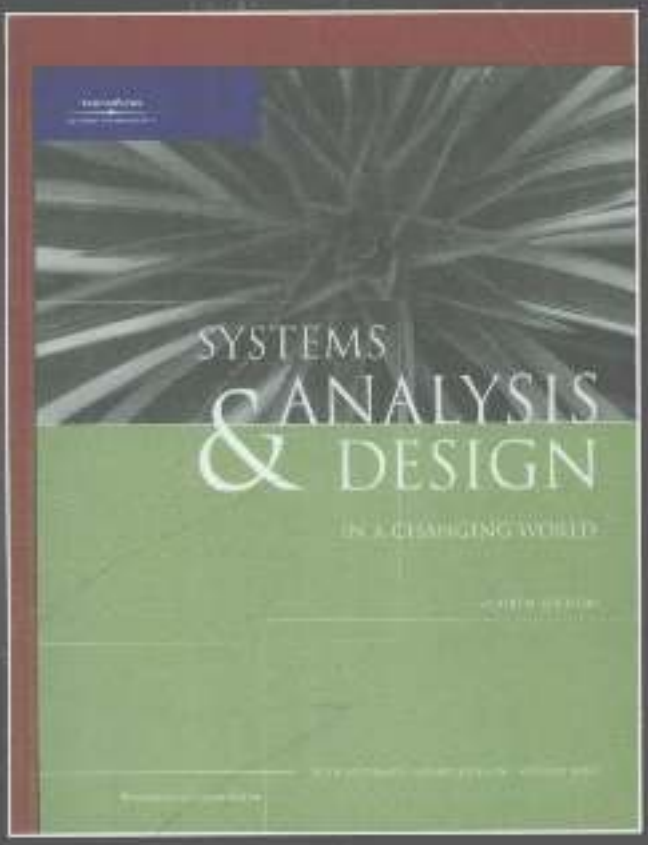
计 算 机 科 学 丛 书

原书第4版

CENGAGE Learning

系统分析与设计

(美) John Satzinger Robert Jackson Stephen Burd 著 耿志强 朱宝 李芳 史晟辉 译



Systems Analysis and Design
In a Changing World
Fourth Edition

机械工业出版社
China Machine Press

CENGAGE Learning

系统分析与设计（原书第4版）

本书继续维持了它一贯的大众化风格并涵盖了现实世界中的两种方法：结构化方法和面向对象方法。新版本采用了流线型的内容表格并且针对各方法使用不同的操作，比以前版本更加容易阅读。通过进行案例学习和增加新的“实践指导”部分，为读者提供了更多的视角，采用最新UML 2.0建模语言，高级面向对象方法和项目管理的最新概念贯穿本书。

本书特色

- 使用案例、实例和插图强调关键概念。
- 每一章节都提供大量亲身实践的机会。
- 包含了丰富的章末练习与测试。

作者简介

John Satzinger 美国密苏里州立大学计算机信息系统学院教授，他同时拥有加州州立理工大学的MBA学位和克莱蒙研究大学的博士学位，并具有15年以上的CIS和MIS大学课程教学和研究经验。他的研究兴趣和专长包括：系统分析与设计、图形用户界面设计、面向对象的开发、数据库和客户-服务器系统开发。

Robert Jackson 美国杨百翰大学计算机科学专业博士，信息系统学院助理教授。他已经发表了大量有关面向对象系统分析与设计、国际软件技术变迁、分布式电子商务以及信息系统教育方面的论文和著作。

Stephen Burd 美国新墨西哥大学的副教授，从1984年开始在此校从事管理信息系统、网络、数据库、硬件/软件课程的教学。分别在美国巴尔的摩大学获得学士和硕士学位、普渡大学获得博士学位。

投稿热线：(010) 88379604
购书热线：(010) 68995259, 68995264
读者信箱：hzjsj@hzbook.com

华章网站 http://www.hzbook.com

网上购书：www.china-pub.com

CENGAGE Learning
www.cengageasia.com

ISBN 978-7-111-25828-5

9 787111 258285

定价：75.00元

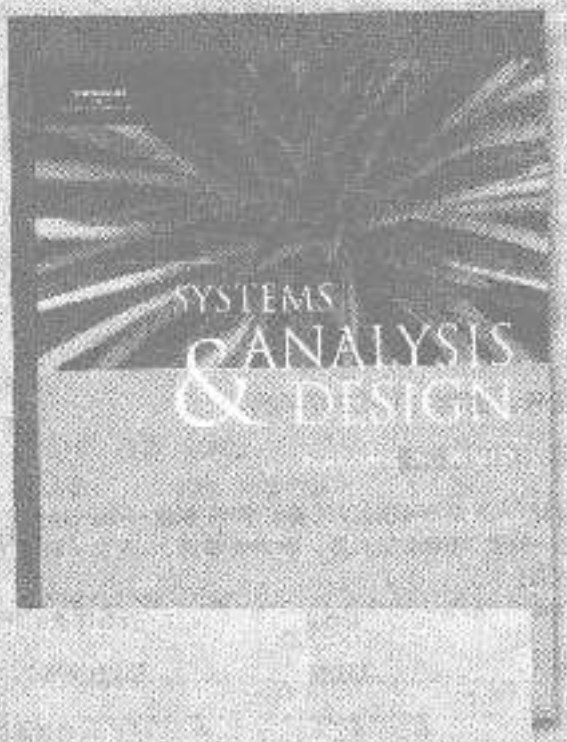
N945/2-4
2009

计 算 机 书

原书第4版

系统分析与设计

(美) John Satzinger Robert Jackson Stephen Burd 著 邱志强 朱宇 李芳 史晓辉 译



Systems Analysis and Design

In a Changing World

Fourth Edition

机械工业出版社
China Machine Press

本书包括现代系统分析员、系统分析任务、系统设计任务、实施与支持四个部分的内容，同时补充了丰富的网络参考资源，既关注概念，又重视方法，也注重实践。全书通过具体、综合的系统项目运作，全过程、完整地介绍了先进的系统分析与设计方法，在承认现实世界中开发环境多变的同时，强调永久价值的基本原则，在强调传统结构化方法和面向对象方法的系统开发两方面的概念、技术、建模的同时，加强了对项目管理的作用和技术方面的介绍，指出在系统开发生命周期内要关注现代结构分析、关注自适应、迭代方法，UP，Scrum，基于Web的开发，极限编程等最新技术，另外，书中每章均提供了大量复习题，思考题，试验练习，案例研究等，以加深对书中理论与技术的理解和应用。

本书内容翔实、结构合理、概念清晰、重点突出、与时俱进，适于计算机，信息，管理及相关专业的本科生、研究生，以及软件工程、系统分析技术人员使用。

John Satzinger, Robert Jackson, Stephen Burd, Systems Analysis and Design: In a Changing World , Fourth Edition (ISBN 978-1-4188-3612-2).

Copyright © 2007 by Cengage Course Technology, a part of Cengage Learning.

Original edition published by Cengage Learning. All rights reserved. 本书原版由圣智学习出版公司出版。版权所有，盗印必究。

China Machine Press is authorized by Cengage Learning to publish and distribute exclusively this simplified Chinese edition. This edition is authorized for sale in the People's Republic of China only (excluding Hong Kong, Macao SAR and Taiwan). Unauthorized export of this edition is a violation of the Copyright Act. No part of this publication may be reproduced or distributed by any means, or stored in a database or retrieval system, without the prior written permission of the publisher.

本书中文简体字翻译版由圣智学习出版公司授权机械工业出版社独家出版发行。此版本仅限在中华人民共和国境内（不包括中国香港、澳门特别行政区及中国台湾）销售。未经授权的本书出口将被视为违反版权法的行为。未经出版者预先书面许可，不得以任何方式复制或发行本书的任何部分，

Cengage Learning Asia Pte. Ltd.

5 Shenton Way, #01-01 UIC Building, Singapore 068808

本书封面贴有Cengage Learning防伪标签，无标签者不得销售。

版权所有，侵权必究。

本书法律顾问　北京市展达律师事务所

本书版权登记号：图字：01-2007-4211

图书在版编目（CIP）数据

系统分析与设计（原书第4版）/（美）萨茨辛格（Satzinger, J.），（美）杰克逊（Jackson, R.），（美）伯德（Burd, S.）著；耿志强等译. —北京：机械工业出版社，2009.4
（计算机科学丛书）
书名原文：Systems Analysis and Design: In a Changing World, Fourth Edition

ISBN 978-7-111-25828-5

Ⅰ.系… Ⅱ.①萨… ②杰… ③伯… ④耿… Ⅲ.①信息系统—系统分析 ②信息系统—系统设计 Ⅳ.G202

中国版本图书馆CIP数据核字（2009）第040417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：杨庆蕾　　陈佳媛
北京慧美印刷有限公司印刷
2009年4月第1版第1次印刷
184mm × 260mm · 36.25印张
标准书号：ISBN 978-7-111-25828-5
定价：75.00元

凡购本书，如有缺页、脱页、缺页，由本社发行部调换
本社购书热线（010）68326294

出版者的话

文艺复兴以降，源远流长的科学精神和逐步形成的学术规范，使西方国家在自然科学的各个领域取得了垄断性的优势，也正是这样的传统，使美国在信息技术发展的六十多年间名家辈出，独领风骚。在商业化的进程中，美国的产业界与教育界越来越紧密地结合，计算机学科中的许多泰山北斗同时身处科研和教学的最前线，由此而产生的经典科学著作，不仅奠定了研究的范畴，还揭示了学术的源变，既遵循学术规范，又自有学者个性，其价值并不会因年月的流逝而减退。

近年，在全球信息化大潮的推动下，我国的计算机产业发展迅猛，对专业人才的需求日益迫切。这对计算机教育界和出版界都既是机遇，也是挑战；而专业教材的建设在教育战略上显得举足轻重，在我国信息技术发展时间较短的现状下，美国等发达国家在其计算机科学发展的几十年间积淀和发展的经典教材仍有许多值得借鉴之处，因此，引进一批国外优秀计算机教材将对我国计算机教育事业的发展起到积极的推动作用，也是与世界接轨、建设真正的世界一流大学的必由之路。

机械工业出版社华章分社较早意识到“出版要为教育服务”，自1998年开始，华章分社就将工作重点放在了遴选、移译国外优秀教材上，经过多年的不懈努力，我们与Pearson，McGraw-Hill，Elsevier，MIT，John Wiley & Sons，Cengage等世界著名出版公司建立了良好的合作关系，从他们现有的数百种教材中甄选出Andrew S. Tanenbaum，Bjarne Stroustrup，Brain W. Kernighan，Dennis Ritchie，Jim Gray，Afred V. Aho，John E. Hopcroft，Jeffrey D. Ullman，Abraham Silberschatz，William Stallings，Donald E. Knuth，John L. Hennessy，Larry L. Peterson等大师名家的一批经典作品，以“计算机科学丛书”为总称出版，供读者学习、研究及珍藏。大开石纹理的封面，也正体现了这套丛书的品位和格调。

“计算机科学丛书”的出版工作得到了国内外学者的鼎力襄助，国内的专家不仅提供了中肯的选题指导，还不辞劳苦地担任了翻译和审校的工作；而原书的作者也相当关注其作品在中国的传播，有的还专程为其书的中译本作序。迄今，“计算机科学丛书”已经出版了近两百个品种，这些书籍在读者中树立了良好的口碑，并被许多高校采用为正式教材和参考书籍，其影印版“经典原版书库”作为姊妹篇也被越来越多实施双语教学的学校所采用。

权威的作者、经典的教材、一流的译者、严格的审校、精细的编辑，这些因素使我们的图书有了质量的保证。随着计算机科学与技术专业学科建设的不断完善和教材改革的逐渐深化，教育界对国外计算机教材的需求和应用都将步入一个新的阶段，我们的目标是尽善尽美，而反馈的意见正是我们达到这一终极目标的重要帮助。华章分社欢迎老师和读者对我们的工作提出建议或给予指正，我们的联系方法如下：

华章网站：www.hzbook.com

电子邮件：hzjsj@hzbook.com

联系电话：(010) 88379604

联系地址：北京市西城区百万庄南街1号

邮政编码：100037

华章教育

译 者 序

信息技术是一个极其活跃，崇尚发展与技术创新的领域，是当今时代决定性的先驱技术。信息化已经成为世界经济和社会发展的大趋势，信息系统的建设是信息化工作中的关键环节，而信息系统开发成功的关键就在于要进行全面的系统分析与设计。然而，现实世界富有挑战性、竞争性和快速多变性。因此，信息系统开发者急需一本顺应时代发展潮流的先进的系统分析与设计的参考书，在考虑开发环境多变的同时，又强调传统方法的基本原则，既要关注概念和方法，又要注重实践。作者通过具体、综合的系统项目运作全过程，完整介绍了系统分析员、系统分析任务、系统设计任务、实施与支持四个部分的内容，另外补充了丰富的网络参考资源。本书英文原著的出版引起了信息领域巨大的反响，得到了许多支持和热情的评论。同时，该书中文译本的出版也极大地吸引了我国广大的读者，不到两个月，已连续两次印刷。原书作者及时总结经验，在实践中不断完善，很快又推出了第2版、第3版、第4版。我们也很荣幸地随之推出其中文译本的第2版、第3版、第4版。

系统分析与设计是一项系统性、实用性十分强的工作。在第4版中，通篇采纳了许多读者反馈的改进意见；考虑读者学习的连续性，完整保留了四部分的体系结构，并根据当前技术发展的需要，加大对面向对象方法、新出现的系统开发技术的介绍力度，同时将本书以前版本的大部分附录内容转移到网上，精简书本中的内容。另外，进一步把落基山运动用品商店（Rocky Mountain Outfitters）的实例与各章中的技术更加紧密地结合起来，承上启下，前后呼应；并在此基础上，通篇增加了“关注Reliable Pharmaceutical Services”案例，进一步加强读者对本书所阐述方法的理解和实践。考虑到当今系统开发的现实性，本书介绍了新技术和新方法，包括自适应、迭代开发方法、软件开发统一过程（UP）、极限编程（XP）、Scrum、基于Web的开发、基于组件的开发等新技术、新方法。

本书第4版的策划更加精心、定位准确、内容翔实、结构合理、概念清晰、重点突出、叙述简明、与时俱进；许多章是模块化的，可根据教学或自学侧重点的不同而选读。每一章都以学习目标、本章要点、实例和概述开始，以本章小结、关键术语、复习题、思考题、实验练习、实例研究和参考资料结束，这样的编写结构与方式有利于教学与自学。这是一本实践性很强的不断完善的现代系统分析与设计专著，可作为本科生、研究生的教材，也可供从事信息系统建设的技术人员、管理人员参考学习。

本书主要由耿志强、朱宝、李芳、史晨辉翻译。同时，徐园、黄明、石晓赞、田清、高川也参与了翻译工作。

由于译者水平有限，书中难免有错误和不妥之处，敬请读者批评指正。

译 者

2008年8月

前　言

作为作者，我们感到非常荣幸，《系统分析与设计》自几年前第一次出版以来，得到了许多支持和热情的评论。最近几年，系统分析与设计领域处于在不断发展与不断完善中。本书将传统的结构化方法和新的面向对象的方法这两个内容放在同等重要的位置上，紧紧跟上了这个领域的发展变化。最近的IS2002模范课程建议，将传统的和面向对象的方法这两个内容放在同等重要的位置上，这也是本书从最初就一直坚持的。在第4版中，我们将继续采用最新的面向对象模型和设计模式，从更深入的层次上更加灵活地覆盖面向对象的分析和设计方法。

目标与前景

本书为本科生和研究生的系统分析与设计课程而编写。系统分析与设计是建立在一系列概念和原理基础上的应用学科，选择并包含了各种快速发展的工具和技术。当今，学习分析和设计的要归功于被有经验的分析员所广泛使用的一些久经考验的技术和刚毕业的学生在工作中非常渴望使用的最新工具和技术。在今天快速发展的环境里，开发信息系统是有难度的，但是如果开发成功，收获则是巨大的。

本书由一个写作小组共同编写而成，在系统分析与设计领域它是一部与众不同的书，内容灵活流畅，兼具深度和广度，我们期望本书的灵活性能足以吸引那些更加强调使用传统方法进行系统分析与设计的教师，以及强调最新的面向对象技术的人们。同时，我们并不想太过简化系统开发的问题。许多新的发展正影响着系统分析与设计。为此，在本书中我们加入了诸如用例、预测和自适应生命周期分析、敏捷开发、UML、Web开发、软件包解决方案、企业资源计划（ERP）、组件等主要发展趋势的内容。

我们也希望通过本书来传授关键的概念和技术，而不是泛泛的描述。因此，我们重点介绍长期有用的基本原理，展示这些基本原理是如何被应用到开发方法中的，并且深入讨论传统结构化和面向对象的分析与设计方法。本书灵活而又创新，全面而又深入，我们相信在读完本书后，你一定会赞同我们的说法。

创新

把关键的系统模型概念整合在一起是本书的一个特色。这些概念既可应用于传统的结构化方法，也可应用于新的面向对象的方法——触发系统用例的事件和作为系统问题域一部分的对象/实体。我们用一章的篇幅讲解确定用例和关键对象/实体的建模。在阅读完这一章之后，我们可以强调结构化的分析与设计或面向对象的分析与设计，或两者兼有。面向对象的方法并不是作为计划外的附加内容加入的，本书从一开始就要求每个人都熟悉面向对象的主要概念。当然，我们并不放弃传统的方法，本书开始就要求每个人都应该熟悉关键的结构化概念。

全面覆盖面向对象（OO）方法

本书所描述的面向对象的方法基于由Grady Booch、James Rumbaugh和Ivar Jacobson所创立的对象管理组织的统一建模语言（UML 2.0）。模型驱动方法从用例和场景开始分析，然后定义用户工作中涉及的问题域类。由用例图、用例描述、活动图和系统顺序图组成需求模型。这里详细讨论了设计模型，特别是顺序图的细节内容、设计类图和包图。而且将设计原理和

设计模式贯穿始终。我们的数据库设计章节包括两种保持对象持久性的方法，即采用关系数据库管理的混合方法和采用面向对象数据库管理（ODBMS）的单一方法。对于本书覆盖的内容和深度，强调面向对象方法的授课者是不会感到失望的。

全面覆盖传统方法

本书描述的传统方法基于由Stephen McMenamin、John Palmer、Ed Yourdon和Meilir Page-Jones改进的现代结构化分析与设计方法。现代结构化分析是一种集成的、模型驱动的方法，它包括事件划分、以实体－联系图（ERDS）为基础的数据建模和以数据流图（DFDS）为基础的过程建模。现代结构化设计也是基于事件划分的，也是使用结构图进行软件设计的。使用关系型数据库管理技术进行数据库设计是一个特色。强调使用结构化方法进行开发的教师将会对本书的描述及本书的深度感到满意。

最新的工具和发展趋势

考虑到当今系统开发的实际需要，本书介绍了一些其他的概念和技术。首先，系统开发和系统开发生命周期（SDLC）被明确定义为反复迭代的过程。虽然本书是按照各阶段的顺序排列组织的，项目的实际开发和项目规划则是迭代的。其次，介绍了采用迭代方法的新技术和方法，包括统一过程（UP）、极限编程（XP）、敏捷建模以及Scrum方法。再次，深入讲解了快速应用开发和基于组件的开发。最后，对客户可选择的软件包解决方案和企业资源计划（ERP）在全书中进行说明。

强调迭代和系统构架

我们仍然很关注传统方法的发展状况。许多研究者以前强调传统方法，但现在也不同程度地关注面向对象方法。然而，不论是传统方法还是面向对象方法，我们都加强了对迭代和三层结构的关注，进而讨论与两种方法相关的SDLC中的预测和自适应方法。

项目管理内容和软件工具

很多本科生培养计划都希望通过系统分析与设计课程来讲授项目管理准则。为了满足这种需要，我们从两个方面来讲授项目管理。第一，本书包括并强调具体的项目管理技术和任务，它与教给学生如何应用具体的项目管理任务来解决系统开发生命周期中的各个阶段和活动相结合，包括迭代开发。第二，在本书网站上的附录中我们给出大量详细的项目管理的概念和原则。这些资料来源于Project Management Body of Knowledge（PMBOK），该组织是由项目管理研究机构发展而来的，是美国项目管理领域中重要的专业组织。

第4版的变化

当我们开始涉及第4版中的更新时，我们重点是改进讲述与传授方式，精简了一些实例，更新了一些在系统分析与设计理论和实践中正在变化的素材。此外，根据我们目前的研究和使用本书授课教师的反馈，我们做了一些重要的改进。

结构化方法和后来出现的面向对象方法现在依然平分秋色。这本书着重介绍了传统的结构化方法，包括数据流图或用例建模、实体－联系图、结构图和关系数据库等，还重点介绍了面向对象方法，包括用例建模、域、设计类图、交互图、包图和状态机图等。可以说，本书深入地介绍和对比了这两种方法。我们扩充了用例内容的介绍，将它作为传统方法及面向

对象方法的需求模型。越来越多使用传统方法和结构的开发团队开始认识到用例和用例描述的好处。我们没有删除关于数据流图的讨论，但是建议有条件的老师可以用用例图代替。

组织结构的改进

我们改变了一些内容的组织结构和顺序，并将一些资料移到了本书相关的网站上。作为在线补充章节和附录。这些变化使得教师在设计他们的课程时更加灵活，同时也使得本书更加容易管理。在这一版中，我们改进了面向对象设计部分第1章的层次和流程，并将修改后的第2章移到了网站上。我们还将关于包和ERP的章节移到了网站上。这样我们就可以在书中提供更多更新的材料，而不用牺牲广度和深度。

软件开发生命周期的预测和自适应方法

另外一个关键的变化是加强了软件开发生命周期的预测和自适应方法的介绍。作为一种定义序列和高迭代生命周期间的连续性方式。项目管理者应该能够定制SDLC来满足特定的项目需求。

加强了面向对象设计内容的介绍

上一版中最显著的改动是大量增加并扩展了面向对象方法的内容。在这一版中，我们进一步精练了原有的讨论和实例，使其在不影响深度的情况下更加易于理解。第11章对实例进行了较大规模的更新。我们还更新了面向对象设计的最新热点一章，但只是把这部分作为独立的在线补充材料放到了本书的网站上。这样，我们在不冲淡传统方法内容的前提下改进，事实上是扩充了面向对象的内容。

加强了实施和支持部分的内容

在这一版中，我们大幅度地更新了实施和支持一章（第15章）。尽管传统的分析与设计课程中综述了实施的过程，但迭代的方法要求更多的低调程序设计人员、实施和集成技术，以及早期迭代中的测试。因此，在整个项目中只考虑分析与设计而不考虑实施和测试是不可能的。

加强了新方法的介绍

我们的书中一直都注重系统分析与设计，系统开发的新概念和新方法的介绍。在这一版中，第2章讨论SDLC的自适应方法时，我们更多地集成了一些具体的方法学。在第16章，我们又更为详细地讨论了这些方法。本书中仍然采用SDLC中的一般的迭代方法，但是教师应该能够保证学生了解许多新的技术和方法学。我们将讨论统一过程方法（UP）、极限编程（XP）、敏捷建模及Scrum开发方法。

组织结构与使用

与第3版类似，本书分为4部分。由于传统的系统设计材料和面向对象的系统设计材料之间的区别越来越大，并且面向对象概念的不断扩大，本版包括16章。另外2章和4个附录放在了本书配套的网站上。根据自己具体情况，教师完全可以跳过某些章节而不失其连续性。有些章节完全是可选的。我们首先对全书做了概述。然后，探讨了在分析与设计课程中采用本教程的不同使用方法。包括对侧重传统结构方法和侧重面向对象方法的教师，以及教授研究生相关课程的教师的建议性的课程大纲。

第一部分：系统分析员

第1章讨论了系统分析员的工作范围，还包括系统的简单讨论和在现代业务组织中解答问题的系统分析员的作用。讨论了RMO的战略信息系统计划，解释了正准备开发的规划项目，即顾客支持系统项目。第2章提出了一个问题：如果我们有一个项目，我们应该怎样来完成这个项目？采用什么方法、工具、技术来开发这个项目呢？介绍了系统开发生命周期（SDLC）的预测和自适应方法，以及基于迭代的不同方法。我们清楚地表明系统开发有很多方法。今天的系统分析员应该熟悉它们。即使学生在课程中或今后的工作中重点应用一种方法，他们也应该能够详细区别结构化的方法、面向对象的方法和其他方法。在详细叙述系统开发生命周期（SDLC）的系统计划阶段，第3章谈到了本课程的核心——系统开发项目。本章介绍了项目规划，可行性评价和项目管理。很快学生对RMO项目有了兴趣，这些材料就会都意义了。

第二部分：系统分析任务

第二部分深入介绍了系统分析技术。第4章详细叙述了SDLC的分析阶段，然后重点介绍了调查系统需求，包括收集信息和调查业主与用户。第5章包含系统需求建模——使用前面已经讲过的包括事件划分和独立对象/实体模型等的方法。第6章继续介绍使用传统方法进行需求建模，包括数据流程图（DFD）、数据流定义、过程描述等。还介绍了一些信息工程模型和技术。第7章继第5章之后继续讨论了采用面向对象的方法进行需求建模。教师可将重点放在第6章或第7章上。以突出课程的重点是传统的还是面向对象的方法，或两者都突出。第8章综述了影响系统解决方案产生的技术环境，然后给出了产生和评价结果的完整的指南，包括软件包解决方案总是可选的这样一个事实。

第三部分：系统设计任务

第9章主要介绍系统设计，详细介绍了SDLC的系统设计阶段，回顾了影响系统设计的科技环境细节，包括网络、客户-服务器架构和三层设计。第10章讨论传统设计方法，包括三层设计的最新想法。第11章和在线补充章节1（参见本书网站）强调了面向对象的设计。第11章为学生讲授如何为每个用例设计交互细节及用例实现。在线补充章节1讨论了更高级的设计状态和设计总结，包括为企业级和基于WEB的系统进行面向对象设计。另外，详细探讨了状态转移和状态机图。为了强调传统的或面向对象的方法，或者二者兼之，教师可以选择将重点放在第10章或第11章。作为第11章的补充，在线补充章节1提供了面向对象设计更为深入的介绍。第12章介绍了数据库设计——关系型、混合型及面向对象的数据库。第13章介绍用户接口和人机交互，包括一般原则，对话设计概念和使用UML图为对话建模。第14章讨论了系统接口，特别注意了系统控制和系统安全。

第四部分：实施与支持

由于现实中有各种各样的开发环境，系统实施越来越多地依赖于技术细节。因此，我们决定对系统实施部分的讨论进行合理的组织以提高其效率。第15章概述了实施、支持传统技术和对象技术。还在第16章全面讨论了快速应用开发和系统开发的新方法，包括统一过程（UP）、极限编程（XP）、敏捷建模和其他方法。同样，尽管全书把包解决方案作为可行的候选来讨论，我们还是在在线补充章节2中对包和企业资源计划（ERP）进行了详细的讨论，包括来自SAP的具体实例。

设计你的分析与设计课程

如前所述，教授分析与设计课程有很多方法，并且各大学之间讲授这门课程的目的也很不同。在一些IS系，分析与设计是一门高级课程，在这里，学生将会应用先前已经学过的知识，如数据库、通信、编程，去进行真正的分析和设计项目。在其他一些IS系，分析与设计被用做系统开发领域的入门课程，并且在更专业的课程之前讲授。有些IS系开设两个课程，在第一个学期强调分析，而在第二个学期强调设计和应用。还有些IS系只开设一门课程同时讲授分析与设计。

分析与设计课程的设计总是很困难的，根据强调传统结构的方法或较新的面向对象的方法进行选择的话，则更加复杂，因此课程设计又要依靠本地课程的优先性了。此外，越来越多的迭代方法在开发中得到应用，这使得顺序排列分析与设计课题更加复杂。例如，采用迭代开发，则两课程序列不能轻易地分割为分析、设计。

基于上述问题，提供一个适用所有这些选项的样本教学大纲是不实际的。目标、课程内容、任务和项目有太多的变化。我们所能提供的是使用本书教授这门课程的一些建议。

传统分析与设计课程

传统分析与设计课程包括系统分析和系统设计的行为，采用结构分析和结构设计、数据库设计、输入/输出/控制设计和对话（界面）设计的任务。通常假设项目采用定制开发，包括Web开发，本课程强调了SDLC、项目管理、信息收集和管理报告。为了学生学习的完整性，一个学期的课程通常仅仅达到完成用户接口的原型设计目标。有时，这门课程也会横跨两个学期，在第二学期里，会继续实施一个实际系统，以求获得更完整的开发经验。

对于这种分析与设计课程的教授方法，一个合理的大纲会省略掉一些章节，如详细叙述面向对象、当前潮流、包（这些概念贯穿全书，因此学生仍然能够熟悉它们）。此外，由于要覆盖的信息量比较大，一些附录如项目管理的细节内容、经济可行性，进度安排和汇报展示等，可以省略掉。

对于强调传统方法的课程，推荐大纲如下：

第1章：系统分析员涉及的领域
第2章：系统开发方法
第3章：项目经理级的分析员
第4章：开始分析：调查系统需求
第5章：系统需求建模
第6章：需求的传统描述方法
第8章：需求、环境与实施的候选方案评估
第9章：进入系统设计
第10章：传统设计方法
第12章：数据库设计
第13章：用户界面设计
第14章：系统界面、控制和安全的设计
第15章：使系统可操作化

面向对象分析与设计课程

除了要特别强调面向对象的模型和技术外，本课程的内容与传统的分析与设计课程类似，

包括面向对象的分析和面向对象的设计、数据库设计、输入/输出/控制设计和对话（界面）设计。通常假设项目采用定制开发，包括Web开发。本课程强调迭代开发方法，包括三层结构、项目管理、信息收集和管理报告。为了学生学习的完整性，一个学期的课程通常仅仅达到完成用户界面的原型设计的目标。有时，这门课程也会跨两个学期，在第二学期里，会继续实施一个实际系统，以求获得更完整的开发经验，通常要强调迭代开发。

对于这种分析与设计课程的教授方法，合理的大纲会省略掉一些细化结构分析和结构设计的章节，描述当前趋势的章节可以被收进来，包括组件和迭代，但是包可能覆盖不到。此外，由于要覆盖的信息量比较大，一些附录如项目管理的细节内容、经济可行性、进度安排和汇报展示等，可以省略掉。

对于强调面向对象开发的课程，推荐大纲如下：

第1章：现代系统分析员涉及的领域
第2章：系统开发方法
第3章：项目经理级的分析员
第4章：开始分析：调查系统需求
第5章：系统需求建模
第7章：需求的面向对象描述方法
第8章：需求、环境与实施的候选方案评估
第9章：进入系统设计
第11章：面向对象设计方法：用例实现
在线补充章节1：面向对象设计的最新热点
第12章：数据库设计
第13章：用户界面设计
第14章：系统界面、控制和安全的设计
第15章：使系统可操作化
第16章：系统开发中的当前趋势

深入研究系统分析和项目管理的传统课程

有些课程深入研究系统分析方法，强调项目管理。这样的课程通常是研究生课程，并且通常认为设计和实施已在更加技术性的课程中包含了。在一些案例中，认为包更倾向于解决方案而不是定制开发，因此定义需求和管理进度相对于设计行为则显得更加重要。

对于这样的课程，附录中的内容，包括项目管理、经济可行性、进度安排和汇报展示等，应该包含进来。具体描述设计的章节可以省略掉。如果合适，包和ERP一章（在线补充章节2）应该包含进来。

对于强调传统方法、深入研究系统分析和项目管理的课程而言，推荐大纲如下：

第1章：现代系统分析员涉及的领域
第2章：系统开发方法
第3章：项目经理级的分析员
在线附录A：项目管理的原则
在线附录B：基于PERT/CPM图的项目进度表
在线附录C：净现值、投资回收期、投资收益率的计算
第4章：开始分析：调查系统需求
在线附录D：向管理者展示汇报结果

第5章：系统需求建模
第6章：需求的传统描述方法
第8章：需求、环境与实施的候选方案评估
第9章：进入系统设计
在线补充章节2：包和企业资源计划

深入研究系统分析和项目管理的面向对象课程

一些课程深入研究面向对象的系统分析（而不是面向对象设计），并强调了项目管理。这样的课程通常是研究生课程，并且通常认为设计和实施在更加技术性的课程中包含了。在一些案例中，认为包更倾向于解决方案而不是定制开发，因此定义需求和管理进度相对于设计行为则显得更加重要。

对于这样的课程，附录中的内容，包括项目管理、经济可行性、进度安排和甘特展示等，应该包含进来。具体描述设计的章节可以省略掉。如果合适，包和ERP一章（在线补充章节2）应该包含进来。

对于强调面向对象方法，深入研究系统分析和项目管理的课程而言，推荐大纲如下：

第1章：现代系统分析员涉及的领域
第2章：系统开发方法
第3章：项目经理级的分析员
在线附录A：项目管理的原则
在线附录B：基于PERT/CPM图的项目进度表
在线附录C：净现值、投资回收期、投资收益率的计算
第4章：开始分析：调查系统需求
在线附录D：向管理者展示汇报结果
第5章：系统需求建模
第7章：需求的面向对象描述方法
第8章：需求、环境与实施的候选方案评估
第9章：进入系统设计
在线补充章节2：包和企业资源计划

比较性的分析与设计课程

一些课程纵览分析与设计领域，全面揭示主要的方法。有时候，这些课程是针对有经验的开发者的研究生课程；有时候，这些课程强调建立在具体实践技术经验上的概念。阅读关键模块的知识可能是目的。然而，通常教师会要求亲手实践项目，在同样的课程中使用传统的和面向对象的几种技术设计同一个系统。

最全面的授课可以包括全书的内容。也可以将关于某些技术具体细节的叙述性内容略去。而一个快速的浏览课程可以覆盖各个章节，从而帮助读者快速的区别和了解模型的知识。如下面的大纲所示，第16章和在线补充章节2可以直接紧跟第8章，然后课程可以继续介绍设计部分。如果该比较性的课程强调系统分析和项目管理，则不必包含设计部分，在线补充第2章后就可以结束了，教师可以根据需要选择。

对于比较性的课程，推荐大纲如下：

第1章：现代系统分析员涉及的领域

第2章：系统开发方法
第3章：项目经理级的分析层
第4章：开始分析：调查系统需求
第5章：系统需求建模
第6章：需求的传统描述方法
第7章：需求的面向对象描述方法
第8章：需求、环境与实施的候选方案评估
第16章：系统开发中的当前趋势
在线补充章节2：包和企业资源计划
第9章：进入系统设计
第10章：传统设计方法
第11章：面向对象设计方法：用例实现
第12章：数据库设计
第13章：用户界面设计
第14章：系统界面、控制和安全的设计
第15章：使系统可操作化

分析与设计课程的迭代方法

分析与设计课程的老师所面临的一个最大挑战是如何处理迭代开发。这对于传统的方法和面向对象的方法而言都是一个难题。课本可以教授分析技术，然后按顺序教授设计技术，但是这与实际情况不符。使这门课程与实际情况相象的一个办法是迭代教学，这就是循序渐进的授课方法。

对于迭代开发方法，课程应该快速浏览分析与设计技术，也许要阅读模型的知识，然后反过来更深入地研究分析与设计部分的资料。在第一次阅读的时候，一些章节的内容可以跳过去。但是，理解并阐述分析与设计模型，并不是真正地创建或开发分析与设计模型。因此，最有意义的方法是首先浏览这些技术并以了解知识为目的，然后，当学生基于课程的项目进行实际创建的时候，可以要求学生重新考虑这些模型。

要求学生第一次就读完所有的内容并全部重读，可能会很困难。因此，授课者可以快速浏览该领域，而不是陷入到具体细节上。接着，第二次阅读就可以增加新的内容，并深入理解以前的内容。例如，第一次阅读的时候就可以强调第5章而跳过第6章或第7章（这依赖于强调传统的方法或者面向对象方法）。可以讲授第9章的设计概述，但是其余的设计部分可以仅限于第10章或第11章（这依赖于强调传统的方法或者面向对象方法）。而第二次阅读的时候再深入理解需求模型和设计部分的深入内容。

还有很多种其他的可能性可以考虑，重要的是，在设计课程的时候，考虑了这个问题。你所提供的任何反馈、看法，或者用迭代方法教授分析与设计课程所做的任何尝试，我们都非常感激。

可以获得的支持

《系统分析与设计（原书第4版）》提供支持教师在课堂上使用的教学工具。本书的附件包括：教师手册、答案、PowerPoint展示和图表文件。需要者请依据书后所附的教学支持服务表与本书原出版社圣智学习出版公司北京代表处联系。

教师手册

教师手册中包括使用本书的一些建议和方法，并提供教学大纲，适用于强调传统结构方法或面向对象方法的老师，也同样适用于教授研究生分析与设计课程的老师。

习题解答

我们给教师提供了复习题的答案、章节练习和案例的参考方案。

PowerPoint演示

本书每一章都包括了被教的PowerPoint展示，教师可以用不同的方式使用这些幻灯片。用做教学辅助的课堂演示或者作为印刷品在课堂上分发。教师也可以加入自己的幻灯片，以补充课程主题。

图表文件

图表文件允许授课者直接使用来自课本中的图表创建自己的演示文件。

致谢

写这本书起因于Course Technology的高级副社长、出版商Kristen Duerr和作者John Satzinger的几次最初的集体讨论。我们认为，一本高质量的分析与设计书需要具有竞争意识的出版商的重大承诺。我们也认为，没有一个人能完成这样一本既灵活流畅，又兼具广度和深度的书籍。因此，Course Technology在组织作者参与工作方面发挥了极强的作用。一开始负责这个项目的责任编辑是Jennifer Locke，她在召集作者并制订本书的方向和最终的形式等方面起到了重要的作用。我们还很幸运地拥有Barrie Tysko负责管理第2版和第3版的项目。他们对第2版和第3版的指导也是具有重要价值的。

十分幸运地，由高级产品经理EuniceYeate-Fogle担任第4版的负责工作，她帮助我们阐明修订版的目标并做了大量改进与补充内容。Eunice还管理着我们其他的Thomson Course Technology书籍——Object-Oriented Analysis and Design with the Unified Process的开发工作，在她的帮助下，我们得以很好地组织本书的创作过程。

我们感谢Cengage Course Technology执行编辑Mac Mendelsohn、Bob Woodbury及高级采编Maureen Martin有预见性的思想和不断的支持。还有许多其他人员参与了本书的制作。Course Technology的Amanda Young对第1版给予了重要的支持。产品经理Summer Hughes与Eunice和Karen一起谨慎、周密地工作，帮助我们整理传统和UML图表中的细微之处。创作小组最大可能地实现了我们预先定义的图解协议和标准。Jennifer Smith为第4版安排教师资料做了大量细致的工作。

我们也要感谢其他一些主要人员所作的特殊贡献。南苏里州西南大学的Richard A. Johnson编写了在线补充章节2关于包和ERP的内容，William Baker提供了展示技术的资料。SMSU、Brigham Young大学、新墨西哥大学和其他机构的许多同事和朋友对我们的工作以不同的方式提供了帮助与支持，也要特别感谢Lavette Tingue、Lorne Olfman和Paul Gray的指导与启发。

最后，我们要衷心感谢为我们努力工作的评论家们，他们对本书的第1版、第2版、第3版和第4版的完成自始至终提出建议。我们非常幸运有这么多知识渊博、观点鲜明的评论家。我

们认真采纳了他们的意见，使得本书更加完善。第1版，第2版，第3版和第4版的评论家有：

Rob Anson, Boise State University
Marsha Baddeley, Niagara College
Teri Barnes, DeVry Institute—Phoenix
Robert Beatty, University of Wisconsin—Milwaukee
Anthony Cameron, Fayetteville Technical Community College
Gerard Catalano, Columbia College
Paul H. Chaney, University of Central Florida
Jung Choi, Wright State University
Jon D. Clark, Colorado State University
Lawrence E. Domine, Milwaukee Area Technical College
Jeff Hefrinton, University of Phoenix
Ellen D. Hoadley, Loyola College in Maryland
Norman Jobes, Conestoga College, Waterloo, Ontario
Robert Keim, Arizona State University
Rebecca Koop, Wright State University
Hsiang-Jui Kung, Georgia Southern University
James E. LaBarre, University of Wisconsin—Eau Claire
Tsun-Yin Law, Seneca College
David Little, High Point University
George M. Marakas, Indiana University
Roger McHaney, Kansas State University
Cindi A. Nadelman, New England College
Bruce Neubauer, Pittsburgh State University
Michael Nicholas, Davenport University—Grand Rapids
Julian-Mark Pettigrew
Mary Prescott, University of South Florida
Robert Saldarini, Bergen Community College
Laurie Schatzberg, University of New Mexico
Terence M. Waterman, Golden Gate University

参与本书创作的所有人员希望读者在这个不断变化的环境里迎接分析与设计的挑战时能有最好的作为。

John Satzinger
Bob Jackson
Steve Burd

目　录

作者的话
译者序
前　言

第一部分　系统分析员

第1章　信息系统分析员涉及的领域 ……1
联合炼油厂的一个系统分析员 ……1
概述 ……2
1.1　解决业务问题的分析员 ……3
1.2　解决业务问题的系统 ……5
1.2.1　信息系统 ……5
1.2.2　信息系统类型 ……6
1.3　系统分析员所需的技能 ……7
1.3.1　技术知识与技能 ……7
1.3.2　业务知识与技能 ……8
1.3.3　人的知识与技能 ……9
1.3.4　诚实与道德 ……9
1.4　分析员周围环境 ……9
1.4.1　面对的各种技术 ……9
1.4.2　应用Web技术提高灵活度 ……10
1.4.3　典型的工作职位与环境 ……10
1.5　战略规划中分析员的作用 ……11
1.5.1　特殊项目 ……11
1.5.2　战略规划 ……12
1.5.3　信息系统战略规划 ……12
1.5.4　企业资源计划 ……13
1.6　落基山运动用品商店及其战略信息系统规划 ……13
1.6.1　落基山运动用品商店概述 ……14
1.6.2　RMO的战略观点 ……15
1.6.3　RMO的组织结构与所在地 ……15
1.6.4　RMO的信息系统部门 ……16
1.6.5　RMO原有的系统 ……17
1.6.6　信息系统战略规划 ……18
1.6.7　客户支持系统 ……19
1.7　系统开发组的分析员（课程核心） ……20
1.7.1　第一部分：系统分析员 ……20
1.7.2　第二部分：系统分析任务 ……21
1.7.3　第三部分：系统设计任务 ……21
1.7.4　第四部分：实施与支持 ……22
1.7.5　网站上的其他材料 ……22
小结 ……22
关键术语 ……23
复习题 ……24
思考题 ……24
实验练习 ……24
实例研究 ……25
参考资料 ……27
第2章　系统开发方法 ……28
Ajax Corporation，Consolidated Concepts和Pinnacle Manufacturing的开发方法 ……28
概述 ……29
2.1　系统开发生命周期 ……29
2.1.1　系统开发生命周期的传统预测方法 ……31
2.1.2　系统开发生命周期的新的自适应方法 ……32
2.2　每个SDLC阶段的活动 ……34
2.2.1　计划阶段 ……34
2.2.2　分析阶段 ……35
2.2.3　设计阶段 ……36
2.2.4　实施阶段 ……37
2.2.5　支持阶段 ……37
2.3　方法、模型、工具和技术 ……38
2.3.1　方法 ……38
2.3.2　模型 ……38
2.3.3　工具 ……39
2.3.4　技术 ……39
2.4　系统开发的两种方法 ……41
2.4.1　传统方法 ……41
2.4.2　面向对象方法 ……46
2.5　系统开发生命周期的变体 ……48
2.5.1　各阶段名称的变体 ……48

2.5.2 以人为重点的变体 ……49
2.5.3 基于开发速度的变体 ……49
2.6 系统开发的当前趋势 ……50
2.6.1 统一过程 ……50
2.6.2 极限编程 ……51
2.6.3 敏捷建模 ……51
2.6.4 SCRUM ……52
2.7 支持系统开发的工具 ……52
2.7.1 CASE工具 ……52
2.7.2 Microsoft Visio ……53
2.7.3 Visible Analyst ……54
2.7.4 Embarcadero Describe ……54
2.7.5 Rational XDE Professional ……54
小结 ……55
关键术语 ……56
复习题 ……57
思考题 ……58
实验练习 ……58
实例研究 ……58
参考资料 ……60
第3章 项目经理级的分析员 ……61
监天共有基金家庭：管理IRA项目 ……61
概述 ……62
3.1 项目管理 ……62
3.1.1 项目成功因素 ……63
3.1.2 项目经理角色 ……64
3.1.3 用SDLC管理项目 ……65
3.1.4 项目管理知识领域 ……67
3.2 项目启动与计划阶段 ……68
3.2.1 启动薄走主运动用品商店的客户支持系统 ……69
3.2.2 项目规划阶段 ……70
3.3 定义问题 ……71
3.4 制订项目进度表 ……74
3.4.1 制订工作分解结构 ……74
3.4.2 制作PERT/CPM图 ……76
3.4.3 为整个SDLC制定进度表 ……79
3.5 确认项目可行性 ……80
3.5.1 风险管理 ……80
3.5.2 经济可行性 ……81
3.5.3 组织上和文化上的可行性 ……85
3.5.4 技术可行性 ……85
3.5.5 进度安排可行性 ……85
3.5.6 资源可行性 ……86
3.5.7 可行性分析 ……86
3.6 为项目组织人员并启动项目 ……86
3.7 RMO项目规划翻新 ……87
小结 ……89
关键术语 ……89
复习题 ……90
思考题 ……90
实验练习 ……91
实例研究 ……92
参考资料 ……93

第二部分 系统分析任务

第4章 开始分析：调查系统需求 ……95
山区摩托运动 ……95
概述 ……96
4.1 更详细的分析阶段 ……97
4.1.1 收集信息 ……97
4.1.2 定义系统需求 ……98
4.1.3 需求的优先级划分 ……98
4.1.4 发现原型及可行性 ……99
4.1.5 产生和评估候选方案 ……99
4.1.6 和管理部门一起复查各种建议 ……99
4.2 业务流程重组和ZACHMAN框架 ……100
4.3 系统需求 ……102
4.4 系统相关者——系统需求的来源 ……103
4.4.1 用户 ……104
4.4.2 客户投资相关者（Stakeholders） ……105
4.4.3 技术人员 ……105
4.4.4 RMO的系统相关者 ……105
4.5 信息收集技术 ……107
4.5.1 主要问题 ……108
4.5.2 复查现有报表、表格和过程描述 ……109
4.5.3 主持与用户的面谈和讨论 ……111
4.5.4 观察并记录业务过程 ……114
4.5.5 建立原型 ……117
4.5.6 分发和收集调查表 ……118
4.5.7 主持联合应用程序设计会议 ……119
4.5.8 研究供应商的解决方案 ……121
4.6 验证系统需求 ……122

4.6.1 What和When ……123
4.6.2 Who ……123
4.6.3 How ……123
小结 ……125
关键术语 ……126
复习题 ……126
思考题 ……127
实验练习 ……128
实例研究 ……128
参考资料 ……130
第5章 系统需求建模 ……131
Waiters on Call餐饮送货系统 ……131
概述 ……132
5.1 模型和建模 ……133
5.1.1 模型的作用 ……133
5.1.2 模型的类型 ……134
5.1.3 用于分析和设计的模型概述 ……136
5.2 事件、活动和用例 ……138
5.2.1 事件分解 ……139
5.2.2 事件的类型 ……140
5.2.3 定义事件 ……141
5.2.4 落基江运动用品商店实例中的事件 ……143
5.2.5 关注每个事件和由此产生的用例 ……144
5.3 问题域的事物 ……145
5.3.1 事物的类型 ……147
5.3.2 开发事物初始列表的过程 ……147
5.3.3 事物间的关系 ……149
5.3.4 事物的属性 ……151
5.3.5 数据实体和实例 ……151
5.4 实体-联系图 ……152
5.4.1 ERD概念的实例 ……152
5.4.2 落基江运动用品商店实例的ERD图 ……155
5.5 类图 ……155
5.5.1 域建模类图符号 ……156
5.5.2 有关对象类的更复杂的问题 ……158
5.5.3 设计类型符号 ……159
5.5.4 落基山运动用品商店实例的域模型类图 ……162
5.6 目标 ……164
小结 ……164
关键术语 ……165
复习题 ……166
思考题 ……167
实验练习 ……169
实例研究 ……169
参考资料 ……172
第6章 需求的传统描述方法 ……173
圣地亚哥月刊：数据流分析系统 ……173
概述 ……174
6.1 用传统的观点和面向对象的观点看待活动/用例 ……175
6.2 数据流图 ……176
6.2.1 数据流图和抽象水平 ……177
6.2.2 RMO数据流图 ……180
6.2.3 物理DFD和逻辑DFD ……184
6.2.4 开始DFD质量 ……186
6.3 详细记录DFD部件 ……189
6.3.1 处理描述 ……189
6.3.2 数据流定义 ……193
6.3.3 数据存储定义 ……195
6.3.4 数据元素定义 ……195
6.3.5 DFD总结 ……196
6.4 信息工程模型 ……196
6.4.1 IE系统开发生命周期 ……196
6.4.2 IE和结构化开发的比较 ……197
6.4.3 处理分解和依赖模型 ……198
6.5 结点和网络通信 ……200
小结 ……202
关键术语 ……202
复习题 ……203
思考题 ……204
实验练习 ……204
实例研究 ……205
参考资料 ……207
第7章 需求的面向对象描述方法 ……208
无限电子公司：供应链一体化 ……208
概述 ……209
7.1 面向对象的需求 ……209
7.2 系统活动——用例场景视图 ……211
7.2.1 用例和参与者 ……211
7.2.2 用例图 ……212
7.2.3 开发用例图 ……216
7.2.4 用例详细描述 ……217
7.3 确定输入和输出——系统顺序图 ……223

7.3.1 系统顺序图符号……221
7.3.2 开发系统顺序图……226
7.4 确定对象行为——状态图……230
7.4.1 复合状态和并发性……232
7.4.2 开发状态图的规则……233
7.4.3 开发RMO状态图……234
7.5 面向对象模型的集成……237
小结……238
关键术语……239
复习题……239
思考题……240
实验练习……244
实例研究……244
参考资料……247
第8章 需求，系统与实施的候选方案评估……248
热带鱼销售公司：链接到正确的系统……248
概述……249
8.1 项目管理的前景……250
8.2 决定范围和自动化水平……251
8.2.1 控制项目范围……251
8.2.2 定义自动化水平……251
8.2.3 候选方案的实施……254
8.2.4 RMO候选方案的评估……254
8.3 定义应用程序配置环境……255
8.3.1 硬件、系统软件和网络……255
8.3.2 开发工具……257
8.3.3 RMO的环境……258
8.4 候选实施方案的选择……261
8.4.1 设备管理……262
8.4.2 软件包、成套软件和ERP系统……262
8.4.3 定制软件系统……263
8.4.4 选择实施方案……264
8.5 与供应商签订合同……267
8.5.1 生成RFP……267
8.5.2 提出评价和选择供应商……269
8.5.3 签订合同……269
8.6 提交结果并做出决策……270
小结……270
关键术语……271
复习题……271
思考题……271
实验练习……272
实例研究……272
参考资料……274

第三部分 系统设计任务

第9章 进入系统设计……275
FAIRCHILD PHARMACEUTICALS：一个生产系统的最终结构设计方案……275
概述……276
9.1 理解设计要素……277
9.1.1 设计的主要组件和层次……277
9.1.2 从分析到设计……278
9.2 设计阶段的活动……280
9.2.1 网络的设计与集成……281
9.2.2 设计应用程序的结构……281
9.2.3 设计用户界面……282
9.2.4 设计系统接口……282
9.2.5 数据库的设计与集成……283
9.2.6 设计细节的原型……283
9.2.7 系统控制的设计与集成……283
9.3 项目管理——协调项目……283
9.3.1 协调项目组……284
9.3.2 RMO的项目组……284
9.3.3 协调信息……285
9.4 配置环境……286
9.4.1 单机结构与多层结构……286
9.4.2 集中式结构与分布式结构……287
9.4.3 计算机网络……287
9.4.4 Internet, Intranet和Extranet……288
9.5 应用程序结构……289
9.5.1 客户—服务器结构……289
9.5.2 二层客户—服务器结构……290
9.5.3 Web服务结构……292
9.5.4 中间件……292
9.5.5 Internet和基于Web的应用程序结构……292
9.6 网络设计……294
9.6.1 网络集成……294
9.6.2 网络描述……294
9.6.3 通信协议和中间件……295
9.6.4 网络容量……296
小结……296

关键术语 ……297
复习题 ……298
思考题 ……298
实验练习 ……298
实例研究 ……299
参考资料 ……299
第10章 传统设计方法 ……300
剧院系统有限公司：新事物，旧事物 ……300
概述 ……301
10.1 采用结构化方法进行应用程序结构的设计 ……301
10.2 自动化系统边界 ……302
10.3 系统流程图 ……303
10.4 结构图 ……306
10.4.1 开发结构图 ……308
10.4.2 评价结构图的质量 ……313
10.5 模块算法设计：伪码 ……314
10.6 结构化应用程序设计与其他设计任务的集成 ……315
10.7 三层设计 ……317
小结 ……320
关键术语 ……320
复习题 ……320
思考题 ……321
实验练习 ……323
实例研究 ……323
参考资料 ……324
第11章 面向对象设计方法：用例实现 ……325
NEW CAPITAL BANK ……325
概述 ……326
11.1 面向对象设计——程序分析和设计的需求 ……326
11.1.1 面向对象程序设计概述 ……327
11.1.2 面向对象设计模型 ……327
11.1.3 面向对象设计过程 ……331
11.2 设计类和设计类图 ……332
11.2.1 设计类符号 ……332
11.2.2 设计类表示 ……333
11.2.3 开发初步设计类图 ……335
11.2.4 设计模式和用例控制器 ……337
11.2.5 一些基本的设计准则 ……338
11.3 实现用例行定义方法——顺序图设计 ……340
11.3.1 对象职责 ……341
11.3.2 “查询可用项目”用例的初步顺序图 ……341
11.3.3 顺序图初步设计的指南和假设 ……344
11.3.4 “维护产品信息”用例的初步顺序图 ……344
11.4 多层设计 ……347
11.4.1 设计数据访问层 ……347
11.4.2 “查询可用条目”用例的数据访问层 ……349
11.4.3 “维护产品信息”用例的数据访问层 ……350
11.4.4 设计可视层 ……351
11.5 用协作图设计 ……352
11.6 更新设计类图 ……355
11.7 包图——将主要部分结构化 ……356
11.8 三层设计的实现问题 ……359
小结 ……360
关键术语 ……360
复习题 ……361
思考题 ……362
实验练习 ……366
实例研究 ……366
参考资料 ……367
第12章 数据库设计 ……368
金冈图书公司：设计一个新的数据库 ……368
概述 ……369
12.1 数据库与数据库管理系统 ……369
12.1.1 DBMS的组件 ……370
12.1.2 数据库模型 ……370
12.2 关系数据库 ……371
12.2.1 设计关系数据库 ……372
12.2.2 实体的表示 ……373
12.2.3 关系的表示 ……374
12.2.4 加强参照完整性 ……375
12.2.5 模式质量评估 ……375
12.3 面向对象数据库 ……381
12.3.1 设计对象数据库 ……381
12.3.2 类的表示 ……381
12.3.3 关系表示 ……383
12.4 混合对象-关系数据库设计 ……387

12.4.1 类和属性 ……388
12.4.2 关系 ……389
12.4.3 数据访问类 ……390
12.5 数据类型 ……391
12.5.1 关系DBMS的数据类型 ……392
12.5.2 对象DBMS的数据类型 ……392
12.6 分布式数据库 ……393
12.6.1 分布式数据库体系结构 ……393
12.6.2 RMO分布式数据库体系结构 ……396
小结 ……398
关键术语 ……398
复习题 ……399
思考题 ……400
实验练习 ……401
实例研究 ……401
参考资料 ……403
第13章 用户界面的设计 ……404
Aviation Electronic的界面设计 ……404
概述 ……405
13.1 输入和输出的识别与分类 ……406
13.1.1 传统和面向对象的输入和输出 ……406
13.1.2 用户界面与系统界面 ……406
13.2 理解用户界面 ……407
13.2.1 用户界面的物理特征 ……408
13.2.2 用户界面的感知特征 ……408
13.2.3 用户界面的概念特征 ……408
13.2.4 以用户为中心的设计技术 ……408
13.2.5 人—机界面研究领域 ……409
13.2.6 有关HCI的隐喻 ……411
13.3 界面设计指导原则 ……415
13.3.1 可视性和可供性 ……415
13.3.2 八条黄金规则 ……416
13.4 对话设计文档编制 ……419
13.4.1 用例、子系统和菜单层次 ……419
13.4.2 对话与故事脚本 ……420
13.4.3 用UML表达对话文档编制 ……422
13.5 设计标准窗体和浏览器窗体的指导原则 ……425
13.5.1 窗体布局与格式化 ……425
13.5.2 数据的键控与输入 ……427
13.5.3 导航与支持控件 ……427
13.5.4 帮助支持 ……428
13.6 网站设计指导原则 ……428
13.6.1 网页设计中的10种好的做法 ……428
13.6.2 网站设计原则 ……429
13.7 RMO对话设计 ……429
13.7.1 电话订购业务员的对话设计 ……430
13.7.2 RMO网站对话设计 ……433
小结 ……435
关键术语 ……435
复习题 ……436
思考题 ……437
实验练习 ……437
实例研究 ……438
参考资料 ……440
第14章 系统界面、控制和安全的设计 ……441
Downslope滑雪用品公司：设计一个安全供应商系统界面 ……441
概述 ……442
14.1 确定系统界面 ……442
14.2 系统输入设计 ……445
14.2.1 输入设备和机制 ……446
14.2.2 定义系统输入细节 ……447
14.3 系统输出设计 ……453
14.3.1 定义系统输出的细节 ……453
14.3.2 设计报表、声明和返回文档 ……455
14.3.3 报表的规范化 ……461
14.4 完整性控制设计 ……461
14.4.1 输入完整性控制 ……463
14.4.2 数据库完整性控制 ……463
14.4.3 输出完整性控制 ……465
14.4.4 预防非法的完整性控制 ……466
14.5 安全性控制设计 ……467
14.5.1 系统访问安全 ……468
14.5.2 数据安全 ……471
14.5.3 数字签名和数字证书 ……473
14.5.4 安全交易 ……474
小结 ……474
关键术语 ……475
复习题 ……476
思考题 ……477
实验练习 ……478
实例研究 ……478
参考资料 ……480

第四部分　实现与支持

第15章　交付系统可操作化 ……481
TRI-STATE HEATING OIL公司：系统开始运行时的优先次序调整 ……482
概述 ……482
15.1　程序开发 ……483
15.1.1　系统实施的程序 ……484
15.1.2　框架开发 ……489
15.1.3　基于小组的程序开发 ……489
15.1.4　源代码的控制 ……490
15.1.5　版本 ……491
15.2　质量保证 ……493
15.2.1　技术复审 ……494
15.2.2　测试 ……495
15.3　数据转换 ……501
15.3.1　重用现有数据库 ……501
15.3.2　重新装载数据库内容 ……501
15.3.3　创建新数据库 ……502
15.4　安装 ……503
15.4.1　直接安装 ……504
15.4.2　并行安装 ……504
15.4.3　阶段安装 ……505
15.4.4　人的问题 ……506
15.5　文档 ……507
15.5.1　系统文档 ……508
15.5.2　用户文档 ……509
15.6　培训与用户支持 ……510
15.7　维护和系统增强 ……512
15.7.1　提交改动申请和出错报告 ……513
15.7.2　实施改动 ……514
15.7.3　计算从耐结构的开发 ……515
小结 ……515
关键术语 ……516
复习题 ……517
思考题 ……517
实验练习 ……518
实例研究 ……518
参考资料 ……519
第16章　系统开发中的当前趋势 ……521
VALLEY REGIONAL 医院：管理一个项目的进展 ……521
概述 ……522
16.1　软件原则和实践 ……522
16.1.1　抽象 ……523
16.1.2　模型和建模 ……524
16.1.3　模式 ……524
16.1.4　重用 ……524
16.1.5　方法和过程 ……524
16.2　自适应开发方法 ……525
16.2.1　统一过程开发 ……526
16.2.2　敏捷型开发观点和敏捷宣言 ……530
16.2.3　极限编程 ……533
16.2.4　Scrum ……536
16.2.5　项目管理和自适应方法 ……538
16.3　模型驱动的体系结构——通用解决方案 ……540
16.4　框架、组件和服务 ……542
16.4.1　对象框架 ……543
16.4.2　组件 ……544
16.4.3　组件标准和基础结构 ……545
16.4.4　组件和开发生命周期 ……546
16.4.5　服务 ……548
小结 ……549
关键术语 ……550
复习题 ……550
思考题 ……551
实验练习 ……552
实例研究 ……552
参考资料 ……554
在线资源[1]
在线补充章节1 面向对象设计的热点问题
在线补充章节2 包和企业资源计划
在线附录A　项目管理原则
在线附录B　基于PERT/CPM图的项目安排
在线附录C　计算净现值、偿还期和投资回收期
在线附录D　向管理者汇报展示结果

[1] 所有在线资源请到华章网站（http://www.hzbook.com）下载。

第一部分　系统分析员

第1章　信息系统分析员涉及的领域

学习目标

阅读本章后，应具备如下能力：

- 解释系统分析员在业务中的关键作用
- 描述分析员可以从事的各种类型的系统
- 解释技术、人员和业务技能对一个分析员的重要性
- 解释为什么道德行为对一个系统分析员的职业是至关重要的
- 描述分析员需要熟悉的多种技术
- 描述能从事分析与设计工作的各种职称和职业
- 讨论分析员在为一个组织进行战略规划时所起的作用
- 描述在系统开发项目中分析员的作用

本章要点

- 解决业务问题的分析员
- 解决业务问题的系统
- 系统分析员所需的技能
- 分析员周围环境
- 战略规划中分析员的作用
- 落基山运动用品商店及其战略信息系统规划
- 系统开发级的分析员（课程核心）

联合炼油厂的一个系统分析员

当计算机信息系统（CIS）专业的学生就有关职业生涯事宜询问Mary Wright时，她回忆了两年来作为一个程序分析员的工作经历。她说，“那一刻仿佛就发生在昨天，我终于从大学毕业，踌躇满志地来到联合炼油厂开始我新的工作。”

联合炼油厂是得克萨斯州西部的一个独立石油炼制公司。他们从自由石油生产商那里购买原油，炼制成汽油和其他石化产品，然后卖给独立批发商。石油炼制产品的需求增长得非常迅速，联合炼油厂也正在以最大能力进行生产。所以，对于联合炼油厂而言，产量计划系统和炼制系统是特别重要的计算机信息系统，而且周密的计划和过程监控能使企业降低成本，增大产量。这种需求增加和能源工业的竞争格局的变化使得信息系统对联合公司显得格外重要。

Mary与来访的学生继续交谈。她说，“最初我做规划，主要确定最终用户要做的事情。为了丰富我的经验，我完成了一些Java和面向对象分析方面的培训。这个工作要比我最初期望的好得多，以至于我对整个IPCS项目的每一件事情都着迷。”

集成过程控制系统（IPCS）项目是年前起草的公司信息系统规划的一部分。联合炼油厂

的首席执行官Edward King一开始就极力争取制订更多、更大的全公司战略规划，包括信息系统五年战略规划。IPCS开发项目安排在这个规划的第三或第四年度的开始，但优先次序突然改变了。石油产品的需求量从未达到这样大，原油供应逐步变得严重不足，同时，政治压力促使价格暴涨。

必须增加产量并降低生产成本。建立另外的炼油厂需要几年的时间，从新的油田获得额外的原油供应也是几年后的事情了。为了公司的发展和增加利润，联合公司的唯一选择是利用现有的工厂和供给。因此，高层经理打算全面实施IPCS项目，从根本上提高产量计划和过程监控。联合公司的每个人在任何时间任何地方都可以访问这些信息。

Mary继续说："看起来IPCS项目是公司关注的唯一事情。我作为初级分析员被分派到这个项目帮助项目经理，这样我就可以接触到每一件工作。我突然陷入到一个又一个的会议中，我好像变成了一个石油工程师，必须消化各种各样的有关炼油和销售业的信息。又好像在进修商学院的课程一样，我与生产管理人员、供应商和市场经理接触，了解石油商务。我去多个地方参观油田和输油管系统（包括阿拉斯加北坡的四日旅行，注意！实际用了两天）。我采访了卖方代表和在能力规划与过程控制系统方面有专长的顾问。一直以来，我花费了大量的时间在计算机上写报告、信件和备忘录，而不是程序设计！"

"至今为止，我们已经为这个项目工作了整整7个月，每当我犹豫的时候，我们的首席执行官King先生总会说些什么来强调IPCS项目对公司的未来是如何重要。他反复对雇员和股东发着相同的话，King先生多次出席过我们的会议，在我向高级管理层介绍系统的关键需求的那天，他甚至就坐在我的旁边。"

"这真出乎我的意料。"

概述

Mary讲述的有关联合炼油厂的项目说明具有战略价值的信息系统对业务组织和其最高执行者是至关重要的。有的系统分析员甚至是像Mary一样刚毕业的大学生，所完成的大部分活动与任务都涉及远远多于程序设计的大量内容。系统分析更多的是理解业务及其目标与战略、定义支持那些目标与战略的信息系统的需求并支持业务，这不像大多数大学生想象的那样。

信息技术对生产力和利润发生了重大影响，因此信息系统成为当今吸引人们的一个热门行业。你们当中的大部分人常常使用最新的技术去进行在线购物、预约房间、在线拍卖、客户支持、电子邮件和无线信息。但是，并不是这个技术本身增加了生产力和利润，而是开发信息系统的专业人员利用技术的力量使解决方案被实现。开发信息系统所面临的挑战是巨大的，因为越来越多的人希望拥有这种可以随时随地访问数据的信息系统。

系统开发成功的关键是要进行全面的系统分析与设计，以便理解业务组织需要从信息系统中获取什么。系统分析是指理解和详细地说明信息系统应该做什么，系统设计是指详细地说明许多信息系统的组件物理上是怎样实施的。本书是一本供系统分析员、开发信息系统的业务专业人员使用的关于系统分析与设计技术的专业用书。

系统分析：理解并详细说明信息系统应该是做什么的过程。

系统设计：详细说明信息系统的许多组件在物理上是怎样实施的过程。

系统分析员：使用信息技术的业务专业人员，利用分析与设计技术解决业务问题。

本章描述系统分析员的领域——工作性质，重要的知识与技能及分析员从事的系统与特殊项目的类型。首先，把分析员的工作描述为解决一个组织的问题。这样随着分析员描述而了解解决问题的过程。第二，由于分析员所面对的大多数问题部分要由信息系统解决，因此，审查各种商用信息系统的类型是重要的。因为系统分析员是业务专业人员，需要具备广博的

技术性、业务性和人事的知识与技能。所以这些都要审查。第三，调查用于信息系统的各种技术与分析员工作的各种工作场所与职位。有时，分析员要从事一些特殊项目，如战略规划、业务流程重组、企业资源计划等。事实上，分析员的工作并不是大多数计算机信息系统专业的学生所想像的那样的。

最后，本章介绍总部设在犹他州帕克城的区域性运动服装经销商落基山运动用品商店（RMO）。RMO正在做一个战略信息系统计划，接下来的几年里需要开发一系列信息系统并实施项目。RMO要启动的这个项目需要一个新的客户支持系统，它将集成电话订单、邮件订单和通过Internet的直接客户订单。全书将使用落基山运动用品商店实例说明分析与设计技术。

1.1 解决业务问题的分析员

系统分析与设计最重要的就是实践，要经得起时间的考验和知识技术的发展。分析员毫无疑问地必须熟悉计算机和计算机程序，必须具有专业技能和程序设计方面的开发专长。他们也必须有足够的兴趣去探索事情是怎样做的，以及如何把这些事情做得更好。

并发信息系统并不仅仅是编写程序。正如一些公开的成功实例一样，开发信息系统要为组织解决问题，系统分析员经常被看做问题的解决者，而不是程序员。那么，分析员要解决什么样的典型问题呢？

- 客户要昼夜随时定购产品。那么，在不增加销售成本的前提下如何一刻不停地处理这些订单？
- 生产计划需要计算机确定每周生产的每种产品的数量。那么，如何估计影响生产的众多参数，然后让计划员在提交一个特定计划之前可以研究不同的方案呢？
- 供应商希望以较小的日批处理方式发载用于制造过程中的部分物资，使库存费用最小化，以降低成本。那么，如何定购最小的份数，考虑每日的运输，并充分利用供应商的折扣？
- 通过跟踪购买模式和客户购买倾向，市场部门需要较好地预见客户需求。那么，如何收集和分析市场部门可用的一些关于客户行为的信息？
- 管理人员要不断地了解公司目前的财务状况，包括账目盈亏、现金流转和股票市场预测。那么，如何收集、分析和提交管理人员所需要的全部金融信息？
- 对工要求他们的福利计划更加灵活。管理人员要建立城信。那么，如何处理灵活的健康计划、职工投资选择权、退休账目和其他针对职工的福利计划？

信息系统开发者要处理类似于以上的问题及更多的问题。其中，有些是“大问题”，并且具有战略意义，而有些则是“小问题”，只影响极少数人，但对这些人而言却是重要的。对于解决业务问题的信息系统，最终所做的所有程序设计都是重要的。但这不仅仅是程序设计问题，程序设计在开发过程的后期才开始。

分析员如何解决这些问题呢？系统分析与设计重点在于理解业务问题并提出解决方案。用信息技术解决业务问题的大致过程，如图1-1所示。很显然，其中的解决方案又是新的信息系统，但这仅仅是工作的一



图1-1 分析员解决问题的大致过程

部分。

分析员必须首先全面理解问题，并了解针对这个问题可能发生的每一件事情——什么人参与？什么业务过程开始起作用？当解决这个问题时会影响哪些其他的系统？然后，分析员要使管理人员确信解决这个问题所带来的利益会超过所花费的代价。有时解决这个问题要花费大量金钱，如果这样就可能不值得去做了。

如果解决这个问题是切实可行的，那么分析员就要详细说明解决这个问题的需求——必须满足什么样的指定目标，需要存储和使用什么样的数据，对数据要做什么样的处理，怎么输出。必须首先明确需要做什么，其次才是怎样去做。

一旦确定了详细的需求，分析员就要设计几套可行的解决方案。每个解决方案需要十分仔细地全盘考虑。通常确定一个信息系统就是对构成信息系统的物理部件的一系列选择，也就是解决“怎样去做”的问题。必须确定如下问题：

- 必需的部件是什么？
- 建立不同的部件应该使用什么技术？
- 这些部件放在何处？
- 部件如何通过网络通信？
- 部件如何构成一个系统？
- 人们如何与系统交互？
- 哪些部件是定制的，哪些是购买的？
- 应该由谁来开发定制的部件？
- 谁来集成和支持这些部件？

总有许多不同的方案必须考虑。挑选最好的方案是亟待解决的问题——这个解决方案应是风险最小、效益最好的。分析员需要考虑解决这个问题的方案是否有成本效益，但也要考虑这些方案与公司战略规划的一致性。这个解决方案有利于组织的基本目标的实现吗？与其他计划的系统能无缝集成吗？使用了适合管理人员确定的战略方向的技术吗？最终用户会接受吗？分析员必须考虑许多因素，并综合，果断地做出决策。

一旦系统分析员与管理人员协调决定推荐某个方案并经管理人员同意，就必须写出详细资料。这里分析员要关心的是如何为即将工作的新系统创作一个“蓝图”（设计说明书）。系统设计说明书包含数据库、用户界面、网络、操作步骤、转换计划和程序模块。设计说明书一经完成，就可以开始实际的系统建设了，包括程序设计和测试。

建立并安装一个信息系统要花费大量的金钱，可能是几百万美元。因此，必须制订详细的计划。要建立并运行一个系统，通常需要大量的程序设计人员参与编程，这些编程人员需要确切地了解系统要完成什么。因而，需要详细的说明书。本书介绍了分析员在系统开发期间创建详细的说明书所使用的各种工具和技术。这些说明书中有些是系统分析的结果，有些是系统设计的结果。

尽管本书是为即将成为系统分析员的读者而编写的，但也为其他将涉及业务问题（可以利用信息系统解决的）的读者提供良好的基础知识。业务经理们必须学习如何利用信息技术解决业务问题的知识。许多普通商学院的学生在两年和四年的学位培养计划中选修系统分析与设计课程，以丰富他们的背景。如工商管理硕士（MBA）和会计学硕士（M'Acc）的培养计划中就有包括使用本书的课程的技术痕迹。记住，系统分析与设计不仅仅针对开发系统，而是实实在在地利用信息技术解决业务问题。所以，尽管经理们并不需建立一个信息系统，但他们需要增长这方面的专业知识，以使工作更有效。

1.2 解决业务问题的系统

我们描述了解决业务问题的系统分析员，提到问题的解决方案通常就是一个信息系统。在讲述如何学习成为一个系统分析员之前，让我们快速地复习一下信息系统的一些概念。

1.2.1 信息系统

系统是一组为实现某些结果相互联系、相互作用的部件的集合体。信息系统是一组完成收集、处理、存储和以输出完成业务任务所需信息作为提交的相互联系、相互作用的部件的集合体。完成一个业务任务通常就是我们前面所说的"问题"。

系统：一组为实现某些结果相互联系、相互作用的部件的集合体。

信息系统：一组完成收集、处理、存储和以输出完成业务任务所需信息作为提交的相互联系、相互作用的部件的集合体。

例如，一个工资单系统要对职工本身及工作信息进行收集处理并存储，然后为组织生成工资和工资报表（在其他事情中）。一个销售管理系统收集有关客户、销售、产品和库存等信息，然后处理并存储这些信息，再提交给制造商以便安排生产。

什么是一个信息系统的相互联系并作用的部件呢？有几种方法可以说明这个问题。任何系统都可以有许多子系统。一个子系统是一个系统中的一部分。因此，子系统可以是理解部件的一种方法。例如，一个客户支持系统可以有一个订单登录子系统，这个子系统可以为客户生成新的订单，另一个子系统可以处理完成订单，包括发货和退还订单，还有一个子系统可以维护产品目录数据库。把一个系统作为一组子系统来考虑，对分析员是十分有用的，这些子系统就是相互联系、相互作用的部件。

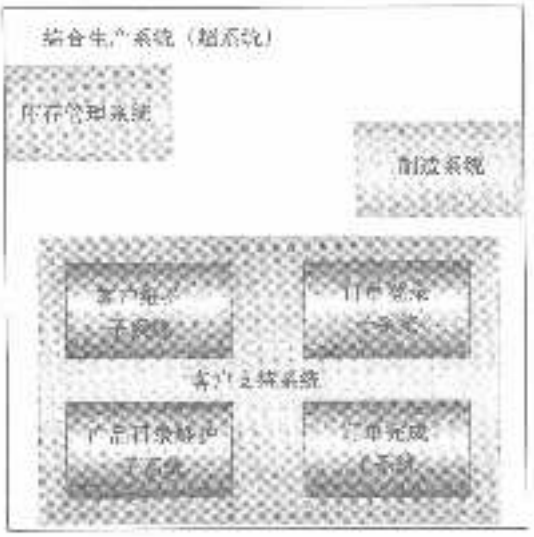


图1-2 信息系统和子系统

子系统：一个大系统中的部分系统。

每个系统又是一个大系统的一部分，这个大系统称为超系统。所以，客户支持系统实际上是生产系统的一个子系统。生产系统还包括其他的子系统，如库存管理和制造子系统。图1-2表示了一个系统是如何划分或分解为子系统的，这些子系统依次可以进一步分解成多个子系统。这种把一个系统划分成多个部件的方法是按照功能分解的原则进行的。

超系统：一个包含其他系统的大系统。

功能分解：把一个系统分为多个基于子系统的部件，这些子系统依次进一步分为多个子系统。

理解系统部件的另一种方法是列出相互作用的各个部分，例如，一个信息系统包括硬件、软件、输入、输出、数据、人和过程，这种观察方法对分析员也是十分有用的，这些相互联系的部件在系统中一起作用，如图1-3所示。

每一个系统在它与它的环境之间有一个边界。任何输入和输出都必须通过这个系统边界，定义这些输入、输出是系统分析与设计的重要部分。在一个信息系统中，人也是关键的部分，他们做一些系统的工作。因此，对系统分析员来说，还有一种重要的边界——自动化边界，在自动化边界的一侧是系统的自动部分，那里的工作是由计算机完成的。而另一侧是系统的手动部分，那里的工作是由人工完成的，如图1-4所示。

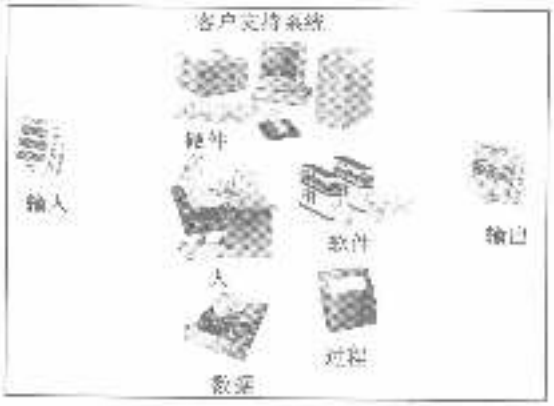


图1-3　信息系统的各部件



图1-4　系统边界与自动化边界的比较

系统边界：系统与环境之间输入和输出必须通过的分界。

自动化边界：一个系统的自动部分和手动部分之间的分界。

1.2.2　信息系统类型

因为组织要完成许多不同类型的活动，因此也就有许多不同类型的信息系统——它们都是新颖的，且使用的是最新技术。在大多数业务中建立的系统包括事务处理系统、管理信息系统、决策支持系统和基于知识的系统。还有支持系统和办公支持系统（如图1-5所示）。你已经在你的介绍性的信息系统课程中了解了这些类型的系统，所以，在这里我们仅对最常见的系统进行简单的复习。

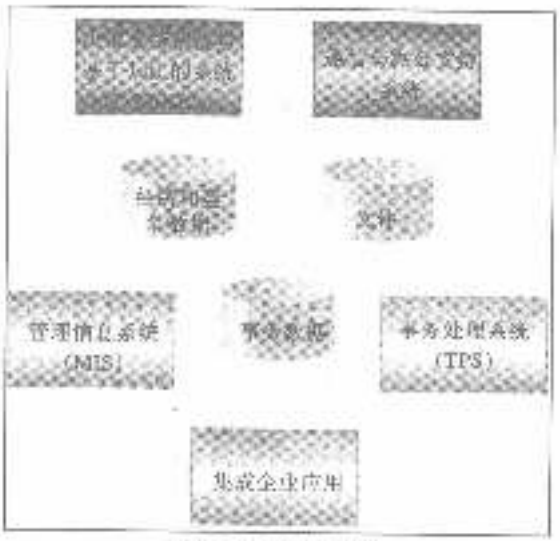


图1-5　信息系统类型

事务处理系统（TPS）收集和记录影响组织的事务信息。每做成一次销售、定购一批物资、做成一次利息付款，就产生了一个事务。这些事务通常会引起贷方或借方在会计分类账上做记录，最终用于财政计算月初账单结算，如收益结算表。事务处理系统是计算机自动处理的第一步，较新的事务处理系统使用一流的技术，并对信息系统开发者带来一些很大的挑战，也带来一些最大的竞争优势和最好的投资回报。在事务处理中，企业对消费者（B2C）和企业对企业（B2B）的电子商务系统是最新的挑战，新的事务处理系统经常被叫做联机事务处理（OLTP）系统。

管理信息系统（MIS）接受事务处理系统收集的信息并为管理人员生成计划和控制业务所需的报表。因为信息已经由事务处理系统收集并存放在业务的数据库中，因此，管理信息系统是可以实现的。执行者所使用的管理信息系统通常包括竞争环境中的外部数据，如关于竞争者、证券市场报告、经济预测、战略规划信息等新数据。

决策支持系统与基于知识的系统（DSS/ KBS）允许用户探究有效的选择或决策的影响。有时这个过程要参照“如果……将会怎样”的模式进行分析，因为用户要求系统回答诸如

"如果第三季度销售额行情下跌至1亿美元以下，并且利率上升到7.5%，怎么办？"的问题，由DSS做出的金融预测能探究这一结果。一些决策支持系统用于日常事务运作的决策，如根据预测商务旅行模型，预测将有多少租用的汽车从一个城市开到另一个城市去度周末。一些系统利用专家获取知识并模拟专家的决策方式。

事务处理、管理信息、决策支持与基于知识的系统通常集成并共享过程与数据。支持企业范围内的操作与数据的高度集成信息系统常指企业应用。有时当一个系统包含许多功能时，区分它是TPS还是MIS是困难的。

业务组织也经常用到许多其他类型的系统。通信支持系统允许职工相互通信并与客户和厂商通信。通信支持现在包括无线个人数字助理（PDA）、支持信息和个人数字处理特征的手机、便利的电子邮件、宽带Internet访问和桌面视频会议。办公支持系统帮助职工创建和共享包括报表、建议和备忘录的文档。办公支持系统也帮助维护有关工作进度表和会议的信息。

事务处理系统（TPS）：收集和记录影响组织的事务信息的信息系统。

管理信息系统（MIS）：接收事务处理系统收集的信息并为管理人员生成计划和控制业务所需报表的信息系统。

决策支持系统和基于知识的系统（DSS/KBS）：允许用户探究有效的选择或决策效果的支持系统或自动制定决策路线。

企业应用：支持企业范围内的操作与数据的高度集成信息系统，通常包括TPS、MIS及DSS/KBS等系统的某些方面。

通信支持系统：允许职工相互通信并与客户和厂商通信的支持系统。

办公支持系统：帮助职工创建和共享包括报表、建议和备忘录的文档的支持系统。

1.3 系统分析员所需的技能

系统分析员（或任何一个做系统分析与设计工作的专业人员）需要各种各样的专门技能。首先，他们需要熟悉如何建立信息系统，这就要求有相当多的技术知识。其次，如前面讨论的一样，他们必须熟悉自己正为之工作的业务，以及该行业如何使用各种类型的系统的情况。最后，分析员需要熟悉相当多的客户及其工作方式，因为这些客户是信息系统的使用者。这三种类型的知识和技能由图1-6所示。

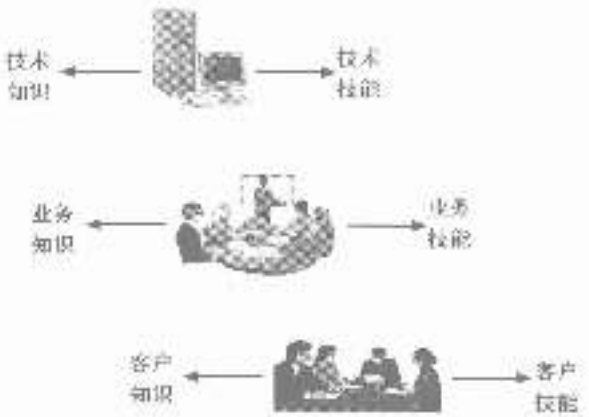


图1-6 系统分析员所需的知识和技能

1.3.1 技术知识与技能

一个系统分析员需要技术性的专门知识，这并不奇怪。尽管一个分析员并没有编程的责任，但熟悉各种不同的技术仍然是十分重要的，如它们用来做什么、如何工作、如何改进。没有一个人可以成为精通所有技术的专家，但许多各种技术的专家可以在一起商议细节。系统分析员应该掌握下列有关基础知识：

- 计算机及其工作原理
- 与计算机有关的设备，包括输入设备、存储设备和输出设备
- 连接计算机的通信网络
- 数据库及数据库管理系统

- 程序设计语言
- 操作系统和各种应用程序

系统分析员也需要了解许多开发系统的工具和技术。工具是帮助规划分析和设计的说明书并完成系统部件的软件产品。在系统开发中使用的一些工具包括：

- 用于开发系统的软件包，如Microsoft Access、Oracle Developer和IBM WebSphere Studio
- 专门的程序设计语言的集成开发环境（IDE），如Sun ONE Studio for Java或Microsoft Visual Studio.Net for VB.NET和C#.NET
- 计算机辅助系统工程（CASE）工具，如 Rational XDE Modeler、Visible Analyst 或 Embarcadero Describe。它可以存储由分析员创建的系统说明书的信息，有时能生成程序代码。
- 程序代码生成器、测试工具、配置管理工具、软件库管理工具、文档支持工具、项目管理工具等

工具：用于帮助规划分析与设计说明书并完成系统部件的软件产品。

技术是用于完成特定系统的开发活动的。如何计划和管理一个系统开发项目？如何完成系统分析？如何完成系统设计？如何实施与测试？如何安装和支持一个新的信息系统？本书的大部分内容解释了如何使用专门技术进行计划、分析和设计，但它也包含了实施与支持的一些概念。这些技术包括：

- 项目规划技术
- 成本/效益分析技术
- 查询技术
- 需求建模技术
- 结构设计技术
- 网络配置技术
- 数据库设计技术

技术：完成特定系统开发活动的策略。

1.3.2 业务知识与技能

对于分析员很重要的知识与技能一般还包括用于熟悉业务组织方面的内容，因为所要解决的问题毕竟是一个业务问题。那么，分析员需要了解什么呢？举例如下：

- 组织要实现什么样的商务功能？
- 组织是如何构成的？
- 组织是如何管理的？
- 在组织中有什么类型的工作（金融、制造、市场、客户服务等）？

一般来说，系统分析员得益于对业务的全面熟悉，所以，他们在大学里基本上都学习了业务管理。事实上，商学院为此经常开设计算机信息系统（CIS）或管理信息系统（MIS）的主修课程。在CIS或MIS学位培养计划中选修会计、市场、管理、经营课程的很重要的目的就是为毕业后求职。诸如计划、进度、预算、灵敏度分析及管理报表的项目管理技术是尤其重要的。

系统分析员也需要熟悉为之工作的组织类型。有些分析员一生专门研究一个特定的行业，如制造业、零售业、金融服务业或宇宙空间业。原因很简单，熟悉一个特定行业需要花费大量的时间。一个非常熟悉某特定行业的分析员能够为这个行业中的公司解决一些复杂的问题。

熟悉一个具体的公司，可对系统的需求与变化提供很重要的指导。通常，对公司里员工

和公司文化的细微之处了解多少很大程度上决定了一个分析员的工作成效。要真正了解一个公司，需要多年的工作经历。若一个分析员了解一个组织如何工作的信息越多就越能做得有成效。分析员需要了解公司的一些细节包括：

- 这个组织是干什么的
- 成功的原因何在
- 它的战略和计划是什么
- 它的传统与价值是什么

实践推荐

在得到系统解决方案之前，确信你已了解了相关组织、文化、任务及目标。 ■

1.3.3 人的知识与技能

因为分析员在开发小组经常与其他员工一起工作，所以，系统分析员需要熟悉人的许多方面，并掌握与人们沟通的技巧。一个分析员要花大量的时间与人们一起工作，设法理解他们对要解决的问题的看法，这一点很重要。一个分析员要熟悉这些内容：

- 人们是怎样想的
- 人们是怎样学习的
- 人们是怎样应变的
- 人们是怎样交往的
- 人们是怎样工作的（包括各个工种和级别）

分析员必须熟悉人们的想法，以便他们能更好地预见人们与计算机系统交互的方法，就像让计算机预见人们的行为一样。由于人们必须学会使用一个新系统，所以设计训练材料和系统帮助时了解人们如何学习是十分重要的。当完成一个新系统时，它能改变人们的工作方法，他们必须准备应变，并且接受了解这种变化所产生的利益。一个分析员必须使用各种技能——包括个人技能和集体技能——去获取所需信息，影响并激发人们与之合作。这类技能包括人际关系技能和协同服务技能。最后，由于信息系统是为支持人们工作而设计的，因此，需要熟悉完成各种工种、各种水平、熟悉从秘书、工厂工人到经理和管理人员的工作。因为分析员需要与整个组织内的如此多的人员接触交往，所以他们就会有相当多的机会去影响这个组织。

1.3.4 诚实与道德

学生们经常低估信息系统领域中的职业特点——诚实与道德的重要性。系统分析员要去调查一些涉及某个组织的许多不同部门的信息问题，这些信息可能不是公开的，特别可能是属于个人隐私，如薪水、健康和工作表现等信息，分析员必须诚实地保守这些秘密。

然而分析员面对的问题也涉及保密的公司信息，包括有关产品和计划产品的专有信息、战略规划或策略，甚至涉及政府军事合约的绝密信息，有时在分析员的工作中可能会涉及公司使用的安全过程或特定的安全系统。

一些系统开发者为咨询公司工作，这些公司是为客户解决特殊问题的。当在工作中接触到非公开的专有信息时，要求他们坚持高度的道德标准。

1.4 分析员周围环境

1.4.1 面对的各种技术

多数学生已经掌握了操作与维修个人计算机的技能，在许多大学的MIS或CIS学位培养计

划中，学生利用Java或VB.Net在PC上完成一些不大的课程设计。但桌面系统并不具有所有业务功能，有时学生并不能认识到这与走入现实社会后将要从事的大规模系统有多么大的差异。

现在许多基本系统是在桌面个人计算机上运行的，但这些计算机可能秘密地通过十分复杂的网络连接到数据中心。一个公司中"简单"的在线订单处理应用可能会涉及一个分布在数百个场所具有数千个用户的系统，这个数据库可能有数百张表，每张表中有数百万条记录。这个系统用了几年的时间才完成，花费了几百万美元。如果这个系统等运1小时，这个公司就可能在销售上损失几百万美元。这样一个系统对于业务是极其重要的。从事支持、维护这个系统工作的程序员和分析员最夜不停地轮班工作，经常当帕出现问题时的报警声。这些系统对于业务的重要性并不言过其实。

在后面的几章中将讨论未来的分析员可能遇到的不同信息系统的配置，包括：

- 桌面系统
- 共享数据的网络桌面系统
- 客户-服务器系统
- 大规模集中式主机系统
- 使用Internet、Intranet和Extranet网络技术的系统

正如一个组织的业务环境不断地变化一样，用于这个信息系统的技术也将不断地变化。技术的快速变化经常促使需求的变化。因此，不断更新知识与提高技能，对从事信息系统开发的每一个人都是很重要的，否则，他将被淘汰。

1.4.2　应用Web技术提高灵活度

企业应用越来越需要基于Web技术提供灵活的开发环境以便于客户、员工与其他业务人员可以随时随地地访问系统与数据。事实上，大多数人都认为对于业务应用Web技术正迅速取代Windows桌面环境与局域网。本文所涉及的分析和设计工具与技术均适用于应用Web技术的开发环境。

电子商务的迅速发展也是应用Web技术而产生的。业务机构对消费者（B2C）系统允许客户通过Web与商家直接交易。客户可以查看公司目录、浏览详细的产品特性，参考经常被询问到的问题，并可以利用购物车的用户界面直接下订单。一旦下了订单，客户就可以跟踪订单状态并可以得到相关运输信息。其他基于Web技术的系统，如业务机构对业务机构（B2B）系统也允许商家互相接触并通过以太网完成交易，如订货、安排运输、处理银行交易、支付员工健康保险费，甚至与国税局联系。先前通过手工处理及日常文书工作的许多B2B交易如今可以利用Web技术通过以太网直接处理。

员工职位明显地影响他们所需求的技术。管理层的员工期望在他们的工作中为他们所使用的信息系统使用Web技术。区域销售代表希望利用以太网查询客户订单。许多员工利用基于Web技术的订单处理、财务及产品计划系统在家工作。甚至员工与人力资源部门交流效益及工薪扣款也使用Web技术。的确，很难想象利用Web技术完善后的企业应用将会对企业多么有用。

1.4.3　典型的工作职位与环境

我们从解决问题和系统开发的角度讨论了系统分析员的工作，然而，应该认识到有许多不同的人在从事系统分析与设计工作，他们不一定都有系统分析员的职位。有时分析与设计工作是由分到项目组提供专门知识的最终用户完成的，分析与设计工作经常与其他任务（如编程或最终用户支持）相结合。一个刚毕业的学生作为一个程序分析员，其工作将是编写程序进行系统维护和项目支持。即便如此，编写程序也要分析需求的变化，设计出解决方法和

处理这种变化的工具。从这方面看，一个新手程序员也需要进行分析和设计。

你可能遇到的一些职位是：

- 程序分析员
- 业务系统分析员
- 系统联络员
- 最终用户分析员
- 业务顾问
- 系统顾问
- 系统支持分析员
- 系统设计师
- 软件工程师
- 系统结构设计师
- Web站点管理员
- Web开发人员

有时系统分析员也可能称做项目领导或项目经理，要准备给做分析与设计工作的所有人考虑各种实情。

在所有不同规模的组织中，如小型业务、中规模地方业务、国家财富500强业务和跨国公司，也都能找到从事分析与设计工作的人。根据业务的规模，所采用的技术和项目类别是有很大区别的。此外，当一些业务有服务于组织内特殊部门的小型信息系统时，有的业务已经有相当集中的信息系统部门或科室。业务规模并不总是与系统规模相对应的，所以，在小公司工作的分析员会有大公司的感觉，而在大公司工作的分析员也会有小公司的感觉。

但是，分析员并不都直接为要解决问题的公司工作，完成分析与设计工作有许多不同的工作约定，包括：

- 为公司工作的程序分析员（部分时间从事分析与设计工作）
- 为公司工作的系统分析员（专门从事分析与设计）
- 独立承包人（在企业经理的指导下，作为分析员或程序分析员进行工作）
- 外包供应商雇员（在公司合同约束下，指定的项目的坐班或不坐班工作）
- 顾问（为公司从事指定的项目，一个指定项目为咨询公司工作）
- 软件开发公司雇员（从事开发和支持的软件包工作）
- 应用服务供应商（ASP）雇员（从事开发和支持的ASP系统解决方案工作）

在其他情况下也能从事分析与设计工作，当然，像Microsoft，Sun Microsystems和IBM等公司雇佣系统开发者创建或改编诸如Office XP这样的软件包和诸如Windows XP这样的操作系统。这些开发者也做分析与设计工作，但要解决的问题通常是不同的。计算机科学专业的大学毕业生更愿意做Windows NT或Office XP工作，在那里，他们的技术可以发挥最大的功效。在本书中，我们描述的分析员是从事解决业务问题的信息系统的设计与分析工作的，而不是操作系统或软件包。

1.5 战略规划中分析员的作用

我们已经描述了要通过开发或维护信息系统解决具体业务问题的系统分析员。分析员也可能与高级经理一起涉及战略经营问题，即涉及组织的未来和确保组织生存与发展的计划与过程。一个大学毕业仅几年的分析员有时会被最高级主管人员召集会议，甚至要求提出实现公司目标的建议。为什么会这样呢？

1.5.1　特殊项目

首先，分析员可能正在致力于解决影响主管人员的一个问题，例如，设计一个主管信息系统。那么，分析员就有可能与主管人员协商，找出他们工作所需的信息。一个分析员可能会被邀请与主管人员一起工作一天，甚至与主管人员一起去旅行，以便得到主管人员工作的感觉。然后，为了更好地了解主管人员的需要，分析员可能要建立并举例说明系统原型。

另一个可能让分析员参与战略经营问题的情形是业务流程重组的研究。业务流程重组试求根本性地改造一个业务运行中的工作性质，这个目标是要根本地改进性能，而不仅仅增加改进。因此，要求分析员参与这个研究，认真考虑原有业务流程，然后提出有重要影响的信息系统解决方案。许多分析与设计的工具和技术可以用于分析和重新设计业务流程，然后提供计算机支持使之工作。

业务流程重组：试求改进一种某业务运行中工作性质的技术，以实现根本性提高性能的目标。

1.5.2　战略规划

大多数业务组织投入相当多的时间和精力完成基本上能适应未来5年或5年以上的战略规划，在这个战略规划期间，高层管理人员试图解决有关公司的诸如他们现在所处何等境地、他们追求何等境地和他们必须做什么才能到达那里等问题。一个基本的战略规划过程要用几个月甚至几年的时间，计划经常要不断更新。这项工作涉及整个组织的许多人，他们一起来完成预测与分析，这些都被纳入整个的战略规划中。一旦一个战略规划制订了，它就能推动整个组织的所有过程，所以，这个组织的方方面面都必须参与并协调规划他们的活动。因此，一个市场战略规划和一个生产战略规划必须适应整个战略规划。

战略规划：高层管理人员试图解决有关公司的诸如他们现在在哪里、他们想去哪里和他们必须做什么才能到达那里等问题的一个过程。

1.5.3　信息系统战略规划

战略规划的一个主要部件是信息系统战略规划。当今的信息系统与组织是紧密联系在一起的。几乎任何计划的改变都要求新的或改进的信息系统。除此之外，信息系统本身经常促进战略规划。例如，随着Internet的快速发展，已经出现许多新型网络公司（例如Amazon.com和eBay）。而其他许多公司Internet已经改变了他们的业务流程并开发新的市场以适应竞争。在其他情况下，新的信息系统技术提供的机会已经导致出现了有意想不到影响的新产品和市场。信息系统及其提供的可能性在大多数组织的战略规划中发挥了很大的作用。

信息系统战略规划：为支持组织的战略规划确定信息系统功能需要提供的技术和应用系统。

信息系统战略规划有时涉及整个组织。通常在首席信息系统执行官的推荐下，最高管理层将批准一个关键项目去为整个组织规划信息系统。

在开发信息系统战略规划中，工作人员要考虑整个组织，预测所要解决的问题，而不是当问题出现后再去应付。有几种技术可以帮助组织完成一个信息系统战略规划项目。咨询公司也经常被请来帮助指导项目。顾问能够提供有关战略规划技术的经验，并对经理和分析员进行培训，以完成计划项目。

通常这样的一个项目会牵涉组织各部门的经理和职员，但一般来说这个项目组是在顾问的协助下由信息系统经理领导的，系统分析员经常专心致志地做信息收集和接待访问工作。

在重新检查许多文档和已有系统后，项目组要设法建立一个反映整个组织能够完成的全部业务功能的模型，也要开发出能够展示整个组织产生和使用的各种数据的模型，项目组要

检查功能完成和产生、使用数据的所有场所。项目组要从这些模型中整理出整个信息系统的数据清单，即应用程序结构计划。随后，这个小组根据现有的系统和其他因素，概述实现要求的系统所需的顺序。

应用程序结构计划：为实现业务职能、一个组织所需的集成信息系统的说明。

在给出所需信息系统的数据清单后，这个小组要决定技术结构计划，即实现整个计划系统所要求的硬件、软件和通信网络的类型。小组必须考虑技术发展趋势，并对专门技术和可能的技术提供者做出承诺。信息系统战略规划的组成部分如图1-7所示。

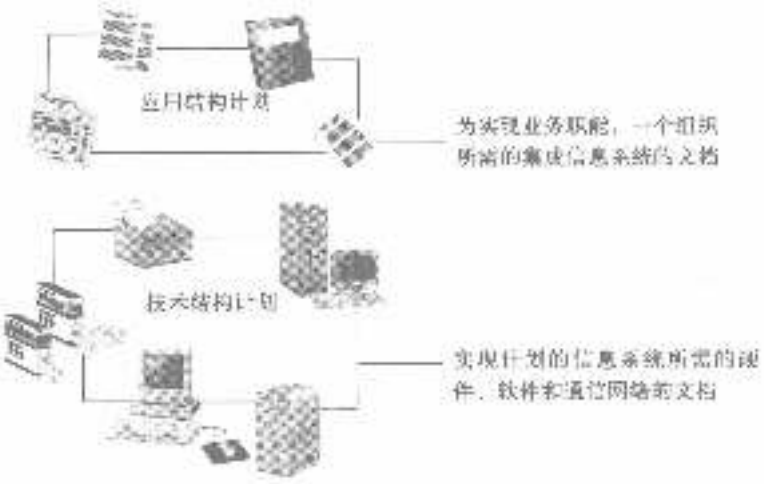


图1-7 信息系统战略规划

技术结构计划：实现计划的信息系统所需的硬件、软件和通信网络的说明。

在理想世界里，一个综合的信息系统计划项目可以解决信息系统经理面临的所有问题。遗憾的是，这个世界在不断变化，计划必须不断更新。未计划的信息系统项目时刻涌现，必须不断地评价需要优先考虑的事情。

1.5.4 企业资源计划

越来越多的组织正在利用一种称为企业资源计划（ERP）的方法提出它们的信息系统需求。企业资源计划（ERP）就是为关键的信息系统提交使用一组集成的软件包，软件供应商如SAP和PeopleSoft为特定行业的公司提供完整的软件包。为采纳ERP解决方案，公司必须认真地研究它已有的过程和信息需求，然后决定哪一个ERP供应商提供的方案最合适。这些ERP解决方案是十分复杂的，它几乎涉及信息系统部门和整个组织的每一个人，要有很大的责任去研究。ERP解决方案的初期费用和支持费用也是十分昂贵的。对于管理人员和全体职员，也会发生许多变化。一旦做出决定要采用一个ERP方案，就很难回到从事商务的老方法和已被ERP解决方案替代的老系统。从事ERP项目的分析员能在项目中发挥重要的作用并将充分利用技术、业务和人的技能。

企业资源计划（ERP）：一个组织为关键信息系统提交使用一组集成的软件包的过程。

1.6 落基山运动用品商店及其战略信息系统规划

在本书中，为证实重要的系统分析与设计技术，以落基山运动用品商店（RMO）的系统开发项目为例说明。RMO是一个运动服装制造商和总经销商，正要启动一个新的客户支持系

统的开发。本书各章都以RMO客户支持系统为例，从现在开始。要设法理解业务性质、公司定义信息系统战略规划的方法和客户支持系统的基本目标。客户支持系统是信息系统战略规划中的一部分。

1.6.1 落基山运动用品商店概述

RMO始于1978年犹他州帕克城的John和Liz Blankens夫妇的梦想。Liz对服装一直很感兴趣，并以半工半读的方式上大学。她为帕克城的当地滑雪商店设计、缝制、出售冬季运动服装，甚至大学毕业后仍继续从事这个副业，不久她就投入了全部时间。

自从在时尚俱乐部上遇到John Blankens后，Liz一直与他约会。John大学毕业后已经在一个零售连锁商店工作了几年，现在刚刚获得MBA学位，他们决定一起设法在零售业扩大Liz的业务以得到一个更大的客户基础。

他们的第一步是凭借一个简单的商品目录直接对客户邮购销售（如图1-8所示）。Liz通过增加了一个设计师和产品监督员立即扩大了制作业务。随着对商品目录兴趣的增加，Blankens夫妇寻找其他的服装种类和装饰品与他们自己的产品一起销售。他们在帕克城也开了一个零售店。

到了2000年初，落基山运动用品商店已经发展成为在落基山和西部诸州的一个重要的区域性的运动服装经销商。亚利桑那州、新墨西哥州、科罗拉多州、犹他州、新墨西哥州、爱达荷州、俄勒冈州、华盛顿州、内华达州和加利福尼亚州东部的娱乐行业有迅速发展的势头。随着人们户外运动兴趣的增加，冬季和夏季运动服装市场已经逐渐发展。滑雪、雪橇、山地自行车、滑水橇、喷气滑翔、慢跑、徒步旅行、ATV自行车、野营、爬山和华氏下降法等运动在这些州开始快速增长。当然，所有的这些运动都需要有相应的运动服装。为适应这个市场，RMO扩大了其运动服的种类，也增加了便服和流行服装种类，以丰富其商品，扩大市场。目前RMO的目录提供了更为广泛的选择（如图1-9所示）。



图1-8　早期RMO的目录封面（1978年秋）

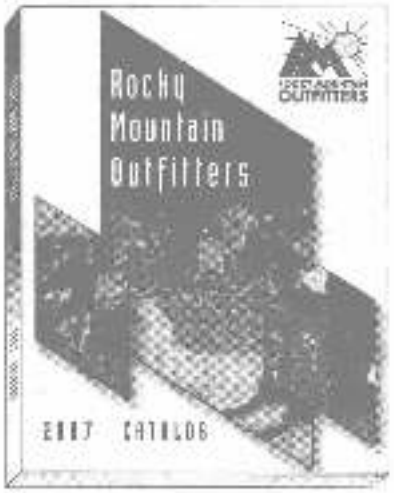


图1-9　现在RMO的目录封面（2007年秋）

落基山运动用品商店现在雇佣了600多人，拥有每年近1.5亿美元的销售额。邮订业务是主要的营业收入来源，达9000万美元。它的零售保持了不小的营业份额。帕克城零售店的年营业额为500万美元，在丹佛最新开张的商店的年营业额为500万美元。20世纪90年代初，

Blankenes夫妇增加了电话定购业务，现在销售额总计有5000万美元。对于客户，这是一个自然的发展，而对于RMO，在体制方面需要做相当大的改变，以适应在线电话定购。

1.6.2 RMO的战略观点

落基山运动用品商店也是最早建立网站展示自己产品的运动服装经销商之一。最初的网站只为RMO进行简单的产品Web展示以加强它的形象，允许潜在的客户申请下载目录。提供一个入口链接到所有户外运动网站。RMO网站的第一次改进是增加了更多的具体的产品信息，包括每周可以通过电话定购的特价商品。最后，在线目录几乎包括了所有的产品，但只能通过邮政或电话下订单。

在21世纪初期，John和Liz已经开始考虑着重偏向业务机构对消费者（B2C）的电子商务，但Liz担心意外的风险和潜在的快速增长。她已经了解到许多小制造商和经销商在完全没有能力维持销售的情况下接入处理在线订单。库存短缺、服务质量差、很差的经销利润和偶发的双重账单可以在短时间内突然使一些成功的并具有良好声誉基础的公司倒闭。Liz决心RMO不犯同样的错误。

John和Liz注意到电子商务的潜力，但他们想谨慎地做好它，而不至于做事后聪明人。Blankenes家族的人计划总是谨慎的，他们也想知道在他们的生意中，信息技术的作用在战略价值方面是否继续扩大。因此，他们决定仔细考虑他们现有的整个信息技术基础设施和建立一个战略信息系统的计划。他们邀请咨询公司帮助他们做战略信息系统计划。

咨询公司提供了两个战略重点建议：

- 供应链管理
- 客户关系管理

供应链管理（SCM）涉及产品开发、产品收购、制造和库存管理的无缝隙集成的过程。客户关系管理（CRM）涉及对直接和间接地与客户进行交互的活动，如市场、销售和服务等提供支持的所有过程。这两个策略有助于业务，特别是像RMO这样的零售业，在提高经营效率的同时为客户提供产品和服务。

供应链管理（SCM）：产品开发、收购、制造和库存管理的无缝隙集成过程。

客户关系管理（CRM）：对直接和间接地与客户进行交互的活动，如市场、销售和服务等提供支持的过程。

信息系统战略规划包括一个详细说明公司需要完成的信息系统开发项目的应用程序结构计划，以及详细说明需要支持这个系统的技术基础设施的技术结构计划。这两个计划的组成部分都是以今后5年确定的RMO的供应链管理目标和客户关系管理目标为基础的。

实践指导

大多数业务执行者明白信息系统在战略中是重要的，他们经常有好的想法与洞察力。确定它们的投入需求。■

下一节提供一些有关RMO的补充背景和概述目前流行的整个信息系统计划。从下一章开始，重点将放在计划中的至关重要的部分——客户支持系统。

1.6.3 RMO的组织结构与所在地

落基山运动用品商店的日常经营仍由John和Liz Blankenes负责，John是总裁，Liz是销售规划与分销副总裁（如图1-10所示）。其他高层管理人员包括市场与销售的副总裁William McDougal和财务与系统的副总裁JoAnn White。系统部门要对JoAnn White报告。

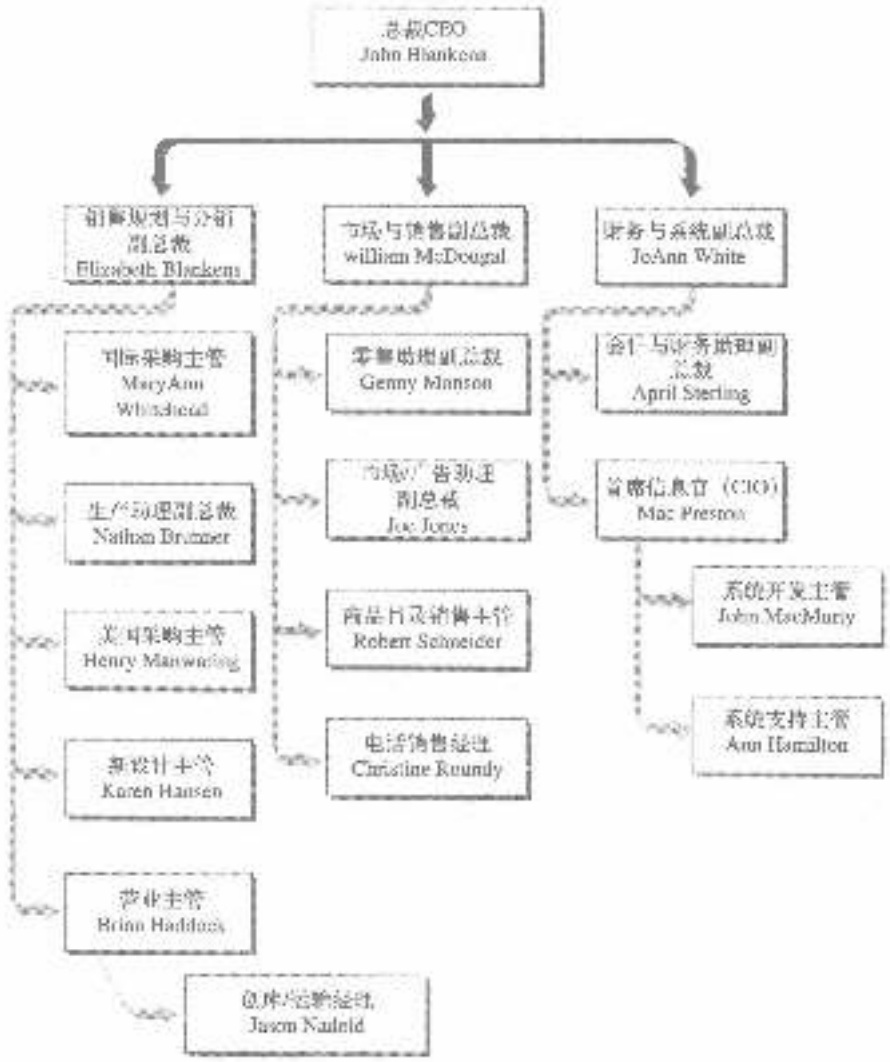


图1-10　落基山运动用品商店组织结构

有113个员工在犹他州伯克城的公司管理部门（人力资源、销售规划、会计与财务、市场和信息系统）工作。有两个零售店：原先的帕克城店和新的丹佛店。制造设施位于盐湖城和离俄勒冈州最近的波特兰。有3个销售/仓库设施：盐湖城、美国新墨西哥州中部大城阿尔伯克基和波特兰。所有邮购处理都在犹他州的普罗沃的某个设备上完成，那里雇佣了58人。位于盐湖城的电话销售中心雇佣了20人。具体分布如图1-11所示。

1.6.4　RMO的信息系统部门

信息系统部门由CIO副总裁Mac Preston领导。这个部门大约有50个员工（如图1-12所示）。随着信息系统计划项目的成功完成，助理副总裁Mac的头衔反映了一种提升。他并不相当于一个完全的副总裁。但这个职务越来越被认为对公司的未来十分重要。Mac向财务会计出身的财务与系统副总裁报告。如果Mac成功地实施这个新的战略信息系统计划，这个信息系统部门最终将对CEO直接报告。

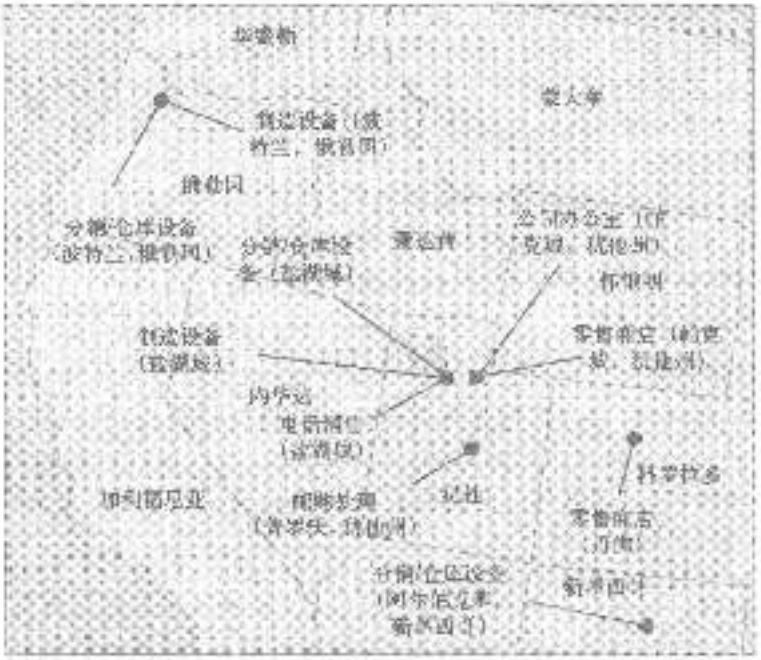


图1-11 普塞山运动用品商店位置

Mac把信息系统组织成两个方向：系统支持和系统开发。Ann Hamilton是系统支持的主管，系统支持包括电信、数据库管理、运行和客户支持这样的功能。John MacMurty是系统开发的主管，系统开发包括4个项目经理，6个系统分析员，10个程序分析员和2个行政支持的雇员。



图1-12 RMO信息系统部门职员

1.6.5 RMO原有的系统

普塞山运动用品商店的大多数计算机技术和信息系统职员在两点城的数据中心工作。一台小型主机通过专用电信线路连接到制造、销售和邮购场所，运行库存、邮购、会计和人力资源等业务。

在商店，销售场所和制造场所的办公职能是通过本地网络的文件服务器支持的。零售店用一个本地服务器运行销售点的软件包，按每批处理更新主机上的库存系统。电话销售中心有一个小型的本地局域网络，一个客户—服务器订单处理应用软件在Windows下运行，对主机进行成批的库存更新。RMO信息网站由一个Internet服务提供商（ISP）负责维护网站内容。

原有信息系统及其技术包括：

- 销售规划/销售，使用COBOL/CICS内部开发的带DB2关系型数据库和VSAM文件的主机应用软件，于12年前实施。
- 邮购，使用COBOL内部开发的主机应用软件，在普罗沃的邮购职员使用专门的终端。这个应用软件快速而有效，但不适合处理电话定购，于14年前实施。
- 电话定购，使用VB和Oracle开发的一个适中的Windows应用软件，作为满足客户电话定

购要求的快速解决方案。它是一个多用户文件服务器的设计，这个设计不能很好地与销售规划/销售集成起来，使用已经达到预期，于6年前实施。

- 零售店系统，带销售点处理和用主机通宵更新批量库存的零售店软件包，于8年前实施。
- 办公系统，在纳克威和其他地方的办公软件、互联网访问和电子邮件服务的一个局域网络，于3年前更新。
- 人力资源，用于薪水和福利内部开发的运行在主机上的一个应用软件，于13年前实施。
- 会计/财务，来自第一流的会计软件包供应商完成的一个主机软件包，于10年前实施。
- RMO信息网站，包含分类信息和其他链接的静态站点，5年前由一个Internet服务提供商提供和维护。

1.6.6　信息系统战略规划

在咨询人员的帮助下，新开发的信息系统战略规划包括技术结构计划和应用程序结构计划。计划小组已经密切考虑到原有系统和RMO业务目标。作为最初的建议，供应链管理和客户关系管理为计划提供了一个框架，这些建议支持RMO战略目标，建立更多的直接客户关系，扩大西部诸州以外的市场。

这个计划的主要特征如下所述：

➢ 技术结构计划

1. 可以把业务应用软件转移到多种场所和计算机系统上。为数据库、Web服务器和电信功能保留了主机，便于增加容量及快速发展。

2. 向在Internet业务过程战略转移，首先要支持供应链管理，其次支持通过网站客户直接定购系统，最后支持客户关系管理（CRM）功能。它可以链接到内部系统和数据库。

3. 预期最后向内联网解决方案发展，如人力资源、会计、财务和信息管理。

➢ 应用程序结构计划

1. 供应链管理（SCM）：实施产品开发、产品收购、制造和库存管理无缝隙集成的系统，以期实现快速销售增长。采用支持咨询的客户开发方式。

2. 客户支持系统（CSS）：实施订单处理和执行系统，把供应链管理各系统无缝隙集成，以支持3种定购处理需求——邮购、电话定购和通过网络的直接客户市场销售。采用客户内部开发方式。

3. 战略信息管理系统（SIMS）：为了战略和运行的决策与控制，实施一个能够提取和分析供应链和客户支持信息的信息系统。采用软件包解决方案。

4. 零售店系统（RSS）：用一个能集成客户支持系统的系统替换已有的零售店系统。采用软件包解决方案。

5. 会计/财务：购买软件包解决方案，是一个内联网应用软件，可以最大限度的计划和控制员工访问财务数据。

6. 人力资源：购买软件包解决方案，掌握intranet应用软件，最大限度地使员工了解人力资源（HR）表、手续和利益信息。

实施应用程序结构计划的时间表如图1-13所示。供应链管理系统的关键部件，特别是库存管理部件必须在客户支持系统启动之前确定下来。作为支持客户关系管理的核心系统，客户支持系统须日必须尽快启动。

供应链管理（SCM）系统和客户支持系统（CSS）需要开发客户软件，要使系统功能与公司特定的需求相符进行客户开发是十分明智的。往往会请顾问帮助公司明确需求和规划供应链管理的综合计划。有几个最主要的咨询公司专门研究供应链管理。



图1.13 RMO应用程序结构计划时间表

尽管解决方案的最终实施要延迟到一个较完整的分析完成以后，但客户支持系统很可能由内部的信息系统职员开发。

计划中的其他系统可能选自目前可以最佳价格买到的软件包解决方案，希望是像财务和人力资源那样的标准商务系统。关键的需求就是任何软件包必须能够使用内联网技术与其他的RMO系统无缝隙地集成。

1.6.7 客户支持系统

本书中描述的RMO系统开发项目指的就是客户支持系统（CSS）。落基山运动用品商店一直以来对自己的客户定位觉得很自豪。RMO的核心能力之一是它发展与维护客户忠诚度的能力。John Blankens能够抓住重要的业务观念的新潮流，十分清楚地发展“顾客至上”的有效的业务过程。这样，尽管客户支持系统包括了所有需求的销售和市场部件，他还是想让每一个人理解系统的主要目标是支持RMO的客户。John有关客户的远见深受处理客户关系管理（CRM）的顾问和软件商们的欢迎。当顾问们对他的职员讲解CRM时，John也深表赞同。

应用程序结构计划详细描述了客户支持系统的一些指定目标。该系统应该包括在从订单登录到出货，到达目的地的全过程中对顾客提供产品涉及的所有功能。例如：

- 客户调查/目录检索
- 订单登录
- 订单跟踪
- 运输
- 延期订货

- 退货
- 销售分析

客户应该能够通过电话、邮政或互联网进行定购。所有的目录项也能够通过落基山运动用品商店的一个精致、完善的网上商品目录得到，并且在线商品目录必须与印刷的目录一致。这样，客户就可以查看印刷的目录，如果他们要选购的话，就可以在线定购。另外，客户可能在印刷的目录中找到某项，然后在线寻找关于此项的其他信息。

定购登录处理需要具备支持图形、网站界面的自助模式，以及电话销售代表和邮寄定购办事员所需的最新式的快速响应的Windows界面。退货、延期订货和订单状况也可以利用在网站和RMO雇员办公桌上进行操作。

尽管某些目标是为系统定义的，但一份完整的系统分析书应该包括对系统的需求的详细说明。这仅仅形成了一些在项目进行中要牢记的指导性原则。

1.7　系统开发级的分析员（课程核心）

我们已经讨论了一个系统分析员在一个组织中能够发挥的许多作用，包括战略规划和帮助确定业务将要做的主要信息系统项目。可是，一个分析员的主要工作是从事一个专门的信息系统开发项目。本书主要介绍如何计划和执行一个信息系统项目，换句话说，就是介绍一个系统开发者的工作范围。本书就是围绕这个主题组织安排的。在这一节中，我们提供本书的一个概要——以BarBara Halif主持的开发过程为例，概括介绍系统开发包括的内容。BarBara Halif负责开发供要启动的RMO客户支持系统项目（见备忘录）。

2007年2月4日

To：John Blankens总裁

From：系统开发项目经理Barbara Halifax

RE：客户支持系统（CSS）项目

John MacMurty建议我给你写封短信，让你知道我有多么激动和自豪，因为我要负责客户支持系统（CSS）项目。去年，我很高兴与你和其他RMO职员一起做战略信息系统计划工作，我知道我十分感谢你与Mac Preston（和John）为这次重要的安排都花了不少好话。

随着SCM系统很好的开展，我们都做好了准备开始CSS项目。我已经检查了所有我们计划的支持开发SCM系统实现与Jack Garcia会面，在IS首席架构中查到的有关供应商与客户系统集成的每一件事情进行一次证实了我们为这些项目需要所做的假设。我下一步要开始做详细的项目规划，现在到正式启动CSS项目的时候了。

当然，我要给John MacMurty写一份正式的形势报告，如果你有什么问题或者有想与我们共享的主意的话，请告诉我们。我知道CSS项目对RMO是多么的重要，我将承担用这个新系统提供完善的客户支持的责任。

感谢这次支援！

BH

cc：Mac Preston, John MacMurty

1.7.1　第一部分：系统分析员

本书的第一部分描述系统分析员的工作，第1章针对要解决的各类问题，所需技能及一个分析员可能工作的职务与场所来描述分析员工作的特性。我们希望至此读者可以明白分析员

要做的工作比所想的和所写的计划多得多。本书的其余部分是围绕在这一章的开头描述的解决问题的方案进行组织安排的。

分析员不仅涉及业务问题，而且能以十分高级的战略观点工作，并在他或她的职业生涯中相对较早地与组织内的各层次的人一起工作。在这一章中，我们也描述落基山运动用品商店和它的战略信息系统计划。本书的其余部分将集中讨论计划的新系统之一——客户支持系统和它的开发。

第2章重点讲述可以用于信息系统开发的各种方法。介绍SDLC，一种作为管理和控制项目的技术。讨论大量工具、技术和方法，包括传统的结构化方法和较新的面向对象的方法。系统的设计者应该熟悉这两种方法的基本知识。本章介绍这两种方法，指出两者的相似性和不同点。

第3章进入系统开发的核心，描述了一个项目是如何计划和管理的。系统开发生命周期（SDLC）使用结构化方法进行项目管理。也介绍了其他项目管理的工具和技术，包括可行性研究、项目进度安排和项目人员。在这几个方面，信息系统项目和其他项目都是类似的。对分析员来说理解项目管理者的角色是很重要的。当计划一个信息系统项目时，可能会出现一些特殊的问题，这要求分析员必须熟悉这些问题以及这些问题对整个信息系统项目规划大的方面的关系和影响。

1.7.2 第二部分：系统分析任务

第4～8章详细地介绍系统分析。第4章讨论收集信息的技术，这些信息是有关新系统要解决的问题。这样，就定义了系统需求。另外，本章也讨论了受这个系统影响的各类人。对所有这些人，需要访问调查并在项目状态上注明日期。介绍了诸如原型法和预排工作的技术，以帮助分析员与所涉及的每一个人沟通。

第5章介绍模型和建模的概念，便于以实用的形式记录系统的详细需求。在讨论一个信息系统时，两个关键的概念特别有用：引起系统响应的“事件”（events）和系统需要存储的有关信息“事物”（things）。事件和事物这两个概念对系统开发无论采用传统的结构化方法还是采用面向对象的方法都是很重要的。为展示在传统方法中影响的事物，作为一种模型，介绍实体-联系图（ERD）。作为在面向对象方法中的事物的一种模型，介绍类图。

第6章和第7章继续讨论建模系统需求。分析传统的结构化方法与面向对象的方法之间的差异。第6章介绍需求的传统方法，重点在过程上，强调了数据流程图（DFD），结构化英语和数据流的定义。第7章介绍需求的面向对象方法，重点在对象和它们之间的相互关系上，强调实例图、状态图和顺序图。系统开发者应该熟悉这两种方法。但对一个给定的系统开发项目，重要的是决定使用哪种方法。这两种需求方法不是同时使用的。但是，许多开发人员现在发现使用用例和用例图对定义功能需求非常有帮助。尽管他们在设计系统时仍然使用传统的设计方法。因此，读者朋友将受益于第7章的案例研究部分。

第8章讲解为实际实施系统产生各种候选方案的技术。讨论每一种方案并仔细地评价了其可行性。最后对管理人员推荐最好的方案。所推荐方案的最后批准对整个项目而言是关键的一个决策点。

1.7.3 第三部分：系统设计任务

一旦选择了可选方案中的一种，就按实际的设计细节开始工作。第9～14章介绍详细的设计要点。第9章提供系统设计的概述，包括设计阶段完成的动作和实施系统阶段的技术环境。介绍传统的和面向对象的方法都使用的三层结构的设计方法。第10章讨论传统方法的系统设

计，介绍了使用的各类模型（系统流程图、结构图和伪码）。第11章讨论面向对象的设计，介绍使用的各类模型（顺序图、协作图、设计类图和包图）。同时也讲解用于评价面向对象设计的一些重要设计模式和方法。

第12章描述为系统设计数据库时涉及的要点，包括使用关系型数据库、面向对象的数据库或关系型数据库与对象技术相结合的混合方法。

第13章讨论系统的用户界面，提供人机交互（HCI）领域的概述和开发用户友好系统的指导原则。这一章包括Windows图形用户界面和用于Web系统的浏览器界面。这些设计概念在传统方法和面向对象方法中都得到应用。

第14章包括对系统界面、系统控制和安全性的设计。系统界面包括各种类型报表的输出设计。这些报表通常都是在线或以书面形式产生的。讨论信息系统控制，包括确保输入正确与完整以及处理正确的重要性。也讨论防止系统非法存取的技术。这些概念也应用于传统方法和面向对象的方法两种情况。

1.7.4　第四部分，实施与支持

第15章描述SDLC的第4阶段和第5阶段：系统实施和系统支持。不论这个系统是如何获得的，这个项目的主要部分正在使这个系统运作并保持这种方法。在实施系统中分析员的作用包括质量控制、测试、培训用户和使系统可操作化（转换）。系统的维护与支持要持续多年时间，包括随着时间的变化去解决问题和增强系统。

新的程序分析员经常会涉及原有系统的维护与支持。系统的维护与支持也是一个项目成本最高的地方。在分析与设计期间所做的决策对维护的简易性和超过生存期的系统全部费用有很大的影响。

本书强调广泛使用迭代和建模的系统开发过程进行系统分析与设计。此外，读者还应该熟悉当前的发展趋势，当前的趋势是强调使用迭代、风险和其他一些技术。第16章讨论统一过程（UP）、螺旋模型、极限编程（XP）、Scrum开发技术、对象框架和基于组件的开发。

1.7.5　网站上的其他材料

在本书的网站上包括一些的重要的附加资料，www.course.com/mis/sad4。先进的面向对象的设计在第1章的在线补充中讨论。另外，软件包的实现方法代替习惯性的系统开发方法是一种可行的方案，在第1章中讨论。更细节的问题在第8章讨论。软件包和企业资源计划在第2章中的在线补充中讨论。本书的网站上有相关附录，包括项目管理、项目规划、财务可行性分析和审查方面的附加参考材料。

小结

系统分析员要用信息系统技术解决业务问题。解决问题意味着要调查问题的大量细节，理解有关这个问题的每一件事，产生几种解决这个问题的可选方案并挑选出最好的解决方案。信息系统通常是解决方案的一部分，信息系统开发涉及编写程序。

系统是一组实现某些结果相互联系、相互作用的部件。信息系统像其他任何系统一样，包含许多部件，并且一个信息系统的结果就是一些业务问题的解决方案。信息系统部件可以考虑为相互作用的子系统，如硬件、软件、输入、输出、数据、人和过程。解决业务的问题，有许多不同类型的系统，包括事务处理系统、管理信息系统、主管信息系统、决策支持系统、通信支持系统和办公支持系统。

系统分析员需要广泛的知识和大量的技能，包括技术、业务和人事的知识与技能。诚实

与道德行为对分析员的成功是极其重要的。分析员会遇到种种经常快速变化的技术。系统分析与设计工作是由各种职务的人员一起完成的。不仅仅有系统分析员同事业有程序分析员、系统顾问、系统工程师和网站开发者，还包括其他一些人。分析员作为独立承包人为咨询公司工作，也为软件公司服务。

系统分析员通过与高层管理人员一起为特殊的系统工作，通过帮助做企业流程重组项目和通过为公司战略规划工作而卷入战略规划。分析员也会以他们的努力帮助业务选择和实施企业资源计划系统。有时一个信息系统战略规划项目是为整个组织实施的。分析员经常会陷入其中。在这一章中描述的落基山运动用品商店信息系统战略规划项目就是一个例子。

通常，系统分析员为系统开发项目工作，解决由战略规划确定的某个业务问题。本书其余部分的重点是：分析员是如何为一个系统开发项目工作，完成项目规划、系统分析、系统设计、系统实施和系统支持活动的。我们使用落基山运动用品商店客户支持系统项目来示范系统开发的整个过程。

关键术语

applications architecture plan	应用程序结构规划
automation boundary	自动化边界
business process reengineering	业务流程重组
communication support systems	通信支持系统
customer relationship management, CRM	客户关系管理
decision support and knowledge-based systems(DSS/KBS)	决策支持和基于知识的系统
enterprise application	企业应用
enterprise resource planning, ERP	企业资源计划
functional decomposition	功能分解
information system	信息系统
information systems strategic plan	信息系统战略规划
management information systems, MIS	管理信息系统
office support systems	办公支持系统
strategic planning	战略规划
subsystem	子系统
supersystem	超系统
supply chain management, SCM	供应链管理
system	系统
system boundary	系统边界
system analysis	系统分析
systems analyst	系统分析员
system design	系统设计
techniques	技术
technology architecture plan	技术结构规划
tools	工具
transaction processing systems, TPS	事务处理系统

复习题

1. 给出一个业务问题的例子。
2. 解决某个问题的主要步骤是什么？
3. 系统的定义。
4. 信息系统的定义。
5. 大多数组织建立什么类型的信息系统？
6. 列举分析员需要熟悉的6种基本技术。
7. 列举分析员开发系统需要使用的4种工具。
8. 列举在系统开发期间使用的5种技术。
9. 分析员一般需要熟悉有关业务和组织的一些什么事情？
10. 分析员需要熟悉有关人的一些什么事情？
11. 分析员可能遇到一些什么类型的技术？
12. 列举涉及分析与设计工作的10种职务。
13. 分析员如何能在他（她）的职业生涯中相对较早地接触高层管理人员和战略规划？

思考题

1. 描述一个你希望解决的你们大学的业务问题。信息技术是如何帮助解决这个问题的？
2. 描述你如何去解决一个你面临的问题，是采用书中描述的系统分析员采用的方案，还是其他任何不同的方案？
3. 这一章描述了许多不同类型的信息系统。给出一个大学可能使用的每一种类型的系统的例子。
4. 技术技能与业务技能之间的区别是什么？解释一个计算机科学专业的大学毕业生为何在一个领域里能力强而在另一个领域里能力弱，讨论一个CIS或MIS毕业生与计算机科学毕业生之间准备上的差异。
5. 解释为什么一个分析员需要理解人们如何想，如何学，如何应变，如何通信和如何工作。
6. 谁的成功需要非常诚实，销售人员还是系统分析员？或者每一个工作的专业人员为了成功都需要诚实与道德？请讨论。
7. 如果系统是客户-服务器而不是大规模集中式主机结构，解释为什么开发一个信息系统需要不同的技能。
8. 咨询公司雇佣的为各种公司工作的顾问为何可能对于理解一个特殊公司面临的业务问题？对顾问容易理解的一个业务问题是什么？
9. 解释为什么一个战略信息系统计划必须涉及信息系统部门以外的人员，为什么要请咨询公司来帮助组织项目？
10. 解释一旦企业资源计划（ERP）已经完成，为什么对此的承诺就很难被取消。

实验练习

1. 对于一个分析员熟悉为之工作的业务的自然状态是重要的。联系一些信息系统开发者并询问他们有关他主的情况。他们对业务的自然状态知道得很多吗？他们喜欢什么类型的课程（例如银行、保险、零售经营、医院管理、制造技术）？他们计划选修其他的课程吗？如果是，选什么课？
2. 考虑各种要求服务类型（为一个特定公司，咨询公司或软件包销售商而工作）。通过查看公司招聘小册子或对站在每个工种前一些研究，他们在寻找一个新的雇员时指明的关键技能是什么？在咨询公司与其他组织之间有任何引人注意的差异吗？

3. 你阅读了洛基山运动用品商店信息系统规划，包括技术结构规划和应用程序结构规划。研究你们大学的系统规划，有专门为今后几年要使用信息技术的计划吗？如果有，描述技术结构计划和应用程序结构计划的一些主要规定。

实例研究

信息技术专业人员协会会议

Alice是本地银行的一个系统开发经理，她正在信息技术专业人员协会（AITP）每月一次的晚餐会上与几个专业熟人会谈。Alice Adams主动说："当我对一个刚毕业的大学生进行面试时，我将确切地告诉他我需要什么样的人才。"AITP不定期地为信息系统专业人员提供聚集在一起和共享经验的机会。通常许多来自各种公司信息系统部门的专业人员参加这种每月一次的会议。

Alice继续说："当我和学生交谈时，我寻找解决问题的技能。我接待的每个学生都说熟悉Java、.NET、Dreamweaver和XYZ或任何一种最新的软件包。但我总问他们一件事情：'一般你如何开始考虑解决问题？'然后我想知道他们是否对银行和金融服务有一些思考，所以，我就问，'这些天银行系统面临的最大问题是什么？'"

Jim Parsons，一个本地医院数据库管理员，大笑起来："是的，我能体会你所说的。如果他们能够意识到一个医院是如何运作的，我们面临的是什么问题，信息技术如何帮助我们解决一些问题，这就会给我留下非常深的印象。这说明他具备很好的观察事物的能力，肯定会引起我的注意。"

Sam Young，一个零售连锁店市场系统的经理，插话说："是，我与你一样，我对申请者的专门技术技能的印象不是那样深。我假设他们有才能和一些技能，我想了解他们如何能做好与人的交流，我想了解他们对我们的生意的特点知道多少，我想了解他们对我们零售店和我们面临的问题有多少兴趣。"

Alice认可说："说得对。"

1. 你同意Alice和其余人有关解决问题的技能，或专门领域的知识，或交流技巧的重要性的观点吗？请讨论。

2. 在向一个医院的信息系统经理申请一个职位之前，你是否应该研究医院是如何管理的？请讨论。

3. 根据你的经历，你认为去一个银行、一个医院或一个零售连锁店工作就实会有差别吗？或者说只要在信息系统部门工作，而不管是那里的信息系统部门都是一样的吗？请讨论。

对落基山运动用品商店实例的再思考



RMO的战略信息系统计划要求在建立客户支持系统（CSS）之前建立一个新的供应链管理（SCM）系统。John Blankens经常讲客户的定位是成功的关键，如果是这样的话，为什么不先建立CSS，这样客户就能立即从改进的客户订单系统的实施中受益？这样不就增加了销售，使利润变大吗？RMO已经有几家工厂生产许多RMO销售的产品。RMO在全球与许多供应商有长时间固定关系，建立了很好的产品目录，业务上有许多忠实的并愿意在线购买的老客户，为什么要等？也许John Blankens在计划时犯了错误。

1. RMO决定在建立客户支持系统之前建立供应链管理系统的理由是什么？

2. 如果延迟建立客户支持系统是错误的，那么对RMO的后果是什么？

3. 如果RMO改变了他们的看法，在建立供应链管理系统之前启动客户支持系统，那么对RMO的后果是什么？

4. 你可能对RMO战略信息系统计划（应用程序结构计划和技术结构计划）做哪些改变？请讨论。

关注Reliable Pharmaceutical Services

Reliable Pharmaceutical Services是于1975年在美国新墨西哥州阿尔伯克基城成立的一个私人公司。它为那些因规模太小而没有自己内部药房的卫生保健机构提供药品配送服务。Reliable在其刚成立的前10年发展非常迅速，到20世纪80年代后期，它的客户包括24个疗养院，3个本地康复机构，2个小型的精神医院和4个小型的特殊内科医院。到1990年，Reliable把它的服务区域从阿尔伯克基扩展到圣达菲，并开始在拉斯库和普洛普开展新的服务领域。

Reliable接收客户机构的患者药物订单，然后每隔12个小时就按照订单用密封箱装好药品发送出去。在阿尔伯克基和圣达菲，Reliable雇佣了大约12个发送员、20个药剂师助理人员（PAs）、6个注册药剂师和10个办公室工作人员。另有15个雇员在拉斯库和普洛普工作。管理团队包含另他6个人，他们是公司的主要拥有者。

每个卫生保健机构人员通过电话提交患者处方订单。许多处方都是长期有效的订单，每次都需要发送直到取消为止。订单在接收的时候被输入计算机中。在每12个小时循环开始的时候，计算机为客户机构的每一楼层或者每一片楼层产生药品清单。药品清单上标明每个病人和给他开的药品，包括何时以及如何吃药。值班领导把药品清单送到药剂师手里，药剂师再把任务分配给药剂师助理人员（PAs）。药剂师监督并协药剂师助理（PAs）们的工作。

一个病人的所有的药品收集起来放到一个密封箱的某一个塑料抽屉里。每一个箱子都标有部门名称、楼层号和侧楼号。每一个抽屉标有病人姓名和房间号。里面有时插入一些隔离板以区分一个病人的多个处方。当订单中的药品都已装配好后，药剂师对药品清单的内容再做最后一次检查并在清单的每一页签上字，然后把两份清单放进药品箱子里，一份放到装配区的文件柜里，另一份放到邮政管作为记账依据。当所有的箱子装配好后，它们就装载到卡车上，然后发送到各个卫生保健机构。

订单输入、记账和库存管理手续是手工和计算机辅助方式的综合。Reliable在个人计算机上综合使用了Excel电子数据表格、Access数据库和旧的定制开发的记账软件，药剂师助理人员使用定制开发的记账软件输入电话订单并产生药品清单。随着公司合同的增加和医疗保险及公共医疗补助手续的复杂化，系统变得越来越笨拙。有些花费用于卫生保健机构，有些用于保险公司，有些用于医疗保险以及公共医疗补助方面，还有一些直接用于病人。开发维护记账软件的公司软件已经不存在了，办公室的员工不得不围绕软件的缺点和局限性做大量讨厌的工作。此外，库存管理需要手工完成。

2001年，Reliable公司缴税4千万美元，赢利550万美元。到2005年，每年缴税降低了4个百分点，每年的赢利也降低了8个百分点。下降的原因包括下面几个方面：

- 在与卫生维护组织和大的国家医疗保健公司管理的机构签订医疗保险及公共医疗补助和合同时的价格控制。
- 像Walgreens和内部药房等国家药品连锁店竞争的增加。
- 低效的操作程序，几乎20年没有进行彻底的检查。

Reliable公司的管理团队花费了去年大部分时间制订了一个战略规划，其关键就是致力于流线型的操作以提高服务质量和降低成本。在将来大的卫生保健公司可以任意制订价格和选择任何公司外购药品服务的大环境下，管理者们把他们的这种努力看成将来生存的唯一希望。他们计划当系统运行带来的效益能够补偿其花费并提高经济水平之后就应该向其他州进行有意义的推广。

本章讨论的两个公司中，Reliable比RMO小得多，但是Reliable仍然需要一个完整的信息系统，以支持它的运行和管理。在每章结束的时候，我们将介绍一个能够应用本章概念的Reliable Pharmaceutical Services案例。

1. 您认为Reliable的信息系统需要雇佣多少员工比较合理？他们需要掌握哪些综合技能？根据他们每天的工作，他们必须如何灵活地合作？

2. 网络技术对Reliable公司开发这个系统有那些影响？网络技术能否改变Reliable公司开展业务的方式？

3. 为Reliable Pharmaceutical Services制订一个未来5年的应用程序结构规划和技术结构规划。在你的规划中，什么系统项目排第一位？什么系统项目紧随其后呢？

参考资料

要更深入地了解信息系统概念，可参阅：

Effy Oz, *Management Information Systems (5th ed.)*, Course Technology, 2007.

Kathy Schwalbe, *Information Technology Project Management (4th ed.)*, Course Technology, 2005.

Ralph M. Stair and George W. Reynolds, *Principles of Information Systems (7th ed.)*, Course Technology, 2006.

第2章　系统开发方法

学习目标

完成本章后，应具备如下能力：

- 阐述系统开发生命周期（SDLC）每个阶段的目的
- 阐述何时对系统开发生命周期使用自适应方法代替预测性的传统系统开发生命周期
- 阐述模型、工具、技术和方法之间的差异
- 描述用于信息系统开发的两种常用方法：传统方法和面向对象方法
- 描述系统开发生命周期（SDLC）的一些变体
- 阐述在系统开发当前趋势中的关键特征：统一过程（UP）、极限编程（XP）、敏捷建模和Scrum
- 阐述自动化工具在系统开发中是如何应用的

本章要点

- 系统开发生命周期
- 系统开发生命周期各阶段的行为
- 方法、模型、工具和技术
- 系统开发的两种方法
- 系统开发生命周期的各种变体
- 系统开发的当前趋势
- 支持系统开发的工具

Ajax Corporation、Consolidated Concepts和Pinnacle Manufacturing的开发方法

Kim、Mary和Bob是3个即将毕业的大学生，他们正在讨论最近来校园招聘CIS专业学生的各公司面试的情况。他们一致认为，虽然一开始他们觉得有点不知所措，但通过面试学到了很多东西。

Kim小心翼翼地说："刚开始我不能确信我是否知道他们在谈论什么。"在面试中，Kim在数据建模方面的知识给Ajax Corporation的面试人员留下了深刻印象。第二轮面试时，她参观了Ajax Corporation本部的数据中心。同时，面试人员花了大量时间向她介绍了公司的系统开发方法。

Kim继续说道："一些人说忘了在学校所学的东西。"Ajax Corporation已从一个小咨询公司购买了一套成熟的方法论——IM One。多数员工认为该方法论很好。Ajax Corporation的老员工认为IM One是独一无二的，他们为此而感到骄傲。为此，他们投入了大量时间和金钱来学习并适应IM One。

Kim强调说："这引起了我的注意。但此后他们开始向我讲述SDLC、迭代、业务事件、数据流图、实体-联系图以及诸如此类的东西。"Kim发现IM One方法中的许多关键概念都来自系统开发结构化方法经常使用的模型和技术。

"我明白你的意思。"Mary说。（Mary是一个非常有天分的程序员，几乎熟悉每一种新的

可供利用的编程语言）。“Consolidated Concepts一直重复使用着OMG、UML和UP，以及一些被叫做Booch、Rumbaugh和Jacobson的人。但随后我发现他们是在用面向对象的方法开发系统，知道我了解Java和VB.NET他们很高兴。当我了解了他们所使用的所有术语后，就不会有什么问题了。他们说将派我出去学习Rational Rose，这是一种面向对象方法的CASE工具。”

Bob的经历与众不同，Bob说：“一些人说分析与设计已不再是大任务了，但是我认为懂得这些将会节省我在校的很多时间。”Bob参观了Pinnacle Manufacturing公司，这家公司拥有支持制造及存货控制的系统开发小组。Bob还说：“他们说他们设法尽可能快地投入并开始编码，没有文档，没有项目规划。然后他们给我看了一些放在办公桌上的书籍，看起来他们好像是读了大量有关分析与设计的内容。我还发现他们使用极限编程及敏捷建模技术，但他们仅仅关注他们的小项目所需要的最佳的实施方式。这证明他们在建立原型时是通过观察风险及编写用户材料来组织不同工作的。我在老板的白板上看到一些类图及序列图的草图。这让我感到非常舒服。

Bob、Mary和Kim一致认为在这样的工作环境中有许多东西需要学习，但同时用来描述关键概念和技术的许多术语和观点他们在学校中已经学过了。他们很高兴他们一直重视学习CIS课程中的基础知识，并且懂得系统开发中所要用到的许多方法。

概述

Kim、Mary和Bob的经历证明，开发信息系统的方法有很多，同时也是非常的复杂。开发信息系统时，项目经理依靠各种各样的辅助工具来帮助他们完成开发过程中的每一步骤。本章所描述的系统开发生命周期（SDLC）为系统开发过程提供了一个大致的框架。但是开发人员需要掌握许多概念，包括方法、模型、工具和技术。在详细讨论系统开发之前，理解这些概念是非常重要的。

本章回顾了当前用于开发业务系统的两种系统开发方法：传统方法和面向对象方法。传统方法是指结构化系统开发（结构化分析、结构化设计及结构化编程）和信息工程（IE）。面向对象方法是指采用较新的面向对象技术进行系统开发。这种方法需要一种不同的思维方式来进行分析、设计和编程。

传统方法和面向对象方法用SDLC作为项目管理的框架。同时本章也介绍了一些SDLC的重要变体。此外，分析员需要了解系统开发过程中的当前趋势，因为这可能会继续影响分析和设计。最后，系统开发人员需要计算机支持工具来完成工作任务，这些工具包括绘图工具及专门设计的计算机辅助系统工程（CASE）工具。本章将介绍这些软件工具的一些使用实例。本章所讨论的大多数模型、工具和技术将在SDLC的分析和设计阶段使用。

在落基山运动用品商店，作为客户支持系统项目的项目经理，Barbara Halifax的工作之一就是确定采用哪种方法来开发系统，她可以使用本章描述的所有方法。然而，我们不会讲她的最终决定，因为在本教程中当我们提及所有方法的更多细节时，始终以该客户支持系统为例。

2.1 系统开发生命周期

第1章阐述了系统分析员解决业务问题。很明显，要使这一解决业务问题的工作富有成效，必须有组织且目标明确。分析员通过把这一工作组织成项目来实现目标。项目是一个有始有终、有规划的任务，它能得到预先确定的结果或产品。系统开发项目这个术语描述了一个有规划的、产生新系统的任务。一些系统开发项目很大，需要许多人进行数千小时的工作，并且可能会持续几年。在第1章介绍的RMO实例研究中，正在开发的系统是一种基于计算机的中等规模的信息系统，需要一个持续时间不超过一年的中等规模的项目。许多系统开发项目

比较小，持续时间也就是一两个月。要使系统开发项目取得成功，分析员必须有详细的规划。成功在很大程度上取决于有组织的、讲究方法的一系列活动和任务，最终将产生一个可靠、强大而高效的信息系统。

项目：一项有始有终、有规划的任务，它能得到预先确定的结果或产品。

信息系统开发过程中最关键、最基础的概念是系统开发生命周期。企业和组织使用信息系统支持业务所需以实现其功能的大量多变的过程。正如第1章所阐述的，存在许多不同的信息系统，而每个信息系统在支持业务过程方面有其自身的重点和目的。每一个信息系统都有其自身的生命，而我们作为系统开发人员就是提供这个系统生命周期。在一个信息系统的生命中，首先是构想，其次是开发项目过程中的设计、建立及部署，最后是形成成品并用于支持业务。然而，就算在其被广泛使用的过程中，系统也仍然是动态的，活的实体，需要通过更小的工程进行更新、修改和维护。建立、部署、使用和更新一个信息系统的整个过程称为系统开发生命周期，或者称为SDLC。

系统开发生命周期（SDLC）：建立、部署、使用和更新一个信息系统的整个过程。

正如前面提到的，一个系统的生命中有多个不同的项目，首先是开发初始系统，之后是对初始系统进行升级。在本章中——事实上本书的大部分内容——我们将把重点放在初始开发项目而不是支持项目。换句话说，我们首要关心的是开发系统及第一时间部署。

在如今多种多样的开发环境下，我们可以根据不同的系统开发生命周期使用不同的方法开发系统。可以假设一些方法已经使用了很长时间了而且具有可变成功率。在不断转变的系统技术世界里，涌现出新的独一无二的构造系统的方法同样也具有可变的成功率。尽管很难找到包含所有方法的单一、综合的分类系统，但我们可以使用一个有用的技术，那就是根据方法是否更具有预测性和适应性对系统开发生命周期进行分类。从完全预测到完全适应，可以将这两个分类点连成一个连续体（如图2-1所示）。

系统开发生命周期的预测方法是一个可以预先规划并组织开发项目，并可以根据规划对新的信息系统进行开发的方法。对构造系统来说，预测系统开发生命周期易懂、易定义。例如，一个公司希望将老的主机库存系统转变为新的网络化客户—服务器系统。在这类项目中，要求员工清楚地知道需求，而且不再需要添加新的程序，因此，这种项目通常需要仔细规划，系统可根据说明书进行构造。

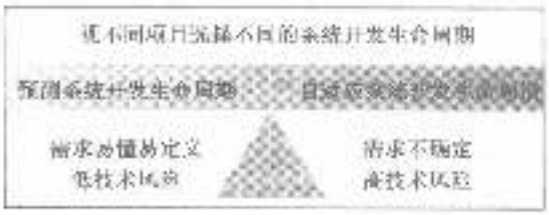


图2-1　系统开发生命周期的预测——适应方法

预测方法：一种可以预先规划并组织开发项目，并可以根据规划对新的信息系统进行开发的系统开发生命周期方法。

在图2-1所示天平的另一端是系统开发生命周期的适应方法。此方法用于系统和用户的需求不明确的时候。这种情况下，项目不能预先规划，在一些初步开发工作之后还需确定系统的一些需求。开发人员仍可以制定解决方法，但要求其必须灵活，而且在项目发生进展的时候可以进行调整。

适应方法：对于不能预先规划的项目，在项目进展过程中可进行调整的更为灵活的系统开发生命周期方法。

事实上，任何一个项目或者说大部分项目都既有预测的元素，又有适应的元素的。图2-1显示了滑动天平两个端点的特性，这两个端点不是互相独立的。预测方法发明于20世纪70～90年代之间，更为传统。很多新的适应方法是随着向对象方法发展起来的，始创于20世纪

90年代并一直发展到21世纪。我们先看一些偏重于预测的方法，然后再看一些新的适应方法。

实践指导

任一指定项目都有预测的元素也有自适应的元素。

2.1.1 系统开发生命周期的传统预测方法

一个新的信息系统的开发需要不同的但相互关联的活动。在预测方法中，我们首先需要一组包括计划、组织和规划项目的活动，通常这组活动称为项目规划活动。这一组活动规划了项目的整个结构。其次，将活动的重点放在理解待解决的业务问题和定义业务需求上。我们把这组活动称为分析活动，目的是更确切地理解系统在支持业务过程中必须做的事情。再次，把活动的重点放在设计新系统上，这组活动称为设计活动，它通过运用先前定义的需求为新系统开发程序结构和算法。为了建立一个系统，还需要另一项活动就是实施活动，这组活动包括为业务用户编程、测试及安装系统。

有时将计划、分析、设计、实施这4组活动称为4个阶段，它们是为管理项目提供框架的元素。另外一个阶段称为支持阶段。这一阶段包括在部署好之后对系统的升级和维护。虽然支持阶段是整个系统开发生命周期的一部分，但总是被排除在初始开发项目之外。如图2-2所示为传统系统开发生命周期的5个阶段。

阶段：与系统开发活动相关的，分为项目规划、分析、设计、实施及支持等部分的活动。



图2-2 信息系统开发阶段

这5个阶段与第1章中所概述的基本问题解决方法非常相似。首先，团队认识到它有问题需要解决（项目规划）。其次，项目开发小组研究并理解问题及解决问题的需求（分析）。一旦弄明白了问题，解决问题的详细方法就形成了（设计）。接下来就可以构造和安装解决问题的系统了（实施）。只要系统一运行起来，为保证它能够持续提供预期的利益就必须对它进行维护和加强（支持）（如图2-3所示）。

SDLC各阶段	目标
计划阶段	确定新系统的作用域，确保项目的可行性，制订进度表和资源分配计划并进行项目其余部分的预算
分析阶段	了解新系统的业务需求和处理要求并做好文档
设计阶段	根据分析阶段的需求定义和制定的决策，设计出解决方案系统
实施阶段	建立、测试和安装可靠的工作信息系统，培训用户并能其受益于系统的使用
支持阶段	在系统初始安装和生命周期的许多年中都保持系统有效地运行

图2-3 SDLC各阶段和对象

SDLC方法在图2-1所示预测/适应天平图的最左端时最具有预测性。这一方法称为瀑布法。如图2-4所示，瀑布法是指项目从一个阶段到另一个阶段连续地进行。首先开发一个详细的计划，接着全面确定需求，之后设计系统，然后编程、测试及安装。一旦经该方法进入到下一个阶段，则不可再返回到上个阶段。事实上，在SDLC的每一步中，瀑布法都需要严格计划和

最终决策。可以想到，一个纯粹的瀑布法是不能很好地完成工作的。作为开发人员，不可能在完成一个阶段中不出现任何错误或者遗留任何重要的部分，必须以后再进行添加。尽管我们不使用纯瀑布法，但它可以为我们理解开发奠定基础。无论开发系统与否，我们仍需要包括计划、分析、设计和实施这4项活动。只是我们不能严格按照瀑布法来做。



图2-4　SDLC的瀑布法

瀑布法：作为一种SDLC方法，将项目的各阶段按顺序实施，其特点是从一个阶段顺序进入另一个阶段。

在图2-1所示预测/适应大平的右端是改进的瀑布法。在这些方法中我们仍需要进行预测，即仍需开发一个十分彻底的计划，但我们发现项目的每个阶段都是互相重叠、影响和依赖的，在开始设计之前必须做一些分析，但在设计中我们会发现在需求方面需要更多的细节，或者一些需求是原先我们没有遇见过的。图2-5展示了这些活动是如何重叠的。



图2-5　系统开发阶段的重叠

另外造成阶段重叠的原因是效率，在项目组成员分析需求的同时，可能考虑并设计各种表格或报表，为了帮助理解用户的需求，项目组成员可能要设计最终系统的一部分，但是当他们做早期设计时，经常要删减部分内容而其他的一直保留到最后的系统中。此外，计算机系统的许多内容是相互依赖的，这就要求分析员既要做分析又要做一些设计。

那么，为什么不完全重叠所有的活动呢？答案是相关依赖性。一些活动自然地要依靠先前工作的结果。在分析员对问题的基本情况还没有一些想法时，项目组成员是不能完成设计的，所以在设计之前就要做一些分析，在项目组成员确定整体设计结构之前，写程序代码也是无效的，因为这将导致以后丢弃太多的代码。

如今，一些公司的信息系统和一些项目都基于改进的瀑布法。对于需要建立易懂的应用项目来说，适合使用改进的瀑布法，也可采用改进瀑布法建立基于面向对象方法的系统。

2.1.2　系统开发生命周期的新的自适应方法

记住在适应方法中，包括计划和模型的项目活动开发方法可以根据项目的进展进行调整。远离大平右端的是一种时兴的方法，称为螺旋模型。螺旋模型包括很多适应性元素，可视为系统开发的首要适应性方法。这种模型用一个螺旋来描述生命周期，从中心开始，一遍一遍地反复向外扩张，直至项目完成。因此这种模型看上去与众不同，使得项目管理的风格也大

相径庭。螺旋模型用图形的方式来表示如图2-6所示。

螺旋模型：一种SDLC方法，其特点是在开发活动中反复绕圈直到项目完成。

实现螺旋模型有很多种不同的方法。图2-6中的例子是从最初的计划阶段（即图的中心）开始的。最初计划阶段的目标就是收集足够的信息以开发出一个最初的原型，计划阶段的活动包括灵活性研究、高层用户需求检查、可选方案的产生及整体设计和实现策略的选择。

最初计划完成之后，就要在第一层原型系统（图中的蓝色环）的基础上进行更进一步的工作。对于每一个原型系统，开发过程都要遵循从分析、设计、构造、测试，到与前面的原型系统组件集成，并且为下一个原型系统做准备的过程。下一个原型系统计划完成之后，活动的循环又重新开始。螺旋模型的方法可适用于任意数目的原型，图2-6中只显示了4个。

原型：显示大系统某些方面的初始工作原型。

螺旋模型方法中一个关键的概念就是集中处理风险，这也是第3章所讨论的项目管理的核心概念。在每次迭代中，尽管有很多进行集中处理的方法可供选择，但螺旋模型推荐确定那些必须进行研究和减轻的风险因素。第一次迭代应该确定系统那些具有最大风险的部分。有时候，最大的风险并不是一个子系统或系统功能的一个集合，相反，最大的风险常常是新技术的技术可行性。如果是这样，第一次迭代就应该重点建立一个原型系统，它能保证技术如期运行。然后，第二次迭代的原型系统就可以集中在与系统需求或其他问题相关的风险问题上了。还有的时候，最大的风险是用户是否能接受变化。这时，第一次迭代必须集中构造一个原型系统，可用来向用户展示新系统将使他们的工作内容丰富起来。

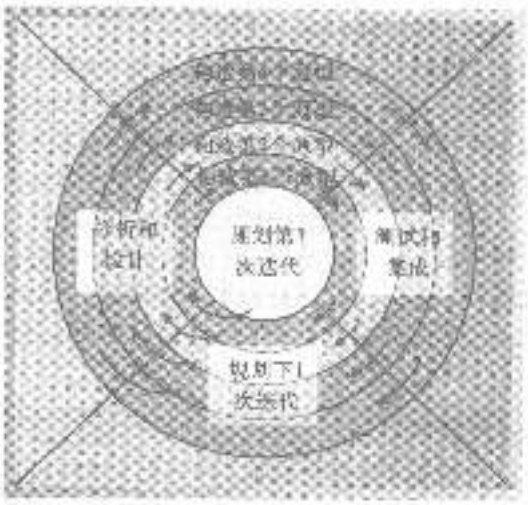


图2-6 螺旋型生命周期模型

图2-6使用了团队迭代显示了逻辑模型。在解决问题中，用迭代将大的复杂的问题分为更小的更易管理的问题。把每个小的问题依次解决了才能解决大的问题。系统开发使用迭代也是出于同样的目的。我们拿到一个系统，开始时把它分割成更小的几部分，然后计划、分析、设计和实施每个小部分。当然，我们还要增加集成的步骤，把各个部分整合成一个完整的解决方案。这一方法经常被称为SDLC的迭代方法。如今一些流行的适应方法把迭代作为此方法的基础元素。图2-7显示了迭代

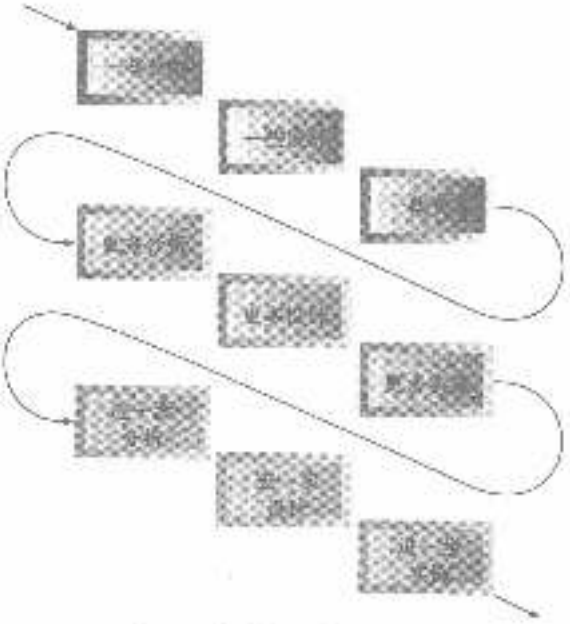


图2-7 迭代交叉生命周期阶段

方法是如何工作的。

迭代意味着任务做了一次，接着又一次，然后又一次，任务是不断重复的。随着每一次迭代，结果得到修正，并且越来越靠近目标。迭代假设我们不可能在第一次就得到正确的结果。在信息系统开发中，在真正知道系统是否能够运转并实现其目标之前，首先需要做一些分析和设计工作，然后，要做更多的分析和设计工作来对系统做出新的改进。按照这种观点，在开始设计工作之前要完成分析（界定所有的要求）是非常困难的。类似地，在不知道（特别是采用不断变化的技术时）如何运作之前要完成设计是非常困难的。因此，可以首先完成一些设计工作，然后再做一些实施工作。此后，迭代过程继续进行——你需要做更多的分析、设计及实施工作。当然，所使用的迭代的数量取决于项目的复杂程度。

有几种方法可以定义每一次迭代。一种方法就是先定义系统所必须包含的一些关键系统功能，并在第一次迭代中实现这些关键功能。这些功能完成后，再去实现那些需要但不太关键的系统功能。在最后一次迭代中实现那些可选的或"最好有"的系统功能。另外一种方法是每次关注于一个子系统。第一个被实现的子系统包含核心功能和其他子系统所必需的数据。然后，在下一次迭代中包含另外一个子系统，依此类推。

有时是要根据复杂性和所包含的风险（比如后面将要介绍的螺旋模型）来定义迭代的。系统中最复杂或最有风险的部分常常首先处理，以便及早发现复杂部分是否能够如期完成，从而在必要的情况下对计划早做调整。有时，为了尽可能早地完成系统的大部分功能，也会把系统的最简单部分首先处理。如何定义迭代依赖于很多因素，并且很可能在我们所遇到的每一个项目中都是不同的。

实践指导

声明项目的各个方面，在早期项目迭代中定位最大风险。 ■

有一种相关方法称为增量开发。在这种方法中，可以通过一次或多次迭代完成系统的各个部分，然后提供给用户使用。这种方法可以尽可能早地将系统的某些部分提供给用户使用，以方便用户。在此基础上，可以继续进行几次更多的迭代来开发系统的另一部分，并集成到第一部分中，提供给用户使用。最终完成系统最后的部分，并与其他部分集成起来。如今，大多数开发都是通过进行不同的迭代度完成的。面向对象的方法总是被描述成渐进迭代。

增量开发：通过一次或多次迭代完成系统的各个部分并依次运行，然后供用户使用的一种开发方法。

2.2 每个SDLC阶段的活动

描述每一个SDLC阶段以及每个阶段的活动通常是如何迭代执行的。接下来我们详细讨论SDLC的每个阶段——项目规划、分析、设计、实施和支持。

2.2.1 计划阶段

计划阶段的主要目标是确定新系统的作用域，确保项目可行性，制定进度表并进行项目其余部分的预算。项目规划阶段分为5个活动：

- 定义问题
- 制定项目的进度表
- 确认项目的可行性
- 安排项目人员
- 启动项目

计划阶段：SDLC最初的阶段，它的目标是确定新系统的作用域并做出项目规划。

计划阶段最重要的活动是准确地定义业务问题和所需解决方案的范围。在这个阶段里，你不会知道即将包含在系统内的所有功能或过程。可是，确定新系统的主要用途和新系统必须解决的业务问题是非常重要的。

很明显制定项目进度表和安排项目人员这两个活动是密切相关的，必须制定一个包括任务、活动和所需人员的详细项目进度表，很幸运的是，有很好用的方法和工具来支持这个活动的成功完成。这些方法与工具将在下一章介绍。大型项目需要用到具有明确的、可识别标志的详细进度表和控制程序。这个阶段的关键部分是在项目开发期间确定在要求的时间里能够获得需要的人力资源。

下一个要点是确认这个项目是可行的。许多项目是作为企业范围内战略规划的一部分启动的。在整个计划中，每个项目必须有自己的优点。可行性分析要调研经济、组织、技术、资源和进度表的可行性。在这一章的后面将详细说明每种类型的可行性分析。

最后，高层管理人员检查这个项目的总计划并启动这个项目。项目启动需要分配资金，分派项目人员和获取其他必要的资源，如办公室和开发工具。项目启动时通常要下达一个项目启动的正式通知。

2.2.2 分析阶段

分析阶段的主要目标是了解新系统的业务需求和处理要求并制作书面文件。分析本质上是一个发现过程，分析期间推动活动的关键词就是发现和理解，这一阶段主要包括以下6个活动：

- 收集信息
- 定义系统需求
- 建立需求发现的原型
- 划分需求的优先级
- 产生并评价可选方案
- 与管理人员一起审查建议

分析阶段：SDLC的一个阶段，它的目标是了解并详述用户的需求。

收集信息是分析阶段的基本部分。在这一活动中，系统分析员接触用户以尽可能多地了解问题域。问题域是一个需要信息系统解决方案并正在被研究的用户业务领域。分析员通过观察用户工作过程，访问调查和询问用户问题，阅读有关过程的现有文件、业务规则和工作职责，评审现有的自动化系统来获取有关问题域的信息。除了收集系统的用户信息之外，分析员应该咨询其他感兴趣的人群，他们可包括中层管理人员、高级经理主管人员甚至有时是外面的客户。收集信息是发现和理解的核心活动。

问题域：用户的业务领域，为此正在开发一个系统。

然而简单地收集信息是远远不够的，分析员必须检查、分析获取的信息并将其结构化，以全面了解对新系统的需求。这一活动称为确定系统需求，其主要技术是通过绘制图表表达新系统的处理需求建立模型。

正如之前所讨论的，指导分析员收集和理解需求的一个重要活动是创建新系统的部分原型，然后供用户检查。用户经常会发现通过工作原型的可选方案能够较容易地表达他们的需求。在定义系统需求中具有普遍意义的说法是“一图值千字”，如果原型是一幅“图”，就能够从最终用户得到有价值的理解与启发。

随着处理需求的揭示，每一个需求都必须划分优先级。通常对自动化支持的要求大于对资金或资源提供的要求。因此，必须确定最重要的需求并要优先开发。分析员要对需求进

行优先级排序，也要为实施系统研究各种方案。实施方案包括建立内部系统、购买软件包或与第三方合作开发和安装一个新系统。

最后，要选择一个方案并推荐给高层管理人员，提供分析阶段各种活动结果的一个摘要说明，并与小组一起确定有关可选方案的决策。

2.2.3 设计阶段

设计阶段的目标是在分析阶段的需求和决策制定的基础上，设计解决方案系统。高级设计包括为软件程序、数据库、用户界面和操作环境的体系结构开发。低级设计需要制定详细的算法和程序开发所需的数据结构。在设计阶段必须完成以下7个主要活动：

- 设计和集成网络
- 设计应用结构
- 设计用户界面
- 设计系统界面
- 设计和集成数据库
- 设计细节的原型化
- 设计和集成系统控制

设计阶段：SDLC中设计系统和程序的阶段。

设计活动是互相紧密联系的，并且一般都具有实质性的重叠。

网络由计算机设备、网络线路和新信息系统中要使用的操作系统平台组成。当今的新系统在网络和客户/服务器环境下安装。设计活动包括构造这些网络环境。有时设计是基于现有的操作环境和战略IT计划的。在其他情况下，开发一个操作环境以提供新系统所需的服务水平的实质性工作是必须做的。

应用程序是新信息系统的一部分，它满足问题域中的用户需求。换句话说，应用程序为业务需求提供处理功能。设计应用程序包括采用分析阶段中为设计合适的计算机程序而开发的需求模型图。

应用程序：新信息系统的一部分，它能够满足问题域中的用户需求。

用户界面是任何新系统的重要部件。在分析活动期间，原型化活动已经定义了一些用户界面的元素。在设计阶段通常把这些元素结合起来产生一个具有表格、报表、屏幕和相互作用的序列的集成用户界面。

大多数新的信息系统也必须与其他现有系统通信，通信方法的设计和这些通信连接的细节必须准确地详细说明。这些称为系统界面。

数据库和信息文件是业务信息系统的一个集成部分。在分析阶段定义的新系统的信息需求的图表，用于设计支持部分新系统应用程序的数据库。有时，用于特定系统的数据库也必须与已在使用的其他系统的信息数据库集成起来。

在设计阶段，验证所提交的设计方案具有正确性和可行性通常是必要的。一个重要的验证方法是创建部分系统的工作原型，以确信它在操作环境下将正确发挥作用。此外，分析员能通过建立新系统的原型来测试和检验可选方案的设计策略。有时，如果原型正确建立的话，它们就能保留并作为最终系统的一部分使用。

最后，必须有效地控制每个系统，以保护数据库和应用程序的完整性。由于全球经济高度竞争，同时涉及技术与安全的风险，每个新系统必须有适当的机制以保护组织的信息和资产。这些控制应该在系统设计过程中，而不是在建成之后再集成到新的系统中。

2.2.4 实施阶段

实施阶段是建立、测试和安装最后的系统。这一阶段的目标是，不仅要有一个可靠、功能全面的信息系统，而且要确保所有的用户都受到培训，并使组织从中获得所期望的好处。所有之前的活动都集合在这一阶段以达成一个有效的操作系统。实施阶段由5个主要的活动组成：

- 构造软件部件
- 检验和测试
- 转换数据
- 培训用户和制作文档
- 安装系统

实施阶段：SDLC中对新系统编程和安装的阶段。

软件构造可以通过各种技术实现，常规的方法是用诸如Visual Basic或Java等计算机语言来编写计算机程序。基于开发工具和先前存在的组件进行软件构造的其他技术现在越来越受欢迎。软件也必须测试，第一种测试就是检验系统确实能工作，其他的测试就是要确保新系统能满足用户的需求。

在实施阶段，分析员也可以创建补充的原型，这些原型用于检验不同的实施策略和确保系统可以处理在其投入使用后可能存在的事务处理量。

几乎每个新系统都要替换现有系统，或是一个完全手动的系统，或是一个早期的自动系统。通常，现有系统的信息很重要，需要转换成新系统所要求的格式。转换数据经常变成项目自身的一个小部分，要对过程进行分析、设计、实施，以便整理数据并将其转换到新系统中。

只有用户理解并能恰当地使用系统，这个系统才是成功的，实施阶段一个重要的活动就是培训新系统的用户，以使该系统发挥最大的作用。

实际切换的转变是这个阶段最后的活动。新设备必须就位并发挥作用，新计算机程序必须安装和运行，数据库必须装入并可用，在使用新系统预期的用途之前，必须装上新系统的专用部分。在当今非常分散的组织里，系统常常在许多地方都必须安装并在整个组织内集成。

2.2.5 支持阶段

支持阶段的目标是在系统初始安装后的几年里保持系统有效的运行。支持阶段起始于新系统安装并投入使用后，持续整个新系统的有效使用周期。大多数业务系统期望系统要持续多年。在支持阶段，为扩大系统的能力，可能会执行一些升级或加强，将需要用到它们自己的开发项目。在支持阶段发生3个主要的活动：

- 维护系统
- 加强系统
- 支持用户

支持阶段：SDLC的一个阶段，其主要目标是保证系统安装后实现地运行。

每个系统，尤其是新系统，常常会有一些不能正确运行的部件。软件开发是复杂而困难的，所以它决不能随意出错。当然，良好组织并精心执行的系统开发的目标是提供一个稳定、完整并能提供正确结果的系统。可是，由于软件的复杂性，以及不可能对每一种可能的处理需求组合都进行测试，总有不能完全测试的情况，因此会有错误出现。此外，业务需求和用户需求随时会发生变化。维护系统的关键任务包括修复错误（众所周知的补丁）和为处理需求的微微调整两部分。通常，安排一个系统支持小组负责维护系统。

多数刚工作的程序分析员在其职业生涯开始之初都从事过系统维护的工作。任务最终完成还包括改变报告中提供的信息、增加数据库列表属性、改变Windows或浏览器形式的设计

几个方面。在这些工作分配下去之前，这些改变需要经过申请且被批准才可。因而每个改变的请求批准过程往往也是系统支持阶段的一部分。

在系统正常运行期间，进行大的调整是正常的。有时政府管理需要维护新的数据或提供新的信息，业务环境的改变——新的市场机遇、竞争或系统构造也需要对系统做大的调整。为了增强这些主要系统的功能，公司必须改进项目和启动升级开发项目。一个升级项目经常会产生新的系统版本。在你的职业生涯中，你可能会有机会参与几次升级项目。

支持阶段的其他主要活动是对系统用户提供帮助。通常采用由知识渊博的技术人员组成帮助台的方法，快速回答用户的问题并帮助他们提高工作效率。培训新用户和维护目前的支持是这一活动的重要工作。作为一个新的系统分析员，你可能会有机会去指导培训或充当帮助台的职员用以帮助你熟悉用户问题和需求。多数新工作的程序分析员在开始他们的职业生涯前，往往要花费一些时间在帮助台上工作。

帮助台：支持人员帮助用户解决涉及信息系统的任何的技术问题或数据处理问题。

2.3 方法、模型、工具和技术

系统分析员使用各种辅助工具来帮助他们完成SDLC中的活动与任务。这些辅助工具包括方法、模型、工具和技术。下面各小节将对其进行逐一讨论。

2.3.1 方法

为完成系统开发生命周期的每一步，系统开发方法提供一些指导方法，包括具体的模型、工具、技术。一些方法是公司内的系统专业人员根据自身经验提出来的，而另外一些方法则是从咨询公司或其他供应商处购买的。

系统开发方法：提供完成系统开发生命周期每一步的详细指导，包括具体的模型、工具和技术。

一些方法（自己开发的或者是购买的）拥有可以放满整个书架的书写文档。这些文档定义了开发人员在项目开发中所需要做的工作，如文档应该写成什么样子，管理报告应该包括哪些部分。其他的一些方法就不会这么正式了，通常在一份文档中包含了应该完成所有工作的大致描述。有时，公司采用的方法是“仅仅遵循某种方法”，但现在这样的自由选择正变得越来越少。然而，大多数人希望方法足够灵活，以便这种方法能适应于不同类型的项目和系统。

如果在一种方法中包含了模型、工具和技术的使用说明，那么你就必须懂得什么是模型、工具和技术。

2.3.2 模型

任何时候人们都需要记录或同现实世界中的某些事物交流一些信息，因此创建一个模型是非常有用的。在信息系统开发中使用的模型和其他所有模型有同样的目的。模型是现实世界中某些重要方面的表示。有时我们使用术语“抽象”来表示模型，因为我们从现实世界中抽象出的某些方面对我们特别重要。让我们来考虑一个飞机模型吧。在谈到飞机的空气动力特性时，有一个能二维展示其全面形状的小模型是非常有用的。有时能够展示飞机机翼横截面细节的制图是我们所需要的。在另外的情况下，也许有关飞机数学特性的列表对于理解飞机如何飞行是必要的。所有这些都是同一架飞机的模型。

模型：现实世界的某些重要方面的表示。

一些模型在外形上类似于真实产品，一些模型是重要细节的绘制表示，另一些模型则是抽象的数学符号。每一种模型强调一种不同类型的信息。在飞机设计中，飞机工程师使用大

量的不同类型的模型。成为一名飞机工程师需要学会创建和使用各种模型，这一点对于信息系统开发人员来说也一样。尽管信息系统模型并不像飞机模型那样标准或精确。现在，信息系统开发人员也正不断取得进步。首先，这还是个新兴领域，许多高级分析员都是自学成才的，认识到这一点是非常重要的。更重要的是，信息系统并不像飞机那样真实可感——你不能真正地看到、抓住或感觉到它。因此，信息系统模型显得更加无形。

开发人员要为信息系统的各个部分建立哪些模型？系统开发中使用的模型包括输入、输出、过程、数据、对象、对象之间的相互作用、位置、网络和设备，以及其他事物的表示。大多数模型是图形模型，包括使用公认的符号和惯例画一张表示图。这些模型通常称为图和表。可以使用流程图来表示程序的逻辑结构。本书的大部分将向读者展示如何理解和创建描绘信息系统的各种模型。

另一种对系统开发和应用都很重要的模型是项目规划模型。如在第3章中所描述的PERT或甘特图。这些模型是对系统开发项目自身的表示，其中突出显示了项目任务及其完成日期。与项目管理有关的另一个模型是一张显示了分派给项目的所有人的图表。图2-8列出了系统开发中使用的一些模型。



图2-8 系统开发使用的一些模型

2.3.3 工具

系统开发中的工具是帮助生成项目中所需模型或其他组件的软件支持。工具也许是创建图表的简单绘图程序。它们也许包括一些存储关于项目信息的数据库应用程序，如数据流定义或过程的书写描述。项目管理软件工具，如在第3章中所描述的微软项目，是用于生成模型的工具的另一个例子。项目管理工具为项目任务和任务相关性创建了一个模型。

工具：帮助生成项目中所需模型或其他组件的软件支持。

工具是为帮助系统开发者而专门设计的。程序员应该熟悉集成开发环境（IDE）。这个环境提供了许多工具帮助程序员进行编程，例如灵巧的编辑器、上下文相关帮助和调试工具。有些工具能为开发人员生成程序代码，有些工具则可以通过反向工程执行文件来获得程序代码，并可以根据代码生成模型，即使在开发人员将文档丢失（或没有生成文档）的情况下也能推断出程序的用途。

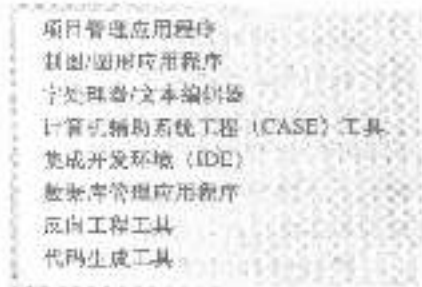


图2-9 系统开发中使用的一些工具

对系统开发人员来说，使用最广泛的工具是CASE工具。CASE表示计算机辅助系统工程。CASE工具将在本章稍后再进行详细讨论。通常情况下，CASE工具能够帮助分析员生成重要的系统模型，然后自动检查模型的完整性，以及该模型和系统其他模型的兼容性。最后，CASE工具可以根据模型生成程序代码。图2-9列出了系统开发中使用的工具类型。

CASE工具：用来帮助系统分析员完成系统开发任务的计算机辅助系统工程工具。

2.3.4 技术

系统开发中使用的技术是一组方法，这组方法可以帮助分析员完成系统开发活动或任务。它通常为创建模型提供逐步指导，或者为从系统用户处收集信息提供更一般的建议。常见的

实例包括：数据建模技术、软件测试技术、用户面谈技术和关系数据库设计技术。

技术：帮助分析员完成系统开发活动或任务的一组方法。

有时候一项技术可能适用于整个生命周期，这样一种技术包括为系统生成一些模型和其他的文档资料。现代结构化分析技术（我们将在以后进行讨论）就是一个例子。第1章讨论的战略系统计划技术和第3章讨论的项目管理技术也符合这样的定义。图2-10列出了在系统开发中经常使用的一些技术。

那么，如何把这些组件组合在一起呢？方法包括一组用来完成系统开发生命周期每一阶段活动的技术。这些活动包括完成各种模型以及其他文档和交付资料。与其他行业一样，系统开发人员使用软件工具来帮助他们完成这些活动。方法中各个组件之间的关系如图2-11所示。



图2-10　系统开发中使用的一些技术

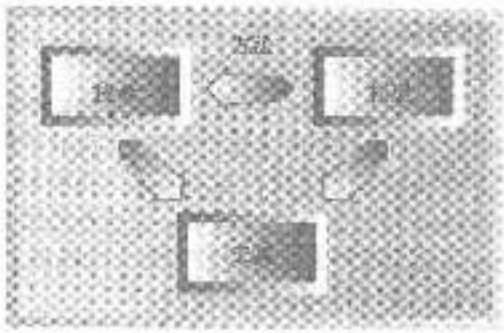


图2-11　方法中各组件之间的关系

作为落基山运动用品商店的新客户支持系统的项目经理，Barbara Halifax的部分职责就是确定用于开发系统的方法（见Barbara备忘录）。

2007年2月14日

To：John MacMurty

From：Barbara Halifax

RE：客户支持系统计划更新

John，这对短信只是告诉你我正为此项目组织安排和组织人员，并且需要确定在这项目中我们将使用什么开发方法。

在我们的方法中，我们可以自由选择如何安排各个阶段及活动。我打算使用迭代方法，并且计划与用户进行更广泛的交流。我也想过使用去年秋天我们讨论的螺旋模型中的一些概念，主要用于风险确认。我还想到从极限编程方法中引入一些概念，比如编程小组，但是这些要在项目的后期编码和测试开始后才用到。

是迫切的问题是开发方法的选择，到底是传统方法还是面向对象方法。我还是一直在和人们讨论这个问题，在分析的早期阶段，两种方法差不多，我想很明显，这两种方法都可以使用，仅仅因为我们有一个Web组件并不意味着我们一定要使用面向对象的方法。换个角度来讲，我们总在某种程度上涉及面向对象和UML方法，我一直在考虑这个项目的风险和回报，不管什么方法，我们都打算使用一个合适的CASE工具进行建模并生成一些代码。

就到这里吧，有什么问题请通知我。

BH

cc: Steven Deerfield, Ming Lee, Jack Garcia

2.4 系统开发的两种方法

系统开发可以使用多种不同的方法。当新工作的系统开发人员开始工作时，这种多样性会使他们感到困惑不解。有时，进行信息系统开发的每一个公司看起来都有自己的开发方法；有时，在同一个公司中的不同开发小组或个人也都可能有自己的系统开发方法。

然而，正如你在开始的例子中所看到的那样，系统开发中有许多通用的概念。事实上，在几乎所有的系统开发小组中，都使用系统开发生命周期的一些变体，这些变体同样包括项目规划、分析、设计、实施和支持阶段。另外，几乎每个开发小组都使用模型、工具和技术，这些模型、工具和技术组成了一个完整的系统开发方法。

所有的系统开发者都应该熟悉两种非常普通的系统开发方法，因为它们构成了几乎所有方法的基础。这两种方法分别是传统方法和面向对象方法。本部分回顾了这两种方法的主要特点，并提供了这两种方法的发展历史。

2.4.1 传统方法

传统方法包含了基于用结构化和模块化编程开发信息系统的技术的许多变体。这种方法通常称为结构化系统开发。结构化系统开发简称信息工程（IE），这是一个很流行的变体。

1. 结构化系统开发

结构化分析、结构化设计和结构化编程是组成结构化系统开发法的三种技术。有时把这三种技术一起称为结构化分析和设计技术（SADT）。20世纪60年代开发的结构化编程技术是人们第一次试图通过提供指导来改善计算机编程的质量。在第一次编程课中，肯定要学习有关结构化编程的基本原理。结构化设计技术发展于20世纪70年代，这种技术使得把分散的程序组合为更加复杂的信息系统成为可能。结构化分析技术发展于20世纪80年代早期，使得开发人员在设计程序之前就对计算机系统的需求了解得非常清楚。

结构化系统开发法：使用结构化编程、结构化分析和结构化设计技术的系统开发方法。

结构化编程 高质量的程序不仅在程序每次运行时都能产生正确的输出结果，而且使得其他程序员以后可以非常容易地阅读和修改程序，因为程序总是需要不断地修改。结构化编程具有一个开始和一个结束，并且程序执行过程中的每一步都是由三种程序结构组成的：

- 顺序程序语句
- 选择是由这一组语句来执行程序，还是由另一组语句来执行程序
- 循环语句

结构化编程：具有一个开始和一个结束的程序或程序模块，并且在程序执行中的每一步都由三个部分组成，即顺序、选择或循环结构。

结构化编程的三种结构如图2-12所示。

在开发这些规则以前，程序员在编程阶段才能确定编程技术，这就导致了程序结构非常混乱。如果程序可以运转，那么大多数的程序员会感到很高兴。如果程序能够产生正确的输出结果，那么他们就更高兴了。但是遵守这些简单的规则会使一个程序变得更容易阅读和解释。

与结构化编程有关的另一个概念是自顶向下程序设计，自顶向下程序设计把复杂的程序分解为程序模块的层次图（如图2-13所示）。在需要的时候，层次图顶层的模块通过“调用”底层模块来控制程序执行。有时，这些模块是同一程序的一个部分。例如，在COBOL中，一个主程序段通过使用“Perform”关键字来调用另一个程序段。在Visual BASIC中，事件过程中的语句可以调用一个通用过程。程序员使用结构化编程的规则（一个开始、一个结束和顺序、选择与循环结构）来编写每个程序模块（程序段或过程）。

自顶向下程序设计：把复杂的程序分解为程序模块的层次图。

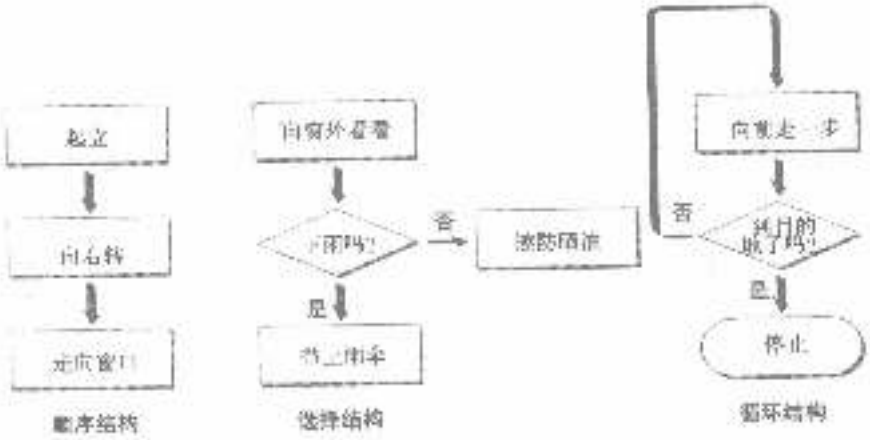


图2-12　结构化编程的三种结构

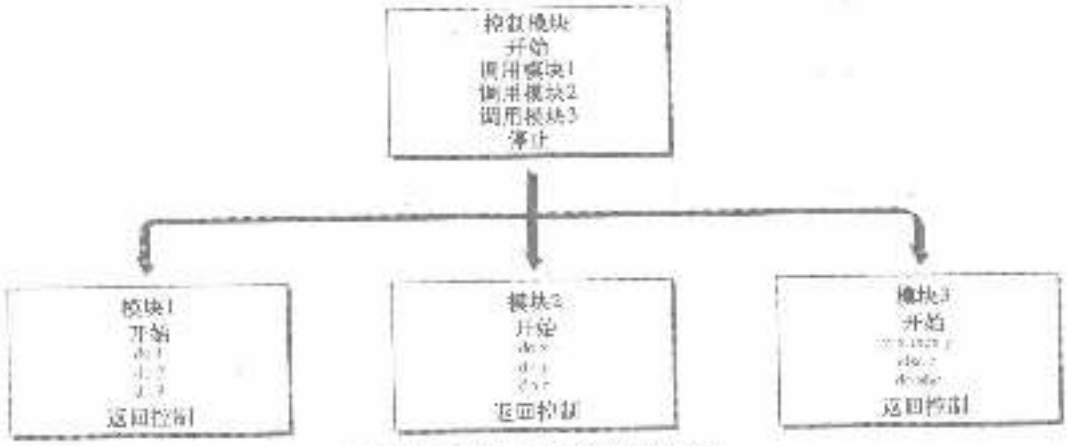


图2-13　自顶向下模块化程序设计

有时开发多个单独的程序，然后把这些程序组合为"系统"一起运行。这些程序的每一个都使用自顶向下的程序设计和结构化编程规则。但这些程序本身可以被组织成一个层次图，就像自顶向下程序设计一样。一个程序可以调用其他的程序，当层次图包括多个程序时，这样一种安排有时被称为模块化编程。

结构化设计　20世纪70年代以来，信息系统变得日益复杂，每一个系统都包含许多不同的功能，而且系统执行的每一个功能也许都是由数十个独立的程序所组成。结构化设计技术是用来为确定下列事物提供指导的：即程序集是什么、每一个程序应该实现哪些功能，以及如何把这些程序组织成一张层次图。模块及模块的安排可以使用一种叫做结构图的模型来图形化地表示（如图2-14所示）。

结构化设计：结构化设计是为确定某些事物提供指导的一项技术，这些事物包括程序集是什么、每一个程序应该实现哪些功能，以及如何把这些程序组织成一张层次图。

结构图：用结构化设计技术生成的显示程序模块层次的图形模型。

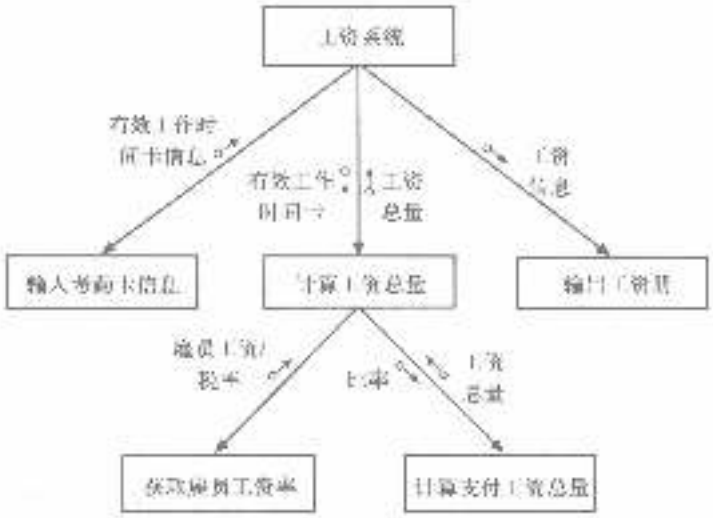


图2-14 使用结构化设计技术生成的结构图

结构化设计的两个基本原则是程序模块应该设计成耦合松散和高度内聚。松耦合意味着每一个模块应尽可能地与其他模块保持相对独立，这使得每一个模块在设计和以后修改时不会干扰其他模块的运行。高内聚意味着每一个模块实现一个清晰的任务。如此以来，很容易理解每一个模块实现哪些功能，并且可以确保如果以后需要对模块进行修改，那么不会由于意外的原因而影响其他模块。

结构化设计技术定义了不同程度的耦合和内聚，并且提供一种在程序真正编写之前对设计质量进行评价的方法，对于结构化编程来说，质量是根据以后出现新的需求时，以程序设计被理解和修改的容易程度来确定的。

结构化设计方法假设系统设计者知道系统需要做什么——主系统的功能有哪些、需求数据有哪些，以及需要的输出结果是什么。设计系统显然不仅仅是设计程序模块的组织结构。因此，结构化设计技术只是帮助系统设计者完成部分而不是全部系统设计生命周期阶段，认识到这一点非常重要。

到了20世纪80年代，文件和数据库设计技术也被用于结构化设计。更新的结构化设计技术假设在系统中使用数据库管理系统，并设计程序模块来和数据库交互。此外，由于越来越多的非技术人员涉及了信息系统，因此用户界面的设计技术也相应地得到了发展。例如，交互系统中的菜单决定了调用层次图中的哪一个程序。因此，用户界面设计的一个关键部分是与结构化设计一起完成的。

现代结构化分析 因为结构化设计技术要求系统设计人员熟知系统应该实现哪些功能，因此定义系统需求的技术获得了发展。系统需求详细地定义了系统必须实现的功能，但并没有规定实现这些功能的具体技术。通过推迟确定实现系统功能的具体技术，开发人员能够把他们的注意力集中放在需要系统做什么而不是如何做方面。如果事先不能充分而又清楚地计算出这些需求，设计人员不可能知道需要设计什么样的系统。

结构化分析技术 帮助开发人员定义系统需要做什么（处理需求），系统需要存储和使用哪些数据（数据需求），需要什么样的输入和输出，以及如何把这些功能结合在一起来完成任务。在结构化分析中使用的表示系统需求的主要图形模型是数据流图（DFD），它表明了系统的输入、处理、存储和输出，以及它们如何在一起协调工作（如图2-15所示）。

结构化分析：结构化分析是这样一项技术，它帮助开发人员定义系统需要做什么（处理需求），系统需要存储和使用哪些数据（数据需求），系统需要什么样的输入和输出，以及如何把这些功能结合在一起来完成任务。

数据流图（DFD）：显示在结构化分析中产生的系统的输入、处理、存储和输出的图形模型。

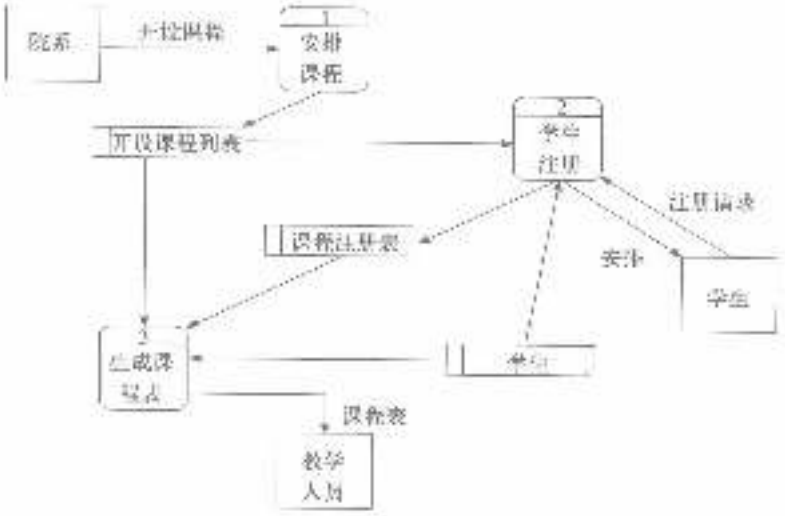


图2-15　使用结构化分析技术生成的数据流图（DFD）

通过识别引起系统以某种方式做出响应的所有事件，结构化分析的最新变体定义了系统的处理需求。例如，在一个订单录入系统中，如果客户订购了一件商品，那么订单录入系统必须处理一张新的订单（一种主要的系统活动）。每一个事件引起不同的系统活动。系统分析员获得这些活动的每一种，并且生成相应的数据流图来显示包括输入和输出的处理细节。

所需数据的模型也可以根据系统需要存储信息的事物类型来创建（数据实体）。例如，为了处理一张新的订单，系统需要知道客户、所需商品及订单的细节。这种模型被称为实体-联系图（ERD）。实体-联系图的数据实体对应于数据流图中的数据存储。图2-16所示为实体-联系图的一个例子。在使用结构化分析、结构化设计及结构化编程方法进行系统开发时的顺序，如图2-17所示。

实体-联系图（ERD）：系统所需数据的图形模型，其中包括在结构化分析和信息工程阶段生成的存储信息的事物，以及这些事物之间的关系。



图2-16　用结构化分析技术生成的实体-联系图（ERD）

结构化方法的缺点　由于系统开发的结构化方法已经发展了很长时间，因此在实际中能够发现结构化方法的很多变体。一些人仍然在使用他们在多年前学过的结构化分析和设计的最初版本，而不知道结构化方法的许多改进。另一些人只是在工作中学到一点结构化技术的皮毛，但从来没有真正地学习过细节问题。

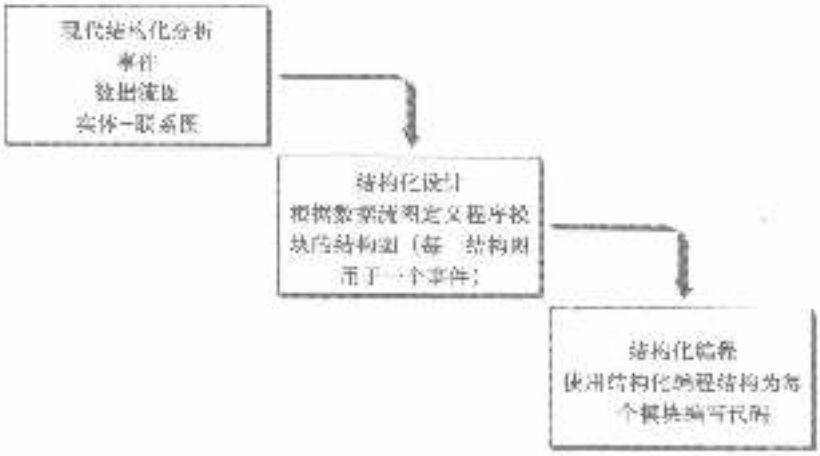


图2-17 结构化分析如何导致结构化设计和结构化编程

很多人认为结构化方法的功能很小，因为这种技术只能解决系统分析和设计活动的一部分而非全部的问题，批评该方法的人希望能够有一种更全面、更严格的技术，从而使得系统开发更像一门工程而不是一门艺术。另外，许多人认为在实际中并不能很好地完成从数据流图（在结构化分析阶段）到结构图（在结构化设计阶段）的转变。还有许多人认为数据建模和实体-联系图的建模过程更重要。尽管包括数据建模和数据库设计，结构化方法仍然是把过程而不是数据作为系统开发的中心环节。

最后，为了确保系统的全面、协调，许多人认为，只有在组织完成了全面的战略系统计划后，才可以启动系统开发。因此，他们想把战略性系统计划技术也包括在系统开发方法中。这种技术不仅要确定建立哪个系统，而且要提供一些确保所有系统兼容的最初的需求模型。为了实现这些目标，一些开发人员转而采用一种具体的结构化开发方法——信息工程方法。

2. 信息工程方法

信息工程方法是对结构化开发方法的一种改进。其第一步是制订一个全面的战略规划（应用程序结构计划）。它定义了组织经营其业务所需要的全部信息系统，这个计划还包括系统需要支持的业务功能和活动的定义。系统用来存储信息的数据实体及组织计划用来支持信息系统的技术基础设施。我们在第1章曾经描述过这种战略信息系统计划。

信息工程方法：传统的系统开发方法比结构化方法更严格、更全面，因为它关注战略规划、数据建模和自动化工具。

每一个新系统项目的第一步都是使用在战略系统计划期间创建的活动和数据实体。随着项目的不断进行，这些活动和数据得到了进一步细化。在每一步，项目组都要创建过程模型和数据模型以及这些模型的集成方式。

长期以来，用来经营业务的数据类型发生的变化很小，但是收集数据的过程却不断发生变化。因此，与结构化方法相比，信息工程方法更侧重于数据，正如结构化方法包括数据需求一样，信息工程也包括过程，信息工程的过程模型，即过程依赖图，类似于数据流图，但它主要强调过程之间的依赖关系，而很少关心数据的输入和输出。就像现代结构化分析一样，信息工程方法也是通过事件来触发处理过程的。

信息工程方法和结构化方法之间最主要的差异是：通过使用集成的CASE工具，信息工程可以提供更加完全的生命周期支持。CASE工具有助于最大程度地实现自动化设计。同时，它也迫使分析员忠实地遵循信息工程方法，尽管有时不得不牺牲灵活性。在许多情况下，分析

员可以通过CASE工具自动生成最终程序代码。CASE工具也可以用于结构化方法，但常常由于刻意追求灵活，而使系统开发不是很严格。

信息工程方法主要应归功于James Martin所做的工作，他写了几本关于信息工程的书，开发了CASE工具来支持信息工程方法。到了20世纪80年代末期，对于大型机系统，信息工程方法的使用已非常广泛。由于信息工程方法缺乏灵活性，因此对于较小的桌面应用程序和客户—服务器应用程序，人们很少使用支持信息工程的CASE工具。到了20世纪90年代，尽管信息工程的许多概念和技术仍然在继续使用，特别是计划方法和以数据建模为重点的方法，可还是很少有公司单独使用信息工程方法来开发系统。

信息工程方法对结构化方法的许多概念进行提炼，形成了一种更加严格、全面的方法。这两种方法都通过查看过程、数据，以及这两者之间的相互作用来定义信息系统需求、设计信息系统和构造信息系统。本书把这两种方法的关键概念合二为一，今后我们将其称为传统方法。尽管许多信息系统项目现在开始使用面向对象的技术——一种完全不同的方法，但在信息系统开发中仍然广泛使用某种版本的传统方法。

2.4.2　面向对象方法

一种完全不同的信息系统开发方法——面向对象方法，把信息系统看做是一起工作来完成某项任务的相互作用的对象集合（如图2-18所示）。在面向对象方法中，既没有过程和程序，也没有数据实体和文件，系统只是由对象组成。对象是一个在计算机系统中能对消息做出响应的事物。这种对计算机系统完全不同的看法要求使用一种不同的方法来进行系统分析、系统设计和编程。

面向对象方法：系统开发的一种方法，这种方法把信息系统看做是一起工作来完成某项任务的相互作用的对象的集合。

对象：计算机系统中可以对消息做出响应的事物。

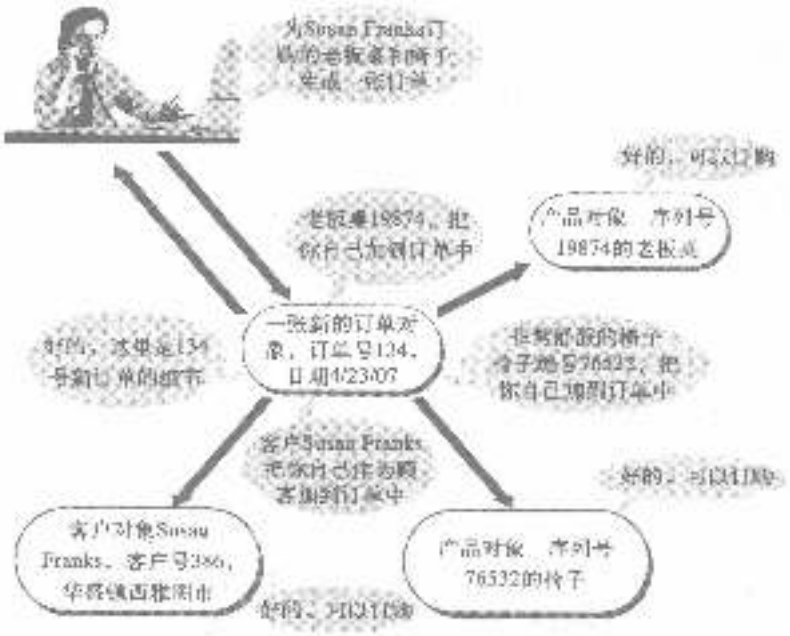


图2-18　面向对象的系统开发方法（从图中的用户如何计算看）

面向对象方法开始于20世纪60年代挪威Simula编程语言的开发，Simula语言用于创建包括像轮船、浮标和峡湾中的潮汐等“对象”的计算机模拟。编写模拟轮船运动的过程化程序是非常困难的，但一种新的编程方法简化了这个问题。在20世纪70年代，人们开发了Smalltalk语言用以解决创建包含诸如下拉式菜单、按钮、复选框和对话框等“对象”的图形用户界面的问题。其他的面向对象语言包括C++以及最近的Java、C#。这些语言集中于编写系统中所需对象的类型定义上，结果一个系统的所有部分都可以看成是对象，而不仅仅是图形用户界面本身。

因为面向对象方法把信息系统看做相互作用的对象集合，因此面向对象分析（OOA）意味着定义系统中所有的对象类型，并显示对象之间是如何通过相互作用来完成任务的。面向对象设计（OOD）意味着定义系统中的人和设备进行运信所必需的其他对象类型，同时细化每一种类型的对象定义以便用一种具体的语言或环境来实现它。面向对象编程（OOP）意味着用一种编程语言书写语句来定义每一类对象的行为。

面向对象分析（OOA）：定义在系统中工作的所有类型的对象，并显示这些对象如何通过相互作用来完成任务。

面向对象设计（OOD）：定义与系统中人和设备进行交互所必需的所有类型的对象，并对每一种类型的对象进行细化，以便可以用一种具体的语言或环境来实现这些对象。

面向对象编程（OOP）：用程序设计语言书写语句以定义各种类型对象的行为，包括对象之间相互传递的消息。

对象是事物的一种类型——可以是一个顾客或一个雇员，也可以是一个按钮或菜单。确定对象类型意味着对事物进行分类。一些诸如顾客这样的事物，既存在于系统之外（真实的顾客），又存在于系统之内（顾客在计算机的内部表示）。一个分类或类表示相似对象的集合。因此，面向对象的开发方法使用类图来表示系统中所有对象的类型（如图2-19所示）。对于每一个类，也许还有更具体的子类。例如，储蓄账户和支票账户是账户的两个具体类型（账户类的两个子类）。类似地，下拉式菜单和弹出式菜单是菜单的两个具体类型。子类体现或继承在其之上的类的特性。

类图：面向对象的开发方法使用的图来表示系统中所有对象类型的图像模型。

面向对象方法有几个主要的优点，其中包括自然性和复用性。对人而言，面向对象方法是自然的或者直观的，因为人们倾向于按照可感知的对象来思考世界，生过程编程语言中驰处可见的复杂过程是非常不自然的，同时，由于面向对象方法包括对象的类，并且组织中的许多系统使用同样的对象，因此无论何时需要，这些类都可以一次次重复使用。例如，几乎所有的系统都使用菜单、对话框、窗口和按钮，而同一公司中的许多系统也采用可重复使用的顾客类、产品类和发货清单类。因此人们不再需要为了创建一个新的对象而“重复发明车轮”。

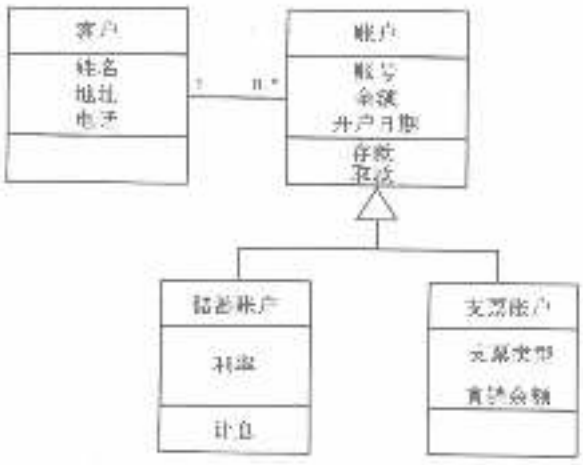


图2-19 在面向对象分析期间生成的类图

显然，面向对象方法和传统的方法有很大不同。但在其他方面，面向对象方法（OO）只

是把许多传统概念简单地重新包装了一下。这使得许多人在一开始很难理解OO方法。本书的第2部分和第3部分将详细讨论这两种方法的相似性和差异性，以阐明每种方法的优点。

如今开发的许多系统将传统方法和面向对象方法结合使用。一些集成开发环境（IDEs）也在同一个工具中结合了传统和面向对象技术。例如，面向对象编程用于用户界面，而过程化编程则用于其他部分。在同一个信息系统开发部门，甚至也有许多系统项目在分析和设计阶段只用传统方法，而其他系统项目则只用面向对象方法。这也是本书既包含传统方法又包含更新的面向对象方法的原因。每个人都应该知道这两种方法的基本概念，但在大学课程中或许只强调了其中的一种方法。

2.5　系统开发生命周期的变体

本书自始至终使用一种常见的系统开发生命周期（SDLC）。但需要强调指出的是，将来的系统分析员将会在组织中遇到许多正在使用的其他变体。本部分给出了变体的一些例子，以便无论遇到什么样的变体，你都会感到很熟悉并容易适应。一些变体基于生命周期阶段，而其他的一些变体则是基于允许的反复度。此外，一些生命周期不但在技术、子系统上有差异，而且对社会的影响也存在差异。

2.5.1　各阶段名称的变体

如今使用的生命周期的一些例子，如图2-20所示，它们在不同的阶段使用不同的名称。在工作中，你或许曾经遇到过这些变体的一种或几种。在瀑布项目中，阶段越多，严格的界限也就越多。第一个例子（早期的生命周期方法）在项目开始时包括可行性研究，它通常假设项目是无计划的，而不是将其看成整个战略系统计划的一部分。系统调查和系统分析一起组成了SDLC的分析阶段。第二个例子是信息工程生命周期。这个生命周期将整个战略规划作为其生命周期的一部分（本文的生命周期中仅包括该项目的计划，尽管通常任务战略规划在这之前就已经开始）。它还包括业务系统和技术设计两个设计阶段。最后，构造和转变是实施阶段的两个部分。



图2-20　在各个阶段具有不同名称的生命周期

第二种生命周期模型（来自于统一过程）仅包括4个阶段，其中几乎没有使用任何传统名

称。我们将在本章的后面部分对统一过程做更多介绍。第四种生命周期模型包括8个阶段，它使用一种与众不同的方法来命名各个阶段——用动词-名词的形式命名成类似活动的名称。本书使用的一般的SDLC模型包括5个阶段，并以一种传统的方式来命名各个阶段：项目规划、分析、设计、实现和支持。每个阶段包括用动词-名词的形式命名的活动：收集信息、定义需求、发现原型等。

无论使用哪种生命周期模型，都要完成相同的修饰任务。对于这些例子中的任何一个，其各个阶段是可以交迭的，并且各阶段的迭代可用于任何这些SDLC实例中。

2.5.2 以人为重点的变体

正在使用中的一些生命周期反映基于系统开发方法内在哲理的差异。一些方法更加清楚地认识信息系统是社会技术系统，也就是说，它们包括社会和技术两个子系统，这两个子系统必须一起考虑和设计以协调工作。因此，一些生命周期明确地强调这两个方面。社会技术系统开发的其他术语是以用户为中心并由用户参与的设计。这里有一个称为多视图的例子，与多视图一起使用的生命周期如图2-21所示。请注意，这里对人的活动进行了详细研究，并且系统的社会技术方面的分析和设计是在同一个阶段考虑的。人机界面的设计作为一个独立的阶段也强调用户和用户参与在系统开发中的重要性。最终用户通常会广泛地参与系统开发中的各个方面。



图2-21 多视图SDLC的各个阶段

社会技术系统：包括社会和技术这两个共同工作于系统的信息系统。

软件系统方法是另一种强调人的作用的方法。这种方法的支持者认为，在系统中对数据和过程进行模型对比较容易，但是要理解现实世界，就要求把人包括在模型中。人有不同的、甚至互相矛盾的目标、理解和看法，这使得系统行为变得不可预测。软件系统方法所创建的一个图表显示了人们和他们对于系统及彼此之间的看法和信念。当分析员花费时间对人的问题进行理解和建模的时候，人们就会暴露出复杂系统中的一些只有通过仔细的设计才可以找到的潜在问题。

2.5.3 基于开发速度的变体

系统开发人员一直在寻找一些方法来加速开发进程，加速系统开发的一个原因是对系统的大量需求，另一个原因就是不断变化的技术和业务环境。如果完成系统需要花费很长时间，那么将很难获得预期的效益。

一种用来从根本上加速开发进程的系统生命周期的变体称为快速应用程序开发（RAD）。RAD的一些变体试图加速每一阶段的活动，例如，通过安排关键人员参与高强度会议来收集信息和快速做出决定，从而加速分析阶段的进程。基于迭代的开发方法是另一种经常与RAD联系在一起的方法，它加速了进入设计和实施阶段的进程。

在SDLC的分析和设计阶段建立系统原型也能加速开发进程。但原型法并不总是只用于RAD。加深对系统需求的理解是原型法的另一个目标，在第4章，我们将把原型法作为一种信息收集技术加以讨论。

一些RAD方法甚至更极端。它们包括立即创建目标系统的工作原型，以便用户可以使用这些原型并对它们做出评价。一旦用户对系统需要做什么达成一致，那么原型就可以扩展为最终系统。如果不进行仔细管理，这些RAD方法可能会有一定风险。

2.6　系统开发的当前趋势

在信息系统领域有一件永远不变的事情，就是事情总是在变化。新的工具和技术总是在出现，有时是公开的，有时是可以预见的，并且系统开发者也总是试图寻找新的、更好的方法来工作。技术的更新以及前面所讨论的生命周期的变体就是系统开发方法正在变化的例子。这一小节将讨论系统开发中几个重要的发展趋势，其中任何一个趋势都有可能流行起来，甚至在未来的系统开发中占有支配地位。系统开发人员或许也可以在合适的情况下，从这些趋势中提取一些关键的概念或技术，与前面所讨论的变体结合使用。

2.6.1　统一过程

你已经了解到一些公司通过购买方法的版权或与咨询公司签订能够获取培训服务的合同的方式，从咨询公司获得完整的系统开发方法。统一过程（UP）是IBM的Rational软件公司提供的一种面向对象的系统开发方法，是统一模型语言（UML）的三个倡导者：Grady Booch，James Rumbaugh和Ivor Jacobson一直工作的公司。UP是他们定义一个完整的方法的一次尝试，这种方法除了提供几个独有的特征外，使用UML进行系统建模。在UP中，术语开发过程与开发方法是相同的。UP是SDLC的一个例子，处于预测和适应的中间。

统一过程（UP）：Rational软件公司提供的一种面向对象的系统开发方法。

你将学习很多关于UML的知识，因为UML是一种标准面向对象的建模符号，但UP不是标准的面向对象开发方法。本文中描述的UML模型可与面向对象的开发方法以各种方式一起使用，部是由于Booch，Rumbaugh和Jacobson的威望，UP正引起更多的关注。当然，UP包括了很多有用的新技术。Booch，Rumbaugh和Jacobson写了几本关于UP的书，并认可同事们所著的关于UP的书，因此，不从Rational公司购买服务也可能学习和使用UP。

UP被设计用来加强系统开发方法中常见的6个系统开发的"最优方法"：

- 迭代开发
- 定义并管理系统的需求
- 使用组件结构
- 创建可视化模型
- 核验质量
- 控制变化

UP定义了4个生命周期阶段：开始、细化(精细规划)、构造和提交（产品化阶段）（如图2-20所示）。开始阶段通过确定面向对象的开发方法中的用例（与用户需求类似）来定义项目的边界。你将在本书的第5章和第7章了解到如何确定用例，以及创建用例图。项目组也要完成可行性研究以确定是否要向该项目投资。

细化阶段集中在这几个迭代工作上：获得部分系统并定义需求，设计解决方案和实施解决方案。小组要通过创建用例图、类图、顺序图以及其他的UML图表来定义需求。最终的成本和效益评估也要在细节阶段结束时完成。

在构造阶段，通过使用附加的迭代过程来继续创建系统，此迭代过程也包括需求、设计、实施操作及创建多版本系统。在过渡阶段，你将系统提供给终端用户，并且集中进行终端用户的培训、安装和支持。

UP的4个阶段与传统的SDLC是不同的。因为这4个阶段并没有定义一般的分析、设计与实施阶段，相反，他们通过在某一点及时地提出项目组的重点所在来顺序地定义项目。为使迭代开发易于管理，UP在每个阶段都定义了规则，包括业务模型、需求模型、分析与设计、实现、测试、使用、配置和改变管理方式以及项目管理。每一次迭代也都包括了这些活动。UP还定义开发者所扮演的许多角色，以及项目进行过程中创建的许多模型。典型的角色包括设计者、用例说明人、系统分析员、实现人员和建筑师。

与任何其他方法相比，UP包含了关于系统开发中的每项活动要做什么、什么时候做等十分详细的信息。本书中给出的技术和模型与UP中所包含的许多技术和模型是一致的，但是并不集中介绍UP。在第16章将详细描述UP。

2.6.2 极限编程

极限编程（XP）是近些年来由Kent Beck倡导而兴起的一种系统开发方法。XP方法在很多技术的基础上添加了一些新的思想，有时候也被称为“轻型”系统开发方法，即指这种方法对于开发者来说既简单又能使开发更加有效。极限编程是SDLC中高度适应方法的一个例子。

开发者开始计划系统项目时，会让用户描述用户自身的经历。用户经历是对用户需要从系统得到的支持的描述。换句话说，即必需的系统功能。开发者用非正式的描述模型快速地将这些需求整理成文档。用户在提供用户需求的同时，还将描述一组验收测试。一旦系统完成，用这些测试证明系统已经提供了必要功能。

随后，开发者将为该项目规划发布一系列版本，随着增量开发的进行，每一个发布中都包含了最终系统的一个工作部分。项目从着手制作最初的发布版本开始，这通常需要进行几次迭代才能完成。第一个版本完成后，第二个版本随之开始。

在很多方面，XP方法很像另一种迭代和增量的方法。但是，XP包含了一些传统的特征，这一点使得XP方法能够普及。比如它需要连续测试、连续集成及大量的用户参与。它还要求所有的程序要由团队来完成。在编码和测试时，要有两个程序员在一个工作站上一起工作。这一点以及其他的特征强调了团队成员之间开放、有效的交流。最后一个特征就是坚信开发人员每周工作不应该超过40小时；一方面防止过度疲劳，另一方面还可以证明如果在项目中采用XP技术和工具，则可以按时完成而不需要开发人员过度劳累。XP方法在第16章中有详细介绍。

2.6.3 敏捷建模

许多系统开发人员喜欢轻型的XP开发方法，还有一些开发人员认识到了对诸如UP这样的综合开发方法的需求。到底使用哪种开发方法主要取决于项目本身的性质。许多开发人员利用XP为项目添加更多的模型和结构。也有许多开发人员采用UP来使开发过程流线化，从而简化开发过程。

目前最新的系统开发方法是由Scott Ambler推广起来的敏捷建模方法。利用这种方法可以鼓励开发者将最好的XP与最好的UP方法结合起来。该方法相对XP方法增加了大量的模型，同时减少了UP方法所拘泥的形式和所需要的文档。敏捷建模方法包括以下核心行为：

1. 迭代式与增量式建模，使用正确的模型，同时生成几个模型，并在小的增量范围建模。

2. 团队工作。与其他人一起建模，获得系统相关人员的积极参与，鼓励集体所有权并将模型公开化。

3. 简单化。创建简单的目录，简单地描述模型，并利用最简单的建模工具。

4. 验证。考虑可测试性，并用代码证明模型是正确的。

当继续学习本书内容时，你可以发现本书所提供的技术和模型，不论是传统的还是面向

对象的，都可以使用敏捷建模方法。

2.6.4 SCRUM

SCRUM是另外一种新的适应开发方法。术语Scrum指的是将扔出去的球拿回来的橄榄球系统。这个名称的由来是因为这种系统开发方法与这项运动有一定的相似性，即快速、自适应和自组织。Scrum背后的基础含义是尽可能快速主动的反映当前形势。

Scrum哲学也基于之前所描述的敏捷方法。Scrum反映了高速变化、动态的环境，用户不能确切知道需要什么，也可能频繁更换优先级。在这类环境中有很多的变化，使得项目可能陷入困境或无法完成。Scrum把重点放在开发团队和工作上，因此适用于这类情况。它强调个体多余过程，描述了在一系列短小项目中开发团队是如何一起工作来构建软件的。这个哲理的关键在于团队对自身组织和工作过程的全面控制。增量开发软件，控制要完成的事情。将在第16章详细描述Scrum。

2.7 支持系统开发的工具

不管使用哪种方法，重要的是尽可能地使用自动化工具来提高系统开发工作的速度和质量。这种工具是之前讨论过的CASE工具。CASE工具是为帮助系统分析员完成系统开发任务而设计的。分析员使用CASE工具创建系统的模型，其中有许多是图形模型。但是，CASE工具不仅仅是一个制图工具。

实践指导

尽可能的使用自动化工具，但是有一点需要牢记，对于一个小项目的小团队来说绘制简单的草图就足够了，不要让工具产生的问题比解决的问题多。■

2.7.1 CASE工具

CASE工具包含一个关于模型信息的数据库，称为资料档案库。资料档案库存储关于系统的信息，其中包括模型、描述，以及把各种模型连接在一起的引用。CASE工具可以检查模型，从而确保这些模型是完整的并遵循正确的制图规则。CASE工具也能对照检查两个模型以确保它们之间是一致的。如果考虑到分析员需要花费大量时间来创建、检查、修正模型，并要确保所有的这些模型能够有机地结合在一起，那么一个CASE工具能够提供多少帮助就是显而易见的了。资料档案库四周CASE工具的各种性能，如图2-22所示。如果系统信息存储在资料档案库中，那么开发小组成员就可以通过多种方法来使用它们。每次当一个小组的成员增加了一些关于系统的信息时，小组中任何其他人都可以立即使用该信息。

资料档案库：一个在CASE工具中存储系统信息的数据库，其中包括模型、描述以及把各种模型连接在一起的引用。

CASE工具通常分为上层CASE工具和下层CASE工具。上层CASE工具在分析和设计阶段为分析员提供诸如创建和检查模型，以及在资料档案库中存储系统信息等支持。下层CASE工具为实施提供支持，主要是根据资料档案库中的详细说明生成程序和数据库模式。为整个生命周期提供支持的CASE工具称为集成CASE，或ICASE工具。

一些CASE工具被设计得尽可能灵活，从而允许分析员使用任何想要的系统开发方法。其他CASE工具只是为非常具体的方法而设计的。本小节就介绍其中的一些工具——专门为某种方法设计的工具以及设计的比较灵活的工具。并不是所有的这些工具都被供应商称为CASE工具。对许多开发人员和管理者来说，一些CASE工具由于不能满足期望的要求而不再使用。如今的供应商把他们的工具称为可视化建模工具、集成应用开发工具或往返工程工具。

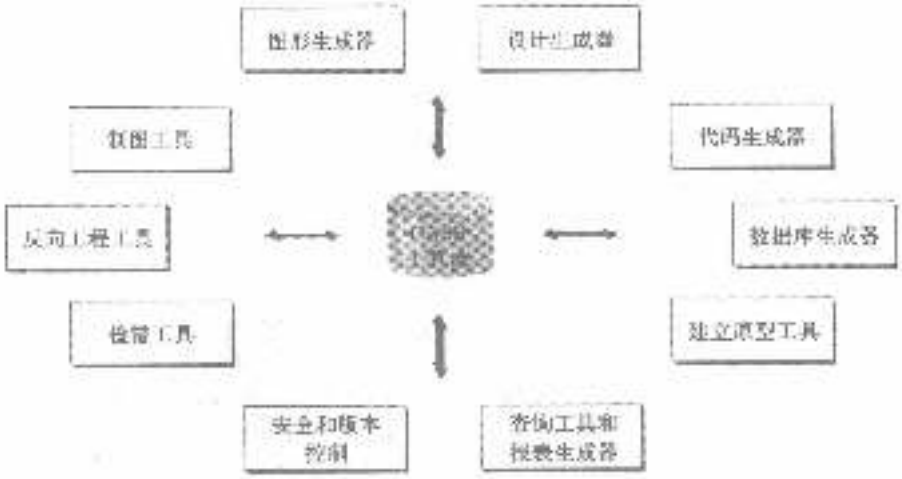


图2-22 包括系统所有信息的CASE工具库

2.7.2 Microsoft Visio

尽管Visio不是真正的 CASE工具，但Visio是一个绘图工具，系统分析员可以用它来创建他们所需要的任何系统模型。Visio带有一个绘图模板集，包含了用于各种业务和工程应用的符号。软件和系统开发模板提供了流程图、数据流图、实体-联系图、UML图，以及其他本书中能找到的图的符号。尽管模板提供了一个用于存储图表元素的定义和描述信息的有限资料库，但Visio并没有提供一个系统项目开发中的完整的资料库，尽管如此，许多系统开发人员还是喜欢Visio所提供的绘制必要图表的灵活性。

Visio软件界面如图2-23所示，从中可以看出Visio工具提供了两个窗口，一个窗口中有几个UML图：类图、用例图、流程图。这些图中的符号可以从左边的模板中选择。可以看出，左边的模板里列出并定义了图中的这些图标。

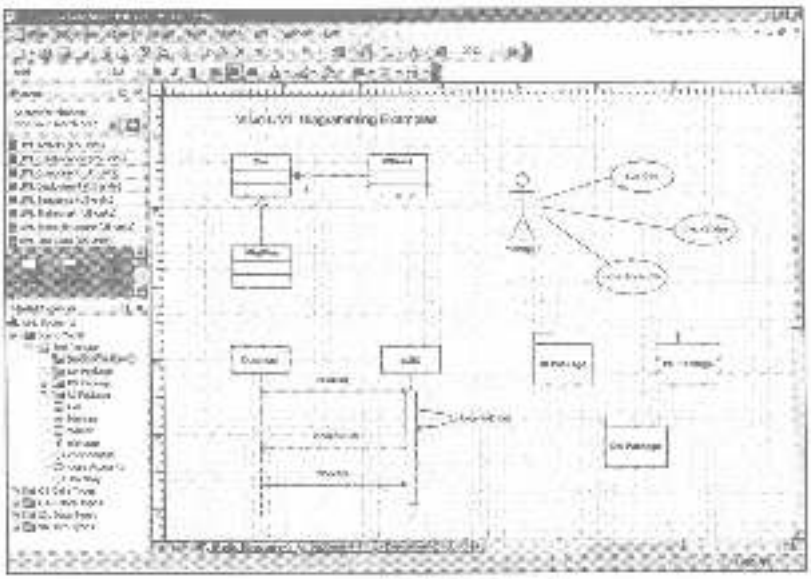


图2-23 Visio用于绘制各种图表

2.7.3 Visible Analyst

图2-24显示了一种由可视系统公司（www.visible.com）生产的称为Visible Analyst的灵活的CASE工具。这种工具使得绘制典型的传统模型，如数据流图和实体—联系图更加容易。同时也能支持面向对象UML模型。Visible Analyst包含了一个用于定义系统组件并提供错误检测和一致性检验支持的资料库。像许多供应商一样，可视系统公司现在已经停止使用CASE工具这个名称。Visible Analyst又在被称为集成应用程序开发工具。

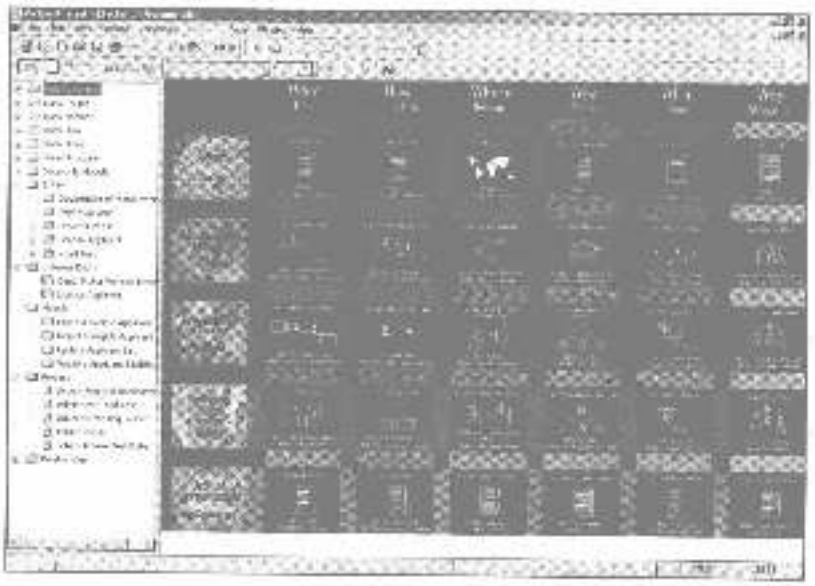


图2-24　Visible Analyst绘制的系统开发员可用的各类图表

2.7.4 Embarcadero Describe

一些具有往返工程特征的新产品正在不断地涌现。由于系统开发是非常反复的过程，尤其在面向对象方法中，因此对图形模型（例如类图）和生成程序代码进行同步操作是很重要的。比如，如果分析员改变了程序代码，那么类图也需要更新。同样地，如果类图改变了，那么程序代码也要更新。不像ICASE工具要从图形模型产生代码，较新的工具将自动地完成两个方向上（往返）的同步过程。如今已成为Borland公司（www.borland.com）的一部分的Peter Coad与TogetherSoft公司率先在它们的称为Together的工具中采用了往返工程。

Embarcadero Describe是使用往返工程的另外一个工具。图2-25显示了Together和一张类图以及同步的Java代码。如果开发人员喜欢书写代码来定义类，则类图会自动更新。如果开发人员喜欢先绘制类图，则定义类的代码会自动随之更新。

2.7.5 Rational XDE Professional

最新的系统建模工具将独立工具与资料档案库的所有特点，与生成并改进程序代码的其他供应商提供的集成开发环境（IDE）结合起来。如图2-26所示，Rational XDE Professional与Microsoft Visual .NET IDE集成起来，使得往返工程及代码生成成为开发者使用的IDE的一部分，开发者不再使用CASE工具及单独的IDE，所有开发者需要的功能性支持都集成在一个工具中。

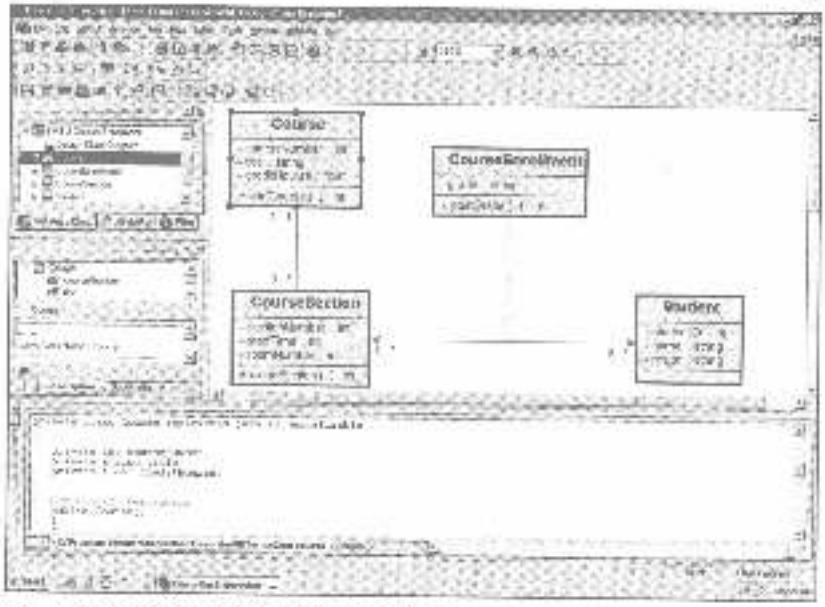


图2-25 具有可视化建模和往返工程功能的Embarcadero Describe Embarcadero技术公司

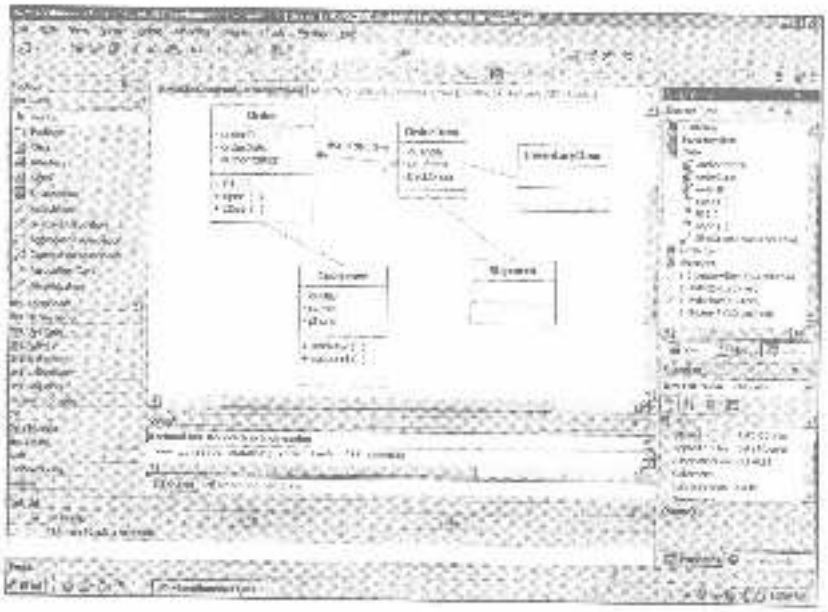


图2-26 Rational XDE Professional与Microsoft Visual .NET IDE集成

小结

系统开发项目是围绕着系统生命周期（SDLC）而组织的。同时SDLC的每个阶段都包括那些对于任何系统开发项目都必须完成的活动。SDLC阶段为：项目规划、分析、设计、实施及支持。系统开发者基于瀑布法顺序地学习SDLC阶段及其活动，然而事实上这些阶段是交迭的。项目本身包括很多分析、设计及实施活动的迭代。

开发信息系统有很多方法。所有的这些开发方法都使用系统开发生命周期（SDLC）来管理项目。这些方法加上模型、技术和工具就构成了系统开发方法学。系统开发方法学为严格完成SDLC中的每一步提供指导。实际上，有许多种不同的方法正在使用。大多数的系统开发方法基于两种信息系统开发方法之一，这两种方法分别是传统方法和面向对象方法。

SDLC的变体有很多种。最原始的生命周期是瀑布模型。这种模型在每一个阶段都有严格的开始点和结束点。如今大多数的变体在各个阶段之间使用交叉迭代。按照这种方法，首先进行一些分析，然后进行一些设计，此后再完成一些实施。这个过程不断迭代进行，从而改进系统。一些变体主要关心使用系统的人，称为用户参与或以用户为中心的设计。其他的变体则以加速系统开发为目标，称为快速应用程序开发（RAD）。

系统开发中当前的发展趋势主要包括统一过程（UP）、极限编程（XP）、敏捷建模和Scrum。这些方法以其对系统开发的创新的观点而越来越具有影响力。

计算机辅助系统工程（CASE）工具是一种为帮助分析员完成开发任务而设计的专门工具，包括建模和从模型直接生成程序代码。本章讨论了一些CASE工具的例子，这些工具现在分别称为集成应用开发工具、可视化建模工具，以及往返工程工具。

关键术语

adaptive approach	适应方法
application	应用程序
analysis phase	分析阶段
CASE tool	计算机辅助系统工程工具
class diagram	类图
data flow diagram(DFD)	数据流图
design phase	设计阶段
entity-relationship diagram(ERD)	实体—联系图
help desk	帮助台
implementation phase	实施阶段
incremental development	增量开发
information engineering	信息工程
iteration	迭代
model	模型
object	对象
object-oriented analysis(OOA)	面向对象分析
object-oriented approach	面向对象方法
object-oriented design(OOD)	面向对象设计
object-oriented programming(OOP)	面向对象编程
phases	阶段
planning phase	计划阶段
predictive approach	预测方法
problem domain	问题域
project	项目
prototype	原型
repository	资料档案库

sociotechnical systems	社会技术系统
spiral model	螺旋模型
structure chart	结构图
structured analysis	结构化分析
structured approach	结构化方法
structured design	结构化设计
structured program	结构化编程
support phase	支持阶段
system development methodology	系统开发方法
systems development life cycle(SDLC)	系统开发生命周期
technique	技术
tool	工具
top-down programming	自顶向下程序设计
Unified Process(UP)	统一过程
waterfall approach	瀑布法

复习题

1. SDLC的5个阶段是什么？
2. 应用SDLC预测方法有什么特点？应用SDLC适应方法有什么特点？
3. 基于问题解决方法的SDLC在第1章里是如何描述的？
4. 简单描述SDLC的各个阶段的目标是什么？
5. 迭代在各阶段间是如何应用的？
6. 模型和工具之间的区别是什么？
7. 技术和方法之间的区别是什么？
8. 在系统开发的两种方法中，哪一种方法是最早的？
9. 在系统开发的两种方法中，哪一种方法是最新的？
10. 在传统的开发方法中，哪一种方法集中于全面的战略系统计划？
11. 在传统的开发方法中，哪一种是更完整的方法？
12. 在结构化编程中所用的3种结构是什么？
13. 和结构化设计技术一起使用的是什么图形模型？
14. 和现代结构化分析技术一起使用的是什么图形模型？
15. 信息工程方法的核心是哪种模型？
16. 解释瀑布生命周期模型的含义。
17. 一遍又一遍地重复活动直到最终实现目标，指的是哪个概念？
18. 先完成系统的一部分，并在系统的其余部分继续完成之前将其投入运行，指的是哪个概念？
19. 什么是以用户为中心和用户参与的设计？
20. 快速应用程序开发（RAD）是什么？
21. 螺旋模型开发方法具有哪些特征？
22. 极限编程（XP）具有哪些特征？
23. 统一过程（UP）具有哪些特征？
24. 什么是CASE工具？为什么使用CASE工具？
25. 用来描述CASE工具的一些新的术语是什么？

思考题

1. 写出一页以内的文章，区别一下分析、设计和实施阶段基本目标的不同之处。
2. 描述一个有3个子系统的系统项目，并讨论该项目如何使用二次迭代？
3. 事实上，既然迭代过程几乎应用于所有的开发项目，为什么像课本一样顺序地介绍分析、设计阶段及活动还是有意义的？
4. 列写一些设计师设计的用来显示他们所设计房屋的不同方面的模型。解释一下为什么要使用多个模型。
5. 汽车设计师使用哪些模型来表示一辆汽车的不同方面？
6. 画出你家里的房间布局。现在写下对于你的房间布局的描述。这些都是你房间的布局模型吗？哪一个更准确、更详细、更容易使不熟悉你房间的人了解？
7. 描述一项用来帮助你完成“准时上课”活动的“技术”。与这项技术一起使用的是哪些“工具”？
8. 描述一项用来确保分配的工作准时完成的“技术”。与这项技术一起使用的是哪些工具？
9. 你使用的其他一些帮助你完成生活中活动的技术是什么？
10. 至少有两种系统开发方法、各种生命周期，以及一长串只能用于一些方法的技术和模型。考虑一下为什么会如此。讨论如下可能的原因，并指出哪个原因是最重要的：① 这个领域非常新；② 技术变化非常快；③ 不同的组织有不同的需求；④ 存在许多不同类型的系统；⑤ 开发系统的人在背景知识上存在巨大差异。

实验练习

1. 去校园安置办公室，然后收集一些在校园里招聘信息系统毕业生的公司信息。你能找到一些用来开发系统的方法信息吗？他们描述了SDLC吗？提到任何CASE工具了吗？访问该公司的网站，看看你是否能够找到更多的信息。
2. 访问一些主要的信息系统咨询公司的网站。试着找到他们用于开发系统的方法信息？他们描述了SDLC吗？提到任何CASE工具了吗？

实例研究

一个“完成大学教育”的方法

和许多本书的读者一样，你可能是一位攻读学位的在校大学生。把完成大学学业当成是一个项目——一个大的项目，持续很多年，并且为其花费的远远多于你的承受能力。一些学生在管理完成大学学业这个项目上比其他人做得更好。而许多学生（当然不会是你）却完全失败，而且大部分学生可能延期完成学业并且超出了预算（同样，当然不是你）。

像任何其他项目一样，为了获得成功，你应该遵循某些“完成大学教育”的方法，即你应该遵循完成从计划开始到成功完成等一系列活动和任务的准则。

1. 你的个人大学教育完成生命周期的各个阶段是什么？
2. 每个阶段的活动有哪些？
3. 有助于你完成这些活动的技术有哪些？在完成大学的过程中你可能会创建什么模型？区分你创建的那些使你完成大学的模型和那些有助于你计划和控制完成大学的过程的模型。
4. 有助于你创建这些模型的工具有哪些？

工厂系统开发项目

Sally Jones被派去管理一个新的系统开发项目，该项目将使Sally Jones公司的一些工作自动完成。需要完成哪些工作是非常清楚的：自动跟踪进行中的工作和已完成的货物清单。不太清楚的是自动化系统对工厂工人的影响。Sally有几个关心的问题：新系统对工人有何影响？需要对这些工人进行大量的培训吗？使用一个新系统将减慢他们的工作速度和干扰他们现在的工作方式吗？对新系统给车间必然带来的变化，工人们将如何接受？

同时，Sally认识到工厂的工人自己也许有一些好的想法：哪些技术可以继续工作，而哪些技术则不能，特别是那些更有可能在工厂环境中保留下来的技术。对于工人来说，哪一种用户界面工作得最好。尽管Sally确实懂得库存记账，但她对工厂操作未懂得不是很多。

1. 拟议中的系统是一个账户系统，还是一个工厂操作系统，还是两者兼而有之？
2. 从Sally使用的角度来说，哪一种生命周期变体是合适的？
3. 本章讨论的哪一个分析和设计活动不仅涉及工厂管理人员，而且还涉及工厂工人？

对落基山运动用品商店实例的再思考



Barbara Halifax写信给她的老板说明她仍在考虑客户支持系统开发项目的许多潜在的方法。她仍正在完成项目的计划阶段，因此到目前为止还没有过去多少时间。考虑如何RMO决定在这个项目中采用面向对象的方法，则需要对开发人员进行的培训。RMO需要对员工进行多么广泛的培训？需要进行什么类型的培训？仅仅是新的编程语言吗？或更广泛一些？在做决策之前项目可以进展到什么程度？

Barbara认为任意一个方法都可以使用，即使包括了一些基于Web的开发，也不一定必须使用面向对象的方法。你认为她正确吗？为什么？一些类型的项目需要面向对象的方法吗？

Barbara还提到她打算使用一些迭代并在项目中尽可能地加入用户的参与。应该考虑什么生命周期变体？她还能做些什么以加速开发过程呢？从统一过程、极限编程、敏捷建模或Scrum中她还能够得到什么启示？

关注Reliable Pharmaceutical Services



通过第1章的练习，相信你已经对Reliable公司的5年信息系统计划有了一些想法。管理人员很重视开发一个基于Web的应用程序，从而将客户机构与Reliable连接起来。但在Web组件实施之前，Reliable必须首先实现对患者、卫生保健机构、处方等基本信息处理的自动化。

然后，Reliable必须开发一个最初的提供信息的Web站点。该网站最终将发展成为一个内部网(Intranet)。Reliable将通过它与客户和供应商共享所拥有的信息和链接。该内部网的一个很重要的要求就是要与1996年的"健康保险可移植性与可说明性条例"，即著名的HIPAA保持一致。HIPAA要求卫生保健提供商及其合约商保证患者的信息在没有授权的情况下不能泄露。确保上述内部网与HIPAA保持一致就需要特别关注内部网的安全性。

一旦基本处理实现了自动化，内部网也构建完成，系统就可以从客户端添加患者信息并通过Web下订单。系统应该使Reliable及其客户端的处理流线化。此外，系统还应该提供有效的查询功能和患者管理能力，从而使Reliable的服务能够与其他竞争者区分开来，有自己的特色，如提供药物交互作用、药物过量警告、依据保险偿还政策进行处方的自动化验证，药物及病人花费信息及汇总等。

1. Reliable可采用的一种系统开发方法是利用SDLC中的瀑布模型来开始一个大的项目，包括详细计划该项目、分析全部需求、设计各个组件、最后实施整个系统。瀑布处完成系统各阶段，

采用这种方法的风险是什么？这种方法将要遇到什么计划和管理上的困难？

2. 另一种方法就是从最需要的部件开始，让它先用起来，然后，再开始其他项目完成其他特定的功能。采用这种方法的风险是什么？这种方法将要遇到什么计划和管理上的困难？

3. 系统开发的第三种方法是利用SDLC的迭代方法来定义一个大项目。简单描述每次迭代中需要包含哪些内容？描述增量开发怎样应用于该项目？与第一种方法相比，迭代方法如何降低了项目风险？与第二种方法相比呢？迭代方法可能给该项目增加哪些风险？

参考资料

一些经典的和较新的资料如下：

Scott W.Ambler, *Agile Modeling: Effective Practices for Extreme Programming and the Unified Process*. Wiley Computer Publishing,2002.

D.E.Avison and G.Fitzgerald, *Information Systems Development: Methodologies,Techniques and Tools (3rd ed.)*. Maidenhead, McGraw-Hill, 2003.

Kent Beck, *Extreme Programming Explained:Embrace Change*. Addison-Wesley Publishing Company, 2000.

Tom DeMarco, *Structured Analysis and System Specification*.Prentice Hall,1978.

C.Gane and T.Sarson, *Structured Systems Analysis:Tools and Techniques*. Prentice Hall,1979.

Ivar Jacobson et al., *Object-Oriented Software Engineering:A Use Case Driven Approach*, Addison-Wesley,1992.

Ivar Jacobson,Grady Booch,and James Rumbaugh,*The Rational Unified Process*.Addison-Wesley,1999.

James Martin, *Information Engineering:A Trilogy* (books 1,2,and 3).Prentice-Hall,1990.

Steve McConnell, *Rapid Development*.Microsoft Press,1996.

Meilir Page-Jones,*The Practical Guide to Structured System Design* (2nd ed.).Prentice Hall,1988.

John Satzinger, Robert Jackon, and Stephen Burd, *Object-Oriented Analysis and Design with the Unified Process*. Course Technology, 2005.

John Satzinger and Tore Orvik, *The Object-Oriented Approach: Concepts, System Development, and Modeling with UML* (2nd ed). Course Technology,2001.

Ed Yourdon,*Modern Structured Analysis*.Prentice Hall,1989.

第3章　项目经理级的分析员

学习目标

阅读本章后，应具备如下能力：

- 解释项目管理的要素和项目经理的职责
- 解释项目启动和项目规划与SDLC各阶段的活动
- 描述如何确定系统的作用域
- 用PERT和甘特图详细地说明一个项目进度表
- 详细地说明一个被提议项目的成本/收益分析并评价所提出项目的可行性
- 讨论如何组织人员并启动一个项目

本章要点

- 项目管理
- 项目启动和项目规划阶段
- 定义问题
- 产生项目进度表
- 确定项目可行性
- 为项目组织人员并启动项目
- RMO项目规划更新

蓝天共有基金家庭：管理IRA项目

Gary Johnson正在为明天的主管人员审查会报告做最后的修改。他是一个有3个系统分析员的小组的负责人。这个小组一个月以前就开始致力于设计一个新系统的开发工作并准备这个报告。这三个人密切合作，收集信息、做估算、准备启动这个项目前的报告和演示文稿。他想起了6个星期前当他第一次得到这个新任务时的情景。

Gary是一个项目经理，在这个项目开发成功之际并在交付使用前的最后测试阶段，他接受了调查一个新系统可行性的任务。在接到新的项目后，他用了一个星期的时间把他目前的管理职责移交给这个新的项目领导人。第二个星期他找了两个有经验的系统分析员与他一起工作。他几乎花了整整一个星期才协调好他们的时间安排，这样，他们就能在未来一个月里与他共同制订这个新项目规划。两个星期后，他和他的小组就能全力以赴投入工作了。

新系统的最初目标是为蓝天共有基金家庭支持退休账户投资的销售和管理。蓝天提供了许多类型的对公众的共有基金。包括对个人退休账户（IRA）的投资。IRA和Roth IRA是一种共有基金的特殊种类，具有十分严格的信息和税收报告需求。蓝天的部分战略规划是替换当前支持IRA的信息系统的一个项目，这个项目规划在6个月内开始。新的立法可能使蓝天提供一种新型的退休共有基金，这种基金资助大学教育：州教育资助计划和科维代尔教育资助存款。这些存款有严格的信息和税收报告的要求，所以它们看起来很适合在IRA基金管理范围内。为了保持其竞争地位，蓝天需要在明年一月开始提供这个新的IRA。对一个新IRA和教育资助计划的需求突然增加了，需要修改战略规划，立即开始开发工作。

尽管系统应该会被批准并由高级资方投资，Gary还是对项目的可行性进行了仔细评估。他首先采取的行动就是要精确地定义问题和准确确认项目的作用域。很明显，他不能估计项目的规模，除非他知道这个系统需要包含什么。所以，他做了一个新系统的商业利益表和一个新系统必须提供的功能详细说明表。这个称为系统能力的功能表帮助他估算开发系统的费用和预期的财务收入。然后，根据二人小组成员确定的可选系统所需系统性能和一些初步集体讨论结果，他开始为这个项目做一个计划和进度表。

Gary按照进度表工作，他分配一个分析员去确认这个项目的可行性并调查在新系统的开发和部署中可能会引起困难的潜在问题。另一个分析员开始计算费用和收益，以确保这个新系统是经济可行的。由于这些结果是相互依赖的，小组成员必须同心协力。

在最后几天里，他们把这些结果写入最终报告，并为主管人员准备演示文稿。在公司主管人员面前介绍总使Gary紧张，特别是对公司这样高度透明性和重要性的项目，更使他不安。尽管紧张，他还是充满信心，他和他的小组做了一项十分专业的工作。他已经开始为下一个任务计划，为共调配人员并进行更详细的系统需求调查。

概述

本书的第1章描述了业务环境。在当今高速发展且竞争激烈的全球经济中，人们对信息系统的需求永不满足。同时还定义了系统分析员的角色，以及他们在信息技术（IT）和IT战略规划中的作用。读者也了解了各种分析员所开发和支持的信息系统的类型。第2章中介绍了系统生命开发周期（SDLC），用来开发系统的方法模型，工具和技术，以及系统开发常用的一些途径。

在这一章，我们将重点讲解在公司内如何开发信息系统的详细情况。用落基山运动用品商店（RMO）客户支持系统项目作为一个特定的例子来讨论RMO项目SDLC的项目规划阶段。由于项目管理在信息系统开发和项目规划中的重要作用，故本章将介绍项目管理的基本原则。项目管理包括成功计划和管理新系统开发所必需的技能和技术。作为一个知识分子和问题解决者，你需要具备技术技能和管理技能，以成为系统开发小组的有用成员。这一章我们将介绍项目管理的基本原则，随后几章将进一步阐述有关项目各阶段的主要管理原则。项目管理包括一些在新系统开发时的，在其计划和管理中取得成功的一些必须的方法和技术。本书的目标是要阐述最新的概念、技术和在为当今组织开发新的信息系统中将使用到的技能。这一章将以实现这个目标为原则，继续介绍和解释系统开发生命周期的概念。系统开发生命周期是一个通用的术语，用于描述开发一个新的信息系统的方法和过程。

这一章的第二部分讨论信息系统项目如何启动。启动项目有两个主要原因，首先，因为新系统是整个战略规划的一部分，所以要启动开发新信息系统的项目。许多业务战略规划都包括一个信息系统部件用于支持业务规划，第二个原因是启动新信息系统项目可以响应最新的业务需求，由于公司内部的一些信息或处理问题，这样的需求通常增加了。

本章主要描述的是生命开发周期中项目规划阶段的活动。一个新项目的计划过程包括重要的几步，例如，定义项目的作用域、比较新系统的预计成本和预期的收益，以及制订项目进度表。这一节解释这些具体的步骤并讲授有关这些步骤的技能。项目管理、成本和利润分析及项目调度都是非常大的主题，故关于每个主题的更多信息在本书的最后都有相关的附录。你可以参考这些附录，以获取有关这些主题的更多的信息。

3.1　项目管理

尽管每个项目组都指派了项目经理来负责项目组工作，但所有有经验的成员都应该一起

参与管理。这部分主要是讨论从项目经理的观点进行项目管理，区分许多管理任务并分配给其他小组成员。RMO客户支持系统项目的经理是Barbara Halifax，她还配有一个系统分析员帮助她完成每一步任务。当项目进行时，所有成员都会被涉入项目管理中。

一个新软件系统的开发，现有系统的增强和升级甚至现有系统软件包的集成和部署，都要在开发项目期间完成。如第2章所讨论的，项目是一个有始、有终、有计划并产生预先确定的结果或产品的事业，而且总是会受到进度表和资源的限制。这个定义非常的宽泛，事实上，适合于在业务背景之内或之外的许多不同的活动。信息系统项目通常非常复杂，有很多的人和事必须组织和协调。不管目标是什么，每一个项目都是唯一的，没有两个完全相同的项目。生产不同的产品，不同的活动需要不同的进度表，需要使用不同的资源。这种唯一性使得很难控制项目，每一个项目都有新的活动，不可能按照以前几乎一样的方式进行。

3.1.1 项目成功因素

项目管理对系统开发项目的成功到底有多重要呢？1994年Standish Group发表了系统开发项目成功的研究结果。结果表明，几乎32%的开发项目在完成之前被取消了。此外，一半以上的计算机系统项目花费了差不多两倍于最初预算的费用。小于一半（大约42%）的项目具有最初预期的规模和功能。事实上，许多系统的完成仅满足了最初需求的一部分。根据公司规模，完全成功的项目（按时、按预算、具有所有功能的）只有9%～16%。到2000年项目开发成功率仍只达到28%，72%的项目可能被取消或延期、超预算、功能不完善。很明显，系统开发是一个十分困难的活动，需要十分仔细计划、控制和执行。

调查项目没有实现预期目标的原因是有意义的。项目失败或只有部分成功的一些主要原因如下：

- 系统需求不完整或发生变化
- 有限的用户参与
- 缺少行政支持
- 缺少技术支持
- 项目规划不够充分
- 目标不清楚
- 缺少所需资源

对成功项目的进一步研究有助于更加明确项目取得成功的原因。一些成功的原因如下：

- 清晰的系统需求定义
- 大量的用户参与
- 上层管理人员的支持
- 完整、详细的项目规划
- 符合实际的工作进度和里程碑

在大多数情况下，成功因素正好是这些失败因素的反向逆转。注意，诸如“技术太复杂”等的原因并没有出现在列表中，这表明项目失败最经常的原因就是项目管理的失败。成功的项目由好的项目管理产生，好的项目管理确保了进程管理是项目的一部分。

接着一个很明显的问题：“我们怎样才能改进项目成功率？”已经获得较多成功的公司从三个不同的角度解决这个问题。首先，把好的项目管理规则引入到项目中去。他们识别项目管理中最佳实践，然后用这些实践去培训项目经理。从本章中或者浏览整本书，你会学到一些好的项目管理规则。其次，采用系统开发方法论。如今的趋势表明通常迭代和适应方法有助于提进项目的成功。第2章介绍了通常采用的开发方法论。在本书中所有的方法论都是基于

概念和技术的。再次，成功的公司把注意力放在影响项目成功的因素上。组织关注于成功项目的制定特性，而且所有的团队成员和系统相关者都加入到最佳实践中。

3.1.2　项目经理角色

项目管理组织和指导其他人按照先前确定好的进度和预算实现计划结果。在项目的初始阶段，制定详细的计划，确定必须做的活动、必须完成的成果及所需的资源。因此，项目管理也可以定义为用来计划项目然后检测和控制项目的过程。

项目管理：组织和指导其他人按照事先确定的进度和预算实现计划的结果。

项目经理定义和执行项目管理任务。项目的成败与项目经理的技术和能力直接相关。前面所列出的项目成功因素为：清晰的需求定义、大量的参与用户、上层管理支持、完整的计划、务实的进度表及里程碑，这些通常是项目经理的职责。他/她必须确保把足够的注意力放在这些细节上。项目经理内外都要负责。

从团队内部角度看，项目经理是项目团队和所有活动的指导者和控制她。项目经理建立过团队结构完成工作。下面列出了部分内部职责：

- 识别项目任务和构建工作分解结构
- 制定项目进度表
- 招募和培训团队成员
- 安排团队成员任务
- 协调团队成员和次团队之间的活动
- 评价项目风险
- 监视和控制项目成果和里程碑
- 更改项目成果质量

从团队外部角度看，项目经理是项目的焦点或主要联系人。他/她必须把团队推广到外面去交流他们的需求。一些主要的外部职责包括：

- 报告项目的状态和进展
- 与识别所需系统需求人员(使用系统的人)建立良好的工作关系
- 直接与客户(项目赞助者)、其他系统相关者工作
- 识别资源需求和获取资源

很明显，项目经理不能完成包括这些职责的所有任务，需要其他的团队成员协助项目经理完成。然而，项目的主要职责还是由项目经理完成。

通过观察世界的各类组织及开发项目所处的方式，我们发现项目经理的角色和项目管理的职业变化非常大。图3-1列出了项目经理的不同职位。在一些公司，项目经理扮演协调者的功能，但没有直线(申报)职权。此外，对于大的开发项目，项目经理可能是一个既具有管理技能又非常了解技术的富有经验的开发员。在那些情况下，项目经理对于其他的成员有直线的职责和权威。

许多职业道路导致了项目管理的产生。在一些公司，项目协调角色由大学毕业生担当。其他公司通过较强的组织和人际能力来判断个人价值。这里指的是那些能够理解技术的人，而不是希望得到一份高端技术职业的人。那些公司给员工提供获取管理和业务技能经验的机会，从小项目协调者的经验提升到项目管理的经验。其他公司把“首席工程师”方法应用到项目管理中，要求每个人必须全面掌握技术用以管理项目。这些公司认为，项目管理要求员工有很强的开发能力，从而理解技术问题和管理其他的开发人员。

头 衔	能力/权威	组织结构	职 责
项目协调员或项目领导	受限制的	项目在部门内进行，或者项目可能有一个强大的"领导开发者"，它可以控制较低产品的开发	开发计划，协调活动，让人们了解状态和进度。对项目的产出没有"直线"权利
项目经理、项目管理或主长	适度的	项目在IT部门进行，而其他的业务功能是独立的	所有项目管理职责加有一些技术职责。管理项目大多是人型的，可能与客户共享项目职责
项目经理或者部门管理员	很高的	项目机构是一个公司的主要高层部分，公司由各类项目或者一个大型权威的IT部门组织起来	通常在技术和项目管理方面有广泛的经验，包括在管理决策和技术问题方面。经常有支持团队文书工作，管理项目可以很大起来

图3-1 项目经理的各种角色

3.1.3 用SDLC管理项目

一个项目经理同时要与几个团体一起工作。客户是将要对新系统开发进行投资的个人或团体，也就是顾客。所以，当我们谈到项目批准或资金投入，都是指客户。对于内部开发，客户可以是一个执行委员会或是一个投资项目的专门副总裁，对于大型的、至关重要的项目，可以成立一个监督委员会（有时称筹划指导委员会），这个委员会由客户和其他关键的高级管理人员组成，这些高级管理人员具有该组织战略方向的远见卓识，他们强烈希望项目能够成功。具有指导大型开发项目的经验是有益的，但这对委员会成员不是一个先决条件。另一方面，用户是实际使用新系统的人。在某种情况下，客户和用户是相同的人。但更多情况下，他们是不同的。用户提供有关新系统需求的详细功能和操作的信息，客户提供业务结构和战略方面的信息。这些是影响系统规模和设计的重要因素。此外，客户批准和监督项目的运行并提供资金。图3-2描述了参与项目开发的各类人员。

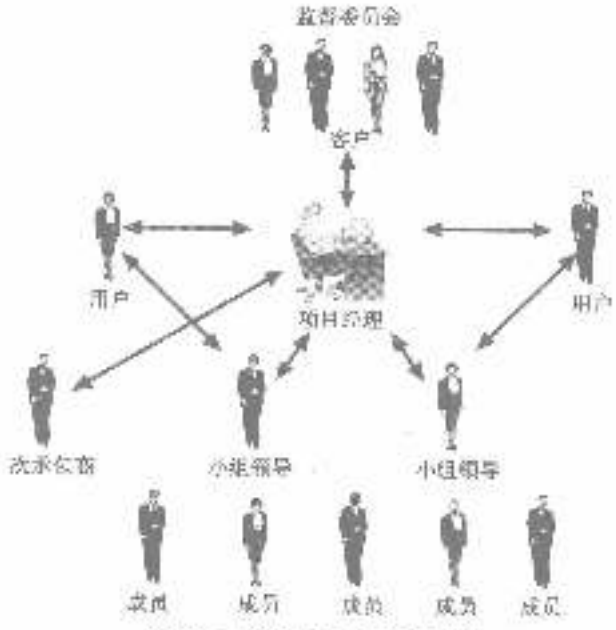


图3-2 系统开发项目的参加人员

客户：投资项目的个人或团体。

监督委员会：检查和指导项目的客户和主要经理。

用户：使用新系统的个人或团体。

与客户、监督委员会之间的交流是项目经理外部职责的重要组成部分。类似地，和组长、组员和分包商一起工作是项目经理内部职责的一部分。一些用户在项目中十分积极，可以作为项目组的一部分。而其他的用户仅仅是业余参与者，不管怎样，项目经理必须确保内部和外部交流适当地进行。

在第2章中，你已经了解了系统开发生命周期的两个类型——预测项目和适应项目。很显

然，所有的项目都需要项目管理，而且对于任一类型的项目，项目经理的职责是一样的。然而，特定项目管理任务的应用取决于所使用的SDLC方法。

了解项目管理任务和项目开发任务的差异是十分重要的。项目管理的定义包括经理指挥其他人完成已计划成果的概念，而项目开发任务是直接联系到新系统的"动手做"任务。为了更好的理解这一区别，我们可以用软件开发项目中的项目管理和建造项目的监控任务相比较。与建筑师一起工作的建筑施工经理根据进度表阅读计划，核查进度表，以及给组员安排工作。这些任务是监控任务，不同于搭建混凝土或砌砖这些"动手做"活动。通常情况下，施工经理不会搭建混凝土或砌砖。他/她忙于通过安排工作，核查进度和解决问题来对项目进行协调。除非软件项目非常小，否则需要把全部经历放在项目管理任务上。

图3-3是图2-5的改编，显示了预测开发项目中各个阶段的重叠。注意到图3-3中项目规划阶段既包括项目管理也包括SDLC任务。发生了重叠是由为计划项目需要主要组员和项目经理的参与。软件开发项目的复杂性要求组员和模态加到识别活动，评估工作要求和构建项目进度表中去。制定详细的计划之后，组员把精力放在SDLC任务上，而项目经理把精力放在管理任务上。



图3-3　预测项目的项目管理和SDLC任务

如图所示，有三个主要的项目管理过程重叠了SDLC过程：执行、控制和结束项目。执行包括以下任务：依照进度表、安排和协调项目组工作，以及与所有项目系统相关者交流。控制任务包括确定进度，必要时采取正确行动，评估作用域变化是否必要，维护未定事件清单及解决问题。收尾项目包括项目的无缝关闭，例如解散其他任务的组员，归结预算和支出，回顾或审计项目结果。图中所显示的重要的一点是这些项目管理任务贯穿这个项目，并与分析、设计和实施相联系的SDLC活动同时发生。

图3-4显示了项目管理任务是如何在适应项目中应用的。图中采用了螺旋生命周期概念（如之前的图2-6所示）和所描述的线性两次迭代（类似于图2-7所示的迭代）。尽管适应方法有许多变化，图3-4表示了普遍适用于所有适应方法的关键思想。一个项目总是由一个主要的计划成员开始。然而，在适应方法中有两个计划焦点：宽泛地说，第一个目标是定义项目的作用域；第二个目标是识别迭代或周期。从这一点上说计划项目是不够详细的。虽然可以确定主要的步骤但是为之后遗留下了细节问题。

在完成主要的计划阶段之后，需要进行详细的迭代，即周期。在一些适应方法中，主要的计划阶段也可以视为是一次迭代。每一个周期都需要详细的计划、执行管理、控制管理和周期收尾。每一个周期还需要用来分析、设计和实施的SDLC活动。当然，SDLC任务的确切特性取决于客户从迭代中获取的成果类型。

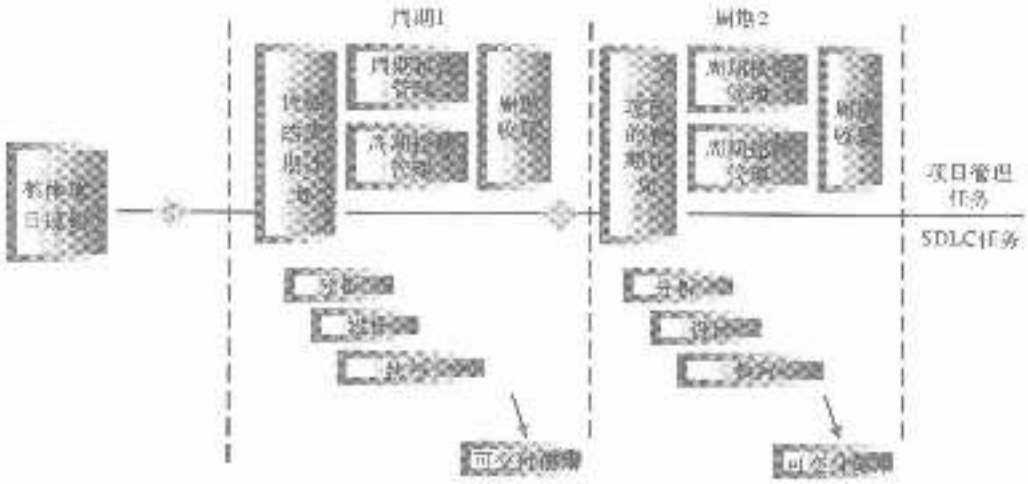


图3-1 适应项目的项目管理和SDLC任务

适应方案的项目管理和预测项目一样复杂和重要。通过比较这两种方法，我们发现在适应项目中，计划任务更加广泛地分布在整个项目生命中。然而，需要一样的项目管理技能。唯一的不同就是如何及何时对执行的任务进行计划、执行和控制。在下一部分，我们将讨论作为一位好的项目经理所需的知识领域。本章的后部分将详述你所需的特定的项目规划技能，不管你是否是项目团队的项目经理还是项目分析师。

3.1.4 项目管理知识领域

项目管理协会（PMI）是推进项目管理的专业组织，主要分布在美国，也分布在世界各地。此外，其他国家的专业组织也在推进项目管理。如果你对增强项目管理技能有兴趣的话，应该考虑加入其中一个组织，获取他们的资料并参加培训。PMI是权威严格的认证程序。事实上，许多公司鼓励项目经理去拿PMI认证，而如今，工业条款频繁表示项目管理是一项十分重要的技能。

作为其中的一部分任务，PMI已经定义为项目管理的知识主体（BOK）。知识主体就是PMBOK，是广泛接受的信息基础，每个项目经理都应该了解。PMBOK已经编纂为9个不同的知识领域。虽然这9个领域不能代表完全理解项目管理，但提供了极佳的基础。

- 项目规模管理：定义和控制需要包含在系统中的功能及项目组要做的工作的范围。
- 项目时间管理：建立一个所有项目任务的详细进度表，然后根据确定的里程碑监控项目进程。
- 项目成本管理：计算初始的成本/收益分析，以及之后的更新和作为项目进程监控的支出费用。
- 项目质量管理：为确保质量建立一个总的计划，包括项目的每一个阶段的质量控制活动。
- 项目人力资源管理：补充和雇佣项目小组成员，也包括培训、激励小组成员，小组建立和举行相关活动，以保证小组成员工作愉快，有活力。
- 项目通信管理：确定所有系统相关人员和小组之间的主要通信，建立所有的通信机制和进度表。
- 项目风险管理：确定和检查整个项目所有潜在的失败风险并制订减少这些风险的计划。

• 项目获取管理：制订提案请求、评价投标、签订合同和监控供应商服务性能。

• 项目集成管理：集成所有其他知识领域为一个无缝的整体。

附录A详细讨论了这些知识领域的每一个方面。其他的书只把重点放在项目管理上，其中的一些书列在章末的“更多资源”部分，并提供了应用在每个知识领域中详细的讨论和专业技术。

随着工作的进步，你会发现记录从别人那里观察到的项目管理技能，以及通过你的工作经历学到的项目管理技能是明智的。在第1章描述系统分析员技能时就讲到这一点。可是，好的项目管理需要更多的技能。一个好的项目经理知道如何制定计划、执行计划、参与问题和随变化做出调整。项目管理技能是可以学习的，立志管理项目的人会主动自我充实，学习必需技能，可以将从本书所学到的知识付诸实践，这样即可磨练项目管理技能。

3.2　项目启动与计划阶段

项目启动有各种理由，通常3个主要的驱动理由如下：① 应对机会；② 解决问题；③ 依遵指示。

大多数公司总在不断寻找方法以增加他们的市场份额或打开新的市场。他们创造机会的一种方法是通过短期和长期的决策计划。在许多方法中，计划是确定新项目的一种优选方法。这种方法的好处是它为开发新系统提供了一个更稳定一致的环境。随着决策计划的制订、项目确认、区分优先次序，在整个计划内安排进度也将逐步展开。以这种方式启动的项目有时称为自顶向下的项目。

为了区分这些项目的优先次序，许多公司采用一种称为权重得分的技术。首先，由IT决策规划委员会确定一组评判新项目重要性的标准。例如，以“开发一个新市场”或“提供高净现值”来做评判标准，这些标准根据他们的重要性给以权重。每一个潜在项目都根据这组标准鉴定等级，最高分的项目优先启动。

权重得分：根据标准以不同的权重区分项目优先次序的一种方法。

项目也会因解决一个突发的业务问题而启动。这些项目要设法减少正确运行业务活动所需信息的处理和目前正使用的系统之间的差距。他们可以作为战略规划中的一部分启动。但更常见的做法是由中层管理人员为解决公司在运行中遇到的一些困难而提出。很明显，高级经理主管人员也知道内部问题，并启动项目解决这些问题。有时这些问题很严重，以致引起战略规划委员会的注意，把它们纳入整个业务战略中。有时一个突发的需求必须立即解决。如新的销售代理计划或生产方评估报告。在这种情况下，个别经理将请求个别开发项目的启动。

最后，为响应外界的动向而启动项目。常见的外界压力是立法的变化，如税法和劳动法的变化。这就需要新的信息收集和外界需求报告。例如，健康保险携带和责任法案用于保障病人的医疗信息安全。这个立法影响医药安全服务系统，比例了在第2章最后一部分已讨论过。立法的变化也能扩大或缩小一个组织能够在市场上提供的产品和服务范围，新的规则和法律能影响战略规划，导致加速新系统的需求。我们已经看到在电信行业许多规则的变化。如有线电视和电话公司为竞争提供蜂窝技术服务的机会而引起的变化。Internet访问和个性化娱乐的变化。

随着新项目的启动，一些阶段也正常运行了。一本典型的项目书描述了新系统的目的，以及可能开始和完成的日期。还有十分重要的一点，即新系统的关键股东和主办人。

不管什么原因启动了项目，它总是需要一个初步的调查以确信这一项目收益超过开发的成本和风险。因此，几乎每一个项目在批准后的首要活动都是精确地确定业务问题、确定项

目的范围，完成包括一个成本/收益分析的可行性分析。我们把所有这些初始计划活动都列入SDLC的项目规划阶段。

实践指导

无论新项目的什么背景，必须在执行之前对其评估可行性。 ■

3.2.1 启动落基山运动用品商店的客户支持系统

如第1章所述，RMO的高级经理主管人员在助外界咨询公司已经为公司的信息技术系统制订了声誉很好的信息系统策略规划。这个规划包括技术结构部件和应用程序结构部件。此规划的实施已经以供应链管理（SCM）项目的启动而开始。Blankens太妇清楚地知道，为了保持良好的客户关系，随着他们的业务遍布广阔地域和基于互联网的销售的开展，他们需要一个系统来支持销售的完成。在客户支持系统（CSS）也在线投用之前，公司不会认识到SCM系统的所有好处。虽然SCM系统取得了几次成本下降的效果，但RMO还是期望通过新CSS系统扩大销售和市场方面的能力得到来自于迅速增长销量的真正的业务利润。

供应链管理系统进展顺利，因为RMO的几个供应商也必须更新他们的系统，所以，这个项目需要几个阶段。第一个阶段按时进行，需求已经确定。整个结构设计固定下来，一部分新系统预期下个年度的早期完成。John Blankens对此进展感到十分兴奋，期盼新客户支持系统（CSS）的启动。他召开公司执行委员会的专门会议评价目前项目的进展和启动新CSS的可能性。会前，他要求财务副总裁JoAnn White带来详细的目前系统预算的财务分析和RMO期盼的不久将来要启动的CSS的财务影响的预测，他也邀请了Mac Preston评价系统开发人员的工作量和员工工作的有效性，为仔细考虑他的建议。其他几个任务分给委员会成员。

通过长时间的讨论，执行委员会决定现在启动这个项目不仅是切实可行的，而且必须要这样做。其他零售商已经证实如果正确计划和执行，基于互联网的销售及其市场能为公司提供巨大的利润。尽管还有些不成熟，电子商务延缓。但为了以后的生存，RMO也应该像其他有营业场所的零售商一样，在互联网上进行强劲的业务是势在必行的。

会议的一个结论是委员会指定Mac启动这个项目。首先，他与系统开发主管John MacMurty会谈，要求他最后确定一个项目经理和另外有经验的系统分析员开始启动项目。他也要求John制订项目合同以确保执行委员会做出的决策的有效性。John开始与经理主管人员接触，动员他们作为监督委员会成员参与工作。他清楚地懂得如果他得到公司高级经理主管人员的有力承诺，那么好的用户就会进入项目。项目成功的关键因素之一是用户的广泛参与。在与全公司经理主管人员两天的讨论之后，监督委员会成立了。由于此系统直接支持William McDougal的部门而几他是项目发起人，所以他担任委员会主席。其他成员是产品目录销售主任Robert Schneider，业务部的Brian Haddock，当然还有John Blankens和Mac Preston。

项目以任命Barbara Halifax为专职项目经理开始启动。正如第1章中的RMO备忘录提到的一样，Barbara已经为RMO工作多年，在加入RMO之前，她在一个大型会计公司的信息系统咨询部工作。咨询方面的经历使她对许多不同公司和系统有着广泛的了解，RMO的高级管理人员对她管理CSS项目的能力完全信任。高级系统分析员Steven Deerfield也被分到这个项目，Deerfield和Halifax以前一起工作过，工作模式非常融洽。由于这个项目是RMO长远战略规划中的一个重要部分，所以为其指派了两个公司中最好的系统分析员，项目合同如图3-5所示，它证实初期使项目启动的活动。

项目名称：客户支持系统

项目目的：提高客户支持水平，应该包括从订单输入到出货接达的所有有关客户功能，包括客户询问/产品目录、订单输入、订单跟踪、运输、退回订单、退货和销售分析。

预期完成：项目启动18个月内

批准预算：$1,500,000

主要参加者：

参加者	职位	主要职责
Barbara Halifax	项目经理	管理整个项目
John MacMurly	主任	监督项目经理 每周检查情况 为监督委员会服务
Mac Preston	信息总管（CIO）	为监督委员会服务
William McDougal	市场/销售高级副总裁	指导项目负责人 批准预算、进度表 为监督委员会服务
Robert Schneider	产品目录销售主任	为监督委员会服务 提供用户支持/资源
Brian Haddock	业务主任	为监督委员会服务 提供用户支持/资源
Jason Nadold	运输经理	提供用户支持/资源

图3-5 RMO项目合同

如同在第1章的叙述的一样，系统的主要目标是支持RMO所要实现的，即建立客户诚信和为客户关系管理提供所有必要的工具。系统通过支持各种客户服务实现这个目标，包括为承担继续电话销售和互联网销售的新能力而提供订单、退回订单和在线产品目录的服务。客户不仅必须通过电话销售代表或互联网确定RMO产品的在线目录，而且能够看到他们过去的购买历史。RMO的经理们希望这个系统有几个响当当的功能支持他们眼中的RMO客户服务。

下一节通过RMO项目的实例，描述项目规划阶段的各项活动。如前所述，这些活动是为项目规划、组织、安排进度和最终获得批准所做的各项项目管理活动。注意，尽管这个项目看起来已经得到高层管理主管人员的默许，但它必须满足所有RMO项目的严格评审标准。尽管只有两个小组成员分管此事，但Barbara和Steve有丰富的经历和优秀的项目管理技能足以应付此事。

3.2.2 项目规划阶段

项目规划阶段由需要进行项目组织和启动的各种活动组成，如图3-6所示。与第2章开始讨论的一样，这些活动是：

- 定义问题
- 制订项目的进度表
- 确认项目的可行性
- 为项目安排人员
- 启动项目

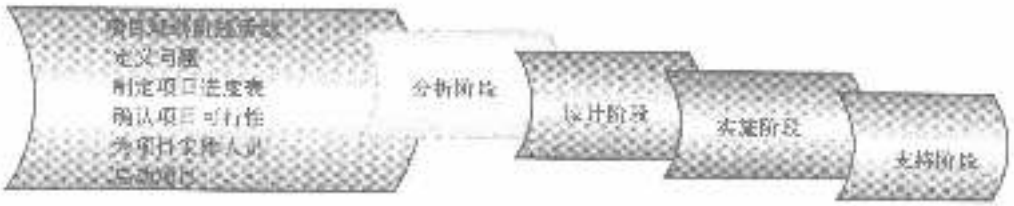


图3-6 项目规划阶段活动

这些活动是主要的项目管理活动。项目规划阶段通常配备两到三个有丰富经验的系统分析员，其中一人出任项目经理，其他系统分析员是具有很强分析技能，以及有管理和控制项目经历的、富有经验的开发人员。这些项目组成员常常首先成为核心组领导，然后在他们周围建立其余小组。在这一阶段成功结束时，分配好资源、制订出进度表并做好预算，项目即将启动。

为帮助了解项目规划阶段的各项活动，这一节首先解释各项活动，然后说明如何为落基山运动用品商店开展这些活动。在图3-7中列出了项目规划阶段的各个活动以及要实现每个活动项目组需要回答的关键问题。比如，在项目规划阶段后期，对"启动项目"活动要回答的关键问题是"准备好开始一个项目了吗？"

项目规划阶段的活动	关键问题
定义问题	理解要从事的工作了吗？
制定项目进度表	如果有可用的资源，项目能够按时完成吗？
确认项目可行性	开始从事这个项目仍然是切实可行的吗？
为项目组织人员	资源可用吗？足够了吗？准备好启动项目了吗？
启动项目	做好启动的准备了吗？

图3-7 项目规划阶段的活动及关键问题

3.3 定义问题

仔细定义问题是项目中最重要的活动之一，它的目标是准确地定义要解决的业务问题，从而确定新系统的作用域。这些活动定义了想达到的目标。如果这个目标定义有误，那么，就会影响所有随后的活动。如前面指出的，项目失败的主要原因之一是目标不清。

这个活动的第一个任务就是检查最初启动这个项目的业务要求。以RMO为例，如果项目是作为战略规划的一部分启动的，那么，就要检查计划文件；如果项目源于部门要求，那么，就要与主要用户商谈以帮助他们熟悉业务问题。随着需求确定，小组也制订一个详细的预期收益明细表，称为业务收益。业务收益表包括了组织希望通过新系统实现的结果。业务收益通常是以通过减少成本或增加收入改变财务结算表的影响来描述的。

业务收益：对组织增加的收益，通常用货币度量。

这个活动的第二个任务是高水平地确定新系统的期望能力，目标是根据能够解决这个问题的信息系统的需求定义问题域。尽管开始定义的期望能力并不看似是定义问题，但理解新系统的作用域和项目产生由来是必要的。

开发组成员获有3个部分，即问题描述、业务收益和系统能力，也就是一个系统作用域文档。这些成员（如系统分析员）与用户和顾客一起制订这个文档，有时这个文档是与项目合同结合在一起的，有时是独立的。图3-8所示为RMO系统作用域文档的一个例子，注意，业务

收益与系统能力是不同的。业务收益重点在公司的财务收益，系统能力重点在系统本身，通过系统提供的能力得到业务收益。

客户支持系统作用域报告

问题描述

产品目录销售在基本上是简单产品商店从小试验开始，下头扩展为公司快速增长的一个部门。最初通过人工和一些简单的程序来为接收订单和实现订单提供支持，到2004年，目录销售的增长快速互联网销售扩大了已有系统的压力，作为一个长远发展规划的结果，RMO决定启动两个主要的系统开发项目。第一个，供应链管理（SCM）系统2004年启动，被计划开始。第二个确定的系统是客户支持系统（CSS），它提供销售、市场和广泛范围的客户支持功能。这个项目是RMO继续发展并维护其在运动服装行业中领导地位的总的长远战略规划的需求部分。

期望的业务收益

从新系统中获得上述业务效益是维护RMO在运动服装行业中的领导地位、较多的直接收益有：

- 减少人工处理订单引起的错误
- 基于较快订单处理而提供快速订单发货
- 采用顾客和电话定购的处理方法降低最终顾客员工成本
- 通过高速交互式的网站极大地增加在线网络销售
- 通过面向大众消费者的广泛的营业额
- 通过广泛改善客户支持和信息提高顾客满意度

系统能力

为获得如上所列的业务收益，客户支持子系统将具有如下能力：

- 是一个能够为在线客户、订单、订单处理和交货信息提供满意支持的系统
- 有快速登录窗口支持传统电话和邮件目录销售
- 包括在线网客户和目录销售能力，包括购买和订单跟踪
- 保留合适的数据库和历史信息，支持市场分析
- 为客户查询者提供客户交易历史记录
- 系统不降低性能的前提下能够处理和实质性地容量增加（300%或更多）
- 支持24小时新订单的运营
- 从多个仓库调整订单发货
- 保留历史记录，支持销售分析和预测市场需求

图3-8　RMO客户支持系统的系统作用域文档

系统作用域文档：包含问题描述、业务收益和系统能力的支持，有助于定义新系统的作用域。

特别是当要推出一个具有现代技术发展水平新系统时，就有必要建立一个初步的原型作为检验概念的依据。新的解决方案，特别是基于新技术的解决方案，可能不是很好地被接受或理解。在这种情况下，项目组可以做一个概念原型检验以说明解决方案是可行的。当需要一个概念检验时，那么，项目作用域文档将参照初期原型结构，测试的结果和用途的适应性，例如，RMO高级管理主管人员要求系统能够自动地为通过互联网购买商品的客户建议其他的附属商品。项目组需要建立一个原型检验这种请求技术是否可行。

概念原型检验：一个初始原型，用于检验业务需求解决方案的可行性。

项目组也经常根据系统的信息流入与流出制订图表，描述系统的作用域。这个图称为关联图，表示系统的主要用户及用户与系统之间交换的信息。图3-9所示的是RMO客户支持系统的一个简单的关联图范例，称为RMO客户支持系统的关联图。从这一点看，项目组超越新系统作用

域文档，这个图仅包括主要的信息需求。事实上，这个图的主要着重点在于从系统输出的信息。第6章提供对关联图更详细的解释和RMO关联图的实例。第7章解释面向对象系统的用例图。

关联图：表示一个系统作用域的图表。

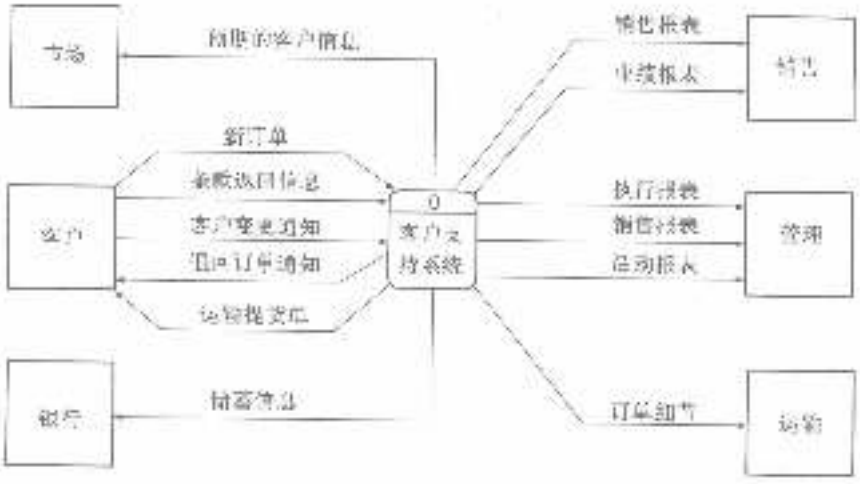


图3-9 客户支持系统的系统关联图

在图3-9中部的圆角矩形代表客户支持系统本身，在它周围的正方形代表实体，它们为系统提供信息或从系统接收信息，带箭头的线表示系统的主要输入和输出。这个图表仅标识了主要的进出系统的信息流，目的是给出所建议的解决方案的一个概观，并不包括细节。

注意，在分析阶段也使用关联图。在项目规划阶段，图表能帮助定义问题域，为了较详细地描述在分析期间所做的调查，这种图表变成了一个起点。

认真确定作用域，对评估完成项目所需的投入至关重要。系统的规模和作用域决定了需要投入的量，这个量又决定了项目的时间与成本。尽管不能对此进行精确的估价，但还是有一些方法可以用来衡量所提系统的规模和范围。大多数方法用来计算活动的数量和系统所支持的用例的数量。第5章讨论了活动和用例，并展示了识别活动和用例的技术。此外，通过计算系统问题域中的数据实体和类可以估测作用域，第5章还给出了这些识别技术。还有一些方法要计算确定的功能点的数量。软件开发成本估算模型（COCOMO）就是这样的方法，它将输入输出的数目、需要维护的文件数目及需要更新的数目等作为功能点来计算。

在完成这个问题定义的活动时，要回答的关键问题是，“我们理解我们很设要做什么吗？”

定义RMO的问题

CSS项目组的Barbara和Steven在与市场和销售副总裁William McDougal及他的助手交谈后，制作了系统作用域文档清单，第4章将较详细地说明有关被调查用户的情况并给出重要信息。在这里，重要的一点是要注意从将要使用这个系统和将要从这个系统中获利的人那里得到信息，他们能提供有价值的见识，确保系统满足业务需求。如前面提到的一样，一个系统开发项目成功的最关键因素是用户的参与。

为完成问题的定义活动需要一个补充的任务。项目组要进行可选方案的初步调查，重新评价项目启动时项目组所做的设想。由于对项目剩余部分的安排和预算假定一个特殊的开发途径，因此，明确这些假设是重要的，使所有的参与者能了解项目进度表的约束条件，使小组能完成一个正确的可行性分析。

例如，把一个现成的计划看成一个可能的解决方案，那么，分析阶段的部分进度表必须包括对照需求进行评价计划的任务。如果最可行的方案看起来是一个完全自行开发的新系统，那么，就要计划详细的分析任务并安排进度。

在Barbara完成问题定义的说明时，Steven对可能的解决方案做了一些初步的调查。他研究了贸易杂志、Internet和其他资源，以确定销售和客户支持系统能否很快购买并安装好。尽管他发现了几个，但似乎没有一个与RMO所需的性能很相称的。他和Barbara与William一起就如何最佳处理进行了几次讨论，他们决定在做出任何有关解决方案的决策之前，最好的方法是继续进行项目的分析阶段。在完成分析阶段的活动后，他们以更多的详细资料，再次对这一决策进行了讨论。现在，Barbara和Steven已开始为新系统制订进度表、预算和可行性说明。

3.4　制订项目进度表

在我们讨论项目进度表的细节之前，让我们阐明3个术语：任务、活动和阶段。从根本上说，阶段是由一组有关的活动组成的，活动是由一组有关的任务组成的，而任务是可识别和安排的最小的一项工作，活动也是可以识别、命名和安排的。例如，假设正在安排设计阶段。在设计阶段，要确定诸如设计用户界面、设计并统一数据库和完成应用程序设计这样的活动。在"设计用户界面"活动时，分析员可能要确定许多不同的任务，如设计客户登录形式和设计订单登录形式。因此，任务、活动和阶段的分解提供了3个水平的层次。

在项目规划阶段，不可能安排整个项目的每一个任务，因为在项目的早期不可能了解太多的所需任务。可是，项目规划阶段的需求之一是要提供完成这个项目的估计的时间和项目总成本。由于项目成本中主要因素之一是对项目组成员支付工资，因此，完成项目的时间预算变得十分重要。制定项目进度表是计划阶段最难但也是最重要的事。项目进度表的制订可以分为下列两个主要步骤：

- 制订工作分解结构
- 建立PERT/CPM图

3.4.1　制订工作分解结构

工作分解结构（WBS）是完成项目所需的各个任务的列表，在计划和执行项目中它是重要的，因为它是制订项目进度表、确定进度标志和管理成本的基础。表3-10所列的是RMO项目规划阶段的一个工作分解结构实例。

工作分解结构：划分一个项目的任务、活动和阶段，是评估和安排一个项目任务的方法。

注意，图3-10与SDLC的计划阶段的活动是很相似的。每个活动可以进一步分解成要独自完成的任务。WBS确定了层次，很像一篇论文的提要。项目要求SDLC每个阶段都有一个WBS，我们这一章讨论的主要是项目规划阶段，因此列出了这一阶段的WBS。显然，项目规划阶段是对整个项目进行进度安排，因此，分析、设计和实施阶段的WBS对问题的定义更加重要。

每个独立的任务都有一个相关的周期和任务所需的资源。本书的附录B中解释了任务的研究计划，包括提出持续时间数、资源数和用于这个任务的资源百分数。换句话说，一个资源对两天的任务仅消耗一半，而实际的结果仅是一天的事。虽然在表中没有显示，但有时候持续时间可以取3个值，即期望时间、保守估计的时间和乐观估计的时间，根据不同的取值项目经理可以制订乐观的、期望的和保守的进度表。不过，我们还是尽量使例子简单一些。

制订一个工作分解结构从零做起是相当困难的，项目组要集体讨论并设法考虑需要完成

这个项目的每一件事情。这个过程采取的是一种自底向上的方法，集体讨论每一个单项任务，希望你考虑到每一件事情，可是，通常两个技术中的一个是用做起始点的，这两技术叫做基于标准的WBS和基于类推的WBS。

阶段、活动和任务	持续天数	资源数	依赖任务
1.0 项目规划阶段			
1.1 定义问题			
1.1.1 会见用户	2	2	0.0.0
1.1.2 确定规模	1	2	1.1.1
1.1.3 书写业务收益说明	1	1/2	1.1.2
1.1.4 书写需求陈述	1	1/2	1.1.2
1.1.5 定义系统性能说明	1	1/2	1.1.2
1.1.6 制订关联图	1	1/2	1.1.2
1.2 制订项目进度表			
1.2.1 制订工作分解结构	2	2	1.1.3,1.1.4, 1.1.5,1.1.6
1.2.2 估计资源、周期和所有事物	1	2	1.2.1
1.2.3 制作PERT图和日程图	2	2	1.2.2
1.3 确认项目可行性			
1.3.1 确认无形成本和收益	1	2	1.2.3
1.3.2 估算有形成本	1	2	1.2.3
1.3.3 估算有形收益和计算成本/收益	2	2	1.3.1,1.3.2
1.3.4 评价组织和文化可行性	1	1	1.3.3
1.3.5 评价技术可行性	2	1	1.3.3
1.3.6 评价法规上可行性	1	2	1.3.3
1.3.7 评价资源可用性	1	1	1.3.3
1.4 获得项目人员			
1.4.1 制订项目资源计划	1	2	1.3.4,1.3.5, 1.3.6,1.3.7
1.4.2 确认和鉴别技术人员	1	1	1.4.1
1.4.3 与用户见面，确认工作人员	1	1	1.4.1
1.4.4 组织项目小组	1	1	1.4.2,1.4.3
1.4.5 实施小组整合训练	3	2	1.4.4,1.5.4
1.5 启动项目			
1.5.1 准备汇报材料	1	1	1.3.7
1.5.2 进行汇报	1	1	1.5.1
1.5.3 配备项目设备和支持资源	3	2	1.5.2
1.5.4 举开正式的启动会议	1	1	1.4.4,1.5.3

图3-10 RMO计划阶段的工作分解结构

基于标准的WBS是从标准计划制订，图3-9所给的例子可以称为一个基于标准的WBS。根据计划阶段定义的标准活动，项目组把每一项活动展开成更详细的任务。公司所使用的特定方法和SDLC在这些任务和活动中起重要作用。关键是要有一系列事先为这类项目定义的活动和任务。为美国政府工作的公司通常有为各种类型项目的标准WBS清单。政府合同经常需要

一个标准的项目定义。项目组以这些标准开始工作，然后制订另外的与项目有关的任务。

除了使用不同的输入作为起始点以外，基于类推的WBS形式上与基于标准的WBS相似。在基于类推的WBS中，项目组设法确认另外一个类似的项目，尽可能与现在这个项目相似，当然，最好是完全一样的。然后，根据以前项目的经验，小组制订一个新的WBS。做相同类型项目的公司可以一再地使用这个技术。例如，通用拿托这样的汽车制造商知道设计一辆新汽车的项目所需的所有各种步骤。这些公司就会使用以前相同的步骤。

一个好的WBS的主要好处在于它提供了对项目持续时间及开发该项目所要付出劳力的最准确估计。如果对主要过程是详细的而不是猜测的估计，那么，时间估计是较精确的。根据一个好的WBS制订出来的进度表也是容易监控的，WBS是项目成功的关键。

3.4.2 制作PERT/CPM图

PERT代表项目评估和检查技术，CPM代表关键路经方法。早期它们是两种截然不同的方法，但现在它们已经融合成一个单一规则进度的技术了。附录B详细阐述了PERT/CPM图的概念和制作方法。在附录中，你能了解如何手工创建PERT/CPM图，也能对辅助建立和维护PERT/CPM图的Microsoft Project软件程序有一些了解。几乎所有实际的项目都使用诸如MS Project或Primavera来维护一个项目进度表，由于每一个项目都是一个动态的活动，所以进度表经常要变化，手工制作一个进度标是很费时间的。

PERT/CPM：基于单个任务或活动对项目进行进度安排的一种方法。

PERT/CPM是一张可以在WBS中确定所有任务的图，它表示了各项任务间的依赖顺序。图3-11所示为RMO的关于PERT/CPM图的一个例子。这个例子仅是RMO进度表中的一部分，展示了PERT/CPM图的基本思路。每一个矩形框代表一个单一任务或一个概要活动，框内是任务名称、唯一标识、持续时间、项目开始和结束日期。任务从左矩形框左边所列的开始日期启动，在右边所列结束日期完成，带箭头的连接线表示任务的执行顺序。如你所看到的这个图，要注意一些任务是一个接一个按顺序完成的，而其他任务是并行或同时完成的。箭头代表任务的依赖关系，指明执行一个项目的正常顺序。

通过展示哪些任务是可以并行进行的，PERT图也可以有助于工作人员的安排。用相互依赖的和独立的任务来权衡小组成员的资格和工作量通常具有散列性。

建立PERT/CPM图是从运用在WBS中制订的活动和任务清单开始的。分析WBS，包括确定每个任务的持续时间和期望的资源，以及各任务之间相互关系。对每一个任务，这个图可以确认最临近它的前任任务和后续任务。你可以想像完成这样一个顺序任务是相当困难的。一般来说，对于任何一个项目，有无数的方法对任务排序。例如，小组应该先设计客户登录表格还是产品定义表格？两个都可以，但必须根据项目最大可能进展做出决策。遗憾的是，MS Project或任何其他的规划工具都不能为你做出这样的决策，在这一步上，自动化起不到作用。

关键路径是通过这个图的最长的路径，包括在定义顺序中必须要做的所有任务。称它为关键路径是因为这个路径上的任何一个任务如果没有按计划做，将延误整个项目。项目经理通常必须十分小心地监控关键路径上的任务。

关键路径：表明项目完成最短周期的PERT图上的路径。

PERT/CPM方法的好处之一就是制作了一个图，能十分容易看清各任务间的相互关系和关键路径。可是，在PERT图上不易检查项目的进展。要检查分散在日程表上的各项活动的进

展需要另外一张图，叫甘特图。本质上它是一个水平条形图，一个条形代表一个任务，有效平轴日是天数。随着移动，甘特图可以很好地用于监控项目进展情况。图3-12所示为甘特图的一种，称为跟踪甘特图。该图中表示的是与图3-10和图3-11中相同的任务。左边一列是任务名；事实上，任务列表看起来更像早期制订的WBS。条形的长度表明任务的周期。在这个MS Project 例子中，任务条形可以是不同颜色的，分别表示关键路径、非起点、部分完成或完全完成。实心的任务是在关键路径上，带阴影的任务不在关键路径上，完成任务或部分完成都以不同深浅灰度来表示。2月处实心的竖线代表今天的日期，所以，项目经理能够很容易地在图中检查任务的状态。竖线左侧的任何一个没有完成的任务都是落后于预订计划的，右侧的任务表示提前完成。跟踪与竖线交叉的任务，可能是提前了，也可能是滞后了，根据所报告的状态，甘特图可能或也可能没有标明这些任务的状态。大多数项目经理发现PERT/CPM图在他们制订进度表时是有益的，一旦项目启动甘特图就会变得更有用。

甘特图：表示项目各项任务和活动进展的一个条形图。

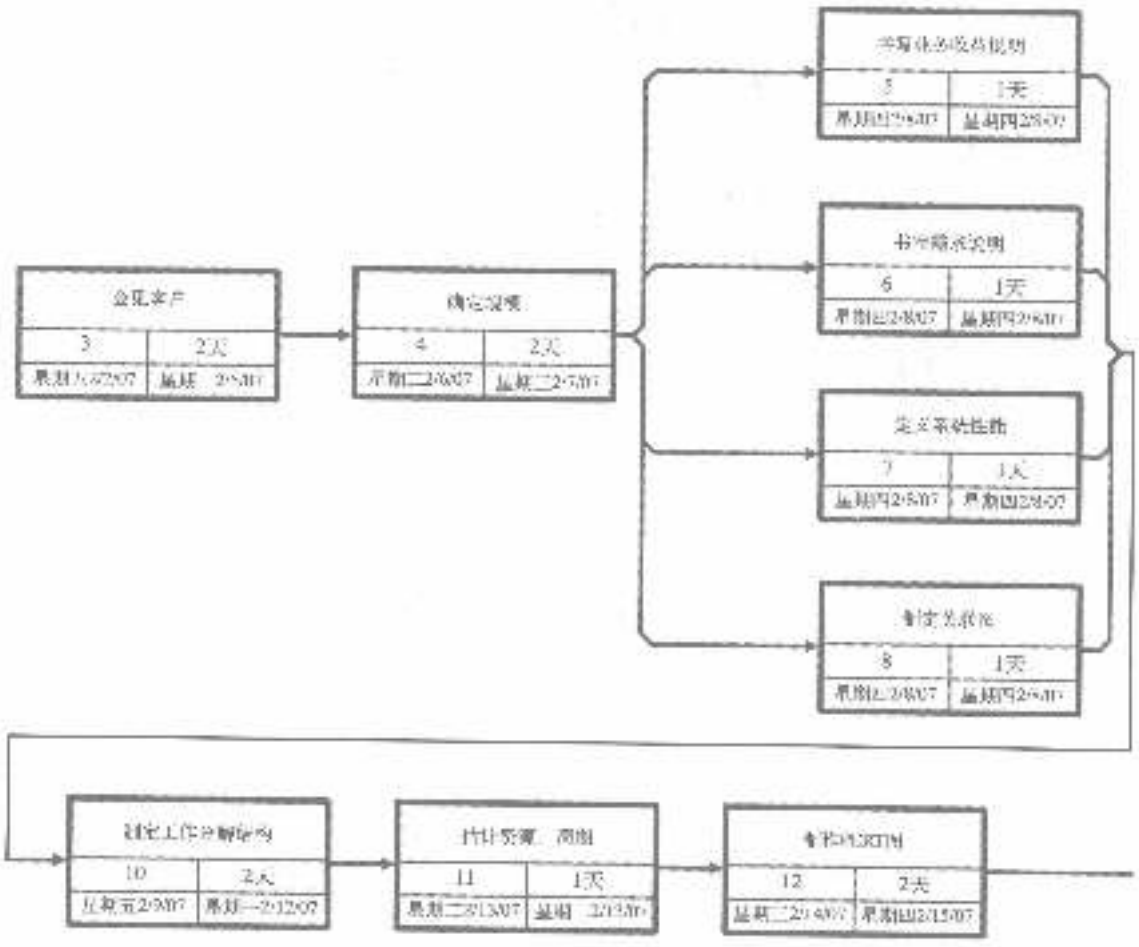


图3-11 客户支持系统项目的部分PERT/CPM表

跟踪甘特图：甘特图的一种，表示当前日期和每项任务完成的工作量的百分比。

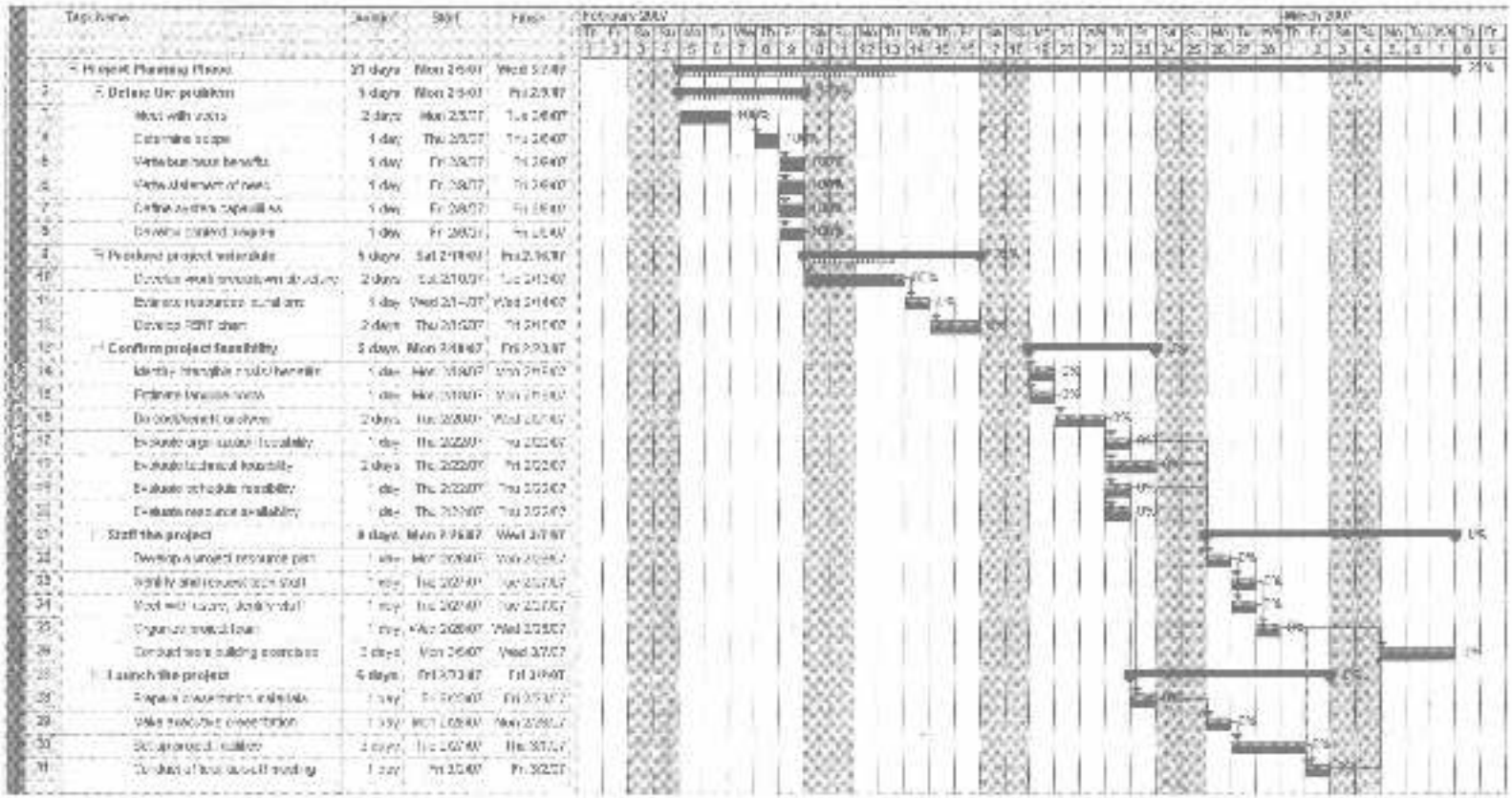


图3-12 客户支持系统项目的跟踪甘特图

3.4.3 为整个SDLC制定进度表

在图3-10、图3-11、图3-12中，只是详细介绍了项目规划阶段的WBS。很显然，SDLC其他阶段也需要进行时间安排。SDLC为每个阶段提供了一系列活动，每个活动又由一些任务组成。基于标准或模拟的WBS为分析、设计、实施阶段的活动分别提供了详细的任务列表。在本书中，你可以通过更多地了解SDLC各个阶段及各阶段的活动来更好地理解任务的需求。

如果假设项目包含SDLC中相互交叠的各个阶段，则图3-13所示的甘特图就从阶段和活动的层次上完整地描述了整个项目的具体细节。注意，此处的每个活动的长度并不代表团队成员从开始到结束一直在进行这项活动，相反，活动的开始及持续在不同时间段所付出的努力也不尽相同。所有的团队成员同时负责多项工作，即同时负责一个或多个活动和任务。因此，浏览整个项目并不能计算出总体劳动成本。但我们可以从此处了解到项目实施阶段截止到10月31日。从中还可看出CSS的开发时间需要9个月，其后将是支持阶段。

各个阶段的交叠是由SDLC中的迭代方法引起的。为了便于计划和确定进度，许多项目经理使用项目管理软件和甘特图规划和跟踪项目的迭代。每次迭代重点是系统的功能性，包括系统的分析、设计和实施活动。有些分析活动包含在每次迭代中，但有些活动则只包含在少数迭代中。例如，每次迭代可能都包括以下分析活动：收集信息、定义系统需求。每次迭代都包含的设计活动如下：设计应用结构、设计用户界面、设计并集成数据库。同样每次都迭代包含的实施活动如下：构建软件部件、验证并测试。各个阶段的其他活动可能并不全部包含在每次迭代中，这要根据项目的计划来确定。

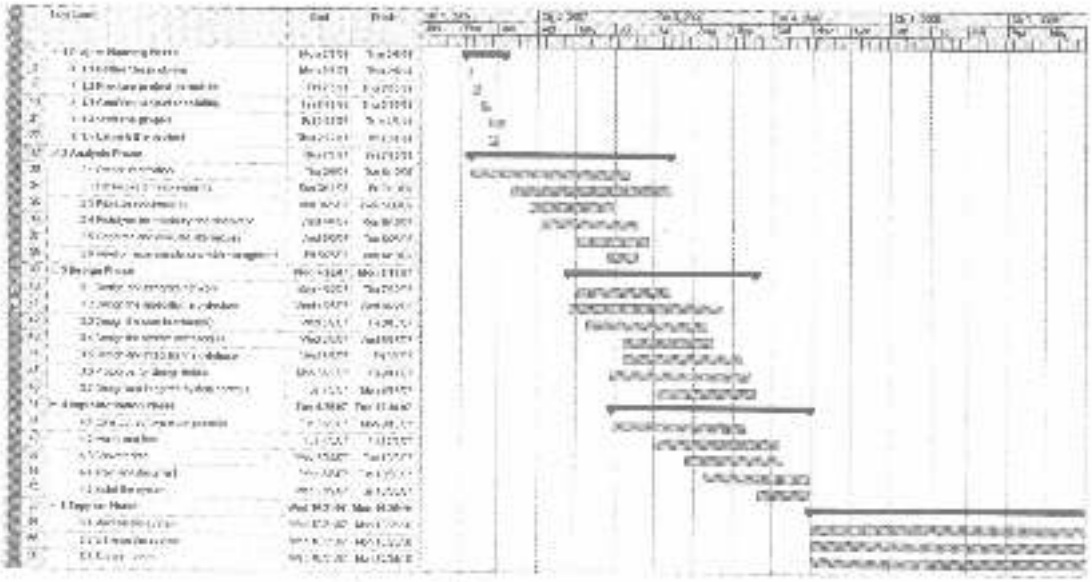


图3-13 完整的客户支持系统项目甘特图

图3-14给出了RMO项目有3次迭代的进度安排。项目的团队成员并不关心项目自身应在SDLC的哪个阶段。相反，SDLC的各个阶段和活动为确定项目规划和整个项目的多重迭代提供了一个框架。

附录B给出了更详细有关规划技术的信息，特别是如何建立进度表，包括使用MS Project

的PERT和甘特图。当完成这一活动时要回答的关键问题是："在给定的可利用资源的时间里能完成这个项目吗？"

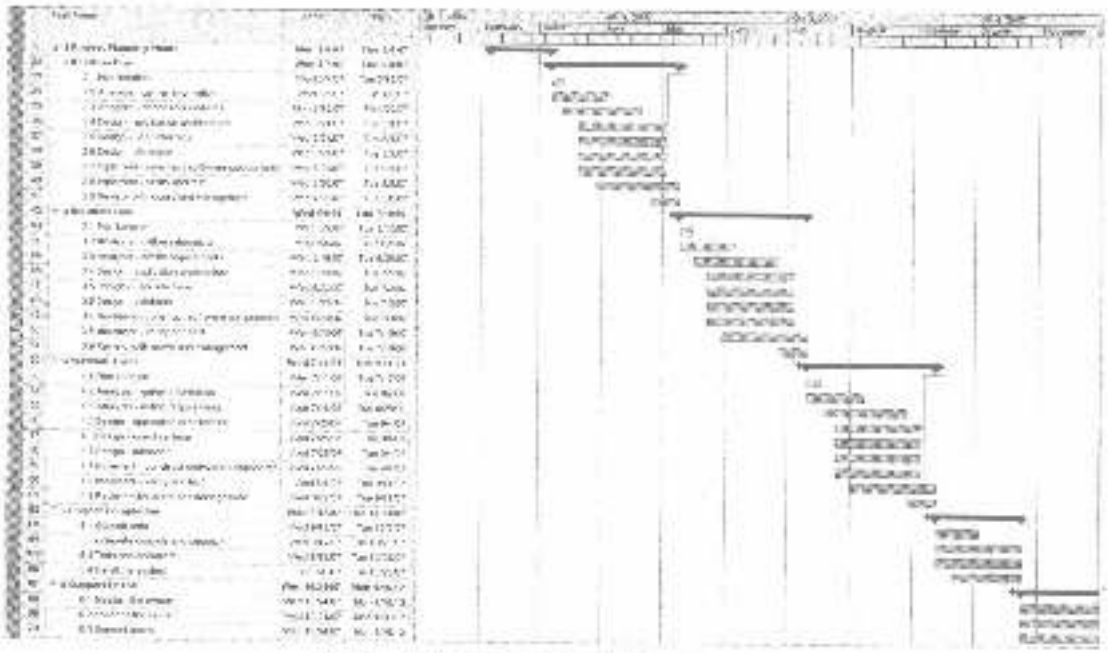


图3.14　客户支持系统项目选定方法的甘特图

3.5　确认项目可行性

项目可行性分析是确认项目是否启动并成功完成的活动。根据定义，项目是独一无二的努力成果。每一个项目都有唯一的影响其可行性的任务。正如本章前面所学，信息系统项目没有良好的记录，甚至一个计划完好的项目有时也会出错或陷入困境。

评价可行性的目标是确定开发项目是否有合理的成功机会。可行性分析本质上是要确认所有失败的风险。项目组应该有能力评价最初的假设，识别危及项目成功的其他风险。在识别这些风险后，如果必要的话，要确定计划和过程以使它们的影响最小。可是，如果有严重危及项目的风险，项目组必须尽可能地发现它们并进行评价。通常情况下，在确认项目可行性时，团队可以考虑以下风险：

- 评价项目的风险（风险管理）
- 确定经济可行性
- 确定组织上和文化上的可行性
- 评价技术可行性
- 确定进度表可行性
- 评价资源可行性

3.5.1　风险管理

可行性分析也包括风险管理。风险管理是具有远见性的项目管理领域，在这个过程中团队要试图识别潜在的危机项目成功的危险。有时，项目经理从各个角度只看到了可行性却忽视了识别确定的风险。我们认为，好的项目管理既需要仔细观察项目的整体可行性也要看到

个别的风险。我们首先提出识别和评价风险的一个简单的技术，然后解释应该为项目可行性和风险管理考虑的各个方面。

风险管理：在项目管理过程中，试图识别潜在的风险，该风险有机会项目的成功。

风险管理贯穿于整个项目生命中。在项目的初始阶段，风险管理的主要活动是识别潜在的风险和评价它的负作用。没有迅速简易的方法用于识别项目可能面临的所有风险。识别风险的最好方法就是集体讨论。核心组员应该是讨论的主要参与者，挑选的客户投资者也应该参与其中。最好选择那些有项目经验的或者以前干过这类项目的人参加会议。任何一次集体讨论会，参加人员都应该各抒己见，以判断和减少坏的项目。

实践指导

项目核心成员和客户投资者一起参与讨论会是识别风险的好方法。 ■

在识别出潜在风险后，团队就可以使用简易的矩阵分析潜在风险对项目的害处。图3-15用简易表描述了这一技术。左列是识别出的风险列表，第二列表为对项目的潜在影响，显示了如果风险发生，将会给项目带来怎样的坏影响。团队给出了3种可能的主观判断：高，中，低。下一列显示了反作用发生的可能性。团队没有计算复杂可能性，而是再次评价可能性是分为高、中还是低。

风险描述	对项目的潜在影响（高，中，低）	发生的可能性（高，中，低）	达到预期的难度（难，中等，容易）	整体危害（高，中，低）
关键雇员（专家）不可用	高	中	中	高
变更法定需求	高	低	难	低
引入计算机时新的组织成员	中	中	容易	中

图3-15 量化风险分析

下一列记录了预测负面事件发生的难易程度，及预测是否及时，以便采取正确的行动，例如，首要的风险就是关键专家不在团队内，如果项目经理发现某天组员应该开始工作了，而他/她却不在，那么显然就很难预测风险，就可能给项目带来负面影响。如果项目经理期望有一个月报警，告知需要的专家不可用，这样就可以容易预测到负面事件，并可以安排一些其他的偶然事件。这一列中团队把风险分为难、中和低三个等级。

最后，给中间三列赋值，团队对每个风险分配整体评价。项目经理通过这些信息观测潜在风险，并阻止负面事件发生或当负面事件发生时给后面设计好准备。

3.5.2 经济可行性

经济可行性有两种测试：① 收益预期值是否大于项目的开发成本？② 在开发期间该组织是否有足够的现金流投资于项目？尽管根据需求或战略规划，项目可能已经得到初步批准，但最后的批准通常需要一个全面的开发成本和预期财务收益的分析。显而易见，一个新系统开发的理由是它将通过节省成本，或增加营业收入增加盈利。项目经济可行性的确定总需要一个全面的成本/收益分析。

成本/收益分析：分析比较成本与收益以了解投资新系统的开发是否有益。

确定成本/收益分析有二步。第一步是评估预期开发和运行的成本。开发成本是新系统开发期间发生的，运行成本是系统交付使用后发生的。第二步是评估预期的财务收益。财务收益是指自新系统安装后预期得到的每年的节省或收益的增加。第三步是成本/收益分析是基于详细的成本与收益的评估而计算出来的。在成本/收益分析期间，有经验的分析员经常犯的错误大多数是没有全面定义成本和收益就进行计算。不能全面支持细节的成本/收益分析是没有价值的。

1. 开发成本

尽管项目经理对开发成本的评估承担最后的责任，但高级分析员需要帮助计算开发成本。最的来说，一般项目成本有下列几种形式：

- 薪水和工资
- 设备与安装
- 软件与许可证
- 咨询费与对第三方的付款
- 培训
- 设施
- 实用程序与工具
- 支持人员
- 旅行与杂项

小组成员	项目薪水/工资
项目经理	$101 340.00
高级系统分析员	$90 080.00
系统分析员	$84 980.00
程序分析员	$112 240.00
程序员	$58 075.00
系统程序员	$49 285.00
总计薪水和工资	$496 000.00

图3-16　RMO顾客支持系统项目的薪水和工资支付的详细说明

薪水和工资是根据项目的人员需求计算的。在制订项目进度和提供计划人员时，项目经理估算薪水和工资的成本。图3-16是为RMO客户服务系统估算薪水成本的一个实例，表中数字是根据确定项目需要的人员和他们分配到项目中时间的长短计算出来的。前面已提到每个人员在项目上的分配时间难以估计，有的团队能同时进行多个项目，如果工作分解结构（WBS）描述详细且精确，薪水和工资的成本就可以准确确定。

为确定预算成本，其他类别的每一种成本都需要详细地计算。项目经理可以对设备、软件许可证、培训费等进行详细估算，这些细节结合起来可以提供开发总成本的预算。图3-17是所有成本的一览表，表中的每一行都必须要有如图3-17所示的详细说明。

费用分类	金　额
薪水/工资	$496 000.00
设备/安装	$385 000.00
培训	$78 000.00
设施	$57 000.00
使用程序	$152 000.00
支持人员	$36 000.00
旅行/杂项	$112 000.00
许可证	$1.8 000.00
总计	$1 336 000.00

图3-17　RMO顾客支持系统项目的开发成本一览表

2. 运行费用出处

一旦系统完成并运行，每年就要有正常的运行费用，这些每年的运行费用必须在新系统的成本与收益计算中说明。分析员一般并不把正常的运行费用包含在这一成本中，仅包括与新系统直接有关的费用和维修费用。下面给出可能纳入新系统运行主要费用的类别：

- 适应性
- 设备维护
- 更新软件费用
- 计算机运行
- 程序设计支持
- 设备分期偿付
- 培训和辅助（帮助台）
- 供给

循环费用	金　额
适应性	$60 000.00
设备维护	$40 000.00
程序设计	$65 000.00
与贴合	$28 000.00
分期偿付	$48 000.00
总计循环花费	$241 000.00

图3-18　RMO顾客支持系统的年运行费用一览表

图3-18归纳了RMO每年的运行费用。作为开发成本，一览表中的每一个条目都应该进行详细的计算。表中仅表示了RMO系统预算的费用，其他的组织可能有不同的运行费用。

3. 收益来源

项目经理和项目组成员可以确定大多数的开发和运行成本，可是，用户和委托人也会得

到收益。因此，委托人和用户必须确定预期收益值。项目组成员要提供帮助，但绝不是自己确定收益值。

通常收益有两个主要来源：减少成本或增加收入。公司运行效率的提高可大大节省成本，减少开支。用持减少成本的领域包括下列几个方面：

- 由于手动功能的自动化或效率增加而减少工作人员
- 维持定量的工作人员而增加工作量
- 减少运行费用，如应急发货的运输费用
- 由于自动编辑或确认减少错误率
- 确保文件或交易的快速处理和周转
- 捕获货币管理上损失的折扣
- 减少不良账单或不良信贷损失
- 由于严格控制，减少库存或商品损失
- 更快地收取可收账款
- 较好的库存管理减少由于过期造成的损失
- 采用批量折扣和购买减少商品成本
- 采用电子数据交换和其他自动化手段减少文书工作的成本

这仅是许多可能产生收益中的一个范例。不像开发成本那样，没有“标准”收益。每个项目都是不同的，所以预期收益也是不一样的。图3-19所示为RMO预期从新的客户支持系统的实施中获得收益的一个例子。

收益/成本节省	金额	注　释
销售部门增加效率	$125 000.00	5人$25 000
电话订购部门增加效率	$25 000.00	1人$25 000
仓库运营增加效率	$87 000.00	
由于网络自动增加效率	$500 000.00	增加50%/年
其他节省（库存，供应等）	$152 000.00	
总的年收益	$889 000.00	

图3-19 RMO收益范例

4. 财务计算

公司可以使用多种方法测定新系统的总收益。一个通用的方法是确定新系统的净现值（NPV）。净现值的两个概念是：① 用如今的美元（现值）计算所有的收益和成本；② 把收益与成本结合起来给出一个净现值。换句话说，未来的收益与成本的流量是同时得到的，以后每年用一定的因子进行折算。贴现系数除了用来把未来的价值变换成现在的价值以外，与利率是一样的。书本的网站上附录C给出了如何计算经济可行性的详细说明。读者应阅读附录C以确保理解这些细节。

净现值（NPV）：一个新系统投资的收益与成本的美元现值。

图3-20是在附录C（图C-1）所做的RMO净现值计算的副本。在这个情形下，用10%的折扣率，这个新系统5年以上可以给出$3 873 334的NPV。

机构用于确定一个投资是否会有收益的另一种方法是确定投资回收期。有时投资回收期称做盈亏平衡点，它是一个时间点，在这个时间点增加的现金流量（收益）正好还清开发和运行的费用。附录C提供了需要计算的详细的方程。图3-20给出了投资回收期的计算。连续累积的净值是逐年计算的。当这个值变成正的那年就是偿还发生的年，在RMO实例中，发生在

第3年内。

RMO成本收益分析	第零年	第一年	第二年	第三年	第四年	第五年	总　计
1. 收益值	$-	$889 000	$1 139 000	$1 514 000	$2 077 000	$2 927 000	
2. 贴现系数（10%）	1	0.9091	0.8264	0.7513	0.6830	0.6209	
3. 收益净现值	$-	$808 190	$941 270	$1 137 468	$1 418 591	$1 817 374	$6 122 893
4. 开发成本	$(1 336 000)						$(1 336 000)
5. 运行成本		$(241 000)	$(241 000)	$(241 000)	$(241 000)	$(241 000)	
6. 贴现系数（10%）	1	0.9091	0.8264	0.7513	0.6830	0.6209	
7. 成本净现值	$-	$(219 091)	$(199 162)	$(181 063)	$(164 637)	$(149 637)	$(913 559)
8. 计收益与成本的净现值	$(1 336 000)	$589 097	$742 107	$956 405	$1 253 958	$1 667 737	
9. 累积的净现值	$(1 336 000)	$(746 903)	$(4 789)	$951 609	$2 205 597	$3 873 334	
10. 投资回收期	2年 + 4796/(4796 + 951 609) = 2 = 0.005或2年又2天						
11. 6年内投资收益率	[6 122 893 − (1 336 000 + 913 559)]/(1 336 000 + 913 559) = 172.18%						

图3-20　RMO的净现值

投资回收期：美元收益抵消美元成本的时间段。

盈亏平衡点：美元收益抵消美元成本的时间点。

投资收益率（ROI）是组织使用的另一个经济度量。NPV的目标是基于预先确定的贴现率确定一个专门的值，ROI的目标是计算一个所需的百分数投资收益率（如同利率），在指定的时间段结束时成本与收益正好相等。表3-20表示了为RMO计算的一个ROI，附录C提供了细节。这个时间段是预期的投资期（即系统的有效使用期限）或一个任意的时间段。

投资收益率（ROI）：从一个新系统投资中而收到的百分数收益的一个度量。

为RMO假设了5年的收益周期，ROI是172.18%。换句话说，开发成本上的投资在5年时间里返回了172.18%以上。显而易见，这个系统在那个时间点及时产生了收益，如果假设一个更长的使用期限，如10年，那么，将得到更高的ROI。

5. 无形收益

以前的成本/收益计算是依赖组织对成本和收益量化的能力。可是，在许多情况下，成本和收益是不能度量的，它们的值也不能确定。如果可以把一个估算的值算入收益或成本，那么，它就看成有形收益或成本。如果没有估算或度量值的方法，那么，它就被认为是无形收益。在一些情况下，无形收益远大于有形成本，至少从委托人来说，即使美元数不能预示一个好的投资，这个系统还是要开发的。

有形收益：用美元可以度量或估算的，对组织自然增加的收益。

无形收益：对组织自然增加的，但不能量化或正确估算的收益。

无形收益包括如下几种事例：

- 提高服务水平（不能用度量的方法）
- 提高客户满意度（不能度量）
- 生存（工业上常用的一个标准性能，或对许多竞争者是常用的）
- 需要自行开发的专门技能（如用新的技术指导计划）

无形成本包括如下几种事例：

- 降低职工士气
- 影响生产力（组织不能估算）
- 失去客户或销售（在一些未知的时间段）

即使无形的不能算入NPV、ROI或偿付的计算，事实上它们也应作为项目进行与否的决定

习于来考虑。

实践指导

无形收益不包括在费用/收益分析中，但它们是开启项目的最重要原因，指导其必须清楚管理的目标。■

6. 资金来源

如同在前面解释的一样，项目组把成本/收益分析与项目预算制订一起完成，经济可行性的两个部分涉及成本/收益分析的好的效果和系统开发的资金来源。组织可以对开发项目以多种方式提供资金，新的信息系统常常是结合现金流量和长期资金提供资金。项目组不涉及为项目争取资金。可是，成本/收益分析的结果会大大地影响财务决策。

3.5.3 组织上和文化上的可行性

如同第1章讨论的一样，每个公司都有它自己的文化，任何新的系统必须适应那种文化。常常有这样一种情况，即新系统可能与原有的标准有很大的不同，而造成系统不能成功使用。涉及可行性分析的分析员应该评估组织上和文化上的问题，以便识别潜在的新系统的风险。这样的问题包括如下几种：

- 当前低水平的计算机能力
- 实际存在的计算机恐惧
- 工作人员或管理人员的失落感
- 由于新系统引起行政上和组织上的潜在变化
- 担心工作职责变化
- 担心由于增加自动化而失业
- 撤销持续长时间的工作过程

列举在组织中存在的所有组织上和文化上的潜在的风险是不可能的。项目管理组为了识别和抵抗这些风险，需要对组织内一些微妙的事情很敏感。此外，分析员询问运行可行性的问题是："什么可能妨碍新系统有效使用并导致损失商业利益？"

一旦识别了风险，就要采取积极的措施。例如，增加举行培训活动以讲授新的过程和提供强化计算机技能。参加新系统开发的高级用户能增加用户的积极性和承担义务。

3.5.4 技术可行性

一般情况下，一个新系统可以给公司带来新的技术，有时新系统扩展了技术的最新水平。其他的项目可能使用原有的技术，但把它结合到新的、未测试的结构中。最后，如果公司内没有专门技术人才，原有的技术也能更新。如果外来的厂商提供能力，通常被看做是这个领域的专家而提供服务，这样他们会担负所需技术水平太复杂的风险。

项目管理组需要十分谨慎地评定所述成的技术需求和有用的专门技术人才。当这些风险被识别后，解决方案通常是公正而简明的。技术风险的解决方案包括进行额外的培训、雇佣顾问或雇佣更多的有经验的雇员。在某种情况下，项目的作用域和方法需要变化以改善技术风险。重要的一点是现实的评估将较早地识别技术风险并有可能采取正确的措施。

3.5.5 进度安排可行性

一个项目进度表的制订总是伴随着某风险的行为。每一个进度表需要许多假设和用不充分的项目信息来做的估算。例如，对新系统的需求和作用域不十分了解，那么，研究的估算时间和必须要估计的最后需求、项目组成员的可用性和能力都是可疑的。

在进行进度安排时容易发生的另外一个高风险就是领导要求新系统必须在一定时间内开始。有时，确定最终期限是一项重要的业务需求。如RMO项目中要求及时完成CSS以方便在假期中网上订购商品。同样，一些大学往往会要求在某些关键日期前完成新系统。例如，学校新的注册系统在入学时没有完成，那么有可能必须延至下一学年。类似的例子说明，系统的时间进度安排的可行性是一个最重要的可行性因素，必须考虑。

如果最终期限可以随意定，那么要制定进度表的意图只为了表明它能够建立一个进度表，遗憾的是，这通常意味着实难。建立进度表应该没有预想的完成时间，一旦进度表完成，对所要求的日期要进行比较，以了解时间表是否一致。如果不一致，则采取改正措施，如减少项目的作用域，就能增加项目按时完成的可能性。

定义项目进度表中的里程碑的目标之一是允许项目经理评估正在出现的进度表失误的风险。如果里程碑一开始有误，就要尽可能早地矫正，可以制订并执行临时计划以减少风险进一步扩大。

为项目安排足够的有丰富经历和专门技术的人员通常是个问题。任何复杂的项目都会有潜在的超出限度和进度延期的可能，识别这些风险的源头是困难的，但有意的努力至少可以识别比较明显的风险，长期项目更难于进行资源分配和克服进度拖拉，解决方案应该包括意外事故计划以防为部资源不到位。

3.5.6　资源可行性

最后，项目管理组必须为项目评估资源的可用性。主要的资源由项目组成员组成。开发项目需要系统分析员，系统技术员和用户的参与。一种风险是项目组得不到所需要的人。另一种风险是所分配的人没有所需技能。一旦项目组行使职责，就不断有成员离开组的风险。如果其他特别吸引人的项目在组织内部执行或合格的成员被其他组织雇佣时，这种现象就会在内部发生。尽管项目经理通常不愿意考虑这种风险，但有技能的人供不应求，有时就必定会离开项目组。

一个成功的项目所需的其他资源包括足够的计算机资源、物理设备和维护人员。这些资源大体上可以得到，但如果在这些资源要用时取得了，则会影响进度。

3.5.7　可行性分析

前述每一种可行性分析都假设项目是可行的，事实上，这一节的标题是“确认项目可行性”，并不是每一个项目都是可行的。一个项目要是可行的，它必须通过所有的可行性测试。换句话说，项目组必须认真检查项目的各个领域并根据相关数据做出决定，如果这个项目有任何一点不可行，项目就必须被调整。如果调整后，项目的不可行性仍没有改观，则项目就必须被停止。现在，对启动一个有高度失败风险项目的可行方案就是什么也不做。项目今天不可行，例如，由于技术困难、高成本或没有足够的专门人才，在今后可能变成可行。对一个项目经理来说，总是难于决定这个项目是否是不可行的，不应该做。可是，启动一个注定要失败的项目就会损害公司和与之有关的部门的利益。

这5个可行性领域的每一个评估都是项目规划阶段的重要部分，当完成可行性分析时要回答的关键问题是：“开始从事这个项目仍然是如实可行的吗？”

3.6　为项目组织人员并启动项目

为项目组织工作人员是项目经理的主要职责。如同附录A中解释的一样，人力资源管理包括发现具有合适技能的人员，然后贯穿整个项目组织并管理他们。组织项目人员有5个任务：

- 为项目制订一个资源计划
- 确定并邀请专门技术人才

- 确定并邀请专门用户人员
- 把项目组分成多个工作小组
- 实施初步的培训和建组训练

基于在项目进度表中明确的任务，项目经理可以制订一个详细的资源计划。事实上，进度表和资源需求通常是同时制订的。如果使用诸如Microsoft Project这样的工具建立进度表，那么对每一个任务所需的资源是总进度表中的一部分。在制订这个计划时，项目经理必须认识到：① 资源通常不是想要就会有的。② 要求小组成员在一段时间内熟悉项目。

一旦制订了计划，那么就要确定并邀请专门人员参加小组。通常项目组需要两种人员：技术人员和用户。技术人员由系统分析员、程序分析员、网络专家和其他技术人员组成，常常根据需要会从一个项目转到另一个项目。项目经理经常要与信息系统的主管或副总裁联系以确认和安排所需资源。在某些情况下，需要补充技术人员。这样人力资源部门就要参与补充，招聘新人员。虽然寻找技术人员是规范的过程，但它需要一些时间去寻找并分配所有所需的项目组成员。

用户成员是分到项目组的组织内部的人员。有时把用户专职分到组里是很难的。对用户而言，分到项目组不是正常工作过程的一部分。可是，如果一些专职的组员能代表用户团体并起到联络作用，项目就会进展顺利。参照项目失败的反馈原因，证实与用户的紧密联系和分到项目小组的成员之间的密切合作可加大成功的机会。

对于小项目，项目组成员可以都在一起工作。但是，一个项目组多于4人或5人，通常就要分成较小的工作组。每个小组将由一个小组领导协调分到小组的任务。这个任务的责任由项目组领导承担。

最后，实施培训和建组训练。总体上可以为项目组进行培训，如当使用新技术、新数据库或新程序设计语言时，要进行培训。在其他情况下，需要对不熟悉所用工具和技术的新组员进行个别的培训，对技术人员和用户应该实施适当的培训。如果项目组成员以前没有在一起工作过，那么建组整合训练就特别重要，在设立有效小组和工作小组时要重点考虑小组的用户人员与技术人员的整合作用。

当完成这个活动时，要回答的关键问题是："资源可以用了吗？训练了吗？准备好启动项目了吗？"

在完成前面的活动后，就到了启动项目的时间。现在已经定义了新系统的作用域，确认了风险，项目从经济和其他方面发现是可行的，制订了详细进度表，确定了项目组成员并准备启动项目。这时通常有两个最后的任务：第一，监督委员会最后定案，为项目发出最后的许可信号，包括必需资金的核发；第二，通过组织内正常的通信渠道发出正式通知，对项目给予信任并要求组织内所有参与的各部门进行合作。换句话说，项目得到了组织上高级管理人员的批准和明确的支持。没有这两步，任何一个项目都不能启动。

当完成这个活动时，要回答的关键问题是："我们做好启动的准备了吗？"

3.7 RMO项目规划翻新

Barbara和Steve花费了整个二月提出了CSS的进度表和计划。尽管Barbara是项目经理，但她与Steve同等地在一起工作。作为一个组，他们集体讨论，加倍相互检查各自的工作。在这之前，他们已经在一起工作，有良好的关系——基于相互的尊重和信任。他们是公正的，知道如何通过争论开展工作，以及对重大问题如何达成一致意见。Barbara也知道Steve所做的工作总是慎重考虑过的并且是十分内行的。他是一个熟练的系统分析员，可以确保在计划阶段所做的工作是可靠的。

整个项目的成功十分依赖于Barbara和Steve在这一阶段所做的计划。在项目规划阶段建立

了所有其他的项目活动的基础。就像Barbara为启动项目的正式动员会计划的一样，她检查项目管理人员的领域，确信她已经处理了所有重要的问题。

为管理项目作用域，她制订了一个商业利润、系统能力和关联图的列表。在这一点上，作用域的定义还是泛泛的，她要确信在分析阶段的信息收集活动期间项目作用域要有十分精确的定义。

她和Steve制订了一个详细的工作分解结构，把信息输入到MS Project。分析阶段的进度表制订得十分详细，但设计和实施阶段却差些。在做出实施方案的决策之后，她增加这些阶段的一些细节。她认为他已经制订了项目时间管理的方案，随着项目的发展，她有跟踪进度所需的工具。

成本和潜在的收益也已经估算出来，并可用于制订一个NPV估算。当分析阶段结束时她重做进度表后，重新计算了NPV，以确信成本和进度是在允许的预算之内。成本管理的其他部分是监控项目生命周期内的成本。MS Project将帮助她跟踪每一项任务的成本。

Steve生可行性分析期间做了大量确认和评估风险的工作，Barbara知道他们将继续寻找风险和评估项目期间潜在的问题，她要求Steve每周花时间评估风险和更新项目最大风险列表。她确信她能够预见一些突如其来的问题。

对于项目通信和项目质量，Barbara建立了项目进程。她建立了一个中心数据库，用来存放项目状态、决策和工作文档，确信所有的小组成员都能知道这些信息。她制订了一个流程的和小组领导的每周一次的状态报表的格式，以及给监督委员会的状态报表的格式。她对监督委员会的状态报表备忘录之一的实例如下所示，这些状态报表都遵循一个标准格式。除正式的状态备忘录以外，她也对John MacMurty写了不少信息备忘录。为了项目质量，内部规程需要小组成员和RMO用户检查所有工作产品。其他的质量规程，如测试计划，将随着项目的进展而建立。

2007年2月17日

To: CSS项目监督委员会

　　William McDougal主席

From: Barbara Halifax, 项目经理

按照John MacMurty的指示，在项目的前两个月我将每两周向监督委员会提交一份状态报告备忘录。以后，我将每月提交一次。

过去一段时间（两周）完成情况

Steve Deerfield和我一直在做两件事。第一，我们完成了一个需求导向的项目作用域的标述。如你所知，我们采访了你们和其他主要风险承担者，确定业务需求和整个项目作用域。

第二，我们已经制订了一个项目进度表，有一个规划和分析阶段的详细的WBS，以及设计和实施阶段不太详细的WBS。我已经附加了一份项目进度表的备份。我们期望在年底前系统投入运行。

下一阶段（两周）计划

在今后两周内，我们将最后定下项目的可行性，主要重点是制订一个初步的成本/收益分析。随着分析阶段更多的信息收集，这个财务分析会变得更精确。

我们也做了一个项目的可行性分析，我们的风险分析将确定项目是否可行，确认高风险领域。最后，我们将制订一个工作人员的计划，确认小组成员，包括技术人员和用户，开始为项目组织人员。

问题、难题、未完成项目——无

她和Steve已经确定了他们想要的其他项目组成员。John生寻找可靠的立即能来工作的分析员或不久能来工作的分析员方面起到很大作用。事实上，Barbara早已经面试了所有要来的成员。为了认识到小组成员一起工作的重要性，她已经安排几天时间让小组成员互相认识，熟练他们内部的工作程序，教给他们将在项目中使用的工具和技术。

总而言之，这是紧张、忙碌而有效的一个月，做了大量的工作，已经为一个成功的项目建立了可靠的基础。

小结

本章的重点是项目管理活动，这些活动构成了SDLC的项目规划阶段的基础。这一章共覆盖了三个主题：① 项目管理系统开发生命周期（SDLC）；② 信息系统项目开始阶段和计划阶段；③ 项目经理和分析员所采用的，并用于完成SDLC项目规划活动的一些技术。

新系统的开发需要一个有组织的、渐进的方法。这种方法称为系统开发生命周期（SDLC），在第2章已讨论过。SDLC定义了系统开发解决方案时需要注意的阶段、活动和任务。项目管理任务包括在项目开始的计划阶段，但贯穿了整个项目的各个阶段。

项目管理是组织和指导其他人员在预订的进度和预算内达到一个计划的目的。项目管理可以分成8个知识领域：作用域、时间、成本、质量、人力资源、通信、风险和采购。

项目的启动要依靠信息系统确认的需求及组织战略规划的先后次序。项目往往是随着问题的出现和指令的发出才被启动。一旦项目被启动，项目规划阶段主要是由项目经理和一个或两个其他的高级分析员一起完成的。项目经理的许多主要职责是在项目规划阶段的活动内完成的。这个阶段由5个活动组成：① 定义问题；② 制订项目进度表；③ 确认项目可行性；④ 安排项目人员；⑤ 启动项目。

为了定义问题，项目经理要仔细研究系统解决方案中最初定义的问题和想法。确定项目的作用域，初始系统的关联图用图形的方式对主要的输入输出进行描绘。项目进度表是通过创建各阶段、活动和人物的工作分解结构来制定的。而这些阶段、活动和任务都是依据SDLC完成项目所必需的。进度表要以确定的原因是阶段和活动常常交叠，并且项目可能要经过多次迭代。创建进度表的方法如PERT/CPM以及关键路径用于研究进度安排的短缺和风险。最终，项目进度表用于估算项目劳动成本，因为劳动是由项目成员消耗在项目任务上的时间所决定的。

确认项目可行性需要评价5类可行性相关的风险。这5类分别是经济可行性、技术可行性、进度可行性、组织可行性和资源可行性。经济可行性可通过成本/收益分析，即比较项目成本与预期的收益得到。虽然无形收益常常是推动项目顺利进展的重要原因，但净现值（NPV）、投资回收期和投资收益率（ROI）的计算仍被用来确定成本/收益分析是否对项目有利。

项目规划阶段用一个小的团队即可完成。这个团队通常由项目经理和一两个重要的分析员组成。当项目进行到分析阶段后，必须补充一些新的成员参加到该项目中。人员组织计划必须解决未来几个月中的人员需求问题。项目准备启动时必须通知主要的管理人员和执行监督人员，负责确保项目成功。

关键术语

breakeven point	盈亏平衡点
business benefits	商业收益
client	客户
context diagram	关联图

cost/benefit analysis	成本收益分析
critical path	关键路径
Gantt chart	甘特图
intangible benefits	无形收益
net present value (NPV)	净现值
oversight committee	监督委员会
payback period	投资回收期
PERT/CPM	项目评估和检查技术/关键路径
project management	项目管理
proof of concept prototype	概念原型的检验
return on investment (ROI)	投资收益率
risk management	风险管理
system scope document	系统作用域文档
tangible benefits	有形收益
tracking Gantt chart	跟踪甘特图
user	用户
weighted scoring	权重得分
work breakdown structure(WBS)	工作分解结构

复习题

1. 列表解释项目规划阶段的各种活动。
2. 列出项目失败的7个原因。
3. 列出项目成功的5个原因。
4. 项目启动的3个原因是什么?
5. 定义项目管理。
6. 解释信息系统管理与项目相似度。
7. 解释迭代开发方法是如何使项目进度表变复杂。
8. 描述用于评价项目的5个可行性。
9. 被用于评价经济可行性的成本收益分析的目的是什么?
10. 解释有形成本收益和无形成本收益的区别。在成本收益分析中哪一项可忽略?
11. 解释PERT图和甘特图的区别。
12. 列出通过安装新系统可能得到的5个或6个有形收益的来源。
13. 列出至少4个开发成本的出处。
14. 关键路径的意思是什么?
15. 系统关联图的目的是什么?
16. 描述项目管理的8个知识领域。
17. 计划阶段特别注重项目管理的活动是什么?

思考题

1. 写一篇论文，讨论项目管理的方法，如何避免本书开始列出的项目失败的原因。
2. 根据下列叙述，制订一张预期商业收益的列表。

   Especially for You Jewelers是大学城的一个小饮宝零售商。在过去的两年里，Especially for

You在它的业务方面经历了极大的发展，可是，它的财务业绩却与它的发展不同步。现在的事务处理系统部分手动、部分自动，不能有效地追踪客户账单和收据。Especially for You难于确定为什么它的成本这样高。此外，Especially for You频繁地实行特价以吸引顾客，它不知道特价是否有利可图，是否带来其他的销售。Especially for You也想增加回头客，所以，它需要一个客户数据库。Especially for You想安装一个新的直接销售和财务处理系统以帮助解决这些问题。

3. 根据下列叙述，制作一张系统能力的列表。

Especially for You Jewelers新的直接销售和财务处理系统是这家珠宝公司未来发展和成功的重要因素。系统直接销售部分需要保留每次销售的记录和能够访问当前系统的成本数据以提供每天的利润和亏损报表。客户数据库要求能够给出购买历史记录，帮助管理人员对原有顾客准备特别邮购和特别销售，每个顾客的详细资方余额和近期账单将帮助解决达到应收账款高度平衡的问题。特别注意许可证和信贷历史报表将帮助管理人员减少应收账款。

4. 根据你对问题3和4的理解，为Especially for You Jewelers制订一个项目合同。

5. 创建一个PERT/CPM图，为一个新系统建立和测试原形表格，并确定关键路径。

任务列表

任务	说　明	持续时间（天）	表代替的原有事物
0	开始	0	
1	会见客户	2	0
2	检查现有表单	1	0
3	确定并详细说明范围	3	1，2
4	创建初始原型	2	3
5	产生测试数据（有效数据）	4	3
6	产生错误测试数据	2	5
7	测试原型	5	4，6
8	最后的文档	3	7

6. 假设你在一个牙科医生办公室工作，要求开发一个系统以保留病人预约的记录。你将如何开始？你首先要做什么？你首先设法要查找的是什么样的事情？如何比较你的解决方法与本章描述的方法？

实验练习

1. 基于下面的活动使用Microsoft Project创建一个项目进度表，打印出PERT图和甘特图。

下表是一个学生的一组任务，这个学生具有国际学习经历，请您为这组任务的几个方案制订进度表。第一种方案，假设必须先完成所有优先的任务，然后开始随后的任务（简单方案）。第二种方案，确定在先前任务结束之前的几天里可以启动的几个任务。第三种方案，修改第二种方案。这样在一个先前任务开始之后的几天里，一些任务可以启动，也可以插入一些检查的任务，如应用程序任务、预备任务、旅行与抵达任务。陈述你对每一种迭代法的假设。

某学生的任务列表

任务标识符	任务描述	持续时间（天）	优先权
1	从国际交流办公室获取表格	1	无
2	填写并递交外国大学申请	3	1
3	收到外国大学的批准	21	2
4	申请奖学金	3	2

（续）

任务标识符	任务描述	持续时间（天）	优先权
5	收到奖学金批准的通知	30	4
6	安排等带资金	5	3，5
7	安排宿舍住宿	25	6
8	获得护照和必需的签证	35	6
9	递交学校预登记表	2	8
10	银行旅行安排	1	7，9
11	确定服装需求和购物	10	10
12	打包并做最后的出发安排	3	11
13	旅行	1	12
14	搬入宿舍	1	13
15	完成班级登记和其他大学文书工作	2	14
16	开始上课	1	15

2. 做一个展示你大学期间进步的项目规划，包括课程预备信息。如果你已经用过Microsoft Project或相似的项目管理工具，在项目管理工具中输入信息。
3. 使用你们班级或其他资源的信息，书写一页说明随着分析阶段项目组的扩大什么类型的小组建立和训练活动是合适的。
4. 询问一个系统分析员有关他或她在工作中使用的SDLC。如果可能，要求分析员给你一份项目进度表的副本。
5. 询问一个项目经理有关她或他对8个项目管理知识领域的选择。
6. 上CompTIA (www.compTIA.com)网站寻找项目经理考试（IT Project +）的要求，写一份专门技术概要和通过考试所需的知识。

实例研究

客户载重货运

Stewart Stockton的年度业绩检查时间到了，信息系统助理副总裁Monica Gibbons准备审查，她检查了Stewart过去一年的工作任务和成绩。Stewart是公司中一个积极进取的系统分析员，她想就如何发展他的事业给他实实在在的忠告。例如，她知道他有成为一个项目经理的强烈愿望和同意增强责任性，当然，他的愿望与公司的要求是一致的。

Customer Load Trucking（CLT）是全国货车运输公司，它擅长高技术设备的快速移动。随着通信和计算机行业的快速发展，CTL面临着来自顾客越来越多的压力，要求更快速、更准确的运送货物。CLT为了能够在瞬间计划和跟踪发货情况已经计划几个新的信息系统。信息系统的专家对跟踪并不很感兴趣，而且由于人才短缺，CTL决定对这些新项目不雇佣项目经理，但在组织内设立强有力的项目经理。

Monica检查了Stewart的履历，发现他在担任上一个项目组的领导时工作很出色，他兼任4人小组的领导和系统分析员，涉及系统分析、设计和程序设计，以及其他3个小组成员的管理工作。他帮助制订了项目进度表，并能使自己的小组按计划工作，工作记录也表明他的小组的工作质量比项目中其他小组要好。她在想应该给他什么忠告以帮助他发展他的事业，她也在考虑现在是否该给他自己的项目。

1. 你认为CLT决定从原有雇员基础上发展培养它自己的项目经理是不是一个好办法？您给CLT什么建议以确保它在公司内有强有力的项目管理技能？

2. 你为Monica制订什么样的标准来评价Stewart（和其他潜在的项目经理）是否准备担负项目

管理职责?

3. 你如何为新项目经理安排工作以确保高水平的成功，或至少增加高水平成功的可能性?

4. 如果你是Monica，你给Stewart什么样的建议来安排他的职位并实现他成为一个项目经理的目标?

对落基山运动用品商店实例的再思考



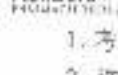
本章确定了任何新项目需要评估的项目可行性的5个领域。可是，每一个领域的可行性评估都可以看成是对项目潜在风险的评估。根据本章和第1章提供的信息，通过你对落基山运动用品商店的理解，建立一张描述RMO这个新项目所面临风险的表，包括4列标题：① 项目风险；② 风险类型；③ 风险的可能性；④ 减少风险的措施。

列出尽可能多的项目风险。风险类型是指项目具有风险可能性的领域或种类，这有助于以不同的种类考虑风险，如经济、企业文化、技术、进度和资源。本章提供了这些领域的一些风险列子。可是，还有许多其他风险会引起项目失败。所以，尽可能考虑周全，扩大每个领域潜在的风险列表清单。

很明显，其他种类的风险与大量客户支持系统和项目有关。可以想像一些公司内部的风险，如经济、市场、法律环境，等等，其他种类的内部风险也可能涉及采购或外购的部分，如开发工具，学习曲线图，购买产品质量差，不合格厂商。

通常的风险管理技术是建立一张表，确认项目最大的10个风险。针对最大的10个风险做好意外事故场景计划。项目管理组要定时地重新评估风险表以确定当前最大的10个风险，在建立表之后，确认哪些风险已列入最大的10个风险中。

关注Reliable Pharmaceutical Services



第2章讨论了基于网络的口服药品系统把它的客户保健机构直接与新的处方和记账系统相连接。考虑SDLC中顺序瀑布法开发的风险及迭代与增量方法开发的风险。

1. 考虑项目启动时的方式，是源于（a）机会、（b）问题，还是（c）指令的结果。

2. 许多系统用户（如卫生保健机构的员工）并不绝对是可靠的用户。项目失败的什么风险与这些混杂的用户团体有关?作为一个项目经理，怎样减小这些的风险?

3. 项目的有形收益有哪些?无形收益呢?有形成本和无形成呢?如何处理项目增加的卫生保健机构的成本和收益。在成本和收益分析中，会把卫生保健机构的有形成本和收益考虑进去吗?为什么?

4. 总体来讲，你认为项目采用的方法（顺序瀑布法、迭代和增量方法）在有形成本收益和无形成本收益中有区别吗?试讨论。

5. 你认为项目采用的方法在减少项目失败的风险中有区别吗?试讨论。

参考资料

Jack R. Meredith and Samuel J. Mantel, Jr., *Project Management: A Managerial Approach* (6th ed.), John Wiley and Sons, Inc., 2004.

Joseph Phillips, *IT Project Management: On Track from Start to Finish*, McGraw-Hill, 2002.

Project Management Institute, *A Guide to the Project Management Body of Knowledge*, Project Management Institute,2000.

Walker Royce, *Software Project Management: A Unified Framework*, Addison-Wesley, 1998.

Kathy Schwalbe, *Information Technology Project Management, Fourth Edition*, Course Technology, 2006.

第二部分 系统分析任务

第4章 开始分析：调查系统需求

学习目标

阅读本章后，你应具备如下能力：

- 描述系统分析生命周期阶段的各种活动
- 解释业务流程重组对分析阶段各种活动的影响
- 描述功能和非功能系统需求之间的差异
- 区分和理解不同类型的用户（这些用户都在系统需求调查对象中）
- 解释开发系统需求所必需的信息类型
- 使用文档调查、面谈、观察、原型、调查表、联合应用程序设计会议和供应商调查确定系统需求
- 讨论通过验证系统需求来保证准确性和完整性，以及使用结构化遍历技术的必要性

本章要点

- 系统的分析阶段
- 业务流程重组和Zachman框架
- 系统需求
- 系统相关者——系统需求的来源
- 信息收集技术
- 验证系统需求

山区摩托运动

Amanda Lamy是山区摩托运动公司（MSMS）的总裁和股东，同时还是摩托车爱好者和商人。MSMS总部设在Denver，是摩托运动装备的零售商，包括船只、摩托艇、越野车和摩托车。公司在Mississippi西部几乎每一个州和加拿大两个省共拥有47家店。

在过去几十年里，摩托车市场，尤其是订制车市场迅速发展起来。Amanda自己拥有3辆订制车，并在一年前决定MSMS扩展进入订制车市场的时机已经成熟。于是她开始在西部寻找业务需求和订制摩托车制造商的合伙人。

仅在一个月以前，Amanda在Tucson完成了与Abeyta订制新波器（ACC）的合伙人的协议签署。其他的需求和伙伴关系被计划在不久的将来。这一伙伴关系使得MSMS有独立分配ACC订制自行车的权力，同时给ACC带来扩大和现代化生产设备及一部分零售销售资助。作为现代化的一部分，MSMS将会建立ACC信息系统，并把这个系统用在其他的订制自行车店中。

MSMS和ACC面临着开发新型系统的困境。MSMS没有制造的经验，而ACC现有的账务

和生产控制系统将人工程序和自动化支持并流了，主要是在Microsoft Excel表格上。有一点计算机知识或经验的业务专家及MSMS内部的计算机专家都不能建立新系统这个业务。现买一个系统不可选，因为市场太小了，没有供应商可以提供。

通过与一个资深软件开发咨询公司的协商，MSMS决定在3天内开一个联合应用程序设计会议。会议参加者包括ACC的业主、会计和销售人员，Amanda，她的首席信息官员，负责运作的副主席，少数MSMS程序员和开发员。咨询公司的参与者包括会议领导人、在订制和小批量生产控制系统方面有经验的开发员，技术支持员工及管理助理。会议在咨询公司办公室的计算机管理会议室进行，那里有专门的原型开发和部署的服务器，还有合适的案例和关联工具。所有参加人员都住在附近的宾馆里，以便最大化可利用的工作时间。

会议开局不利，由于ACC代表缺乏计算机经验，他们在高自动化会议室内感到特别不舒服，而且害怕完成不了手上的任务。早上大部分时间都花费在让ACC代表和其他参与人员适应这个过程上。会议领导指派一位MSMS员工在整个会议中为ACC员工执行所有的“动手做”计算机任务，参与者通过早上和延长的午餐中的交流产生了友情。

在每个人都适应之后就开始认真工作了。第一天，参与者详细叙述系统的整个范围和主要功能，并详细描述ACC和MSMS财务功能之间的联系，采用简易的能够产生相互作用的原形模型。第二天，把注意力放在市场和生产方面。早上，团队制作了表述MSMS商店销售员如何展示订制自行车选择的模板，为创造更详细的订单做准备。大部分参与者吃午饭时，两位MSMS开发员浏览了订制自行车的图片，为在线设计程序建立了用户界面原型。

下午，把原型扩展成更为完整的订购入门系统，这个系统具有ACC所需的细节，然后安排和完成生产。第二个晚上，MSMS开发员工作到深夜，把这一原型具体化。第三个早上，参与者评价原型并给出了进一步改进的建议。他们把早上的剩余时间和大部分下午的时间用于开发需求，以及为生产管理系统设计规则，规则包括规划、部件管理，以及说明时间和材料。经过一小时对所有已确定需求的评议，会议结束，并做出了设计决策。决策包括一份公开的开发清单，草拟的规划和项目预算。

联合应用程序设计(JAD)为整个项目提供了一个良好的开端。关键系统相关者的参与使得聚集信息、制造原型、识别关键需求成为可能。尽管在联合应用程序设计会议中定义的一些需求之后还需要修改，但有些需求已经植入到系统中，不再需要修改了。

概述

在前面的章节中，你已经了解到系统开发主要由4个阶段组成：计划、分析、设计和实施。本章将主要讨论一些在分析阶段使用的技巧及相关的任务。正如在第1章中所讨论的那样，分析员在系统开发中可以使用许多技巧。用来完成系统分析的两个技巧是：①为系统需求调查寻找事实；②根据系统需求为业务过程建模。尽管分析阶段包括许多相关活动，但这两个具体的技巧对分析而言是至关重要的。在这一章，你将学会一些寻找事实和进行调查的分析技巧，在以后的章节中将介绍如何为系统建立模型。

在寻找事实和调查活动期间，你将掌握业务过程和日常事务的一些细节。事实上，在这些活动期间的目标就是尽可能多地了解业务活动的细节，使你可以像和你面谈的用户一样熟悉业务活动。为什么要成为业务领域方面的专家呢？这是因为只有在熟悉了业务活动后，才能确保系统满足了业务要求，也只有到那时才能使你那宝贵的组合技巧发挥作用。你为问题的解决带来了新的方法，并且通过使用新的、更好的信息技术方法来完成业务目标，从而可以为公司做出更大的贡献。现在，许多用户非常熟悉他们完成工作的方式，以至于他们不能想像有更好的、更先进的方法来获得同样的最终结果。你的技术知识，以及你刚掌握的有关

问题域的知识可以为实施业务过程带来独特的解决方案。

成为问题领域专家的另一个好处是在用户中建立了可信度。在问题领域的知识增加了你的建议可信度，同时也可以确保你的建议满足用户的具体需求。在新系统开发期间，你可能会提出许多处理日常业务事务的提议和建议。新信息系统的安装通常要求在业务操作程序方面做出重大改变。如果你能遵循用户的业务操作，那么他们就更有可能接受你的建议。否则，你也许只会被当做不能真正理解问题的门外汉。

接下来的部分首先给出系统分析阶段各种活动的一个概述。这部分定义了系统需求，并探究了分析员所遇到的几种不同类型的系统需求。同时也介绍了分析员使用传统方法和新改进的方法来进行业务过程学习，以及对关信息收集的一些技术。最后，本章将探讨对收集到的信息及在分析阶段活动中所构建的模型进行质量控制的必要性。

4.1 更详细的分析阶段

你会发现，在公司中遇到的所有关于系统开发方法的描述以及所有专门的系统开发方法论都与生分析和设计阶段的一些活动相似（如图4-1所示）。当然，不同的开发方法需要利用不同的技术来实现他们的活动，在许多情况下，他们只是名字不同而已，其潜在的功能任务都是一样的。在有些时候，需要创建不同的模型来完成一项活动，但是这些活动通常都包含了对同一关键问题的回答。



图4-1 分析阶段的活动

分析阶段需要非常详细的定义：信息系统需要完成什么来为公司提供所期望的利润，应该提出许多方案，并且从其中选择最优的解决方案。以后，在系统设计阶段，再对选中的方案进行详细的设计。在分析阶段，6项活动必须全部完成。它们之间是互补的，并且通常同时完成。例如，分析员不断地收集信息，并根据这些信息定义需求。这两项活动始终贯穿于整个分析阶段，而不仅仅发生在分析开始。

4.1.1 收集信息

分析阶段会收集大量信息，系统分析员从今后该系统的使用者那里，或者通过和他们交谈，或者通过观察他们的工作可以得到一些信息。通过回顾计划文档和方案说明，分析员可以得到另外一些信息。现有系统的支持业要进行仔细的研究。分析人员可以通过参考类的公司（尤其是供应商）在遇到相似的问题时是如何处理的，从而获得一些额外的信息。简言之，分析员需要和几乎每一个今后要使用新系统或已经使用类似的系统的人进行交谈。并且，他们还要阅读所有与现有系统有关的资料。

开始时，分析员常常会低估可以从用户所做的工作那里获得的信息量。分析员必须成为系统所支持的业务领域的专家。例如，如果要实现一个订单录入系统，就必须熟悉订单的处理方法（包括计算在内），如果要实现一个贷款处理系统，就需要精通用来验证信用的一套

规则，如果为银行工作，就要把自己当做一个银行家。最成功的分析员能够完全融入到他们公司的主要业务之中。

分析员需要收集技术信息。分析员应该通过对现在的用户和未来的用户的活动的区分和理解，以及对目前和将来活动的发生地点的区分，并通过区分公司内部和外部所有其他系统的接口来理解现有的系统。除此之外，分析员需要确定用来满足系统需求的软件包。这些将在本章的后面有专门的讨论。

在完成这项活动时，应该回答的关键问题是：我们是否已经拥有了全部的信息来定义系统所必须完成的工作？

4.1.2　定义系统需求

如果所有的必要信息已经收集到了的话，把它们记录下来是很重要的。其中一部分信息是描述技术需求的（例如，所需的系统性能或期望的交易数目等）。其他的信息包含了功能需求——需要系统完成什么样的工作？定义功能需求并不是简单地写下一些事实和数据，而是要创建许多不同类型的模型来帮助记录和联系系统需求。

对分析员来说，建模过程是一个学习过程。随着模型的建立，分析员可以越来越了解系统。在对各种信息进行收集的同时，建模过程也在继续。这一期间，分析员要不断与最终的用户一起来确保每个模型的完整性和正确性。此外，分析员还要研究每一个模型，对它们进行添加、重排，并且还要检查它们彼此是否合适，就算分析员相当确定系统的需求，而且已经完全说明了，一份附加的信息说明仍可能需要更多的修改，也要再一次开始精练的过程。建模需要持续相当长的一段时间，并且通常没有明确的结束标志。这种不确定性会使程序员觉得有些不适应，但这是无法避免的。

我们需要开发两种系统模型。需求模型（或模型的集合）是一种逻辑模型，逻辑模型能够很详细地显示系统需要完成哪些功能，而不依赖于任何技术。从中立的角度上来看待技术，开发组首先要将精力集中在“需要什么”上，而不是“它将采用什么形式”。例如，某个模型可以将系统的输出规范成一个数据元素列表，而无须考虑其在纸张或屏幕上显示的形式。这种模型所关注的是用户需要什么样的信息。而另一方面，物理模型展示了系统实际上是如何实现的。输出的物理模型将会包括形式上的各种细节。

逻辑模型：能够很详细地显示系统需要完成哪些功能，而不依赖于任何技术的模型。

物理模型：表明系统将如何真正实现的模型。

逻辑模型和物理模型之间的区别是一个用于区分系统分析和系统设计的关键概念。通常系统分析包括创建详细的逻辑模型，系统设计则包括创建详细的物理模型。在分析阶段所产生的各种设计方案是物理模型，但是它们不是很详细。

专用模型的创建与用来进行系统分析的技术有关。现代结构化分析技术采用数据流图（DFD）和实体—联系图（ERD）。信息工程方法采用依赖图和实体—联系图。面向对象的技术产生了类图和用例图。关于这些模型的例子在第5、6、7章会有详细的描述。

在完成这项活动时，应该回答的关键问题是：我们需要系统做什么（在细节上）？

4.1.3　需求的优先级划分

一旦我们已经充分理解了系统的需求，并且需求的细节模型也已经设计完成了，这时确定哪种系统和技术被需求对系统来说是非常关键的。有时候，用户建议了一些额外的系统功能，但是这些功能不是必需的。然而，用户和分析员都要问问自己到底哪些功能是真正重要

的，哪些功能也很重要但却并不是绝对需要的。这时候，那些理解公司和用户所做工作的分析员在解决这个问题上时会更有洞察力。

为什么要对用户提出的功能进行优先级的划分呢？因为资源往往是有限的，分析员必须常常准备判断系统的作用域。所以，了解究竟什么是绝对需要的非常重要。除非分析员仔细的评估优先级，否则系统的需求会随着用户不断提出的要求而不断地膨胀（这种现象叫做需求扩充）。

在完成这项活动时，应该回答的关键问题是：系统要完成的最重要的事是什么？

4.1.4 发现原型及可行性

在系统分析过程中，构建新系统的一些原型是很有价值的。在分析过程中，构建原型（通常称为发现原型）的主要目的为了更好地理解用户的需求。发现原型的构建不是为了用来实现所有的功能，而是用来检验业务需求某种实现方法的可行性。许多时候，用户总是试着不断提高商务处理效率或使处理过程流线化。因此，为了简化对新的商务处理过程的调查工作，分析员需要构建原型。通过使用简单的投影或报告，分析员可以与用户讨论新系统如何支持新的处理过程，他们可以示范新系统的新的商务处理过程。这些原型有助于用户发现一些以前从未考虑过的问题，可以使他们（包括分析人员在内）跳出原来的思维模式。

如果系统含有一些新的技术的话，在项目刚开始的时候对这种技术是否具有解决商务需求的能力进行评价是非常重要的。那样项目组才可以确保技术的可行性。利用原型，可以验证该技术所能够实现的功能。同样，如果系统含有一些创新技术的话，用户在定义他们需求的时候，需要把新技术所能提供的各种可能性可视化。利用原型，可以满足这样的要求。

在系统分析阶段中的原型构建有助于回答两个关键问题：可以证明这种技术能够实现想让它完成的那些功能吗？还有同样重要的：是否已经构建出一些原型，可以使用户完全理解新系统的潜在功能？

4.1.5 产生和评估候选方案

对系统的最终设计和实现会有各种不同的方案。所以仔细地定义并评估所有的可能性是很重要的。当需求的优先级确定了以后，分析员可以产生几个可选方案，消除一些不重要的需求。此外，技术同样也可以给系统带来一些解决方案。除了上面要考虑的那些因素以外，类似于是自行建立开发系统还是让外面的公司来进行开发的决定也影响着最终的开销。而且，一个或多个成熟的软件包就可能满足用户的所有需求了。

很明显，对项目组来说，有很多可以参考的方案，每一种方案都需要在一个高的（概括的）层次上进行描述或建模。每一种方案都有它自己的开销、利润，以及其他一些特点需要进行认真的衡量和评估（就像第3章中所描述的可行性研究那样），然后才可以选出最好的方案。选择一种方案并不像听起来那么容易，因为开销和利润是很难计算的。并且许多设计细节还不是很确定。在项目规划阶段，分析员始终考虑的是项目总体的可行性。在分析阶段才确定每种方案的可行性。

在完成这项活动时，应该回答的关键问题是：创建系统的最好方案是什么？

4.1.6 和管理部门一起复查各种建议

收集信息、定义需求、划分需求的优先级，发现可行性的原型，以及产生评估各种方案，

所有这些活动都是并行执行的。分析阶段的最后一项活动（和管理部门一起复查各种建议）通常是在所有分析活动已经完成或将要完成的时候进行的。管理部门应该可以通过定期的项目报告了解整个项目的进程。最后，项目经理需要提交一份解决方案并从管理部门那里获得最终的决定。分析人员所要考虑的问题如下：项目是否应该继续下去？如果要继续的话，哪一个是最好的方案？如果已经有了推荐的方案，完成这个项目修订后的所需预算和进度表又是什么？

向资深的主管人员提交一份推荐书是整个项目管理上的一个主要的检验点。每一个可选方案（包括已取消的）都必须探究。尽管项目中大量的工作事先已经进行了调研，但是取消这个项目仍可能是最好的选择。也许利润并不像原先设想的那么多，也许开销要比原先设想的多得多，或者是由于千变万化的业务环境，从项目提出以后公司的目标可能发生了改变，使得这个项目对公司来说并不重要了。一旦出现这些情况，最好的方案很可能是取消这个项目。

如果这个项目值得去做，项目组已经详细地做了关于系统需求的文档说明和建议的设计方案，项目经理就要制订出一份更加准确的预算估计和进度表。如果高层的管理者理解继续项目的基本原理，他们就可能会提供所申请的各种资源。要牢记，将项目带入设计阶段从来不是自动完成的。好的项目管理技术需要对项目的可行性进行反复的评估，并且需要经常进行正式的管理总结。

在完成这项活动时，应该回答的关键问题是：我们应不应该继续设计和实现我们提出的系统？

分析阶段的每一项活动都包括系统相关者和任务，并且包括回答一个或更多的关键问题。这些活动和关键问题可以用图4-2来描述。

分析阶段的活动	关键问题
收集信息	是否已经推荐了全部的信息来定义系统所必须完成的工作？
定义系统需求	需要系统做什么？
需求的优先级划分	系统要完成的最重要的事是什么？
构建可行性的发现原型	可以证明这种技术能够实现制计它完成的那些功能吗？是否已经构建出一些原型可以使用户完全理解新系统潜在功能？
产生、评估方案	创建系统的最好方案是什么？
和管理部门一起复查各种建议	应不应该继续设计和实现我们提出的系统？

图4-2　分析阶段的活动及相应的关键问题

4.2　业务流程重组和ZACHMAN框架

第1章中讨论的业务流程重组（BPR）是20世纪90年代兴起的一场运动，它已成为许多新的信息系统创建的推动力量。随着全球经济竞争加剧，许多公司发现必须重新思考他们的内部结构和业务过程。业务流程重组是一种根本的战略思想，它将公司内部的处理过程流线化，并使之尽可能的高效运转。以前，旧的业务流程规则是："如果还没有坏，就不要修理。"一种较新的思维方式则是："总有更好的解决方法，让我们改进它。"业务过程重组（BPR）用一种更革命的思想，从而把这种方式推进得更远。这种思想是："让我们对基本方法提出质疑，从而发现一种能带来巨大和深刻的提高的全新方法。"在过去的10年中，随着全球竞争日益激烈，许多公司发现，有必要彻底重新思考关于他们该如何进行自己的业务，诸如此类的一些最基本的问题。

现代信息技术使得BPR可以以多种方式发挥作用，包括存储和处理大量、完全的数据，无论何时何地在需要的情况下提供高质量信息，在组织单位和职能部门之间支持快速的交流，以及将原来的人工决策变为自动决策等。例如，许多公司在为其客户提供个性化服务上已展开攻势。客户关系管理（CRM）这个时髦的词语在第1章已讨论过了。许多系统供应商都提供专门的软件来支持CRM。要实现这些目标，首先必须要有高度的自动化技术来支持处理它所产生的大量细节信息，其次要有非常高效的业务处理过程来利用这一技术。

这种巨大改进的经典例子之一就是福特汽车公司北美分部的账目支付功能。在20世纪80年代中期，账目支付部门雇佣了500多人。最初的项目打算开发一个信息系统来实现生产率20%的增长率。然而，当项目小组和管理人员寻找应用自动化改善性能时，他们发现马自达公司（福特公司在该公司拥有20%的股份）在账目支付部门仅仅雇佣了5个人。尽管相对而言福特公司是一个大得多的公司，但是它仍然拥有超过马自达公司100倍的人来完成基本上同样的功能。用一种更好的眼光来看待可能发生的事情，项目小组使用一种更高的自动化水平完全重新设计了支付功能。本质上，账目支付功能被包含在一个更大的购买功能里，从而使得从购买之时起跟踪支付变得自动化。在项目结束时，福特公司只用100多人就完成了账目支付功能，最终实现了400%的增长。

将一个系统开发项目作为BPR项目来解决将会对系统分析阶段活动带来巨大的影响。信息收集及定义系统需求的活动将不再过多地关注当前系统的运作方面，而是更多地关注于发现新的和一些从未尝试过的方法，以达到同样的目标，满足同样的业务需求。在BPR项目中，系统分析员更倾向于脱离项目最初的规模，而追求系统需求的相关信息及潜在的改善和效率。在收集信息及产生和评估候选方案时，分析员从大量内部和外部系统相关人员处提取输入信息。由于业务流程及相关技术在BPR项目中可能会发生较大的改动，因此应该更加重视可行性分析以及发现的原型建立。

20世纪90年代另一个重要的成果是企业架构中Zachman框架的开发。John Zachman先前在IBM公司开发了一个全面的信息系统结构支持处理方法，正如在第1章中所提到的，这一开发的重点放在应用结构和技术结构的概念上。Zachman框架有助于为整个组织调节系统需求和组件，而且它涵盖了系统组件的全局观点和详细说明。这一框架促进了计划者和分析员研究业务过程重组和互相协调的数据。

框架如图4-3所示。每一栏把系统组件抽象出来——数据、功能、网络、人、时间和动机。项目参与者关注每行显示的这些抽象词对应的各个方面：计划者的范围、业主的业务模型、设计者的系统模型、建造者的技术模型和程序员/承包商的详细陈述。例如，人们关心计划应用的范围，那么就会问什么对业务重要（数据），组织如何工作（功能），事情在哪发生（网络），谁参与（人），事情何时发生（时间），为什么要做事情（动机）。项目团队就是识别他们应该生产的指定成果或模型的细胞。在项目之后，因为要详细设计系统，从事技术模型的构造者要把重点放在物理数据模型（数据）、系统模块设计（功能）、陈述结构/用户界面（人）、控制结构（时间）及规则设计（动机）上。技术模型构造人员的创造是在前行业务模型和系统模型的基础上的。

为信息系统聚集信息和定义需求的时候，Zachman框架为系统化定位所有关键组件和透视提供了快捷的方法，这对任何一个系统都是十分重要的。

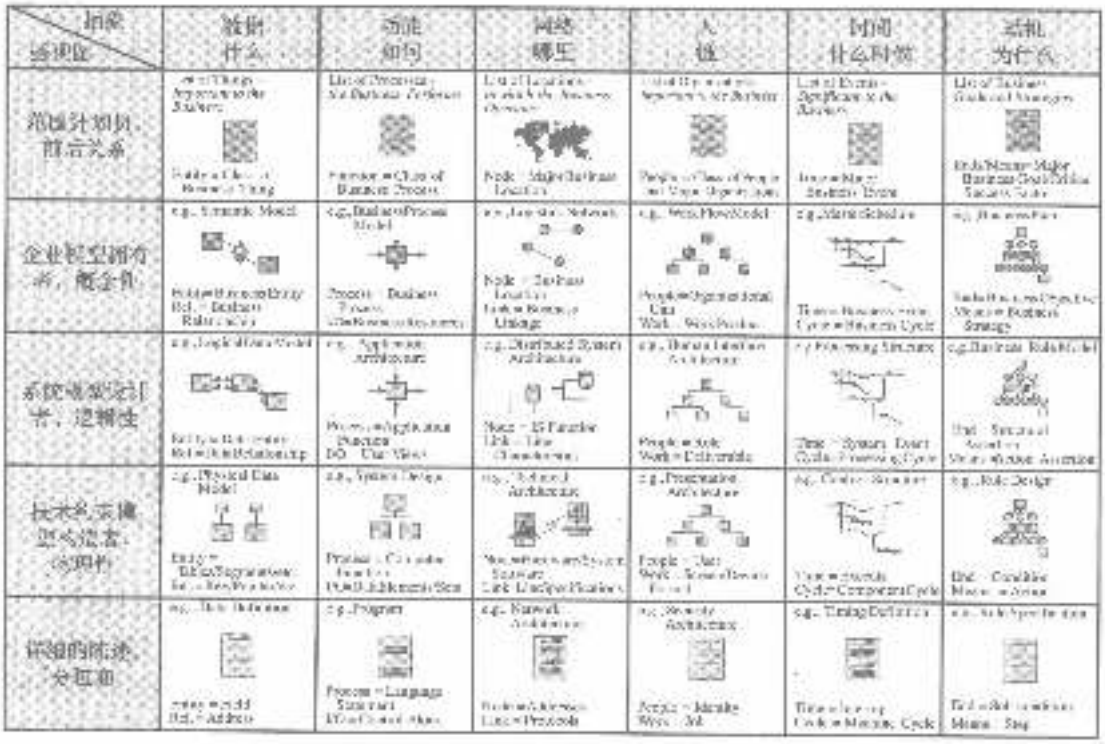


图4-3　企业结构的Zachman框架（改编自框架改进Zachman研究所）

4.3　系统需求

系统需求是新系统必须完成的功能及其局限性。通常，分析员把系统需求分为两类：功能需求和技术需求。回想一下计划阶段的活动之一就是确定系统的作用域。在此活动中，分析员确定一组系统功能。在分析期间，分析员详细地定义和描述这些功能，换句话说，分析员把这些高层功能分解为详细的系统需求。

系统需求：系统所提供功能的详细定义。

功能需求是系统必须完成的活动，也就是系统将要投入的业务应用。功能需求直接来自于计划阶段确定的系统功能。例如，如果你正在开发一个工资系统，那么需要的业务应用也许包括这样一些功能：书写支票、计算佣金数量、计算工资税、维护雇员的相关信息和向IRS报告年终预缴减免。这些就是新系统的功能。确定和描述所有这些业务应用需要花费大量的时间和精力，因为功能列表及其关系非常复杂。

功能需求：描述系统必须完成的活动或过程的一种系统需求。

功能需求是根据公司进行业务交易的过程和业务规则确定的。有时，这些过程详细的记录在文档中，因而易于确定和描述。一个现实中可行的例子是所有的新雇员必须填写一张W-4表格来输入他们在工资系统中的相关税性信息。其他一些业务规则也许非常隐蔽，难以发现。RMO就是这种类型的一个例子。在RMO中，每个通过电话订货的顾客如果订购了某些促销商品，那么就可以获得额外的2%的折扣。这些由处理电话订单的职员出售的促销商品由于没有公开做广告，因此其折扣率是特别的。发现类似这样的规则对于系统最终的设计是非常关键的。如果开始没有发现这些规则，那么你设计的系统也许只允许使用固定的折扣率。当客户

系统开发过程的后期会发现你的设计不能满足这些规则的要求。

非功能需求是这个系统的具有特征。它不同于系统必须完成或支持的行为。有许多不同类型的非功能需求，包括如下几部分：

非功能需求：是这个系统的固有特征，它不同于系统必须完成或支持的行为，例如技术、性能、可用性、可靠性及安全性。

- 技术需求描述了与组织的环境、硬件和软件相关的操作特征。例如，新系统的客户组件可能要求运行在用Windows操作系统和IE的笔记本或台式机上，服务器组件可能要求必须使用Java编写，并且相互之间使用组件交互标准CORBA（公用对象请求代理结构）或SOAP（简单对象访问协议）进行通信。

技术需求：是一种系统需求，描述了与组织的环境、硬件和软件相关的操作特征。

- 性能需求描述了与工作方法相关的操作特征，比如生产能力和响应时间。例如，系统的客户部分可能要求在所有的屏幕上都是半秒的响应时间，而服务器组件则可能需要在同样的响应时间内支持100个并发的客户会话。

性能需求：是一种系统需求，描述了与工作方法相关的操作特征，比如生产能力和响应时间。

- 可用性需求描述了与用户相关的操作特征，比如用户界面、工作流程、在线帮助及文档。例如，基于Web的界面可能要求符合整个组织范围内的图形设计准则，如菜单布局和格式、色彩设计、组织标志的使用及必要的法律免责声明。

可用性需求：是一种系统需求，描述了与用户相关的操作特征，比如用户界面、工作流程、在线帮助及文档。

- 可靠性需求描述了系统的可靠性，比如系统出现服务损耗、不正当处理，以及错误检测和恢复。可靠性需求通常是性能需求的一个子集。

可靠性需求：是一种系统需求，描述了系统的可靠性，比如服务损耗、不正当处理，以及错误检测和恢复。

- 安全需求描述了哪些用户可以在什么样的条件下执行系统功能。比如，对某些系统输出的访问可能受限于特定级别的管理人员或某些部门的员工。一些来自家庭的访问需要得到授权，而其他的一些访问只是来自该机构的本地网络。安全需求也可以应用在网络通信和数据存储等两个方面。例如，一个公司可能需要在通过Internet传送数据的时候要使用到加密技术，并且通过用户名和密码机制来控制对数据库服务器的访问。

安全需求：是一种系统需求，描述了用户对特定功能的访问以及访问的条件。

对于新系统的完整定义，功能需求和非功能需求这两种类型都是必不可少的，并且在系统分析阶段，它们都包含在系统需求调查中。功能需求通常记载在图表和文本模型中（和第5章～第7章描述的一样），而非功能需求则通常记载在针对模型的叙述性描述里。

4.4 系统相关者——系统需求的来源

系统功能需求信息的主要来源是新系统的各种系统相关者。系统相关者是对系统的成功感兴趣的人。通常，把系统相关者分为三类：① 用户，那些实际使用系统处理日常事务的人；② 客户，那些购买和拥有系统的人；③ 技术人员，确保系统运行在公司的计算机环境下的人。图4-4显示了对新系统感兴趣的各种系统相关者。前面我们已经讨论了用户和客户之间的差异，在分析阶段，分析员还需要考虑到技术人员。在决定系统需求时最重要的步骤之一就是确定各种系统相关者。过去，由于在项目中只包括了一些系统相关者，并且系统仅仅是为这些人设计的，因此随着新系统的建立产生了一些问题。因此，首要任务之一就是确定对

新系统感兴趣的各类系统相关者。第二个任务就是确保从各类系统相关者中识别出关键的人作为业务领域的专家。

系统相关者：对新系统的成功感兴趣的所有人。

4.4.1 用户

用户角色，也就是系统用户的类型，应该从两个方向进行定义：水平方向和垂直方向。在水平方向上的意思是说分析员必须在业务部门中寻找信息流。例如，一个新的库存系统也许将影响进货部门、仓库、销售部门和生产部门。你必须确保这些部门的每个人向你讲述他们的需求。销售部门可能提供一些需求信息来帮助你确定系统什么时候以及如何更新系统库存量，或者可能向你提供在销售后，发货前提交一定数量的某种产品的信息。生产部门也许需要库存系统的信息来帮助制订生产计划。水平方向表明许多不同的部门，甚至那些看起来和新系统无关的部门，都应该包括在系统需求定义中。



图4-1 对新系统开发感兴趣的各种系统相关者

在垂直方向上，我们需要职员、中层管理人员及高层管理人员提供信息需求。这些系统相关者中的每一个都将对系统有不同的信息需求，因此必须在设计时把这些信息需求包括在内。接下来的部分讲述了垂直方向上各种用户的特征和信息需求。这些相同的特征也适用于水平方向上的各个部门。

1. 业务用户

业务用户是使用系统处理公司的日常事务的人。通常把这些操作称为事务。事务是公司内完成的一项工作，例如"输入订单"。在第1章，介绍了事务处理系统是一个处理业务操作的系统。业务用户提供了日常的业务事务信息，同时他们也提供了系统应该如何来支持这些日常事务的信息。

事务：在一个组织中完成的一件工作或一项活动，这样一件单一的事项。

2. 信息用户

信息用户是需要从系统中获得现有信息的人。他可以是操作用户也可以是其他人。在某些情况下，企业也许想让信息对顾客是直接可用的。然而，系统不允许顾客在业务交易中输入信息，而仅允许其查看某些信息。查询用户可以向分析员提供这样一些信息：哪些信息是每天、每周、每月或每年都可以使用的，信息使用哪种格式对用户而言最便于浏览。

3. 管理用户

管理人员负责使公司高效的完成每天的日常事务。因此，他们需要从系统中获得统计和概要信息。管理人员帮助分析员回答如下类型的问题：

- 系统必须生成哪些类型的报表？
- 系统必须维护哪些类型的业绩统计数字？
- 系统必须保存哪些类型的大量信息？新系统必须支持多大数量的交易？

- 系统中是否有足够的控制来避免错误和人为破坏？
- 可以向系统请求获得多少信息？这些请求的频率可以有多快？

4. 主管用户

一个公司的高级行政人员不仅对刚刚讨论的日常问题感兴趣，更对企业的战略规划问题感兴趣。通常他们想从系统中获得信息，以便他们能够比较清楚利用是否得到了全面改进。他们也许想把系统和其他系统连接起来，从而使得系统可以向他们提供业务发展趋势和方向等方面的战略信息。

5. 外部用户

现在越来越多的系统允许外部实体直接访问系统。客户可以通过互联网直接访问系统。供应商可以通过访问系统来检查库存并初始化账单交易。由于这些用户不是公司的常规人员，所以他们比较难以区分和访问。然而，现在他们属于系统开发所需要的重点考虑对象。

4.4.2 客户投资相关者（Stakeholders）

尽管项目小组必须满足用户的信息处理需求，但它也有责任满足客户的需求。在第3章把客户定义为给项目提供资金的人或团体。在许多情况下，客户和主管用户是同一组人。然而，客户也可能是单独的一组人。例如，客户可能是母公司的理事会或董事会。把客户包括在重要的系统相关者列表中是因为项目小组必须在项目的整个开发过程中始终向客户提供项目进展的概要情况。客户或领导委员会的直接代表通常也负责批准或者决资金的使用。

4.4.3 技术人员

尽管技术人员并不是真正的用户群，但他们是许多技术需求的来源。技术人员包括建立和维护公司计算机环境的人。这些人在诸如编程语言、计算机平台和其他设备方面对项目提供帮助。对某些项目来说，项目小组始终包括一组技术人员。而对另外一些项目来说，只在需要时才把技术人员包括在内。

4.4.4 RMO的系统相关者

为了说明系统相关者对系统的不同看法，下面介绍为RMO开发的客户支持系统。

调查系统需求的一个重要部分是确定所有的系统相关者。在收集信息时，如果没有对一些用户、客户和重要技术人员进行咨询，那么所得到的需求信息将是不完备的。在RMO中，新的订单处理系统的操作用户不仅包括处理邮件订单的职员，而且也包括通过电话处理订单的内部销售代表。他们对系统应该向他们提供什么功能持有不同的观点。销售代表关心为顾客查寻产品信息，证实产品的有效性和确定发货日期。负责邮件订单的职员关心的是把订单信息扫描进系统，而不是通过键盘把订单信息输入系统。仓库管理工人（他们负责在收到订单后装载货物）需要如下信息：装载完毕的订单信息、将要装载的订单信息、延期订货的订单信息。此外他们也需要标准操作界面来实现订单汇总发货和打印装货单。

John和Liz Blankens作为公司细有者，对已订购和装载的产品报表特别感兴趣。此外，他们对现金产品的季节性趋势也很关心。对于提供运动设备的企业，紧跟商品潮流而在淡季时退出是非常重要的。

RMO客户支持系统的部分开发资金来自RMO的内部流动资金。然而，资金也可以通过银行的专用信用部门获得。对于季节性需求，RMO通常有一个短期贷款额度。由于对资本货物来说，这个项目是一项比较长期的投资，因此Blankens夫妇为新系统争取到了另一种贷款。由于投资银行对项目的成功也非常感兴趣，因此在这种情况下，项目小组成员至少与银行人员

由谁来看银行想要系统保存资金信息的哪些特殊格式。

最后，由于系统将运行一些新技术——Internet和分布式系统——所以要求技术人员有大量的相关知识。图4-5所示为RMO公司的上层成员，在图中指出了需要包含在需求定义中的一些人。灰色的位置表明了主管人员和中层管理者，他们都作为系统相关者被涵盖在内。项目经理将列出所有与系统需求定义相关的用户的名单。这张公司图只是一个开始，其他的部门经理和一些关键雇员也将被添加进来。

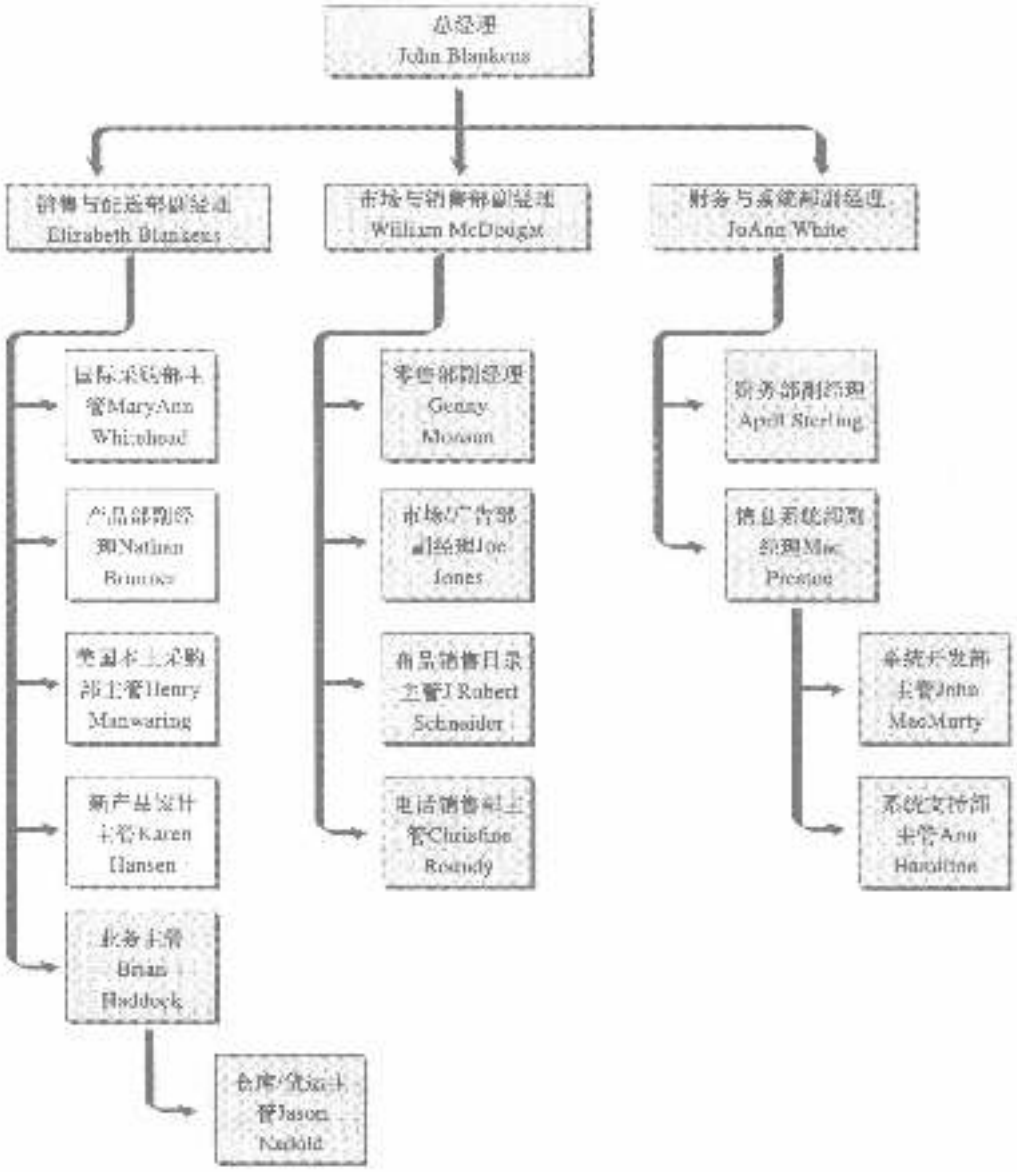
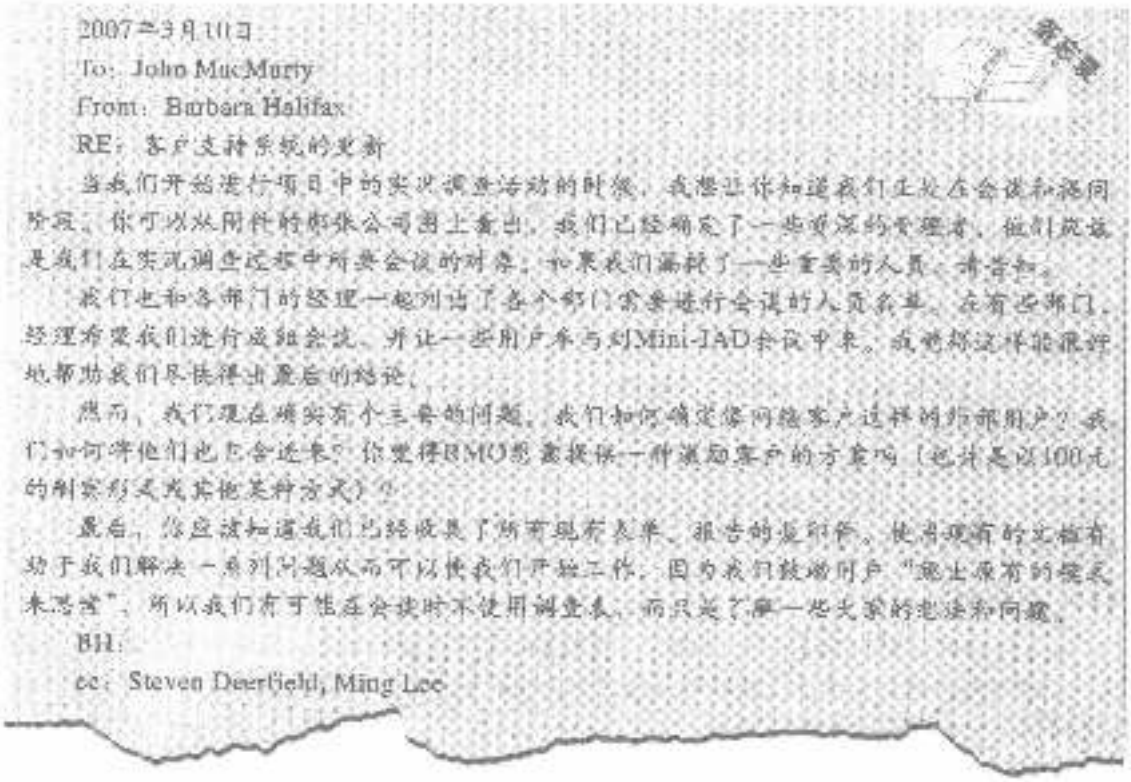


图4-5　表明RMO公司新系统的系统相关者

项目组如何确定要和哪些用户进行会谈呢？通常这是一个很难回答的问题。不过可以通过分析新系统的作用域来开始这个过程。在确定系统的作用域以后，工作组必须认真分析出所有以任何方式从系统中获取所需信息的人。这时，宁可多包含一些相关的人，也不能漏掉

一些重要的需求来源。Barbara Halifax给John MacMurty发了一份备忘录向他报告她对CSS相关者的划分的最新进展及她对信息收集的规划（见Barbara的备忘录）。



2007年3月10日

To：John MacMurty

From：Barbara Halifax

RE：客户支持系统的更新

当我们开始进行项目中的实况调查活动的时候，我想让你知道我们正处在会谈和提问阶段。你可以从附件的那张公司图上看出，我们已经确定了一些资深的管理者，他们就是我们在实况调查过程中所要会谈的对象。如果我们漏掉了一些重要的人员，请告知。

我们也和各部门的经理一起列出了各个部门需要进行会谈的人员名单。在有些部门，经理希望我们进行成组会谈，并让一些用户参与到Mini-JAD会谈中来。我觉得这样能很好地帮助我们尽快得出最后的结论。

然而，我们现在确实有个主要的问题。我们如何确定像网络客户这样的外部用户？我们如何将他们也包含进来？你觉得RMO想要提供一种调动客户的方案吗（也许是以100元的制宾形式或其他某种方式）？

最后，你应该知道我们已经收集了所有现有表单、报告的复印件。使用现有的文档有助于我们解决一系列问题从而可以使我们开始工作。因为我们鼓励用户"跳出原有的镜头来思考"，所以我们有可能在会谈时不使用调查表，而只是了解一些大家的想法和问题。

BH：

cc：Steven Deerfield, Ming Lee

4.5 信息收集技术

系统开发的分析阶段的目标是理解业务功能和获得系统需求。在此过程中经常发生的问题是，是否需要研究和记录对现有系统的描述，或者是否仅仅需要记录下新系统的系统需求。当结构化方法，以及在第3章中介绍的其他方法第一次提出时，系统分析员首先记录对现有系统的描述，然后从对现有系统的描述中推断出新系统的系统需求。当时，系统需求的开发过程由4个步骤组成：① 确定现有系统的物理过程和活动；② 从现有物理过程中提取出业务逻辑功能；③ 为将在新系统中使用的方法开发出业务逻辑功能；④ 定义新系统的物理处理需求。这种方法的缺点是需要花费大量的时间。根据长期经验结果，存在另一个问题，即系统开发者常常只是简单地将现有的系统自动化，其结果是不论现有的系统效率有多低，系统开发者只是将现有的程序简单地原地进行自动化。

在如今这样一个竞争激烈的社会，许多公司正在使用新的信息技术来增加其自身的竞争优势。许多公司完全重新设计并使其内部处理程序流线化，以便在公司内部、公司与公司之间充分利用新技术的优势。这种重新设计在术语上叫做业务流程重组，在本章的后面我们将会对其进行详细的讨论。

实践指导

为了避免"分析瘫痪"，在一开始就把重点放在系统需求上。■

分析的目标并未改变，但是，开发系统需求的方法提高了。有一组完备、正确的系统需求是至关重要的，但在生活节奏越来越快的今天，既没有时间也没有资金来总结所有的老系

统，记录全部的低效程序。现在的分析员通过比较现有的业务功能和新系统的需求，使用一种更加快速的方法。如图4-6所示，现在的分析活动的中心是为新系统立即开发一组逻辑系统需求。系统分析员仅在需要理解业务需求时才去检查现有系统，不会特意去定义旧系统的特定流程，也就是说，焦点放在新系统上。有时调查一下旧系统才是专业分析员应该做的，他们需要非常详细地理解业务需求（记住，“要多多沟通”），但是他们不需要掌握那些陈旧低效的方法。事实上，在目前的开发环境中，系统开发者最有价值的能力之一应该是可以对问题提出新的看法。

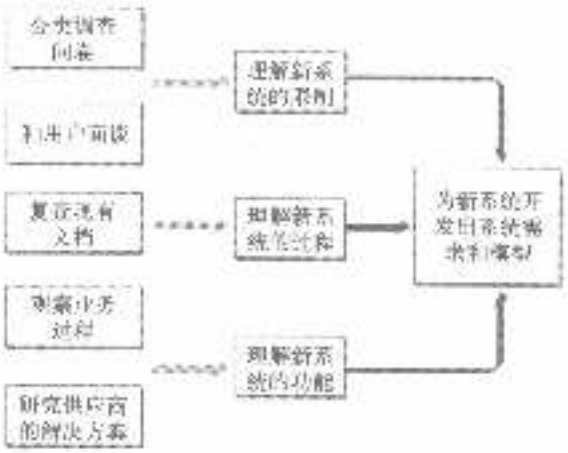


图4-6　收集信息和建模之间的关系

分析人员在收集信息的同时可为新系统开发逻辑模型。在系统设计阶段，项目组开发出物理模型（由它说明系统是如何构建的）。分析人员将会把精力放在某些主要的问题上，并使用各种方法来开发系统的逻辑模型。

4.5.1　主要问题

新系统分析员要问的第一个问题是：“我需要收集哪方面的信息？系统有哪些需求？”通常，你需要获得使你能够建立新业务系统逻辑模型的信息。在你进行系统调查时能够对你提供指导的三个主要问题如图4-7所示。

主　题	对用户来说的含义
业务过程和操作是什么样的？	你要干什么？
业务过程应该怎样完成？	如何完成它？或需要哪些步骤？
需求什么样的信息？	你要使用哪些信息？你要使用什么样的表单或报告？

图4-7　信息收集中的主要问题

1. 业务处理过程是什么样的

在第一个主题“你要干什么？”的核心是要理解业务功能，这个问题是可以进行“沟通”的第一步。在大多数情况下，用户将根据对现有系统的理解做出回答。作为一个系统分析员，你必须仔细的识别出在改进的系统中，这些业务功能哪些是重要的，哪些是需要保留的，而哪些是可能要删除的。例如，销售人员可能指出当顾客订购货物时，他们要做的第一件事就是检查顾客的历史信用记录。但在新系统中，销售人员也许从来不需要使用这个功能，因为该功能可由系统自动完成。在这种情况下，系统保留了这项功能，不过是由计算机取代了职员来进行。

2. 业务过程应该怎样完成

第二个主题“如何完成它？”是讨论从现有系统转向新系统的方法，核心是新系统应该如何支持这项功能，而不是它在现有系统下如何执行。因此，前两个问题同时进行以发现系统需求，然后根据新系统开始定义系统需求。用户通常只熟悉现有的系统，但是对系统分析员来说，超越现有的处理过程才是最关键的，他（她）必须能够使用户看到由新技术带来的业务处理方法更高效、更实用。

3. 需求什么样的信息

最后一个主题“需要哪些信息？”通过定义新系统必须支持的具体信息详细描述了第二个问题。对第二个和第三个问题的回答形成了定义系统需求的基础。对许多新的系统分析员来说，他们有一个共同的缺点就是没有对所有需要的信息进行区分。对这个问题和前一个问题来说，一定要详细，详细，再详细。为了得出正确的解决方案，分析员必须了解任何细节的不同情况。

如果把调查集中于这三个主题，那么就能够在调查中提出一些机智、有意义的问题。之后，当大致了解了模型，就能够明确地表达出另外一些有意义的细节问题。

作为一个系统分析员，必须要掌握的最重要的技巧之一就是理解用户需求的能力。当掌握了提问和建立模型的技巧后，解决问题和分析问题的能力将得到提高。记住，作为系统分析员，你的价值并不在于知道如何去建立一个具体的模型或如何用一种具体的语言去实现编程，而是在于分析和解决业务信息问题的能力。对这种技巧来说，至关重要的是知道如何不仅有效地要求获取去理解和确定系统需求的业务规则。有效的系统需求是完整的、全面的和正确的。高效的分析员能够在最少干扰用户时间内利用最少的资源很快地进行项目设计，并且可以保证通过收集到的信息能够得出完整、全面和正确的需求说明。

下面介绍了信息收集的各种方法。所有这些方法已被证明是有效的，尽管有些方法看起来比另一些方法更有效。在大多数情况下，分析员把几种方法结合起来使用，从而提高了效果和效率。这种方法的组合提供了一种全面的事实发现方法。最常使用的一些方法如下：

- 复查现有的报表、表格和过程描述
- 主持与用户的面谈和讨论
- 观察并记录业务过程
- 建立原型
- 分发和收集调查表
- 主持联合应用程序设计（JAD）会议
- 研究供应商的解决方案

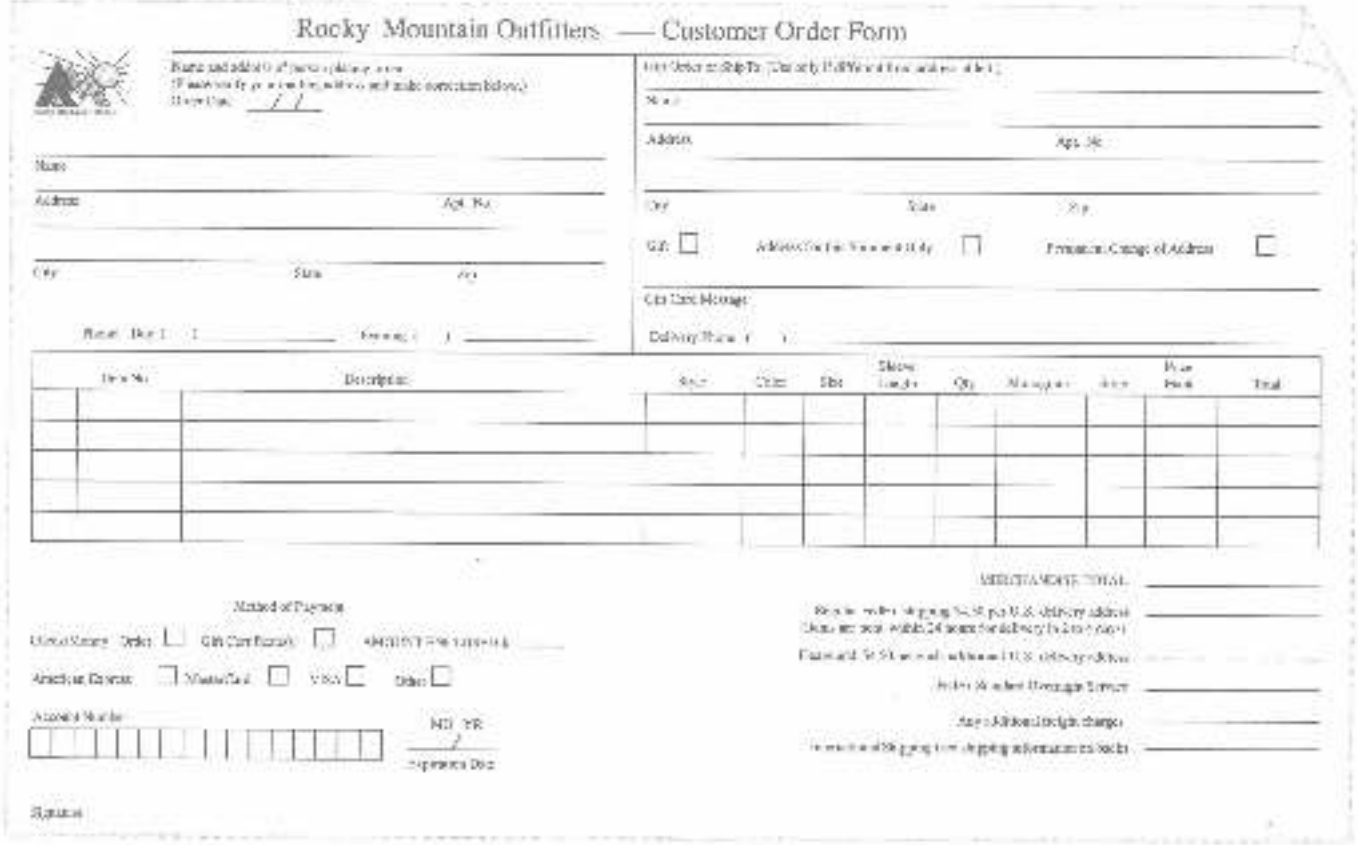
4.5.2 复查现有报表、表格和过程描述

这一步应该作为事实发现活动的第一步。对于现有的过程和形式而言，信息源有两个。一个信息源在公司外部，即业界的专业公司和其他一些公司。从其他公司获取信息可能不是很容易，但是它们往往是一些重要信息的潜在来源。有时候，业界的杂志会报道一些最新的发现或一些关于实践方面的研究信息。如果项目组的成员对这些信息不在意，项目组就可能会忽视这些信息的作用。同样随着系统跨越的公司越来越多，外部的信息也成为系统需求的重要来源。

另一个信息源来自公司内部，即现有的业务文档和过程描述。这种内部的复查有两个目的。首先，它是获得对过程最初理解的一个好方法。通常，新系统分析员对他们正在研究的工业或具体应用程序了解得不会太多，而对现有系统的初步复查将使他们很快跟上开发速度。

开始时，分析员可以请求用户提供他们正在使用的表格和报表的复印件。分析员也可以请求用户向他们提供过程手册和工作描述的复印件。对这些材料的复查有利于分析员对业务功能的理解，这些资料也构成了进行详细面谈的问题的基础。

使用文档和报表的第二种方法是亲自参加面谈，表格和报表既可以为面谈提供可视化的帮助，也可以为讨论提供工作支撑（如图4-8所示）。讨论可以集中于每一张表格的使用、目标、分发和信息内容。这种讨论还应该包括那些启用这些表单的业务事件。不同的事件可能使用相同的表单，然而那些关于业务事件和业务处理的信息才是至关重要的。使用真实信息填写的表格有助于分析员正确理解表中的字段和数据内容。



Rocky Mountain Outfitters — Customer Order Form

Name and address of person placing order
(Please verify your name, address and make corrections below.)
Order Date: / /

Name
Address Apt. No.
City State Zip
Phone: Day ( ) Evening ( )

Gift Order or Ship To: (Use only if different from address above.)
Name
Address Apt. No.
City State Zip
Gift ☐ Address Confirm Vacation Only ☐ Permanent Change of Address ☐
Gift Card Message
Delivery Phone ( )

Item No.	Description	Style	Color	Size	Sleeve Length	Qty	Alterations	Price	Price Each	Total

MERCHANDISE TOTAL
Shipping and Handling: Shipping $4.50 per U.S. delivery address
Shipping and Handling: 24 hours for delivery (in 2 to 5 days)
Shipping and Handling: 5 hours for delivery address
Order Number Overnight Service
Any Additional Freight charges
Information of Shipping from shipping information on back

Method of Payment
Unused Money Order ☐ Gift Card Remark ☐ AMOUNT FOR CARD ☐
American Express ☐ MasterCard ☐ VISA ☐ Other ☐
Account Number
MO YR
Expiration Date
Signature

图4.8 RMO的订单表格样例

复查现有过程的文档将帮助你识别出在面谈中也许不会提及的业务规则。书写过程也可以帮助你发现业务过程中存在的不一致和冗余。然而，需要注意的是，过程手册通常不是最新的，常会有错误。此外，分析员应该和用户一起复查从现有文档得到的设想和业务规则，以确保它们的正确性。

4.5.3 主持与用户的面谈和讨论

与系统相关者进行面谈显然是理解业务功能和业务规则的最有效方法，但它同时也是最耗费时间和资源的。在这种方法中，项目组的成员（系统分析员）与单个用户或用户组举行会议。分析员首先要准备一系列有关系统开发的详细问题，然后再与用户进行讨论。在项目组理解并记录下所有的过程需求，显然，这将花费一些时间，因此通常要求与每个用户或用户组举行多次会议。

举行面谈清单

面谈之前
- 确定面谈目的
- 确定要包含的相关用户
- 确定参加会议的项目小组成员
- 建立要讨论的问题和要点列表
- 复查有关文档和资料
- 确定会议的地点
- 通知所有参加者有关会议的日的、时间和地点

进行面谈
- 介绍自己
- 说明目的
- 记录发生的事情和情况
- 输入清单项目
- 计划后续
- 检查并记录下无回答或未解决的问题

面谈之后
- 复查笔记的准确性、完整性和可理解性
- 把访谈的信息转化为适当的模型和文档
- 确定需要进一步澄清的问题
- 适当的时候向参加会议的每一个人发一封感谢信

图4-9 准备与用户进行面谈的清单实例

为了进行有效的面谈，分析员需要对以下三个方面进行组织：① 为面谈做准备；② 主持面谈；③ 面谈的后续工作。图4-9是一个清单实例，在这个清单中总结了面谈的要点，这对于准备和主持与用户的面谈是非常有用的。

1. 准备面谈

每一次成功的面谈都需要精心的准备。在准备面谈的过程中，首先是最重要的步骤，即确定面谈的目标，换句话说，也就是在面谈中要完成哪些事情。写下面谈目标以便牢固地记在头脑中。第二步是确定面谈中应该包括哪些用户。前面的这两步结合得非常紧密，因此通常把这两步一起完成。这两步相当重要，因此即使不为面谈做任何其他准备，也必须至少完成这两步。面谈的目标和参加者决定了面谈中的其他任何事情。

参加面谈者包括用户和项目成员。通常，每次面谈至少应该包括两个项目成员。这两个项目成员不仅在面谈结束后可以互相比较笔记以确保准确性，而且在面谈进行当中可以互相帮助。用户数量大小的变化取决于面谈的目标。通常，当会议的主题不宽泛或者只是进行事实调查的时候，最好只有少量用户参加，一次超过三个用户的面谈有可能使得讨论时间变长。如果会议的主题是探讨性的，例如，在BPR项目中探讨新过程变化，那么最好要多用户参与。多用户参加会议往往给总结和评价新思路带来一定的好处。然而，管理这样的会议以确保所有与会人员得到高质量的收获是非常困难的。本章的后面将讨论专业的帮助人员和正式的发现技术（例如联合应用程序设计），如果会议的主题比较复杂或者比较紧急，那么就需要上述两种手段，以及很多用户的参与。

实践指导

确保在用户面谈时至少有两个项目成员参加。 ■

准备面谈的第二步是为面谈准备一些详细的问题。分析员可以根据先前获得的表格和报表写出一些具体的问题，并做好笔记。通常，建议准备一些与面谈目标相一致的问题。无限

制问题和有限制问题都是准备面谈的合适问题。无限制问题如“你如何完成这项功能”，鼓励分析员和用户对问题进行讨论和说明。有限制问题如“你每天处理多少张表格”，可以用来获得具体的事实。一般而言，无限制问题有助于开始对问题进行讨论，并且鼓励用户说明所有的业务过程和业务规则细节。

准备面谈的最后一步是做出最终的面谈安排并把这些安排通知所有参加者。具体的时间和地点要确定下来。如果可能的话，尽量选择一个安静的地点以避免外界的干扰。每一个参加者都应该知道会议的目标，而且在适当的时候，参加者也应该有机会预览一下将要面谈的问题或材料。面谈需要花费大量的时间，但如果每一个参加者都预先知道将要完成什么的话，那么面谈就可以非常高效的进行了。

2. 主持面谈

新系统分析员通常对主持面谈感到非常紧张。然而，请记住，在大多数情况下，用户对能得到一个可以帮助他们完成工作的更好的系统感到很兴奋。保持良好的礼貌通常能确保面谈顺利进行下去。以下是一些具体方法。

衣着得体。分析员的衣着打扮应该至少与客户最好的用户一样得体。在许多公司的工作环境下，例如在银行或保险公司的管理人员出席的情况下，那么西装就是合适的。而在工厂或生产环境下，工作服也许是合适的。衣着得体的目的是要使用户感到分析员不仅具有分析能力，而且具有专业精神，而不是使用户感到紧张。

准时到达。可能的话，尽量早到一点。如果是在会议室举行会议，那么分析员应该确保会议室适时开放。如果参加会议的人员很多或者会议时间很长，那么可以计划在中间休息时适当的准备一些点心。

限制面谈时间。准备面谈的过程和面谈本身影响面谈所需的时间。当你确定了面谈目标并准备好了问题之后，面谈的时间应该控制在一个半小时左右。如果面谈需要更多的时间来讨论一些其他的问题，那么中断本次讨论并安排另一次面谈会议通常是比较好的方法（我们将在后面部分讨论面谈的其他技术，这些技术包括举行个天会议）。若用户有其他的职责任务，系统分析员一次也只能集中了这么多的信息。通常，举行几次比较短的面谈效果要比举行一次马拉松式的面谈会议效果好得多。这一系列面谈提供了收集各种资料的机会，这些材料将在随后的过程中被不断细化。在几次比较短的面谈会议后，分析员和用户都将会对系统有一个比较好的理解。

寻找异常和错误情况。找机会问一些“如果……那会怎么样？”这样一些问题，例如，“如果没有到达该怎么办？”，“如果签名丢失了该怎么办？”，“如果结余发生错误该怎么办？”，“如果两张订单表格完全一样该怎么办？”。好的系统分析员的长处就在于能理解所有这些“如果……那会怎么样？”的问题。要有意识地去确定所有的特殊情况，并与用户深入探讨。仅仅依靠书本来掌握这项技能是不够的，还需要日后的经验来完善它。通过认真练习，你就会掌握这项技能。

深入调查细节。除了寻找意外情况外，分析员必须进行深入调查以确保获得对过程和规则的完全理解。作为一个系统分析员，最难掌握的一项技能就是获得足够的细节知识。通常，获得对过程运转方式的一般了解是很容易的，但不要害怕问一些细节性的问题，直到彻底理解了过程的工作方式和使用的信息。忽略了细节，是不会对系统做出有效的分析的。

认真做好记录。做手写记录是一个好主意。通常，录音机会使用户感到紧张。然而，做记录表明你认为你正在获得的信息是重要的，从而使用户感到他们受到了称赞。如果两个分析员分别主持面谈，那么以后就可以比较所做的记录，在记录中找出并记下任何未回答或仍未解决的重要问题。一组好的记录不仅为下一次面谈会议的成功打下了基础，而且也为建立分析模型提供了基础。

图4-10所示为面谈会议的一个议程安排例子。显然，不需要完全遵守一个特定的议程。然而，正如图4-9所示的面谈清单一样，这幅图将帮助你记住在面谈中需要讨论的问题和项目。复制下面的清单，然后使用它。当形成了你自己风格的议程后，就可以按照你喜欢的方式对它进行修改了。



图4-10 面谈会议日程样例

3. 面谈的后续工作

后续工作是每一次面谈的重要组成部分。后续工作的首要任务是吸收、理解和记录面谈所获得的信息。通常，分析员通过构造业务过程的模型记录面谈的细节，并且写出非功能需求的文本描述。一旦会谈结束这些任务就要完成，然后所有的讨论结果分发给参与者进行确认。如果构造模型的方法非常复杂或者这种方法对用户来说非常陌生，那么分析员在会议结束后解释并核实这个模型。这些内容将在本章最后一节进行讨论。

在面谈过程中，可能会问到一些用户回答不上来的"如果……那会怎么样？"的问题。通常新系统会提出一些政策性的问题，而这些问题管理人员在以前从来没有考虑过。不要丢失或遗忘这些问题，这是非常重要的。图4-11所示为RMO一些具有代表性问题类型的表格样式。如果有几个面谈小组在同时工作，那么他们可以汇总一张组合的问题列表。其他要加入到列表中的条目包括对解决问题的解释和日期。

最后，针对需要进一步详细说明的问题域或者面谈中错过的信息，分析员可以生成一张新的问题列表。这将使你为下一次面谈做好准备。

实践指导

维持一个面向未解决问题或疑问的开放条目列表。

重要问题控制表						
编号	问题	预定日期	终止日期	项目人员	用户联系人	建　议
1	部分发货	6-12-2007	7-15-2007	Jim Williams	Jason Nadold	部分发货或者等待全部发货?
2	邀请和折扣	7-01-2007	9-01-2007	Jim Williams	William McDougal	认信中是否包括折扣?
3	额外折扣	7-01-2007	8-01-2007	Mary Ellen Green	William McDougal	如何处理说明商品?

图4-11　未解决条目表样例

4.5.4　观察并记录业务过程

除了与用户进行面谈以外，另一种极为有效的收集信息的方法是直接在用户工作的地方观察他们的活动，并且记录下所观察到的业务过程。这第一手材料对于准确理解在整个业务过程中发生了哪些事情是无价之宝。

1. 观察

俗话说，一张图表胜过一千句话。系统开发也是如此。和其他活动相比，观察业务过程可以帮助你理解业务功能。然而，当你在观察现有的业务过程的时候，你必须能够根据相关的业务过程将新系统可视化，也就是说，观察现有的业务过程的目的是为了了解基本的业务需求，不要忘记这些已有的过程往往需要且应该进行改进，从而更有效。不要陷入认为实现此过程只有一种方法的死胡同。

分析员可以使用好几种方法来观察用户的工作，对办公室进行快速浏览或者自己亲身实践用户的工作。快速浏览可以获得对办公室布局、计算机设备的要求和使用，以及工作流总体情况的大致了解。安排几个小时观察用户是如何实际完成他们的工作的，这种方法提供了对实际使用计算机系统和处理业务事务的细节的理解。通过像用户一样接受训练和做实际工作，分析员可以发现学习新程序的困难之处、系统易于使用的重要性，以及现有过程和信息源的绊脚石和瓶颈。

以同样的仔细程度对所有过程进行观察是不必要的。快速浏览对于其个过程来说也许就已经足够了，但对另一个更加重要、更加难于理解的过程来说也许需要更长的观察时间。要记住，你的目标是获得对业务过程和规则的全面理解，那么你就应该计划好把时间花在什么地方以获得对系统的彻底了解。就像面谈一样，在观察过程中分析员如果齐心协力，那么效果通常是比较好的。

通常情况下，观察使用户感到紧张，因此要尽可能地小心谨慎。有好多方法，例如，和用户一起工作或同时观察几个用户，可以使用户避免紧张。理解并关心用户的需要和感情，通常可以带来积极的效果。

2. 使用活动图来进行记录

通过和用户进行面谈，并观察他们的工作过程，会收集到一些和业务过程相关的信息，因此需要把它们记录下来。使用工作流图来记录这些信息是一种很有效的方法。最终可能会使用工作流图来描述新系统，但目前还是把注意力集中在如何记录当前的业务工作流程。

工作流是处理业务事务或客户请求的一系列步骤。工作流可以很简单，也可以很复杂。一个复杂的工作流可能包含上百个处理步骤，并会涉及公司各个不同部门的员工。作为一个分析员，你也许会试着凭记忆去回忆并理解工作流（这可不是个好主意），或者用很长的一段

文字来记录，或者使用图表来进行记录。简单的工作流图的优点在于它特别直观，可以和用户一起进行复查以确保其正确性。使用工作流图的主要好处在于它们是项目组和用户之间强有力的连接途径。

工作流：处理业务事务的一系列步骤。

在工作流建模中很少采用单一的方法，如流程图、数据流图和活动图。数据流图可以很好地在工作流中抽取各种数据，但它们不能表示控制流。流程图和活动表是专门用来表示处理步骤中的控制流的，但它们不能表示数据流。所以，分析员使用一种叫做活动图的工作流图。活动图也是工作流图，它用来描述各种用户（系统）的活动，每项活动由谁来做，以及这些活动的顺序是什么。

活动图：一种工作流图，用来描述用户的活动以及这些活动的顺序。

图4-12列出了在活动图中使用的一些基本的符号。椭圆代表工作流中个体的活动，连接的箭头代表着各个活动之间的顺序。黑色的圆圈表示工作流的开始和结束。菱形表示决策点，处理流程会在那里产生分叉，沿着某条处理路径继续进行。粗的实线叫同步条，它可以把一条处理路径分成多条并行的处理路径，也可以将多条并行的处理路径进行合并。活动图矩形区表示完成某些活动的实体。由于在工作流中通常会有不同的实体（这里指不同的人）执行工作流中的不同的步骤，活动图矩形区符号对工作流进行了分组，表示每个实体应该完成的活动。

同步条：活动图中的一种符号，用来分解或合并顺序路径。

活动图矩形区：活动图中的矩形区域，它代表着单个实体所完成的活动。

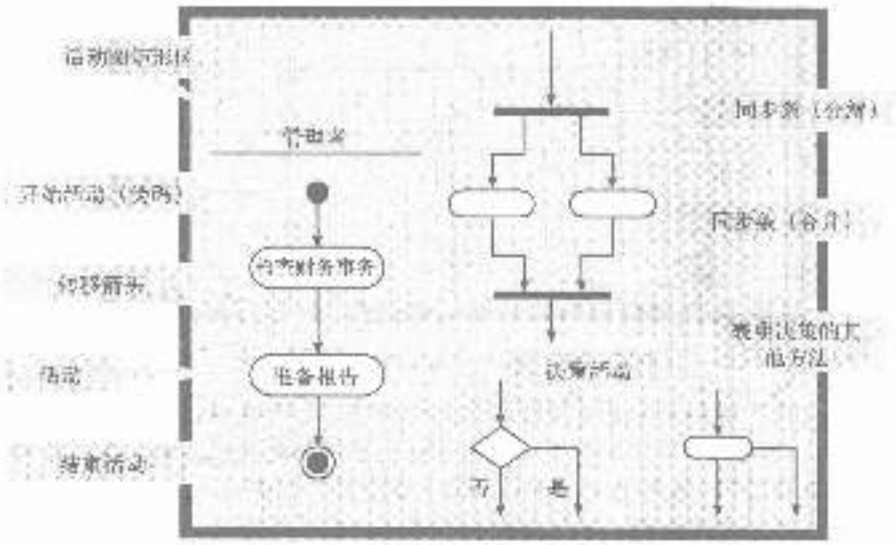


图4-12 活动图中的符号

图4-13所示为一个工作流的实际活动图。这个工作流代表客户从销售员处询问报价的过程。如果只是一个简单的查询请求，销售员可以输入数据并得到相关的报价。如果是一个复杂的查询请求，销售员要从技术专家那里获得一些帮助才能得到报价。在这两种情况下，计算机都将计算出报价的各种细节。

现在假设你正在与一个销售员面谈并观察报价的产生过程，如图4-13所示，你可以看到工作流是如何进行的。客户询问报价开始工作流，销售员完成工作流中的下一步，她记录下

报价查询的信息并决定她是自己完成还是需要帮助。如果她不需要帮助的话，就把信息输入计算机系统。如果销售员需要帮助，那么将由技术专家来完成下一步。专家们检查询价请求并保证被请求的组件可以集成到计算机系统功能组件中。检查请求的活动相当复杂，如果愿意的话，可以将它划分成更细的步骤来执行。这里，我们就不再进一步细化了。然后，专家将信息输入系统。这时计算机将产生详细的报价。注意，不论选择哪种方式，它们都会引起这一普通的活动。最后，客户检查报价并决定是否需要更改，或者这报价是否可以接受。在这个简单的例子中，顾客通常都会购买一些东西，所以这个工作流显然并不是完全准确的。

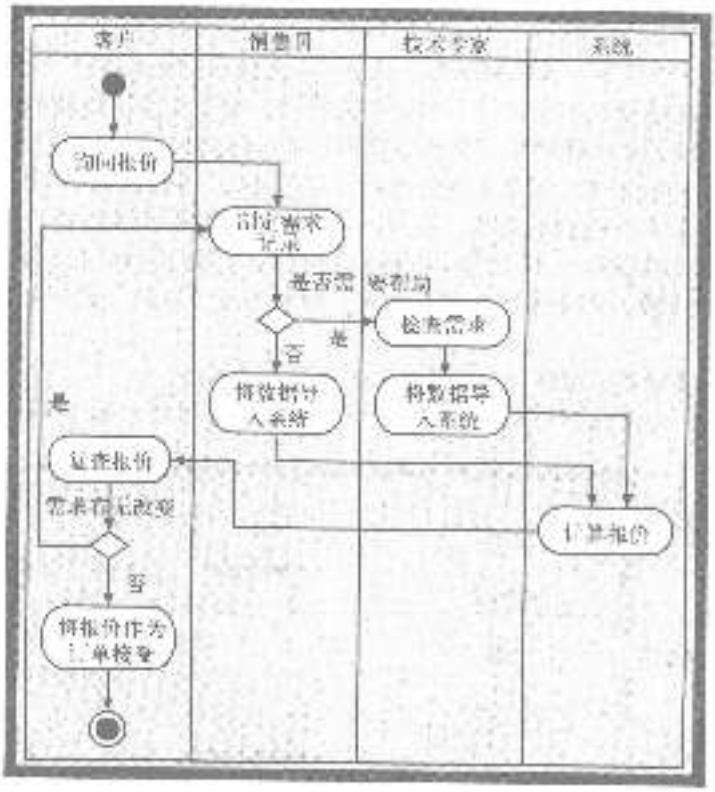


图4-13　用一个简单的活动图来说明工作流

请注意，活动图主要关注的是各个活动之间的顺序。它很直观，易于理解。事实上，使用活动图来记录工作流的一个优点在于用户会发现它们理解起来很容易，可以使用类似于活动图的图形表示来和用户一起检查对某些特定的工作流程的理解情况。

图4-14描述了为一个工作流。在这幅图中说明了一些新的概念。假设前一个例子中的客户想继续那个订单。下面的这幅图表明了获取订单生产进度的工作流。销售员将已经打印好的报价（现在已经成为一个订单了）送给工程部。这里要说明的是记录正在被传送这样一个事实。为了表明一个记录正在被传送，可以在连接箭头尾部做一个记录符号。这个箭头现在就成为了传送记录的通道，而不仅仅是一些活动。当工程部做出规范说明以后，就发生两个并行的活动：采购订单所需的原料并为自动铣床编写代码。这两个活动是完全独立的，可以同时发生。注意，一个同步条将原来的路径分成两条并行的路径，另一个同步条将它们重新合并在一起，最终按照计划完成订单。

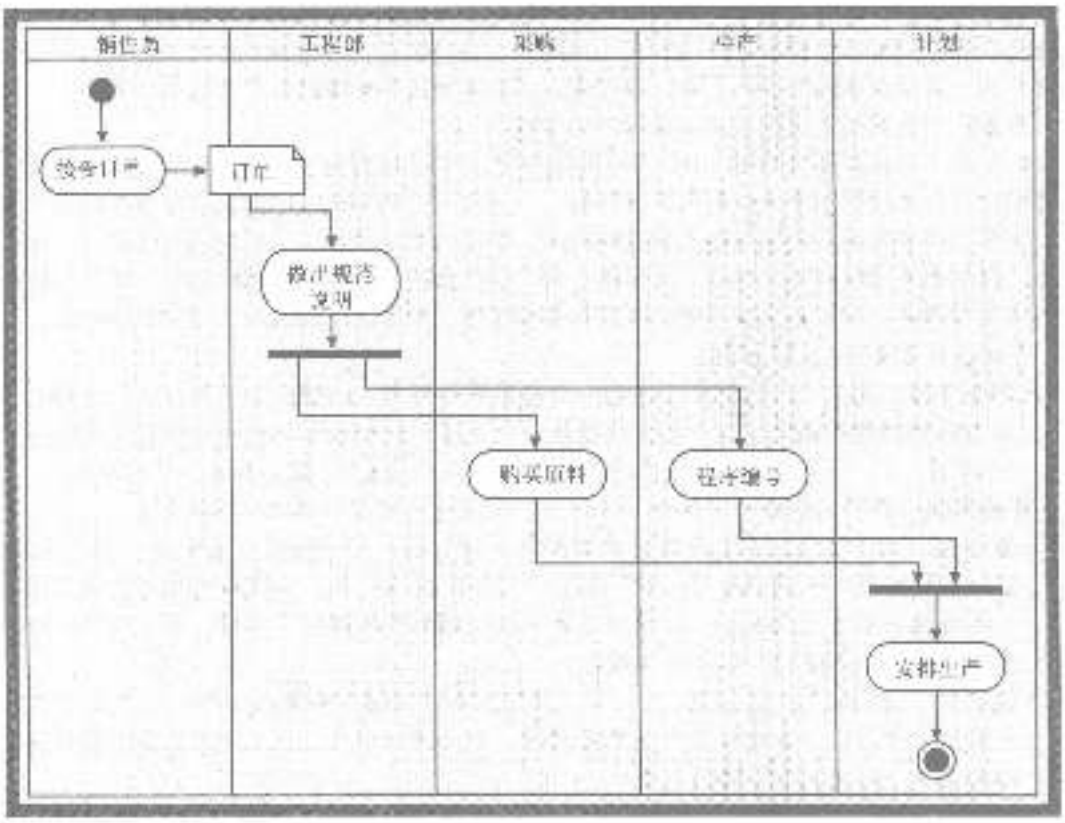


图4-14 有并行路径的活动图

通过创建活动图来记录工作流是很直观的。第一步是确定不同的实体从而创建合适的活动泳道区。接下来只要跟着工作流的各个步骤，用合适的椭圆来代表活动就可以了。用箭头来连接这些椭圆，这样就能够表示工作流的流向了。下面是一些简单的准则。

- 使用决策符来表示一个“或/或者”的情况——只能选择其中的一条路径，不能同时选择两条。作为一种速记符号，可以将一个活动（用椭圆表示）和一个决策（用菱形表示）合并在一个带有两个退出箭头的椭圆内，如图4-12右侧所示。这个符号代表了一个决策（或/或者）活动。无论何处，只要有需要“验证”或“检查”的活动，都会需要一个决策——一条用来处理“接受”的路径和一条用来处理“拒绝”的路径。可以把“或/或者”的路径合并成一个普通的活动（就像图4-13中计算报价那样）或者合并成其他的连接箭头。
- 对并行的路径使用同步条——在这种情况下，两条路径同时得以执行。既要有起始同步条也要有结束同步条。也可以使用同步条来代替循环，如“do while”这样的程序循环结构。把一个同步条放在循环开始之处，并写上“进行每一次处理”。把另一个同步条放在循环结束的地方，并写上“结束每一次处理”。

4.5.5 建立原型

在第2章中曾经提到，原型是指更大、更复杂实体的一个最初的、可以运转的模型。原型可以用于许多不同的目的，有许多不同的名称可以描述这些目的：废弃原型、发现原型、设计原型和进化原型。每一种原型都可以用于项目的不同阶段来测试和验证我们的思路。正如前面已

经解释过的那样，在分析阶段，原型用来测试系统的可行性并帮助其确定过程需求。这些原型也许是以简单的屏幕或报表程序的形式出现的。在设计阶段，可以建立原型来测试各种设计和界面方法。甚至在实施阶段，也可以通过建立原型来测试各种编程技术的效果和效率。

原型：一个规模更大的系统的最初可运转模型。

原型是一个非常强有力的工具，你可以发现它几乎应用在每一个项目的开发中。本章的前面曾经说过，发现原型用于单纯的发现目的，一旦测试完所要测试的概念后它就被丢弃了。例如，如果使用原型来确定屏幕格式和处理顺序，那么一旦定义结束，原型就被丢弃了。而另一方面，进化模型不断发展、完善，甚至最后可以成为最终的、实际使用的系统。正如后面在第16章将会看到的，原型方法要不断的修改和增加原型，直到使其最终成为可安装的系统。

下列是有效原型的一些特性：

- 可操作性。通常，一个原型应该是一个能运转的模型，而重点是可运行性。开始的简单原型被称为**实体模型**，这个实体模型是一个仅显示其外观而不提供执行能力的电子表格（如屏幕）。一个原型能够实际执行并提供"外观和感觉"，但也许缺少某些功能。

实体模型：最终产品的一个样例，这个样例只能进行观察而不能实际执行。

- 集中性。为了测试一个具体概念或者验证一种方法，一个原型应该集中于单一的目标。额外的执行能力不是具体目标的一部分，应该被排除在外。尽管有可能把几种简单的原型结合为一个更大的原型，但是目标集中性的原则仍然适用。以后，项目组可以把原型组合起来，以测试几个组件的集成性。
- 快速性。我们需要一些诸如CASE等工具以便快速地建立和更改原型，因为原型的目的是验证一种方法，因此如果方法是错误的，那么就必须有一些工具对原型进行快速的修改和测试，以便确定正确的方法。

在项目中集成原型活动是非常简单的。分析员需要牢记的是：为建立原型要有一种全面、透视的态度和决心，同时在建立所有原型的过程中重点是要始终保持一致。

4.5.6　分发和收集调查表

调查表在信息收集中的作用是有限的和具体的，调查表的好处是它使得项目开发小组可以从大量的系统相关者处收集信息，甚至当系统相关者在地理上分布很广时，他们仍然能通过调查表来帮助实现定义系统需求。

通常，项目组使用调查表可以获得各种系统相关者对信息需求的初步了解。这些最初的信息可以帮助分析员确定哪些领域需要通过文档概要、面谈和观察来进一步研究。调查表也可以帮助分析员回答大量的问题，这些问题诸如"你每天输入多少条订单？"以及"输入一条订单需要花费多少时间？"。最后，调查表可以用来确定用户对系统各个方面的意见，诸如"在1～7的等级范围中，可以访问顾客过去购买历史记录这一功能的重要级别是第几级"这样一些类型的问题。通常称为**有限制问题**，因为它们引导回答问题的人员提供对该问题的直接回答。有限制问题不能激发对问题的讨论和详细描述。然而，有限制问题的优点是问题的答案总是局限于一组选择。回答的表格可以用来确定平均数或趋势。

有限制问题：要求进行简单或明确回答的问题。

图4-15所示为一个显示三种类型问题的调查表示例。第一部分用有限制的问题来确定定量信息。第二部分由回答者是否同意陈述观点这样一些问题组成。所有这些类型的问题被用来制作表格和确定定量平均值。最后部分要求对过程或问题做出解释。这类问题适合于用于最初调查，帮助引导进一步的事实发现活动。

RMO调查表

本调查表将发给所有的电话订单销售人员。正如大家所知道的，RMO正在开发一个新的客户支持系统，这个系统可以为顾客提供订单处理和客户服务功能。

以此调查表的目的是获得一些最初信息来帮助分析员定义新系统的系统需求，此后还将举行进一步的会议，使每一个人都可以详细地阐述系统需求。

第一部分：根据一个典型的4小时轮班工作情况，回答下列问题。

1. 你接到了多少个电话？ _____

2. 订购一件商品需要多少个电话？ _____

3. 有多少个电话询问关于RMO产品的信息，也就是说，仅仅是询问？ _____

4. 估计一下在这段时间内顾客要求的产品发生缺货的情况有多少次？ _____

5. 在这些缺货的请求中顾客想要还期订货的情况占百分之几？ _____%

6. 有多少次顾客试图从过期的目录中订购商品？ _____

7. 有多少次顾客在交谈中取消了订单？ _____

8. 有多少次由于顾客的信用不好而拒绝其订货？ _____

第二部分：根据你同意或反对的强烈程度，在下列表格中1～7范围内的适当数字上画上圆圈。

问　题	非常同意						强烈反对
与顾客交谈时有可用的大量产品描述对做好工作是有帮助的	1	2	3	4	5	6	7
如果我有可用的顾客以往购买记录，那么对做好工作是有帮助的	1	2	3	4	5	6	7
如果我有所订商品的非应对件信息，那么我可以为顾客提供更好的服务	1	2	3	4	5	6	7
计算机响应时间缓慢，从而导致响应顾客需求发生困难	1	2	3	4	5	6	7

第三部分：请写下您的意见和建议。

请简要地指出现有系统的问题，这些问题您希望在新系统中得到解决。

_____

_____

_____

图4-15　调查表样例

实践指导

在调查表中，要限制无限制问题的数量。

调查表并不是一种很好的有助于掌握过程、工作流和技术的工具。先前识别的问题，例如“你如何处理这个过程？”，使用面谈和观察可以得到更好的回答。鼓励对一些问题进行讨论和详细说明，这些问题称为无限制问题。尽管一张调查表可以包含非常有限的无限制问题，但是那些包含许多无限制问题的调查表经常不能收回来。

无限制问题：要求对问题进行讨论而不是必须对问题做出简短的回答。

4.5.7　主持联合应用程序设计会议

联合应用程序设计（JAD）是用于加快系统需求调查的一种方法。先前所讨论的通常的面谈和讨论方法需要花费大量的时间。在这些方法中，分析员首先要和用户进行会谈，然后通过做笔记和建立原型记录下讨论结果，然后再对模型进行复查和修正。那些未解决的问题放置在一个未解决条目表中，这些问题也许需要举行另外几次会议和复查才能形成最后的解决方案。这个过程可能持续几周甚至几个月，这取决于系统规模的大小和用户及项目小组资源的可用性。

联合应用程序设计（JAD）：联合应用程序设计是一项定义需求或设计系统的方法，即让所有相关的人一起参加某个单一会议。

实践指导

JAD会议能够加速定义需求的进程。

JAD的目的是把所有这些活动压缩为用户和项目小组成员一起参加的更近的JAD会议。单独的JAD会议也许会持续一天或一周。在会议进行期间，要为系统的某一个具体方面完成所有的事实发现、建立模型、政策决定和校验等活动。如果系统规模较小，那么整个分析工作在JAD会议期间也许就可以完成。举行一个成功的JAD会议的关键因素是重要的系统相关者都要出席会议，从而促成并做出决定。实际与会者取决于具体的JAD会议的目的。需要包括在JAD会议中的人员如下：

- JAD会议的领导者。会议领导者是小组中最重要的成员之一，他们经验丰富，且受过专门训练。通常，一次JAD会议包括的人很多，每次会议都要有一个必须实现的具体目标的详细日程，而且讨论必须按照这些目标的方向进行。保持目标的集中要求具有熟练技巧和经验的人能够巧妙地使参加者专注于各自的工作。虽然我们习惯于指定一个系统分析员作为会议领导者，然而经验表明，成功的JAD会议往往是由接受过小组决策训练的人主持的。
- 用户。在本章的前面，我们定义了各种类型的用户，让所有适当的用户参与JAD会议是很重要的。通常，当发现了某种需求时，管理人员就需要做出相应的决策，如果管理人员没有出席会议，从而不能做出决定时，项目就停下来了。由于业务上的压力，很难使高级管理人员在整个会议期间一直出席。在这种情况下，应该安排管理人员每天参加一次或两次会议，从而使管理人员对政策讨论发挥作用。
- 技术人员。来自技术支持人员的代表也应该出席JAD会议。总是有一些技术方面的问题和决定需要做出回答。例如，参加者也许需要获得计算机和网络配置、操作环境和安全性等问题的细节知识。
- 项目组成员。JAD会议应该包括来自于项目组的系统分析员和用户专家，他们帮助进行讨论，阐明要点，控制所需的细节水平，建立模型，记录结果，以及大致查看系统需求的定义是否达到了必要的细节水平。会议领导者是一个推动者，但通常不是细节和需求方面的专家。项目组成员是确保完全实现系统目标的专家。

JAD会议通常在具有支持设备的专用房间进行。首先，由于整个过程非常紧张，所以重要的是要避免通常的日常干扰，有时也许需要一个相对静默的环境，或者张贴通知"请勿打扰"。另一方面，利用电话访问管理人员或技术人员通常是有帮助的，这些人虽然未曾与会会议，但是可以不时地邀请他们提出最后的政策或技术决定。

在JAD会议室中的资源应该包括高架投影仪、一块黑板或白板、活动挂图，以及为参加者提供足够的空间。JAD会议是工作会议，因此应该提供所有工作上必需的零碎用品。

近来，JAD会议开始充分利用电子设备来提高效率。如果参加者把个人或笔记本电脑连接到网络上，那么就可以增强分析能力和文件处理能力了。随着用记述性描述或模型记录的需求，或者随着一些简单的发现模型的建立，每个人都可以使用这些信息。通常在计算机上提供一个CASE工具来帮助实现屏幕、报表格式及文件设计的可视化。CASE工具也可以为在会议期间提出的所有需求提供一个中央资料档案库。

群体支持系统（GSS）运行在计算机网络上，允许所有的参加者在公共的工作簿大型提出建议（如果需要的话可以匿名）。这种方法可以帮助在群体讨论中腼腆的参加者变得更加活跃，从而有助于做出小组决定。当做出决定后，GSS也能提供存储最终需求的能力。正常情况下，JAD会议是和同一房间的所有人一起进行的，然而，在广域网上的GSS可以为分布在各

地的参加者提供虚拟会议环境。

群体支持系统：群体支持系统是一个计算机系统，该系统可以使得多个用户在同一时间参加讨论，每个用户使用自己的计算机。

图4-16所示为支持JAD会议的会议室的示例。经常进行项目开发的大公司通常都有这样的房间。图4-16所示的房间中有一些工作站，可以用来在JAD会议期间开发模型图和原型。通过使用计算机中的计算机支持协同工作（CSCW）软件来方便建议和讨论，这间房间甚至可能变得更高级。使用CSCW软件，在必要时某些管理人员甚至可以在外地参加会议。

如前所述，使用JAD的风险之一是加速做出决定。由于JAD的目标是很快对政策决定和需求细节做出结论，因此所做的决定有时也许不是最优的，有时，某些细节被不适当地定义或者完全丢失了。然而，在减少项目开发的努力和缩短时间方面，JAD会议取得了很大的成功。



图4-16 高科技的JAD设备

4.5.8 研究供应商的解决方案

公司期望新系统能够处理的许多问题或许已经被其他公司解决了。在大多数情况下，咨询公司可能会处理同样问题，有时候，一些软件公司已经为某些专门的业务需求开发出相应的系统软件包了。像Data Sources这样的黄页列出了上千个硬件公司、软件公司、咨询公司及解决方案开发商的联系方法。作为分析阶段的一部分，了解并利用这些现成的技术是很重要的。

研究现有的各种方案有二方面的积极作用，但同时也有一个危险的方面。首先，研究这些可选的方案有助于用户去思考如何更好的实现业务的功能，看别人是如何去做的，然后将他们的思想应用到自己公司的企业文化和结构上来，这样通常可以为业务需求提供一种可行的解决方案。

其次，事实上有些解决方案已经是一流的方案了，如果不研究这些方案的话，开发者可能在系统形成概念化之前就创建了一个毫无用处的、落后的系统。公司需要的解决方案不仅要能够解决一些基本的业务事务，还要能紧跟当前竞争的发展趋势。

再次，购买一个现成的解决方案通常要比新建一个系统更便宜，并且风险也更小。如果某个解决方案能够满足公司的需求并且可以直接买到，则这将是一个更安全、更快捷、更省钱的路径。在第8章中，将会讨论如何在购买系统和新建系统之间进行选择。在项目开发的早期，应该研究一下其他的可选方案，且不要急于做出最终的决定，除非你已经研究了所有的可选方案。

在这个过程中的危险（或者说是某种告诫）就是有时用户甚至系统分析员希望立刻购买某种方案，但是，如果在这个过程中过早的购买某个方案（如系统软件包），那么公司的需求就很有可能没有真正得到仔细的研究。这就好比买了一件衣服却不知道它的尺码一样。许多公司在购买了一些系统之后才发现它们只能满足半数的需求，不要落入那个危险的陷阱。

在寻找供应商解决方案时遇到的第一个困难是要找到谁有满足该业务需求的解决方案。许多大的软件、硬件公司像Oracle、IBM、Microsoft及Computer Associates都有自己专门的解决系统。有一些系统解决方案商名录——包括硬件商、软件商以及开发公司。其中，Data

Sources是比较好的一个，也可以从互联网上搜索更多的类似名录。有时，这些名录可以在图书馆中找到——公司的技术图书馆、城市图书馆或邻近学校的图书馆。

另外，可以参考行业杂志。例如，零售业有几种行业杂志。系统供应商经常会在行业杂志或行业展上做广告。另一种方法是从同行业的其他公司那里得到一些相关的信息。通常，用户都有共朋友在竞争的公司中工作，他们有时会知道某个特定的供应商能够为他们的需求提供解决方案。尽管在销售和市场上各个公司之间竞争激烈，但他们都同属于一个行业组织，他们之间共享一些行业信息是很正常的，这其中就包含与系统解决方案相关的信息。

一旦得到供应商的名单列表，接下来要做的就是对每种方案的细节进行研究了。得到销售和市场方面的资料很容易，但是得到系统的详细说明却要生难得多。这里可能会用到的方法有：① 技术说明书；② 一份样品或试验系统；③ 参观那些允许你观察他们系统的现有客户；④ 实地考察；⑤ 报告的打印稿和屏幕输出。

最后一步要做的是检查所获得的信息的细节。根据所获取的信息的不同，项目组既可以独立检查，也可以和用户一起进行检查。在多数情况下，应该和关键用户一起对信息进行核查，这样才能够更好地理解和识别解决业务需求的各种不同方法。无论如何，在理解商务、定义可能的活动时，对已有的解决方案进行研究是行之有效的一步。

4.6　验证系统需求

既然你已经学会了信息收集的技术，又知道如何从用户那里获得系统需求，现在要做的就是要确保这些信息的正确性。系统分析员常常以为他们已经理解了用户的需求，但是却没能抓住业务过程的最精妙之处。显然，当系统开发好以后再去纠正这样的错误，其代价是相当昂贵的。事实上，一些研究表明，在开发过程中修正一个需求错误的代价要比在需求分析时修正这个错误的代价高上百倍。

如果我们把开发一项新的信息系统和建造一间房屋做比较的话，那么在分析阶段确定的需求就好比是房子的设计蓝图。房屋的建筑完全取决于这些图纸。如果图纸有错（例如，如果承重墙不能够支撑结构上的支撑），那该怎么办呢？如果这个错误在到第二层建好以后才被发现的话，那么重建的代价将是很大的。所以，设计图纸必须确保准确。那么建筑师如何保证图纸的正确性呢？答案是在为了建造之前对设计图纸的正确性进行检验。

系统的需求存在着同样类似的问题。系统设计和结构依赖于需求的正确性。当编程工作开始的时候再对需求进行验证就太迟了，代价就太大了。对系统需求的检验和验证工作越早进行越好。

这时候你已经收集了一些用户需求的信息，或许也已经制订了一些工作流图。在下一章中，将介绍如何使用模型来描述系统的需求。在实际的设计和编程工作开始之前，所有的这些元素都必须经过仔细的测试。在编写计算机程序时，程序员必须通过各种方法来测试程序代码的准确性。在计算机上，通过执行程序（输入正确的数据和观察结果输出）来测试计算机程序的正确性。但是，不能以同样的方式来测试需求，因此必须使用一种不同的方法。

可以使用不同的技术来验证从用户那里获取的信息，以及根据那些信息得到的系统需求。为了验证内部的一致性，分析员可以通过构建模型来验证它们在数学上是否一致。在下一章中，将介绍有关这方面的更多的知识。有一种称为结构化遍历的强有力的技术，既可以用来检验用户的需求，也用来可以验证内部的一致性。

结构化遍历，有时简称为遍历，是指对调查结果和根据这些结果建立的原型进行复查。结构化遍历被认为是结构化的，因为分析员已经把复查过程形式化为一组过程。结构化遍历的目的是发现存在的错误和问题，其基本思想是在理解系统需求的过程中建立需求文档，然

后检查其中是否存在错误、遗漏、不一致之处及一些其他的问题。可以与项目小组中的其他同事对调查结果进行非正式的复查，但结构化遍历肯定要更正式一些。

结构化遍历：对调查结果和根据这些结果建立的原型进行复查。

要着重指明的关键一点是：结构化遍历并不是一种性能复查。只有当管理人员参与了最初的事实发现活动，而在检验和验证中也必不可少时，才必须把他们包括进来。复查是对分析员工作的复查，而不是对分析员本人的复查。为了理解更结构化的方法，本部分回顾了结构化遍历的四个要素：What, When, Who和How。

如附录A中所述，项目经理的一个主要责任就是要确保最终系统的质量。通常在项目进度的压力下，系统分析员会认为“我做得已经很好了，不需要复查了”。但是，对于项目经理来说，如果跳过复查这一步将是非常不明智的。由于可能会导致严重的后果，所以必须把能够确保需求完整性和准确性的特定任务和过程包含进项目规划中去，忽视这些，以后就会是在项目中出问题。结构化遍历可以用来验证收集到的信息，然而，必须确实地执行它们才能对规范模型（将在接下来的几章中讲述）进行验证。本节着重探讨结构化遍历的过程。

4.6.1 What和When

在结构化遍历期间，需要复查的第一项是作为分析阶段一部分而生成的文档。这些文档可以是对一个过程的叙述性描述、一张显示工作流的流程图或记录整个过程的模型图。通常，进行多次包括1—6页文档复查的较小遍历的效果要比进行包括30页细节的遍历的效果好得多。所有书写工作都是独立的，它们可以在一次遍历中进行复查。每隔一周或两周与项目小组成员举行小的遍历工作是很常见的。遍历的频率并不一定严格限制——文档建立后，就应该尽快地计划遍历。

4.6.2 Who

遍历中包含的双方是自己的工作需要被复查的人和复查工作的人。对于检验，即检验内部一致性和正确性，最好在遍历中包含一些其他有经验的分析员。他们善于发现系统中的不一致性和有关问题。对于验证，既确保系统满足各种系统相关者的需求，也应该适当地包括一些系统相关者。将要复查工作的特点决定了谁应是复查者。如果它是一张显示业务过程的图，那么提供最初定义的用户应该包括在内；如果它是设计细节的技术说明，那么技术人员应该包括在复查中。有时，复查者也许是项目小组成员。在其他情况下，他们是外部用户或技术人员。那些能够验证工作正确性的人也应该被包括在内。

4.6.3 How

就像面谈一样，在结构化遍历中，准备、执行和后续工作也是必不可少的。

1. 准备

分析员（其所做的工作将要被复查）为复查准备好材料。接着他指定相关的参加者并向他们提供材料的副本。最后，他应该为遍历安排好时间和地点，并通知所有参加者。

2. 执行

在遍历期间，分析员一点一点地提供材料。如果材料是一张报表或流程图，那么分析员应该根据图表向复查人员解释每一个组件。一种有效的技术是设计一个样本测试实例，并按照已定义的流程处理它。复查者首先寻找系统中的不一致性和问题，然后指出这些问题。资料管理员（主持人的助手）把复查者提出的建议记录下来。主持人自己不应该成为资料管理员，因为他们不应该从解释文档的过程中分心。其他人应该记录下错误、评价和建议，以确

保准确性。

实践指导

在结构化的遍历过程中，应该有一个资料管理员来记录所有的错误、评价和建议。 ■

在遍历期间，分析员不应该提出对问题的改正方法和解决方案。分析员最多可以提出一些建议解决方案，但在遍历期间不应该对文档进行修改。在遍历时，主持人有点紧张是难免的，因此让他们在会议之间做出明智的决定是不公平的。如果发现了对用户需求的误解，简要的复查也许是妥当的。然而，如果一个错误非常复杂，那么安排另外的面谈来澄清误解则是比较好的。遍历不应该成为一个事实发现的会议。复查者应该只提供反馈信息。主持人在收集了全部的建议之后，可以在不受外界干扰和批评的情况下，尽自己的最大努力去改正这些错误，然后再把这些反馈信息综合成更详细的材料。

3. 后续工作

做出所需的更正组成了后续工作。如果复查的材料有大的错误或问题，那么也许需要进行另外一次遍历。否则，修改错误，然后继续下一步活动。

图4-17是一张复查表格的样例。这张表格是RMO在复查会议中使用的表格之一，关于销售折扣率和规则。在图中没有显示的是几张附属图，其中包括过程的一组流程图。在这个例子中复查的材料仅仅是一系列有关折扣率的业务规则。而复查者是来自用户群体的高级管理人员。由于销售折扣率的业务规则是非常重要的，同时管理人员要根据折扣率来做出政策决定，所以他们对在讨论和面谈会议中发现的业务规则进行复查是当然之选。

遍历检查表

项目控制信息

项目：在线商品目录系统，客户支持子系统

复查的项目部分：复查销售折扣率的业务规则

小组领导人：Mary Ellen Green

记录员：Jim Williams

遍历细节

日期、时间和地点：

2007年4月10日，上午10:00点，MIS会议室

复查材料的描述：

这是对业务规则小组集成进行走和说之前进行的一次复查。这里有一页显示了折扣处理流程的附属的概念过程。这里也有另一幅显示设置折扣率过程的流程图。我们认为复查一些重要的问题，从而确保所有人都理解了应当做出的政策决定。

参加复查者：

William McDougal、Genny Monson和Robert Schneider

遍历结果

____××接受，签名：_____

_____小的改进。对改进的描述：

_____重复、计划新会议地点，描述需要再做的工作：

遍历结束时间：_____，签名：_____

图4-17　结构化遍历评价表

小结

分析阶段可以划分成下面6个主要的活动：

- 收集信息
- 定义系统需求
- 对系统需求进行优先级排序
- 构建系统原型，检验可行性并发现问题
- 产生、评估候选方案
- 和管理部门一起复查各种建议

业务流程重组已经成为一件提高业务过程而普遍使用的方法，所以它能够对分析阶段产生深远的影响。它完成了一套业务过程完整的重新设计方案。在BPR下，新系统的开发不仅仅依靠自动操作现有程序。相反，全部的过程需要进行完整重新思考。采用一种新方式来使用IT技术的目的是为了保证服务效率和服务水平有较大改进。由于系统分析员有了专用的问题解决方案及分析和建模技术，所以他们在BPR中起重要作用。

通常我们把系统需求分为两类：功能需求和非功能需求。功能需求用于说明新系统必须支持的基本业务功能，而非功能需求则包括系统性能目标、操作环境及其他非功能性问题。

为了找出系统需求，系统分析员必须和新系统的各种系统相关者一起工作。我们把系统相关者分为三种：① 用户，即每天实际使用系统的人；② 客户，即支付和拥有系统的人；③ 技术人员，即确保系统在公司的计算机环境下运行的人。在确定系统需求时最重要的步骤之一就是识别这些各种类型的系统相关者。

调查系统需求的基本问题是："我需要哪些信息？"本章提供了一些大致的指导。当学习了有关建模的更多知识以后，你将对你究竟需要哪些信息有更好的理解。一般而言，需要从下列二个方面寻找信息：

- 业务过程和操作是什么？
- 业务过程如何完成？
- 信息需求是什么？

我们主要使用7种技术来收集信息，而使用另外一种技术来确保所收集信息的正确性。7种事实发现的技术是：

- 复查现有的报告，表单和过程描述
- 主持与用户的面谈和讨论
- 观察并记录业务过程
- 建立原型
- 分发和收集调查表
- 主持JAD会议
- 研究供应商的解决方案

原型是指一个更大、更复杂的实体，它是最初的、可以运转的模型。原型的最初目的是开发一个可以运转的模型，这个模型可以用来测试一个概念或证实一种方法。建立发现原型来定义需求，然后通常就不再使用它了，或者至少不再用于最终的编程。

联合应用程序设计是这样一项技术，它通过和所有的关键参与者举行几次马拉松式的会议来加速系统需求的调查进程。讨论能立即得到需求定义和政策性决定，不会因与各个独立的小组举行面谈及试图消除分歧而延误。如果能正确地使用这项技术，它将是一种强大、有效的工具。

确保分析准确性和完整性的复查技术称为结构化追历，请记住，结构化追历的目的是复查和改进工作，而不是对性能进行复查。

关键术语

activity diagram	活动图
closed ended questions	有限制问题
functional requirement	功能需求
group support system (GSS)	群体支持系统
joint application design (JAD)	联合应用程序设计
logical model	逻辑模型
mock-up	实体模型
nonfunctional requirements	非功能需求
open-ended questions	无限制问题
performance requirement	性能需求
physical model	物理模型
prototype	原型
reliability requirement	可靠性需求
security requirement	安全需求
stakeholders	系统相关者
structured walkthrough	结构化追历
swimlane	活动图矩形区
synchronization bar	同步条
system requirements	系统需求
technical requirement	技术需求
transaction	事务
usability requirement	可用性需求
workflow	工作流
Zachman framework	Zachman框架

复习题

1. 功能需求和非功能需求之间有何差异？
2. 解释一下发现原型和进化原型的使用方法。
3. 列出三种事实发现的技术，并分别描述。
4. 结构化追历的目的是什么？
5. 介绍一下准备面谈会议的具体步骤。
6. 在信息收集活动期间，对供应商的调研有什么好处？
7. 在事实发现期间，你应该把哪些类型的系统相关者包括在内？
8. 当确定需要包括的用户时，水平和垂直方向表示什么？
9. JAD是什么？什么时候使用它？
10. BPR是什么？它和系统分析有何关系？
11. 用什么技术来对用户的需求进行验证？
12. 描述未解决条目表并解释为什么它们很重要。

13. 对系统分析来说，正确性、完整性和全面性代表什么意思？
14. 列出并说明7种信息收集技术。
15. 工作流图的目的是什么？
16. 画出在工作流图中使用的各种符号，并说出它们的功能。

思考题

1. 调查系统需求时最重要的问题之一是要确保需求是完整、全面的。为了确保在面谈会议期间得到所需信息，可以采取哪些措施？
2. 为了确保在面谈人员列表中已经包括了所有合适的系统相关者，可以采取哪些措施？如何复查这些列表？
3. 在调查期间你将遇到的问题之一是“需求扩充”。即用户要求增加另外的特色和功能。发生这种情况是因为有关用户有许多尚未解决的问题，并且系统调查也许是用户第一次向别人讲述他们的需求。如何控制系统的不断增长，以及包括不应该属于系统一部分的新功能呢？
4. 观察用户工作总是困难的。它通常使得你和用户都感到不太舒服。为了确保由于你的访问而不至于使用户的行为发生改变，应该怎么办？为了使得观察看起来更自然一些，应该怎么做？
5. 如果你从所会见的两组人中对同一过程得出相反的结论，你会怎么办？如果其中一个是职员而另一个是部门经理，你又会怎么办？
6. 假设你是由4个分析员组成的小组的领导人，并且其中一个分析员从来没有对其工作做过结构化走查。你如何帮助这个分析员开始遍历？你如何确保遍历是有效的？
7. 在未解决条目表中列出了一些需要解决的问题。你正在努力和用户接触以获得及时决定。你如何鼓励用户做出这些最后的政策决定？
8. 假设你将要开始第一次分配给你的系统分析任务。但你的顾客并不喜欢花费时间和精力来训练新的、没有经验的分析员。为了使得你看起来能够胜任工作和准备充分，你应该怎么办？你如何与顾客接触？
9. 在RMO的事例中，你已和Jason Nadold在发货部举行了会谈。你的目标是确定如何进行运输及新系统需要哪些信息需求。列出你将使用的问题清单，其中包括无限制问题和有限制问题。为了确保能发现一些异常情况，请同时列出你将使用的一些问题和技术。
10. 根据下面的描述开发一个活动图。在建模的时候，请留意任何不明确或有问题的地方。如果要做出假设，请标明。

    Open Access Insurance System项目的是为车主提供汽车保险。一开始，预期的客户填写一份关于客户和他的交通工具的信息的保险申请。这些信息将发送给一个代理，并由代理向各个保险公司转发这些信息以获得相应的保险报价。当代理收到回应以后，他为客户做出最好的保险类型和覆盖等级的选择，并将保险的建议方案和报价单的复印件发给客户。
11. 根据下面的描述开发一个活动图。在建模的时候，请留意任何不明确或有问题的地方。如果要做出假设，请标明。

    采购部门处理公司其他部门的采购请求。在公司中，提出采购请求的人是采购部门的客户。采购部门的工作人员接受采购请求并监督它直到被订购、完成。

    如果要采购的产品价格低于1500美元，工作人员可以直接与一个采购订单并送往许可的供应商。如果要采购的产品价格超过1500美元，采购请求必须先让生产该产品的供应商进行竞标。工作人员选择合适的标，并把采购订单送给该供应商。
12. 根据下面的描述开发一个活动图。在建模的时候，请留意任何不明确或有问题的地方。如果要做出假设，请标明。

运输部门接收所有尚未完成的采购订单上的货物。当工作人员接收到一份运货单时，他（她）在未完成采购订单上查找这些货物，然后发出多份运输包装收据，一份送往采购部门，他们更新记录表示采购订单已经完成，另一份送往财务部门，这样就可以付款了。第三份送给在家中等待的客户，这样他（她）就可以收到货物了。

一旦货款已经支付，财务部门便给采购部门发出通知。当客户收到货物之后，他（她）也会给采购部门发出通知。当采购部门收到这些验证信息之后就可以结束这次采购订单了。

实验练习

1. 与将要包含在过程中的某个人举行非实发现的面谈（这个过程可以用于企业或公司中）。这个人也许是在大学里，也许是在你隔壁的一家小公司，也许是在大学中的学生志愿办公室，也许是在医生或牙医的办公室，也许是在志愿公司，也许是在本地的教堂里。识别一个过程，例如，保存学生记录、顾客记录或成员记录。列出问题列表，然后举行面谈。记住，你的目标是彻底理解过程，也就是说，你要成为这个单一过程领域里的专家。
2. 除了把这个练习作为观察练习外，其余的要求与练习1相同，或者观察其他人做这项工作，或者你自己亲自完成这项工作。写下你所观察到的过程细节。
3. 把你的一组同学召集起来，然后对从练习1和2所得到的结果进行结构化遍历。记录下你的面谈或观察结果，然后用带有描述性说明的流程图记录下这个过程。之后和你的同事们共进行一次遍历。或者以另一个任务为例，例如思考题9，遍历你为该任务所做的准备工作。请按照本部分描述的步骤来实现。
4. 对下列的主题之一进行研究，然后使用至少二个独立的库资源写1-2页的研究报告。
   a. 联合应用程序设计
   b. 作为发现机制的原型
   c. 业务流程重组
   d. 计算机支持协同工作（CSCW）
   e. 工作流系统
   f. 结构化遍历
5. 以RMO客户支持子系统为例，做出一列需要进行研究的过程列表。可以结合以前研究如L.L.Bean公司、Land's End公司和Amazon.com公司做零售管理项目的经验，来考虑一下本例。看看商品目录，检查一下零售商在网上所完成的网上销售，然后考虑一下用来支持这些销售活动的内在业务过程。列出这些业务过程，并描述你对每一过程的理解。

实例研究

John和Jacob有限公司在线交易系统

John和Jacob有限公司，是一家过去几年来一直很成功的地区性经纪人公司。在这个行业里，对顾客的竞争是很激烈的。大的国有公司有许多服务提供给顾客。剧烈的竞争也来自商品折扣和网上交易公司。然而，John和Jacob有限公司已经在美国东北部中上层收入人群中培养了大量的用户群基地。为了保持对顾客的竞争性优势，John和Jacob有限公司正在开发一个新的在线交易系统。系统计划实现许多新的功能，这些功能将为其顾客提供新服务。

Edward Finnigan（项目经理）正在确定应该包含在系统需求开发中的所有用户组。对于应该包括哪些人，他并不十分有把握。以下是他正在考虑的问题。

- 用户：对公司的30个交易办公室的每一个来说，交易系统都将是在线的。显然，将要使用系

统的经纪人需要有输入功能，但如何实现这个功能呢？对于什么是确保需求完整的最好方法，他也并不一分清楚，同样，他也不知道使用哪种方法不需要花费大量时间。把所有这些办公室都包括在内，可以增加用户对系统的热情和支持，但这将花费大量的时间。而且，把更多的经纪人包括在内将导致观点出现分歧，他必须对这些分歧的观点进行调解。

- 客户：交易系统也将包括确认信息、交易报表和给客户的结算单。网上访问也在计划中，这将使得客户可以进行网上交易并检查他们的账目。因此，Edward需要考虑的另外一个问题是，是否要把John和Jacob有限公司的客户也包括在系统需求开发中。通常情况下并不需要邀请顾客参加系统开发。然而，知道如何为John和Jacob有限公司的客户提供最好的服务，这就更好了。Edward对这个问题很敏感，因为一些经纪人向他表示，许多客户不喜欢现有系统的结算单样式。他想把客户包括在系统需求开发中，但不知道应该如何去做。
- 其他系统相关者：Edward知道他应该把其他一些系统相关者包括在内，从而帮助定义系统需求。然而，对于应该和谁接触，他也并不十分清楚。他应该去找高级管理人员？应该和中层管理人员进行接触？还是应该包括像账目和投资等back-office功能？对于应该如何组织，如何确定应该包括哪些人员，他也不能十分肯定。

1. 在新的在线交易系统开发期间，对Edward来说，和经纪人（即用户）交流的最好方法是什么？他是否应该使用调查表？他是和公司30个办公室中的每一个经纪人都进行面谈好吧，还是和代表整组的一个或两个经纪人进行面谈更好些呢？Edward如何确保需求信息是完整的，而且不用花费太多的时间？

2. 对于新系统的顾客输入功能，Edward如何把顾客包括在过程中？他如何使顾客有兴趣参加？Edward可以使用哪些方法来确保他已包括的用户是John和Jacob有限公司整个顾客组的代表？

3. 当Edward考虑应该把哪些其他系统相关者包括在内时，他应该使用哪些标准？找出一些方法来帮助Edward建立一张需要包括的人员名单。

对落基山运动用品商店实例的再思考



Barbara Halifax，CCS项目的项目经理，已经制作完该项目中系统相关者的列表了。如本章前文所示，有一些资深的主管将参与进来。他们不用提供主要的录入工作。当然，为Bill McDougal部工作的人需要来做这些工作。不仅因为他是项目的主办者，而且他的助手们都为这个系统以及该系统可能带来的业务利润增加而感到激动。Barbara和这些主管保持了良好的工作关系。

Barbara同样也找出大量的部门经理和一些员工，他们将可以提供详细的需求，他将他的系统相关者分成两组。第一组主要负责定义用户的需求，第二组包括那些并不直接使用系统但是需要从系统中获取报告和信息的人。他希望这些人的要求都能够得到满足。

作为一名经验丰富的项目经理，Barbara有许多事情要处理。她使用项目经理核对表来保证自己不忘记一些重要的事情。项目经理十分重要，并且潜在的压力很大。

当她在检查自己的列表的时候，她发现有几项活动她在CCS项目中还没有考虑到。她考虑在让项目组开始接见用户之前，自己应该先和项目组的成员对这些问题进行讨论。下面就是引起她注意的一些问题：

- 制作一份用户联系计划
- 控制好用户对系统的期望
- 控制系统范围，防止需求扩充

根据你在本章中学到的（可参照附录A），如果你是Barbara你会怎么做？显然，你希望为公司提供最好的解决方案，但是你也需要控制好项目涉及的范围及用户，只有这样，系统才可能开发

成功并能及时地安装。

1. 说明在本项目中，你将在用户联系计划中包含哪些要点。

2. 你能给项目组哪些建议，使他们可以控制好用户对系统的期望？

3. 你现在可以做出什么样的计划来保证系统的范围是现实的——也就是说，系统可以在给定的时间和预算内解决需求问题？

关注Reliable Pharmaceutical Services

Reliable公司计划开发一种能够为客户卫生保健机构提供订购和供给药品服务的extranet。就像他们从内部药房买药一样，这种extranet使得Reliable公司的供应商就像该公司的内部公司一样运行。关于最终系统的这些观点对定义系统需求和获取这些需求的信息有着深远的影响。

1. 什么样的信息收集方式最适合于从Reliable公司自己的管理队伍和其他员工中获取需求？什么样的适合从客户卫生保健组织获取？从供应商获取呢？

2. 在客户卫生保健机构中的患者是否应该参与信息收集的过程？如果是，为什么？以何种方式参与进来？

3. 考虑从供应商和客户收集信息。在定义需求时，系统分析员对这些组织的观察需要多深入？对于供应商和客户不情愿提供内部操作详细信息的情况，Reliable公司应该如何处理？

4. 内部人员、供应商、客户中，哪些团体是最可能受益的原型？为什么？

5. Reliable公司是否应该采用业务流程重组（BPR）作为主要的定义系统需求的方法？建议系统和开发的哪些特征赞成BPR，哪些特征反对BPR？

参考资料

Vangalur S Alagar. *Specification of Software Systems*. Springer-Verlag. 1998.

Soren Lauesen, *Software Requirements: Styles and Techniques*. Addison-Wesley. 2002.

Stan Magee *Guide to Software Engineering Standards and Specifications*. Artech House,1997.

O'Rourke,Fishman,Selkow,*Enterprise Architecture Using the Zachman Framework*. Course Technology, 2003.

Suzanne Robertson and James Robertson, *Mastering the Requirements Process*. Addison-Wesley, 2000.

Karl Wiegers. *Software Requirements*, Microsoft Press, 1999.

Jane Wood. *Joint Application Development* John Wiley & Sons. 1995.

Ralph Young, *Effective Requirements Practices*. Addison-Wesley, 2001.

第5章 系统需求建模

学习目标

阅读本章后，你应具备如下能力：

- 解释建立信息系统模型的多种原因
- 描述三类模型并列出一些用于分析和设计的专用模型
- 阐述事件如何确定系统活动和用例
- 确定和分析系统对实数出有益的事件
- 解释如何用问题域中事物的概念来定义需求
- 比较数据实体和对象之间的异同
- 确定和分析系统中需要的数据实体和域类
- 阅读、解释并创建实体-联系图
- 阅读、解释并创建类图

本章要点

- 模型和建模
- 事件、活动和用例
- 问题域中的"事物"
- 实体-联系图
- 类图
- 目标

Waiters on Call餐饮送货系统

Waiters on Call是饭店的一项送餐服务，由Sue Brickford和Tom Brickford于2003年首次使用。Brickford兄弟在上大学期间都曾在餐馆打工，他们一直梦想开一家自己的餐馆。但遗憾的是，最初的投资均以失败告终。Brickford兄弟发现，许多餐馆都提供外卖，而且一些餐馆（主要是比萨饼店）还提供送货上门的服务。然而，他们认识的许多人好像希望有更全面的食品选择的送货上门服务。

Sue和Tom认为电话订餐是最佳的选择——一项不需要很大初始投资的餐饮服务。Brickford兄弟和全城各种知名的餐馆订约，得到客户订单并负责将全部饭菜送货上门。当饭店准备好预订的饭菜后，按批发价交给Waiters on Call，而饭菜送到后，客户按零售价支付，并付给他们服务费和小费。Waiters on Call刚起步时规模很小，仅包括两家餐馆和一个在就餐时间工作的配送司机。随着生意越做越大，Brickford兄弟意识到：必须有一套专门的计算机系统来支持业务运转。他们聘请了一个咨询员Sam Wells来帮助他们判断需要哪种系统。

Sam问道："在你们打理生意时，哪些事件促使你们决定采用计算机？告诉我通常这些业务是如何进行的。"

Sue回答道："噢，是这样，当客户打电话订餐时，我需要把它记下来，然后通知给相应的餐馆。我需要决定派哪一个司机去送货，因此要司机打电话告诉我他们什么时间有空。有

时，客户会又打电话更改订单内容，因此我必须找到原始订单，然后通知餐馆做更改。”

“好的，那你又怎么管理现金呢？”Sam问道。

Tom插话道：“司机从餐馆取饭菜时会从餐馆直接拿到账单的副本，账单和我们的计算应该是一致的，然后司机送货时收取相应的现金并加上服务费。在下班时，司机报账。我们把司机收到的钱汇总起来，与我们的记录进行比较。所有的司机都交完账后，我们需要开张银行存款单，存入当天的总收入。每周末，我们按提前约定的批发价来计算欠餐馆多少钱，把结算单和支票寄给他们。”

“那你们还想从这个系统中获取什么别的信息吗？”Sam接着问。

“如果每周末能统计出每个餐馆有多少订单，城里每个区有多少订单，以及诸如此类的信息就更好了。”Sue补充道。“这能帮助我们制订广告策略及与餐馆的合同。而且我们需要月财务状况统计结果。”

在Sue及Tom说话时，Sam记下了一些要点，画了几张草图。之后，他经过一段时间的仔细考虑，总结出了Waiters on Call的状况。他说：“在我看来，你们需要一个系统，在发生如下事件时能进行一些处理工作：

- 客户打电话下订单，需要记录订单
- 司机完成一次送货，需要记录送货完成
- 客户打电话修改订单，需要更新订单
- 司机上报工作情况，需要登录
- 司机上交一天的收入，需要协调收据

然后，你们需要系统在特定时间产生所需信息。例如，在以下情况时：

- 产生日结算存放单
- 产生周末餐馆支付账单
- 产生周销售报表
- 产生月财务报表

根据你们所描述的经营运转方式，我认为你们需要系统存储如下类型的信息。我们称为数据实体或域类：

- 餐馆
- 菜单
- 客户
- 订单
- 订单支付账单
- 司机

“而且我估计你们也需要维护关于餐馆和司机的信息。另外，新餐馆加入、餐馆更改菜单、雇佣新司机和司机离职等情况都要用到这个系统。我的设计正确吗？”

Sue和Tom一致认为Sam所说的系统恰好满足了他们的需要，他们确信已经为自己的工作找到了所需要的顾问。

概述

上一章描述了SDLC中的整个系统分析阶段，然后介绍了在完成第一个分析活动——收集系统、系统相关者及系统需求的信息时所涉及的许多任务和技术。需要大量的信息来正确地定义系统的功能和非功能需求。本章及第6章、第7章将介绍通过建立各种模型来将功能需求整理成文档的一些技术。这些模型是在分析阶段的活动中建立的，通常称为定义系统需求。

尽管分析阶段的所有活动事实上是并行进行的。

在讨论了模型的类型和所扮演的角色之后，我们把注意力集中在两个关键的概念：（触发活动或用例的）事件和（用户问题域的）事物。这两个概念在传统和面向对象的方法中都能帮助我们定义系统需求。本章涉及了传统方法和面向对象方法的多种模型，包括那些基于RMO的客户支持系统。记住，任何已给的系统开发项目，或者使用了传统方法，或者使用了面向对象的方法。然而，有个关键的概念——事件和事物是两种方法的共同特性。第6章将继续讨论关于传统方法模型的需求，第7章讨论面向对象方法的模型需求。

5.1 模型和建模

分析员可以使用一组模型来充分描述信息系统需求（如第2章所述）。回想一下，模型是当前系统某些方面的一个体现。由于系统很复杂，需要建立多个模型来涵盖细节信息。这些信息是在分析阶段由分析员收集和分析产生的（参见图5-1）。第4章介绍的活动图是一种模型类型的实例，它把注意力集中在用户和系统活动上。此外，还需要使用许多不同类型的模型在不同细节层次（或不同抽象层次）上表现系统，这既包括在高层次上对系统的概括，也包括针对系统某些特定方面的细节描述。一些模型分别显示了问题和解决方案的不同部分，例如，一个模型表示输入，另一个模型表示数据存储。一些模型是从不同角度反映同一个问题和解决方案，比如，一个模型可能从局外人的角度显示对象的相互影响，而另一个模型则可能按先后顺序表示对象如何相互影响。



图5-1 分析员需要一组模型来理解系统需求

5.1.1 模型的作用

一些开发者认为模型是在分析和设计工作完成后产生的文档。但事实上，建立模型的过程能帮助分析员澄清和改良设计。分析员在建模完成之后学习并研究模型的各个部分，分析员也可以在建立模型时提出问题，并且随着建模过程的进行回答这些问题。新模块增加进来后，可以评估改变后的结果并再次提出问题。从这一点来看，建模过程本身对分析员有直接的帮助。尽管分析员从不向其他任何人展示某一特殊模块，建模所使用的技术本身还是很有价值的，但通常情况下，模块在分析和设计过程中都是和他人共享的。

建模在系统开发中之所以重要的另一个关键原因是它描述信息系统的复杂性。信息系统十分复杂，而且系统的某些部分难以剥了，系统各个部分的模型有助于问题的简化，并能够使分析员的精力一次只集中在系统的几个方面上。分析员使用如此多的模型，就是因为每种模型都侧重于系统的不同方面。实际上，分析员建立的一些模型就是为了把这些不同方面集成起来，以显示其他模块是如何彼此结合的。

由于收集和处理的信息数量庞大，并且每个分析员都要耗费大量的时间，所以分析员需要经常回顾这些模块，以便于回想前面所完成的工作细节。人的大脑仅能记住有限的信息，因此我们需要一些能帮助记忆的工具。模型提供了以一种容易理解的形式为后期使用存储信息的方式。

使下交流也是为什么要建模的另一个常被提及的原因。分析员在建模过程中能掌握信息，而且使用模型也能降低信息系统的复杂性。此外，模型在支持项目小组成员之间和与系统用户的交流过程中也起着非常重要的作用。如果一个组员负责建立输入和输出模型，另一个组员负责建立将输入转化为输出的处理过程模型，那么要保证这两个模型相互匹配，两个组员间就必

须进行交流。第二个组员在开发处理模块前，必须要知道需要哪些输出结果。同时，这两个组员都要知道存储了哪些数据（数据模型），这样他们才知道需要什么样的输入，以及需要什么样的处理过程来访问所需要的数据。模型正好帮助项目小组成员之间进行交流和协作。

同时，模型也有助于与系统用户之间进行交流并促进理解。通常分析员和多个用户一起讨论模型，使分析员获得用户系统需求的理解反馈。用户需要看到清晰、完整的模型来理解分析员提出的系统框架。此外，分析员有时也和用户一起开发模型，因此建模过程可以使用户更好地理解新系统所能提供的各种可能的功能。用户在使用模型时也需要相互交流。而且，分析员和用户可以一起使用模型向负责系统审核的管理人员汇报系统性能。

最后，分析员建立的需求模型可以作为以后的开发小组生维护和升级系统时的文档。考虑到投资在新系统上的资源十分巨大，所以开发小组保留一个清晰的记录是十分重要的。在实施过程中一个关键的活动就是把文档正确、完整地以一种后来的开发者能够使用的方式集成起来。大多数文档都由在项目开发过程中建立的模型组成，图5-2概括了建模在系统开发过程中重要的几个原因。

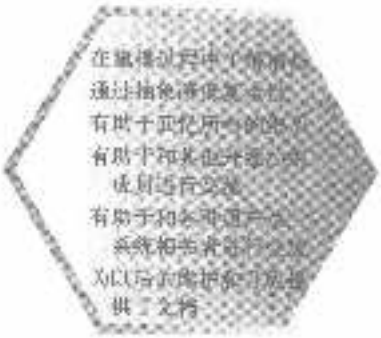


图5-2　建模的原因

尽管这本书的重点在于模型和建模的技术，但是仍然要记住系统项目要根据需求模型的数量和它们的形式而变化。简单的小型系统项目并不需要那些详细展示每一个系统细节的模型，特别是在这个项目组有这种类型系统的独立经验时。有时候关键模型能在很短的时间内被非正式的建立起来，虽然模型经常被第2章提到的CASE工具和可视化模型工具建立，但是有用的和重要的模型有时候会被迅速写到桌上的一张餐巾纸或者候机大厅里的一个信封的背面。随着SDLC活动，重复的方法被用来建立需求和设计模型。第一个模型的草稿列出了一些（并非全部）细节。接下来的重复性动作或许添加了更多细节或纠正了前一个的错误。

5.1.2　模型的类型

分析员在开发信息系统时要用到许多类型的模型。所使用的模型类型依赖于所要表达的信息的特性。模型可以分为三种常见的类型：数学模型，描述模型和图形模型。

1. 数学模型

数学模型是描述系统技术方面的一系列公式。数学模型用来表示系统精确的方面，这些最适合用公式或数学符号表示。例如，用等式来表示所需的网络吞吐量，用函数计算查询所需要的响应时间，这些模型就是技术需求的例子。此外，在科学和工程应用中，往往使用精确的数学算法计算结果，数学符号是表示这些功能需求的最好方式，并且它也是科技和工程方面的用户表达这些需求的最自然的方式。一个负责科技和工程应用的分析员最好能够精通数学。

数学模型：描述系统技术方面的一系列公式。

但对于较为简单的业务系统需求分析来说，有时候数学符号也是很有效的。例如，在工资管理系统中，使用一个总工资等于标准工资加加班费的模型就很合理。财产清单记录、产品价格折扣或提升加薪，都可以用简单的公式来建模。

2. 描述模型

并非所有的需求都可以用数学来精确定义。对于那些无法用数学精确定义的需求，分析

员使用描述模型。这些描述模型可以是描述性的备忘录、报表或列表。图5-3所示为针对RMO用户支持系统描述模型的示例。最初与用户的调查谈话也许需要分析员以描述的形式大概记录下来。比如从电话订单代表那里获得的电话订单处理。有时用户在给分析员的报表或备忘录中也描述了他们所做的工作。分析员在编辑所有信息时，可以把这些描述性的叙述转化成图形模型符号。

描述模型：描述系统某些方面的叙述性的备忘录、报表或列表。

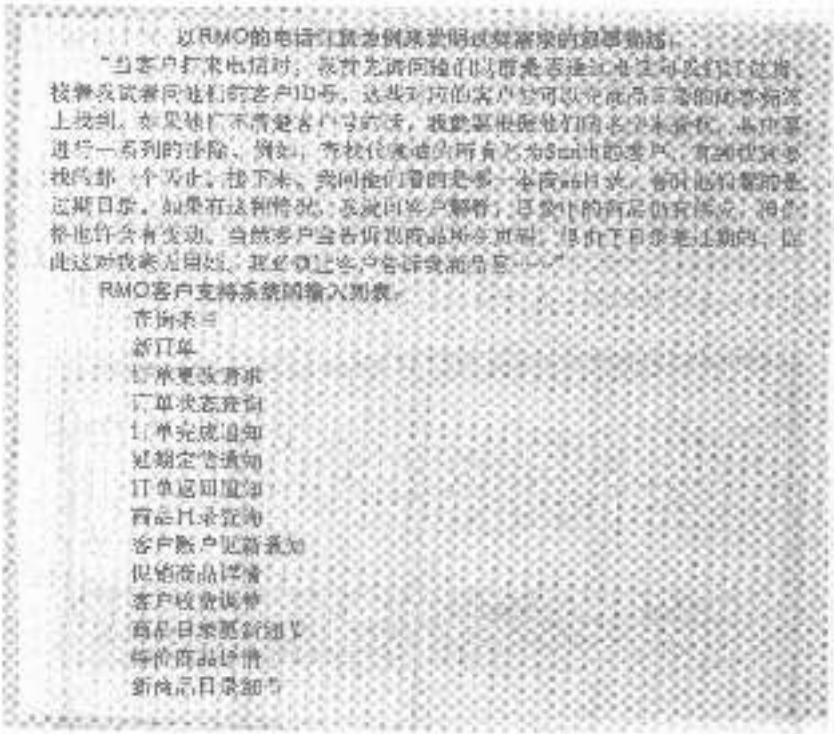


图5-3 描述模型

有时叙述性描述是记录信息的最好方式。用例描述经常被写成一篇或两篇短篇文本。主执行者和系统之间，需要更详细的用例描述的细节。许多有用的信息系统模型都包括简单的列表，例如，特征列表、输入列表、输出列表、事件列表和用户列表等。列表是一种准确、详尽、有效的描述或叙述性模型。图5-3包含了客户支持系统的简单输入列表。

描述模型的最后一个例子包括以一种精确的方式写下处理过程或程序步骤。这种方式通常被称为结构化英语或伪码。程序员熟悉建模算法所使用的结构化英语或伪代码，这两种方法可以得到相同的结果。因此，这种算法是非常精确的处理过程模型。

3. 图形模型

图形模型可能是分析员建立的最有用的模型了。图形模型包括图表和系统某些方面的示意性表示。图形模型有助于理解那些很难用语言来描述的复杂关系。一句古老的谚语曾这样说过：一张图胜过一千句话。在系统开发中，一张仔细建立的图形模型可能比100万句话都有用！

图形模型：图表和系统某些方面的示意性表示。

一些图形模型事实上看起来很像实际系统的一部分，例如，界面设计或报表布局设计等。但在大多数分析员的工作中图形模型多使用一些符号表示较抽象的东西，如外部实体、处理

过程、数据、对象、消息和连接等。分析阶段生往用一些关键的图形模型来表示系统中比较抽象的部分，因为分析阶段的重点集中在系统需求的高度抽象的问题上，而不去关心如何实施等细节。更具体的界面设计和报表格式模型是在系统设计阶段完成的。

在系统开发中要使用很多图形模型。每个图形模型突出（或抽象）了信息系统某些方面的重要细节。理想情况下，每一类图形模型应该使用特有、标准的符号来表示一些信息。这样，无论什么人都能看懂模型。然而，可供使用的符号很少——圆、正方形、矩形、线等，因此当你第一次学习模型符号时必须小心。在实践中，你会发现每种模型使用的符号略有差别。现在统一建模语言（UML）用在面向对象的方法中，提供了一种针对模型的图形标准。然而用在传统方法中的图形还缺乏标准化。

5.1.3　用于分析和设计的模型概述

分析阶段的活动被称为定义系统需求，其中包括建立多个模型。由于这些模型详细定义了系统需求而没有局限于某一具体技术，因此这些模型通常被称为逻辑模型（参见第4章）。分析员创建了很多种类的逻辑模型来定义系统需求。图5-4列出了其中经常使用的一些模型。当前Barbara Halifax一直让她的项目组为客户支持系统创建需求模型连续工作。

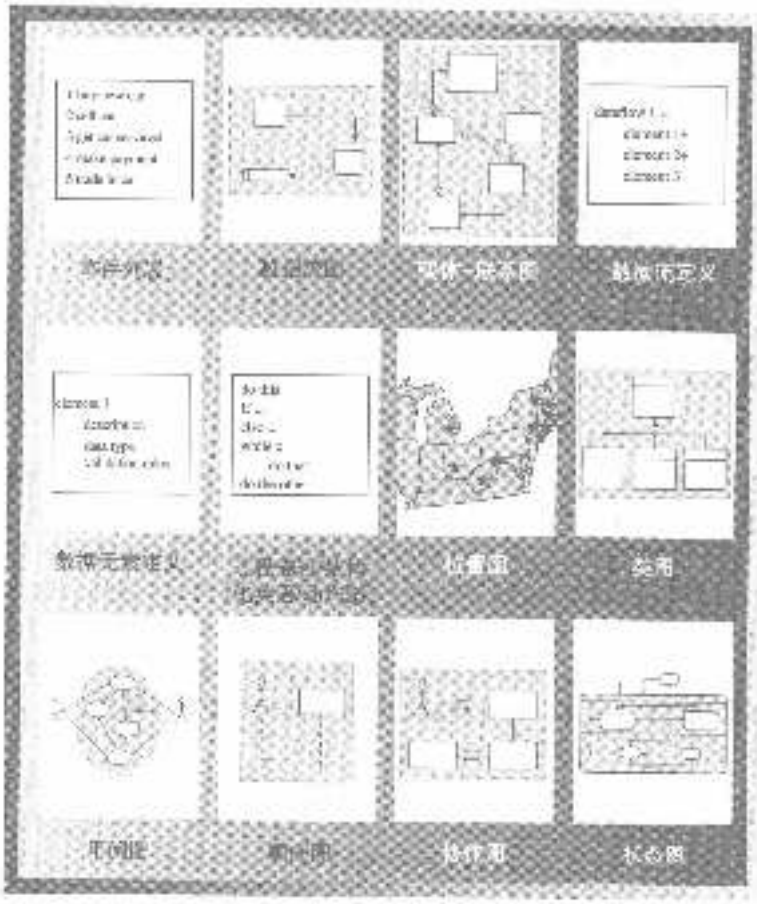


图5-4　分析阶段创建的模型

在设计阶段也会建立许多模型（如图5-5所示）。这些都是物理模型，因为它们显示了如

何使用具体技术来实现系统的某些方面。这些模型中，有一些要么是分析阶段所建立的需求模型的扩展，要么直接来自于需求模型。有些模型（如类图）既在分析阶段使用，也在设计阶段使用。

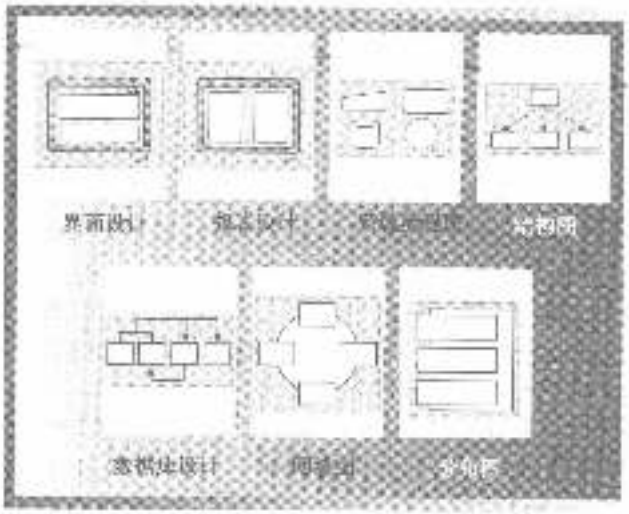


图5-5 设计阶段创建的模型

本章和第6章、第7章详细介绍了一些需求模型。这里先介绍事件和事物——所有开发方法中针对需求模型的常见的两个关键概念。本章所讨论的模型包括事件列表、用例、事件表、实体-联系图及类图。当Barbara Halifax和她的项目组开始为RMO新的客户支持系统建模时，他们要确定新系统需要提供的事件和事物（参见Barbara的备忘录）。

2007年3月20日

To：John MacMurty

From：Barbara Halifax

RE：客户支持系统更新

John，我只是想简单正报一下，我们正处于分析阶段，当前我们正在整理收集的信息和完成系统的需求建模。

我们已经确定了影响系统的大部分事件，比如客户请求一个目录、客户下订单、运输部门完成订单、客户返回表款、管理人员调整客户的赊购等，根据这些事件的系统活动/用例已经开始成形。我们已经让负责电话订单的推销员、负责邮件订单的人员及仓库人员检查了我们的设想，看上去已经接近目标了。

我们还定义了需要存储的大部分关键数据，诸如客户、目录、产品项、订单、订单项、装运管及发货人信息。在数据存储方面，我们与供应链项目的人员协同工作（我还是总建议见Jack Garcia）。

我们将继续进行建立需求模型的工作，下次热门会上应该就会有些图表展示给你。

谢谢！

BH

cc: Steven Deerfield, Ming Lee, Jack Garcia

5.2 事件、活动和用例

现在你已经了解了模型的基本类型，下面将学习如何为一个信息系统建立功能需求模型的细节。实际上，所有的系统开发方法都是以活动或用例概念开始建模过程的。用例是系统执行的活动，通常响应用户的要求。用例这个术语源于面向对象方法，但是现在也用于传统方法中的功能需求建模。如果在学习过程中把重点放在传统方法上，那么用例基本上等同于活动。

用例：系统执行的活动。

有几种方法用来确定活动或用例。一种方法是列出所有用户并考虑他们需要系统为他们的工作做些什么。一次只重点考虑一种类型的用户。那么分析员可以系统地指出确定用例时的问题。另一种方法是，开始于现有系统并列出当前系统具有的所有功能，然后让用户加入新的功能需求。第三种方法是让所有用户描述他们使用系统的目的。

图5-6列出了落基山运动用品商店的几个用户（又称为参与者）和他们为用户支持系统的工作目标。通过询问用户他们的目标，分析员能够把重点放在新的过程处理，而不是仅仅使现有的程序自动化。许多分析员发现这种方法对于得到一个用例的初始列表非常有效。然而，确定用例的最广泛的方法称为事件分解技术，在下面的章节中将会介绍。

用户/参与者	用户目标
订单职员	存找可用的条目 创建新订单 更新订单
运送职员	记录订单实行 记录结局交货
销售经理	产生特价商品 制作日常活动报表

图5-6　通过用户及其目标确定用例

无论采取何种方式确定用例，分析员必须在合适的粒度级别确定用例。例如，一个分析员可能将客户名字显示于表格中确定为一个用例，第二个分析员可能将增加一个新客户的整个过程确定为一个用例，第三个分析员可能将针对客户一整天的工作定义为一个用例，这其中包括增加新客户、更新客户记录、删除客户、追踪赊账客户或联系以前的用户。第一个实例粒度太小而不是非常有用。第二个实例定义了一个完整的用户目标，它是合适的用例分析级别，第三个实例（针对客户一整天的工作）太过宽泛也不是十分有效。

确定用例时一种合适的粒度级别是基本业务流程（EBP）。EBP是由一个人在特定地点为响应交易事件所执行的一项任务，它增加可度量的业务价值，使得系统和数据保持在一致状态。（参见本章最后的"参考资料"部分Larman在2005年发表的一文中有关于EBP的更多讨论。）图5-6中，可以成为用例的RMO客户支持系统目标是创建新订单、记录订单实行、产生特价商品等。这些用例是基本业务过程的很好的例子，每一个都由一个用户在特定的时间和地点执行，而且当执行完成之后系统和数据保持在一致状态。

EBP：由一个人在特定地点为响应交易事件所执行的一项任务，它增加可度量的业务价值，使得系统和数据保持在一致状态。

实践指导

设置用几种方法确定用例，然后交换检查以确保没有用例被忽略。 ■

我们注意到每一个EBP（每一个用例）在响应业务事件时发生。事件发生在某一特定的时间和地点，可描述，并且系统应该记录下来。系统的所有处理过程都是由事件驱动或触发的，因此当通过确定用例定义系统需求时把所有事件罗列出来并加以分析是很有意义的。

事件：可以描述、值得记录的在某一特定时间和地点发生的事情。

5.2.1 事件分解

在决定系统用例时，分析员将重点放在事件分解。事件分解是这样一种技术，它首先确定哪些是系统必须做出响应的事件，然后决定系统如何响应——系统用例。当为一个系统定义需求时，先调查清楚能对该系统产生影响的事件是十分有用的。更准确地说，什么事件发生时需要系统做出响应？通过询问对系统产生影响的事件，可以把注意力集中在外部环境上，而把整个系统看成一个黑盒子。这最初的调查帮助你主要从高层次上全面考察系统，而不是集中在系统内部工作上。这也使你把注意力集中在系统和外界用户和其他系统的接口上。最终用户，即真正使用系统的人，也习惯于按照那些影响他们工作的事件来描述系统需求。因此，当用户使用系统时，把重点集中在事件上也是非常恰当的。最后，把重点集中在事件上也提供了一种划分（或分解）系统需求为用例的方法，这样你就可以分别研究各个部分了。

事件分解：集中于确定系统必须做出响应的事件然后决定系统如何响应的一种分析技术。

图5-7所示为一些对商店的赊购账处理系统很重要的事件。系统需求基于6个事件分解为用例。客户触发3个事件：花费、付账或变更住址。系统用3个用例响应：记录付款、处理花费和维护客户数据。3个事件在系统内部根据如下时刻触发：发出月报清单的时刻、发出过期通知的时刻及完成周末汇总报表的时刻。系统用如下用例响应：生成月报清单、发出过期通知和生成汇总报表。根据事件描述系统可以使得赊购账系统的重点集中在业务需求和基本业务过程上。下一步就是为开发人员分配工作，一个分析员可能集中处理由人触发的事件，而另一个分析员集中处理内部触发的事件，这样的划分有助于对系统进一步详尽的理解。结果是一个由处于合适的分析级别的业务事件所触发的用例列表。

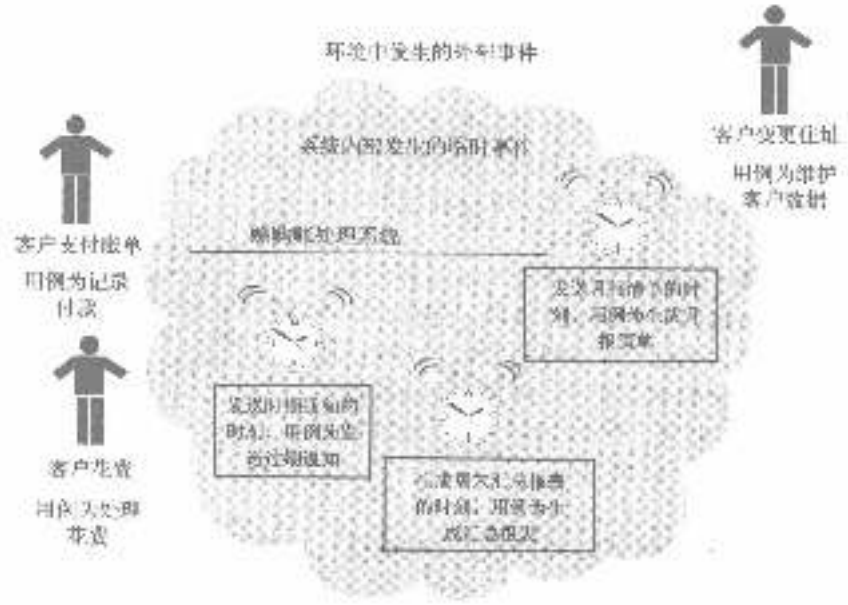


图5-7 生成用例的影响赊购账处理系统的事件

早在20世纪80年代，实时系统中采用的现代结构分析就强调了事件这个概念对于定义系统需求的重要性。实时系统要求系统能立刻响应环境中发生的事件。最早的实时系统包括诸如生产过程控制和电子导航等的控制系统。例如，在过程控制中，如果一桶化学材料满了，那么系统就需要关闭注入阀门。相关事件是“桶满了”，系统需要立刻响应这个事件。在飞机

导航系统中，如果一架飞机的高度低于5 000英尺，系统就需要开启高度过低的警报。

现在大多数信息系统的交互性都非常强。因此这些信息系统可以被认为是实时系统，事实上是用户希望系统能实时响应每一件事情。当此，现在采用事件分解方法来确定业务系统中的用例。

5.2.2　事件的类型

在利用事件分解技术确定用例时要考虑三种类型的事件：外部事件、临时事件和状态事件。分析员开始工作时要尽可能多地识别和列出这些事件，在与系统用户的交谈中不断细化这些事件列表。

1. 外部事件

外部事件是系统之外发生的事件，通常都是由外部实体或动作参与者触发的。外部实体（或动作参与者）是一个人或组织单位，它为系统提供数据或从系统获取数据。为了识别关键的外部事件，分析员首先要确定所有可能需要从系统获取信息的外部实体。客户就是一个典型的外部实体的例子。客户想订购一种或多种商品。这对于像落基山运动用品商店（RMO）需要的订单处理系统来说，是一个非常重要的事件。但是与客户相关的还有其他事件，有时一个客户想退掉已订购的一件商品或者一个客户需要接发票支付订货的费用。诸如此类的外部事件正是分析员所要寻找的那一类，因为分析员要定义系统需要完成哪些功能。这些外部事件将会导致一些系统必须处理的重要事务。

外部事件：系统之外发生的事件，通常都是由外部实体或动作参与者触发的。

当描述外部事件时，必须给事件命名。这样能更清楚地定义外部实体。同时，在描述中还应该包括外部实体需要进行的处理工作。因此，事件客户下订单描述了一个外部实体（客户）及这个客户想做的事情（订购商品）。这一事件直接影响着系统。此外，如果系统是一个订单处理系统的话，系统还需要处理该客户的订单。

重要的外部事件还可能来自于公司内部的人或组织单位的需求，例如，管理部门需求得到一些信息。在订单处理系统中一个典型的事件可能是管理部门检查订单状态，也许管理者想跟踪一个关键客户的订货情况，那么系统就需要定期提供该信息。



图5-8　外部事件检查列表

当外部实体提供了系统只需要存储下来以备将来使用的新信息时，就会触发另一类外部事件。例如，一个老客户通知他的地址、电话号码或雇主发生了变动。通常每一种有关外部实体的事件都可以描述为处理数据更新，比如客户更新账户信息。图5-8所示为一个检查列表来帮助识别外部事件。

2. 临时事件

第二类事件是临时事件，临时事件是由于达到某一时刻所发生的事件。许多信息系统在事先定义的时间间隔产生一些输出结果，例如工资系统每两周（或每月）生成工资支票。有时输出结果是管理部门需要定期获取的报表，例如业绩报表或异常报表等。这些事件与外部事件不同，因为系统自动产生所需要的输出结果而不需要用户进行操作。换句话说，没有外部实体或动作参与者下达命令，而系统自己在需要的时候产生所需的信息或其他输出。

临时事件：由于到达某一时刻所发生的事件。

分析员通过询问系统必须完成任务的具体时限来确定临时事件。在最后期限之前应该生成哪些输出结果？在这期限到达时需要系统完成哪些处理任务？分析员常常通过定义当时系

统需要产生的结果来确定这些事件。在前面提到的工资系统中，临时事件就应该被命名为每周生成工资单。需要月末汇总报表的事件可以命名为该生成月末销售汇总报表了。图5-9所示为一个检查列表用来识别临时事件。

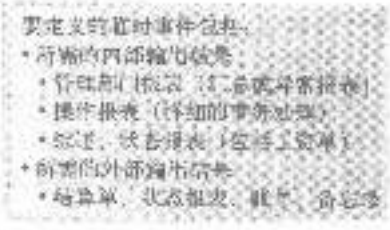


图5-9 临时事件检查列表

临时事件不一定非要在确定的日期发生，它们可以在预先定义的一段时间过后发生。比如，在商品售出之后给客户发账单，如果15天之后客户还没有支付账单，那么就发出一个过期通知。临时事件可能在账单发出的15天之后被定义为该发过期通知了。

3. 状态事件

第三类事件是状态事件。它是当系统内部发生了需要处理的情况时所引发的事件，例如，产品的销售导致了库存记录的变化。当库存降到了需要重新订货点之下时，就有必要重新订货。该状态事件可以被命名为到达订货点。通常状态事件作为外部事件的结果而发生，有时，状态事件和临时事件相似，唯一不同的地方在于，状态事件无法定义事件发生的时刻。重新订货事件也可以被命名为库存该重新订货了，这样听起来就像临时事件。

状态事件：当系统内部发生了需要处理的情况时所引发的事件。

5.2.3 定义事件

通常要定义影响系统的事件并不容易，但是一些方法可以用来帮助分析定义事件。

1. 事件/条件和响应

有时候很难区分事件和一系列导致该事件发生的条件。以一个客户从一家零售商店买衬衫为例（见图5-10），从客户的角度来看，这个购买过程包括了一系列事件：第一个事件可能是客户要穿一件衣服，接着客户可能要穿一件条纹衬衫，然后他发现条纹衬衫已经穿旧了，所以他决定开车去购物中心。之后他决定进入Sears商店，在那里他试穿了一件条纹衬衫，接着他决定离开Sears，到沃尔玛去试穿衬衫，最后在那里他决定买那件衬衫。分析员必须考虑诸如此类的一连串事件，然后确定出能影响系统的事件。在本例中，客户在商店里手拿着衬衫决定购买，并且说“我要买这件衬衫”时系统才开始接受影响。



图5-10 导致影响系统的一个事件的一系列行为

在其他情况下，很难区分外部事件和系统响应。例如，当客户购买衬衫时，系统需要信用卡号码，客户就提供信用卡。那么，提供信用卡的行为是一个事件吗？在本例中，这不算是一个事件。它只是在处理原始交易时发生的一部分交互行为。

要确定一个事情的出现是事件还是随事件而发生的一部分交互行为，采用的方法就是看二者之间是否有较长的停顿或间隔——也就是说，系统能否无中断地完成事务处理吗？或者系统暂停下来等待下一次的事务处理？一旦客户想购买衬衫，处理过程会持续下去直到交易完成为止。交易一开始中间就没有明显的停顿。一旦交易结束，系统就暂时终止，重新等待下一次交易的开始。

另一方面，当客户用商店信用卡账户购买衬衫时，独立的事件就发生了。当客户在月末支付账单时，其处理过程是购买事件的一个交互行为吗？在本例中不是。系统记录下这次交易，然后开始处理其他事情。系统并没有终止下来等待支付行为结束，而是在此后发生一个独立的事件，该事件导致系统给客户发一个账单（这个事件是一个临时事件：就发送月底账单了）。最后，又发生了另一个外部事件（客户支付账单）。

2. 事件序列：跟踪事务处理的生命周期

在定义事件时，跟踪针对某一外部实体或参与者而发生的一系列事件流常很有用。在落基山运动用品商店（RMO）新的客户支持系统的例子中，分析员要考虑由于增加一个新客户所引发的所有可能的事务（见图5-11）。首先，客户想要一本商品的目录或者询问其中一些商品的情况是否有效。这一事件导致数据库中增加了客户姓名、地址记录。接着，客户也许想发送订单。也许她/他将来想要修改订单，比如改变衬衫的尺寸或者买另外一件衬衫。接下来，客户也许想查询订单的状态，以获得发货时间。也许客户搬家了，想修改一下以前登记的地址，方便以后的邮寄。最后，客户也许想退回某一种商品。研究此类过程有助于定义事件。



图5-11　导致许多事件的某一特定客户的“事务”序列

3. 技术依赖事件和系统控制

有时，分析员很关心那些对系统很重要，但不直接影响用户和事务处理的事件。这样的事件一般包括设计选择和系统控制。在分析过程中，分析员会将这些事件暂时放在一边。然而，在设计阶段这些事件却很重要。

一些影响设计的事件包括外部事件，这些外部事件包括使用物理系统，如登录等。尽管这类事件对系统的最终使用很重要，但具体实现细节可以暂缓考虑。在这一阶段，分析员只

需要将精力集中在功能需求——即系统需要完成的工作中。逻辑模型不需要指明如何去实现系统，因此在模型中应该省略实现的细节。

这类事件大多数包括系统控制。系统控制是为保证系统完整性而加入的防范和安全程序。例如，为了实现系统安全控制需要登录系统。其他的控制可以保证数据库的完整性，如每天备份数据。这两种控制对系统来说都是非常重要的，所以在设计阶段就会加入系统中。但在分析阶段花时间在这些控制上只能是仅仅在需求模型中增加了一些用户毫不关心的细节（用户认为信息服务本身自然会考虑这些细节）。

系统控制：为保证系统完整性而加入的防范和安全程序。

一个有助于确定哪个事件应该用于控制的方法是：假定技术是理想的。理想的技术假设认为：只有在最佳条件下系统才需要做出响应。这样的事件才应该在分析阶段被考虑进去。所谓的最佳条件是指没有设备损坏，处理和存储能力没有限制，用户操作完全遵照系统要求且没有误操作。通过假定这个技术是理想的，分析员能排除诸如数据库被备份了这类事件，因为可以假定磁盘永远不会损坏。之后，在设计阶段，项目小组把这些控制加进来，因为此时这个技术假定显然已经不理想了。图5-12所示为一些在设计阶段之前都可以忽略的事件。

理想的技术假设：该假设假定只有在最佳条件下系统才需要做出响应，这样的事件才应该在分析阶段被考虑进去。



图5-12 在设计阶段之前可以忽略的事件

5.2.4 落基山运动用品商店实例中的事件

落基山运动用品商店（RMO）客户支持系统包括许多事件。其中许多事件和刚才所讨论的很相似。图5-13列出了一些外部事件。其中一些重要的外部事件都与客户有关：客户想确定商品情况，客户发送订单，修改订单或取消订单。而其他一些外部事件和RMO部门有关：收到订单发货，销售部门给客户发送促销材料，销售部门更改商品目录。分析员可以询问所有需要使用系统为其服务的人和部门，从而确定出外部事件列表。

客户支持系统还有很多临时事件，见图5-14。这些事件大多数为有关部门生成周期性报表：生成订单汇总报表、生成完成情况汇总报表、生成商品目录活动报表。分析员通过确定系统需要定期生成的常规报表和说明来确定临时事件的列表。

顾客想确定商品信息的有效性
顾客发送订单
顾客修改或取消订单
顾客或管理人员想查看订单状态
发货部门按订单发货
发货部门确认延期订货
顾客退回商品（因为商品有缺陷或顾客改变了主意而全部或部分退回商品）
潜在顾客索要商品目录
顾客修改账户信息
市场部门想给顾客发送促销材料
管理人员调整对顾客的收费（更正确来的缺陷或是打了折扣）
销售部门修改商品目录（增加、修改、删除记录及更改商品价格）
销售部门开始促销某种商品
销售部门制定新的商品目录

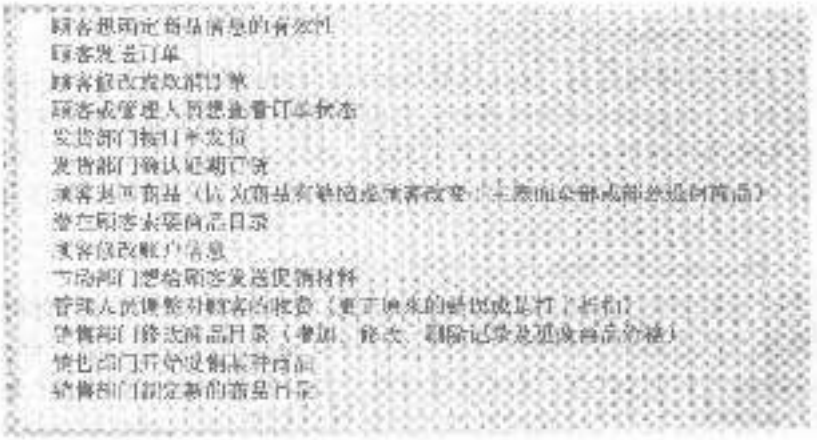


图5-13　落基山运动用品商店（RMO）客户支持系统的外部事件

生成订单汇总报表
生成交易汇总报表
生成完成情况汇总报表
生成未来顾客活动报表
生成顾客调整/打折报表
生成商品目录报表



图5-14　落基山运动用品商店（RMO）客户支持系统的临时事件

5.2.5　关注每个事件和由此产生的用例

确定事件列表时，分析员要注意每个事件附加的信息，以备将来使用。对于每个事件来说，要确定的最重要的信息就是系统需要响应的用例。这些信息被填入事件表中。一个事件表包括行和列，描述事件及其细节信息。事件表中的每行都记录了一个事件及其用例的信息，表中的每列代表了事件及其用例的一个关键信息。图5-15列出了事件客户想确定商品信息是否有效的信息。注意，由此产生的用例称为查询可用的商品。

事件表：一个用例列表，该表以各个事件为行，以各个事件的关键信息为列。

事件：引起系统执行某种操作的事件

来源：对外部事件而言，外部实体就参与者是数据进入系统的来源

响应：系统产生了什么样的输出结果（如果有的话）

事件	触发器	来源	活动用例	响应	目的地
客户想确定可用商品信息	商品查询	客户	查询可用商品	可用商品细节	客户

触发器：系统如何知道事件发生了？对外部事件而言，就是进入系统的数据。对于时事件来说，就是定义好的。触发系统运行处理过程的时间点

注例：当事件发生时系统做了什么？用例是为功能需求定义重要事件

目的地：哪个外部实体获得了输出结果

图5-15　事件表中每一事件的信息

事件表的信息描述事件和由此产生的用例的重要方面。首先，对每个事件来说，系统怎么知道其一事件发生了呢？用来通知系统某一事件发生了的信号称为触发器。对于一个外部事件，触发器就是系统必须处理的数据到达了。例如，当客户发送订单时，新订单的详细信息就可以作为输入数据。知道数据的来源也很重要。在本例中，新订单的信息来源是客户——一个外部实体或参与者。对于临时事件，触发器是某一个时间点。例如，在每天生意结束时系统就知道到了生成交易汇总报表的时刻了。

触发器：用来通知系统某一事件发生了的信号。这一事件可以是需要处理的数据到达了或到了一个时间点。

来源：为系统提供数据的外部实体或参与者。

其次，当事件发生时，系统该做些什么呢？系统所做的或者说系统对事件的响应称为用例。当客户发送订单时，系统就执行用例生成一张新订单。当到了生成交易汇总报表的时刻，系统就执行用例生成日销汇总报表。

最后，用例导致系统产生什么响应呢？响应是系统的输出结果。当系统产生交易汇总报表时，那些报表就是输出结果。一个用例可能会有多个响应。例如，当系统生成一张新订单后，系统需要把订单确认信息发给客户，把订单详细内容送到发货部门，而把交易记录发送到银行。目的地就是系统发送响应（输出结果）的地方，也就是外部实体或参与者。有时用例根本不需要响应。例如，如果客户想修改账户信息，那么新信息被记录在数据库中，但不需要产生任何输出结果。在数据库中记录信息就是用例的一部分。

响应：系统产生的一个输出结果，该结果将被送到某个目的地。

目的地：接收系统输出数据的外部实体或参与者。

事件列表，以及每个事件触发器、来源、用例、响应和目的地，都可以放在事件表中并跟踪记录以便将来使用。事件表是用来记录有关信息系统需求关键信息的比较方便的方法。图5-16所示为落基山运动用品商店（RMO）客户支持系统的事件表。在第6章使用传统方法绘制数据流图来定义功能需求时会用到这张表。在第7章使用面向对象方法绘制用例图和写用例描述的时候也会用到这张事件表。

实践指导

使用事件表作为用例的信息导出对系统进行功能需求。 ■

5.3 问题域的事物

定义系统需求用到的另一个关键概念包括对事物的理解和建模。系统需要存储这些事物的信息。对于用户来说，他们在工作中需要处理的诸如产品、订单、发票和客户等都可以看做是事物，这些事物也必须是系统的一部分。他们就是经常在系统的问题域中提到的事物。例如，一个信息系统需要存储客户和产品信息，因此分析员就有必要考察清楚有关的信息。通常这些事物类似于与系统交互的外部实体或参与者。例如，客户（外部实体）发送了订单，但是系统需要存储这个客户的有关信息。但在其他情况下，这些事物又不同于外部实体。又例如，没有产品的外部实体，但系统却需要存储关于产品的信息。

在传统的开发方法中，这些事物构成了系统存储信息的相关数据。对于任何一个信息系统来说，需要存储的数据类型肯定是信息系统需求的一个关键方面。在面向对象的方法中，这些事物就是在系统中相互交互的对象。无论使用哪一种方法来开发信息系统，识别和理解这些事物都是关键的初始步骤。

客户支持系统事件表					
事件	触发器	来源	活动用例	响应	目的地
1. 客户要确定商品信息的有效性	商品查询	客户	查询可用商品	有效商品信息细节	客户
2. 客户发送订单	新订单	客户	生成新订单	实时连接 订单确认 订单细节 交易处理	信用卡部门 客户 发货部门 银行
3. 客户修改或取消订单	订单修改请求	客户	修改订单	修改确认 订单修改细节 事务处理	客户 发货部门 银行
4. 生成订单汇总报表的时刻	周末、月末、季度末、年末		生成订单汇总报表	订单汇总报表	管理部门
5. 生成交易汇总报表	每天下班时		生成交易汇总报表	交易汇总报表	会计
6. 客户或管理人员想查询订单状态	订单状态查询	客户或管理人员	查询订单状态	订单状态细节	客户或管理人员
7. 发货部门按订单发货	订单完成通知	发货部门	记录订单完成情况		
8. 发货部确认延期订货单	延期订货单通知	发货部门	延期订货单通知	延期订货单通知	客户
9. 客户退货	订单退回通知	客户	生成退货订单	退货确认 事务处理	客户 银行
10. 生成完成情况汇总报表的时刻	周末、月末、季度末、年末		生成完成情况汇总报表	完成情况汇总报表	管理部门
11. 潜在客户索取商品目录	索取商品目录的请求	潜在客户	提供商品目录信息	商品目录	潜在客户
12. 生成潜在的客户活动报表的时刻	月末		生成潜在的客户活动报表	潜在的客户活动报表	市场部门
13. 客户修改账户信息	客户账户信息修改通知	客户	更新客户账户信息		
14. 市场部门制给客户发送促销品	促销品细节	市场部门	分发促销品包	促销品包	客户和潜在客户
15. 管理人员调整对客户的收费	客户收费的调整	管理人员	生成客户收费调整记录	收费调整通知 事务处理	客户 银行
16. 生成客户调整租赁情况报表的时刻	月末		生成客户调整报表	客户调整报表	管理人员
17. 销售部门修改商品目录	商品目录修改细节	销售部门	修改商品目录		
18. 销售部门促销某种商品	促销商品细节	销售部门	生成促销商品记录		
19. 销售部门制订新的商品目录	新商品目录的细节	销售部门	生成新商品目录	商品目录	客户和潜在客户
20. 生成商品目录活动报表	月末		生成商品目录活动报表	商品目录活动报表	销售部门

图5-16 落基山运动用品商店（RMO）客户支持系统的完整事件表

5.3.1 事物的类型

与定义事件时一样，分析员应该与用户讨论他们日常工作中处理的事物类型。分析员可以提出几种事物类型，帮助识别所讨论事物的类型。一些事物是实实在在的，因此容易识别，但另一些事物却难以明了。不同类型的事物对不同的用户都很重要，为此有必要从所有类型的用户处收集信息。

图5-17所示为一些常见事物类型。实实在在的事物通常很明显，如飞机、书或交通工具。在落基山运动用品商店（RMO）的事例中，商品目录和其中的每个商品都是重要的实实在在的事物。在信息系统中另一类常见的事物类型是人所充当的角色，如雇员、客户、医生或病人。在落基山运动用品商店（RMO）的事例中，客户显然就是一个非常重要的角色。



图5-17 事物类型

其他类型的事物可以包括组织部门，如分区、部门或工作组。同样地，地点或位置在某些特定的信息系统中也是十分重要的，如仓库、商店或部门办公室。最后，有关订单、服务电话、合同或航班等如此类的突发事件或重要的交互行为信息可以被看成一件事物。在落基山运动用品商店（RMO）的事例中，一张订单、一次发货和一次退货都是重要的事件。有时这些事件被认为是事物之间的关系。例如，订单是客户和某一库存商品间的关系。最初分析员仅仅简单地把这些作为事物罗列出来，然后根据不断的分析和设计方法的要求对其加以调整。

分析员通过考察事件列表中的事件和咨询用户来确定这些事物的类型。例如，对于每个事件来说，它影响了哪些事物（系统需要知道这些事物，并存储其信息）？当客户发送订单时，系统需要存储该客户的信息、客户订购的商品和订单本身的详细信息，如日期和支付条款。

5.3.2 开发事物初始列表的过程

从刚刚讨论的一般准则可以看出，分析员可以使用很多信息来源来开发一个系统用来存储信息的事物初始列表。可以遵循这样一个很有用的过程，即从列出用户在谈论系统时提到的所有名词开始。比如，可以把事件表中的事件、活动或用例、外部实体或动作参与者、反触发器和响应等作为潜在的事物。然后将出现在关于已有系统的信息中的其他的名词或在与系统相关者讨论中出现的名词添加到列表中。

1. 第一步 使用事件表和关于每一事件的信息，确定所有名词

对于落基山运动用品商店（RMO）客户支持系统，这里的名词包括RMO、客户、产品项、

订单、确认单、交易、运输、银行、需求改变、汇总报表、管理、交易报表、账户、延期订单、延期订单通知、退货、退货确认、完成报表、潜在客户、目录、市场、客户账户、促销资料、收费调整、目录细节、销售及目录活动报表。

2. 第二步　使用已有系统、当前过程及当前报表或表单中的其他信息添加必要信息的项目或种类

对于落基山运动用品商店（RMO）客户支持系统，这些项目必须包括更详细的信息，比如价格、尺寸、颜色、样式、季节、库存数量、支付方式、运输地址等。其中，一些可能是另外一类，而另一些可能是已定义事物的具体内容（称为属性）。

3. 第三步　将列表精简并记录假设或要检查的问题

上述名词的列表创建以后，有必要对其进行精简。通过对每个名词提出以下问题来确定该名词是否应当含在列表中：

- 是系统需要了解的独特的事物吗？
- 在你所工作的系统的范围之内吗？
- 系统需要记住多于这些项目中的一个吗？

对每个名词提出以下问题来确定是否应该从列表中将其排出：

- 与你已经定义的某个其他事物是同义词吗？
- 真的是从你已经定义的其他信息中产生的系统输出吗？
- 这一输入真的可以导致记录一些你已经定义的其他信息吗？

对每个名词提出下列问题来确定是否应该对其进行进一步的研究：

- 可能是你已经定义的其他事物的一些具体信息（属性）吗？
- 是当假设变化时你可能需要的事物吗？

图5-18所示为RMO客户支持系统事件表及其他资源中的一些名词，并对每个名词都进行了注释。

建议的名词	将该名词作为事物来考虑的一些注释
业务人员	我们知道都有哪些人，不需存储
延期订单	一个具备确认单类型，还是订单状态值？需要考虑
延期订单确认	从其他信息产生的输出
银行	仅有一个，不必存储
目录	需要记住，不同季节和年份的目录。应该保留
目录活动报表	可以从其他信息产生的输出，不必存储
目录细节	与目录类似吗？或者与这个目录中产品项类似？需要研究
改变请求	将导致对订单改变进行记录的输入
收费调整	导致一次交易的输入
颜色	关于产品项的一条信息
确认	从其他信息产生的一个输出，不需存储
信用卡信息	订单的一部分？还是客户信息的一部分？需要考虑
客户	是一个关键事物，带有很多细节，应该保留
客户账号	如果包含RMO支付计划，则是必要的，需要研究
完成报表	从运输信息产生的输出，不需存储
库存数量	关于产品项的一条信息，需要研究
产品项	RMO在含在目录中用于销售的。应该保留
管理人员	我们知道都有哪些人，不需存储

图5-18　RMO系统中关于“名词”的事物的部分列表

确定的名词	将该名词作为事物来存储的一些注释
市场人员	我们知道都有哪些人，不需存储
销售人员	我们知道都有哪些人，不需存储
订单	关键的系统应应，应该保留
支付方式	订单的一部分，需要研究
价格	产品项的一部分，需要研究
促销资料	一个输出？还是存储在范围之外的文档？需要研究
潜在客户	可能与客户类似，需要研究
退货	订单的反面，应该保留
退货确认	退货信息产生的输出，不必存储
RMO	只有一个，不必存储
承诺	日录的一部分？还是还有更多的？需要研究
运输	用于跟踪的关键事物，应该保留
运输人员	他们常有变动且我们需要跟踪订单，应该保留
运输部门	我们的部门，不需存储
运输部门地址	客户的一部分？还是订单的？还是运输的？需要研究
尺寸	产品项的一部分，需要研究
样式	产品项的一部分，需要研究
信息报表	从其他信息产生的输出，不必存储
交易	每个交易都很重要并需要记住，应该保留
交易报表	交易信息产生的输出，不必存储

图5-18 （续）

5.3.3 事物间的关系

对上述列表中的事物进行记录和精简后，分析员进一步研究和记录其他的信息。事物间的很多关系对系统非常重要。关系是指某些事物间自然发生的联系，如客户发送订单及雇员在某一部门工作等（见图5-19）。我……订购和工作于就是两个自然的发生在特定事物间的关系。信息系统需要存储雇员和部门的信息，但同样重要的是，系统也需要存储某些关系的信息——比如，John在财务部工作，Mary在市场部工作。同样，存储由John Smith发出的订购衬衫的1043号订单也是非常重要的。

关系：某些事物间自然发生的联系，比如客户发送订单及雇员在某一部门工作等。

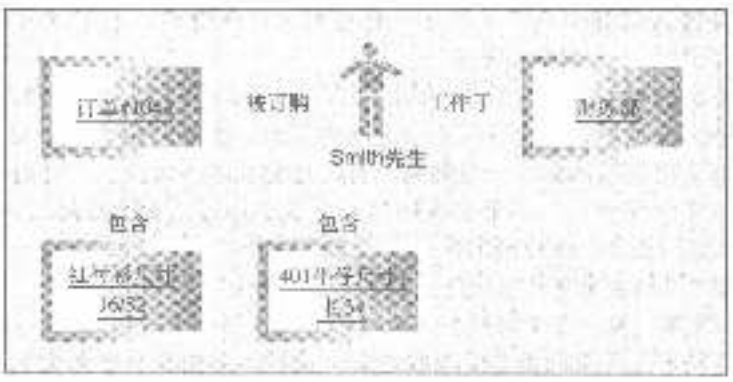


图5-19 事物之间自然发生的关系

事物间的关系是两方面的。例如，客户发送订单描述的是一个方面的关系。类似地，订单由客户发送描述的是另一个方面的关系。理解关系的双向性是很重要的，因为有时候系统从一个方面记录关系比从另一个方面记录关系重要得多。例如，落基山运动用品商店（RMO）肯定需要知道客户订购了什么商品，这样才能准备发货。但是，公司要知道所有订购了某一特殊商品的客户名单这一需求起初并不明显。如果公司要给所有订购了残次品或需要取回货品的客户发通知，该怎么办呢？知道这个信息是十分重要的，但是操作用户却可能无法马上从识到这一点。

根据每件事物的关联数目来理解每种关系的本质是非常重要的。例如，一个客户可能发送了许多份订单，但是一张订单只能被一个客户发送。所以关联数目被称为关系的基数。基数可以是一对一、一对多的。此外，基数是针对于关系的每一个方面提出的，在面向对象的方法中，术语重数常常表示关联的数目。图5-20所示为有关订单基数/重数的例子。



图5-20　关系的基数/重数

基数：发生在事物间关联的数目，例如一个客户发送了多个订单及一个雇员在一个部门工作等。

重数：基数的同义词（用于面向对象的方法）。

有时需要描述的不仅仅是基数，而且要描述基数可能的取值范围（即基数的最小值和最大值），例如，某个客户可能从来没有发送过订单。在这种情况下关联数为0，或者该客户发送过一次订单，此时存在一个关联，最后客户可能发送了两张、三张甚至更多的订单。因此，客户发送订单这个关系有一个范围0、1或更多，通常记为0或more。0是基数的最小值，"more"是基数的最大值，被称为"基数的限制"。

在某些情况下，至少需要一个关联（一个和可选关系正相反的强制关系）。例如，只有当某客户发送了订单，系统才开始记录该客户的信息。因此，基数范围可解释为"客户发送了一个或多个订单"。

一个一对一的关系也可以精炼成包括最小值和最大值的基数。例如，一个订单是由一个客户发送的——如果没有客户也不可能有订单。因此，一是最小的基数值。这就是强制关系。由于每张订单不可能对应多个客户，因此一也是最大的基数值。有时这种关系可以解释为"一个订单必须且只能由一个客户来发送"。

这里描述的关系是两种不同类型事物间的关系——如客户和订单。这种关系被称为二元关系。有时关系是同一类型的两件不同事物之间的关系。例如，婚姻这个关系就是两个人之间的关系。这种类型的关系被称为一元关系，有时也称为回归关系。一元关系的另一个例子是组织体系，在这个体系中，一个单位要向另一个单位报告，如包装部门向发货部门报告。发货部门再向调度部门报告，调度部门再向市场部门报告。

二元关系：两件不同类型事物之间的关系，如客户和订单的关系。

一元（回归）关系：同一类型的两个事物间的关系，如一个人和另一个人的婚姻关系。

关系还可以存在于二种不同类型的事物之间，这种关系被称为三元关系。甚至还可以存在于多种不同类型的事物之间，此时这种关系被称为n元关系。例如，某一张订单可能和某客户、和某个销售代理之间有关联，这就是三元关系。

三元关系：三种不同类型事物之间的关系。

n元关系：n（n为任意数）种不同类型事物之间的关系。

存储关系的信息和存储事物的信息同样重要。记录每个客户的姓名、地址信息虽然很重要，但记录该客户订购了那些商品也同样重要，甚至可能更为重要。

实践指导

初期中于确定问题域中的每一个"事物"，同时也要集中于他们之间的联系/关系，这些对于系统用户来说同样重要。■

5.3.4 事物的属性

如前面的图5-18讨论的那样，大多数信息系统都存储并使用每个事物的一些具体信息。这些特定的信息被称为属性。例如，每个客户都有姓名、电话号码和信用限额等信息。每条信息都是一个属性。分析员需要明确每个系统需要存储的事物属性，一个属性可以用来唯一地标识某事物。如雇员的社会保险号码、其次交易的订单号。能唯一地标识事物的属性被称为标识符或关键字。有时标识符是已经存在的（如雇员的社会保障号、车牌或商品的编号）。有时系统需要分配一个具体的标识符（如发票号、交易号）。

属性：有关事物的一条特定信息。

标识符（关键字）：能唯一地标识事物的一个属性。

一个系统可能需要记录很多相似的属性。例如，一个客户有好几个名字——首名、中间名和姓，而且可能还有一个绰号。复合属性是指包括了许多相关属性的属性，因此分析员可以选择一个复合属性来代表所有的名字，也许可以将这个复合属性命名为客户全名。一个客户也可能有多个电话号码——住宅电话号码、办公室电话号码、传真号码和移动电话号码。分析员可能开始时描述最重要的属性，然后再不断扩展列表，因此属性列表可能会非常长。图5-21所示为一个客户属性及其相应的取值。

复合属性：包括了许多相关属性的属性。

所有客户具有如下属性	每个客户的每个属性都有一个值		
客户编号	101	102	103
名	John	Mary	Bill
姓	Smith	Jones	Casper
住宅电话	555-9182	423-1298	874-1297
单位电话	555-3425	423-3419	874-8546

图5-21 属性及其对应的值

5.3.5 数据实体和对象

到目前为止，当描述对于系统很重要的事物时，主要用到的例子都是系统需要存储其信息的事物。在传统的系统开发方法中，这些事物被称为数据实体。数据实体、数据实体间的关系和数据实体的属性都可以使用实体－联系图（ERD）来建立模型。计算机处理数据通过实体间的相互作用，生成数据实体，修改属性值及把一个实体和另一个实体联系起来。实体－联系图（稍后将详细介绍）可以用来建数据库（通常是关系数据库）。在系统开发的结构化方法和信息工程方法中都用到了实体－联系图。

数据实体：系统需要存储的有关信息系统传统开发方法的信息。

另一种思考事物的方法是把事物看成在系统中彼此相互作用的对象。在面向对象方法的用户环境中的对象（有时也叫做问题域中的对象）类似于传统方法中的数据实体。二者的区别在于，系统中的对象不仅存储信息，而且具有一定的功能。换句话说，对象既具有属性又具有行为。这个简单的差别却对认识和建立系统的方法产生了巨大的影响。在需求建模的早期阶段，它与传统方法相比几乎没有什么区别。图5-22从数据实体和对象的角度，比较了传统方法和面向对象方法。

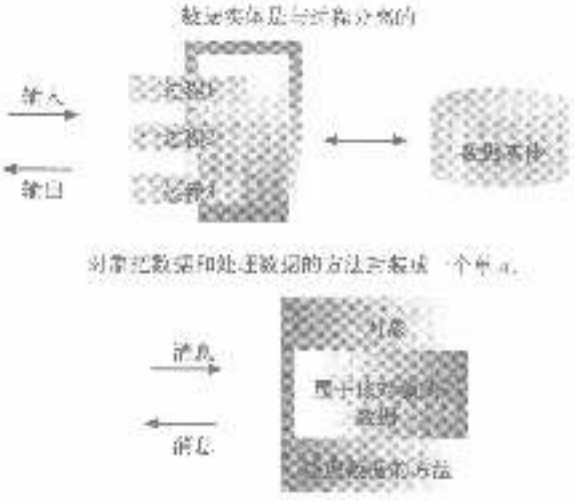


图5-22　数据实体和对象的比较

在面向对象方法中，每个特定事物就是一个对象（如图5-21中的John、Mary、Bill），事物的类型被称为类（在本例中是客户）。之所以使用类这个词，是因为所有的对象都要按照事物的类型来进行分类，类、类之间的联系和类的属性可使用类图来建立模型。此外，类图还表示出了该类对象的行为，称为方法。

类：所有相似的事物所属的类型或分类。

类的方法就是类的所有对象所具有的行为。行为就是对象自处理的操作。对象可以在需要的时候修改自身的属性值，而不需要外部处理程序来修改这些属性值。要让一个对象执行某种操作，可以让另一个对象给该对象发送一个消息。一个对象可以给其他对象发送消息，也可以给用户发送消息。如同第2章介绍的那样，整个信息系统实际上就是相互作用的对象集合。

方法：类的所有对象所具有的行为。

由于每个对象都有属性值和对属性进行操作的方法（包括对象的其他行为），因此可以说一个对象被封装（即覆盖或保护）成为一个封闭的单元。本章随后将继续讨论类图。

封装：把所有对象覆盖或保护起来，使其包含属性值和对这些属性进行操作的方法，从而使得对象成为自我封闭的（或受保护的）单元。

5.4　实体－联系图

传统的系统开发方法（即第2章介绍的结构化技术和信息工程技术）都把重点集中在新系统的数据存储需求上。数据存储需求包括数据实体、数据实体的属性及它们之间的关系。正如前面提到过，用来定义数据存储需求的模型被称为实体－联系图（ERD）。

5.4.1　ERD概念的实例

在实体－联系图中，矩形代表数据实体，连接矩形的直线代表数据实体间的关系。图5-23所示为一个简化的实体－联系图，图中有两个数据实体——客户和订单。每个客户可以订多个订单，但每个订单只能由一个客户订。从一个方面来看基数是一对多，而从另一个方面来看基数则是一对一。连接订单实体的直线端有一个像“乌鸦爪”的符号，该符号表示“多个”订单。关系线上其他的符号代表基数的最小值、最大值限制。参见图5-24对关系符号的解释。图5-23的模型实际上表示，一个客户最少可以订0个订单，最多可以订多个订单。从另一个方面来看，该模型表示，一个订单必须且只能由一个客户来订。这种标记方法表示了精确的系

统约束。这个限制反映了管理部门制订的业务策略。而分析员则必须发现这些策略。分析员不能随意决定两个客户不可以共享一个订单，但是管理部门却可以制订这样的策略。

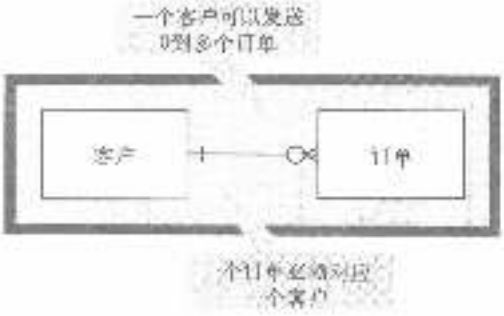


图5-23 一个简单的实体-联系图

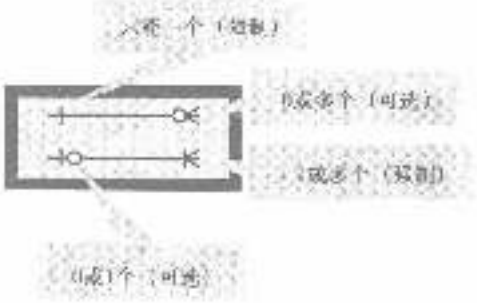


图5-24 关系的基数符号

图5-25所示为一个扩展了的模型，其中包括了订单中具体商品条目（一个订单中包含一个或多个商品），每个订单最少有一个商品，最多可以有很多种（不存在一个商品都没有的订单）。例如，一个订单可能包含了一件衬衫、一双鞋和一条皮带，每件商品都和该订单关联。这个例子也列出了每个数据实体的属性：每个客户都有一个客户编号、姓名、地址和几个电话号码。每个订单有一个订单号、订货日期等。订单中的每件商品有商品ID、数量和价格。每个数据实体的属性都列在其名字的下面。而关键标识符属性列在第一列。



图5-25 带有属性的扩展ERD图

图5-26所示为实际的数据在事务处理中呈现的形式。客户John下了两个订单。第1个是在2月4日订的，订了两件衬衫、一条皮带。3月29日他下了第2个订单，包括一双靴子、两双旅鞋。客户Mary还没有发过订单。记住，一个客户可以订0到多个订单。因此，Mary和任何订单都没有关联。Sara在3月30日订了二双凉鞋。

分析员在建模的过程中，常常对这个ERD图进行细化，细化的一个方法就是分析多对多关系。图5-27所示为一个多对多关系的示例。在大学中，课程通常是作为课程项提供的。一个学生可以选填多个课程项，每个课程项也可以让多个学生选填。因此，课程项和学生之间是多对多关系。在很多情况下都存在多对多的关系，可以使用两端带有“乌鸦爪”符号的直线表示这种关系从而给这些关系建模、使用。如果关系数据库是根据ERD图中的多对多关系来设计的，由于关系数据库不能直接实现多对多关系，所以必须建立一个单独的表，该表包括了关系两端的关键字。第12章将详细介绍关系数据库。

然而，随着分析的深入，我们通常会发现多对多关系还需要存储别的数据。例如，在图5-27的ERD图中，每个学生在某门课的成绩该存放在什么地方呢？这是非常重要的数据。尽

数据模型显示了一个学生选修了哪一课程项。但是模型中却没有存储成绩。解决的方法是增加一个数据实体，该实体表示学生和课程项之间的关系，有时把它称为关联实体。没有存储的数据就作为关联实体的属性。图5-28所示为包含关联实体——课程注册的扩展ERD图，此时学生的成绩就是课程注册这个关联实体的一个属性。

关联实体：表示两个数据实体间多对多关系的数据实体。

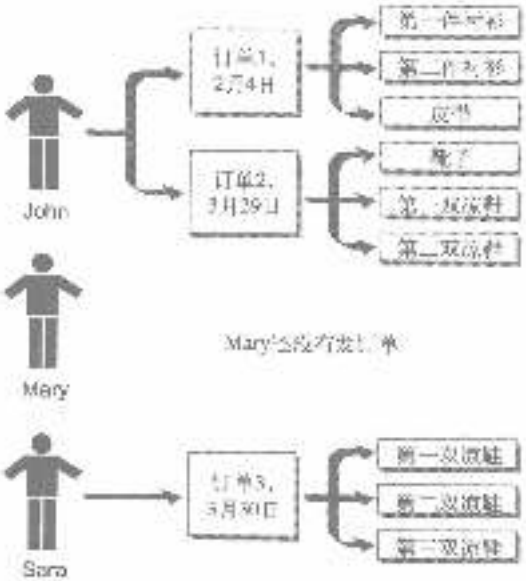


图5-26　客户、订单和订单中的商品

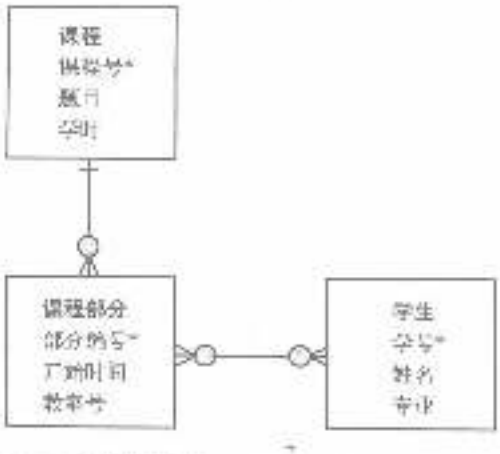


图5-27　大学课程注册ERD图（含有多对多关系）



图5-28　细化的大学课程注册ERD图（包含关联实体）

从左向右看图5-28中的关系，这个ERD图表示一个课程部分对应许多课程注册，每个都对应各自的成绩，而每个课程注册又对应一个具体的学生。从右向左看，表示一个学生对应很多课程注册，每个都对应各自的成绩，而每个课程注册又对应一个具体的课程部分。用这个模型实现的数据库将能够产生成绩列表，列出所有学生每门课程对应的成绩及每个学生的成绩单。

在建模的过程中，还要对ERD图进行其他方面的细化。一个重要的细化过程是被称做规范化的过程，在设计关系数据库时会使用到该过程，我们将在第12章详细讨论这方面的内容。

5.4.2 落基山运动用品商店实例的ERD图

落基山运动用品商店（RMO）的实体-联系图就是把前面提到的客户和订单的例子修改了一下，大部分数据实体都来自于图5-18。图5-29所示为修改后完整的模型图，但是其中没有表示出属性，通过这些属性能够更容易地集中于数据实体及其关系。

每个客户可以发送0到多个订单，每个订单中包含一个到多个商品，也就是说一个订单可以订一件衬衫和两件毛衣，订单上的每件商品都和库存中的商品对应。对于衬衫来说就是有特定的颜色和尺寸，尽管该属性在图中没有表示出来，每种库存商品还应该有一个属性来记录这种颜色和尺寸衬衫的库存量，由于有很多颜色和尺寸（每一种都有自己的数量），每种库存商品还和产品相关联，在该产品中还记录了每种商品的共性（供货商、性别、描述）。

一个早期版本的模型描述每个产品项包含在一个或多个目录中，而每个目录包含一个或多个产品项，是一个多对多的关系。由于这个关系有些属性，特别是正常价格和优惠价格，需要记录下来，因此，这个模型在目录和产品条目之间添加了一个称为目录产品的关联实体，每个目录可以对同一个产品列出不同的价格（比如滑雪鞋在春季目录中可能更便宜）。

落基山运动用品商店（RMO）的实体-联系图中也包含了货运信息。由于该图中包含了订单需求，但不包含零售项目，订单中的每件商品都是货运的一部分。一次发货可能包含多个订单中的商品，而每次发货都只由一个发货员来处理。



图5-29 落基山运动用品商店客户支持系统的实体-联系图（ERD）（图中未显示有关属性）

图5-29所示的实体-联系图包括了许多关于系统数据存储需求的具体信息。请确认你可以根据图示理清所有的关系，并且试着描述出包含在一个订单中所有数据实体的例子。试着列出每一个数据实体的关键属性来验证你的理解。画出与图5-26类似的草图来显示ERD中所描述的真实数据。你也可以先看看下节要介绍的类图及第12章的关系数据库设计来验证你的理解。

一旦模型建立起来以后，模型像实体-联系图一样也需要仔细地分析一遍，就像分析一遍整个程序的逻辑结构一样。因此，正如第4章所讨论的那样，能把所有的模型分析一遍并加以“调试”是系统开发中的一项重要技能。

为了检查对这张图的理解，考虑一下，一张订单中的商品能不能由不同的发货员来发货，如果能，如何在这张图中表示出来？答案是能。实际上订单中的一些商品有可能是延期订货，因此当这些商品最终发货时可能就在另一批发货中了。处理这批发货的就可能是另一个发货员。

模型中体现的其他需求是每个订单包括1到多个订货交易。一个订货交易是指该订单的付款和退款记录。在客户第一次支付订单账款时，创建一个订货交易。然而在此之后，客户可能在订单中增加一件商品，就需要额外付款，这就包含了第2次订货交易。最后，客户也许退回一件商品，就需要退款，这样就创建了第3次订货交易。

5.5 类图

面向对象的方法也强调对用户工作所包含事物的理解，就像前面所讨论的那样，这种方法给对象建立类模型，而不是建立数据实体。与数据实体类似的是对象类也有属性和关联。

基数（称为面向对象方法中的多样性）的概念也同样适用于类。我们前面已经介绍过，二者的差别主要在于对象既存储信息也执行系统中的实际处理过程。这些处理过程（对象的行为）可以执行是因为对象既有属性又有方法。由于对象具有行为，因此传统方法和面向对象方法的需求模型在形式上会大不相同。此外，设计模型也是明显不同的。追起初定义需求的时候，建模方法与面向对象的方法是相似的。

类图用来显示系统对象的类。符号基于标准建模语言（UML），UML已经成为面向对象系统开发的建模标准。一种UML类图描绘用户工作领域的事物，被称为域模型类图。另一种UML类图用于在设计软件类时创建设计类图。我们简要的讨论这两种UML类图。

在类图中，矩形代表类，连接矩形的线代表类之间的关系。图5-30所示为一个类——客户。类的符号是一个由三部分组成的矩形。头部包括类名，中间部分列出类的属性，底部列出类的重要方法。类图通过类和类之间的关系绘制成。首先要会学习创建域模型类图的UML符号。前面用到的很多实体—联系图的读者将会利用UML类图符号重新绘制，这样就能加以比较了。实际上，现在的很多开发者甚至在传统方法中使用UML类图取代ERD。接下来，将会介绍类图中用到的其他层次，然后介绍更多有关设计类图的符号。



图5-30　UML类符号（由类名、属性和方法三部分组成）

5.5.1　域建模类图符号

图5-31所示为一个简化的域模型类图，由顾客、订单和订单条目三个类组成。每一个类只包括两部分，方法在这个域模型类图中没有显示。实际上，分析员经常用只有两部分的类符号来表明这是一个域模型。UML要求类名以大写字母开头，属性以小写字母开头。在图中可以看到，每一个顾客可以下多个订单，每一个订单只能由一个顾客来订。为了更清楚的表示，"订购"和"组成"的关系也可以引入图中，不过这些细节是可选的。从一个方面来看重数是一对多的，而从另一个方面来看重数则是一对一的。重数符号，如订单类旁边点线上的星号所示，表示有多个订单。图5-32对重数符号进行了总结。另一个关系表明一个订单包括一个或多个订单条目，一个订单条目只能关联一个订单。



图5-31　一个简单的域模型类图

实践指导

首先把重点集中于问题域类，它们是用户工作环境中的"事物"，而不是最终要设计的软件类。■

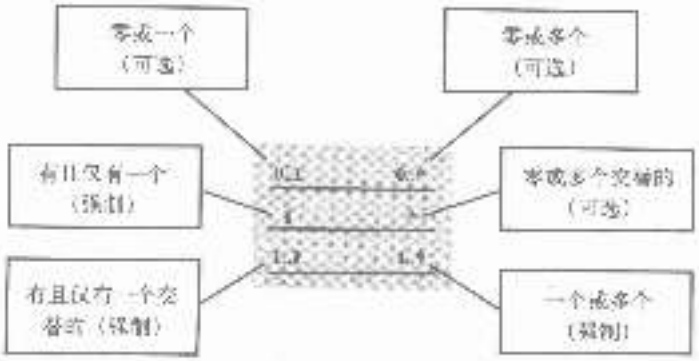


图5-32 多重关系

图5-33所示为一个课程注册的例子作为域模型类图。回顾一下，一个课程可能有0个或多个课程部分，每一个课程部分可以有0个或多个学生注册，每一个学生可以注册0个或多个课程部分，这是一个多对多关系。但是因为必须要存储每一个学生的课程的成绩，所以模型必须要修改，就像在ERD模型里一样，如图5-34所示，类图里增加了一个称为课程注册的关键是来存储成绩属性。一条虚线将关联类连接到课程部分和学生之间的关联线。

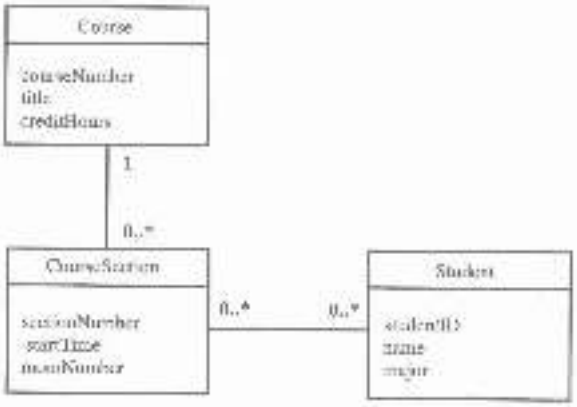


图5-33 一个具有多对多关系的大学课程注册域模型类图

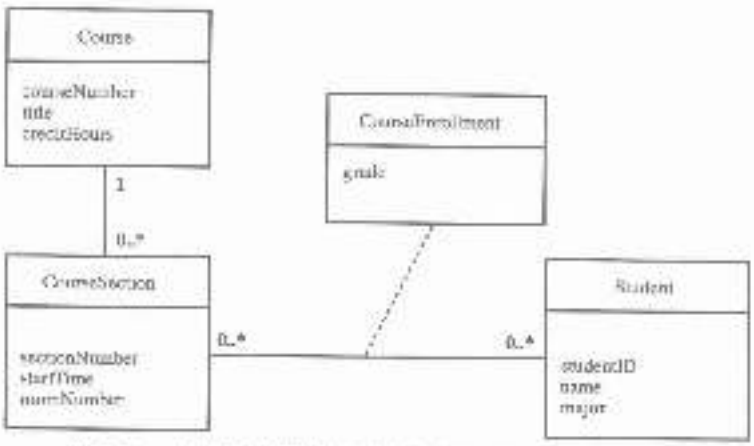


图5-34 一个优化的具有关联类的大学课程注册域模型类图

从左向右看图5-34中的关系，该类图表示一个课程部分对应许多课程注册，每个都对应各自的成绩，而每个课程注册又对应一个具体的学生。从右向左看，该类图表示一个学生对应很多课程注册，每个都对应各自的成绩，而每个课程注册又对应一个具体的课程部分。在这个域模型基础上实现的系统能够产生成绩列表，列出所有学生每门课程对应的成绩及每个学生的成绩单。

5.5.2　有关对象类的更复杂的问题

尽管有关事物的问题并不是面向对象方法所特有的，但是使用面向对象的方法会比使用传统的方法能到更多的问题。这些问题就是人们用来理解现实世界中事物的两种附加的方法：概括/具体层次图和整体—部部层次图。本节将介绍这些概念以及如何用类图来对其进行表现。

1. 概括/具体层次图

概括/具体层次图是基于人们按照事物的异同来将其分类的思想建立的。概括就是把相似类型的事物进行分组。例如有很多种类的机动车辆——小汽车、卡车和坦克。所有的机动车辆都有某种共同的特点，因此机动车辆就是一个更概括的类，具体就是把不同种类的事物进行分类——例如某类小汽车中包括跑车、轿车和体育用车。这些小汽车在某些方面相似，而在其他方面却不同。因此，跑车就是小汽车中的一个具体类型。

概括/具体层次图：把类按照从最概括的父类到最具体的子类的顺序进行排列的层次图，有时也被称为继承层次图。

概括/具体层次图用来把事物按照从最概括到最具体的顺序进行排列。如前面所介绍的那样，分类就是定义事物的类。在层次图中的每个类的上面也许有更一般的类，这个类称为父类。同时，每个类的下面也许有更具体的类，这个类称为子类。在图5-35中，一个小汽车有3个子类和1个父类（机动车辆）。UML类图符号用一个指向父类的三角来表示概括/具体层次图。

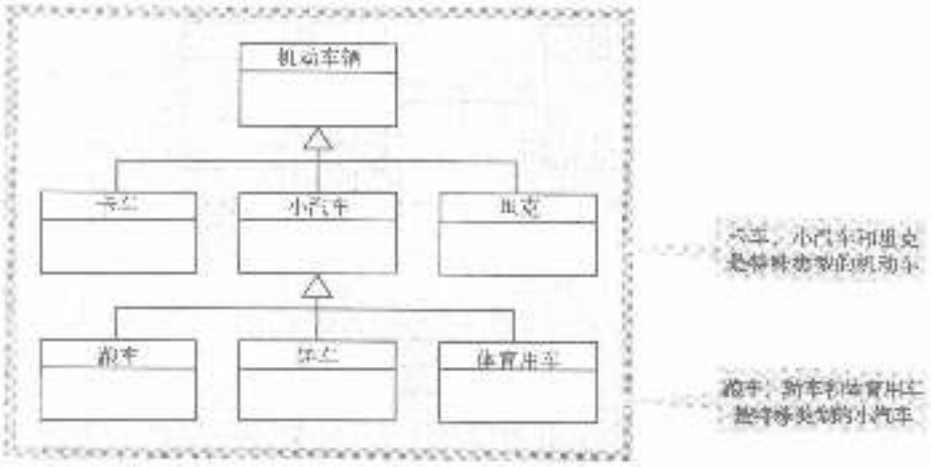


图5-35　机动车辆的概括/具体层次图

我们已经提到过，人们是使用概括/具体层次图来理解现实世界的。也就是说人们是通过把某些知识领域细化分类来学习的。一个知识丰富的银行家可以具体地讲解贷款和存款账户的种类。一个像落基山运动用品商店的John Blankens那样经验丰富的商人可以把各种户外运动和服装的种类说得清清楚楚。因此，当分析员询问用户的工作时，必须要努力去理解用户在工作中使用的知识，并把这些知识按照概括/具体层次图表示出来。从某种意义上来说，开

发现基山运动用品商店新的客户支持系统的动机就是因为John认识到RMO必须要有一个新系统来处理各种特殊类型的订单（网上订单、电话订单和邮件订单）。图5-36所示为这些特殊的订单类型。

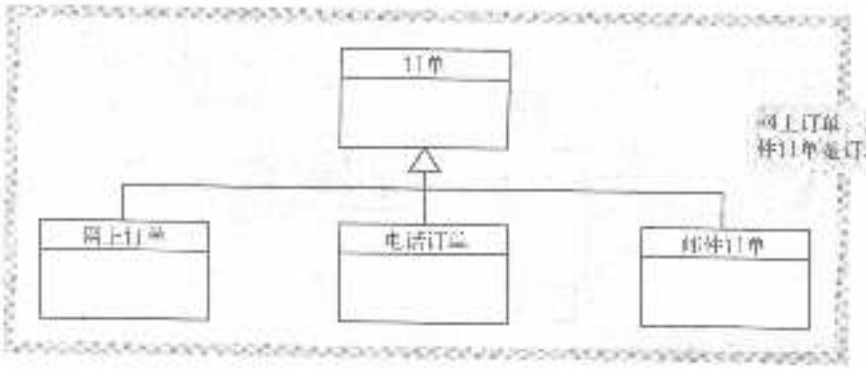


图5-36 订单的概括/具体层次图

继承允许子类共享其父类所具有的特征。回到图5-35，小汽车也是机动车辆，但是它有更具体的特征：跑车也是小汽车，不过多了些别的特征，从这一点来看，子类继承了父类的特征。在面向对象的方法中，继承是一个关键的概念，这是由概括/具体层次图所决定的，有时这种层次图也被称为继承层次图。

继承：允许子类共享其父类所具有的特征的概念。

2. 整体－局部层次图

人们认识事物信息的另一种方法是根据它们的各个部分定义它们。例如，学习计算机系统可以使你认识到计算机是由不同的部分组成的，这些不同部分是：处理器、主存、键盘、磁盘存储器和监视器。键盘并不是计算机的一种特殊类型，而只是计算机的一个部分。然而，就键盘本身来说它是一个完全独立的事物。整体－局部层次图描述了这种在人们试着将对象及其组件联系起来时所发现的关系。

整体－局部层次图：按照类之间的关联组件将类进行结构化的层次图。

整体－局部层次图有两种类型：聚合以及合成。术语聚合用于描述一种关联形式，这种关联详细说明了集合（整体）及其组件（局部）之间的整体－局部关系，这里的各个部分都可以独立存在。图5-37说明了计算机系统中聚合的概念，图中用菱形符号来表示聚合。术语合成用于描述更强的整体－局部关系，其中的各个部分一旦关联，就不能够独立存在。常用实心的菱形符号来表示合成。

聚合：对象及其各个部分之间的一种整体－局部关系。

合成：对象及其与它不可分割的各部分之间的一种整体－局部关系。

整体－局部层次图，不论是聚合或合成，主要使得分析员可以描述类之间关联的细微差别。对于任何关联关系，基数/重数都适用，比如计算机可以有一个或更多的磁盘存储设备。

5.5.3 设计类图符号

目前我们见到的UML类图的例子都是域模型类图，设计类图是对类图的细化，也用于在新系统中表现软件类。在第11章中将会学到将域模型类图转换为设计类图的步骤。暂时只需要记住设计类图表现更多关于软件的实际设计。图5-38中的类图包括一些方法来强化软件类

共有属性和方法这种概念。

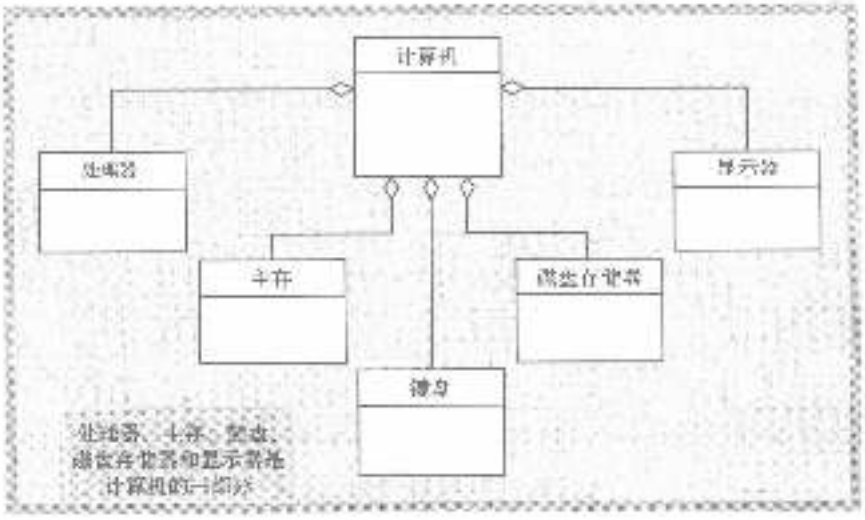


图5-37　计算机及其各部分之间的整体－局部（聚合）关系

图5-38所示为一个维护银行账目系统的设计类图的一部分。这个系统包括客户类。由于客户类的方法是标准的，故此图中没有显示。每一个类都设计成知道如何创建一个新的自身的实例，更新自身属性值并报告自身信息。账目类列出了两个方法，因为它们是银行账目所独有的，并且是系统处理过程的中心。这两个方法包括存款和取款。

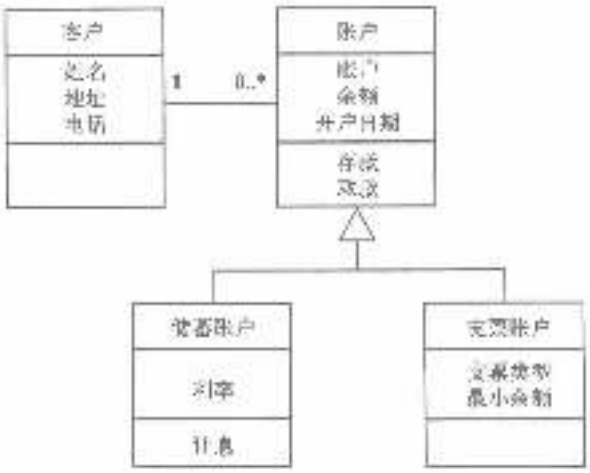


图5-38　银行账目系统设计类图

银行账目系统包括一张账括/具体层次图：账目是父类，而储蓄账目和支票账目则是两个子类。在连接类的一条线上所画的三角符号表示继承。子类从父类中继承属性和行为。因此支票账目类从账目类继承了两种方法以及所有的属性。类似地，储蓄账目类继承了同样的方法和属性，但是储蓄账目类知道怎么计算利率，而支票账目则不能，从而使得一些属性和方法对这两个子类都是共有的，而其他一些属性和方法则不能共有。

在这个例子中，继承就意味着当储蓄账目对象创建（或生成实例）时就需要为4个属性赋值，3个继承于账户类，一个是自身具备的。而支票账目对象需要为5个属性赋值，3个继承于账户类，两个是自身具备的。支票账目对象和储蓄账目对象一样都可以用于储蓄，储蓄账目对象可用于计算利率，而支票账目对象则不可。每个对象或实例可以维护信息，并且可用于调用它的某个方法。

客户类和账户类之间存在一个关联。每一个客户可以有零个或多个账户，还要注意，储蓄账户和支票账户类继承了与客户的关联关系，因此一个客户事实上与储蓄账户或支票账户有对应的关联关系。实际上，银行没有提供任何类似简单账户类的类，这个类只是用来让特

殊类型的账户去继承属性、方法和关联。账户类是一个抽象类，它是不能被实例化的。如图5-38所示，这个类图中的抽象类的名字用斜体表示。支票账户、储蓄账户和客户都是可以实例化的具体类的例子。

抽象类：一种不能被实例化（即不能创建对象）的类，它为了使其子类能够继承它的属性、方法及关联。

具体类：能被实例化的类（即可以创建对象）。

图5-39所示为带有方法的课程注册设计类图。课程类中包含一个添加课程的方法。课程部分类可以打开进行注册，然后关闭。学生类使得一个学生可以被接收并最终获得学位。一个学期一旦结束，课程注册类可以邮寄成绩。

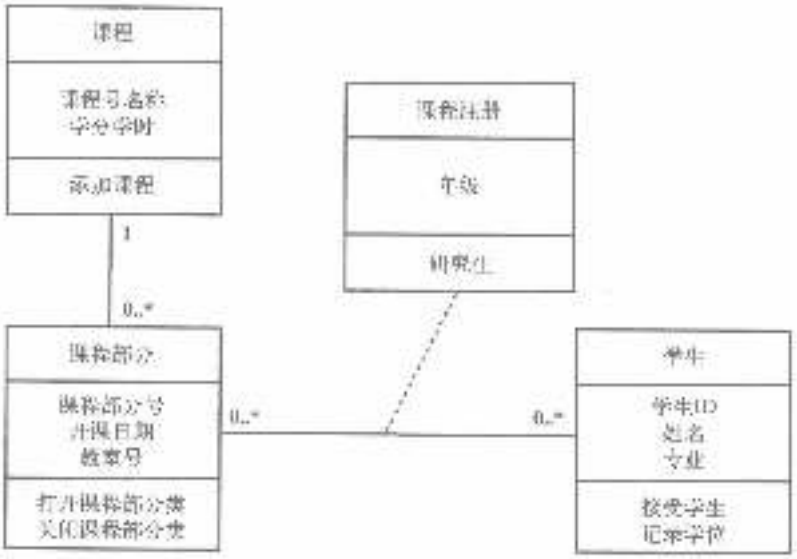


图5-39 带有方法的大学课程注册设计类图

这个例子中没有标示出概括/具体层次，但在一个大学课程注册系统中一些特殊类型的学生（本科生和研究生）及特殊类型的课程（有学分和无学分）是可以存在的。图5-40所示为课程注册实例的一个扩展版本。这里加入了学期类，因为课程是在特定的学期开设的。该图还显示了从学生类到本科生类和研究生类的概括/具体层次图。例如，每一个本科生类的对象是一个特定类型的学生，了类继承父类所有的属性、关联和方法。学生类用斜体表示是因为他是一个抽象类。

属性上也展现了其他的一些符号。在构造类时，确定主键非常重要。在UML中，属性的性质由花括号体现，课程类中的课程号{主键}就是如此。其他类也有主键。属性课程部分数量加下划线来表明它是一个类级别的属性。通常情况下，对于某个属性一个对象具有唯一的值。然而，类级别的属性具有一个值，这个值用于这个类所有的对象。在Java中通过静态属性实现，在Visual Basic.NET中通过共享属性实现。在这个例子中，系统要存储一个值来体现可供选择的课程部分数量，每添加一门新课程，这个值增一。

另外，还用到其他的一些通用类图符号。类级别的方法（静态方法或共享方法）也被加以下划线（图5-40没有体现出来）。一些属性是计算值，前面加以斜线（/）的属性表明这个属性是计算得到而不是存储的。例如，属性课程部分数量是一个计算值，所以它将显示为“课程部分数量”。在第11章中将会介绍很多类图的细节问题，这些会作为面向对象设计的一部分

进行学习。在这一点上，记住需求是通过域类型建模的，这些域类型不显示方法。设计类型会显示方法和其他一些软件的具体细节。我们在这里展示出更加完善的UML类型只是为了展示大部分重要的类型符号。

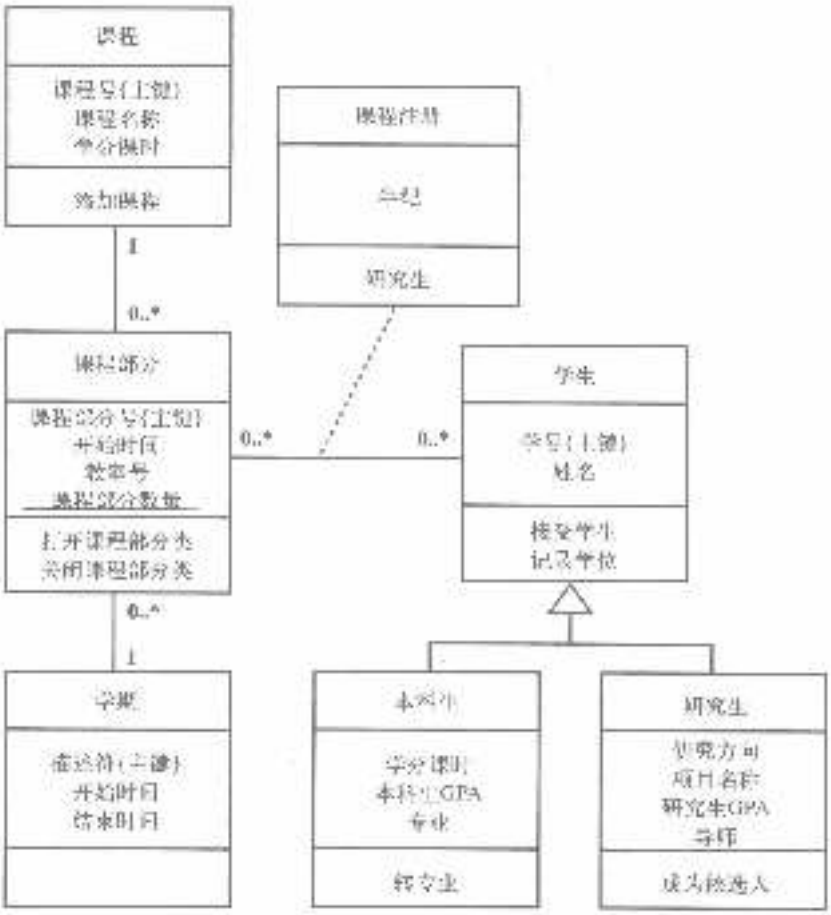


图5-40　扩展的课程注册设计类图

5.5.4　落基山运动用品商店实例的域模型类图

落基山运动用品商店的域模型类图如图5-41所示。它与前面图5-29所示的实体-联系图非常类似。尽管与ERD相同，在展示模型的概图时，类图中的属性可以省略，但图中示出了所有类的主要属性。

图中包含了概括/具体层次，说明订单可以是前面讨论的三种类型的一种——网上订单、电话订单和邮件订单。注意，所有类型的订单都具有订单的属性，而每种特殊类型的订单又都有一些额外的属性。订单是一个抽象类（名字用斜体表示），因为任何订单都是三种特殊类型之一。

其他类和类之间的关联与RMO实体-联系图类似。产品目录是附属于目录和产品条目之间关联的一个关联类。关联关系的重数用最小值和最大值来表示。图5-41中没有示出整体-局部关联（聚合和合成），尽管可能由诸如订单交易是订单的一部分或者产品条目是目录的一部分等引起争论。整体-局部关系及关联关系在实现的时候是类似的，因此在本例中没有差别。

许多分析员在进行业务系统类图模型建立时在类图中并不将聚合和合成标出。

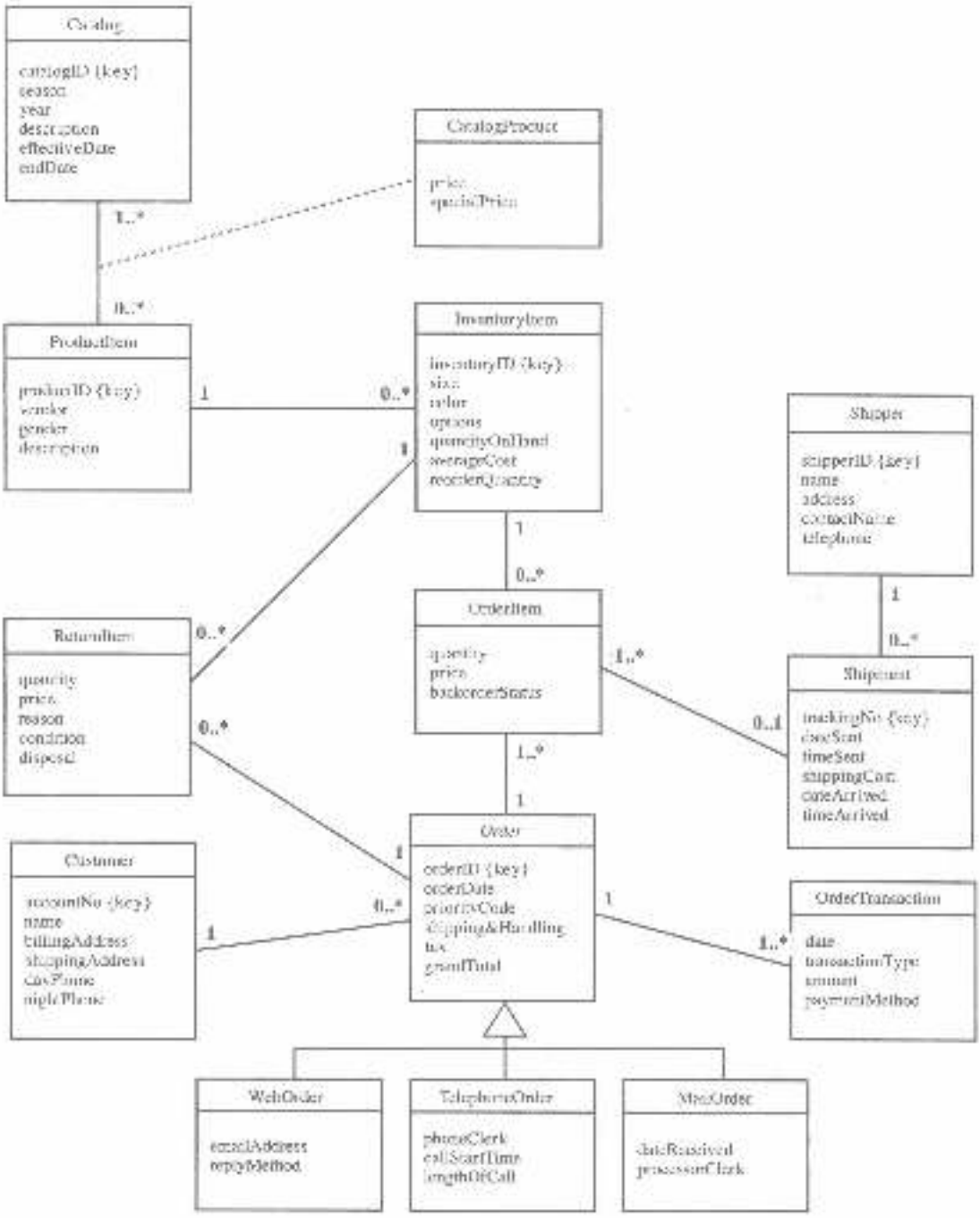


图5-41 萍基山运动用品商店域模型类图

注意，在这张图中没有显示出方法，在项目的系统分析阶段生成的初始类图将不包括方法（前面已经提到），称为域模型类图。随着在分析和设计阶段对对象行为的进一步研究，才将方法添加到类图上。现在，请记住每一个类对象不仅有属性而且有行为。例如，发货员类

知道如何创建一个新的发货员对象，如何删除一个发货员对象，如何更改发货员的名字或地址，以及如何连接到某一发货部门。同样，所有其他的类也都具有这些标准的能力。

5.6 目标

在分析阶段新系统生成的需求模型时可能有很大差异。这种差异取决于项目小组是使用传统方法还是使用面向对象方法。本章讨论的事件（触发用例和用户问题域里的）和事物这两个关键概念都是建模过程的起点。我们将在接下来的两章中分别讨论这两种方法，它们都是以相同的初始信息开始的。图5-42所示为定义事件和事物后如何区分两种方法。

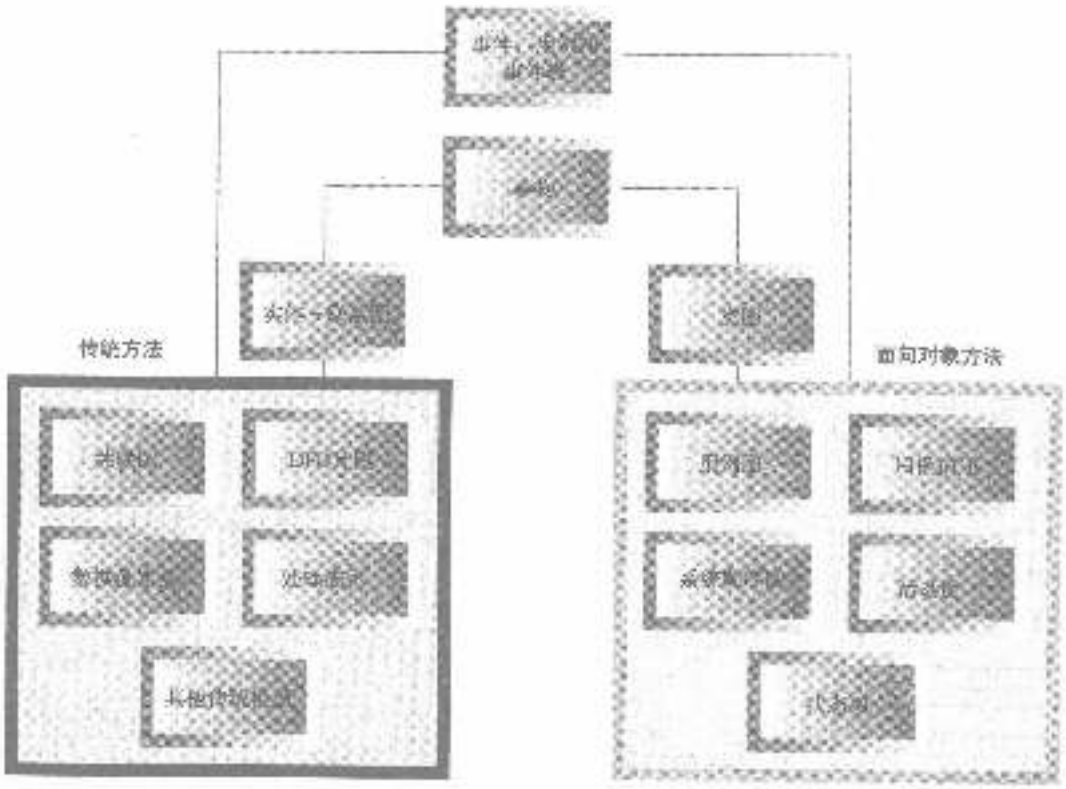


图5-42　传统方法和面向对象方法的需求模型

传统方法是获得事件表中的用例并根据表中的信息生成一组数据流程图（DFDs），这些图包括关联图、DFD片段、实体—联系图（ERD）定义了包括在DFD中的数据存储需求。需求的其他信息包括数据流定义和过程描述。这些模型和其他一些传统模型将在第6章中介绍。

面向对象的方法是首先获得事件表，然后生成用例图和一组用例描述。用例描述和类图用于生成系统需求的其他模型，包括活动表、顺序图和状态图。我们将在第7章中介绍这些模型。

小结

本章是介绍系统功能需求建模技术的三章中的第1章，在这一章中强调了在分析阶段完成的被称为定义系统需求的这一项工作。我们生成系统各个方面的模型来记录需求，触发用例的事件和用户工作环境中的事物是所有系统开发方法中针对需求模型的两个最关键的概念。传统的方法使用实体—联系图（ERD），而在面向对象方法中使用类图为问题域的主要模型。

模型是十分有用的，原因是：它包括了生成模型时发生的学习过程，能降低系统的复杂性，记录需要记忆的许多细节，可以同小组成员和用户进行交流，并记录下将来系统支持时的系统需求。在建模过程中，可以使用许多类型的模型，包括数学模型、描述模型和图形模型。本章介绍了许多图形模型，其中一些模型用于分析阶段，而另一些模型则用于设计阶段。

在建模初期中关键的一步是识别和定义系统功能需求的活动或用例。通过确定需要系统做出响应的事件可以很好的发现用例。事件是可以被描述的、有记录价值的在某一特定时间和地点发生的事情。外部事件发生于系统之外，通常是由操作系统的用户触发的。临时事件发生于一个确定的时间点，如每天工作结束或每个月末。静态事件的发生基于系统内部的变化。每个事件的信息都记录在一张事件表中，表中列出了事件、事件触发器、触发来源、系统必须完成的用例、作为系统输出的响应及响应的目的地。

其他的关键概念包括用户做系统需要记住的工作时处理的事物，例如产品、订单、发货单和客户。在用户处理的事物中有许多自然发生的关系：一个客户发出订单，一张订单要求一张发票。关系的基数（重数）是指包括在关系中的关联数：一个客户可以发出多张订单，而一张订单只能由一个客户发出。属性是事物某一方面的信息，如客户的名字或地址。传统的方法把这些事物看做存储数据的数据实体。面向对象方法则把这些事物看做属于一个类的对象。这个类不仅具有属性而且具有行为（称为方法）。尽管这两种开发方法使用同样的基本概念来定义数据实体和对象，但是由于对象行为，使得对象在面向对象方法中使用的方式和在传统方法中使用的方式有很大差别。

传统的方法使用实体—联系图来表示数据实体、数据实体的属性及数据实体之间的关系。面向对象方法使用UML类图来表示相同信息，称为域模型类图。类图符号也用来生成设计类图。设计类图显示每一个类的方法。在类图中使用的另外两种概念（尽管有时它们也用于实体—联系图）是：概括/具体层次图和整体—局部层次图。前者允许了类继承父类，而后者允许一组对象及其各部分关联成一个整体。

接下来的两章将分别讨论传统方法和面向对象方法中产生的需求模型。

关键术语

abstract class	抽象类
aggregation	聚合
associative entity	关联实体
attribute	属性
binary relationships	二元关系
cardinality	基数
class	类
composition	合成
compound attribute	复合属性
concrete class	具体类
data entities	数据实体
descriptive model	描述模型
destination	目的地
elementary business process(EBP)	基本业务流程
encapsulation	封装
event	事件

event decomposition	事件分解
event table	事件表
external event	外部事件
generalization/specialization hierarchies	概括/具体层次图
graphical model	图形模型
identifier(key)	标识符（关键字）
inheritance	继承
mathematical model	数学模型
methods	方法
multiplicity	重数
n-ary relationship	*n*元关系
perfect technology assumption	理想的技术假设
relationship	关系
response	响应
source	来源
state event	静态事件
system controls	系统控制
temporal event	临时事件
ternary relationship	三元关系
trigger	触发器
unary（recursive）relationship	一元（回归）关系
use case	用例
whole-part hierarchies	整体-局部层次图

复习题

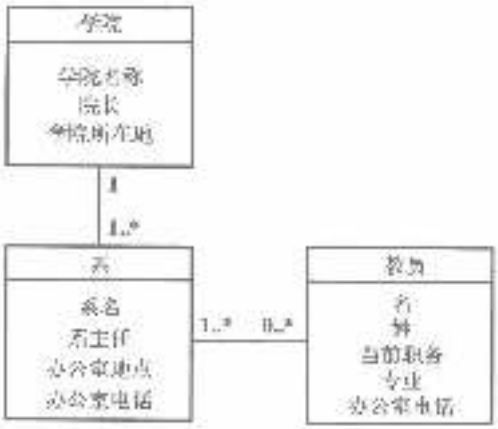
1. 在系统开发期间建立模型的原因是什么？
2. 模型的三种类型是什么？
3. 用于定义系统需求的两个关键概念是什么？
4. 什么是事件？
5. 事件的三种类型是什么？
6. 哪一种类型的事件导致数据输入系统？
7. 哪一种类型的事件发生于确定的时间点？
8. 哪一种类型的事件不会导致数据输入系统，但总是能导致一个输出结果？
9. 哪一种类型的事件可以称为雇员辞职？
10. 哪一种类型的事件可以称为该发薪水了？
11. 举出一些系统控制例子。
12. 理想的技术假设指的是什么？
13. 事件表中的列表示什么？
14. 什么是触发器？什么是来源？什么是活动或用例？什么是响应？什么是目的地？
15. 在传统方法使用的模型中“事物”称为什么？
16. 在面向对象方法中“事物”称为什么？
17. 什么是关系？

18. 什么是关系的基数（也称为重数）？
19. 描述一下实体-联系图中如何表示最小和最大基数。
20. 什么是一元、二元和n元关系？
21. 什么是属性和复合属性？
22. 什么是关联实体？
23. 实体-联系图中的符号表示什么？
24. 类图中的符号表示什么？
25. 除了属性值之外对象中还封装了哪些部分？
26. 什么是概括/具体层次图？
27. 子类从超类型的类继承属性和方法？
28. 整体-局部层次图的两种类型是什么？
29. 一个类的哪三条信息被放置在类图符号的三个部分中？
30. 类图中连接类的一条线上的三角符号表示什么？
31. 抽象类和具体类的区别是什么？
32. 关联类在类图上如何表示？
33. 在域模型中显示了什么类型的类？

思考题

1. 举例说明三种类型的模型。要求这些模型可以用来设计汽车、房屋及信息系统。
2. 请解释需求模型为什么是逻辑模型而不是物理模型。
3. 请解释基本业务流程（EBP）在确定用例时的重要性。
4. 回想一下图5-8所示的外部事件检查列表，然后联系一个大学课程注册系统，举例说明在事件检查列表中的每一种类型事件，使用命名外部事件的方法来命名每一个事件。
5. 回想一下图5-9所示的临时事件检查列表，学生成绩单应该是内部输出还是外部输出？教员类列表应该是内部输出还是外部输出？课程注册系统其他的一些内部和外部输出是什么？使用命名临时事件的方法，你将为触发输出结果的事件取什么名字？
6. 在课程注册系统中，为学生注册课程这一事件创建一项事件表。在这张表中列出了事件、触发器、来源、用到、响应和目的地。为产生成绩单时刻事件创建另一项事件表。
7. 考虑如下客户在银行中发出的一系列活动，哪个活动是分析员应该为银行账目交易处理系统定义的事件？①Kevin得到了奶奶送给他的一张作为生日礼物的支票；②Kevin想买一辆小汽车；③Kevin决定把钱存下来；④Kevin来到了银行；⑤Kevin排队等候；⑥Kevin在他的储蓄账户里存了一笔钱；⑦Kevin得到了储蓄存折；⑧Kevin索要了一本介绍汽车贷款的宣传手册。
8. 考虑一下"乐观的技术假设"，这种假设认为：只有当系统在最佳条件下需要做出响应时，事件才应该在分析阶段被考虑进去。根据这种假设在基山运动用品商店的事件表中的任何事件是否能被除去，解释一下：为什么只有在非完美情况下，才需要输入登录系统和备份数据的时刻这样的事件？
9. 为下列事件画出一张包括最小和最大基数的实体-联系图：系统存储两件事物——汽车和汽车拥有者——的信息。汽车有牌子、型号和出厂日期等属性。汽车拥有者有姓名和职业等属性。假设一辆汽车必须有一个拥有者，而一个拥有者可以拥有许多汽车。但一个拥有者也可能没有任何汽车（也许她刚刚卖掉了所有的汽车，但是你仍然需要为她在系统中保留一条记录）。
10. 为在上面思考题9中所述的汽车和汽车拥有者画一幅类图，并在图中画出具有特定属性的跑车、轿车和小型货车子类。

11. 考虑一下图5-28所示的实体-联系图。在这幅细化了的ERD图中显示了具有一个关联实体的课程注册系统。这个模型是否允许一个学生一次注册多门课程？是否允许一个课程部分包含多名学生？是否允许学生注册同一课程的不同部分并为每个注册取得成绩？是否存储所有学生的所有课程的成绩信息？
12. 再考虑一下图5-28所示的实体-联系图。在图中增加下列信息并列出你的所有假设。一个教员通常教多门课程，但有的学期也许一门课也不教。每个课程必须至少有一个教员，但有时好几个小组教一门课程。此外，为了确保所有的课程是相似的，通常指定一个教员作为课程协调员来监督课程，而且每一个教员也可以是多门课程的协调员。
13. 如果在上题中所画的实体-联系图显示了教员和课程之间是多对多的关系，那么进一步分析你也许会发现需要存储一些其他的信息。这些信息包括什么？（提示：对每一门课程教员是否有具体的上课时间？对教员上完每门课程你是否给出了评价？）扩展ERD图来存储这些额外的信息。
14. 为上题中完成的课程注册系统画出类图。一定使用正确的关联类的符号（见图5-34）。
15. 考虑一个系统，在这个系统中需要存储大学计算机实验室中的计算机信息，如每台计算机的特性和位置。在模型中需要包括哪些事物？在这些事物中有哪些关系？这些事物包括哪些属性？最后为该系统画一个实体-联系图。
16. 为上题描述的计算机实验室系统画一个类图。
17. 参考图5-38中所示的银行账目类图，扩展这个模型来显示具体的客户类型：个人客户和商业客户。所有的客户都有名字和邮件地址。商业客户有一些其他的属性如客户估价分类，联系人和联系人电话。个人客户有住宅电话和单位电话等属性。此外，扩展这个模型显示银行只有多个部门，并且每种账目由一个部门提供服务。当然，每个部门可以处理多种账目。
18. 考虑图5-41所示的露营山运动用品商店的域模型类图。如果建立网上订单，它应该有多少种属性？如果建立电话订单，它应该有多少种属性？如果一个客户用电话订购了一件商品，在这项交易中总共生成了多少个新对象？
19. RMO中的产品条目和库存条目并不完全相同。一个产品条目是诸如Leather'R'Us公司提供的男式皮衣、猎服等事物，而一个库存条目是指指定具体的颜色和尺寸，如一个大小中等的棕色皮猎服。如果RMO在其目录中增加了一种新的滑雪衫品种，而库存中只有3种颜色和6种尺寸可供使用，那么总共需要增加多少个对象？
20. 考虑下列有关学院、系和教员之间关系的域模型类图，如下图所示。



a. 在模型中显示了哪些关系？
b. 一个教员有多少种属性？哪些属性（如果有的话）是从另一个类中继承来的？

c. 如果向系统中增加1个大学、1个系和4个教员的信息，那么有多少个对象要加到系统中？

d. 一个教员可以同时在多个系工作吗？请解释一下原因。

e. 一个教员可以同时工作在两个不同的系吗？假如这两个系一个是在商学院，而另一个系是在艺术和科学学院，请解释一下原因。

实验练习

1. 与一个图书管理员交流一下，请图书管理员介绍一下在什么情况下图书馆有系统需要做出响应。列出这些外部事件。然后询问一下图书管理员有关要求系统生成报告书、通知、报表或其他输出的时间点或最后期限。列出这些临时事件。用这种方式描述系统对图书管理员来说是否自然。类似地，请图书管理员介绍一些系统需要存储的事物信息。试一下你是否能够让图书管理员列出事物的重要属性并描述出事物之间的关系。描述这些事物对图书管理员来说是否自然？根据你所获得的信息创建一张ERD图或类图。

2. 参观一个餐馆或大学食堂并同服务员谈话（或与一个是服务员的朋友谈话）。就像你在练习1中所做的一样，找出外部事件、临时事件和被操纵你或对象。在一个餐馆中订单处理事件是什么样的事件？完成事件表，然后画出ERD图或类图。

3. 回想一下大学课程注册过程并同咨询人员、注册人员和所在系的人员进行交谈。考虑一下整个学期的进行顺序。学生触发的事件是什么事件？所在系触发的事件是什么事件？导致信息流向学生的临时事件是什么事件？导致信息流向教员或系的临时事件又是什么事件？

4. 再回想一下你自己大学的有关信息。使用域模型类图符号为下列类型生成概括/具体层次图：① 系类型；② 学生类型；③ 课程类型；④ 助学金类型；⑤ 房屋类型。在上述的每一种情况下都要求包括父类和子类的属性。

实例研究

Spring Breaks'R'Us旅游服务预订系统

Spring Breaks'R'Us旅游服务预订系统（SBRU）公司负责为在校大学生提供春假旅游服务。每年秋天，旅游供应的宾馆向SBRU提供有关春假期间每周可用的房间、房间大小及房间占用率等信息。因为每个宾馆在每个季节提供不同时间长短的房间预订，并且预订的房间的占用率随着不同的星期有所变化。宾馆通常有可用的不同大小的大量房间，因此大学生可以预订适当的房间。例如，两人可以预订一个双人房间，而四人可以预订一个四人房间。

在每年的12月，SBRU生成一张宾馆、空闲星期、房间占用率的列表，然后将这张表分发给全国各个大学的校园代理人。当一组学生提出在某一星期预订某一宾馆房间的请求时，SBRU为这些学生指定具有足够空间的房间，并向每一个学生发送一个确认通知。当春假的截止日期来到时，SBRU向每一宾馆发送一张随后几周的学生预订房间列表。当学生到达宾馆时，他们直接向宾馆支付房间费用。宾馆直接向SBRU的账目系统发送到帐支票。这个账目系统独立于预订系统。当春假结束时学生就可安全返校读书了。

1. SBRU预订系统必须对什么事件做出响应？建立一张完全的事件表，在这张表中包括事件、触发器、类型、用例、响应和每一事件的目的地。确保只考虑预订系统中的触发处理过程的事件，而不要考虑SBRU账目系统或宾馆使用的系统所触发的事件。

2. 列出所提到的数据实体（或类），列出每一数据实体（或类）的属性。列出数据实体（或类）之间的关系。

3. 哪些类可以进一步细化为一张概括/具体层次图？为它们中的每一个列出父类及其子类。

房地产多编目服务系统

房地产多编目服务系统向本地房地产经纪人提供一些信息，这些信息可以帮助他们向客户销售房屋。每个月，经纪人通过与房主签订合同列出待售的房屋列表。经纪人为房地产公司工作，这家公司向多编目服务公司发送列表上的房屋信息。因此，在社区中的任何代理机构都可以获得列表上的信息。

列表中的信息包括地址、建造年代、面积、卧室个数、浴室个数、房主名字、房主电话号码、房屋要价和状态代码。任何时候，代理机构都可以直接请求获得和客户要求相匹配的列表信息。因此代理机构可以向多编目服务公司发出请求。多编目服务系统提供房屋信息、列出房屋经纪人的信息及经纪人工作的房地产公司的信息。例如，一个经纪人也许把给列表上的代理人打电话询问一些其他的问题，或者他也许想直接给房屋主人打电话约好时间看房子。多编目服务公司每月两次（每月15号和30号）出版包含所有列表信息的书。这些书被送给所有的房地产经纪人。许多房地产经纪人还得到这本书（这本书比较容易浏览）。因此尽管信息经常是过时的，但仍然会提供这本书。有时经纪人和房主要改变列表信息，如降低价格、更正以前的房屋信息或标明房屋已出售，当经纪人要求房地产公司做出以上改变时它就向多编目服务公司发送这些变化请求。

1. 对于那些事件多编目服务系统必须做出响应？建立一张完整的事件表，在这张表中列出事件、触发器、来源、用例、响应和每一事件的目的地。

2. 画出一张表示多编目服务系统的数据存储需求的实体-联系图，在图中要包括以上所提到的属性。你的模型是否包括了卖方、买方和结算的数据实体？如果确实如此，请重新考虑一下。包括多编目服务系统需要存储的信息在内的这些信息也许与房地产公司需要存储的信息有所不同。

3. 画出和ERD图相应的域模型类图，但在类图中要显示出具有不同属性的列表类型。本例中的描述假设所有的列表是一个家庭使用的房屋。多家庭或商业务列表又怎么样呢？

国家巡查罚单处理系统

国家巡查罚单处理系统的目的是记录驾驶员的违规情况，保存驾驶员支付的罚款记录（当驾驶员接受罚款或被法官发现行车违章时），并通知法官应对罚款不能及时支付的违章人员发出逮捕令。一个独立的国家巡查系统负责记录事故情况并查证经济责任（保险单）。而第二个系统负责根据罚单和事故记录为保险公司生成驾驶记录表。最后，第四个系统负责发放、恢复或吊销驾驶员的执照。这四个系统显然是集成的因为它们共享同一个数据库。但除此之外，它们由国家巡查的不同部门独立使用。国家巡查的操作（警察所做的）是完全独立的。

用于罚单处理系统的数据库部分包括驾驶员数据、罚款数据、警察数据和法官数据。驾驶员数据、警察数据和法官数据由系统使用。系统生成和维护罚款数据。驾驶员的属性包括执照号码、名字、地址、出生日期和执照批准日期等。罚单的属性包括罚款号码（每一个号码都是唯一的，并且预先打印在警察罚款本的每一张表单上）、位置、罚款类型、罚款日期、罚款时间、申诉、申判日期、判决、罚款数量和支付日期。法官和警察数据包括各自的名称和地址。每一个驾驶员也许有0个或多个罚单，而一张罚单只能用于一个驾驶员。警察可以开出多个罚单。

在警察向驾驶员开出罚单的同时，一张罚单的副本被上交并输入系统。与此同时，在数据库中生成了一张新的罚单记录并生成相应的驾驶员、警察和法庭之间的关系。如果驾驶员服罚，他或她在预先打印好的信封里装入罚单规定的罚款数目，然后邮寄给国家巡查部门。在有些情况下，驾驶员声称自己是无辜的要求法庭延期付款。如果信封寄回时没有支票，并且申诉请求框内写了一个"X"，那么系统在罚单记录上号下请求，寻找驾驶员、罚单和警察信息，然后向相应的法庭送一张罚单详细表，同时生成申诉日期调查表并寄给驾驶员。调查表上的说明告诉驾驶员填入方

便的日期并把调查表直接邮寄给法庭。一旦收到这些信息，法院就安排下一次审讯日期并向驾驶员通知日期和时间。

当审讯结束，法庭向罚单系统发送判决，然后在罚单上记录下判决和审讯日期。如果判决证明驾驶员是清白的，那么为保险公司生成驾驶记录报表的系统将删除罚单。如果判决证明驾驶员是有罪的，那么法庭给驾驶员另一个号码罚款数目的信封，以便驾驶员以后邮寄罚款。

如果驾驶员不能在要求的期限内支付罚款，罚款处理系统生成一张逮捕请求通知并把它寄给法庭。这通常发生在最初的信封在两周内没有收回时，或者法庭提供的信封在审讯日期后两周内没有收回时。此后的事情就由法庭决定。有时法庭要求吊销驾驶员的执照，然后处理驾驶员执照的系统负责处理吊销事务。

1. 罚单处理系统必须对什么事件做出响应？建立一张完整的事件表，表中列出事件、触发器、来源、用例、响应和每一事件的目的地。

2. 画一张表示罚款处理系统数据存储需求的实体-联系图，要求在图中包括所提到的属性。解释一下，为什么理解系统是如何和其他的国家巡查系统集成在一起非常重要？

3. 画出和ERD相应的类模型类图。但假设有不同类型的驾驶员。驾驶员类型的分类随着国家的变化而变化。例如，一些国家对未成年人的执照进行限制，并对商务车辆驾驶员发放特殊的执照。调查一下你们国家的需求，然后为驾驶员类建立一张概括具体层次图，在这张图中显示出每一种具体类型的驾驶员所具有的不同属性。同理，为罚单类型考虑一下同样的问题。在类图的概括/具体层次图中加入一些具体的罚单类型。

对落基山运动用品商店实例的再思考



在RMO系统中，当列出所有名词并对事物的初始列表做出一些决策时（见图5-18），RMO小组决定为成员系统包含一个RMO支付计划（与公司收费账目计划类似）。客户账户是否要作为一个数据实体或类，许多零售业的连锁店为了方便客户，都有它们自己的收费账目以增加销售和更好地对客户购买行为进行跟踪。

考虑一下如果管理人员决定将RMO收费账目和支付计划合并作为客户支持系统一部分的话，将对系统造成什么影响。

1. 讨论这样的变化对项目规模的影响。在收集信息和定义需求时，这些新功能将如何改变小组中的系统相关者列表？这些变化会对其他的RMO系统或计划中或进行着的系统项目造成影响吗？这些变化会对Barbara Halifax最初所开发的项目规划造成影响吗？换句话说，这些变化是次要的还是重大的？

2. 在事件表中需要添加什么事件？将这些附加的事件对事件表中的条目进行完善。由于收费账目和支付计划的合并，现有事件中哪些活动或后例需要改变？请解释一下。

3. 由于收费账目和支付计划的合并，系统中还需要存储哪些额外的事物及事物间的关系？对实体-联系图和类图进行修改以反映这些变化。

关注Reliable Pharmaceutical Services



在第1章已经介绍了关于Reliable公司的背景和处方处理的操作。如本章所讨论的一样，可以通过以下两个步骤开始定义新系统需求：第一，收集所需系统的信息；第二，把注意力集中在需要系统处理的事件和系统需要存储信息的事物上。整个系统包含了许多事件和事物。在本章的实例练习中，我们主要关心针对系统事件的一个子集及数据实体或类的一个子集。后续章节的练习将增加Reliable公司需求的范围和复杂性。

1. 创建事件表，根据下列特定的系统处理，列出系统需求信息：当卫生保健机构为病人开写

处方的时候，向Reliable提供订单详细信息，Reliable系统立即记录关于订单和处方的信息。处方订单在一天中从各个客户卫生保健机构提交上来。每隔12小时，Reliable向药剂师提供一个详细记录了最近订单的清单。药剂师安排好每个客户的订单后，就会记录指令完成（更详细的内容请参考第1章最后的Relinble案例描述）。此外，系统需要添加或更新患者信息，添加或更新药品库存信息，生成购买订单来补充药品库存，记录库存调整信息并生成各种管理报告。到现在为止，所有的情况都忽略了生成账单、付款和保险的过程。

2. 创建实体-联系图，用于针对下列系统部分显示数据存储需求：为每个数据实体添加一些属性，并给出最小和最大基数值。为了处理处方订单，Reliable系统需要知道关于患者、住院部和病房的信息。每个住院部可能有一个或者多个病房。一名患者被分派到一个特定的病房中。一个订单包括一个或多个处方，每个处方都是针对一种特定药物和特定的患者。因此一个订单可以包含多个处方，这些处方可以针对多于一个的患者。显然，仔细跟踪和保持记录是至关重要的。此外，每个患者可以有许多处方，一个药剂师对应一个订单。

3. 为面向对象的方法创建域模型类图，用于显示在第二步中描述的同一种需求。这包括了每个类的属性，以及最小和最大的重数。要确定各关联类并使用正确的符号。

4. 试讨论以下内容：理解每个订单包含对应多于一个患者的处方是不是很重要？是不是在最开始的时候很难挑选出信息的类型？你是否能看见最初的指示，或者是否只有在你能感觉到模型的时候才能创建这种模型。

参考资料

一些经典的和较新的资料如下：

事件与事件分析：

Stephen McMenamin and John Palmer, *Essential Systems Analysis*. Prentice Hall,1984.

Ed Yourdon, *Modern Structured Analysis*. Prentice Hall, 1989.

数据建模、实体-联系图以及数据库管理：

Peter Rob and Carlos Coronel, *Database Systems: Design, Implementation, and Management, Seventh Edition*. Course Technology, 2007.

对象、对象行为和类体：

Grady Booch, Ivar Jacobson, and James Rumbaugh, *The Unified Modeling Language User Guide*. Addison-Wesley, 1999.

Craig Larman, *Applying UML and Patterns* (3rd ed.). Prentice-Hall, 2005.

第6章 需求的传统描述方法

学习目标

阅读本章后，应具备如下能力：

- 解释用例建模时传统方法与面向对象方法的差异
- 列出传统系统的组成元素和在数据流图中描述这些元素的符号
- 说明数据流图如何在不同的抽象级别上描述系统
- 设计数据流图、数据元素定义、数据存储定义和处理描述
- 阅读并解释能被并入传统结构化分析方法的信息工程模型
- 设计一些表来说明处理和系统定位数据存取的分布性

本章要点

- 基于传统的观点和面向对象的观点看待活动/用例
- 数据流图
- 详细记录DFD部件
- 信息工程模型
- 节点和网络通信

圣地亚哥月刊：根据数据流分析系统

Arturo Romero和Lei Xu正在检查一些数据流图的草图，这些数据流图是为圣地亚哥月刊新的广告计费系统设计的。Arturo是定义新系统需求的分析员，Lei是主管广告账务的经理，她非常清楚当前系统是如何运转的。他们两个人以前讨论过几次，在最近一次讨论中（就是上星期）他们检查了当前系统事件、需求处理及过程参与者的详细情况。Arturo做了许多页的注释和大量的当前系统的示例表格和报表。自从上次讨论后，他已经几次打电话给Lei要求做进一步的讨论。

开始讨论时Arturo说道："我花了相当一段时间来消化你上星期给我的材料和信息，但我想我能理解并把我们讨论的全部系统活动或用例整理成文档。这次讨论的目的是再检查我写下来的需求处理以确保其完整性和准确性。所以让我们从创建的一些图表开始吧。"

Arturo在桌上摆出三张图表。Lei简略地看了一下，说道："我以前从没有见过这种图，它们就像一种弹子游戏的规划图。我觉得这些实体－联系图有些奇怪。"

Arturo回答说："我想这次讨论会稍微慢一点，因为你第一次看到这种类型的文档。我们一边看，我一边解释这些图吧。你可以随时提出问题。我们的工作质量取决于你对这些图的理解，因此不要有任何顾忌。"

Arturo继续说："这些图叫数据流图，或者简称为DFD。它们把你的系统分割成一系列处理功能。这些处理功能用圆角长方形表示。箭头表示处理功能之间或处理功能与文件之间的数据流向。"Lei指向图上的一个方形说："我猜这种方形代表购买广告空间的公司？"

Arturo回答："是的。这种方形代表一些人和组织机构，他们向系统提供输入或期望从系统中得到输出数据。"

Lei说："我想我懂了。我能认出大多数你为处理功能和数据所取的名字。我不太清楚其他这些符号代表什么。它们是用来命名我们存储在手工文件和数据库中的事物的吗？似乎

与我们的系统不能精确的对应。"

"是的，不对应，"Arturo回答："它们是我们在两个星期前创建的实体-联系图中的实体，但我们先暂时忽略这些。我们可以按照预订一个广告时的处理顺序进行。碰到这些实体时我们再讨论。"

Arturo和Lei继续讨论数据流图（DFD）及其相关事项，一个多小时过去了，Arturo的笔记本已经记满了好几页，并上数据流图上有25处用红笔做了校正或注释。Lei说："我想不出什么来了，今天就这样吧。"

Arturo回答："你已经给了我大量的事去做，我们就此打住吧。你看我们是否可以在星期四上午9:00再花两个小时讨论一下？"

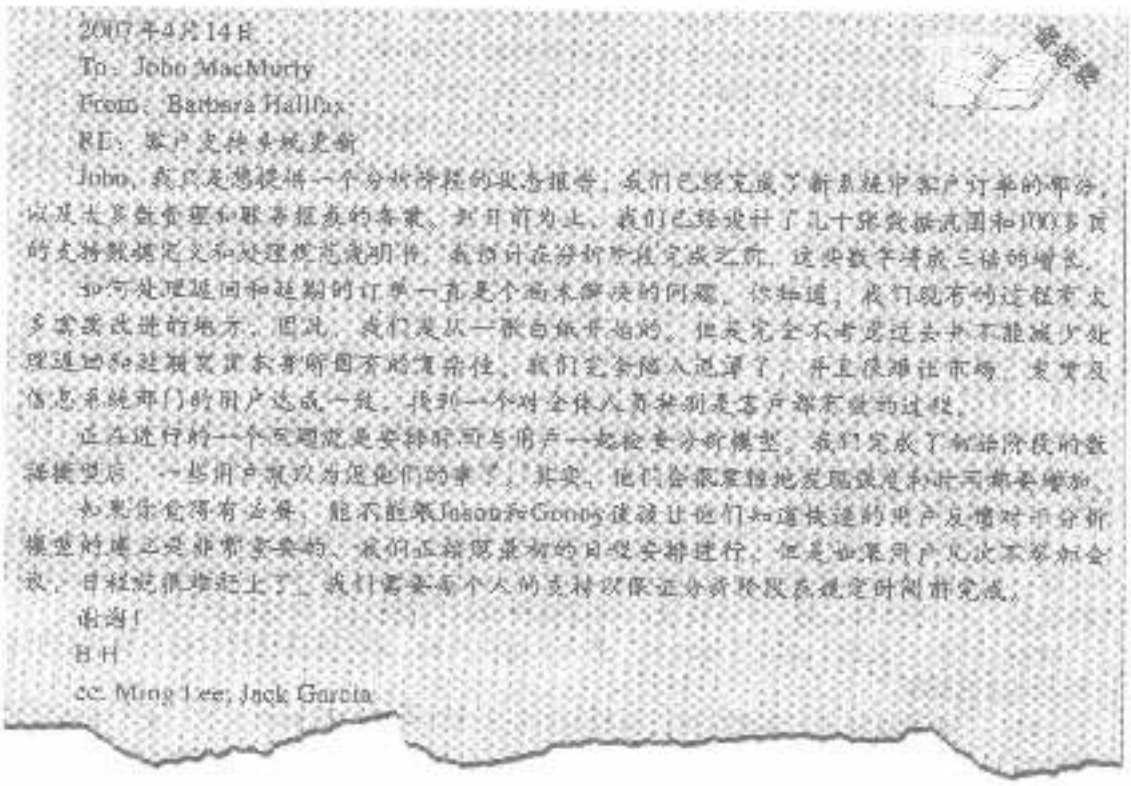
Lei说："可以，我那时应该有空。你是否可以多带一些数据流图，或许你又会展示一些更奇妙的东西？"

Arturo笑道："最艰难的事情还在后面呢。但我肯定会有更惊奇的东西给你看。"

概述

第5章描述了在使用传统方法和面向对象方法进行系统开发的过程中与建立系统需求模型相关的两个关键概念：（触发用例的）事件和（用户工作环境中的）事物。而在这一章，我们的重点将转向当事件发生时系统具体做什么。即活动和计算机程序与数据间的交互。

本章通过描述传统的结构化方法来表示活动和交互。我们将描述和介绍在传统方法中使用的图形和其他模型，同时提供洛基山运动用品商店（RMO）的客户支持系统模型的例子，以展示在传统的方法中这些模型是如何联系在一起的。此外，我们简要地讨论如何把传统方法、信息工程方法及它们相应的模型组合在一起来描述一个系统。第7章将介绍用面向对象的方法描述活动和交互的具体细节。



2007年4月14日

To：Jobo MacMurty

From：Barbara Halifax

RE：客户支持系统更新

Jobo，我只是想提供一个分析阶段的状态报告。我们已经完成了新系统中客户订单的部分，以及未来数据库和服务报表的备案。到目前为止，我们已经设计了九十张数据流图和1000多页的支持数据定义和处理规范说明书。我们计在分析阶段完成之前，这些数字将成三倍的增长。

如何处理退回和延期的订单一直是个很难解决的问题。你知道，我们现有的过程有太多需要改进的地方。因此，我们是从一张白纸开始的。但是完全不考虑过去也不能减少处理退回和延期发货事务所面对的复杂性。我们已经深入进去了，并且很难让市场、发货及信息系统部门的用户达成一致。找到一个对全体人员都满意的过程。

正在进行的一个问题就是要获取与用户一起检查分析模型。我们完成了初始阶段的数据模型后，一些用户就认为这是他们的事了。其实，他们会需要随时发现他们的需求有变化或增加。

如果你觉得有必要，能不能和Jobo和Grono谈谈让他们知道我们的用户反馈对分析模型的成功是非常重要的。我们正按照原来的日程安排进行，但是如果用户无法不参加会议，日程就很难赶上了。我们需要每个人的支持以保证分析阶段在规定时间前完成。

谢谢！

BH

cc: Ming Lee; Jack Garcia

不论用传统的方法还是面向对象的方法对活动和交互进行建模都是一个困难的过程。建模是一项困难而又耗时的任务，必须严格地确定活动和交互的细节。分析员和用户必须共同评估模型的完整性、正确性和质量。就像RMO进展备忘录中所描述的那样，协调项目参与者的工作，使关于详细系统需求的意见达成一致是一系列复杂的项目管理活动。

6.1 用传统的观点和面向对象的观点看待活动/用例

系统开发的传统方法和面向对象方法的区别在于当一个事件发生时系统如何响应。传统方法把系统看做一个处理的集合体，一些由人完成，另一些由计算机完成。计算机处理就像常规的计算机程序一样——由按顺序执行的指令组成。当执行处理时，它与数据进行交互，读出数值，又把其他的数值写回到数据文件中。这一过程可能也要与人进行交互，比如当一个指令要求用户输入一个值或者在计算机屏幕上显示信息给用户看。所以，系统的传统方法包括处理、数据、输入和输出。在为系统对事件做出的反应进行建模的过程中，传统方法包括重点强调这些组件的一系列处理模型。

相比之下，面向对象（OO）方法把系统看成是一个相互影响的对象的集合。这些对象就是在第5章中讨论过的问题域中的事物。对象是有行为的（可做方法）。这些方法可以使对象与其他对象或系统使用者进行交互。一个对象通过发送消息请求另一个对象做某事。就其本身而论，面向对象方法不存在常规的计算机过程和数据文件。对象执行活动并记录下数值。当为系统响应事件建模的时候，面向对象方法包括显示对象、对象的行为及对象之间交互的一系列模型。

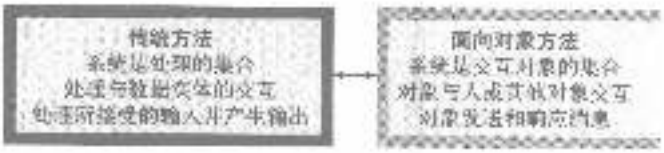


图6-1 传统方法和面向对象方法

图6-1总结了传统方法和面向对象这两种方法的不同。因为这些不同，所以传统方法和面向对象方法需要不同的模型，如同图6-2中总结的一样。本章的其余部分将探究图6-2左侧的传统方法。

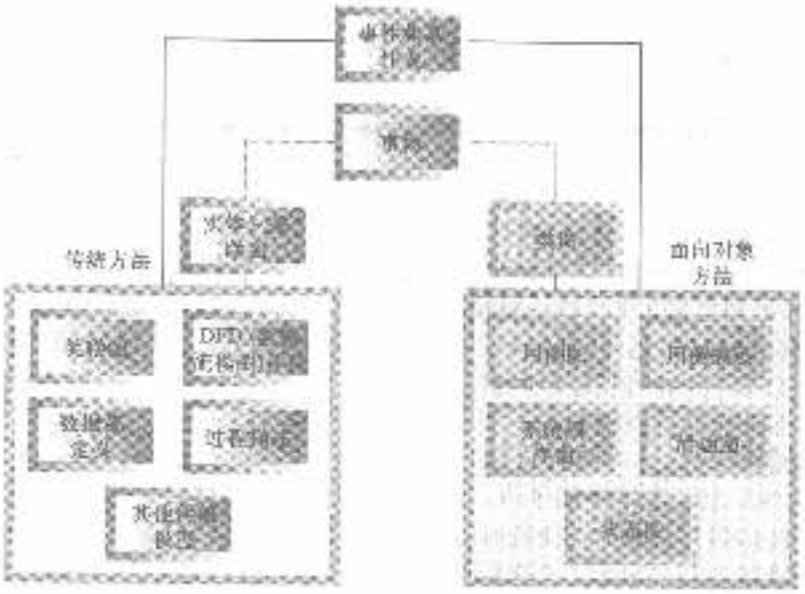


图6-2 传统方法和面向对象方法的需求模型

6.2 数据流图

在信息系统开发中，传统方法把活动描述为由人或计算机执行的过程。数据流图已被证明是独立过程模型中非常有价值的图形化模型，当然，还有其他的过程模型如在信息工程方法中使用的过程依赖图和用于业务流程重组的工作流图，但数据流图是用得最广泛的过程模型。

数据流图（DFD）是一种图形化的系统模型，它在一幅图中展示信息系统的主要需求：输入、输出、处理和数据存储。任何从事项目开发的人都能从DFD中很快看出系统一起工作的各个部分，这也是数据流图普及的一个原因。DFD很容易理解是因为它是图形化模型，而且它只有5个符号（见图6-3）需要学习。最终用户、管理人员和所有信息系统工作人员只需稍加培训即可读懂和理解DFD。

数据流图（DFD）：用处理、外部实体、数据流及数据存储来表示系统需求的图表。

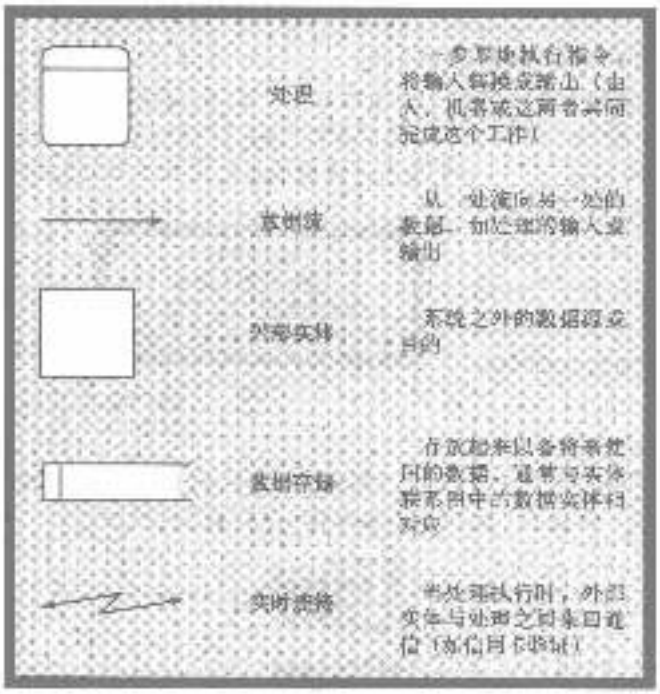


图6-3 数据流图的符号

图6-4是数据流图的一个例子，表示RMO的客户支持系统的一部分。正方形表示外部实体，即客户，它是在系统外数据的来源和目的。圆角矩形是名为查询可用条目的处理，它可用数字1标号。一个处理定义了转换输入到输出的规则。带箭头的线是数据流。图6-4显示了客户和处理1之间有两条数据流：一条叫做“条目查询”的输入处理和一条叫做“可用条目明细”的输出处理。最后一个符号三边矩形表示数据存储。每一个数据存储代表一个文件或数据库中的一部分，它用来存储一个数据实体的信息。在这个例子中，数据流（带箭头的线）从数据存储指向处理，表示处理从名为“目录”、“产品条目”、“库存条目”的三个数据存储中

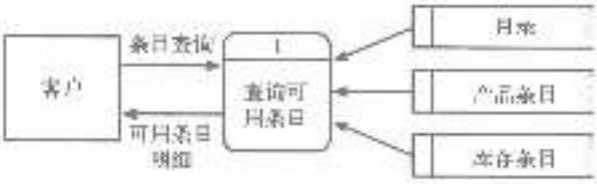


图6-4 显示处理查询可用条目的DFD（RMO案例中的DFD片段）

查询信息。

外部实体：在系统边界之外的个人或组织，它提供数据输入或接受数据输出。

处理：在DFD中的一个符号，它代表从数据输入转换到数据输出的算法或程序。

数据流：在DFD中的箭头，它表示在处理、数据存储和外部实体之间的数据移动。

数据存储：保存数据的地方，以使将来由一个或多个处理来访问这些数据。

你也许注意到了图6-4中的处理对应于第5章中的RMO事件表中的一个活动（参见图5-16）。这个事件是客户想查询的可用条目，触发器是条目查询，来源是客户，响应是可用条目细节，响应的目标是客户。所以，这个数据流图以图形的方式显示系统用例来响应一个事件。

但是DFD的另一个信息没有包含在事件表中：数据存储包括条目可用性的信息。图6-4中的每一个数据存储在实体－联系图（ERD）中代表一个数据实体，如第5章所述（参见图5-29）。在DFD中的处理使用了在系统的ERD中所提供的数据实体及其属性信息。所以，数据流图将被事件触发的处理和在ERD中定义的数据实体相结合。图6-5总结了DFD的组成部分、在事件表中描述的事件及在ERD中定义的数据实体这三者的一致性。

实践指导

当使用传统方法时，确定用例然后用数据流图片段为每一个用例细节建模。 ■

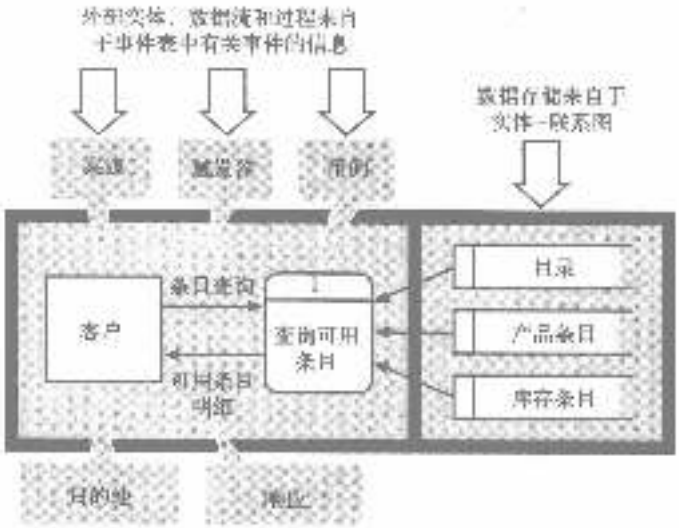


图6-5 结合了事件表和ERD的DFD

6.2.1 数据流图和抽象水平

有许多种类型的数据流图用于描述系统需求。刚才描述的例子是DFD的一部分，它显示了响应一个事件的过程。其他的数据流图用于显示一个更高层（系统更概括的概念）或更低层（系统更详细的概念）的处理。这些不同的系统概念（高层的和低层的）被认为是抽象水平。

抽象水平：把系统分解成一个逐渐细化的分层集合的建模技术。

数据流图的另一个非常有用的特性是能够表现系统高层和低层概念。在一个DFD中高层次处理可以分解成若干独立、低层次、详细的DFD。详细的DFD中的处理可以进一步分解成其他的图形，以便提供多水平的抽象。

图6-6所示为每一水平上的数据流图如何提供关于下一个较高水平上的处理的附加信息。

最高层的DFD将课程注册系统作为一个单个的处理，并给出了最抽象的表示。中间的DFD给出了关联图的内部细节，底层的DFD给出了中间的DFD中处理1的内部细节。下面将对每个DFD抽象水平有进一步的描述。

1. 关联图

关联图是指描述系统抽象概念的DFD，所有的外部实体和进出系统的数据流都在一张图中显示，并且整个系统被表示成一个处理。图6-6中的最高层的DFD是一个简单的大学课程注册系统的关联图，这个图与三个外部实体交互：学术部、学生和教员。学术部提供有关课程的信息，学生申请注册课程，教员在注册阶段完成后得到班级列表。

关联图：单个处理符号中解释系统为所有处理活动的DFD。

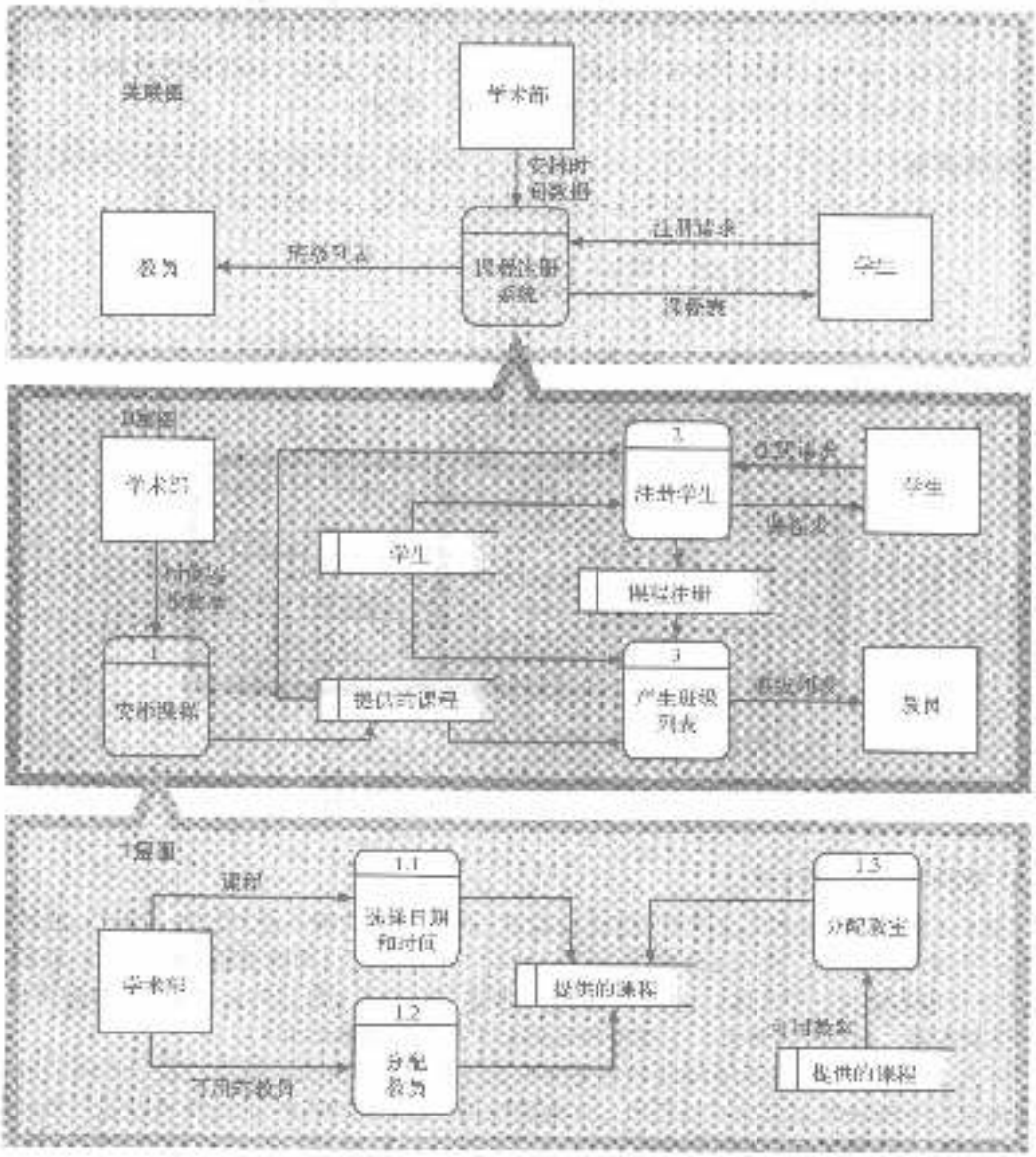


图6-6　课程注册系统的DFD抽象层次

关联图在表达系统边界时很有用。系统的范围是通过单个的处理和外部实体所表示的事物来定义的。提供和接收数据的外部实体在系统范围以外，其他任何事情都属于系统的范围。数据存储不画在关联图中是因为数据存储本身被认为是属于系统内部的（就是说，它们是表示系统处理的内部实现的一部分）。然而，当建模的系统和另一个系统对其共享时，数据存储可以被显示。

关联图通常与第5章描述的时间表一起被建立。每个对应一个外部事件的触发器变成一个输出数据流，并且它的来源变成一个外部代理。每一个响应变成一个输出数据流，其目的地变成一个外部代理。对应临时事件的触发器不是数据流，所以没有对应临时事件的输入数据流。注意，关联图DFD能够直接从事件表创建。两种模型从不同角度描述了同一种系统需求信息。

2. DFD片段

一个DFD片段是为由事件表中的事件触发的每个用例创建的。每个DFD片段是一个显示系统如何响应某个事件的独立模型。分析员通常是一次创建一个DFD片段，将精力集中在系统的每一个部分中。在事件表和关联图完成之后，DFD片段才被画出来。

DFD片段：用一个单一处理符号表示系统响应一个事件的DFD。

图6-7所示为简单课程注册系统中的三个DFD片段。每一个DFD片段用一个单独的处理符号代表对一个由事件触发的用例的所有响应处理。但是这些片段展示了在处理、外部实体和内部数据存储之间的交互细节。在DFD片段中的数据存储代表ERD中的实体。每个DFD片段仅显示要响应该事件的那些数据存储。

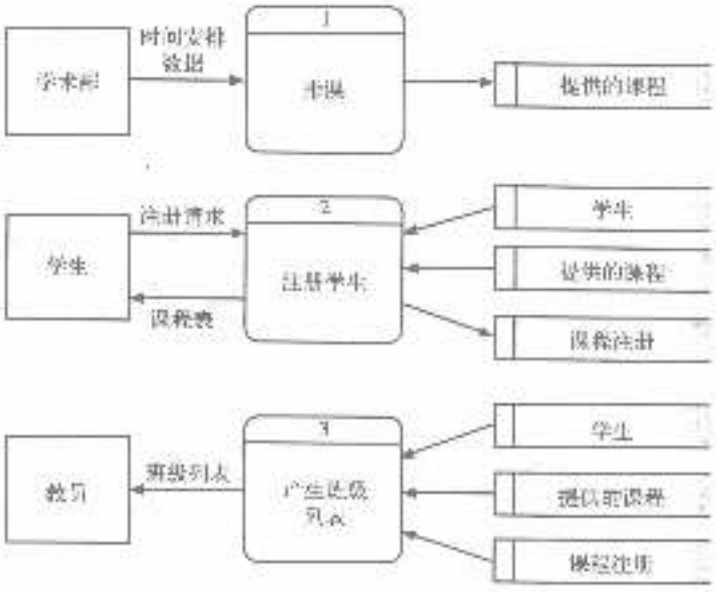


图6-7 课程注册系统的DFD片段

3. 事件分割的系统模型

系统或子系统的所有DFD片段可以组合到一个单个的DFD中，这个DFD就称为事件分割的系统模型或0层图。图6-8所示为图6-7中的三个课程注册系统的DFD片段如何组合成0层图的处理。

事件分割的系统模型或0层图：一个为系统需求建立模型的DFD，建模过程中对应于系统

对于系统中每个事件使用单个处理。

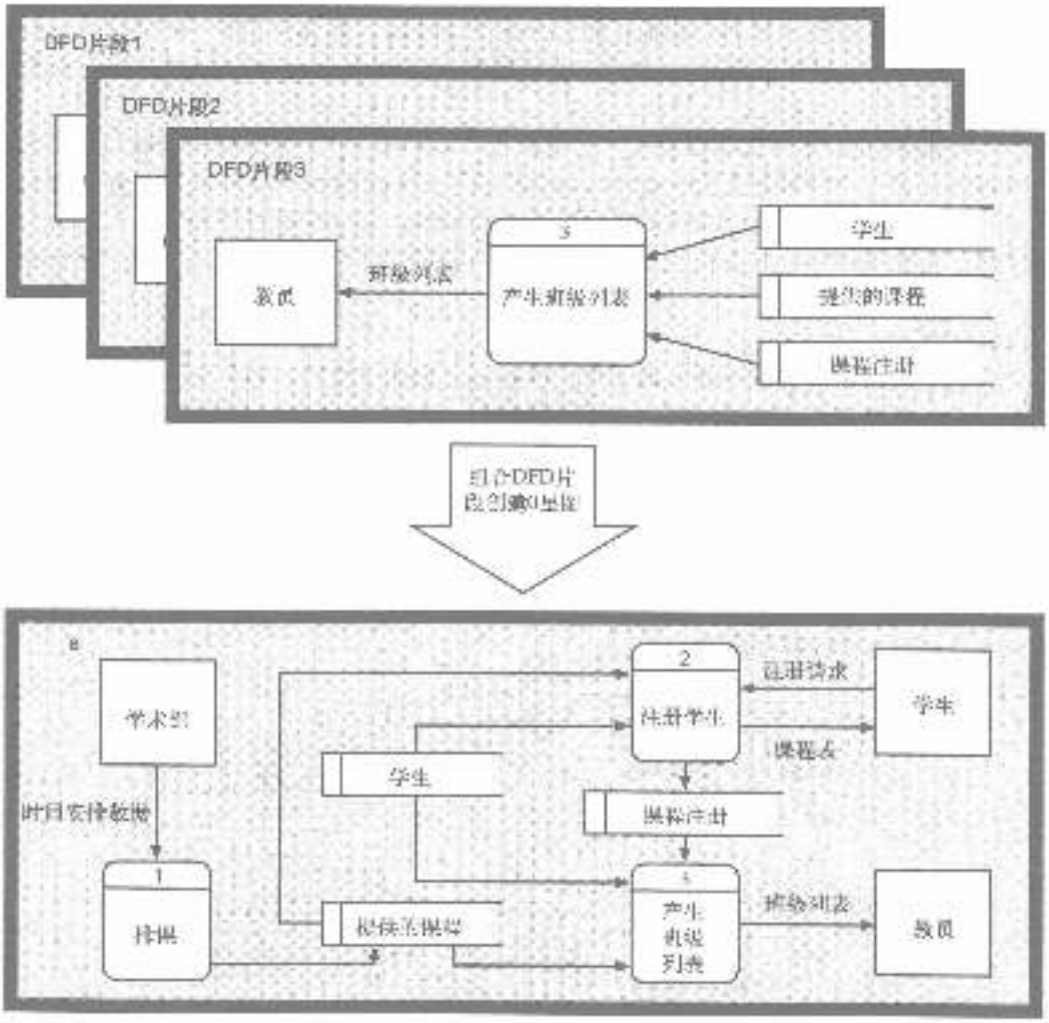


图6-8　课程注册系统中将DFD片段组合以创建事件分离的系统模型

0层图主要用来作为一个表示工具。它对整个系统或子系统进行比关联图更加详细的汇总。但是，分析员常常避免设计0层图，有如下两点原因：

- 信息内容与DFD片段的集合重复。
- 对表常常复杂而不实用，特别是对于需要响应许多事件的大系统而言。

如同本章的后面将介绍的，冗余性和复杂性是分析员无论何时都要尽可能避免的两个DFD特征。

6.2.2　RMO数据流图

图6-9所示为RMO客户支持系统的关联图。通常，就像前面所讨论的那样，关联图上的数据流和外部实体都是从事件表直接获取的。但是由于RMO客户支持系统要响应20个事件，该图中为了简化，将一些事件的数据流图组合在一起。在一个只有10－15个事件的较小系统的例子中，应该包括关联图中的所有数据流。

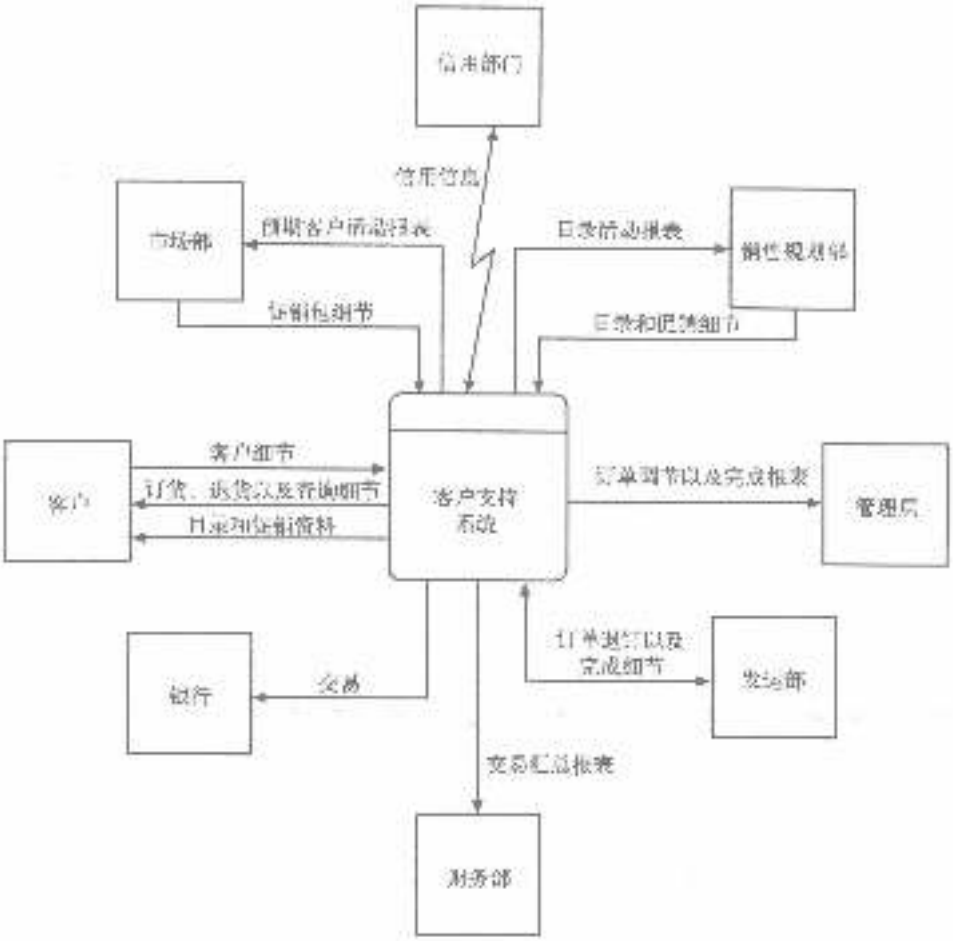


图6-9 RMO客户支持系统的关联图

当一个系统响应许多事件时，常常将系统分成多个子系统，并且为每个子系统创建一个关联图。图6-10按照用例的相似性，包括与外部实体和数据存储的交互及必要处理的相似性，将RMO客户支持系统分成子系统，图6-11所示为订单子系统的关联图。注意，来自于这个子系统事件表的所有数据流都显示在DFD上。



图6-10 RMO子系统和对应每个子系统的用例



图6-11　RMO订单子系统的关联图

图6-12给出了RMO订单子系统的DFD片段。注意这里有五个DFD片段，分别对应图6-11列出的每一个订单子系统用例。

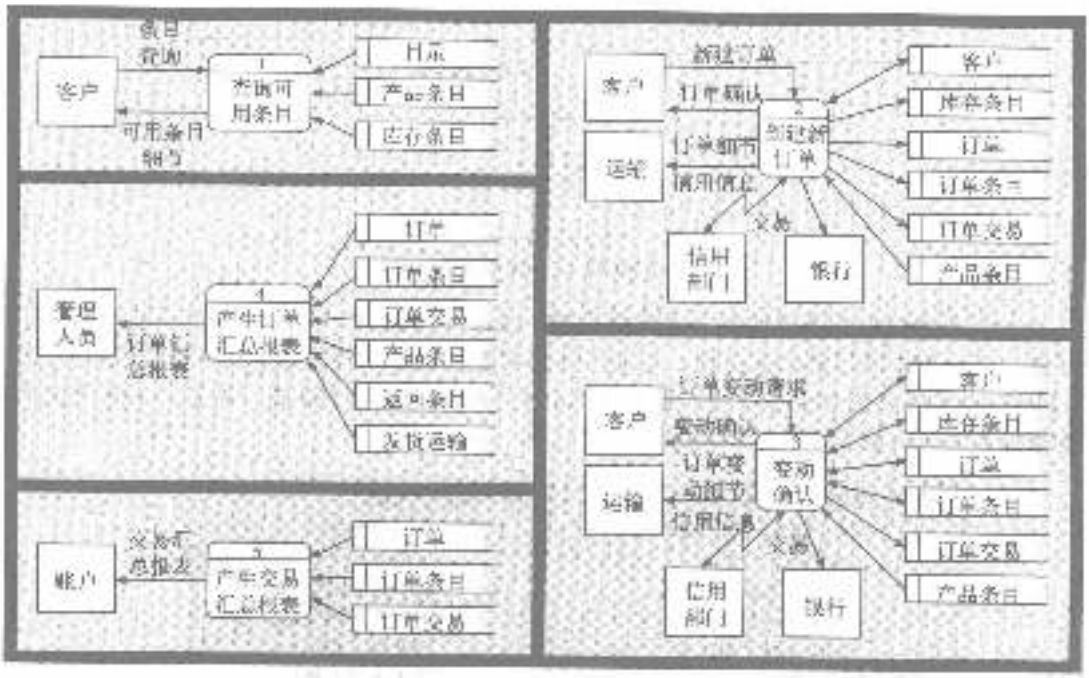


图6-12　RMO订单子系统的DFD片段

同样，图6-13显示了RMO订单子系统的0层图。这是从图6-12中DFD片段得来的。为了使图更加简单和提高图的可读性，在图6-12中的7个数据存储合并成图6-13中的单个数据存储。回想一下，0层图仅用作辅助性说明。DFD片段显示了哪些过程与独立的数据存储相关联。

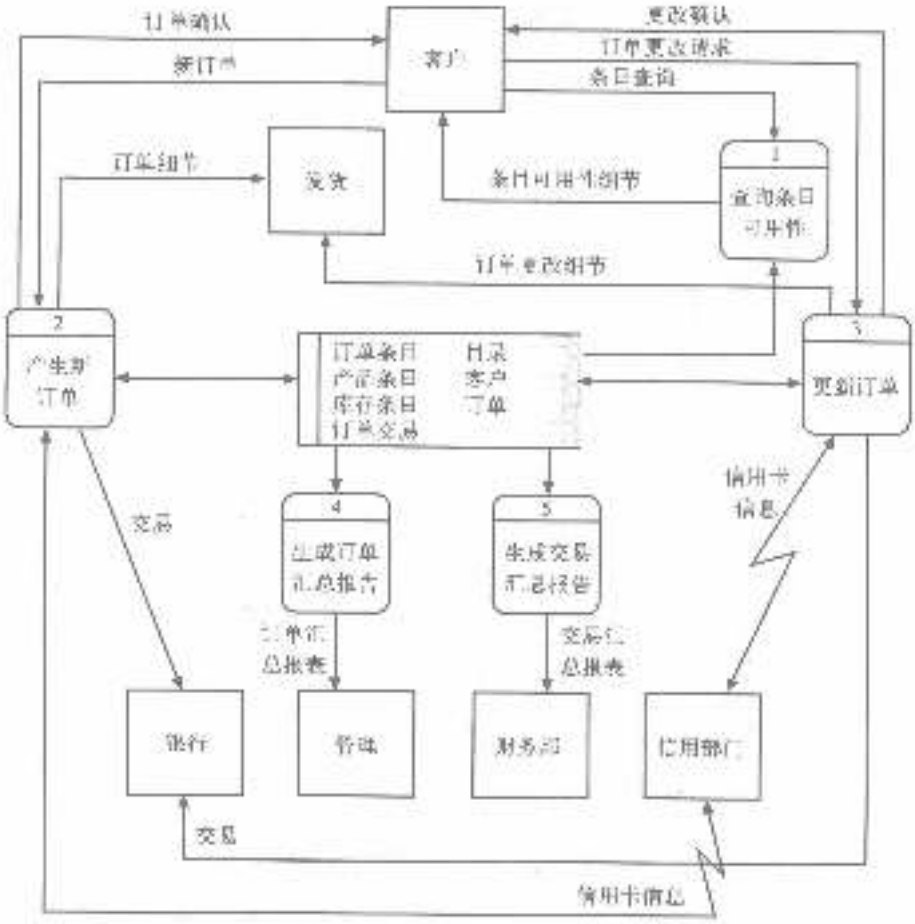


图6-13 订单子系统的事件分割模型（0层图）

分解处理以查看活动的细节

有时一些DFD片段包括许多处理，这些处理需要系统分析员做更详细的研究。在任何建模步骤中，进一步的分解都将帮助系统分析员了解更多的需求，同时也可以产生必要的文档。图6-14所示为RMO的DFD片段2创建新订单较详细的分解图的一个例子。将它命名为图表2是由于它显示处理2的内部信息。子处理被编号为2.1、2.2、2.3和2.4，但系统中的编号方式不一定总表示子处理的执行顺序。

这个例把处理2分解为4个子处理过程：记录客户信息、记录订单、处理订单交易和产生确认信息。这些子处理过程被认为是完整活动的4个主要步骤。这种分解也是细化工作的一种办法。另一个分析员可能得到不同的分析结果。

第一步开始于客户提供组成“新订单”这个数据流的信息。新订单数据流包含客户和客户想要订购条目的所有信息。如果是新客户，处理过程2.1将客户信息存储到一个叫做“客户”的数据存储中（创建新的客户信息或根据要求更新已有的客户信息）。请记住，数据存储代表了在第5章中讲到的ERD中的客户数据实体（见图5-29），处理2.1就把有关订单的其他信息——名为“订单细节”的数据流发送给处理过程2.2。

处理过程2.2使用“订单细节”数据流，并在“订单”数据存储中通过加入数据创建一个新的订单记录。然后，对应于每一个订单条目，在“产品项目”、“库存项目”的数据存储中

查询库存和产品价格。如果当前有足够的库存，就创建一个订单条目记录，同时修改库存数据。比如，如果有三个条目被预订，则需要创建一个订单记录和三个订单条目记录。

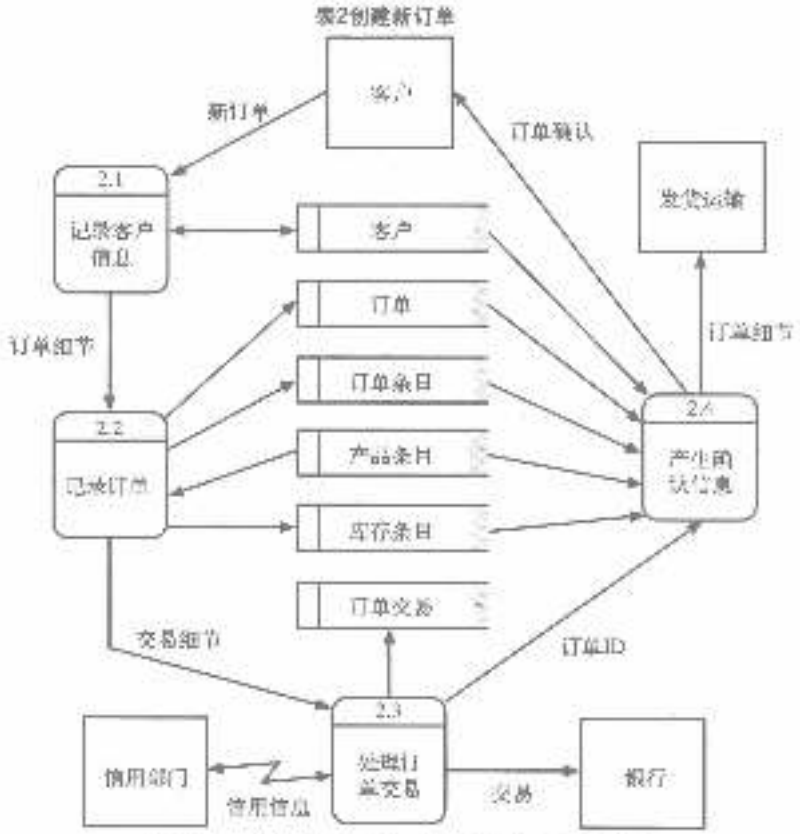


图6-14　创建新订单的一个更详细的图（图表2）

处理过程2.2根据订单（每一个条目的总数等于单价乘以数量）累计总数，同时发送称为"交易细节"的数据流到处理过程2.3以记录交易。"交易细节"包含订单号、数量和信用卡信息。处理2.3需要获得一个与信用部门保持实时连接以取得客户的信用卡的信用权限。这里用实时连接而不是数据流是因为在处理过程执行时，数据需要很快地来回流动。如果信用卡是可用的，就在数据存储"订单交易"中创建一条交易记录，同时一条交易数据流直接传到银行。

最后一个处理产生客户的订单确认信息和发给运输部门的订单细节。根据订单号，处理2.4可以在订单、客户、每一个订单条目（加上产品条目中的条目描述）中查询数据并产生必要的输出。

6.2.3　物理DFD和逻辑DFD

DFD可以是一个物理系统模型，也可以是逻辑系统模型，也可以是两者的混合。如果DFD是逻辑模型，则假设可以用完美的技术（如第4章所述）实现这个系统。如果DFD是物理模型，则在DFD中应包含一个或多个假设的实现技术。这些假设可能有很多的形式，并且可能很难找到。

考虑在图6-14中的图表2是否是一个逻辑模型。首先，什么类型的计算机可以做这些处理，

这一点是否清晰？它可以是一个桌面系统吗？亦或是一个主机系统？还是一个网络客户—服务器系统？或者刚才描述的所有处理是否可以在根本没有计算机的情况下手工完成？类似地，数据存储是否是连续的计算机文件？亦或是在关系数据库中的表？还是在文件柜中的文件？系统如何从客户那里得到数据流“新订单”，以便能够被处理？是在一个Windows应用程序中单击检查框或列表框？还是在一个网页页面上？或由打字员以打字方式手工地填写表格？或通过电话与一名职员通话？或在电话中与一个声音识别程序对讲？

所有已描述的选择都是可能的。并且如果这个模型是逻辑模型，你应该无法说出系统如何实现。同时，要处理的需求（必须是此在进行的）应该还算详细，直到指出必需的属性值。模型甚至应该更加详细，并且仍旧是一个逻辑模型。

现在将图6-15与图6-6中的图表1进行对比，考虑图6-15中的DFD是否是一个物理模型。许多元素表明了有关实现技术的设想，其中包括：

- 特定的技术处理
- 特定的参与者的处理名称
- 特定的技术或参与者处理顺序
- 冗余的处理、数据流、文件

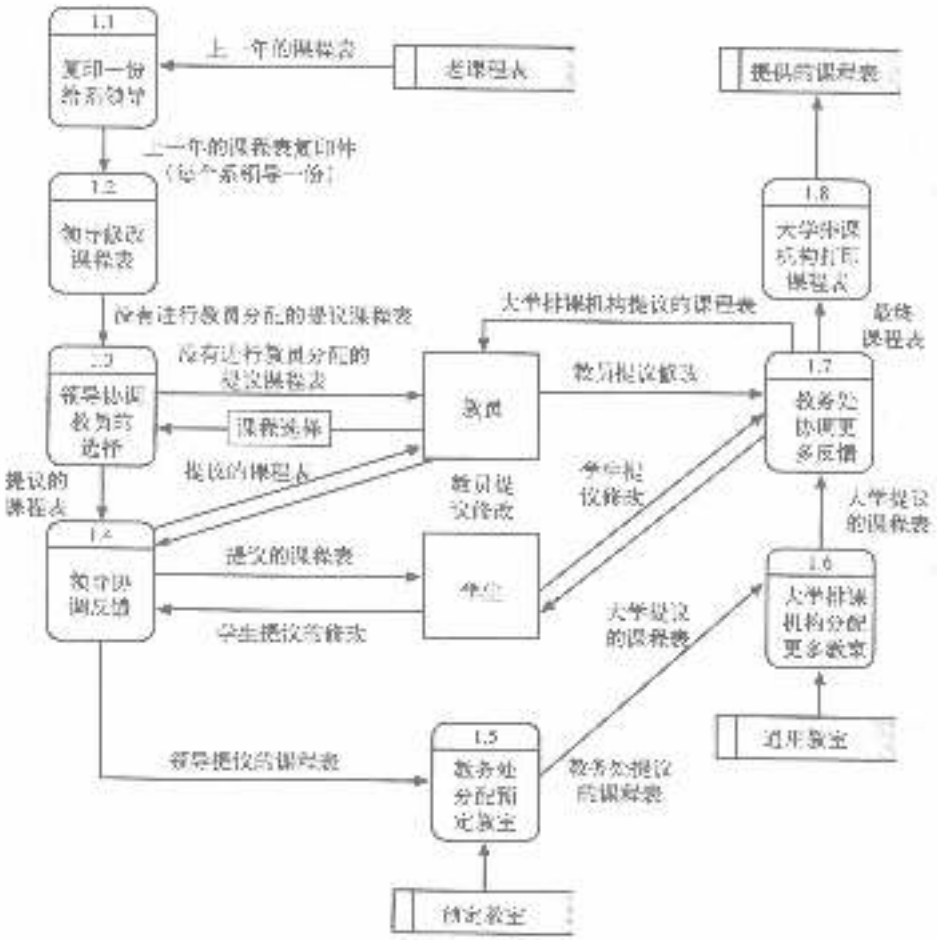


图6-15 安排课程的物理DFD（避免在分析时创建物理DFD）

最显而易见的技术假设包含在处理1.1。复印本质上是一个手工作业，它暗示数据存储"老的课程表"和流进、流出处理1.1的数据流都是纸文件。这些数据存储和数据流是可以电子化的，但如果这样，那么"为什么需要一个处理来做电子副本"这样的问题就会出现。

许多处理的名称特别参考了系统中的参与者，如参考了"领导"、"教务处"和"大学排课机构"。所有这些都表明处理是由特定的个人或部门来执行的。处理中的数据顺序流是个人或部门执行每个处理的副产品。可以想象另外的实现方式，这些方式使用更少的处理，不同的处理执行顺序或不同的个人和部门的处理分配。DFD清晰地建立了一个关于处理执行顺序和职责的非常具体的决策集合的模型。

这个DFD也包含相似的和多余处理逻辑的过程。例如，开始接受教员输入，但后面仍要执行两次错误检查（处理1.4和1.7的数据流）。另外，教室也从两个不同的数据存储（在处理1.5的"预订教室"和处理1.6的"通用教室"）中进行过两次不同的分配，如前所述，这些特性表明关于技术和职责分工的特别设想。冗余错误检查表明可能在前一个处理中出错。一个用理想技术实现的系统不需要内部的错误检查。教室分配分别在两个文件和处理之间进行，可能与技术相关（如没有一个处理可以一次成功地分配所有的教室），或它表明了教室分配责任的历史划分。

没有经验的系统分析员经常会开发出如图6-15所示的DFD。开发这种模型的方法是非常简单的：就是按照当前系统中事物本来的方式建模。这种方法的问题在于会把旧系统的设计假想和技术限制不自觉地包含到新系统中来。如果分析和设计不是同一些人和小组，这种情况很容易发生。这些新的设计人员可能没有认识到有些包含在DFD中的需求仅仅是当前系统的事物存在形式的一些简单反映，而不是系统中事物将来所必然存在的形式。

通常在分析的最后阶段和设计的最早阶段开发和使用物理DFD。在开发较详细的设计模型之前，对于描述可选系统实现方法，它们是非常有用的模型。但是应该避免在分析阶段的所有活动中出现物理DFD，除非在生成选择方案阶段。即使在生成选择方案阶段，分析员也应该谨慎地表示物理DFD，以便让读者知道这个模型表示一个原逻辑系统需求的可能实现。

6.2.4　评估DFD质量

高质量的DFD是可读的、内部一致的及能准确描述系统需求的。描述的准确性主要取决于是否与用户做了充分的交流并查阅了足够多的资料。通过在DFD结构上应用一些简单的规则，一个项目小组可以保证DFD的可读性和内部一致性。系统分析员应该在开发DFD时或准备好草图后的质量检查过程中使用这些规则。

1. 最小化复杂度

人们在对复杂的信息处理上是有局限性的，当太多的信息同时出现时，人们把这种现象叫做**信息超量**。当信息超量发生时，一个人很难理解呈现在面前的信息。避免信息超量的关键是把信息划分为小的且相对独立的子集，每一个子集有一定数量的可逐个考察和理解的信息。

信息超量：当太多的信息同时呈现给一个人时所发生的一种难以理解的情况。

DFD分层结构是把信息划分为小的且相对独立的子集的一个例子，这样可以逐个考察每一个DFD。读者要了解某个处理更加详细的信息可以跳转到该处理的下一层。如果要知道一个DFD如何与其他DFD相关联可以跳转到上一层的DFD去考察。

分析员要在任何一个DFD中避免信息超量可以遵循以下两条DFD构造规则：

- 7±2
- 接口最小化

7±2规则（也称为Miller数）来源于心理学研究。心理学研究表明一个人可以同时记住或

操纵的信息“块”的数量介于5～9之间。信息块的数量太大就要引起信息超量。信息块可以是包括名称、一个列表中的单词、数字或一个图片中的元素在内的许多事物。

7±2规则：一种限制模型中组成元素个数或元素之间的连接数不超过9的模型设计规则。

7±2规则在DFD中的应用如下：

- 单个DFD中不应有超过7±2个处理；
- 单个DFD不应超过7±2个数据流进出一个处理、数据存储和数据元素。

这些规则是一些一般的准则，不是不可违反的。打破这些规则的DFD可能仍是可读的，但这种违规现象应看做是潜在问题发生的一个警告。

接口最小化是与7±2规则有关的。接口是指一个问题或描述中的一部分与其他部分的连接。与信息块一样，一个人可以同时记住或操纵的连接是有限的。所以应保证连接数最小。DFD中的处理表示业务和处理逻辑块，它们通过数据流与其他处理、实体和数据存储相关联。带有大量接口（数据流）的单个处理会复杂到不能理解。这种复杂性也许会作为7±2规则的违反行为直接在处理分解中显现出来。分析员通常可以这样来解决这个问题，即把这种处理分解为两个或更多的处理，以使分解后的处理接口更少。

接口最小化：一种通过限制模型中各个元素之间的连接数来达到简单的目的的模型设计原则。

处理或对或组且在它们之间有大量的数据流，是与接口最小化规则相冲突的另一个例子。通常这样的条件预示着处理中的任务处理划分都比较差。解决这个问题的办法是重新分配处理任务使在它们之间需要更少的接口。在处理之间对工作的最好划分是保证最简单。而保证最简单的划分则要求处理之间使用最少的接口。

2. 保证数据流一致性

分析员通过查找DFD中各种类型的不一致性可以发现错误或忽略的东西。以下是三个经常发生且容易判别的一致性错误：

- 一个处理和它的处理分解在数据流内容中有差别
- 有数据流出却没有相应的数据流入
- 有数据流入却没有相应的数据流出

处理分解以一种更详细的形式展示了一个高层处理的内部细节。在大多数情况下，DFD层次中流入和流出处理的数据内容应与分解后的DFD中流入和流出所有处理的数据内容一致，这种一致性叫做平衡，且高层DFD与处理分解DFD称为“在平衡中”。

平衡：进出处理的数据流与进出处理分解DFD的数据流在数据内容上保持一致的状态。

注意在上一段中数据内容这个术语的使用。数据流的名称在不同的层次可能不一样，其原因很多，如将一个组合的数据流分解为更小的数据流。所以，分析员必须仔细看清楚数据流的内容而不能只看到它的名称。由于这个原因，只有在所有的数据流均已定义后方可进行平衡的详细分析。

如果由于在高层忽略了一些数据流而引起数据流图不平衡，则不平衡的DFD也是可接受的。例如，一个大系统的0层图通常省略错误处理的细节，假设当订购了一个商品，但库存量不够或该商品不再继续生产，在0层图有一个处理叫做完成订单，在这种情况下就没有任何相关的数据流了，而在处理完成订单的分解图中系统分析员可以加上一个处理和一些数据流去处理这些不能继续的条目。

DFD不一致性还可以发生在单个处理或数据存储的数据流入或流出之间。根据定义，处理将输入数据转换成输出数据，在一个逻辑DFD中，数据不应该没有意义地传给处理。以下的一致性规则来源于这些事实：

• 流入处理的所有数据必须流出该处理或用于产生流出该处理的数据；

• 流出处理的所有数据必须曾流入过该处理或由流入该处理的数据产生。

图6-16所示为违反上面第1条规则的例子。比较图6-16和图6-12中的第一个DFD片段，并注意它们各自流入处理的数据流的区别。查找条目可用性只需要确定条目和对相应数据存储的访问信息。在图6-16中，额外的数据输入（一个完整的订单）流入处理，而处理更多的是访问数据存储而不是产生作为数据输出流的可用条目细节。如图6-16所示的处理有时称为黑洞，因为输入的全部或部分数据没有输出。

黑洞：带有输入数据的且不用其来产生输出数据的处理或数据存储。

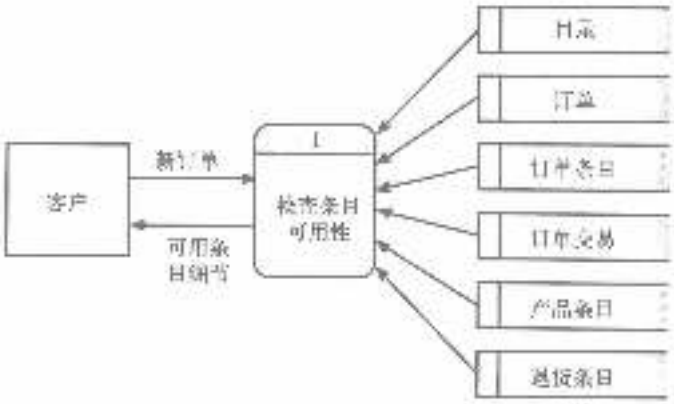


图6-16　带有不必要数据输入的处理——一个黑洞

图6-17所示为违反上面第2条规则的示例。比较图6-17与图6-7中的底层DFD片段，并注意它们各自流入处理的数据流的区别。在图6-17中，不足的数据输入到处理以产生数据输出。来自课程和课程注册的必要的数据输入丢掉了。如图6-17中所示的处理有时称为奇迹，因为数据没有任何可见的来源奇迹般地从处理中出现了。

奇迹：带有没有任何产生来源数据元素的一个处理或数据存储。



图6-17　有不可能的数据输出的处理——一个奇迹

分析员有时候可以通过检查DFD很简单地找到黑洞和奇迹。在其他情况下，对数据字典和处理描述进行仔细的检查是必要的。在图6-18中，数据元素A、B和C流入了处理但没有流出。数据元素A用来决定用什么样的公式计算X的值，所以它是必须输入的，而B和C则在处理的输出上没有起作用，因此它们应该作为不必要的输入流而被排除在输入元素之外。

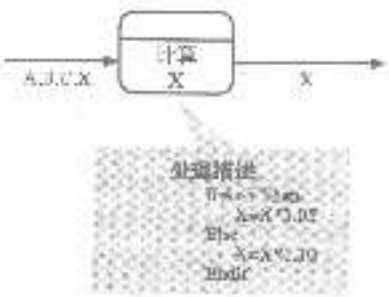


图6-18　带有不必要数据输入的处理

在图6-19中，数据元素A、B、Y从该处理流出。数据元素A流入该处理，数据元素Y的值是根据数据元素A

的值按照一种算法计算出来的，但数据元素B没有流入该处理且没有通过内部处理逻辑计算出来。所以这就表明：或者数据元素B是输出数据流的一个错误（B应该排除），或者在内部处理逻辑中忽略了该元素（规则使B丢失了）。

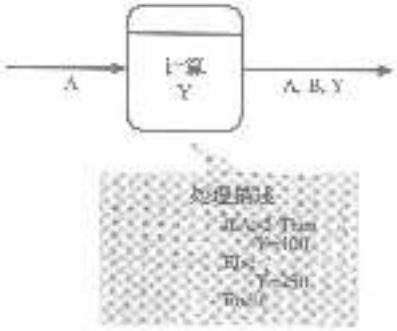


图6-19 有不可能的数据输出的处理

注意，上述两个一致性规则不仅仅用于处理，对数据存储也有效。任何从数据存储读出来的数据元素必定在以前写进去过。类似地，任何写进数据存储的数据元素必定在以后要读出来。考察进出数据存储的数据一致性会由于以下的事实而变得复杂：一个数据元素也许能在完全不同的DFD上进出数据存储。

检查数据流的一致性是一个简单但很乏味的事情。幸运的是大多数的CASE工具自动执行数据流的一致性检查。但这些CASE工具为了精确的识别处理的内部逻辑对系统分析员有很高的需求。没有精确的过程描述，CASE工具（或人）不可能知道什么数据元素作为输入，以及内部处理逻辑产生什么输出。

6.3 详细记录DFD部件

在传统的方法中，数据流图在一个图上描述了所有的三种类型的内部系统部件——处理、数据流和数据存储，但仍需要更进一步来描述细节。首先，需要详细描述每一个最低层处理；其次，系统分析员需要根据数据流包含的数据元素来定义数据流。数据存储也需要根据数据元素定义。最后，系统分析员也需要定义每一个数据元素。

6.3.1 处理描述

在DFD中的每一个处理都必须正式定义。定义处理有多个可选的方法，其中包括已经讨论过的处理分解。就像前面所讨论的那样，在处理分解中，一个高层的处理正式地包括低层处理的DFD定义，而这些低层处理又可以依次进一步分解成更低层的DFD。

最终处理可以达到无须再由DFD定义的状态。当处理简单到足以用其他的方法如结构化英语、决策表或决策树把它描述清楚时，就达到了这个状态。在这些方法中，每一种处理都被描述成一个算法，并且分析员通过确定最紧凑、可读性最好且无歧义的方法来选择最合适的表达格式。在大多数情况，结构化英语是最好的方法。

结构化英语非常仔细地使用简短语句描述处理。结构化英语看起来像程序中的语句，但它用不着计算机的概念。在这里也要遵守结构化编程的规则，并且为了清楚也要求缩进。例如，图6-20所示为一套选举后投票处理的简单指令。一些语句是非常简单的指令，另一些则是重复指令，还有其他一些指令程序执行某个指令集合。程序总是从最上面开始，到最下面结束。所以，在这里应用结构化编程的规则。注意，虽然用结构化英语描述的处理没有必要是一个计算机程序（它可由一个人执行）但它仍是一个逻辑模型。这是明确的，所以任何人只要照着这些指令都可以得到相同的结果。

结构化英语：一种书写处理规范的方法，它将结构化编程技术和叙述性英语结合起来。

图6-21所示为RMO的一个处理描述的例子。注意，处理描述是如何提供更多的关于处理功能的特殊信息的。如果这个处理描述方法变得复杂，则分析员应使用另一种。超长（如超过20行）或多层缩进（表明复杂的条件逻辑）说明一个结构化英语可能太复杂。多层缩进有时可以将描述转换成等价的决策表或决策树。其他情况下，处理可能需要分解。

```
Process Ballots Procedure

Collect all ballots
Place all ballots in a stack
Set Yes count and No count to zero
Repeat for each ballot in the stack
    If Yes is checked then
        Add one to Yes count
    Else
        Add one to No count
    Endif
    Place ballot in counted ballot stack
Endrepeat
If Yes count is greater than No count then
    Declare Yes the winner
Else
    Declare No the winner
Endif
Store the counted ballot stack in a safe place
End Process Ballots Procedure
```

图6-20　一个结构化英语的例子

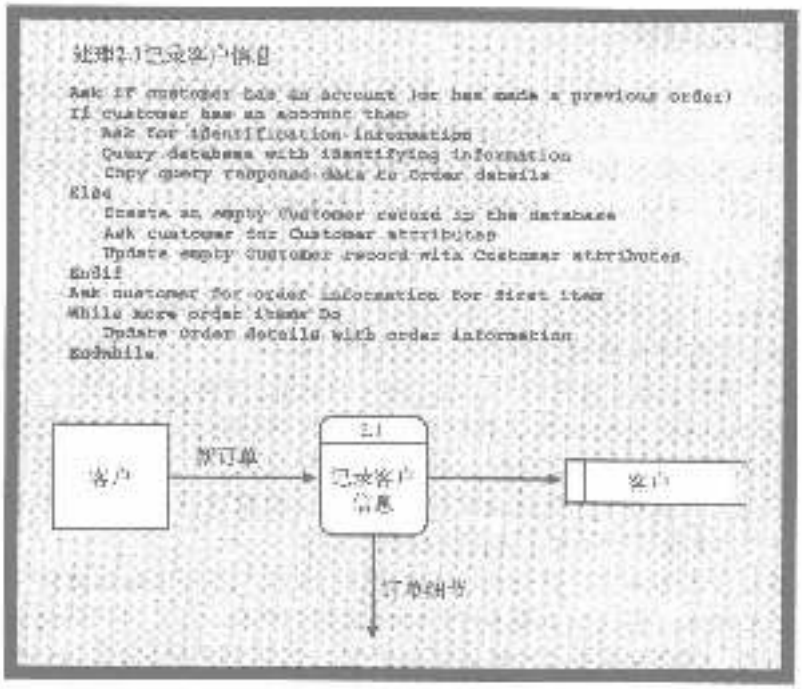


图6-21　RMO处理2.1（记录客户信息）和它的结构化英语处理描述

结构化英语非常适合用来描述带有一系列处理步骤和相对简单控制逻辑（如一个简单的循环语句或一个if-then-else语句）的处理。但结构化英语不适合描述有下列特点的处理：

- 复杂的决策逻辑
- 很少有（或没有）顺序处理步骤

当需要考虑大量的变量和这些变量许多可能的组合时，这样的决策逻辑就变得复杂起来。如果使用结构化英语描述带有复杂决策逻辑的处理时，势必会导致描述冗长且很难理解。例如，考虑如图6-22所示的计算运输费用的结构化英语描述。注意，这个描述相对较长且包含

较多的控制语句（if，else，endif等语句）。

```
If YTD purchases > $250 then
    If number of items ordered > 4 then
        If delivery date is next day then
            delivery charge is $35
        Endif
        If delivery date is second day then
            delivery charge is $10
        Endif
        If delivery date is seventh day then
            delivery charge is $1.50 per item
        Endif
    Else
        If delivery date is next day then
            delivery charge is $6 per item
        Endif
        If delivery date is second day then
            delivery charge is $2.50 per item
        Endif
        If delivery date is seventh day then
            delivery charge is zero (free)
        Endif
    Endif
Else
    If number of items ordered > 4 then
        If delivery date is next day then
            delivery charge is $15
        Endif
        If delivery date is second day then
            delivery charge is $15
        Endif
        If delivery date is seventh day then
            delivery charge is $10
        Endif
    Else
        If delivery date is next day then
            delivery charge is $7.50 per item
        Endif
        If delivery date is second day then
            delivery charge is $3.50 per item
        Endif
        If delivery date is seventh day then
            delivery charge is $2.50 per item
        Endif
    Endif
Endif
```

图6-22 计算运输费用的结构化英语处理描述

决策表和决策树可以比结构化英语更简捷地描述复杂决策逻辑。图6-23和图6-24分别使用决策表和决策树展示了在图6-22中结构化英语所描绘的逻辑。把决策逻辑描述为表或树的形式比相应的结构化英语的可读性要高。其中，决策表更加严密，而决策树更易读。有时，分析员在决定那种方式最能准确地描述特定处理之前这三种方法都应用来描述该处理。

决策表：处理逻辑的一种表格表示方法。其中包括决策变量、决策变量值、行为或公式。

决策树：用枝状树形结构组织起来的一系列处理逻辑进行图形化的描述。

YTD购买>$250	是						否					
商品数量（N）	N>4			N≤4			N>4			N≤4		
发货日期	次日	第2天	第7天	次日	第2天	第7天	次日	第2天	第7天	次日	第2天	第7天
发运费用（$）	25	10	N*1.50	N*6.00	N*2.50	Free	35	15	10	N*7.50	N*3.50	N*2.50

图6-23 计算运输费用决策表

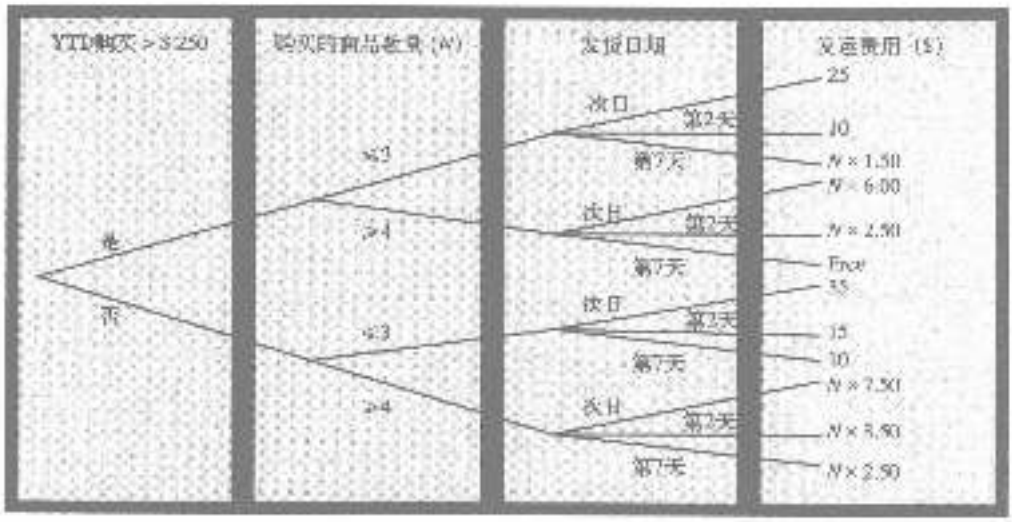


图6-24　计算运输费用决策树

以下是用于构建决策表的步骤：

① 辨别决策变量和它的可能的取值（值域）；

② 把每一个决策参数值（值域）数目的乘积计算出来作为计算决策变量的所有可能的组合数；

③ 画一个表，这个表要比第2步所计算出来的数目多一个栏目（这多出的栏用于写决策变量名、处理动作或计算的描述）。每个决策变量、处理动作和计算公式在表中都有对应的一行；

④ 把可能取值数目（值域）最小的决策变量放在第1行，左第1列写下这个变量名，把余下的列按照这个变量取值（值域）进行分组；

⑤ 把可能取值数目（值域）次小的决策变量放在第2行，在第1列写下这个变量名，计算该变量和上面所有变量的取值数目（值域）的乘积，把所得的值作为这一列分组数来划分余下的列，并把取值（或值域）以规定模式插入其中；

⑥ 按第5步的方式继续插入行，直到所有的变量均已考虑在这个表中；

⑦ 为每一步计算或动作加一行，对每一个计算单元，为那些在同一列但在该单元上面出现的决策变量组合插入适当的常数值或者公式。如果当决策变量出现单元上方列中所示的值时如果动作执行了，则应在同一列放置一个打勾标志。

现在让我们依照以下的步骤来看一下如何构建图6-23所示的决策表。这里一共有三个决策变量：当年已购买（YTD购买）金额、订购商品数量，发货日期。YTD购买额有两个相应的变化区域：小于250元和大于等于250元。注意，变量范围必须是完全且有效的。订购商品数量也有两个相应的变化区域：小于等于3和大于等于4。发货日期有三个可能的值：次日，第2天和第7天。因此，共有2 × 2 × 3 = 12种组合，所以在表中需要13列来容纳决策变量名，计算公式和活动名。

由于当年已购买（YTD购买）金额和订购商品数量均有两个取值范围，因此它们中任何一个都可以占有第1行。这里选择YTD购买额。由于YTD购买额有两个取值范围，所以分成两组，每组有12 ÷ 2 = 6列，并且给每个可能的值分类。下一行是订购商品数量，它也有两个取值，所以要分成4组，每组3列（即12÷2÷2，12÷2是将产品数量分组，除以2是将YTD购买额进行分组）。正如图例中所示，将订购商品数量的取值范围以一种有规则的模式插入各组列中。发货日期现在该插入到表中了，由于它是最后一个决策变量，不需要在它下面再分出列

来，我们只是简单地把它的取值范围有次序地插到各列中。

最后一步是把包含计算公式和运输费用的值插入到最后一行上。每一个单元包含一个值或组合决策变量值的公式。例如，如果某客户YTD购买额超过250元，订购单上超过3个订购项目或者选择第7天发货，则将会免去该客户的运费，如果某客户YTD购买额少于250元或订购单上是3个或少于3个订购项目或选择次日发货，则该客户要加35元的运费。

如果决策表用于表达实现一个或多个动作的话，而不是像前一个例子一样计算数值，这个表必须为每一个动作开辟一行，在这些行中的单元要打勾以表明哪一个动作在什么条件下执行。图6-25所示为这种类表格的一个简单例子。表中含有两个动作行。如果在决策变量值的正下方的单元中打了勾，则表示该动作要执行。例如，如果客户是新客户且运输中包含一个25天后延期订购的条目，则运输要加急并且在运输包中要有详细的退货说明；如果客户不是新客户且没有包含一个在25天后延期订购的条目，则什么动作也不会发生。

新客户	是		否	
延期25天的订购条目	是	否	是	否
包含详细的退货说明	√	√		
加急发送	√		√	

图6-25 一个有多个动作行的简单决策表

使用与上面所说的构建决策表类似的步骤，你可以构建出决策树来。主要的差别是在决策表中的行作为决策树中的列，反之亦然。为了讲清楚这一点，假想在图6-23中从左上至右下画一些线，然后沿着这条假想的线转置这个表，并把这个转置后的表与图6-24做一比较。决策表与决策树的另一个重要的差别是在决策树中使用有标号的分支而不是分组的列来表示变量值。

实践指导

传统方法的功能需求文档包括：

(1) 一个具有属性的ERD；

(2) 一组DFD（关联DFD、DFD片段和其他需要的细节DFD）；

(3) 处理描述；

(4) 数据流定义、数据存储定义和数据元素定义。 ■

6.3.2 数据流定义

数据流是数据元素的集合，所以数据流定义将列出所有的数据元素。例如，一个简化了的“新订单”数据流包含客户名、信用卡号码和商品种类数及每一类的数量（如图6-14所示的过程2.1）。一些数据元素实际上是其他数据元素的结构，例如一个客户名包含首名、中间首字母和姓。这些数据元素中的大部分由系统保存，所以它们与ERD中的数据实体属性一致。

数据流定义：数据流内容和内部结构的文本描述。

一些数据流定义包含一个重复的结构。在“新订单”的例子中，每一个数据流包含许多条目及其数量（一个重复的组）。记录这些结构的文档是非常重要的。数据流定义的符号也是多种多样的。有一个方法非常简单地就能列出所有的数据元素，如图6-26所示。有许多值的元素可以呈现出来。另一个方法是使用如图6-27所示的代数公式。数据流“等于”或“包含”一个元素加



图6-26 只列出数据元素的数据流定义

另一个元素等。可能有许多值的元素组要用大括号括起来。这个例子展示“新订单”“等”客户名“加”信用卡信息“加”“一个或多个”库存条目种类和数量。在这个例子中，可以把客户名单独作为一个元素结构进行定义。

```
New-Order = Customer-Name + Customer-Address +
Credit-Card-Information + 1{ Item Number + Quantity }
```

图6-27　代数公式方式的数据流定义（“新订单”）

图6-28和图6-29展示了一个复杂的报表及其相应的数据流定义。这个报表的结构是在产品上重复分组，而产品又嵌入库存条目的分组。数据流定义通过以下方式表示这种结构：用大括号括住项目组，同时在外层再用大括号括住产品组。

Rocky Mountain Outfitters — Products and Items

ID	*Name*	*Season*	*Category*	*Supplier*	*Unit Price*	*Special*	*Special Price*	*Discontinued*
RMO125	*Outdoor Field Sportcoat*	*Mens*	*C*	*8201*	*$39.00*	*	*$0.00*	*No*
Description	*Outdoor Nylon Jacket with Lining*							

Size	*Color*	*Style*	*Units in Stock*	*Reorder Level*	*Units on Order*
Large	Blue		1500	150	
Large	Green		1500	150	
Large	Red		1500	150	
Large	Yellow		1500	150	
Medium	Blue		1500	150	
Medium	Green		1500	150	
Medium	Red		1500	150	
Medium	Yellow		1500	150	
Small	Blue		1500	150	
Small	Green		1500	150	
Small	Red		1500	150	
Small	Yellow		1500	150	
Xlarge	Blue		1500	150	
Xlarge	Green		1500	150	
Xlarge	Red		1500	150	
Xlarge	Yellow		1500	150	

ID	*Name*	*Season*	*Category*	*Supplier*	*Unit Price*	*Special*	*Special Price*	*Discontinued*
RMO127	*Hiking Walkers*	*All*	*Footwear*	*7093*	*$49.95*		*$0.00*	*No*
Description	*Hiking Walkers with Patterned Tread Durable Uppers*							

Size	*Color*	*Style*	*Units in Stock*	*Record Level*	*Units on Order*
10	Brown		1000	100	
10	Tan		1000	100	
11	Brown		1000	100	
11	Tan		1000	100	
12	Brown		1000	100	
12	Tan		1000	100	
13	Brown		1000	100	
13	Tan		1000	100	
7	Brown		1000	100	
7	Tan		1000	100	
8	Brown		1000	100	
8	Tan		1000	100	
9	Brown		1000	100	
9	Tan		1000	100	

图6-28　RMO客户支持系统的一个样例报表

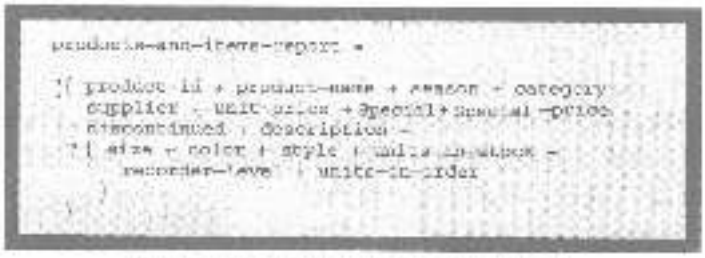


图6-29 RMO产品与条目报表的数据流定义

6.3.3 数据存储定义

由于在DFD中的数据存储在ERD中表示数据实体，所以它无须特别定义（除了给读者一些有关ERD的提示）。如果数据存储与ERD不相关联，则分析员可以用定义数据流的方法把数据存储定义为一个元素集合（可能使用一个结构）。

6.3.4 数据元素定义

数据元素定义描述数据类型，如字符型、够型、浮点型和布尔型。每一个元素应该清楚地表明它表示什么。有时这些描述非常特别。有时用订单收到时的支付日期定义售出日期。另外，售出日期也可以是订单发出的日期。有时在同一个公司不同的部门对同一个元素有不同的定义，所以对分析员来说，确切地说明元素的意义是很重要的。

数据元素定义的其他部分根据数据类型的不同而不同。字符型通常要定义其长度，例如，中间名字母可能最多只需要一个字符的大小，但是名字要多长才合适呢？数值通常要定义其最大值和最小值作为其有效范围。有时，某个元素允许一个特殊的值，如有效码。如果这个元素是一个代码，则定义有效码及其意义是非常重要的，例如，代码A表示立即发运，代码B表示存放一天，代码C表示需要紧急确认的发运。图6-30所示为一些样例数据元素的定义。

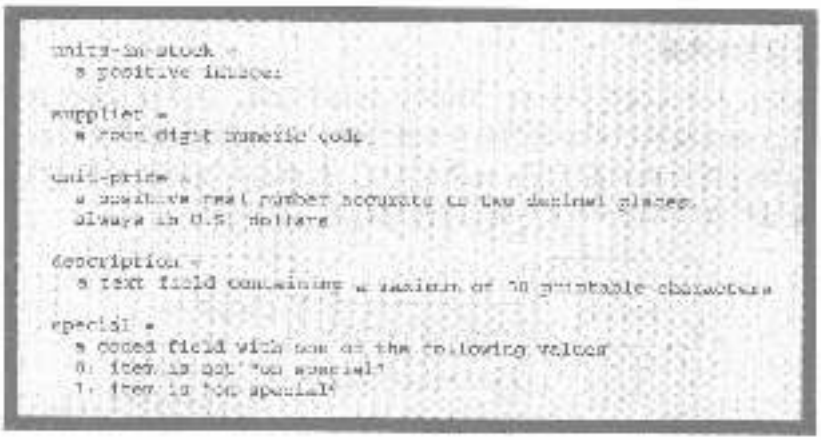


图6-30 数据元素定义

分析员需要保持所有这些数据元素定义的一个中心存储，主要作为一个项目的参考和保证一致性。数据字典是关于数据流、数据存储和数据元素的定义库。在一个较小的项目中，数据字典可以是一个简单活页记事本或者按词查找的文件。在较大型的项目中，通常项目管理或者CASE工具都有这个数据字典。同时数据字典也可以有过程描述。

数据字典：关于数据流、数据存储和数据元素的定义库。

6.3.5 DFD总结

图6-31展示了传统分析模型的各个组成部分：实体-联系图、数据流图、处理定义和数据定义。这4个组成部分构成了大多数系统一系列互锁的详细说明。数据流图提供对系统的最高层的考察，在这里结合了处理、外部实体、数据存储和数据流。其他的组成部分更详细的描述了数据流图的某些方面。

这些模型几乎都是在20世纪70年代和80年代开发的，它们是传统结构化分析方法的一部分（参见在"参考资料"的Yourdon1989）。它们用于记录一个系统逻辑需求的完全文档。但是，一些分析员从其他方法中借鉴其他模型来增强这个结构化的模型。这样的模型也许被用来描述没有包含在结构化模型中的信息，或者可以使用传统不同的方式来表示相同的信息。本章后面的部分描述一些"借来"的模型，以及它们是如何能增强结构化分析模型的。

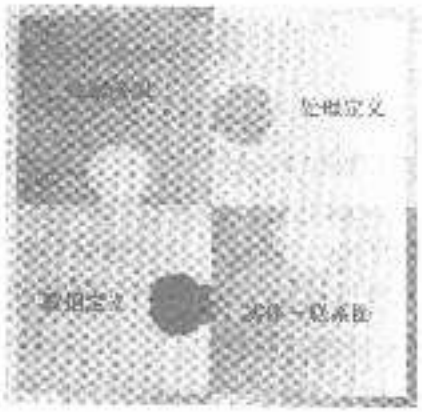


图6-31　传统系统分析模型的各个组成部分

6.4 信息工程模型

信息工程（IE）在20世纪80年代初由James Martin开发。在那时，IE与结构化系统开发方法虽然有很多共同的特性但却被认为是相互竞争的。由于两种方法都比较成熟，所以许多分析员开始寻求如何将两者最好的特性结合起来。这一节给出了IE方法的一个概览并描述一些IE系统模型，随后描述如何将它用于本章前面介绍的结构化分析模型中。

信息工程（IE）：一种比结构化方法更严格完全的重点集中在战略规划，数据建模和自动化工具上的系统开发方法。

6.4.1 IE系统开发生命周期

图6-32所示为IE系统开发生命周期（SDLC）的各个阶段。IE系统开发生命周期的第一个阶段集中在对企业建模和创建协调的信息系统规划。企业根据它的策略目标，规划、信息技术政策、存储数据需求和主要功能区域建立模型。数据和企业功能模型中的交叉部分可用于确定特定的业务范围以对IE SDLC中第二阶段进行进一步的分析上。

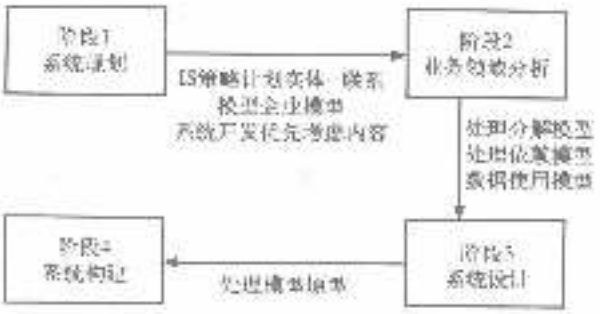


图6-32　信息工程系统开发生命周期的各个阶段

在IE系统开发生命周期的第一阶段出现的两个关键特性是把业务作为一个整体来关注以

及详细数据存储。IE首先根据企业的策略、规划、目标、现状和组织结构描述企业和它的环境，然后才开始详细说明信息技术设施和信息处理应用的需求。集中考察数据存储基于这样的一个假设：业务数据是一种有组织的资源，它不像业务过程那样经常变化。内部和外部数据需求被假定用于引起处理需求和过程结构，但反之则不成立。

IE系统开发生命周期的第二步是描述在企业中的每一个业务范围的处理需求。每一个业务范围被分解为相关处理的一个层次。使用处理分解模型描述层次，并使用处理依赖模型描述处理之间的内部关系。使用数据应用模型集来描述处理和数据之间的关系。

IE的系统开发生命周期的第三步是开发业务处理的详细设计。IE使用很多不同的技术和模型描述处理。这些方法强调用户输入处理设计和处理模型的正确性。开发人员开发图形化处理模型，用户检查这些模型是否正确。

IE系统开发生命周期的最后一步是系统构建。IE方法使用CASE工具和代码生成器实现构建阶段。过程模型和在前一阶段开发的原型用于自动生成系统部件。代码生成器以CASE工具开发出来的数据和过程模型作为输入，以产生的应用程序和数据库创建命令作为输出。以前IE曾经以过程性编程语言（如COBOL语言和C语言）和关系数据库管理系统作为目标，后来的IE CASE工具和代码生成器以面向对象环境作为目标。

6.4.2 IE和结构化开发的比较

图6-33对IE和传统的结构化方法的特征进行了比较。两种方法最显著的区别就是它们的范围。IE用来说明企业范围的信息处理需求。它的第一个阶段将企业作为一个整体，在组织结构、目标和策略范围内描述数据和信息处理。

	信息工程	结构化方法
什么样的系统体系最合适？	企业	任意
与公司战略规划是否一致？	是	否
技术和模型相互间结合紧密吗？	高度紧密	中等紧密
分析的重点是什么？	数据	处理
其他方法的技术可以很容易地结合进来吗？	否	是

图6-33 信息工程和结构化方法的比较

相比之下，结构化方法既不对应用程序的范围做假设，也不对先于结构化分析的规划活动做假设。结构化方法可以适应于整个企业、企业内相对较小的系统或介于两者之间的任何系统。结构化分析不需要明确企业级的规划和描述活动。所以，它的模型不直接与企业的组织结构、企业策略和目标、存在的和规划的信息技术设施等信息结合。

两种方法论的另一个重要差别是相应的工具和技术紧密结合的方式。IE的紧密集合表现在：前一阶段的输出是下一个阶段的输入。这种方法论是完整的且是全封闭的。所以，在IE内引入其他技术或模型是非常困难的（且通常是不必要的）。相反，在结构化方法中所有的技术或模型只是中等程度地结合在一起。在这些方法中有许多间隙，因此从诸如IE之类的其他方法中引入技术或模型是可能的（且通常是期望的）。

在IE的第一阶段一个企业的ERD就开发出来了。这个模型提供了一些信息，这些信息在以后IE的建模和活动设计过程中将被广泛使用。结构化分析也开发和使用ERD，尽管在结构化方法中它的作用不如在IE方法中这么重要。因为两种方法均使用了ERD，因此，我们就可能在结构化方法中引入ERD的IE模型。下一节描述一些IE模型，这些模型给在本章前面描述的结构化方法提供了一个很好的补充。

6.4.3　处理分解和依赖模型

与结构化分析相比，IE使用不同的方法去建立处理模型。IE处理模型显示了三种类型的信息：

- 处理分解为其他处理
- 处理间的依赖关系
- 内部处理逻辑

分析员首先通过把业务领域分解成子处理来进行业务范围的模型化。图6-34所示为RMO客户支持系统的处理分解图示例。图中显示系统（业务领域）被分解成前面图6-10所描述的几个主要的子系统。订单输入子系统被进一步分解成能轮流分解的处理。在图中，左上角有三点的处理表明该处理的进一步分解画在另外的纸上。

处理分解图：在不同的抽象层表示处理之间层次关系的一种模型。

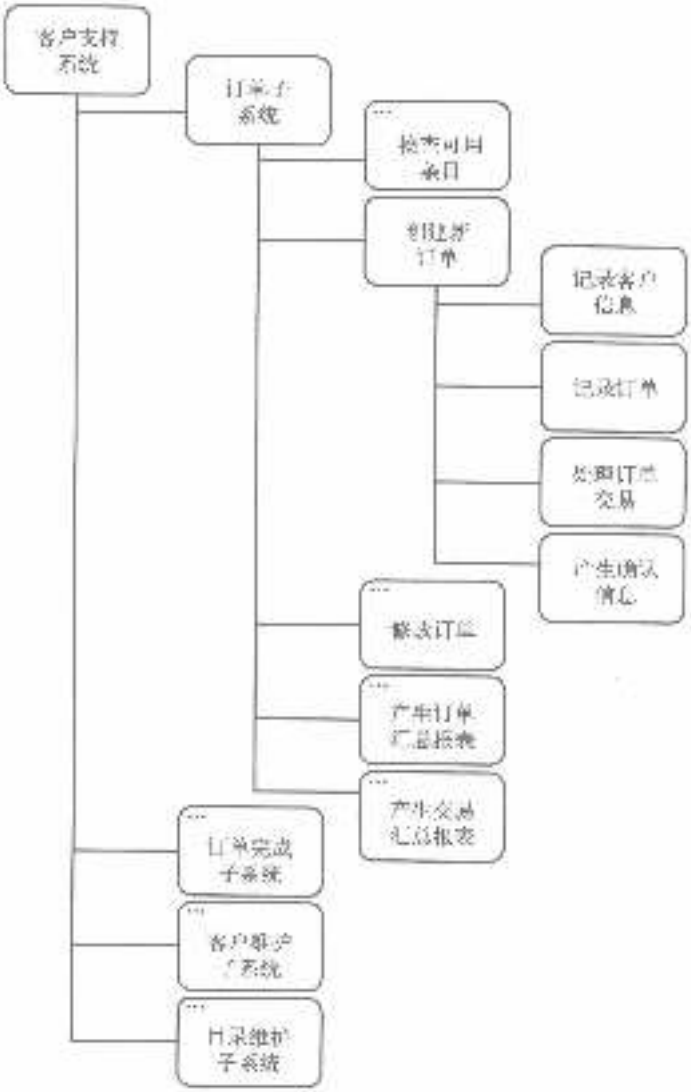


图6-34　RMO客户支持系统的处理分解图

把图6-34的信息内容与以前的RMO DFD（如图6-9和图6-14）示例做比较。使用IE和结构化开发的过程描述有一些相似的地方。一个显而易见的相似之处是用以代表处理的符号。两种方法都使用椭圆形符号，只是它们画在纸上时方向不一样。

另一个相似之处是在不同的抽象层处理之间的层次关系。结构化DFD并不是直接将处理分解放在不同的DFD上，而是使用层次编码系统把各种各样的DFD结合在一起。处理依赖图直接在一张纸上表现处理之间的层次关系。两种方法使用不同的方式表现相同的处理分解信息。

图6-35所示为一个名为创建新订单的处理依赖图，这个图对应于图6-14所示的处理分解图创建新订单。在图中标有收到订单的大箭头表示触发第一个处理事件，连接处理的线和箭头表示处理之间的依赖关系。处理产生确认信息依赖于处理订单交易，即表示直到处理订单交易处理完毕产生确认信息才可以开始。

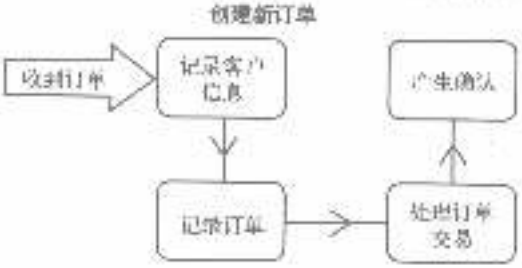


图6-35 处理依赖图

处理依赖图：用各种实体描述处理顺序及其交互的一种模型。

依赖关系与数据流的差别是微妙的但也是重要的。处理依赖图不直接显示处理之间的数据流（注意，连接处理没有名字和标号的连线）。产品确认信息依赖于处理订单交易是因为确认一个没有完成的订单（如一个信用卡没有通过验证的订单）是没有意义的。即使一个处理产生数据，另一个处理使用数据，这种处理之间的数据流也不直接反映在处理依赖图中。

图6-36所示为一个带数据流的处理依赖图。处理依赖关系用粗线表示，而数据流用细线表示。虽然这种图与结构化的DFD完全不同，但IE也称这样的图为数据流图。图6-14与图6-36的差别是显著的。虽然处理的标号和顺序一样，但在信息内容上有比较大的差别。结构化DFD明确地画出了进出处理的数据流。这些数据流（和实体）在处理依赖图中被忽略了。结构化DFD从一个处理到另一个处理传送数据，而处理依赖图通过数据存储传送数据。原因是在业务领域的分析阶段，IE认为处理从数据库读写数据。

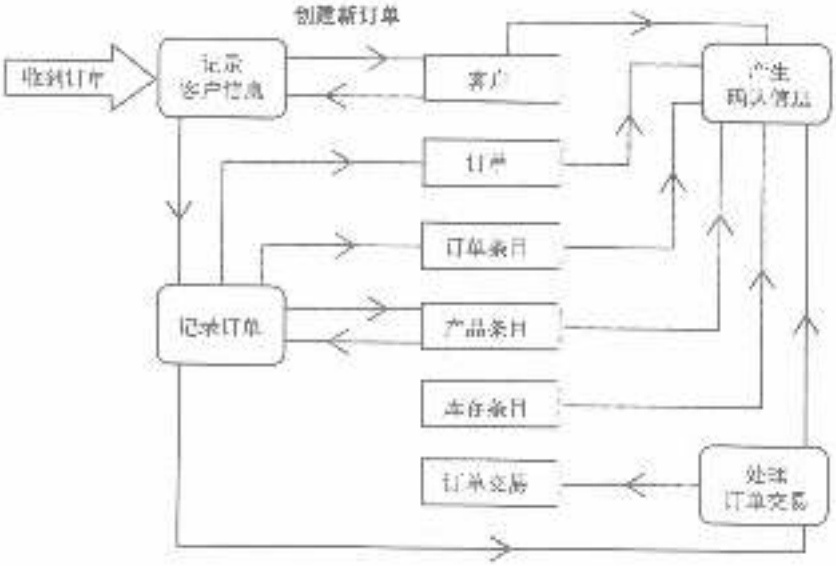


图6-36 带数据流和触发事件的处理依赖图

上面这张图的差别展示了描述一个逻辑（而非物理）处理模型的不同方法。IE有意避免显示实体和处理之间的数据流，因为这里可能引进不必要的有关处理实现的物理假设。IE假设数据流的“什么时候”和“哪里”是设计结果，直到设计阶段数据源、目的和路线才需要明确的定义。

相反，结构化分析需要描述进出实体和处理的数据流。在处理之间的数据流可以强加在数据存储之上。但结构化分析的规则既不要求也不鼓励这么做。使用IE方法很难创建一个非逻辑的模型。结构化方法提供一套可同时适应于逻辑和物理模型的建模工具。现在应由分析员决定当开发逻辑DFD时需要什么样的限制（如果有的话）。

处理分解和处理依赖图都对结构化方法有用。处理分解图有助于用图形化方法展示大量处理之间的层次关系。处理依赖图在考察和描述处理关系和数据流的逻辑本质时是一个有用的选择。

在传统的结构化模型中加入其他模型的主要危险是会出现冗余。结构化模型设计时应尽可能做到没有冗余，以便使模型更容易修改。IE模型描述的信息在结构化的DFD上也描述了。所以，在结构化模型中做的任何修改也需要在相应的IE模型中修改。这就使完成一次修改更加困难，并且增加了模型之间有不一致性的可能。

6.5　结点和网络通信

当为结构化系统分析主要集中在逻辑建模中，有时在分析时会忽略诸如处理结点和网络等的物理问题上。但是，关于处理、数据和用户分配等大量的信息在设计阶段的早期是需要加以考虑的。例如，包括以下信息：

- 用户结点的数量
- 用户在特定结点的处理和数据访问的需求
- 处理和数据访问需求的容量和时间

有些信息是需要做出前期的设计考虑的。如计算机系统的分布性、应用程序软件和数据库部件。同时，网络用户容量和处理结点等也需要加以考虑。信息工程方法论详细地说明了一些表，这些表描述了ERD中的实体关系和其他信息系统及企业部件。这些表描述了在一个企业中的数据源和数据的使用。前面的例子包括了这种表，它描述哪个处理最先捕获有关实体的信息，以及哪个处理以后使用这些数据。IE在SDLC的早期开发这些表，然后用它们来指导以后的处理设计和结构。

许多分析员早在传统系统开发生命周期的分析阶段收集和总结结点信息。在分析阶段收集这些信息使得分析员能够在后面的两个分析阶段的活动中进行更好的决策，这两项活动是生成和评估选择方案及与管理人员检查推荐方案。结点信息在设计阶段的许多活动中也是十分有用的，如设计和集成网络、设计应用程序结构以及设计和集成数据库。

收集结点信息的第一步是确定和描述要成等要工作的地点。可能的地点包括业务办公室、仓库和生成车间，以及不太明显的地点，如客户或提供商的办公室里、员工家里、饭店里、车里。所有的这些地点均需列出，且应该画一个结点图以便以图形化的方式对这些地点进行汇总。图6-37所示为一个RMO的结点图。结点图告诉分析员需要哪些网络连接，并且包括可以提醒每个人处于所有结点的用户都应该被考虑到哪个系统中。

结点图：可以确定系统中所有处理结点的图表或地图。

下一步是列出每个结点的用户都要执行的功能。使用事件表，分析员可以列出每一个活动在那里执行。如图6-38展示了一个活动结点矩阵，它总结了这些信息，每一行是一个系统活动，每一列是一个结点。许多活动可以在多个结点执行。

活动结点矩阵：描述处理和执行处理的结点之间关系的一个表。



图6-37 RMO的结点图

活动	地点				
	公司办公室（帕克城）	分布的仓库（盐湖城、阿尔伯克基、波特兰）	邮购订单（普罗沃）	电话销售（盐湖城）	客户直接交互（预先考虑的）
查询可用条目	×	×	×	×	×
创建新订单			×	×	×
更新订单				×	×
查询订单状态	×	×		×	×
记录订单完成情况		×			
记录延期订单		×			
创建订单返回		×			
提供目录信息			×	×	×
更新客户账户	×		×	×	×
分配促销包	×				
创建客户费用调整	×				
更新目录	×				
创建特殊商品促销	×				
创建新目录	×				

图6-38 RMO客户支持系统的活动结点矩阵

回想一下，RMO也有一个进行中的有关库存管理的系统项目。库存管理系统对于生产设备可能有很多的操作，但客户支持系统不会有。另外，公司有计划要将零售店与库存管理系统而不是与客户支持系统合并。所以，这些结点没有出现在活动结点矩阵中。

有时需要创建一些其他的矩阵以强调存取需求。一个方法是在活动数据矩阵中列出活动和数据实体（对象的类）。这个矩阵展示需要存取数据或对象的活动。在传统方法的DFD片段或在面向对象方法的顺序图中都可以找到这类信息。不管使用哪种方法，创建一个矩阵来总结这类信息都是有用的。

活动数据矩阵：描述存储数据实体、访问它们的结点及访问类别的一个表。

图6-39所示为RMO的一个活动数据矩阵。矩阵的单元中，外加信息用于分清什么活动对应这个数据。字母C表示创建新数据的活动，R表示读取数据的活动，U表示更新数据的活动，D表示删除数据的活动。首字母缩写CRUD（Create、Read、Update、Delete）经常用于描述这类矩阵。

CRUD：Create、Read、Update、Delete的首字母缩写。

活动	数据实体										
	目录	客户	库存条目	订单	订单条目	订单交易	包装	产品条目	返回条目	发运部	发运商
查询可用条目			R								
创建新订单		CRU	RU	C	C	C	R	R		C	R
更新订单		RU	RU	RUD	RUD	RUD	R	R		CRUD	R
查询订单状态		R		R	R	R				R	R
记录订单完成情况					RU					RU	
记录延期订单					RU					CRU	
创建订单返回		CRU		RU		C			C		
提供目录信息	R		R				R	R			
修改客户账户		CRUD									
分配优销包装	R	R	R				R	R			
创建客户费用调整		RU				CRUD					
更新目录	RU		R				RU	R			
创建特殊商品促销	R		R				R	R			
创建新目录	C		R				CRU	R			
C=创建新数据 R=读取已有数据 U=更新已有数据 D=删除已有数据											

图6-39　RMO的活动数据矩阵

小结

在事件表和实体-联系图（ERD）的结合中数据流图（DFD）被用来为系统需求建立模型。DFD对系统建成的模型包含一些处理、数据流、外部实体和数据存储。DFD相对容易阅读，因为它们使用几个符号图形化地描述了系统的关键特性。由于有大量的特性需要描述，所以人们开发了许多类型的DFD，如关联图、DFD片段、子系统DFD、事件划分DFD和处理分解DFD。

每一个处理、数据流和数据存储都需要有详细的定义。处理可以被许多方法定义，如结构化英语、处理规格说明书、决策表、决策树和处理分解DFD。当内部处理太复杂以至于用其他方法无法给出一个可读性高且简短的定义时，我们就使用处理分解DFD。数据流根据其部件数据元素和内部结构而定义，数据元素可以进一步由其类型和允许值定义。数据存储对应于ERD中的实体，因此不需要再定义。

数据流图可以用从其他方法论中借来的模型进行补充，以提供更进一步的信息和考察需求的另一个选择。信息工程（IE）模型是对传统分析模型的一个有用的补充。特别有用的模型包括：处理分解图、处理依赖图、结点图、活动结点矩阵和活动数据矩阵（或CRUD矩阵）。处理分解模型提供了关于对多个处理层的DFD如何关联的一个很好的概括。处理依赖图提供处理细节的考察，它强调在存储实体间进行交互，而不是使用处理之间的数据流。

结点图、活动结点矩阵和活动数据矩阵描述系统结点的重要信息。结点图概括了用户使用系统的地理位置，活动结点矩阵描述哪个处理在什么位置完成，活动数据矩阵描述在哪里及如何使用数据存储。

我们现在已经介绍了在传统的系统分析方法中用来描述需求的所有的模型，第7章将介绍在面向对象的系统分析方法中用来描述需求的模型。第8章将介绍从系统分析到系统设计的转换。

关键术语

activity-data matrix	活动数据矩阵
activity-location matrix	活动结点矩阵
balancing	平衡
black hole	黑洞
context diagram	关联图
CRUD	CRUD
data dictionary	数据字典
data flow	数据流
data flow definition	数据流定义
data flow diagram（DFD）	数据流图
data store	数据存储
decision table	决策表
decision tree	决策树
DFD fragment	DFD片段
event-partitioned system model, or diagram0	事件划分系统模型或0层图
external agent	外部实体
information engineering（IE）	信息工程
information overload	信息超量
level of abstraction	抽象水平（或抽象层次）
location diagram	结点图
minimization of interfaces	接口最小化
miracle	奇迹
process	处理
process decomposition diagram	处理分解图
process dependency diagram	处理依赖图
rule of 7±2	7±2规则
structured English	结构化英语

复习题

1. 列举至少三种DFD。每种类型的图用于描述什么？

2. 列举DFD的五个部分（符号），简要地说明每种符号的意义。
3. 分析员如何决定一个人或组织是DFD中的外部实体还是一个或多个处理？
4. 在事件划分DFD中的处理可以通过一个详细DFD和处理规格说明书描述。分析员如何决定使用哪种形式更合适？
5. 说明在一个DFD上如何表示一个事件表的每一列（即该用什么符号）？
6. 如何在DFD中描述ERD中的实体？如何在DFD中描述ERD中的关系？
7. 什么特征在物理DFD中出现而绝不会在逻辑DFD中出现？
8. 当评估DFD的质量时，分析员主要考察什么特征？
9. 什么是黑洞？什么是奇迹？如何发现它们？
10. 为什么分析员使用决策表或决策树而不是结构化英语描述某个处理？
11. 列举作为传统分析阶段建模工具有用补充的IE处理模型（如DFD、数据定义和处理规格说明书）。相对于传统模型，描述IE的优缺点。
12. 同时使用传统模型和IE处理模型的缺点是什么？
13. 列举并简要描述信息工程的生命周期的各个阶段。
14. 什么是活动结点矩阵？它如何与DFD相联系？
15. 什么是活动数据矩阵？它如何与DFD和ERD相联系？

思考题

1. 假设你准备用一个DFD来描述一个由抵押经纪人创建、同意和结束一次抵押贷款的过程。这个抵押经纪人是作为一个外部实体还是作为一个或多个处理？为什么？对结点代理、信用部门和处理抵押业务的银行来说又是什么呢？
2. 考察图6-6所示的课程注册系统。为了实现一个完全的功能系统，还需要补充其他的处理吗？提示：悬浮和奇迹可以显示出在DFD遗漏的处理步骤。
3. 为图6-12中的处理3（修改订单）开发一个处理依赖图。你做出的DFD与图6-13有什么相似之处？设法重画这三个图以消除冗余。与原图相比，你的修改图是否更可读？
4. 假设RMO的订单输入子系统的事务总结报表（如图6-12所示的处理5）包含了所有订单的清单，这些订单在用户输入的日期范围内创建。报表头包含报表名称、日期范围和打印报表的日期时间。对每一个订单，报表列出了订单号、订购日期、订购总数和支付方式。在每一个订单内部，报表列出了所有的包括项目号、订购数量（或退货数量）和价格等的订单项目和返回条目。报表汇总包括所有的订单总数、平均数、平均条目价格和平均返回条目价格。为这个报表书写一个数据流定义，并为产生这个报表的处理写一个规格说明书。
5. 为图6-6中的所有处理创建一个处理依赖图。
6. 为图6-6中的处理1创建一个处理分解图（带数据流）。
7. 为图6-6中的课程注册系统创建一个活动数据（CRUD）矩阵。

实验练习

1. 开发一个物理DFD。这个DFD为你的一次货物采购活动建模。其时间范围是从你写下购买清单到你把所购买物品堆放到家中。把你的DFD构建成线性顺序的处理。再为同样的处理开发一个逻辑DFD。设法开发一个对以下两种情况都有效的图：你现在购买货物方式的逻辑描述和你不用离开家门就可以购买货物方式的逻辑描述。
2. 考虑在你的学校中学位、专业和研究方向的需求。在学校目录册中查找需求，并用结构化英语

重写它们。开发一个等价的决策表和（或）决策树，哪一个更易理解？为什么？

3. 复印一份你的学校成绩证明书。写一个数据定义来描述这些内容。再写一个数据元素定义来表示科目成绩，信用和学位。

4. 按照你在学校的完成方式定义图6-7中的处理2。使用任何处理分解和处理规格说明书的组合。如果你要开发处理分解DFD，别忘记定义所有的数据流。

实例研究

房地产多编目服务系统

参考第5章实例研究中的房地产多编目信息服务系统的描述。使用系统事件列表和ERD作为一个切入点做下列练习：

1. 画一个关联的DFD；
2. 画一个事件划分DFD；
3. 画所有的处理分解DFD。

国家巡查罚单处理系统

参考第5章实例研究的对国家巡查罚单处理系统的描述。使用系统事件列表和ERD作为一个切入点做下列练习：

1. 画一个关联DFD；
2. 画一个事件划分DFD；
3. 画所有的处理分解DFD；
4. 为在已有的系统描述中完全描述的数据流创建数据流定义。

落基山运动用品商店实例的再思考



本章中包含了许多描述RMO订单子系统的DFD，但是没有包含描述RMO订单实现了系统、客户维护了系统或目录维护子系统（参见图6-10中的子系统事件列表）的DFD。复习RMO的事件表（图5-15）和ERD（图5-28），并完成下列任务：

1. 为图6-12中没有记录的所有事件设计DFD片段。

2. 设计一个单个的DFD，显示对所有事件的处理，为每个子系统使用一个处理并给出所有必要的数据存储。对该图进行简化，将所有的外部实体放在外部边界上，必要的话，可复制它们，以便对它的成和交的数据流最小化。将所有的数据存储放在图的中间。

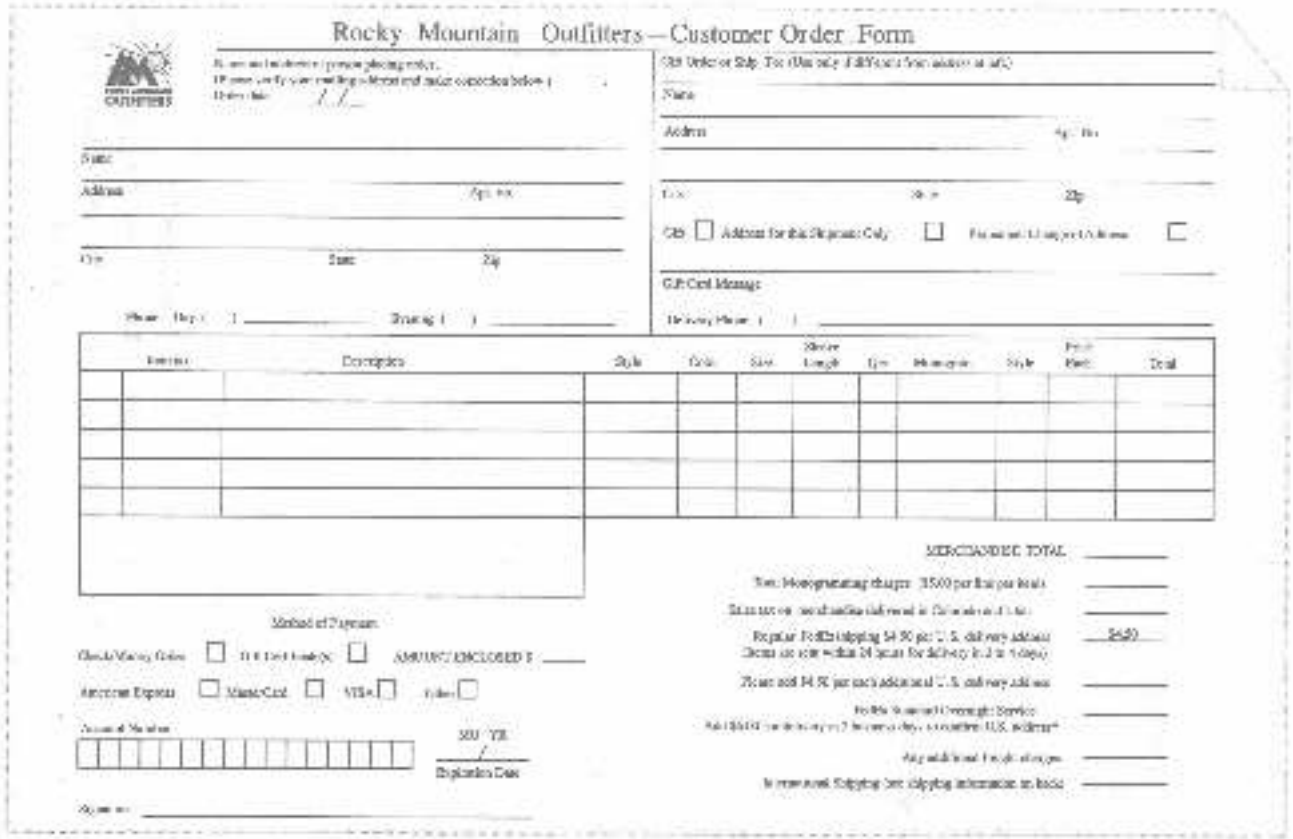
3. 为图6-40所示的RMO客户订单表设计一个数据流定义。

关注Reliable Pharmaceutical Services



通过下列任务继续完成对Reliable公司服务系统的建模工作：

1. 基于第1章的系统描述和第5章所开发的事件表，为该系统创建关联图；
2. 从事件表和ERD（在第5章中开发的）中为每一个事件创建DFD片段；
3. 通过组合问题2中建立的DFD片段，创建事件分割模型（0层图）；
4. 根据第1章的描述创建逻辑DFD，用于显示事件产生订单时间（发货）的处理细节，要切记建模数据的运动和处理，而不是物理货物（如药品）的运动和处理。创建为了全面确定系统需求所需的处理描述和数据定义；
5. 考虑在第1章描述的记账程序的建模问题。是否应该开发一个记账程序的物理DFD？为什么？



Rocky Mountain Outfitters—Customer Order Form

Name

Address

Apt. No.

City State Zip

Phone Day ( ) Evening ( )

Gift Order or Ship To (Use only if different from address at left)

Name

Address

Apt. No.

City State Zip

Gift ☐ Address for this Shipment Only ☐ Permanent Change of Address ☐

Gift Card Message

Delivery Phone ( )

Item No.	Description	Style	Color	Size	Sleeve Length	Qty	Monogram	Style	Print Block	Total

MERCHANDISE TOTAL

Total Monogramming charges $5.00 per line per item

Sales tax on merchandise delivered in Colorado and Utah

Regular Profile shipping $4.50 per U.S. delivery address $4.50
Items are sent within 24 hours for delivery in 2 to 4 days

Please add $4.50 per each additional U.S. delivery address

Profile Standard Overnight Service
Add $10.00 for delivery in 2 business days to continue U.S. address*

Any additional freight charges

Is now used Shipping (see shipping information on back)

Method of Payment

Check/Money Order ☐ Gift Card Enclosed ☐ AMOUNT ENCLOSED $

American Express ☐ MasterCard ☐ VISA ☐ Other ☐

Account Number

MO YR
Expiration Date

Signature

图6-40 RMO客户订单表

参考资料

J.Martin. *Information Engineering: Book I Introduction*. Prentice Hall, 1988.

J.Martin. *Information Engineering: Book II Planning and Analysis*. Prentice Hall, 1989.

Stephen M.McMenamin and John F.Palmer. *Essential Systems Analysis*. Yourdon Press, 1984.

G.A. Miller. "The magical number seven, plus or minus two: Some limits on our capacity for processing information." *Psychological Review*, Volume 63(19561,pp.81-97.

Edward Yourdon, *Modern Structured Analysis*. Yourdon Press.1989.

第7章　需求的面向对象描述方法

学习目标

阅读本章后，你应具备如下能力：

- 开发用例图
- 撰写用例和场景描述
- 开发活动图和顺序图
- 开发状态图为对象行为建模
- 解释如何用UML图表协同工作来为面向对象的方法定义功能需求

本章要点

- 面向对象的需求
- 系统活动——面向对象的用例/场景视角
- 确定输入和输出——系统顺序图
- 确定对象行为——状态图
- 面向对象模型的集成

无限电子公司：供应链一体化

无限电子公司是一家仓储式销售商，他们从不同的供应商手中买来电子设备，然后再卖给遍及整个美国和加拿大的零售商们。他们在洛杉矶、休斯顿、巴尔的摩、亚特兰大、纽约、丹佛和明尼阿波利斯都有办事处和仓库。他们的客户既包括像Target公司这样的全国范围的大型零售商，同时也有中等规模的独立的电子商店。

目前，许多大的零售商正致力于供应链一体化。信息系统过去只关注内部数据的处理，然而，如今的零售连锁希望他们的供应者成为完整的供应链系统的一部分。换句话说，信息系统现在必须在公司之间进行沟通，以使供应链更有效率。

为了保持它的批发销售商的领导地位，无限电子公司对其系统进行了调整，使之能够协调供应商（电子设备制造商）和用户（零售商）之间的关系。为了实现这个目标，他们利用面向对象技术开发了一个全新的系统。面向对象技术使系统与系统之间的接口连接变得容易，使用预先定义好的组件和对象将加快开发过程。幸运的是，许多系统开发人员已经开始学习面向对象的开发方法，并且他们热衷于为系统开发项目应用这种技术和模型。

William Jones正在给一批系统分析员讲解面向对象的开发（这些人是被安排来接受这种新方法的培训的）："我们将使用面向对象的原理开发大部分新系统。新系统的复杂性和它的交互功能使面向对象方法成为开发需求的自然之选。这与你过去的思维过程可能稍有不同，但是面向对象的模型和新的面向对象的程序语言十分相似。"

William继续说："从对象的角度来考虑一个系统是很有趣的，这也和你们在编程课上学到的面向对象的编程技术是一致的。开发用户界面时，你可能会首先学着去考虑对象。界面上的所有控件，如按钮、文本框和下拉框都是对象。每个对象都有自己的一系列的触发事件能够激活程序功能。"

"现在，你们只要将这种思维过程拓展开来，把像订单、雇员这样的事物都想象成对象，我们可以称之为问题域或业务对象以便把它们和诸如窗口、按钮这样的屏幕对象区分开。在分析过程中，我们要找到每个业务对象的全部触发事件和方法。"

"那我们怎样做呢？"一个分析员问到。

"继续你的事实发现活动并为每一个业务过程制作一个说明书。在说明书中的业务对象之间的交互方式决定了你是如何识别触发事件的。我们把这些触发事件看成在对象之间相互传递的消息。关键的技巧是你需要依据对象而不是过程来考虑。这样有时使我们假设自己就是一个对象。我会说'我是一个订单对象，其他的对象将会要求我有什么样的功能和服务呢？'一旦掌握了诀窍，以面向对象的角度工作，将会工作得很顺手，在开发图表时也很容易看清楚系统需求是如何展现的。"

概述

需求定义的基本目标在于理解：理解用户的需求、理解业务过程如何运行、并理解系统如何支持这些业务过程。正如在第2章中指出的一样，系统开发者使用一套工具和技术来发现和理解一个新系统的需求。这种行为是系统开发生命周期中系统分析阶段的重要组成部分。在面向对象的开发中，这类行为特指为面向对象的分析（OOA）。此过程首要的一步在于深入理解这一过程，需要用到第4章中关于事实发现的技术。事实发现行为也称为发现活动。显然，发现必须先于理解。在本章中，将介绍发现的下一个阶段——建立理解。

作为一种定义和记录系统需求的方法，第5章介绍了模型和建模活动的概念。第5章介绍模型的过程中，我们把注意力集中在系统需求的两个主要方面：包含在用户工作中的事件和事物。正如你所学到的，事件发生在系统必须响应的业务环境中。事件被定义和记录在事件表中，新系统必须能够通过运行系统活动（也称为用例）响应业务事件。

一个新系统同时也需要记录和存储包含在业务过程中的事物信息。在手工系统中，信息记录在纸上并存储到档案柜中。在自动化系统中，信息存储在电子文件或数据库中。系统的信息存储需求或者用传统方法中的实体-联系图（ERDs）进行记录，或者用面向对象方法中的类图进行记录。

在本章中，将介绍如何使用面向对象的分析模型和技术理解和定义新系统的需求。你应该理解：面向对象的分析和面向对象设计之间的界限并不明显，因为系统的设计就是对分析阶段中用于定义需求的模型进行改进和扩展得到的。回想一下，我们曾提到面向对象方法使用迭代的方法进行开发，这种方法首先定义一些需求，然后进行一些初步的设计并实施，然后反复迭代需求、设计和实施的过程。实际上，尽管我们不关心这里提到的需求定义的迭代特性，但它仍然是面向对象方法的组成部分。第11章把需求扩展到完整的面向对象的设计中，以便作为新系统程序设计的基础。

7.1 面向对象的需求

和第5章讨论的一样，使用模型来记录需求的一个最大的好处在于它能帮助系统开发人员仔细和清楚地考虑处理的细节及系统相关人员的信息需求。在阅读本章并进行相关练习的过程中，应该密切注意模型如何需要你来发现和理解用户需求。开发模型有很多好处。在整个建模的过程中面向对象的系统需求就确定并记录下来了。

本书所涉及的面向对象建模符号基于统一建模语言（UML）2.0版本。UML是被认可的标准的面向对象建模语言。UML标准由对象管理组织（OMG）持续下来，OMG是一个由800多个软件销售商、开发商和组织组成的共同体，他们致力于发展和传播面向对象系统。它成立

于1989年，OMG的使命就是在分布式计算系统的开发中提高应用对象技术的理论和实践水平。OMG维护和审核面向对象建模标准的任何变化。因此，UML标准能够得到持续发展，但是他们仍将从系统开发人员（和学生）的利益出发保留标准化的部分。2.0版本是最新的标准，这本书的模型是遵照它建立的。关于UML和OMG更多的详细信息请参见OMG的网站：http://www.omg.org。

如图7-1所示，系统开发过程开始于确定事件和事物。事件触发称为用例的基本业务过程，事物是包含在基本业务过程中的问题域对象。问题域对象在新系统的开发和数据库的设计中都是至关重要的。新开发人员经常会问这样的问题：是先定义用例和还是先定义对象的类。事实上，这两个方面是紧密联系在一起的。它们通常一起定义。有经验的开发人员经常在确定类和用例之间不断切换，在完成一套需求之前他们通常要进行几次这种操作。在定义需求时，如果发现需要改变图和模型，请不要气馁。

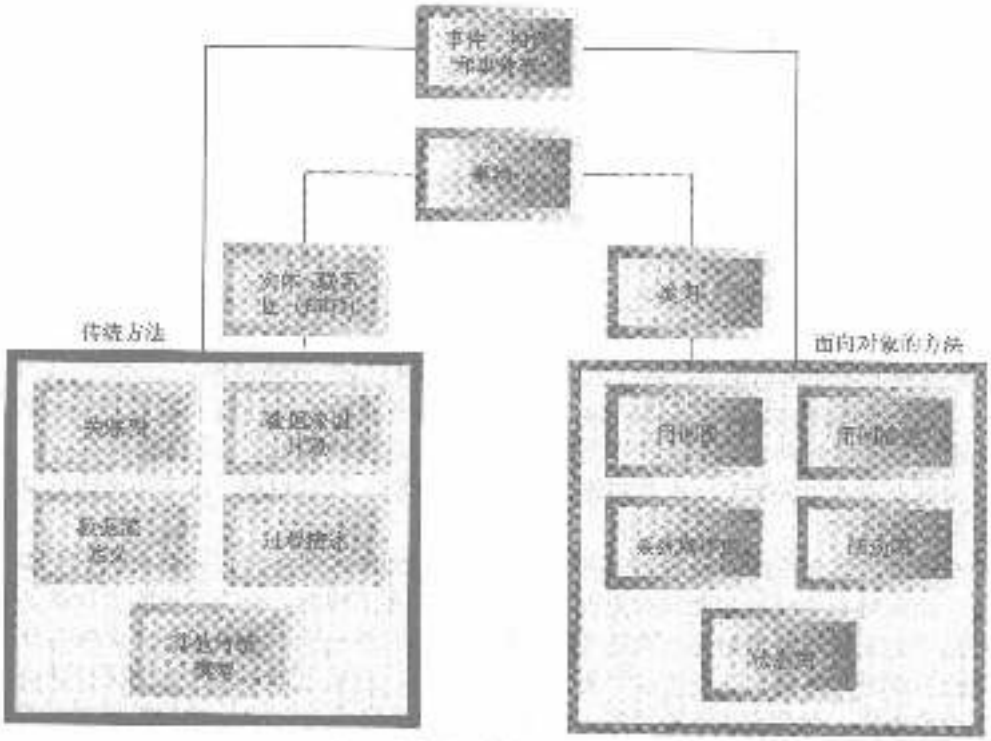


图7-1　传统和面向对象的需求图

面向对象的方法需要几个相关的模型来创建一套完整的说明。尽管在开始的时候因为有许多不同种类的图而显得有些复杂，但是在使用它们的过程中，你将会学到如何把它们像拼图一样组合在一起来生成完整的说明。特别地，面向对象的方法"区分和克服"了复杂系统中的一些问题。每个模型描述了系统的不同方面，所以你只要在一段时间内把注意力集中一个方面就可以了。但是你必须学会所有不同的模型和它们组合在一起的方式。在本章的最后，将讨论如何把所有的图结合在一起形成一个完整系统功能需求视图。作为一个刚开始学习UML的人员，你应该集中学习每一个新的模型，并理解在确定整个系统的过程中它所扮演的角色。

本章主要讨论一个关于模型的集合，它根据面向对象方法中的用例来捕获系统需求。四

种模型（用例图、用例描述、活动图（第4章讨论的）和系统顺序图）被用来从不同的观点描述系统用例。系统顺序图是交互图的一种。类图（第5章讨论的）是另外一种类型的图，它被用来定义问题域中对象的类。在许多实例中，分析员使用所有模型完整地定义系统需求。然而有时候只要使用两种或三种方法就可以准确的确定需求。

用例图的目的是识别新系统的“用法”或用例，换句话说，就是识别如何使用系统。用例图来源于事件表中标题为“用例”的列。用例图是用于记录系统活动的一种简便方法。有时可以用一个简单、易于理解的图来确定整个系统的用例。在其他情况下，可以用一些小型的用例图。

用例图：一种用以显示不同的用户角色和这些用户角色如何使用系统的图。

同时每个用例都必须详细描述。一种方法是详细记录下用户和系统共同完成用例的步骤。每个用例还可以用图表来定义。正如第4章中所介绍的，活动图可以用来描述由组织中的人完成的任何业务过程。另一方面，也可以用来描述包括手动和自动系统活动的过程。所以活动图可以用来定义用例。

系统顺序图（SSDs）用来定义一个用例的输入和输出，以及在用户和系统之间交互的顺序。它们用于联系详细描述或活动图。在顺序图中，这些进入系统的信息流被称为消息。还要标识用户和描述消息的细节信息。

系统顺序图：在用例或场景中，用于显示外部参与者和系统之间的消息顺序的图。

消息：用例内部对象之间的通信。

在第5章你学到了关于对象的类和类图的知识。类图用来标识真实世界的“事物”，这些事物决定了编写程序类的结构（及数据库结构）。在系统的面向对象视图中，每个事物都被认为是一个对象。在第5章，我们解释了所确定的对象属于问题域类并且这些类由具体的事物（如用户）和更抽象的事物（如订单和航线）组成。构造一个类图有助于确定真实世界对象的信息，这些真实世界的对象将是新系统的组成部分。

图7-1中所标识的另一种图是状态图。状态图表（简称状态图）描述了每个对象的状态的集合。类图中所标识的一些对象有些状态情形需要跟踪，这些状态直接决定了对象的处理过程。客户订单有几个比较重要的状态条件，它们控制订单的处理过程，举例来说：订单只有在完成后才会发货。状态图标识这些状态条件并标明允许的处理过程。同时，状态图也可用于设计过程，以确定系统本身的各种状态，并且能够处理可允许事件。因此，状态图既可以看做是分析工具，也可以看做是设计工具。

状态图：一种用以显示对象在各阶段中的生命和转换的情况的图。

7.2 系统活动——用例/场景视图

用例分析的目标是用来标识和定义系统必须支持的所有业务过程。分析员在综合等级和详细等级两个层次上定义用例。事件表和用例图为一个系统提供了所有用例的综合。关于每个用例的详细信息使用用例描述、活动图和系统顺序图，或者这些模型的组合进行描述。

7.2.1 用例和参与者

用例是系统运行的活动，通常由系统用户来响应需求。可以把用例看成是系统中的特定情形，此时系统必须完成一定的用户目标。例如，为忠RMO系统。RMO系统必须执行的一个处理是处理新客户订单。所以，该系统的一个用例就是产生新订单。注意，焦点在自动系统上，也就是说系统必须执行的创建订单的活动上。

所有的用例中都包含了使用系统的人。在UML中人被称为参与者。参与者通常处于自动

化系统边界之外，但是它也有可能是系统手动部分的成员。这里参与者的概念与我们在事件表中所定义的事件的源在内涵上略有不同，事件的源是指事件的发起者，例如一个用户，并且它通常在系统（包括手动系统）的外部。相反，在用例分析中的参与者实际上是亲自与计算机系统进行交互的人。通过这样的方法来定义参与者（即那些和系统进行交互的人），可以更加准确地定义那些自动化系统所必须响应的交互。这种紧密的关系有助于定义自动化系统自身的一些特殊需求，并且当我们从事件表中移出用例的详细信息时我们能够重新提炼它们。有一种方法能够确定参与者处于正确的等级，这就要假设参与者都有手。认为参与者都有手，将有助于将那些能够接触自动系统的人定义为参与者。但是要记住，一些参与者并不是人，他们也可能是其他的系统或者从系统接受服务的设备。

实践指导

确保参与者与自动系统有直接联系。

另一种考虑参与者的方式是把它看成一个角色。例如，在RMO案例中，用例产生新订单的情况可能会涉及一个销售代表在电话里与一名客户进行交谈。另一方面，如果这位客户直接在Internet上订货，那他也是个参与者。最后一种方式考虑参与者和用例，认为用例总参与者想要完成的目标。一种标志这一目标的方法是：订单职员使用系统来生成新订单。注意，这句话中的参与者（订单职员）和用例（产生新订单）都被标识了。事实上，这种用语句的格式声明用例的技术对于理解用例和参与者之间的关系非常有效。

7.2.2　用例图

图7-2所示为如何在一个用例图中记录一个用例。一个简单的棒状小人表示参与者。这个小人图取了一个可以表示其扮演角色的名字。用例用一个在里面标有名称的椭圆所代表。参与者与用例之间的连线表示了有哪些参与者参与哪种用例。虽然手不是标准UML符号的组成部分，但是在这个图中的参与者被画出了手，这样有助于读者掌握参与者必须能够直接访问自动系统。

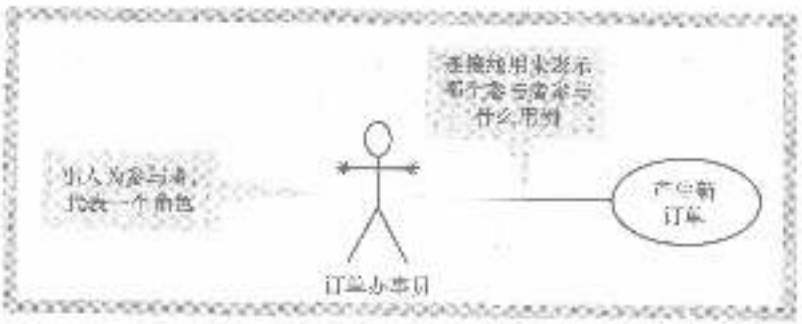


图7-2　有一个参与者的简单用例

用例图是概括有关参与者和用例信息的一个图形化的模型。为了对用例进行分析，我们把系统作为一个整体，并设法识别系统中所有主要的使用。

1. 自动化边界和组织

图7-3对图7-2进行了扩展，增加了一些参与者和用例。在这个实例中，订单员和用户都可以直接访问系统。正像有关系线连接表示的那样，每个参与者可以操作所有的用例。矩形用来表明一个参与者而不是一个人。在这个实例中，库存系统参与者可以调用用例查看可用的条目。在全套的用例上画了一条边界线。这个边界是自动化边界。它表示了环境（参与者的

居生地）与自动系统的内部功能之间的边界。

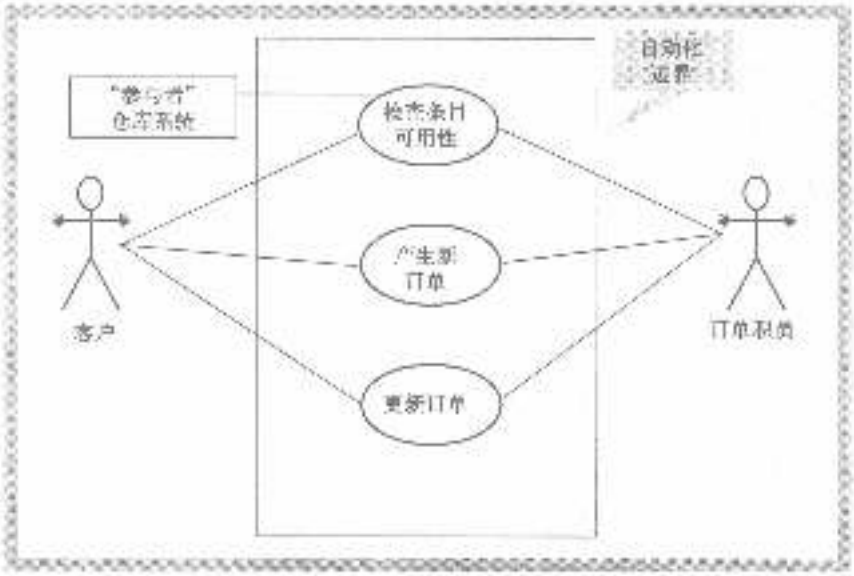


图7-3 RMO订单输入子系统的用例图（显示了系统边界）

有很多方法可以用来组织用例图，以便更易于理解和开发。一种方法是显示所有由特定参与者调用的用例。这来自于用户的观点。这种方法也经常用于需求定义，因为系统分析员可能与特定的用户一同工作并确定用户要求系统实现的所有功能。图7-4阐明了这一观点，显示了所有由客户参与者调用的用例。分析员可以扩展这种方法以包括属于一个特定部门的所有用例。在分析过程中，分析员关注于确定用户需求，所以从用户的观点组织用例是非常有用的。

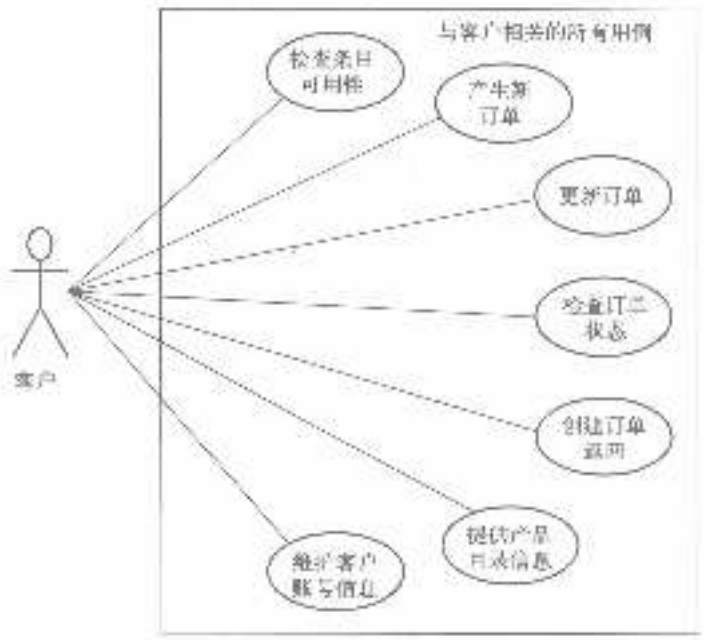


图7-4 与客户相关的所有用例

另一种方法是从系统及子系统的观点组织用例。有时这种类型的组织方式反映了用户部门，例如一次只能关注于账户或仓库操作，但是没必要这样做。系统开发者可能想通过系统的子系统来组织用例，以便对开发活动和小组任务进行分组。图7-5说明了这种方法，显示了许多由子系统组织的RMO用例。在这个图中，介绍了一个称为"包"的新符号。一个包将相似的组件分组在一起。包的符号是一个带标签的矩形，包的名字写在标签上。在图7-5中，包表明了子系统。这个图包含了4个单独的子系统和相应的用例，每一个子系统作为一个包。

包：用来表示一组相似元素的符号。

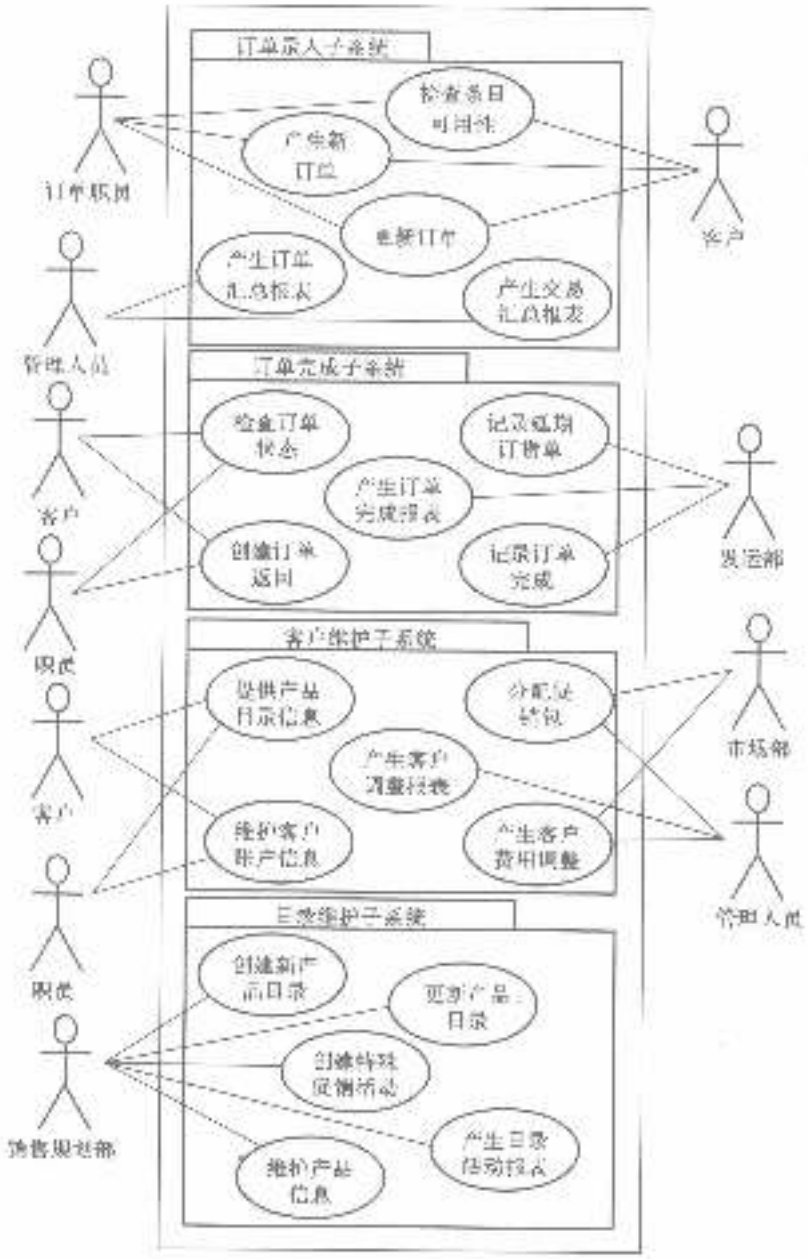


图7-5　客户支持系统的用例图（通过子系统）

2. <<包含>>关系

通常在用例图的开发过程中，一个用例需要用到通用子程序所提供的服务。例如，订单输入子系统的两个用例是产生新订单和更新订单，这两个用例均需要检查客户的账号是否正确，因此，可以定义一个通用的子程序来完成这个功能，并且它变成了一个附加用例。图7-6所示为一个名为核查客户账号的用例，它被刚才的两个用例调用。在这些用例之间的关系由带箭头的连接线表示。箭头的方向表明那一个用例作为主要用例的一部分被包括进来。这种关系可以读做产生新订单<<包含>>核查客户账号。有时这种<<包含>>关系也称为<<包括>>关系或<<使用>>关系。

图7-6也表明了查找可用的条目可能是<<包含>>关系的一部分。所以，分析员可以定义两种类型的<<包含>>用例：一种是通用内部子程序，如核查客户账号，它不被外部参与者直接引用；另外一种是能被外部参与者直接引用的，如查询可用的条目。

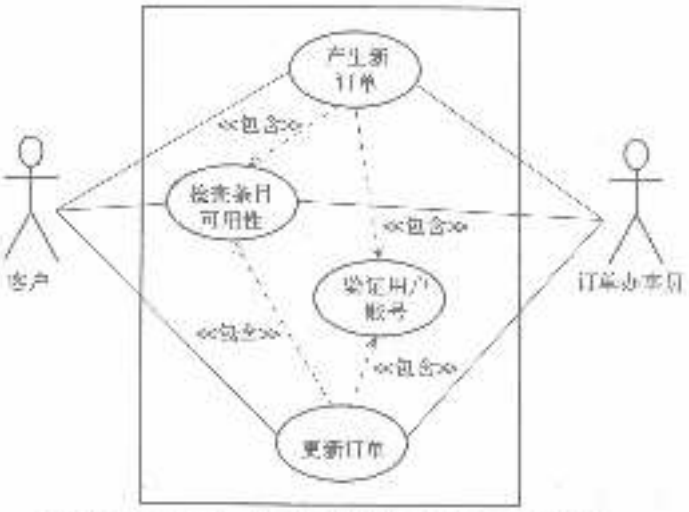


图7-6 有<<包含>>关系用例的订单输入子系统的例子

3. 用例图与事件表的比较

先前已经说明了用例图和事件表包含了许多同样的信息，事件表确实是一个关于所有用例信息的目录。你或许会问自己一个问题："如果它们是一样的，那么我需要开发两个模型吗？"事实上，给出的项目中，你都不必开发两个模型。一些分析员更倾向于以列举用例作为开始，而不是以事件作为开始，并且分析员会直接建立用例图（正如下一节将要讨论的）。事件表可用作传统结构化开发的基础，也可用作面向对象开发的基础。

然而，在两个模型之间存在着差异。首先，每一个模型的观点都有些细微的不同。事件表通常注意业务过程。它通过标识业务事件及这些事件的外部、初始化源的信息来关注业务过程。外部的源是引起业务事件初始化的原因，并且它们能从自动系统中轻松的移除。另一方面，用例图强调了自动系统。因为它只与自动系统相连，所以参与者与自动系统有联系，并且不一定是业务事件的发起者。

两种模型之间另一点差别体现在标识临时事件和状态事件时。由于用例通常被外部参与者初始化，所以如果分析员不仔细标识每一个事件，那么临时事件和状态事件经常被忽略。如果用例定义得太窄，那么这将成为用例建模的一个缺陷。如第13章中所讨论的那样，在线系统菜单常常包括用于表示事件表中临时事件的菜单选项，以便这样的事件能够被用户触发并作为纯粹的临时事件。因此，建议为每个临时事件和状态事件创建用例以确保这些需求不被忽略。

需要记住很重要的一点：分析员将同时完成事件表和用例图。分析员将不断更新事件和用例。这种精炼的过程发生在调整每个用例范围的平衡中。例如，在开发事件表的过程中，添加新用户和更新用户信息两个事件已经被标识出来。从系统的观点出发，这两个业务事件的用例几乎是一样的，因为它们都包含了更新用户文件。可以定义一个单独的用例来支持这两个业务事件，这个用例可以命名为维护客户账户信息。如果满足如下三个标准，则定义一个用例来支持多个业务事件是很常见的：首先，本质上相同的处理发生在自动系统内部；第二，本质上相同的信息被更新；第三，本质上相同的信息从事件中输入和输出。在业务事件对单一、简单的数据文件或表进行基本的文件维护时，这些条件常常能够满足。有时单一事件可以触发非常复杂的处理需求，这使得将系统活动分解为两个用例来更好的管理系统复杂性变得更加有意义。在所有的这些情况下，必须修改事件表和用例以使模型保持同步。

7.2.3　开发用例图

如前所述，开发用例图有两个切入点。如果分析员分析了业务过程并创建了事件表，那么他（她）就会用事件表标识用例。仔细分析表中的每一个事件以确定系统为了支持这个事件、发起这个事件的参与者以及由于这个事件而触发的其他用例所执行的处理。当从一个模型转向另一个更详细的模型的时候通常需要不断的精炼。所以仔细分析事件和事件表是非常重要的。在额外的分析过后，开发人员可以将一个单一的事件标识为用例，如果所需的处理很相似，可以把几个事件组合成一个单独的用例，或者如果处理很复杂，也可以标识多个用例。多个用例的标识通常发生在它们有《包含》关系和两个用例从一个大型用例分解得到的时候，或者发生在一个附加用例按照通用子程序的方式被定义的时候。前面已经讨论过这种情况了。

如图7-5所示的客户支持子系统就是使用这种方法进行开发的。你会注意到，在图中定义的绝大多数用例直接来自于图5-16的事件表。事实上，图7-5中用例的名称来自于事件表的用例栏中的描述。这一部分有两个例外。由于临时事件通常可以手动发起，所以要为每个临时用例使用标识外部参与者的选项。另一个例外是事件号为13的客户修改账户信息事件。在这个实例中，用例定义扩展到与所有维护客户信息相关的场景中。同样，这个用例被命名为维护客户账户信息，这指的是添加、更新和删除。这些都是用例图对事件表进行精炼更新的例子。

如果没有创建事件表，那么开发用例图的另一个切入点是标识使用系统的参与者和它们执行的功能。为了做到这一点，必须记住两个前提条件。首先，为一个自动系统创建系统边界，这样你标识的参与者就能和这个系统进行联系（通过手）。第二，必须假定拥有完美的技术，确定用例总基于业务事件而不是像登录系统一样的技术活动。只有给出这些前提，才可以通过以下两步的迭代来开发用例图。

① 标识系统的参与者。注意参与者通常由用户扮演，不能把参与者列成如Bob、Mary或Hendricks先生这样的形式，而应该标识出这些人所处的特定角色。记住，在一个系统中同一个人可以有多个特定角色。这些角色可以是：订单职员、部门经理、审计员等。理解和辨识系统所有可能使用的角色是重要的。其他的系统也可以成为系统的参与者，如图7-3所示。

② 一旦确定了参与者的角色，下一步就要开发在使用自动系统中这些角色的目标列表。目标指参与者执行完成一些业务功能的任务。目标通常是类似营销、接收用户反馈或订单发货这样的任务。这些目标是能够被标识和描述的工作单元。在目标完成时，系统数据在一段时间内是不会改变的。

这两个步骤常常应用于项目组成员和用户的集体讨论中。并没什么捷径可以用来发现或标识用例。即使所关注的是自动系统，还是需要对业务过程细致的分析，以理解参与者使用系统的所有方式。

在标识事件表中的事件或直接开发用例图的时候，还会用到另外一种很重要的技术，称为CRUD分析，这种分析方法将标识后的用例与域模型类图进行比较。分析员在初始化用例图之后作CRUD分析，这主要是为了复查他们的工作。CRUD代表创建（Create）、读取（Read或报告Report）、更新（Update）和删除（Delete）。CRUD分析在第6章里首次被提及（见图6-39），它是一种与信息工程（IE）紧密联系的技术。CRUD分析需要类图中的每个类都有足够的用例来支持创建新对象实例、读取或报告这些对象、更新这些对象并在许多例子中删除对象实例。用例不应该命名为创建或者更新，但是潜在的处理可以添加新的实例或者更新上存在的实例。例如，一个名为记录支付情况的用例并没有清楚地指出一个新的支付对象被创建，但是用例的详细描述却指出这个对象被创建了。用例产生新订单可以创建订单条目对象和更新库存条目对象。在其他的案例中，许多用例以维护两字开头命名，以覆盖常规的添加、更新、读取和删除操作。

为了做CRUD分析，只需要看一下在域模型类图中的每个类并确定在用例图中有用例来支持所有适合该应用的CRUD功能。但是，切记在集成系统中，一个系统可能负责创建对象，而另外的系统可以只更新它们。CRUD分析方法提供了一种交叉检验方法，不是最终的解决方案，并且提供了确认重要的系统集成需求的机会，这些需求如果不用这种方法进行分析可能就不明显。

7.2.4 用例详细描述

前面已经指出，创建用例图只是用例分析的一个组成部分。用例图帮助标识各种处理，这些处理由用户执行，并且要得到新系统的支持。精细的系统开发往往需要我们了解较底层的细节。为了创建易于理解的、鲁棒的并且满足用户需要的系统，我们必须理解所有详细描述的步骤。内部，为了完成业务过程，用例要包括完整的步骤顺序。例如，通常业务步骤的变化存在于单一用例中。产生新订单用例，根据哪些参与者调用用例，将有不同的活动流。订单职员通过电话建立新订单的过程与用户通过Internet建立订单的过程是非常不同的。对产生新订单用例，每个活动流都是一个有效的顺序。这些不同的活动流称为场景，或者有时也称为用例实例。所以，场景是在一个用例中的一个内部活动，它代表通过用例的唯一路径。

场景或用例实例：用例中步骤的一个特定顺序。一个用例可以有几个不同的场景。

用例可以使用各种图表和描述来精心制作。一个比较有用的记录用例的绘图技术是活动图（在以后讨论）。活动图开始是在第4章介绍的。许多分析员更喜欢写出叙述性的用例描述，这样就可以根据需求在不同的细节层次上进行描述。通常可以按三个独立的详细描述等级进行用例描述：简单描述、中间描述和完全展开描述。根据分析员的需求不同，书写描述和活动图可以用于任何组合。

1. 简单描述

简单描述被用于非常简单的用例，特别是当要开发的系统是一个很小且易于理解的应用时。一个简单的用例通常有单一的场景和即使有也很少的异常条件。用于与活动图进行连接的简单描述为简单用例提供了可靠的描述。图7-7提供了产生新订单用例的简单描述。通常，像产生新订单这样的用例是很复杂的，它们可以用中间或者完全展开描述进行描述。接下来将说明这些描述。

产生新订单描述

当用户电话订购时，订单职员和系统会检验客户信息，创建一个新订单，将各条目加入订单中，当收支付款项，创建这个订单交易，最后完成订单

图7-7 产生新订单用例的简单描述

2. 中间描述

中间等级的用例描述扩展了简单描述，它包括了用例的内部活动流。如果有多个场景，那么其中的每个活动流都被单独进行描述。如果它们需要，还可以记录下其他条件。图7-8和7-9所示为记录两个场景的中间描述，这两个场景分别是订单职员创建电话订单和客户建立网上订单。这两个场景先前是作为产生新订单用例的工作流而被标识的。注意，每个场景都描述了用户和系统所需要执行的处理；同时还列出了异常条件，每一步都进行了标号，以方便阅读。在许多方法中，这种描述都是结构化英语，它包括了顺序、决策和循环体。

订单职员创建电话订单场景的活动流
主要流程：
1. 客户致电RMO，描述订单职员
2. 订单职员检验客户信息，如果是新客户，则用维护客户账户信息用例来增加新客户
3. 职员开始创建一个新订单
4. 客户要求将新条目加入订单
5. 职员检验该条目，把它加入订单中
6. 重复第4步和第5步，直到所有条目都加入订单中
7. 客户指示订单结束，职员输入订单结束，系统计算总数
8. 客户提交款项，职员输入数值，系统检验付款项
9. 系统完成订单
异常情况：
1. 如果一个条目没有现货，客户可以
a. 选择不购买该条目
b. 将该条目作为延期交货条目添加
2. 如果由于信用卡无效，客户所付款项被拒绝，那么
a. 取消订单
b. 订单挂起，直到收到支票

图7-8　产生新订单电话订购场景的中间描述

客户创建Web订单场景活动流
主要流程：
1. 客户访问RMO主页，并链接到订单页面
2. 如果是新客户，客户链接到客户账号页面，填写适当的情况创建客户账号
2a. 如果客户已存在，客户登录
3. 系统开始一个新订单，显示目录结构
4. 客户搜索目录
5. 当客户找到正确条目，他/她请求将其加入订单，系统将其加入购物车
6. 客户重复第4步和第5步
7. 客户要求结束订单，系统显示订购条目的汇总
8. 客户进行任何更改
9. 客户请求进入支付界面，系统显示支付界面
9a. 客户输入付款信息，系统显示汇总信息并发送确认邮件
10. 系统提交订单
异常情况：
1. 如果系统已有客户忘记密码，那么
a. 客户可以调用忘记密码处理程序
b. 客户创建一个新的客户账号
2. 如果由于信用卡无效，客户所付款项被拒绝，那么
a. 取消订单
b. 订单挂起，直到收到支票

图7-9　产生新订单Web订购场景的中间描述

3. 完全展开描述

完全展开描述是记录用例的最正式的方法。虽然它要花费较多的工作在这个层次上定义所有的细节，但是它仍然是描述用例内部活动流的首选方法。软件开发人员面临的一个主要困难是开发人员要不断理解用户的需要。但是如果创建了一个完全展开用例描述，那么就更有可能完全理解业务过程和系统支持它们的方式。图7-10所示为产生新订单用例的电话订购场景的完全展开用例描述。图7-11所示为同一用例的Web订购场景。

用例名称	产生新订单	
场景	创建新的电话订单	
触发事件	客户致电MO，请求目录中某条目	
简单描述	当客户电话订购时，订单助理使用系统查找的客户信息，创建一个新订单，将条目加入订单中，检验支付款项，创建这个订单交易，最后完成订单	
参与者	电话销售职员	
相关用例	包含：检验条目可用性	
系统相关者	销售部：提供主要定义 运输部：检验信息内容是否足够满足要求 市场部：收集客户统计资料研究购买模式	
前提条件	客户必须存在 目录、产品以及库存条目对于需求必须存在	
后续条件	订单和订单特殊条目必须创建 对于订单支付必须创建订单交易 手头的库存条目数量必须实时更新 订单必须与某个客户相关联	
主体流	参与者	系统
	1. 销售职员接听电话，与客户建立连接 2. 职员检验客户信息 3. 职员创建一个新订单 4. 客户要求在订单中加入条目 5. 职员检验条目（“检验条目可用性”用例） 6. 职员将条目加入订单 7. 重复4、5、6，直到所有条目都加入订单中 8. 客户指示订单结束，职员输入订单结束 9. 客户提交款项，职员输入数值	3.1 创建新订单 3.1 显示条目信息 6.1 添加创建订单条目 8.1完成订单 8.2 计算总数 9.1 检验所有款项 9.2 创建订单交易 9.3 提交订单
异常情况	2.1 如果客户不存在，职员暂停该用例，调用维护客户信息用例 2.2 如果客户有信用卡，职员请该客户描述客户服务代表 4.1 如果条目没有现货，客户可以 a. 选择不购买该条目 b. 将该条目作为延期交货条目添加 9.1 如果由于信用卡无效，客户所付款项被拒绝，那么 a. 取消订单 b. 订单挂起，直到收到支票	

表7-10 产生新订单电话订购场景的完全展开描述

用例名称	产生新订单	
场景	创建新的Web订单	
触发事件	客户登录RMO网站，请求购买条目	
简单描述	客户登录，请求新的订单表格，客户开始搜索目录，采购目录中的条目。系统将采购的条目加入订单中，最后客户输入信用卡信息	
参与者	客户	
相关用例	包括：注册新客户，检验条目可用性	
系统相关者	销售部：提供订单定义 运输部：检验信息内容是否足够满足要求 市场部：收集客户统计资料研究购买模式	
前提条件	目录、产品以及库存条目对于需求项必须存在	
后续条件	订单和订单排列条目必须创建 对于订单支付必须创建订单交易 干头的库存条目数量必须实时更新 订单必须与某个客户相关联	
事件流	参与者	系统
	1. 客户访问RMO主页，并就找到订单页面 2. 如果是新客户，客户链接到客户账号页面，填写适当的信息创建客户账号 2a. 如果客户已存在，客户登录 3. 客户搜索目录 4. 当客户找到正确的条目，把/她错误特定加入订单 5. 重复第3步和第4步 6. 客户要求结束订单 7. 客户进行任何更改 8. 客户请求进入支付界面 9. 客户输入付款信息	2.1 创建新的客户记录 2a.1 验证客户账号 2.2 创建一个新的购物车订单，用目录结构链入订单表格 3.1 根据搜索项和选项显示目录中的产品 4.1 将条目加入购物车订单 6.1 显示购物车条目，包括总数和到期金额，以及确认和提交按钮 8.1 显示支付账户条目的新信息页面 9.1 接收支付，提交订单，发送确认邮件
异常情况	4.1如果有条目没有现货，客户可以 a. 选择不购买该条目 b. 将该条目作为延期交货条目添加 8.1 如果由于信用卡无效，客户的付款请求被拒绝，那么 a.取消订单 b.订单挂起，直到收到支票	

图7-11　产生新订单的网上订购场景的完全展开描述

图7-10和7-11也可以记录其他场景和用例的完全展开描述的标准模板。第1行到第2行用于标识所记录用例的内部用例和场景。在较大或较正式的项目中，还可以给用例添加带有扩展名的唯一标识符，以标识特定的场景。有时候还会加上制作该表格的系统开发人员的姓名。

第3行标识了发起该用例的触发器。这个触发器与第5章介绍的事件表中的描述完全一样。可以从两个角度描述触发器。一方面是标识触发处理的业务事件。例如，在图7-10中，产生新订单的过程开始于客户致电给RMO。这个角度是以外部世界为中心。另一个角度是引起自动系统首先认识到用例已经开始运作的活动。从第二个角度来看，触发器可以描述为“职员

输入新订单请求"。由于新系统开发过程中该阶段的目标是理解用户需要，因此较好的角度是第一个——标识触发整个处理的外部事件。

第4行是用例和场景的简单描述。分析员只需复制他们先前建立的简单描述。第5行标识了一个或多个参与者。这一行复制了用例图本身所包含的一些信息。第6行标识了其他用例和它们与该用例相关联的方式，例如，我们前面讨论的<<包含>>关系。另外，还可以确定一些更复杂的关系。但重要的一点是，给其他用例添加交叉引用，以更全面地理解用户需求的各个方面。

系统相关者一行标识了相关的人员，而不是特定的参与者。他们可以是那些没有调用用例但是对用例的结果感兴趣的用户。例如，在图7-10和7-11中，没有人在市场部创建新订单，但是他们要对输入的订单做统计分析。所以，市场部人员对获取和存储在产生新订单用例中的数据非常感兴趣。对系统开发人员来说，考虑所有的系统相关者是确保他们能充分理解系统需求的至关重要的一步。

下面两行提供了用例执行前后系统状态的临界信息，分别称为前提条件和后续条件。前提条件表明在用例开始之前什么条件必须为真，换句话说，它标识了用例开始执行前系统的状态，包括，必须已经存在什么样的对象，哪些信息必须可用，甚至用例开始之前参与者是什么样的状况等。

前提条件：在用例初始化之前必须为真的一组条件。

后续条件标识了在用例结束的时候什么必须为真。描述前提条件的同样条款也适用于描述后续条件。例如，在更新各种财务账目的用例处理中，一些账目是收支不平衡的，所以，这种用例的后续条件就是更新所有账目并使收支平衡。

后续条件：在用例执行完成时必须为真的一组条件。

实践指导

前提条件和后续条件对于理解用例的处理是很重要的。 ■

模板中的最后两行描述了用例活动流的详细信息。在这个例子中采用了两列说明的形式，这些说明标识了参与者运行的步骤和系统需要的响应。项目编号有助于标识步骤的顺序。有些开发人员更喜欢用一列说明，与中间等级类似。最后一行描述了可选活动和异常情况。异常情况的编号同样有助于将异常与用例描述中的特定步骤联系在一起。

4. 活动图描述

记录用例场景的另外一种方式是使用活动图。第4章中已经介绍了作为工作流图的一种形式的活动图，介绍了活动图用于记录业务过程工作流，非常易于理解。活动图是一个标准的UML图。在这个实例中，活动图将作为一种有效的技术用于记录每个用例场景的活动流。

图7-12和7-13是记录了图7-10和7-11两个场景的活动图，在图7-12中客户和订单职员交互，轮流使用系统。由于用例的目的是说明参与者(带有手)和系统之间的相互作用，所以图中包含了订单职员和计算机系统的活动图矩形区。然而，为了帮助理解场景的所有活动流，触发这一步骤的客户也要包括进来。注意，在图7-12中客户泳形区是一个可选的附加项，它有助于理解整个工作流。我们也可以看到同步线的使用，它用在这个图中是为了定义重复部分的结束点，也就是一个循环。在图7-13中用户是与计算机系统相互作用的参与者，所以只需要两个泳形区来描述场景中的步骤。

活动图可以用来支持任何等级的用例描述，和你看到的一样，活动图和完全展开描述中的两列描述非常相似。创建活动图的好处在于它更加形象，并且使得理解整个活动流更加容易。这两个例子只显示了主要的活动流，没有异常情况。异常情况也可以通过增添更多的活

动椭圆来显示。工作流的早结束可以通过一个指向意外结束活动的出口箭头来表示。描述意外结束活动与描述正常结束活动很类似，只是圆形里面的是一个大X而不是一个黑点。

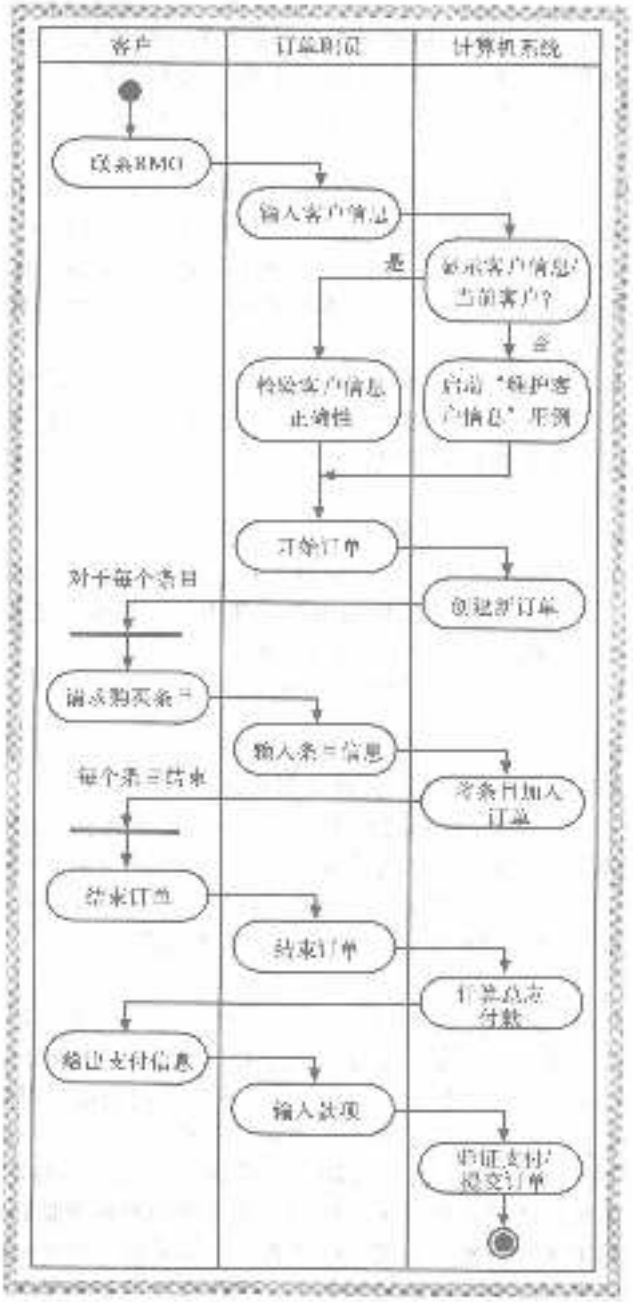


图7-12　电话订购场景的活动图

浏览图7-12和图7-13，产生新订单用例的两个场景截然不同。尽管这些场景执行同样的基本功能，但是界面和选项集合完全不同。下一节将继续介绍，活动图还有助于开发系统顺序图。

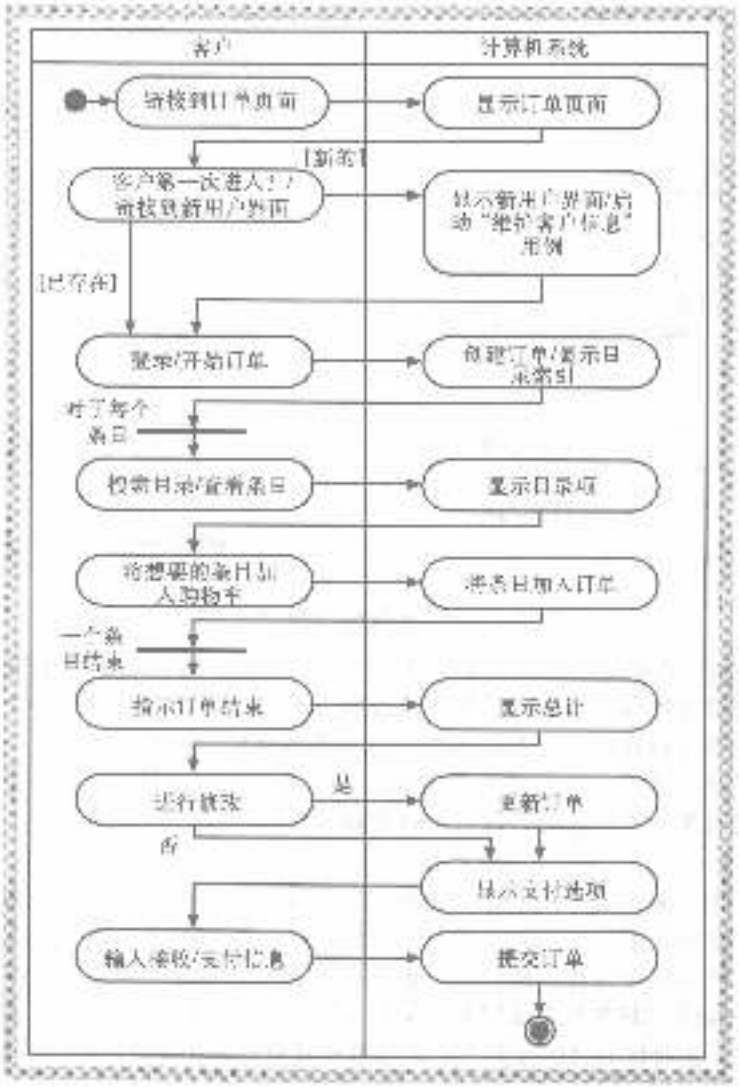


图7-13 Web订购场景的活动图

7.3 确定输入和输出——系统顺序图

在面向对象方法中，信息流是通过向参与者或内部对象来回发送消息形成的。系统顺序图（SSD）用于描述进出自动系统的信息流，所以一个系统顺序图记录输入和输出并标识了参与者和系统之间的交互。系统顺序图是交互图的一种，在接下来的章节及实际的工业实践中，我们常常互换地使用术语交互和消息。

交互图：用于显示对象间的相互作用的协作图或顺序图。

7.3.1 系统顺序图符号

图7-14所示为一个普通的系统顺序图。与用例图一样，用线条画代表和系统交互的参与者——人（或角色）。在用例图中，参与者"使用"系统，但是在系统顺序图中参与者如何通过输入数据和获得输出数据来和系统进行交互才是重点。这两个图有着相同的思想，不同的

描述等级。



图7-14　系统顺序图的例子

标记为：系统的方框是一个代表整个自动系统的对象。在系统顺序图和所有的交互图中，分析员使用对象符号代替类符号。对象符号表明方框指的是一个独立的对象而不是所有类似对象的类。这个符号很简单，它由带下划线的对象名称和一个矩形框组成。带下划线的对象名称前面的冒号是可选的，但习惯上常常使用，它是对象符号的一部分。在一个交互图中，消息通过独立的对象（而不是类）进行发送和接收，在一个系统顺序图中，只有代表整个系统的对象才被包括进来。

处在参与者和：系统下面竖直的虚线称为生命线。在整个系统顺序图期间内，生命线或对象生命线仅仅是这个对象（参与者或对象）的扩展。在代表消息的生命线之间的箭头代表了参与者和系统的发送和接收。每个箭头都有一个起点和终点。消息的起点就是箭头尾部的生命线所指示的发送它的参与者或对象。同样地，消息的目标参与者或对象是由箭头所指向的生命线指示的。生命线的目的就是指示参与者和对象发送和接收消息的顺序。在这个图中，消息的读取顺序是从顶部到底部的。

生命线或对象生命线：在顺序图中的一个对象下面的竖线，用以显示这个对象的时间阶段。

标有记号的消息用来描述消息的目的以及被发送的任何输入数据。消息标记的语法有几个选项。图7-14显示了最简单的形式。记住，箭头用来代表消息和输入数据。但是术语消息在这里到底是什么意思呢？在顺序图中，消息被认为是在目的对象上调用的一种活动，它更像一条命令语句。注意，在图7-14中，输入的消息被称为inquireOnItem。订单职员向系统发送请求或消息，以找到条目。与消息一起发送的输入数据被括在圆括号中，并且这个例子中它是一种用来确定特定条目的数据。语法仅仅是带有用括号括起来的输入参数的消息名。这种语法形式常常附加在实线箭头上。

返回消息在格式和意义有一些微小的差别。注意，箭头是虚线的。如图中所示，虚线箭头用来指明一个响应或应答，它们通常紧跟在启动消息的后面。标记的格式也是不同的。由于它是一个响应，所以只有响应中发送的数据才被标明。这里没有请求服务的消息，只有正在返回的数据。在本例中，一个有效的响应可以是所有返回消息的列表。例如，条目的描述、价格和数量。然而，细写的表示方法同样是令人满意的。在这个例子中，返回消息被称做条

日信息。同时需要另外的记录来显示详细内容。在图7-14中，这种另外的信息显示为注释。任何UML图中都可以添加注释，这主要是为了使下解释。条目信息的细节信息还可以记录在支持性说明中，甚至只是被客户类中的属性进行引用。

通常，同样的消息可以被发送多次。例如，当向一个订单中输入一个条目的时候，就要多次发送为一个订单增加条目的消息。图7-15a举例说明了显示这种重复操作的符号。消息和它的返回值放在一个较大的矩形框中。大矩形框顶部的小矩形框是一些描述性的文本，用以控制大矩形框内部消息的行为。所有条目的循环这个条件表明框内的消息重复多次或与多个实例关联。

图7-15b显示了一个备用符号。方括号和其中的文本被称做矩形框中消息的真/假条件。真/假条件前面的星号（*）表明只要真/假条件的值为真消息就重复。分析员使用这种缩写符号有几个原因。首先，消息和返回数据可以显示在一个步骤中，注意，返回数据作为返回值标识在赋值操作符（:=）的左边。这一变化显示了一个返回值。第二，真/假条件放置在消息中。注意，在这个例子中，真/假条件用于控制循环，同时，真/假条件也用于评估任何形式的测试，以决定消息是否被发送。例如，[信用卡支付]可以用来控制是否向系统发送了进行信用卡号码验证的消息。最后，消息上也有星号，所以，对于简单的重复消息，符号可以变得更短。然而，如果几条消息一同被重复执行或者有多个消息，每一个都有自己的真/假条件，则越详细的符号就越清晰准确。

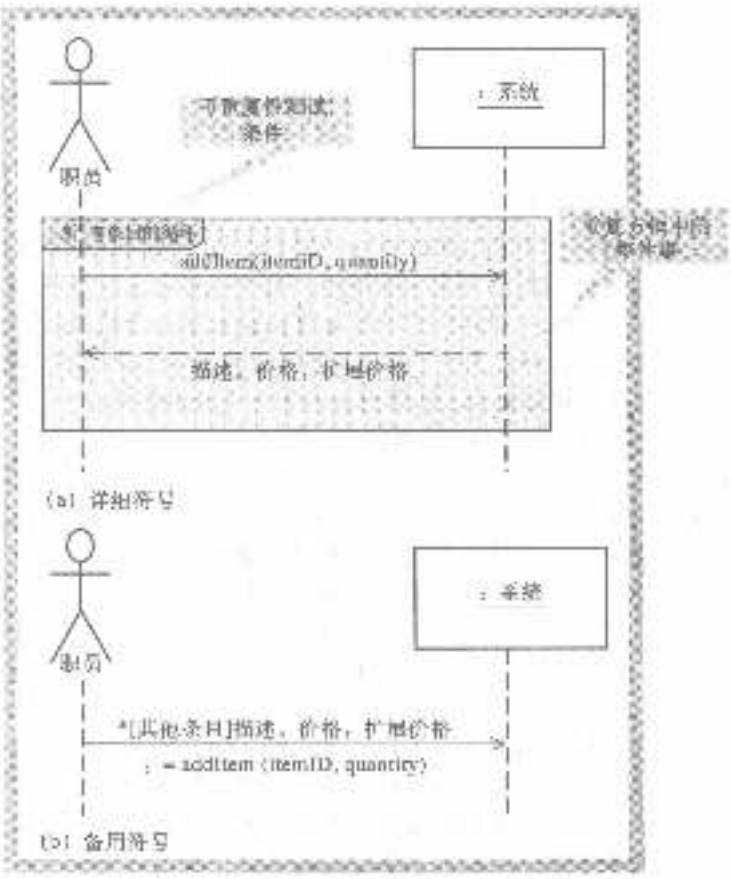


图7-15 重复消息

一条消息的完整符号如下所示：

[真/假条件]返回值：=消息名（参数列表）

消息的任何部分都可以被省略。简单来讲，符号的组成说明如下：

- 星号（*）表示消息的重复或者循环。
- 方括号[ ]表示真/假条件。它只是对消息的检测。如果它的值为真，消息就被发送，否则消息就不发送。
- 消息名是对所需服务的描述，在虚线返回消息时，它被省略，而仅显示返回的数据参数。
- 参数列表（带有圆括号表示启动消息，没有圆括号表示返回消息）显示了消息传递的数据。
- 与消息处于同一线上（需要：=）的返回值是用于描述从目的对象返回到源对象的消息响应数据。

真/假条件：对象之间消息的一部分，通过在消息发送前计算其值来决定消息是否可以发送。

实践指导

细心并准确地开发系统顺序图。在进行细节设计和用户接口设计时将称为临界组建。 ■

7.3.2 开发系统顺序图

系统顺序图通常用来与用例描述相联系，以帮助记录用例中一个单独的用例或场景的详细信息。为了开发一个系统顺序图，你需要有用例的详细描述，可以是如图7-10和图7-11所示的完全展开形式，也可以是如图7-12和图7-13所示的活动图。这两个模型标识了用例中的一系列活动，但是它们并没有明确的标识输入和输出。系统顺序图将提供输入输出的详细说明。使用活动图的一个优点就是很容易确定输入或输出何时发生。输入和输出总是发生在活动图中的箭头从外部参与者到计算机系统这一阶段。图7-16是图7-12（RMO产生新订单用例的电话订购场景）的简化版本。显然，简化版本缺了很多东西，但是它使我们可以不需要考虑真实世界的所有复杂性，而更多的关注处理，关注系统顺序图开发最基本的东西。

在简化的活动图中，有三个重要的组成部分：客户、订单职员和计算机系统。在开始顺序图之前，必须首先确定系统边界。在这个实例中，系统边界就是订单职员矩形区和计算机系统矩形区之间的竖线部分。由于顺序图的目的就是描述自动计算机系统的输入和输出，所以顺序图将只包括订单职员和计算机系统。在顺序图中将两个参与者都包含进来也是正确的，但只包含系统和其中发送输入和接收输出的参与者将使得重点更加突出。

基于活动图开发顺序图可被分为以下四步：

1. **标识输入消息**。图7-16中，有三个地方的工作流穿过了订单职员和系统之间的边界线。在每一个工作流穿过自动化边界的地方，都需要输入数据，因此同样需要消息。

2. **用先前介绍的消息符号来描述从外部参与者到系统的消息**。在许多例子中，你都需要描述系统所需服务的消息名和需要传递的输入参数。图7-17所示为产生新订单用例的系统顺序图，说明了三个消息。注意，消息名要反映参与者向系统请求的服务，如startOrder、addItem及completeOrder。也可以使用其他的名字，如可以用enterItemInformation代替addItem。

所需要的其他信息还包括每条消息的参数列表。要准确的确定哪些数据条目必须传递进来是比较困难的。事实上，开发人员会发现要确定数据参数通常需要数次的迭代，才能找到一个正确而又完整的列表。一个很重要的规则就是根据类图来确定数据参数。换句话说，类中一些恰当的属性可以用做参数列表，仔细查看这些属性，加上对系统需求的充分理解将有助于你找到正确的属性。

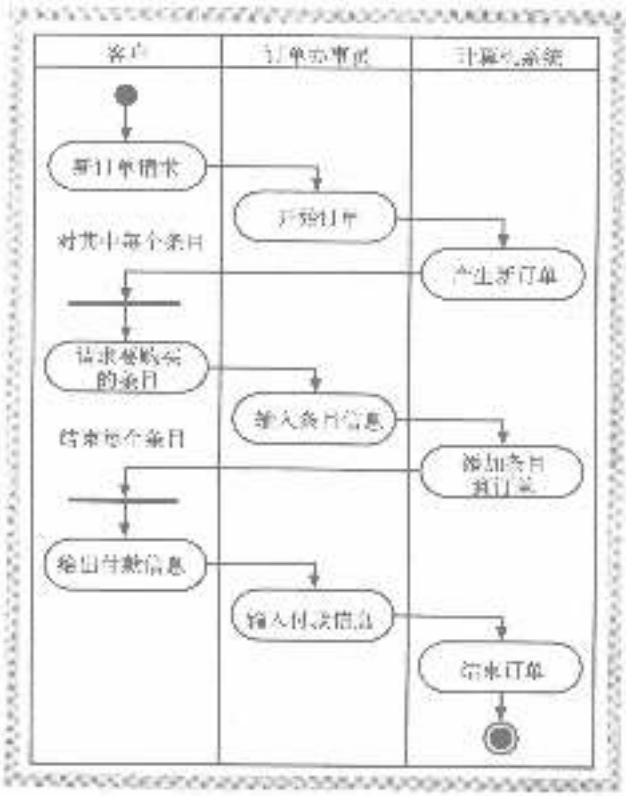


图7-16 电话订购场景的简化活动图

在例子中的第一条消息startOrder，本用例的前提条件是说用户应该存在。后续条件就是订单必须与一个客户相关联。因此，在用例的简化版本中，第1条消息将客户的标识accountNo传递进来。除了accountNo外，系统不需要其他参数来定位已存在用户的详细信息。

在第2条消息addItem中，需要参数来标识目录中的条目及要购买的数量。参数catalogID，prodID及size用于描述要添加到订单中的库存条目。当然，Quantity字段只确定了数量。

活动图中第3条消息，是输入支付数量。这个参数与OrderTransaction类中的amount属性一致。

3. 在输入消息上确定并添加特定条件，包括迭代和真/假条件。本例中，有迭代括和与之相关的真/假条件（在方括号中）。

4. 确定并添加输出返回消息。记住，有两种选择可以显示返回消息，可以在消息上添加返回值，或者也可以用虚线箭头表示一个独立的返回消息。活动图可以提供一些关于返回消息的线索，但是并没有一个标准的规则。当工作流中的转换箭头从系统出发到达外部参与者的时候，不一定总会产生一个输出。图7-16中有两个箭头是从计算机系统矩形区出发到达客户矩形区。然而，图7-17中只需要一个输出消息。图7-16中从产生新订单活动出发的箭头并不需要输出数据。在这个例子中，所确定的唯一输出处于显示添加到订单中的条目细节这一中间消息上。这些细节包括描述、价格和扩展价格（价格乘以数量）。另外的一些消息也可以显示输出信息，例如，第一个输入消息的用户名和地址和第三个消息的订单确认。

我们的目标是理解和发现，所以应该与用户一起工作，以便准确定义工作流是如何运作的，以及什么样的信息需要作为输出被传递和提供。这是一个迭代过程，在这些体反映用户需要之前，需要多次修改它们。在RMO开发项目过程中，Barbara Halifax（项目经理）和用

户一起重新研究了许多图（参见Barbara Halifax的备忘录）。

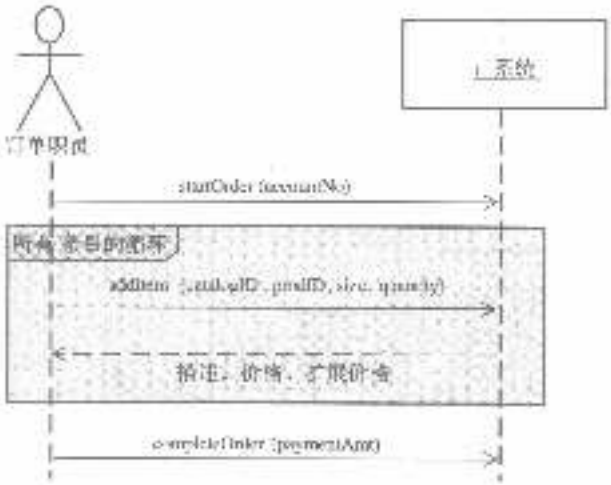


图7.17　产生新订单用例的电话订购场景的简化系统顺序图

2007年4月14日

To：John MacMurty

From：Barbara Halifax,Project Manager

RE：客户支持系统状况

John，这里是我们最近两个星期的工作状况报告及下阶段的工作计划。我同时也附上一份说明我们已经完成那些任务的日程表的副本。你会看到我们是很好地按着日程表来工作的，由于从销售部收到最终的确认有所耽搁，有些任务比预期的要多花一些时间。

前面一段时间（两星期）所完成的工作

在过去的两个星期中，我们在为以所定义好的用例开发完全展开用例描述、活动图和系统顺序图方面取得了一些进展。到目前为止，除了四个异常外，我们已在迭代中为所有的用例完成了系统顺序图。在最近与用户的会面中，定义了四个新的用例，我们决定将两个包含在迭代中，另外两个延迟到后面的迭代中，延迟的用例的详细文档也将在后面完成。

此外，当我们和用户一起用结构化遍历复查这些图的细节时，我们发现我们漏掉了一些类的关键属性。对于一些新的分析员来说，这确实是一款醒神酒，它告诉他们质量控制和结构化遍历我们的工作是多么重要。

下阶段（两星期）的工作计划

在下一阶段中，我们将开始查看这次迭代的一些设计活动，包括一些初步的数据库设计，我们还将研究实施操作方案以及网络需求。

问题、结论、未定条目

现在还没有什么主要的问题，在突出条目日志中我们有大约20条记录，但是没有一条能拖住我们。在监督委员会会议中你可以向部门领导强调一些尽可能快的解决这些条目的重要性。

BH

cc：Steven DeerField,Ming Lee,Jack Garcia

现在我们为产生新订单的Web订购场景开发系统顺序图。这不仅是一个更复杂的例子，更重要的是将要介绍如何为配置基于Web的系统开发需求。图7-13是一个基于Web订单的活动图，注意，这个工作流是比较复杂的。

图7-18是完成后的基于Web场景的系统顺序图。在图7-13中，从客户到计算机系统的工作流8次穿过了自动系统边界，其中的一些工作流是可选的。第1条消息和它的响应消息通过请求新订单页面（请求新订单）开始了一个用例，系统不需要输入数据来执行这两条消息所需要的处理，所以不需要输入参数。下一个输入消息是请求新客户页面（新客户主页），在这个消息中，有一个真/假条件用来检测这是否是一个新客户，这样，只有新客户条件取值为真时，消息才被触发。由于顺序图的目的只用来显示消息，而不用来显示处理逻辑，所以并没有显示扩展到其他用例的消息，只是增加简单的注释来提醒开发人员相应的跳转。



图7-18 产生新订单用例的Web订购场景的系统顺序图

第3条消息仅允许用户真正启动一个订单（开始订单）。这条消息显示客户账户号码是

个输入参数。在实际开发用户界面时，这个信息或者已经存在于系统中了，因为在添加一个新客户的时候它就在界面上了。但是，作为一个输入参数，开发人员了解这个参数必须是有效的，或者来自于用户或者从其他的页面获取。

接下来的处理是向订单中添加条目信息。图7-13所示的活动图显示了一个增加条目的循环过程。它被迭代框捕获。然而，工作流中的一个活动称为查询目录/查看条目。虽然循环并不明确，但是查询通常意味着某种类型的循环，所以，在图7-18中查看产品的输入消息中，为了表示重复添加了星号。迭代框和星号创建了缺失的循环条件。注意，这两个消息中，返回值的方法用于返回数据。其余的消息和响应见活动图。

本章的开头部分已经解释了在面向对象开发中用于对新系统的处理方面进行描述的模型集。用例图提供了必须支持的所有事件的综合。书面叙述或活动图提供的场景描述给出了每个用例内部步骤的详细信息。前提条件和后续条件描述有助于定义用例间的关联，也就是进行处理之前和之后什么必须存在。最后，系统顺序图描述了发生在用例中的输入和输出。这些模型一起为系统的处理需求提供了全面综合的描述，也为系统设计打下了基础。

我们已经很清楚地解释了用例，现在让我关注一下如何获得重要的对象状况信息。

7.4　确定对象行为——状态图

有时对于一个计算机系统来说，维护问题域对象的当前状况信息非常重要。例如，顾客想知道一个特定的订单是否已经发货，经理想询问客户订单并想知道是否已经付款。因此，系统要能够记录客户订单的状况。在需求定义阶段，分析员要确定和记录哪些域对象需要状况检查，以及什么交易规则决定有效的状况条件。回顾RMO，一个交易规则的例子是客户订单直到付款后才能发货。

现实世界对象的情况经常称为对象的状态。准确的定义，对象的状态就是在对象的生命周期中当其满足一些标准、执行一些行为或等待一个事件时所发生的一些状态情况。对于现实世界对象来说，将对象的状态和状态情况等同看待。

状态：在对象的生命周期中当其满足一些标准，执行一些行为或等待一个事件时所发生的状态情况。

状态情况的命名规则帮助确定有效的状态。一个状态可能有一个简单状态情况的名字，如正在装载。其他状态更加灵活。它们的名字由动名词或动词短语构成，如正在发货或正在工作。例如，一个特定的订单对象当一个客户订购商品时生效。当它创建之后，该对象处于增加新订单条目的状态，然后是等待送达订单条目的状态。最后当所有订单条目都运送完毕之后就是完成状态。当你发现自己正式使用名词来命名一个状态，你可能对状态或对象类有错误的理解。状态的名字本身不是一个名词，而是用来描述对象的（名词）。

状态描述为非永久性的状态情况是因为外部事件可能会打破一个状态或引发对象进入一个新状态。对象处于一种状态直到一些事件触发它转换或过渡到另一个状态。转换是指一个对象从一种状态转到另一种状态的活动。转换是使得一个对象离开其种状态而转为一种新状态的过程。状态是非永久性的是因为转换可以将其打破和结束。一般来讲，与状态相比转换被认为是短期持久的，并且不能被打断。状态和状态之间的转换相结合为分析员获得业务规则提供了方法。在前面的RMO例子中，一个客户订单在它能够转换为送达状态之前必须首先经过付款状态。在一个称为状态图的UML图中可以获取和证明这些信息。

转换：一个对象从一种状态转到另一种状态的活动。

可以为任何有着复杂行为或需要跟踪多个状态的问题域类开发状态图。然而并不是所有的类都需要状态图。如果问题域类中的对象不存在必须为该对象控制程序的状态，则状态图

就可能是非必要的。例如，在RMO类图中，订单类可能需要状态图，而订单转换类则不需要状态图。当付款完成时创建一个订单转换，不需要跟踪其他情况。

状态图由代表对象状态的椭圆和代表转换的箭头构成。图7-19描述了一个简单的打印机的状态图。因为通过实际条目学习状态图更加容易，所以通过计算机硬件的几个例子开始学习。基础知识介绍完之后，将介绍问题域软件对象的建模。状态图的开始点是一个称为伪状态的黑点。黑点之后的第1个椭圆是打印机的第1个状态。在这个例子中，打印机开始于关闭状态。状态由一个圆角矩形（几乎像椭圆，但比椭圆更方一些）代表。状态名字处于椭圆内。

伪状态：状态图的开始点，由一个黑点表示。

如图7-19所示，离开关闭状态的箭头称为一个转换。转换使得对象离开关闭状态而转换为开启状态。当转换开始以后，它通过把对象带到一个称为目标状态的新状态而完成。一个转换开始于一个从初始状态（先于转换的状态）指向目标状态的箭头，并且用一个字符串加以标记来描述转换的组成。

目标状态：对于一个特定转换，转换完成之后对象所到达的状态。

初始状态：对于一个特定转换，转换发生处对象的原始状态。



图7-19 打印机的简单状态图

转换标签由三个组成部分：

转换名称（参数，……）[监护条件]/行为描述

在图7-19中，转换名称是按下开启按钮。转换像一个被触发的触发器或发生的事件。名字应该反映触发事件的行为。在图7-19中，没有发送给打印机的参数。判定条件是安全盖关闭。对于要发生的转换，判定条件必须为真。前向斜线将状态与行为和过程分开来。行为描述表明在转换完成和对象达到目标状态之前必须经历的过程。在这个例子中，打印机在达到开启状态前将执行自检。

转换名称是引发转换和导致对象离开初始状态的消息事件的名字。注意，它的形式与系统顺序图中的消息非常类似。实际上，你会发现消息事件名称和转换名称使用几乎相同的语法。消息和转换之间还存在另一种关系。转换由到达对象的消息引起。消息名称的参数部分是接来自于消息参数。

消息事件：转换的触发器，导致对象离开初始状态。

监护条件是转换的限定条件或测试。它也是在转换发生之前必须满足的一个简单的真/假判定条件。对于一个转换，首先必须触发，然后判定条件必须为真。有时一个转换只有一个

判定条件，并且没有触发事件。在这个例子中，触发是持续的，所以无论判定条件何时变为真，转发将发生。

监护条件：判定转换是否可以发生的真/假判定。

同想顺序图中消息有类似的测试，称为真/假条件。这个真/假条件是消息发送端的测试。在消息可以被发送之前这个真/假条件必须为真。相反，判定条件在消息的接收端。消息可能收到，但是只有当判定条件也为真时才触发转换。这种测试、消息和转换的联合为定义复杂的行为提供了极大的灵活性。

Action表达式是在转换发生时执行的程序描述。换句话说，它描述将要执行的行为。三个组成部分（转换名称、判定条件和行为描述）中的任何一个都可以为空。如果转换名称或者判定条件为空，则自动为真。三个组成成分也可以用AND和OR任意组合。

Action表达式：作为转换的部分执行的活动的描述。

7.4.1　复合状态和并发性

在学习如何开发状态图之前，我们要介绍另外一种状态——复合状态。在现实世界中，一个对象在同一时间处于多个状态是非常常见的。考虑图7-19中的打印机，我们注意到当它处于一个状态时也可能在做其他事情：有时在打印，有时空闲，当它第一次被打开时通常进行自我检查。所有这些情况都是在它开启的时候发生的，被认为是同时发生的状态。同一时间处于多个状态的情况称为并发或并发状态。表示并发的一种方式是同步线和并发路径，就像在活动图中完成的那样（见图4-14）。所以可以用同步线来分裂一个转换，这样一条路径到达开启状态，一条路径到达空闲状态，另一条到达自检状态。我们定义路径为一组有序的相互连接的状态和转换。

并发或并发状态：同一时间处于多个状态的情况。

路径：一组有序的相互连接的状态和转换。

表示并发状态的另外一种方式是在其他高层状态中嵌套状态。这些高层状态称为复合状态。

复合状态：包括其他状态和转换的状态（也就是路径）。

复合状态表现更高层的抽象性，它包括嵌套的状态和转换路径。图7-20是图7-19的扩展，为打印机阐述了这一概念。打印机不仅处于开启状态，同时也处于空闲或工作状态。开启状态的圆角矩形被分为两个部分。上面的部分包括名称，下面的部分包括嵌套的状态和转换路径。

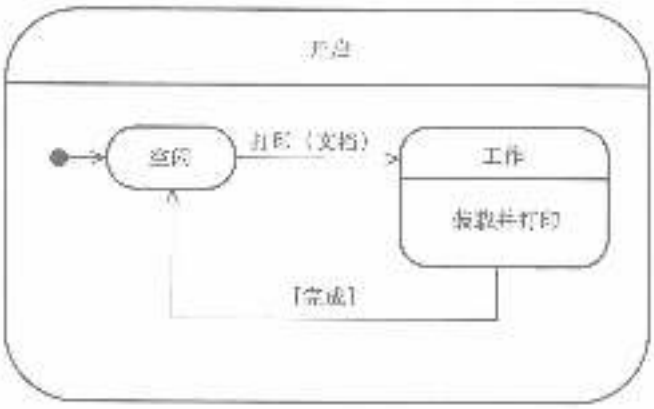


图7-20　打印机对象的复合状态实例

当打印机进入开启状态，它将自动地从嵌套点开始移向空闲状态。所以打印机同时处于开启状态和空闲状态。当收到打印消息，打印机转换为工作状态，但是依然保持在开启状态。有时为工作状态也引入一些新的符号。在这个例子中，下面部分包括行为描述，也就是当打印机处于工作状态时所发生的行为。

我们可以通过允许在一个复合状态中有多个路径来进一步扩展复合状态和并发的概念。也许一个对象有一组完整的活动并发的状态和转换（多路径）。为记录一个简单对象的并发多路径，我们画了一个复合状态，它的下面部分被分成多个部分，每一个部分对应一个行为并发路径。例如，假定打印机有一个输入口来放纸张。打印机工作时在空闲和工作两个状态之间循环，可能想描述两条独立路径，一条描述纸张输入口的状态，另一条描述打印机的状态。第1条路径有空、满和不满状态。第2条路径包括空闲和工作两个状态。这两条路径是独立的，一个组成部分内状态间的转换和另一个组成部分内状态间的转换是完全独立的。

如前所述，有两种方法来记录这种并发行为。首先，可以用一条由一条路径变为三条路径的并发线表示。第二，可以用复合状态，图7-21在图7-20的基础上扩展了打印机的例子，在这个例子中，在复合状态中有两个并发路径。上面的并发路径描绘打印机纸张输入口。这两条路径是完全独立的，并且打印机在每条路径里的状态和转换都是独立的。当按下关闭按钮时，打印机离开开启状态。很显然，当打印机离开开启状态时，它也离开这个嵌套状态的所有路径。打印机是否处于某状态或在转换中并没有关系。当按下关闭按钮时，所有活动结束，打印机退出开启状态。学习了状态图的基本符号后，我们接下来介绍怎样开发状态图。

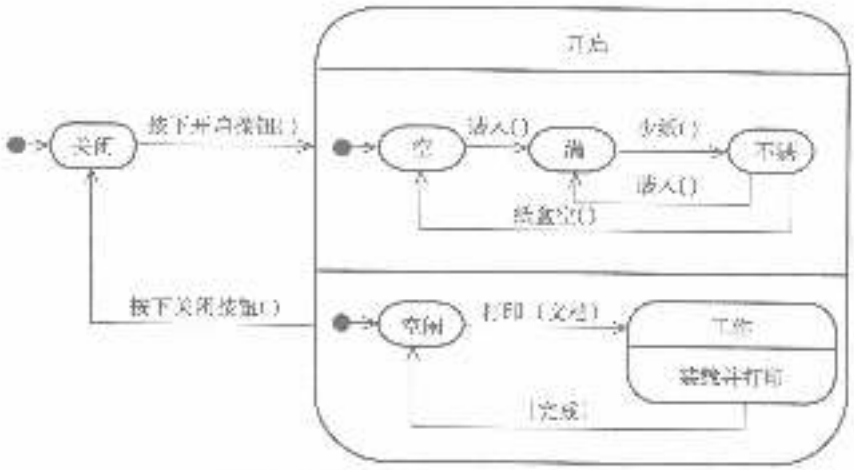


图7-21 处于开启状态的打印机并发路径

7.4.2 开发状态图的规则

状态图的开发遵循一组规则。这些规则帮助你为问题域类开发状态图。通常开发状态图的主要问题就是正确地为对象确定状态。把自己设想为对象本身会有所帮助。假想为客户很容易，但是如果说“我是个订单”或“我是一次运送。我如何产生？我处于什么状态？”就会相对困难些。不管怎样，如果你能够以这种方式考虑，对于你开发状态图将会大有帮助。

对于新分析员来说另一个主要困难是确定和处理带有嵌套的复合状态。通常这个困难的主要原因是在思考并发行为时缺乏经验。最好的解决办法就是记住并发状态图是一个重复的活动，比开发其他类型的图更加具有重复性。分析员很少一次就能得到一个正确的状态图。我们通常画出状态图并反复修改和优化。另外，要记住当定义需求时，你只知道对象行为的

大概情况。就像开发详细顺序图一样，在设计期间将有机会优化和修正重要的状态图。

最后，别忘记询问异常情况，尤其是当看到单词verify 和check时。通常，有两种离开某状态的转换需要检验，一个是接受，一个是拒绝。

下面列出帮助你开发状态图的步骤：

1. 回顾类图并找出需要状态图的类。记住我们通常只跟踪那些有多个对系统非常重要的状态的类。从看起来有最简单的状态图的类开始，例如后面要讨论的RMO中的订单条目类。

2. 对于组中每一个选出的类，列出你能确定的所有状态。在这一点，是简单的集思广益的讨论。如果你工作于一个小组，与整个小组成员进行集体讨论。记住你在定义软件类存在的状态。然而，这些状态也必须反映要在软件中展现的现实世界对象的状态。有时考虑物理对象非常有帮助，确定物理对象的状态。然后把这些合适的状态转换为相应的系统状态或状态情况。考虑对象的生命周期也非常有帮助，它们在系统中如何产生？何时以及如何将它们从系统删除？有活动状态吗？有非活动状态吗？有等待状态吗？考虑对对象施加的活动或由对象完成的活动。通常当这些活动发生时对象处于特定状态。

3. 通过确定导致对象离开特定状态的转换来开始开发状态图片段。例如，如果订单处于准备运送状态，例如“开始运送”的转换将导致订单离开该状态。

4. 按正确顺序将这些状态-转换组合排序。集合这些组合成为更大的片段。在这些片段集合成为大路径的过程中，很自然地开始为对象寻找自然的生命周期。继续通过组合这些片段来构建更长的路径。

5. 回顾这些路径并找出独立且并发的路径。当一个条目可以同时处于两个状态。将有两种可能。这两个状态可能处于独立的路径，就像打印机例子中的工作和纸。这发生在当状态和路径是独立时，且一个改变不影响另外一个。另外，一个状态可能是复合状态，两个状态可能是嵌套的。一个状态在另一个状态中。为复合状态确定候选者的一个方法是确定它是否与其他几个状态并发，以及其他这些状态是否依赖于初始状态。例如，当打印机处于开启状态时有几个其他状态和路径可以发生。这些状态都依赖于打印机处于开启状态。

6. 查找其他的转换。通常，在第一遍的构建过程中会有一些可能的状态-转换-状态组合会错过。确定它们的一个方法就是考虑每一对状态组合并询问状态之间是否有有效的转换。从两个方向检查这些转换。

7. 用合适的消息事件，判定条件和行为描述扩展每个转换。为每一个状态引入行为描述。很多工作可能在状态图片段构建阶段已经完成。

8. 回顾并检查状态图。我们通过仔细回顾检查状态图。通过以下方法检查每一个状态图：

   a. 确保状态确实是类对象的状态。保证状态的名字确实描述了对象所处的状态。

   b. 从产生到被系统删除跟踪对象的生命周期。确保所有可能的组合都已覆盖到，状态图的所有路径都是正确的。

   c. 确保状态图覆盖所有异常情况和正常行为流。

   d. 再次检查并发行为（多路径）和可能的嵌套路径（复合状态）。

7.4.3　开发RMO状态图

我们通过开发两个RMO状态图实践这些步骤。步骤一是回顾域类图并选择可能有需要被跟踪的状态的类。在这个例子中，我们选择订单和订单条目类。我们假定客户想知道他们订单的状态和订单中单独条目的状态。其他可并发状态图的类有库存条目，用来跟踪有货或缺货条目；运送，跟踪运到情况；客户，跟踪活动和非活动的客户。我们在这里要关注订单类和订单条目类。我们选择订单条目类是因为它简单，从简单的类入手通常是最好的方式。为

外，它也是一个依赖类，它依赖于订单类。最后，最好使用由底向上的方法，从低层条目开始会有少的连锁影响。

1. 开发订单条目状态图

通过确定可能的状态开始。一些必要状态是等待运送、订单延迟和运送。这时一个有趣的问题产生了：订单条目可以部分运送吗？也就是说，如果一个客户一个商品订购了10件，而库存只有5件，那么RMO要先运送5件而把剩余5件延迟订单吗？要看清这个决定的几个分支，系统和数据库要设计成能够跟踪和监视详细信息以支持这种能力。RMO的域类型（图5-41）表明一个订单条目可以与0个（还没运送）或1个（全部运送）运送相关。基于这些描述，不允许定义订单条目的部分运送。

这是体现建模优势的又一个例子，如果我们没有开发状态图模型，这个问题可能永远也不能提出。开发详细模型和图是系统开发者所执行的重要活动之一。它使得分析员提出基础问题。有时新的系统开发员认为建模是在浪费时间，尤其对于小的系统更是这样。然而，在写程序之前充分理解用户的需求从长远角度来看是省时间的。

下一步就是为每一个状态确定离开转换。图7-22是一个表，这个表列出了已经定义的所有状态，以及每一个状态的离开转换。表中加入了一个额外的状态"新增加"，它覆盖了当订单中加入一个新条目时所发生的所有情况。这时订单并没有完成或付款，所以订单并没有准备运送。

状态	导致离开该状态的转换
新增加	新增完成
准备运送	运送条目
订单延迟	条目到达
运送	没有定义离开转换

图7-22 订单条目的状态和离开转换

第四步就是组合这些状态—转换对成为片段并按正确的状态顺序构建状态图。图7-23所示为部分完成的状态图。订单条目从开始到结束的活动流很明显。然而，至少一个转换被忽视。应该有一些路径进入订单延迟状态，所以我们认为第一次开发出来的状态图还需要完善。我们将马上进行确定。

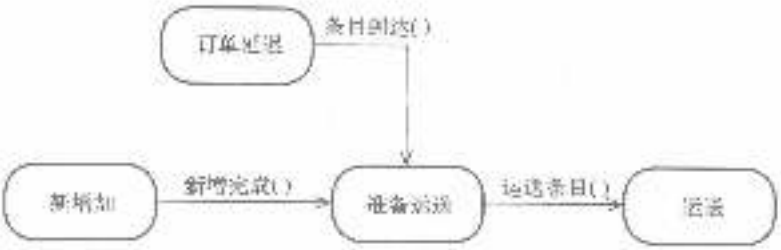


图7-23 订单条目部分完成的状态图

第五步就是查找并发路径。在这个例子中，没有出现一个订单条目可以同时处于任何两个状态的情况。当然，因为我们选择从简单状态图入手，这也正是我们所期待的。

第六步就是查找其他的转换。这一步我们充实其他必要的转换。第一个就是从新增加到订单延迟的转换。接下来检查每一对状态看看是否有其他可能的组合。特别地，查找反向转换。例如，订单条目可以从准备运送状态到订单延迟状态吗？当运送员发现仓库没有足够的货物而系统却显示有足够货物时这种情况将发生，其他反向循环，如从运送状态到准备运送状态，或者从订单延迟状态到新增加状态，都是没有意义的，也没有包括进来。

第七步就是用正确的名称、判定条件和行为描述完成所有转换。增加了两个新的转换名称。第一个是从开始的黑点转换为新增加状态。这个转换创建一个新订单条目对象。系统术语就是实例化。消息进入系统被赋予相同的名称——增加条目()。最后一个转换是将订单条

目从系统中移除。这个转换从运送状态到最后的带圆圈的黑点，这个带圆圈的黑点就是最后的伪状态。假设当其从系统中删除时会存档备份，所以这个转换命名为存档（）。

为转换增加行为描述是为了表示由对象引起或对象执行的一些特殊行为。在这个例子中，只需要完成一个行为。当一个订单从准备运送状态到订单延迟状态时，系统要引发一个新的对提供商的购买订单来购买更多的条目。所以，在标记延迟订单（）转换中，行为描述表示为创建一个购买订单。图7-24所示为订单条目的最终状态图。



图7-24　订单条目的最终状态图

最后一步，回顾并检查状态图。也就是质量检查阶段。经常试图省略这一步，一个好的项目经理会保证系统分析员有时间来对模型进行快速的质量检查。这时项目预算是非常合适的。

2. 开发订单状态图

订单对象比订单条目对象稍微复杂一些。在这个例子中，你将会看到状态图的一些其他的特征。这些特征支持更高级的对象。

图7-25是一个表，该表列出了定义的状态和离开转换。这些在第一次进行开发时似乎是需要的。从上往下看，状态反映了订单的生命周期。首先，一个订单产生并准备增加条目——开启增加条目。RMO的客户表示他们想在24小时内将订单保持在该状态以便增加更多的条目。当所有条目增加完之后，订单准备运送。接下来将运送并处于运送状态。这时，运送和等待延迟订单之间如何联系还不是很清楚，这个关系将在状态图的开发过程中被挑选出来。最后，订单处于运送状态，付款完成之后就会关闭。

状态	离开转换
开启增加条目	订单完成
准备运送	开始运送
运送	运送完成
等待延迟订单	延迟订单到达
已运送	付款完成
关闭	存档

图7-25　订单的状态和离开转换

在步骤4中，产生和组合片段来生成粗糙状态图。图7-26描绘了粗糙的状态图，状态图由那些对于多数部分来说都正确的片段构成。然而，我们注意到等待延迟订单状态存在一些问题。

对于一些分析员来说，我们认为运送状态和等待延迟订单状态是并发的。另外一个称为正在运送的状态也是需要的，在这个状态下运送职员正在运送商品条目。表示订单生命周期的一种方法是当运送开始时将其置于运送状态。这是它也进入正在运送状态，订单可以在正在运送和等待延迟订单之间循环。离开复合状态只发生在正在运送状态，也处于运送状态中。很显然，离开内部状态订单也离开运送这个复合状态。

执行步骤5、6和7，我们发现要增加新转换，需要从初始伪状态创建转换。转换还必须显示条目何时增加、何时运送。通常我们将这些循环活动置于那些离开某状态并回到同一状态的转换。在该例中，这个转换称为增加条目（）。注意，它如何离开开启增加条目状态又回到该状态。图7-27所示为该完成水平下的状态图。

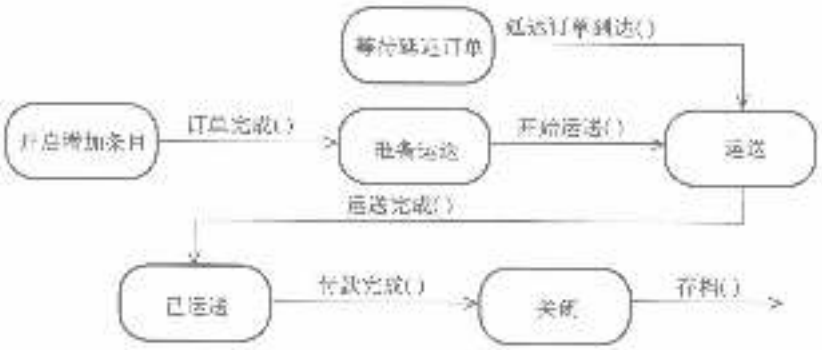


图7-26 粗糙的订单状态图

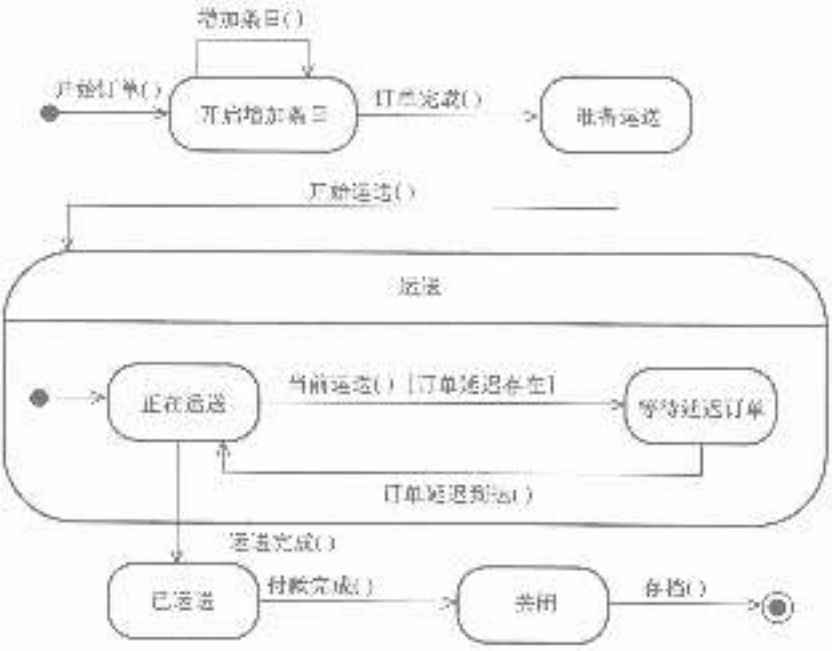


图7-27 中等详细的订单状态图

为对象甚至是业务对象开发状态图的好处就是帮助获得和搞清楚业务规则。从状态图我们可以看到，当订单处于开启增加条目状态时运送不能开始。当订单处于准备运送状态时不能增加新的订单条目。如果订单处于运送状态，我们知道它或者处于工作状态或者等待延迟订单。

还需，详细建模的好处就是帮助我对系统需求的更好的理解。我们现在来看一个大图，把不同模型合为整体，来看看他们是如何协作的。

7.5 面向对象模型的集成

本章描述的图表使得分析员可以完整地描述系统需求。如果你正在使用瀑布系统开发生命周期方法开发一个系统，那么在继续设计之前，需要开发一整套的图来代表所有系统需求。然而，当为你正在使用迭代方法，所以只需构造一次特定的迭代所必需的图表就可以了。一个完整的用例图对于理解新系统的总体规模是很重要的。但是包含在用例描述、活动图和系

统顺序图中的支持细节只需为特定迭代中的用例完成。

实践指导

开发并整合域类模型，确保你已理解业务需求。

域模型类图是一个特殊的案例。域模型类图更像完整的用例图。对整个系统而言它应该尽可能的完整。如第5章中RMO所示的一样。系统的问题域类的数量为系统总体规模提供了一个额外的指示。许多类的精炼和实际执行将等到以后的迭代中进行，但是域模型应该基本完成。域模型对于确定新系统所需要的所有域类是必要的，虽然我们在本章中不关心数据库的设计，但是域模型还可以用于数据库的设计。

通览本章，你可以明白一个图的构造如何用另一幅图所提供的信息来完成。同样也可以了解新图表的开发通常有助于精简和纠正先前的图。此外还要注意详细图的开发对于充分理解用户需求至关重要。图7-28说明了面向对象开发中需求模型间的主要关系。左边的用例图和其他图用来获取新系统的程序。类图和其依赖图获取新系统的类信息。实心箭头表示主要的依赖关系。虚线箭头表示次要依赖关系。这种依赖性通常从顶部流到底部，但是一些箭头是双向的，它在两个方向都产生影响。

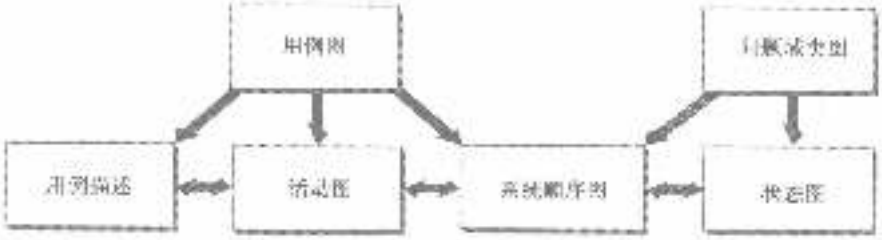


图7-28　面向对象需求模型中的关系

注意，用例图和问题域类图是两个主要的模型，依靠这些主要模型其他的图可以从中获取信息，应该尽可能完整地开发这两种图表。本章的开始部分介绍的类图和用例图之间的CRUD分析将有助于确保尽可能完整地开发这两种模型。以叙述格式或者活动图的形式进行的详细描述，是很重要的用例内部记录，而且必须完全支持用例图。这些详细描述对系统顺序图的开发也是很重要的。所以，详细描述、活动图和系统顺序图关于特定用例必须具有一致性。随着系统开发的进行，特别是当你开始做详细系统设计的时候，你将会发现理解模型之间的关系是保证模型质量的重要因素。

小结

面向对象方法中有一整套的图表集合，一起用来记录用户的需要和定义系统需求。这些需求可以用以下模型来说明：

- 域模型类图
- 用例图
- 用例详细模型，叙述形式或活动图
- 系统顺序图（SSD）

用例图记录了系统可用的各种方法。它可以独立开发或根据事件表进行开发，一个事件触发一个用例。一个用例由参与者、用例和连接线组成。一个用例标识了系统支持的一个独立的功能。参与者代表使用系统的某人或某物的角色。连线用以表示哪个参与者触发哪个用例。用例也可以作为一个适用于程序激发调用其他的用例。用例之间这种类型的连接叫做

<<包含>>关系。

用例的内部活动首先由内部的活动流描述。有可能有好几个不同的内部流，代表同一个用例的不同场景。所以，一个用例可能有多个场景。这些详细或者以叙述的形式或者以活动图的形式被记录下来，以描述工作流。可以为每个用例定义前提条件和后续条件，这有助于理解对执行用例系统的其他影响。

系统顺序图（SSD）提供了用例处理需求的更详细地说明。系统顺序图记录了系统的输入和输出。每个SSD的应用范围通常是一个用例或者用例的一个场景。SSD由参与者（和用例中标识的参与者一样）和系统两部分组成。系统被看成是一个黑盒子，因为不需要记录内部处理。代表输入的消息由参与者发送给系统。输出消息由系统返回给参与者。消息的顺序自顶向下的。

在定义需求时，域模型类图得到进一步精简。类图中显示的商务对象的行为是要研究和建模的需求的一个方面。状态图用来建模对象状态和用例中发生的状态转换。本章中讨论的所有模型都是内部相关联的，一个模型中的信息可以解释其他模型中的信息。

关键术语

action-expression	Action表达式
composite state	复合状态
concurrency, or concurrent state	并发或并发状态
destination state	目标状态
guard-condition	监护条件
interaction diagram	交互图
lifeline, or object lifeline	生命线或对象生命线
message	消息
message event	消息事件
origin state	初始状态
package	包
path	路径
postcondition	后续条件
precondition	前提条件
pseudostate	伪状态
scenario, or use case instance	场景或用例实例
state	状态
state machine diagram	状态图
system sequence diagram	系统顺序图
transition	转换
true/false condition	真/假条件
use case diagram	用例图

复习题

1. 什么是OMG？
2. 什么是UML？它用于什么类型的建模？
3. 用类模型的两个基本的组成部分是什么？它的目的和目标是什么？

4. 用例描述和活动图的区别是什么？
5. 场景指的是什么？用于描述场景的另外一个术语是什么？
6. 为什么参与者要有下？
7. <<包含>>关系用于什么？
8. 用例图和事件表边界条件的区别是什么？
9. 关于用例，活动图可用于做什么？
10. 什么是前提条件？什么是后续条件？在用例开发中它们为什么重要？
11. 开发系统顺序图的目的是什么？什么样的符号用于系统顺序图？
12. 开发系统顺序图的步骤是什么？
13. CRUD分析指书是什么？它是如何工作的？
14. 描述本章中介绍的模型以及它们的相互关系。
15. 状态图的目的是什么？
16. 列出转换描述的构成元素，哪些元素是可选的？
17. 什么是复合状态？它用于做什么？
18. 术语路径指书是什么？
19. 保护条件的目的是什么？
20. 区分本章中介绍的几种模型和它们之间的关系。

思考题

1. 复习开发类图的知识，基于以下描述开发一个域模型类图，要包括关联和重数。

这个例子是一个简化了的大学图书馆新系统。当然，图书馆系统必须跟踪书的情况。同时要维护关于书的标题以及副本的信息。书的标题维护关于名称、作者、出版商和目录号等信息。每个副本维护副本号、版本、印刷日期、ISBN、本书状态（它是否被借出）和归还日期等信息。

同时图书馆系统也要跟踪有从图书馆借书人的情况。由于它是一个大学图书馆，所以有几种类型的借书人，他们有各自不同的特权。这里包括：教职工借书人、研究生借书人和本科生借书人等。借书人的基本信息包括：姓名、地址和电话号码等。对于教职工借书人，还要包括诸如办公室地址和电话等的信息。对于研究生借书人，还要包括研究项目和导师信息等。对于本科生借书人，还要包括项目和所有学分信息等。

图书馆系统也要跟踪借出书本信息。图书馆借出书本是一个抽象对象。当一个借书人携带一些书去借书台办理借书手续的时候，借出这个事件就发生了。随着时间的过去，一个借书人可以多次从图书馆中借书。一次借出事件和许多本书相关联。（一段时期内一本书可被借出多次。数据库中保留了过去借出的信息。）所以，在本例中，应该为借书这一事件建立相关类。

如果借书人想要书书已被借出，他可以预约。这是另一个类，它并不代表一个具体的对象。每个预约只针对一个借书人和一个标题。预约日期、优先权和完成日期等信息需要维护。当借书完成时，系统会将这本书与借出联系起来。

2. 为大学图书馆系统开发一个用例图。

① 根据下面书描述，建立初步的用例图。

借书人根据图书馆的信息来检索书名，同时检索这本书是否可以被借出。如果一本书的所有副本都被借出了，那么借书人可以根据书名预订这本书，当借书人把书拿到借书台的时候，管理员为这些书办理借出手续，同时管理员也办理归还手续，库存管理员要跟踪新到达的情况。

图书馆的管理者有属于自己的活动，他们要按分类打出关于书的标题表，还要（在线）检查所有过期未还的书。当一些书被损坏的时候，他们会删除关于这些书的副本的信息，管理者

还需要检查那些书已被预约。

② 根据习题1中开发的类图，做CRUD分析，并列出你发现的所有用例。或者如果你改变了用例的名称，也写出来。在这个例子中，也可以从另外一个大学的数据库中访问和下载借书人的信息。

3. 复习开发类图的知识，基于以下描述开发一个类图，要包括关联和重数。

一个门诊部有三个牙科医生和几个牙齿保健人员，他们需要一个系统来帮助管理患者信息。这个系统不需要保留任何药品记录，它只处理患者管理部分。

每个患者都有关于姓名、出生日期、性别、第一次和最近一次就诊日期的记录。患者的记录是以家庭为单位组织在一起的。每个家庭都有户主姓名、住址和电话号码等属性。并且每个家庭都有一个医疗保险记录，这个医疗保险的记录包括保险公司名称、地址、联系人和电话号码。

在这个门诊部里，每个牙科职员都有一个记录以跟踪谁为其个患者服务（牙医、牙齿保健人员、x射线工作人员）。由于系统关注患者管理记录，所以牙科职员只有姓名、地址和电话号码等很少的信息。同时需要维护患者就医的信息，如日期、保险公司账号（由患者付费）、支付代码和付款金额等。一个就诊信息对应一个患者建立，但是在这个系统中一个患者可以去许多科室就诊。在一次就诊中，患者可能与不止一位牙科工作人员发生联系，例如，x射线工作人员、牙医和牙齿保健人员等可能都会参与进来。事实上，一些牙医在某一方面有一定的专长，并且可能一位患者的病情需要多位牙医进行诊治。因为每个工作人员都是按照一定程序进行工作的，所以系统要保有关于该程序的详细信息。这些信息包括：程序类型、描述、涉及的牙齿、前期付款金额、所有费用、支付金额和保险公司拒绝支付的费用等。

最后，系统要跟踪发票信息。该系统中有两种类型的发票：保险公司发票和户主发票。这两种类型的发票很相似，它们都列出了每一次就诊情况、涉及程序、前期付款金额和实付款金额等。显然，保险公司支付的所有金额和患者自己承担的部分是不同的。虽然发票只是打印出来的一个报告，但是它同时也包含了一些信息，如发票日期、总金额、已支付金额、应付款金额和实收金额、收款日期及拒付总额（保险公司并不总是支付账单上的所有费用）等。

4. 为牙科门诊开发一个用例集。

② 基于以下描述开发用例图。

接线员跟踪患者和户主的信息。他/她向系统中输入患者和户主的信息。同时也跟踪患者就诊信息。门诊管理人员也可以输入和维护患者信息。另外，门诊管理人员也维护牙科工作人员信息。

门诊管理人员负责开具发票。患者发票每月开出一次，并送到户主手中，而保险发票每周开出一次。当开具发票的时候，门诊管理人员要对照系统中的信息来检查两次发票，这样做的目的是为了确保发票合计正确。当收到付款时，门诊管理人员也要向系统输入该信息。

每个牙科工作人员有责任把自己的工作程序输入到系统中。

门诊管理人员也负责打印过期发票报表。这主要是为了列出尚未支付费用的户主清单。有时，牙医要看一周/月内的工作清单，这时他们就需要这个报表。

③ 基于前一个问题中开发的类图的CRUD分析，扩展你的用例图。

5. 解释图7-29所示的用例图。解释使用这个系统的不同的角色，以及每一个角色需要完成什么样的功能。解释这种关系，以及用例图相互联系的方法。

6. 对照下列描述操作：

① 为每个场景开发活动图；

② 为每个场景开发完全展开用例描述。

高质量建筑供应系统有两类客户：承包商和普通大众，针对不同人的销售方法是不同的。

当承包商购买东西的时候，销售人员把他们带到承包商专用柜台前。接待员向系统输入承包商的姓名。系统就会显示承包商的信息，包括他现在的信誉。

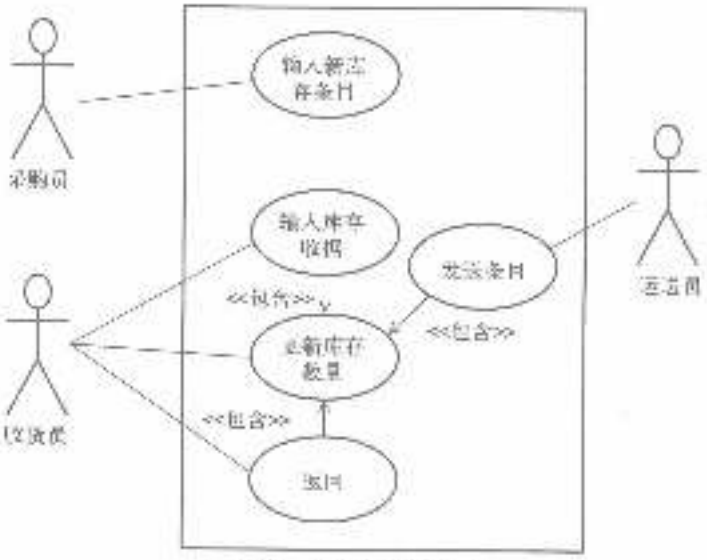


图7-29　库存系统用例图

接着接待员为承包商建立一个新的销售单据。然后，接待员检索卖出的所有项目。系统自动发现每一项的价格并把它们累加到单据上。购买的最后，接待员指示销售结束。系统比较总金额和承包商现在的信用卡余额，如果信用卡余额大于总金额，则结束销售。系统为所有项目建立电子单据并且承包商的信用卡余额将减去销售的金额。一些承包商喜欢保留他们的购买记录，所以他们需要打印后详细单据。另外一些承包商则对打印单据不感兴趣。

销售情况输入收银机，然后当条目确定打印一张收据。付款方式可能是现金、支票或信用卡。办事员必须输入付款方式以确保收银机在交易时结算。当选择信用卡付款时，系统打印信用卡单据，让客户签字。

7. 根据下列描述，针对一个汽车保险系统中将一辆新车加入一个已有保单中的用例，设计活动图或完整的开发描述。

客户打电话给保险公司，并提供他/她的保单号，办事员输入这个信息，系统显示基本的保单。然后办事员检查信息，以确保保险费适用以及保单有效。

客户给出要添加汽车的品牌、模型、年份和车辆识别代号(VIN)，办事员输入这些信息，系统验证这些数据是否有效。然后，客户选择期望的保额类型及每种类型的数量，办事员输入这些信息，系统逐一记录并根据保单限制验证所请求的数量。输入所有的保额后，系统验证保额总数，包括保单上的其他汽车。

最后，客户必须确定所有的驾驶员及他们驾驶汽车的时间比例。如果有一个新驾驶员加入，则调用另一个用例增加新驾驶员。

整个过程最后，系统更新保单，计算新的保险费，打印新的保单说明，邮寄给保单所有人。

8. 假设前面的汽车保险系统中的类和关系列表如下。针对将一辆新车加入一个已有保单中这个用例，列出其前提条件和后续条件。

系统中的类：

- 保单

- 被保人
- 保险车辆
- 保险总额
- 标准保险总额（按驾驶分类列出标准保险总额的价格）
- 标准车辆（列出已有的各类车辆）

系统中的关系：

- 保单有被保人（一对多）
- 保单有保险车辆（一对多）
- 车辆有保险总额（一对多）
- 保险总额是标准保险总额的一种
- 车辆是标准车辆的一种

9. 根据问题6的描述和活动图，设计一个系统顺序图。
10. 根据习题7的描述或活动图，设计一个系统顺序图。
11. 如图7-30所示为移动电话状态图，回答以下问题（注意，这种电话具有不同于普通电话的特殊特点，请基于状态图进行回答。）

a. 打开电话会发生什么？
b. 电话打开后会进入什么状态？
c. 关闭电话的三种途径是什么？
d. 在活动（通话）过程中能关闭电话吗？
e. 电话如何才能达到活动（通话）状态？
f. 当有人在通话时可以接通电话吗？
g. 当有人通话时手机可以进入充电状态吗？解释那些活动是允许的，哪些是不允许的。
h. 那些状态是与其他状态相并存的。列出一个两行表显示并存的状态。

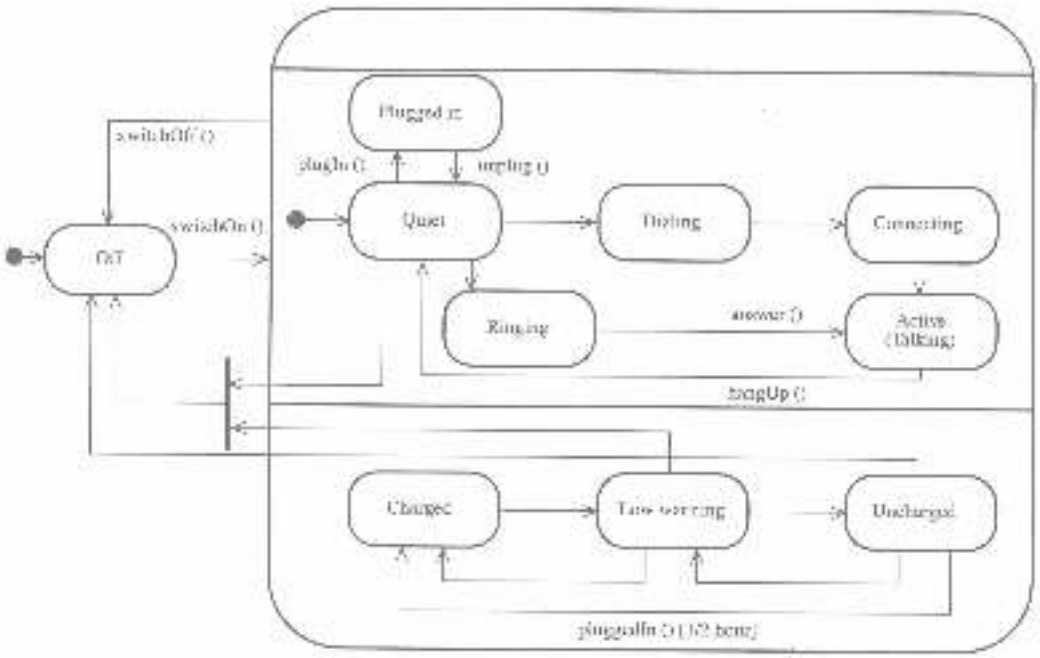


图7-30 移动电话状态图

12. 基于以下对包裹运送公司运送的描述，首先确定所有的状态和它们的转换，然后画状态图。

当从客户接收包裹后运送首次被确认。当包裹处于系统中时，它就被认为是活跃的并在运送中。每当它到达一个检查点，如到达中间目的地，将进行检索并创建一个记录来表明检查的时间和地点。当包裹装上运送卡车的时候状态发生变化，它仍是活跃的，但它也被认为处于运送事件状态。当然，当包裹运送后状态将改变。

有时，运送的目的地在公司服务范围之外。在这些情况下，公司要与其他运送服务商联合工作。当包裹移交到另一个运送商时，包裹标记为移交。在这种情况下，新运送商将记录跟踪号码（如果提供的话）。公司要求新运送商在包裹送达之后提供状态改变通知。

不幸的是，有时包裹会丢失。在这种情况下，包裹在闭回内会保持活动状态，但是会打上有可能丢失标签。如果闭回内包裹还没有找到就认为丢失了。这时候，客户可以开始丢失程序来报告损失。

实验练习

1. RMO客户支持系统所提功能同样出现在几个其他公司中。根据你的在线购物经验，建立具有网络客户（类似表7-4）功能的用例区。这些网站包括L.L.Bean（http://www.llbean.com/），Lands'End（http://www.landsend.com/），Amazon.com（http://www.amazon.com/）和Barnes and Noble Booksellers（http://www.barnesandnoble.com）。
2. 为你在练习1中定义的用例，开发完全展开用例描述。
3. 基于练习2中开发的活动流，为用例和场景开发系统顺序图。为每个用例添加前提条件和后续条件。
4. 分析练习1中网站的信息需求，执行CRUD反分析（从用例到领域模型类区）将有助于标识所有的类，开发域模型类图。
5. 找到一个微软件开发的本地公司。这个公司是咨询公司或者有许多信息系统专员的公司。它有比较严格的软件开发方法。通过一次会面来确定他们使用的开发方法。许多公司坚持使用面向对象方法和传统结构化技术相结合的方法。另一些公司有的项目是用传统方法，而其他的一些项目则使用面向对象方法。找出这个公司使用何种建模方法来为需求说明服务。比较本章所学到的技术和你的发现。
6. IBM Rational是IBM的全资子公司。UML的作者曾经是Rational Rose的头号总裁。因此，IBM Rational是开发支持UML的CASE工具和面向对象建模方法的早期的领头羊。你可以下载IBM Rational的UML CASE（IBM Rational XDE Modeler）工具的测试版本，并且可以使用这个工具来画RMO图。它已经成为广泛用于工业领域的工具。同时，你的大学可以参加种子项目来获赠实验室工具的副本。网址如下：http://www.rational.com/。

实例研究

房地产多端目服务系统

参考第5章的实例研究中描述的房地产多端目信息服务系统。以系统的事件列表和ERD作为开始点，开发以下的面向对象模型：

1. 将ERD转换到域类图。
2. 开发一个用例图。
3. 为每个用例开发完全展开用例描述或者活动图。
4. 为每个用例开发系统顺序图。

国家巡查罚单处理系统

参考第5章的实例研究中描述的国家巡查罚单处理系统。以系统的事件列表和ERD作为开始点，开发以下的面向对象模型：

1. 将ERD转换到类图。
2. 开发一个用例图。
3. 为用例图中的两个主要用例分别开发完全展开用例描述，如记录交通罚单和开庭日期表。
4. 为这些用例开发系统顺序图。
5. 为一张到款开发状态图。

城市影碟出租系统

城市影碟公司是一家连锁店，它有11个影碟商店分布在中西部的主要市区。好几年前它还只有一个商店，现在已经发展壮大到现在这个规模。Paul Lowes是这个连锁店的业主，他知道要与国内的其他连锁店竞争，就需要一个技术先进的影碟出租系统。现在要求你开发新系统的系统需求。

每一个商店都有一些电影和游戏碟出于出租。跟踪每个影碟的名称是重要的，这样就可以知道它的类别（经典、戏剧、喜剧等）、出租类别（新发行、标准）、影片等级，以及如电影制片人、发行日期、成本等常用信息。除了要跟踪每个影碟的名称外，这个系统还需要跟踪每个副本以便知道购买日期、它的条件、类型（VHS或DVD）及其出租状态。用户功能还需要提供维护库存的信息。

客户是这个系统的最重要的对象，当然也要跟踪。该公司认为每一家庭都将会是一个客户，所以给每一家庭发邮件和促销信。对于特定的客户，好几个人可以被授权借影碟和游戏碟。每一个客户的主要联系人可以为家庭的其他成员建立借出尺度。例如，如果父母还限制孩子只能借PG和PG-13的电影的话，系统可以跟踪这些。

每一次影碟出租，系统必须跟踪哪一部电影或游戏的哪一个副本出租了。以及出租日期和时间、返回日期和时间，借出家庭和借出人。每一次出租认为是处于打开状态，直到所有的影碟和游戏都还了。在借出影碟时客户就在商店支付租金。

对于这个实例，开发如下的图：

1. 一个域模型类图。
2. 一个用例图，分析用户功能，同时要做基于类图的CRUD分析。
3. 与租借和退还电影有关的用例，以及维护客户和家庭成员信息的用例的活动图。
4. 为问题3的每个用例建立系统顺序图。
5. 基于本章先前提供的用例描述及你在影碟出租商店工作情况的了解，为电影碟副本开发状态图来识别所有可能的状态。

EyesHavelt.com图书交易系统

EyesHavelt.com 图书交易系统是一个电子商务交易系统，它完全在网上进行交易。公司在购书者和卖旧书者之间充当了信息交换所的角色。

如果一个人想卖书，他必须先到EyesHavelt.com进行注册。他必须提供现在的住址，电话号码及现在的电子邮件地址。然后，系统会为他开一个账户，以后他将以卖书者的身份通过一个安全的经过鉴证的入口与系统进行交互。

卖书者可以通过网上的专门的表单来在系统上列出他的书。这个表单包含了所有与这些书相关的信息，如它的目录、普通保险条款及索价。卖书者可以列出任意多本书。系统为所有注册过的书都进行了编号，以便购书者可以通过搜索引擎来找到某本书。搜索引擎可以通过标题、作者、

口或及关键字来进行搜索。

想买书的人可访问站点，并查找他们想要的书。当他们决定购买时，他们必须打开自己的信用卡账户来支付书款。系统在安全的服务器上保持所有这些信息。

当一个有支付的购买请求发生时，EyesHavelt.com给卖书者发送电子邮件通知他那本书被选中了。系统同时将这本书标记成已售出。系统需保持着这样的信息，直到它接收到书已被发送的通知。　且卖书者收到他所列出的书已被卖出的信息，他必须在48小时内以电子邮件的方式通知购买者这次购买行动已经被记录了。在卖书者发送通知信息的24小时内，书本的发送必须进行。当开始发送书时，卖书者必须通知购书者和EyesHavelt.com。

当收到发送通知以后，EyesHavelt.com将订单设置为已发送状态。在每个月的月末，一张支票会邮寄给卖书者，他们的书本订单已经处于已发送状态有30天了。这30天的等待时间可以让购买者在书没有送到或书并不是如同网页宣传的那样时通知EyesHavelt.com。

如果他们愿意的话，购书者可以为卖书者输入服务代码。服务代码表明卖书者为购书提供了怎样的服务。有些卖书者非常活跃，他们将EyesHavelt.com作为其主要的售书途径。所以服务代码是对潜在的购买者来说是一个很重要的指示。

对于这个实例，开发如下的图：

1. 一个域模型类图。
2. 一个用例图。
3. 为两个用例（添加卖书者和记录图书订单）开发完全展开描述。
4. 为问题3中的每个用例建立系统顺序图。

对落基山运动用品商店实例的再思考



图5-16是RMO的事件表。以这个事件表为基础，我们开发出图7-5中的用例图。这一章说明了产生新订单详细模型（活动和系统顺序图）。

使用RMO案例描述中提供的信息和书中的图（图5-16和图7-5），我们分别为下列客户参与者用例分别开发了用例的完全展开描述和系统顺序图①更新订单用例；②创建订单更省用例。两种操作。

关注Reliable Pharmaceutical Services



前一章已经描述了可靠药品服务系统的活动和处理。使用前面的描述方法，特别是第1章中的基本描述方法和第5章的详细描述方法以及下面，本例中附加的描述方法，开发面向对象的需求模型。

公司处理（用例开发）

在订单填写过程中，信息必须被记录在系统中，同时要记住几个要点。显然，必须记录新订单。在每个变化的初期需要打印出用例清单。事实上，由于在一个长期存在的处方的例子中，需要也很长的时间才能完成的应用一个处方，所以药品一发出（填写处方）信息必须填写到系统单上，同时注意发送药品的数量和哪一个药剂师填写了变化的处方。

与第5章解释的一样，所有病人、住院部，员工和保险公司等的基本信息同样需要记录在系统中。

信息需求（类图需求）

可靠药品服务系统需要知道关于病人，住院部和病房（病人住的地方）的信息。每个住院部至少有一个病房。一个病人被指派到一个特定的病房。

处方是一个很复杂的实体。它们包括了一些基本信息，如ID号、订单原始数据、药品、剂量

单元（药丸、茶匙、栓剂）、剂量单位（毫克、茶匙的数量）、剂量周期或频率（每天、两天一次、每隔一天、每4小时）和注意事项（和事物一起服用、饭前服用）。另外，有几种类型的处方，它们都具有独一无二的特征：一些是只用一次处方的订单，一些是特定剂量（药丸）的订单，一些是时间周期（开始日期、结束日期）的订单。同时需要维护处方订单的信息。当住院部需要这种处方的时候，订单事件发生。由于处方可以持续一段时期，所以处方是独立于订单之外的实体。这个系统同时记录了哪个职员接收和填写了原始订单。

该系统同时也拥有所有药品的基本数据。每种药品都有一些通用信息，如名称、化学成分和制造厂商。然而，该系统同样也保留了每类药品更详细的信息，如每个药丸的重量。一种药品可能有许多种剂量单元和类型。

此外，要维护订单填写的相关信息。例如，在一个处方中关于药丸数量的说明，每次发给病人一粒或几粒。系统必须记录实际发放药丸的数量。同时也要维护药剂师或助理药剂师填写订单的情况。这里假定所有的药品都是按12小时一次分发的。

系统要保留买药人的一些基本信息，如姓名、住址和联系人。在第一个阶段不必担心买药人是用支票还是其他的支付方式。这些功能在下一个阶段会被添加进来。

基于先前的工作，同时根据前几章的实例，做如下描述：

1. 根据需要精简并扩充第5章开发的域模型类图。

2. 开发用例图，根据第5章建立的事件表，确保包括问题中类别的CRUD分析，并讨论基于CRUD分析还需要什么样的用例。

3. 为与下列事件有关的用例开发活动图：输入新订单、创建清单列表和填写订单。至少需要三个活动图。为每个用例撰写完全展开用例描述。

4. 为问题3所开发的用例开发系统顺序图。

5. 为某订单开发状态图。

参考资料

Grady Booch, James Rumbaugh, and Ivar Jacobson, *The Unified Modeling Language User Guide*. Addison-Wesley,1999.

E.Reed Doke, J.W. Satzinger, and S.R. Williams. *Object-oriented Application Development Using Java*. Course Technology, 2002.

Hans-Erik Erikson, Magnus Penker, Brian Lyons, and David Fado. *UML 2 Toolkit*. John Wiley & Sons, 2004

Martin Fowler. *UML Distilled Third Edition: A Brief Guide to the Standard Object Modeling Language*. Addison-Wesley,2004.

Ivar Jacobson, Grady Booch, and James Rumbaugh. *The Unified Software Development Process*. Addison-Wesley,1999

Philippe Kruchten, *The Rational Unified Process, An Introduction*. Addison-Wesley, 2000.

Craig Larman. *Applying UML and Patterns: An Introduction to Object-oriented Analysis and Design and the Unified Process 3rd Edition*. Prentice-Hall, 2005.

Object Management Group,*UML2.0 Superstructure Specification*, 2004

James Rumbaugh, Ivar Jacobsen, Grady Booch, *The Unified Modeling Language Reference Manual*. Addison-Wesley,1999.

第8章 需求、环境与实施的候选方案评估

学习目标

阅读本章后，你应具备如下能力：

- 根据新系统期望的范围和自动化水平，划分系统需求优先级
- 描述让一种战略性决策将程序开发环境与设计方案结合起来
- 确定系统开发的可选方案
- 根据组织的需要和资源，评估并选出一种开发方法
- 描述建议需求（RFP）的关键元素，并评估供应商关于外购方案的建议
- 为管理层开发专业的研究结果报告

本章要点

- 项目管理的前景
- 决定范围和自动化水平
- 定义应用程序配置环境
- 候选实施方案的选择
- 与供应商签订合同
- 提交结果并做出决策

热带鱼销售公司：链接到正确的系统

Robert Holmes很难确定他的项目该怎么进行下去。他在为他的公司——热带鱼销售公司开发一个基于Internet的订单系统。六个软件提供商对此提出了各自的建议，他和他的项目组必须找出一种方法对这些建议进行有意义的比较，从而决定哪一种方案最适合公司的要求。然后，他还必须给出他们的分析结果报告和建议。

问题是六种建议方案各不相同。他和他的小组花了大量的时间来开发建议需求（RFP），并把RFP发给了几家提供客户解决方案的公司。他们仔细研究了RFP，并确保它所包含的业务需求定义非常精确，但即使有了这种设计详细的RFP，仍然没有任何两种方案是相同的。他必须设计一种方法对这些方案做出公平的比较，否则他如何知道哪种解决方案对公司最好呢？

公司早已决定设计一份RFP，在设计过程中得到了许多外来的帮助。这个项目看起来相当大，而信息系统的工作人员非常少而又缺乏经验。最经济的解决方案来自于一个公司，它有一套现成的标准订单系统。这种方案的优点是可以迅速而又相当经济地安装系统并使之运行起来，但是该系统不能很好地满足公司的全部需求。Robert不能确定系统所缺少的功能对于他的公司到底有多重要。此外，这个系统必须根据工作程序和形式做一些修改，当然修改之后还必须保证系统仍可以正常工作。

另一个方案则是建立一个Internet销售的全新的高技术系统，并为供应商和运输商提供了电子界面。这个系统是一个全面的，完全自动化支持的电子商务解决方案。这个方案将可以保留实质性的交易，顾客及订单历史信息，而且保证这些信息是实时可用的，此外系统还将包含自动的库存管理功能。虽然这个系统的许多功能目前公司并不需要，但它必定将本公司

带到高技术解决方案的前沿。然而，这种方案的价格是前一种方案的三倍。Robert不知道公司是否有能力支付。

其他方案介于上述两种方案之间。一个公司建议从头开发一个系统，这样能使系统和公司的要求紧密结合，从而很好地满足公司的要求。另一个公司有一个基础系统，建议修改基础系统，然而，这个基础系统是为一个不同的行业而设计的，并且也不是基于Internet的，因此必须进行实质性的修改。还有一个方案，系统似乎具有所期望的大部分功能，但仅能在UNIX机器上运行，要适应公司的当前环境——Novell网络，也必须花一些工夫进行修改。

Robert已安排在晚些时候会见信息系统的主管Bill Williams，他希望Bill可以就如何解决这个问题给出一些建议。

概述

正如在前面几章中所讨论的那样，系统开发中分析阶段的六项主要活动如下：

- 收集信息
- 定义系统需求
- 设计系统原型，检验可行性并发现问题
- 划分系统需求优先级
- 生成和评估可选方案
- 与管理层一起审查推荐方案

在前几章已经学习了分析阶段的事实发现、定义系统需求和原型设计，也学习了怎样使用传统方法或面向对象的方法来设计需求。这一章将解释分析阶段的后三项活动，即把工程的重点从发现（需求）与分析转移到解决方案与设计上来。这最后的几项活动是整个项目的关键，它们决定了实际系统设计和实现的方向。

首先，我们将讨论项目管理定位。这是分析阶段最后三项活动共同的基础。正如第3章所讨论的那样，项目经理的主要职责之一就是定义新系统并控制其范围。为系统需求划分优先级的目的就是为了精确地定义系统的范围。系统范围的决定将直接影响到项目的费用和进度，这些也是项目经理的职责。另外，对可选实施方案的评估也控制着项目的其他部分。最后几项活动的结果将决定系统最后阶段的详细进度表。

其次，我们讨论如何对系统的需求进行评估和优先级排序。通常在分析过程中发现的需求要比系统应该包含的需求多得多，所以开发组必须将这些需求分类并进行排序，然后决定系统到底应该包括哪些需求。一般来说，可以结合二至三种系统需求，根据它们所需要的资源产生出一种综合的可选方案，然后由监督委员会决定哪种方法最可行。监督委员会通常由主管、用户和技术经理组成。本章将讨论几种不同的方法来进行优先级排序和选择系统的范围和自动化水平。

在本章的另一个主要部分我们要讨论生产环境的各种可选方法，包括硬件配置和操作系统的选择。我们必须要认真考虑新系统现有的或计划设计的环境，即要考虑新系统需要什么硬件，什么系统软件，什么网络，以及是否标准是否支持。本章将对配置和开发环境的选择和约束进行简要论述，也会用示例说明一些关于环境的重要观点，这将放在RMO及其新系统中进行讨论。

再次，本章要介绍设计和实施的一些可选方案。我们将关注那些实际构造和安装系统的不同选项，一旦系统的范围和环境被确定下来，就可以对几种不同的方案进行评审了。这些可选方法包括从建立一个新系统、从别处购买一个系统及外购一部分开发和日常操作等。我们将评审出最好的方案并讨论出做决定的步骤。此外，还将讨论如何进行开发和使用RFP的

一共说明。

尽管这三项活动被认为是在分析阶段的最后活动，但在通常情况下它们是与其他的活动并行发生的。例如，开发需求时，可能在需求被定义的时候对它们进行优先排序才有意义。开发组了解需求时通常也将同时选择操作系统。这些决定需要经过认真的考虑和不断的评估。因此，尽管许多实际的工作可能在事实发现和需求开发时已经完成了，我们依然把它们称做最后且最关键的活动。此外，应该记住，项目管理的预测方法和SDLC允许较早地决定设计和实现，但整个项目期间自适应方法都将不断重估这些决策。

最后，这一章将讨论把结果以专业的格式组织起来并提交给上层管理部门的必要性。在项目中，这是一个逻辑上的里程碑，它用来评定用户需求，更新可行性分析并为项目赢得投入和资金。

8.1　项目管理的前景

如第3章所述，在计划阶段的活动需要项目经理投入大量的精力。本章所讨论的分析阶段的活动也是如此。除了管理部门以外，这些活动还有一些重要的技术需求。因此项目组的项目经理和资深技术人员必须一起工作才能保证这几项活动成功实现。

系统开发项目的规模和复杂程度不尽相同，形式也往往千差万别。对于庞大复杂的项目，一种有效的管理方法是设计出决策度量系统，用它来衡量各种方案，并根据相对得分来对这些方案进行评估。对于小型的且不太复杂的项目则可以使用一些非正式的技术，采用相对较少的度量标准。这一章将讨论几种有助于评估各种方案的技术。

附录A中指出项目管理包括8个部分：范围、时间、费用、质量、人力资源、通信、风险及采购。本章中讨论的活动和它们中的7个部分相关。我们将讨论与分析阶段的各种活动——将系统需求划分优先级、评估各种可选方案，以及与经验丰富的管理人员一起评审各种推荐方案三项活动相关的一些项目管理的任务。

为系统需求所建立的优先级直接影响着项目的范围。在进行项目需求优先级排序时，项目经理就要准确定义项目必须包含的一些功能，并设置一个基准来控制并指导项目的其他部分。为了控制项目的范围和便于项目管理，可以列出一个用户和项目开发人员达成共识的固定的功能列表。如果新系统中应该具有哪些功能一直迟而未决，项目经理要想控制项目的规模几乎是不可能的。

随着项目范围、环境和实施的确定，将走一步确定进度表以帮助管理项目时间。事实上，在许多项目中，只有在做出这些决定以后，进度表才能真正得以完成。例如，当项目组决定为新系统购买一些组件或从外面雇佣一些编程人员时，项目进度表必须反映出这些决定。

项目成本管理包括项目费用的预算及其控制。项目费用和进度表深刻地影响着与项目范围、环境及实施相关的决定。通常，项目经理必须重新计算费用/利润比，从而来保证项目在经济上的可行性。在许多项目中，继续/终止的决定往往在项目经理重新计算费用/利润后做出。

项目经理的另一个重要职责是将发现的问题提交给监督委员会。项目通信管理包括收集并解释所有的重要决定、可行性分析、风险、利润、进度表及项目投资的持股人的花费。

当项目组决策时，尤其是关于环境和实现的技术决策时，项目经理必须确定和评估每种可选方案相关的各种风险。要对每一个即将考虑的方案做出完整的风险分析和可行性分析。由于要做出关键性的决策，进行详细而彻底的风险分析对项目经理来说也就是十分重要的。

当对实现方案进行评估时，项目经理可以开始进行与采购管理相关的活动了。要对提供商做出确认和评估，要开发RFP并且也要对各种建议做出评价。此时，合同谈判也可以开始了。一个好的项目经理必须具备良好的采购技巧以确保建立可靠、专业的提供商关系，并做

出正确的购买决定。

最后，即使某些特定的任务和项目质量管理没有直接的联系，但很显然，质量是所有活动的最终目标。

项目管理自始至终贯穿于整个项目的生命周期中。但在两个阶段项目管理任务最为普遍，一个是初始设计阶段，一个是在后来的分析活动评估阶段。在做出关键决策和确定项目方向时也最能体现出项目经理的经验和技巧。

需求、环境和实现方法相互关联，相互影响的，因此影响它们的决策是一起做出的。在接下来的章节中，我们将对各个话题进行单独讨论，但需要记住，实际中它们是相互交织在一起的。首先，我们来介绍需求和项目范围。

8.2 决定范围和自动化水平

系统需求优先级排序包括定义新系统的范围和自动化水平。范围和自动化水平是与新应用系统紧密相连的两个方面。系统范围界定了系统将包含哪些业务功能。例如，在目前的RMO销售点系统中，范围包含邮件处理和电话销售，而不包含网络销售。自动化水平指的是计算机对所包含的功能可以支持的自动化程度。在新系统中，低水平的自动化电话销售要求接听电话的职员手头有打印好的目录，以便校验用户需求。系统仅支持订货信息的简单数据输入，而较高水平自动化的电话销售将在线得到目录和客户信息。因此职员可以获得自动的数据入口和对库存项目，用户名字和地址信息的自动校验。

8.2.1 控制项目范围

开发项目的一个普遍难题是需求扩充。顾名思义，新系统在定义了需求和制订了决策之后，功能仍可能继续增加。控制这种问题的一种方法是使标识、分类和优先考虑新系统所包含功能的整个过程规范化，使得大家对系统的功能达成共识。在第5章中已经介绍了事件表可以用来标识和定义系统所必须支持的所有业务事件。这里我们继续使用事件表来控制新系统应该支持的功能，这是控制项目范围的一个有效方法。

在分析阶段，用户需要得到的业务功能常常比安排和预算所允许的多。分析员必须确定哪些功能是关键的，必须包含，哪些功能是可以推迟考虑的。确定范围的一个常见的方法就是列出每一个可能的功能需求并衡量它们的重要性，可以使用类似“必需的”，“重要的”，“期待的”这样的等级来进行分类。确定每个功能的优先级通常与对每个功能自动化水平的描述有关。

8.2.2 定义自动化水平

自动化水平是系统为每项功能提供的某种支持的描述。对一个应用系统的大部分功能而言，至少可以划分为三级自动化水平：低、中、高。对于低等自动化，计算机系统提供简单的记录保存，并提供数据输入界面以捕捉信息，并且把这些信息插入到数据库中，而且对简单字段类型的编辑和对数据输入的有效性进行检查。例如，低等自动化水平的订单输入功能将有一个订单输入界面用来输入客户和订单信息，系统日期可以用做订单日期，订单的每一行条目都需要手工输入。系统可能或不能自动地计算价格，通常不检查在库数量和希望的发运日期。等订单输入完毕，数据就存储到了数据库中，订单输入功能也完成了。

分析员也为每个功能定义中等程度自动化。它可以是一个单一的中间点，也可以是各种中间范围的选择。中等自动化通常结合了低等自动化与高等自动化的一些特性。在当前的技术和预算内，分析员们是最大的努力去猜测，什么是必要的，什么是合理的。

当系统接管尽可能多的功能处理时，就是高水平的自动化。通常，系统分析员确定高水平的自动化要比确定低水平的自动化困难。因为低水平的自动化基本上是当前人工操作过程的一个自动化版本。但是，产生一个高水平的自动化系统要求群策群力并考虑"盒子之外有什么"，从而创建新的过程和程序。第4章已经讨论了与业务流程重组（BPR）有关的思想。BPR的思想是重新考虑业务功能的实现方法，以获得处理速度和服务水平上的根本性提高。成功的业务功能重组依赖于提供高度自动化支持的计算机系统。

图8-1是一个既包含RMO客户支持系统（CSS）的范围又包含其各项功能自动化信息等级的表。这个表包含了原始事件表（见图5-16）中的所有业务事件，同时也包含了在系统分析时所标识的7个新功能。这个表的目的是说明新系统所需要完成的所有潜在的事件和功能。每个业务功能以"必需的"、"重要的"、"期待的"这样的等级来进行分类。使用者和客户根据业务需要和新系统的目标来评定功能的优先级。例如，系统的目标之一是增加客户支持，那么就必须具有允许RMO对客户的请求做出响应的功能，而且至少处于一定的自动化等级。

这个表同样也包括了每个功能的不同自动化等级。分析员不需要去描述表中每个自动化等级的各个特征，支持描述将对每个单元格做出详细的说明。这个表为系统的各个功能、各功能的优先级及在不同自动化水平上实现各功能的不同方法提供了一个总体概要。

以RMO的订单输入功能为例。我们首先来确定一下能实现的最好的客户服务。有这样一个问题："为什么顾客需要在他最方便的时间订购他需要的东西？"同样，"一个顾客在完成订购之后关于他的订单包想知道什么？"（通常，我们会重组整个过程，并且会问这类问题："顾客希望什么时候或多久收到他们订购的项目？"但这里我们只讨论订单输入过程这一部分。）

为了回答订单输入的问题，RMO全体员工决定要给他们在线订购的顾客提供所有根据分类目录购买的好处：方便，可以在白天或晚上的任何时间订购、不拥挤、可从家里订购、选择广泛、订购简单并且保护隐私。RMO也打算尽可能灵活地提供商场购买的好处：可以检查条目，可试用和比较尺寸、颜色和样式，靠近可以有许多商品及相关配件，为了匹配和兼容可同时检查几个元件等。

综合上述的想法，一个高端系统应有以下的特点：

- 顾客可以在线看到产品目录，包括真彩色和三维图形。对于高科技产品，目录应包括详细的描述和图形说明产品的结构和特殊性能。这种服务可以通过网络提供给Internet顾客。对于电话客户，可通过点播连接到电视机的电话线提供产品目录。
- 产品目录应是交互的，允许顾客用适当的图形想象力将几种产品组合起来一起显示（如一个模拟人穿着衬衫、夹克和短裤的图像）。
- 用户与目录和订购系统的接口连接可以是声动的也可以是键盘激活的。
- 系统应该同时提供顾客可能需要或期望购买的相关产品的建议。
- 系统应确认所有的产品都有存货并建立一个确定的发货时间（系统的最行实现部分应支持24小时以内或更短的时间发货，保证当天发货更好）。
- 如无存货应立即向制造商或其他供应商定货（系统应迅速将交易发送到其他系统），以保证RMO能在适期将货物发送给顾客。
- 在线实时付款，就像在商店一样。
- 顾客可以通过网络或电话查看所有已经订购的历史记录并给予任何一次订购的状况。

有趣的是我们谈到的所有功能都可以用现有的技术来支持。问题是RMO是否会及时调整这方面的成本。无论如何，我们已为新系统的订单输入部分定义了高等自动化。

功能（从事件表扩展而得）	优先权（必需、重要、期待）	低等自动化	中等（最有可能）自动化	高等自动化（带"+"的表示中等自动化加算性）
检查项目可用性	重要的	现存存货数量的定期列表	实时，内部和Web	+促销活动
下订单	必需的	办事员输入数据	办事员实时和客户通过Web	+促销和出库方案
改变或取消订单	重要的	办事员日夜工作	办事员实时和客户24小时通过Web	办事员实时和客户通过Web直到发运
检查订单状态	重要的	办事员日夜工作	办事员实时和客户通过Web	+自动通知
完成订单	必需的	打印下拉列表和发运标签	下拉列表、发运标签、实时更新	自动仓库，实时更新
产生欠交（退期）订单	重要的	办事员输入数据	实时	+系统自动和通知供应商
返回项目	重要的	办事员输入数据	实时，办事员更新重新进货和客户	自动库存和账户更新
邮寄目录	必需的	打印标签	个性化的封面信件	+完全个性化
更正客户账户	重要的	数据输入	实时	+活动自动化
发送促销材料	重要的	打印标签	个性化的封面页	基于购买历史个性化
调整客户费用	必需的	数据输入	实时更新	+活动自动化
更新目录	必需的	数据输入	实时	+基于销售历史自动建议
创建促销材料	重要的	数据输入	实时	基于销售历史推荐
创建新目录	必需的	保存产品、价格等记录	保存产品、价格、图片、布局等记录	数字扫描和页面布局
系统报表				
生成订单汇总报表	重要的	需要时打印	在线查看和实时	数据可视化工具
生成活动报表	重要的	需要时打印	在线查看和实时	数据可视化工具
生成事务汇总报表	重要的	需要时打印	在线查看和实时	数据可视化工具
生成客户调整报表	重要的	需要时打印	在线查看和实时	数据可视化工具
生成完成报表	重要的	需要时打印	在线查看和实时	数据可视化工具
生成目录活动报表	重要的	需要时打印	在线查看和实时	数据可视化工具
新标识的事件				
维护客户购买历史	重要的	存档汇总报表文件	存档、打印促销通知	自动、实时的销售促销
给制造厂家提供反馈	期待的	打印报表	每日更新	实时和趋势分析
从销售数据提供EDI反馈给供应商	期待的	打印报表和历史记录	每日更新	实时和趋势分析
与发运商系统联系	期待的	无链接	每日更新并E-Mail通知客户	自动反馈和通过Web链接自动跟踪发货情况
运行数据仓库和引导数据分析	期待的		趋势分析	趋势分析、数据可视化工具
自动的销售促销	期待的		基于促销	基于销售促销和销售历史
使用DSS引导扩展销售分析	期待的		打印报表	数据可视化工具

图8-1 有三级自动化水平和优先权的RMO客户支持系统功能

8.2.3　候选方案的选择

当系统功能的优先级已排定并且自动化的等级也经过分析后，项目组就可以评审所有的候选方案了。初步的决定可能是基于单个的需求或其重要程度，但是通常把所有的方案放在一起进行评估。这样可以为建议系统提供一个更加全面、宏观的看法。近些年来，许多公司已经开始独立新系统以便在市场上赢得竞争上的优势。此外，越来越多的公司正将业务转向包括供给和投递的电子商务上来。通过独立更加全面的、战略性的标准，公司可以为他们的新系统制订更好的长期决策。下面列出了一些关键标准：

- 战略规划。开发新信息系统的最初决策通常是长期战略规划的产物，前面曾经讨论过，战略规划既是为了长期的组织战略也是为了运用信息技术来支持组织计划。由于决策考虑到新系统的个体能力，所以战略规划常被当做整体的衡量标准。例如，如果一个组织的长期目标是开发一个供应链管理系统（SCM），此系统提供和供应商之间的自动接口，那么即使在第一个阶段不会利用这些接口，系统设计时也必须支持它们。
- 经济可行性。显然，自动化的水平越高，需要的资金也就越多。通常，开发组得出几组不同的自动化能力和层次，然后项目对这些不同的分组进行预算。随着更加详细的需求信息的获取及开发某种功能的困难的估定，就可以得到一个更准确的费用/利润的分析。
- 计划安排和资源可行性。要在系统中包含更为先进的特性不仅将增加费用，还会延长完成时间。减少这种影响的一个有效方法是以后升级系统。所有的业务软件开发者都采用这一方式。对内部开发这也是一种可行的选择。新系统所包含的功能通常比最终希望的要少，但用户可以从这种新系统中获得经验，而信息系统的员工可以从过去的经验中学习。这样他们共同努力对系统进行完善直到达到希望的自动化水平。
- 技术可行性。必须对期望的方案进行技术可行性的检查，不仅如此，还应认真考虑组织内部是否有专门技术开发和实现这个系统。为了得到更高水平的自动化，经常需要雇佣外部公司或与他们签合同以获得专门技术。通常，在选择那些组织不需使用的最新技术的方案时尤其需要谨慎。一些资金雄厚、资源丰富的公司通常也是如此。对一些边缘项目来说，详细的风险分析是非常重要的。
- 操作、组织和文化的可行性。业务过程中的变化包含着风险，而这种风险是必须加以控制的。自动化水平越高，范围越广，越需要将业务功能及其手工处理流程进行重新组织。利润可能是实质性和巨大的，但是必须提供对这种变化的支持，以维持用户对新系统的热情和义务。信息系统工作人员常常低估改变人们工作程序和活动的重要性和难度，因此让那些有过团体经验的人帮助管理这些改变将是一种很好的想法。

实践指导

可行性因素——经济可行性、时间可行性、资源可行性、技术可行性、组织可行性及它们的日益增加的风险，风险用于评估项目的初始可行性及各候选方案的可行性。 ■

8.2.4　RMO候选方案的评估

到目前为止，RMO对包括功能和系统达到的自动化水平的选择尚在初始阶段。最终的决策取决于开发组为系统实现所定的候选方案。这些候选方案将在后面的章节中阐述。

基于初步的预算和资源可用性分析，RMO的项目组决定新系统将包括图8-1中所有必需的或重要的功能。对每一项功能，项目组都按照希望的自动化水平做详细的分析。自动化水平的基本决定会同时影响几种功能。例如，自动化的三种水平会影响检查项目可用性、下订单、

改变或数销行单等。列在图8-1的三种基本选择是：① 24小时处理输入信息；② 雇员和客户通过Web的实时输入；③ 类似中等水平的系统，增加基于促销乃至客户购买历史的销售宣传。实际上，这三种候选方案可以进一步细分为更多的选择，如Web和非Web、基于促销的销售激励而不是基于客户购买历史。对自动化水平的根本性决定需要与所列的几种功能保持一致。

图8-2列出了各种功能，其中带阴影的是应该包含的功能及其自动化水平。由于一些基本决定影响到几个功能，所以没必要对每一种功能的费用/利润比、操纵效果、技术可行性和资源影响都做详细的计算。RMO管理层不会接受低水平的自动化。目前大多数系统都已经提供了这种水平的自动化。从表中可以看出，RMO为系统的大多数功能选择了中等或最可能的自动化水平。高等自动化不但在软件上而且在硬件处理能力上都要求有一个实质性的飞跃。例如，下订单功能的高端支持要求雇员和基于Web的系统鼓励客户根据促销和购买历史购买其他的产品，这就需要一个巨大的可高速访问的在线数据库。目前，RMO的当前的预算并未包含实现该功能的费用。

在刚刚确定的7个功能中，RMO管理层决定三个功能必须加入项目中。第一个是将维护客户历史的功能加入并用于特定的促销开发中，以便为高端支持做好准备。可行性因素的分析表明这种选择不会大量增加项目费用或延长日程。

第二个要增加的功能是加快更新生产设施存货水平的速度。这种选择的费用/利润分析表明它可以直接减少欠交(脱期)订单和出库数量。

第三个要增加的功能是建立一个子系统，对销售趋势进行更复杂的分析。这种子系统将利用销售订单数据库和基于客户历史的时间序列，因此它是建立在各个客户的相关数据上的。对于这项新的功能，项目组很难精确计算出新功能的费用/利润比值，但是销售经理说服了监督委员会，说这种功能将是RMO未来竞争的关键因素。所以，将采用一部分资源来增加这项功能。由于这个子系统相对独立，因此，为了使它对计划安排的影响最小，项目组将在系统完成后的几个月内实现它。

8.3 定义应用程序配置环境

在开发一个新信息系统时，系统分析员首先要考虑的是应用程序配置环境。**应用程序配置环境**是指新应用系统将要安装到的计算机硬件、系统软件及网络的配置。项目的一个重要组成部分就是要确保定义了应用程序配置环境且配置环境与应用程序需求匹配。在生命周期过程中，分析的主要目的在于详细定义配置环境，便于从中进行选择并为设计的初期提供足够的信息。在设计过程中还要附加详细说明。

应用程序配置环境：新系统的计算机硬件、系统软件及网络的配置。

8.3.1 硬件、系统软件和网络

在计算机应用程序刚出现的那几年，只有一种应用程序类型和一种配置环境：运行在集中式主机上的批处理方式应用程序，使用存储在磁盘和磁带上的文件，同时带有脱机的数据输入设备（例如打孔机）。随着计算机技术的逐步成熟，应用程序类型所涵盖的范围包括如下几部分：

- 在微型机和个人电脑上运行的单机应用程序
- 在线交互应用程序
- 分布式应用程序
- 基于Web的应用程序

功能（从事件表扩展而得）	优先权（必需、重要、期待）	低水平自动化	中等（最有可能）自动化	高档自动化（+意味着中等+）
检查项目可用性	重要的	现有库存数量低于期列表	实时，内部和Web	+促销活动
下订单	必需的	办事员输入数据	办事员实时和客户通过Web	+促销宣传和忠实方案
修改或取消订单	重要的	办事员日终工作	办事员实时和客户24小时通过Web	办事员实时和客户通过Web直接发运
检查订单状态	重要的	办事员日终工作	办事员实时和客户通过Web	+自动通知
完成订单	必需的	打印下拉列表和发运标签	下拉列表、发运标签、实时更新	自动仓库，实时更新
产生欠交(脱期)订单	重要的	办事员输入数据	实时	+系统自动到通知供应商
返回条目	重要的	办事员输入数据	实时，办事员更新重新进货和客户	自动库存和账目更新
标签目录	必需的	打印标签	个性化的封面信件	+完全个性化
更正客户账户	重要的	数据输入	实时	+活动自动化
发送促销材料	重要的	打印标签	个性化的封面页	基于购买历史个性化
调整客户费用	必需的	数据输入	实时更新	+活动自动化
更新目录	必需的	数据输入	实时	+从销售历史的自动建议
生成促销材料	重要的	数据输入	实时	基于销售历史的推荐
维护帐目及	期待的	保存产品、价格等记录	保存产品、价格、图片、布局等记录	数字印刷和页面布局
系统报表				
生成订单汇总报表	重要的	需要时打印	在线查看和实时	数据可视化工具
生成活动报表	重要的	需要时打印	在线查看和实时	数据可视化工具
生成事务日总报表	重要的	需要时打印	在线查看和实时	数据可视化工具
生成客户更整报表	重要的	需要时打印	在线查看和实时	数据可视化工具
生成完成报表	重要的	需要时打印	在线查看和实时	数据可视化工具
生成目录活动报表	重要的	需要时打印	在线查看和实时	数据可视化工具
新定义的事件				
维护客户购买历史	重要的	存档汇总报表文件	存档、打印促销通知	自动、实时的销售宣传
检测生厂家提供反馈	期待的	打印报表	每日更新	实时和趋势分析
从销售数据提供EDI反馈给供应商	期待的	打印报表和历史记录	每日更新	实时和趋势分析
与发运商系统联系	期待的	无连接	每日更新并E-Mail通知客户	通过Web链接自动反馈和发货跟踪
运行数据仓库和引导数据分析	期待的		趋势分析	趋势分析、数据可视化工具
自动的销售促销	期待的		基于促销	基于销售促销和销售历史
使用DSS引导扩展销售分析	期待的		打印报表	数据可视化工具

表8-2 RMO可选功能和自动化水平的初步选择

随着应用程序类型的不断增加，支持这些应用程序的硬件、系统软件和网络也呈现出多样性的趋势。现在，计算机的体积小到手持设备，大到巨型计算机。此外，分析员也可以选择许多支持软件，例如，操作系统（如UNIX和Windows）、数据库管理系统（如Oracle和DB2），组件架构软件和标准（如CORBA和.NET）及Web服务器软件（如Internet信息服务器 IIS和Apache）。现在应用程序软件取决于复杂的结构，包括客户端和服务器端硬件设备、系统支持软件、计算机网络和确保这些组成部分平稳运转的标准。

在选择或者定义配置环境时，分析员需要关注以下几个重要特征：

- **系统需求的兼容性**。用户位置、访问和更新速度、安全和处理容量等需求对环境需求有着深远的影响。例如，高容量处理过程系统（如信用卡支付过程系统）需要可靠的高速网络、强大的服务器，以及兼容的操作系统和数据库管理系统（DBMS）。
- **硬件和系统软件的兼容性**。尽管随着时间的推移硬件和系统软件的兼容性已经得到了很好的改善，但还是存在一些问题。例如，由于Oracle和Sun Microsystems在软件和硬件的开发中经常合作，所以Oracle的数据库管理系统能够在Sun服务器上Solaris（Sun的UNIX的版本）运行良好。同样，微软的操作系统和数据库管理系统能够在使用Intel处理器的计算机上良好运行。确保硬件和系统软件良好的兼容性简化了系统的安装和配置，提高了系统的运行能力且减少了长期的运营成本。
- **外部系统所需接口**。现代应用程序经常与外部系统进行交互。这种外部系统由诸如信用报告代理、客户、供应商和政府等的实体进行操作。实现外部接口需要特定的系统软件，有时还需要特定的硬件。例如，信用报告代理通过基于Web的XML需求或J2EE组件提供相关服务。一个应用程序要与信用报告系统进行交互必须能够支持一种或者两种此类接口，并且包括与这些接口兼容的系统软件。
- **IT战略规划和体系结构计划的一致性**。由于硬件和系统软件有多种选择，所以公司会发现要支持多种不同类型软硬件十分困难，而且需要花费大量资金。大多数中型和大型公司都制定了战略应用和技术结构计划，把资金和人力局限在一定的硬件和软件候选范围上。例如，公司可以选择由UNIX、Oracle、J2EE分布式系统等组成的标准平台，并选择与Sun Microsystems和Hewlett-Packard公司的产品兼容的硬件。虽然这种环境并非对于每种应用程序类型都是最好的，但是使用这种环境将减少了基础设施维护的总费用，并且对某些公司而言最大化了长期的系统兼容性。
- **费用和进度安排**。配置环境可能会随着项目的费用和进度安排的变化而改变，环境的选择要符合IT战略规划和现有系统，这样才能以最快和最经济的方式获得需求、配置和支持。

总而言之，分析员必须定义应用程序配置环境，以确保应用程序符合一定的需求，符合公司的IT计划，在可以接受的预算和进度安排内获取并配置完成。

8.3.2 开发工具

分析员还必须考虑和选择开发工具。开发环境主要包括：编程语言、计算机辅助软件工程（CASE）工具和用于开发应用程序软件的其他软件。通常特定的配置环境限制了开发环境的选择。例如，选择基于Microsoft .NET的配置环境限制了必须使用由微软（如Visual Studio .NET）和少数相关的第三方供应商提供的兼容开发工具。配置和开发环境同样也限制了系统软件的选择（如.NET应用程序需要使用微软服务器操作系统、IIS和SQL Server等软件）。

开发环境：编程语言、CASE工具以及用于开发应用程序软件的其他软件。

通常公司进行系统开发都有自己偏好的语言，分析员也熟悉这些编程语言的特点。随着技术的变化，新的编程语言通常能够提供其他性能，分析员可以从许多开发语言中进行选择——

从结构化语言（如COBOL）、面向对象语言（如Small Talk，C++，Java）到基于WEB的语言（如JavaScript和PHP）。然而，使用一种新的编程语言需要做额外的工作，并且要投入资金为开发团队提供必要的培训。

如果一家公司在CASE工具投入了大量资金，由于CASE通常指定了运行语言和方法，那么使用这个工具将限制将所有新的开发。并且，一些CASE工具还只为一种数据库管理系统和分布式软件标准产生数据库描述和分布式代码模块。

即使公司还没有购买CASE工具，开发工具（如编译器、调试器和集成开发环境）的选择也将受限于目标操作系统、数据库管理系统和组件或网络服务标准等。例如，一个包括UNIX、Oracle和J2EE在内的配置环境，通常使得开发者选择JAVA编程语言和一组相关工具（如Oracle Jdeveloper、Sun ONE Studio或IBM WebSphere）。

如果公司已经使用一种特定的数据库管理系统，这也将限制开发工具的选择。大多数数据库管理系统供应商也提供了一整套开发工具，与不是针对特定数据库管理系统的开发工具相比，这些开发工具能够加速某些应用程序类型的开发过程，例如，Microsoft Access和Visual Basic，Microsoft SQL Server和Visual Studio.NET，Oracle Application Server和Jdeveloper。

总之，应用程序开发环境的选择（特别是操作系统、数据库管理系统和分布式软件标准等）限制了开发工具的选择。因此，一个分析员在决定配置与开发环境是否适合某个特定的应用时，应综合考虑这两个因素。

8.3.3　RMO的环境

伴随着公司的成长，为支持业务功能，RMO已逐步在各地建立起系统环境。目前，RMO有两个主要的制造工厂，这些工厂为三个仓库提供产品。仓库同样也存放其他工厂生产的产品。目前，RMO的制造工厂、仓库、零售店、邮购中心、电话采购中心及数据中心均已联网。

1. 当前环境

图8-3所示为RMO中当前的计算机环境。现在已经有一台主计算机位于休克城的总部办公室，它用专线与生物城域、波特兰、阿尔伯克基的三个仓库分布点相连，还有一条专用线连接到俄州普罗沃城的邮购中心。邮购中心和分布点都直接连接到主机并允许终端实时连接。这种通信技术是建立在专有的主机交易技术基础上的。商品销售与分布系统是用COBOL语言写的，同时将IBM的DB2数据库技术及VSAM文件相结合。这个系统是RMO的员工在外来技术顾问的协助下开发出来的。

位置和部门	设备	连接
休克城—数据中心	主机	
休克城—零售店	客户-服务器	每日拨号
生物城—工厂	当地的LAN	每日拨号
生物城—仓库	中型计算机	专线连接数据中心
生物城—电话订购中心	客户-服务器	每日拨号
普罗沃城—邮购中心	客户-服务器	专线连接数据中心
波兰—仓库	中型计算机	专线连接数据中心
波兰—工厂	当地的LAN	每日拨号
月饼—零售店	客户-服务器	每日拨号
阿尔伯克基城—仓库	中型计算机	专线连接数据中心

图8-3　RMO当前的处理环境

系统使用电话拨号线路连接盐湖城和波特兰的制造工厂。这些工厂也都有自己的局域网系统以支持一定的生产信息。对每日中心库存系统的更新由拨号连接以批量的方式完成。

零售商店有本地的客户-服务器零售系统。这种系统通过收款机来收集销售和财务信息。这些信息也发送到保存在主机的中心账目和财务系统中。这种传送也是每日以批量的方式完成的。

盐湖城的电话订购系统是一种在客户-服务器环境中运行的相当小的Windows应用程序。它由RMO的员工建立。是一个独立的应用程序，与库存及分发系统的其他部分集成得不是很好。每天信息成批地传送到帕克城的系统上。

其他的应用，例如人力资源和通用账目也在帕克城中的主机系统上运行。

2. 建议的环境

与目标环境相联系的许多决策是在战略规划阶段制订的，这些决策为这个组织确立了长期的方向。在其他情况下，随着新系统采用最先进的技术，战略规划也要进行修改。在RMO这个例子中，许多技术决定是在供应链管理（SCM）的初始阶段做出的。由于新的CSS必须与SCM完美的结合起来，所以技术决定必须既和前面的决定相一致，又要和长期计划相统一。

由于环境决定是全局的战略决定，RMO专门开会来讨论技术上的各种方案并做出决定。与会者包括首席信息官Mac Prestin，系统开发主管John MacMurry、项目经理Barbara Hakifax。为了保证能提供各种所需的细节，辅助技术人员也参加了会议。

为了保证所有的参与者都知道潜在的方案，Barbara提出并介绍了图8-4所示的信息。该图表明了潜在的实现候选方案，与为SCM项目做出决策的那张表很相似。表中根据技术类型和中心化程度列出候选方案。前三个候选方案考虑是否为：

- 转移到Internet技术上来；
- 使用内部LAN/WAN技术；
- 混合使用这两种技术。

方 案	描 述
1. 把所有的功能基于浏览器实现（Intranet/Internet）	使内部应用和外部的客户都支持基于Web的浏览器界面。该方案将提供一致性界面并与持续发展的电子商务相适应
2. 使用内部LAN/WAN技术	内部事务处理将更快，数据库不需要与Web打交道，且需要把目录放在Web上
3. 混合方案1和2	使用Web与客户交互，但使用内部LAN/WAN做后台处理。有一个面向SCM和其他网络系统的接口
4. 使用主机作为中心数据点和服务器	支持高容量的事务。可以作为所有系统的中心数据库。提供较高的安全性、控制能力和一致性
5. 在多个服务器上使用分布式数据库	分布式数据库提供快速响应和更高容量负载、增加灵活、升级较困难
6. 使用完全的OO组件，在J2EE对象	该方案可以完美结合各应用程序（SCM、CCM和其他系统），干净地实现RMO在未来的面向对象转移中的位置，但这种方法需要中间件集成软件
7. 在后端关系数据库基础上使用OO作为用户界面	使用Visual BASIC或Java来开发应用程序。使用DB2或关系性Oracle来进行数据库处理。对高容量来说这种方法非常有效
8. 使用OO作为用户界面，外加J2EE对象进行系统间的通信	该方案将RMO转移到完全的OO环境。需要中间件来集成系统

图8-4 处理环境选择

接下来的两种候选方案集中在设备上，是否为：

• 使用主机中心处理器。

• 选择分布式客户-服务器处理器。

最后，要考虑数据库的位置和类型。公司要决定是使用传统关系数据库技术还是转向更先进的面向对象数据库。对CSS所做出的任何决定都要和前面为SCM所做出的决策相一致。

RMO想使用最先进的技术，但又不想要一个高风险的项目，也不想尝试那些尚未被验证的新技术。图8-5列出了RMO战略方向的主要内容。

问　题	方　向
与其他系统的必要接口	1. 自动进给数据到SCM系统 2. 提供到普通分类出计算的接口 3. 为外部系统（如管用检查和发运商）提供自动进给数据的接口 4. 可能转向XML实现通用接口语言
设备配置	1. 使用多CPU的服务器实现前端应用程序 2. 由大型中央处理器提供数据库支持
操作系统环境	1. Windows Server 2003前缀服务器 2. 大型机下的MVS
网络配置	1. Windows网络 2. Web服务器的IIS
语言环境	使用VB，Java和PHP来实现应用程序和网络开发
数据库环境	1. 维持大型机上的DB2数据库 2. 如果使用OO数据库，再评估长远策略
CASE工具	只需使用图表功能，有多种可选方案

图8-5　RMO处理环境的战略方向

目前，经过良好测试的技术可以在多处理器的基础上提供客户-服务器的处理来支持高容量的网络事务。微软的网络服务器IIS可以用来提供网络支持。大型机上现有的DB2数据库是提供高效后台处理的最可行的方案。虽然数据库必须重新设计，并且需要为新系统重新进行构建，但是最基本的处理环境是不变的。

所有用COBOL编写的应用程序都将被新系统所替代，这个新系统将使用合适的Java、VB、VBScript及PHP来进行编写。

采用这种方法，大型机将继续作为中央数据服务器。其他的两个是应用程序服务器，用户将使用连接到应用程序服务器上的个人客户端计算机。Barbara Halifax将图8-5和半月来的情况报告一起交给MacMurty（参阅备忘录）。她同样提出了其他需要决策的公开问题。尽管操作环境的选择决定对这个项目的进程至关重要，其他一些重要的问题也仍需要进行说明。

2007年4月28日

To：John MacMory

From：Barbara Halifax

RE：客户支持系统的状况

感激你参与和支持在上星期举行的关于决定应用程序环境配置的会议，以及你对我们所做的决定的赞同。这些决定和我们在初步计划中所做的假设是一致的。以下是我们目前的情况：

最后阶段完成

我们花费了大量的时间来最后定案系统的范围。我们和每个部门的负责人一起审查了该部门的一系列业务事件（从事件表中），我们也审查了数据模型来确保我们已经标识出所有的必要属性。对新系统的范围和需求我们已经得很好实现，对这点我很有信心。

上周的大部分时间，我都用来准备关于决定应用程序环境配置的会议。

下个阶段的计划

如你所知，下个星期我们仍将有一个重要的会议要召开——做出进行/不进行的最后的决定。我们已经有了定义好的系统规范说明。这个我以后会给出的。我们将完成所有关于可行性问题的重新评估，这些问题包括进度表、费用/利润以及开发项目本身和配置新系统所承担的风险。我已经和伊莉莎白一些监督委员会的成员一起审查了这些问题，所以这里应该没有什么意外的情况。在这个会议上，我们将把项目剩余部分的规定和资金正式规范化并最终定案。

BH

cc：Steven Deerfield, Ming Lee, Jack Garcia

8.4 候选实施方案的选择

到目前为止，我们已经描述了开发项目中的分析和事实发现活动。项目组要决定系统的范围、自动化水平及处理环境，同样，项目组也要决定设计、编程和系统安装的实际实现方法。实现这个目标有许多方法。例如，如果应用十分标准，也许仅需要购买一个支持它的计算机程序或系统。甚至对于更为复杂的系统，其他公司可能也已经开发了可供购买的标准系统。如果不选择购买方案，还可以内部建立系统。即使这样，也还有许多选择。对于一些服务或特殊技术，可以雇佣一些外面的程序员来实施。重点是现在组织必须计划这个项目的剩余部分，而这确实存在大量的选择。

图8-6所示为实现一个系统的不同实施方案。左侧的轴表示构建与购买。横轴表示系统重新开发与外购装项目，每个轴都表示一个连续的区域。例如，可以购买完整的系统，或者完全自己设计解决方法。而介于这两个极端中间的系统就是：部分购买部分自己设计，换句话说，可能购买基本的框架，但是需要进行一些修改，或者对某些部件的接口重新编程使之适应现有的系统。同样，对自己开发或者外购全部或部分系统也有很多选择。

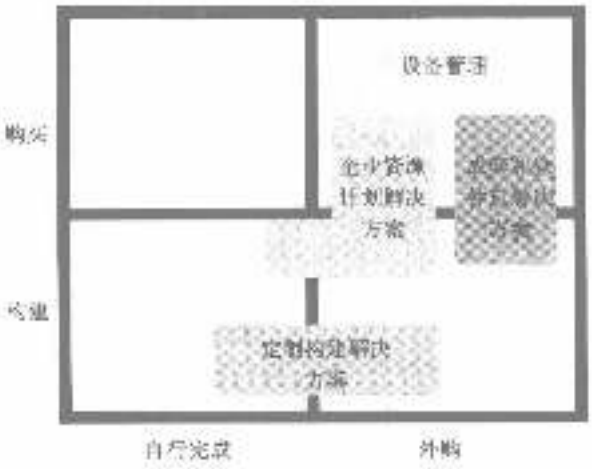


图8-6 实施选择方案

两个轴之间的图形表示建立系统的各种一般性方法。当与另一家公司签订一个包括开发和操作的完整系统的时候，就要进行设备管理了。在它的下面是软件包或成套系统。尽管两者稍有区别——软件包是经过压缩封装的通用的，而成套系统则是一个定制包——它们都要进行一些修改以适应现

有的环境。所以，这些方案通常都有一个“构建”组件。企业资源计划（ERP）方案的开始都需要一个标准的系统，但它们也需要与公司的业务处理进行全面充分的结合。ERP方案和整个组织以及它所有的系统结合得十分紧密，因此通常需要做大量的工作才能实现。定制软件系统需要由内部开发人员或外包顾问和程序员进行程序开发。下面将详细讨论每一种候选方案。

8.4.1　设备管理

设备管理是为整个组织外购所有数据处理和信息支持功能。设备管理不是一个真正的开发技术或实现方案，换句话说，它是一个将所有系统的开发、实现和操作都转交给外部供应商的战略性决策。例如，一家银行可能雇佣一家设备管理公司提供所有的数据处理能力。计算机、软件系统、网络甚至技术人员全部属于这家外部公司。实质上，银行已经选择让另一家公司成为它的信息系统部门。

设备管理：所有数据处理和信息技术在一个销售商那里外购。

外购全部的信息系统的各种功能是一种长期的战略性决策，它将涉及整个组织而不仅仅是单一的开发项目。因此，即使我们把它作为实施候选方案之一来讨论，这个决定通常也并不是任意一个项目组可以决定的。它通常是上层执行者的决策。组织和设备管理供应商之间的合同通常包含一个8－10年的设备管理服务，资金达几百万美元。电子数据系统（EDS）公司资产达数十亿美元，它的主要收入是通过向许多工业部门提供设备管理服务获得的。它支持的行业部门包括银行、保健（如Blue Cross和Blue Shield）、食品、保险、零售和政府部门。在这些不同行业中，EDS可以为这些不同的行业提供高质量的设备管理服务，这主要是因为它拥有一批经验丰富的行业专家。

8.4.2　软件包、成套软件和ERP系统

软件包包含一些用于特定用途的、可供购买的软件系统。严格的定义意味着软件包中的软件无须修改即可使用。我们的个人电脑上都有软件包，如字处理、账目/总分类账软件包。我们买来的执行组件不带源代码，但有文档，安装之后即可使用，不需修改或也不能增加新的功能。买来后我们只能使用它和它的内置选项。软件包的优点在于它运行良好，提供大量的功能但并不昂贵，而且它编制了良好文档说明，几乎没有错误且运行稳定。

软件包：已开发好的软件，可以整包购买。

在组织信息系统的总体战略规划中，软件包占有一席之地。首先，许多软件包可以作为整个项目的一部分提供服务。例如，一个标准的报表系统软件包可以向用户提供报表功能。一般来说，如果可能的话，应该尽力使用一些软件包来完成那些标准功能。

成套系统是由外部公司提供一个完整的解决方案，包括硬件和软件。使用部门只需将其打开即可。在大多数情况下，外部供应商专攻某些特定的行业，为这个行业提供应用软件。差不多几百家中小型公司，大多数都在专攻一些特殊需要的系统。这些成套系统公司在贸易杂志上为一个行业刊登广告。例如，法律公司的法律系统、影像商店的影像系统、医生或牙医的病人记录系统、零售公司的销售系统、建筑公司的建筑管理系统、图书馆的图书系统等。这样的例子举不胜举。

成套系统：包括软件和硬件一起卖给他人的一种完整的解决方案。

成套系统中的一个关键问题是这些系统常常不能准确地满足组织的需求。在这种情况下，组织经常会做大量繁重的工作来对它进行修改，以使其可以和计算机系统相匹配。在成套系统中，为解决这种问题的另一种方法是让供应商定制部分的系统。一个组织通常会购买这个

基础系统，一定数量的定制修改和一个服务协议。供应公司对这个组织特别的需求进行分析，并修改代码。服务协议有效期为几个月或几年，范围涉及简单的错误修正到更大程度的修改。在一些情况下，只提供执行代码，而在其他情况下，执行代码和源代码都会提供给这个组织以便它可以自己修改。有些所有的修改由供应公司完成，有时购买方也可以让程序员与供应商的项目小组合作开发以降低定制成本并获得新系统的经验。同样，也有可能进行大量的组合。这是获得中小型应用软件的一种非常流行的方法。这些应用软件并不完全符合标准。

过去，成套系统只用于组织内部的特定系统。而近几年，一些大公司已经将这种方法引进到企业系统当中。这些被称为企业资源计划（ERP）的系统支持着整个组织的所有操作功能。如SAP和PeopleSoft等的公司已经在向组织引进ERP上取得了成功。显然，如果支持是涉及整个企业的，那么配置将成为主要的任务。许多这样的项目耗时一年以上进行安装，花费达几百万美元。

ERP系统的优势是，相比自行开发系统，它的费用低且风险小。费用低是因为有60%－80%的应用软件已存在于基础系统中，风险小是因为基础系统通常已经开发完成且测试良好。其他组织已经在使用这种系统，因此它有很好的成功记录。

ERP系统的劣势在于，系统即使是定制的，它也可能不能准确满足组织的需求。通常，在组织的需求和所提供的功能之间总有一些差异。所以公司必须修改其内部程序，并尽力让它的用户去适应所提供的这种解决方案。在本书网站上，增补的第2章“Package and Interprise Resource Planning”中对ERP进行了更详细的讨论。

8.4.3 定制软件系统

定制软件系统是指提供商或内部开发人员为组织准确的需求而定做的系统，该系统部分或者全部由外部组织开发。新系统是在系统软件开发周期的基础上重新开发的。然而，开发小组主要是外部顾问。在一些情况下，开发小组全部来源于顾问公司的员工，另一些时候，开发小组则是内部开发成员与外部顾问的组合。

定制软件开发的优势是组织可以购买大量的经验和专门技术来建立新系统。通常顾问公司都有类似系统的开发经验，并且对特定的行业和应用有广泛的领域知识。它也拥有大量有经验的人员去解决复杂的技术难题。另外，为了满足项目进度和开发期限，可以迅速调入大量有经验的技术人员进行项目的开发。外包和合同开发是IS产业成长最快的一部分。

当然，定制软件开发的主要劣势则是费用。组织不仅要支付新系统的开发费用，而且还要按小时付给顾问工资。很典型的例子是，当组织没有内部的开发专门技术或有一个必须完成的挑战性的计划时，它会选择定制软件开发。通常，新系统投资的期望回报必须大大地高于这种方法的费用。一般情况下，定制系统是含有高事务量的大系统。例如，保险系统对其程序要求处理上百万条需求，如果处理一条需求可以降低费用1或2美元，那么由于系统的高事务量就可以迅速节省几百万美元。

大多数大中型公司都有自己的信息系统开发组。实际上，开发组的成员在这样的公司里可以找到很好的工作机会。自行开发的一个主要难题是项目的某一部分可能需要职员经验所不能解决的一些特定的专门技术。对于中型公司更是如此。一种解决方案是让公司人员管理和开发整个项目，而在需要额外专业技术的领域聘请专门顾问帮助解决问题。整个项目的进度和控制仍由组织来维持，并能在必要的时候获得帮助。

这种方法的优点在于对项目和项目小组的知识水平的控制。由此公司人员也会更好地了解组织的内部文化和不同业务部门的特殊处理。另一个主要优点是组织可以通过自行开发系统来建立内部的专门技术队伍。

自行开发的主要缺点是内部员工可能意识不到他们何时需要帮助。有时，这种“这里没有发明”的行为，即“如果我们没有想到它或设计它，那它就是不好的”，常常由于没有使用很好的且价格合理的解决方案，而使开发复杂化。有时技术难题比预料的要复杂得多，而内部人员却认识不到他们需要得到专家的帮助。

8.4.4　选择实施方案

有时选择实施方案很简单，然而，有时在各种方案之间做出选择可能会很困难，尤其是涉及外部供应商的时候。例如，一个解决方案可能包含一些必需的功能，但没有包含所有的功能；另一个解决方案可能包含必需的功能，但只能在某个开发所希望的平台或操作系统上运行；一些解决方案可能为当前问题提供快速而廉价的解决方法，但却对将来的发展只有局限性；另一些解决方案有利于长期发展，但却昂贵且需要长期开发。供应商们可能一个建议采用成套系统，另一个则建议自行开发系统，还有的建议创建一个带有特定的数据库管理系统和平台的成套系统，其他的则建议项目联合开发。如何选择的问题就像是“苹果和橘子”的比较，每个外部供应商都会极力表现自己，因此通常情况下他们提出的解决方案之间几乎没有共同点。系统分析员的目标是就建立一系列共同准则来尽可能一致地比较这些候选方案。

实践指导

记住选择新系统并不仅仅是一个选择“自行开发或购买”或者“外购”的问题，而是要考虑到实现和支持方法有许多综合因素。■

1. 确定标准

开始选择前，必须确定用来比较各种方案的标准。将使用这些标准来比较各种方案，但是各种方案之间的差异可能导致一些标准对于这些方案的适用性也不尽相同，尤其在比较软件包和成套定制系统时所采用的标准或评估方法往往会存在一些差异。例如，对于软件包和成套系统，总有一些对系统的功能性、可靠性和其他一些重要特征有疑问的用户。对于定制系统来说，确定供应商团队的技术能力也许更为重要。对于将购买已有方案与潜在的定制或新的开发相结合的方案，这两种标准都有助于详细审查。

对于外部提供商和内部IS部门所提供的方案还有可能采用不同的标准和评估方法。然而，当针对内部产生的方案应用不同的标准时，对内部提供商有一种潜在的威胁。理论上说，如“供应商可靠性”这样的标准及长期费用应该对内部和外部供应商进行同样的评价。但实际上，由于感觉内部IS人员和部门风险较低，控制较容易，因此一些标准通常被忽略或者得到重视的程度远远不够。由此，项目经理和监理委员会必须对选择标准和评价方法进行仔细审查以确保对内部和外部候选方案进行公平而完整的比较。

在选择实施方案时有如下三个主要的方面可以考虑：

- 适用需求
- 技术需求
- 功能需求

适用需求非常重要但不直接与计算机系统本身相关。适用需求的第一个主要组成部分就是可行性评估，这一步在选择范围和希望的自动化水平部分已经讨论过。每一个经过考虑的实现方案必须满足在可行性分析中所定义的费用、技术、操作和时间安排。下面列出了几个本节中新增的标准：

- 供应商的业绩记录
- 供应商提供的技术支持水平

- 有经验人员的可利用程度
- 开发成本
- 效益的预期值
- 实施前的时间（进度表）
- 对内部资源的影响
- 对内部专门知识的需求
- 对组织的影响（再培训、技术水平）
- 数据转换的成本预算
- 服务的支持和保证（来自外部供应商）

很明显，对组织而言，其中部分标准比其他标准更为重要。例如，在前面的列表中，我们可能只希望向声望较好、稳定且有经验的供应商购买，因此，供应商的业绩记录就显得尤为重要。另一方面，时间进度的安排上有空间，因此，短时间的计划安排就不是很重要了。列表中每一项的相对的重要性可以用一个数字来衡量。图8-7所示为RMO的通用标准和加权因子的一个样表。这个表采用了五点衡量计数法，较重要的标准给一个较高的数字，如5或4，那些不太重要的标准分配一个较低的数字。权值乘以原始的分数就是每个分类的扩展得分。

通用需求 的选标准	权重（5=高， 1=低）	方案1： 自行开发		方案2： 软件包#1+修改		方案3： 软件包#2+修改		方案4： 定制开发	
		原始的	扩展后的	原来的	扩展后的	原来的	扩展后的	原来的	扩展后的
有经验人员的可利用性	4	3	12	3	12	3	12	5	20
开发成本	1	2	15	5	15	3	9	1	3
效益的预期值	5	5	25	3	15	4	20	3	15
实施前的时间	4	1	8	5	20	4	16	2	8
对内部资源是否有影响的影响	2	2	4	4	8	5	10	4	8
内部专业知识需求高度	2	2	4	4	8	5	10	4	8
组织影响最小化	3	4	12	3	9	4	12	4	12
供应商的业绩记录	5	5	25	4	20	4	20	4	20
所供的技术支持水平	4	3	20	1	12	3	12	3	12
所供的服务支持和保证	4	5	20	4	16	4	16	4	16
合计			145		175		137		132

图8-7 RMO通用需求的矩阵

图8-7顶端的四个方案代表不同的实现选择。第一个选择是自行开发系统，第二个和第三个选择是不同的成套系统，它们都是从基础软件包开始的并几对它进行修改。最后一个选择是与一个开发顾问公司签订合同从头开始开发一个全新的系统。这四个选择只是用于演示不同权重的可能值。

功能需求代表系统所必须包含的各种功能。这些在分析阶段进行开发设计的需求在事件表中有定义，并且在数据流图或用例图中都有描述。每个项目都有唯一的一组以系统需求为基础的功能需求。

图8-8所示为RMO客户支持系统的部分功能需求列表，它采用的加权技术与通用需求的相同。

除了功能需求和通用需求以外，每个新系统通常都要有一组必须满足的技术需求。技术需求是系统的约束条件——系统必须在这种约束条件下运行。这个范畴包括了系统所有其他的需求、它的操作方法、性能、功用等。下面列出了在技术需求中需要考虑的一些因素：

- 健壮性（软件不能崩溃）
- 编程错误（软件计算正确）
- 代码质量（可维护性）
- 文档（用户和系统，在线和书面）
- 容易安装
- 灵活性（软件容易调整以适应新的功能和新的环境）
- 结构（可维护，易于理解）
- 用户友好（使用自然而直观）
- 性能（响应时间）
- 可伸缩性（处理大容量的能力）
- 与操作系统环境的兼容性（硬件和操作系统）

功能需求 评定标准	权重(5=高，1=低)	方案1： 自行开发		方案2： 软件包#1+修改		方案3： 软件包#2+修改		方案4： 实施开发	
		原始的	扩展后的	原来的	扩展后的	原来的	扩展后的	原来的	扩展后的
查询条目	4	5	20	4	16	5	20	5	20
创建客户订单	5	5	25	5	25	5	25	5	25
修改订单	4	5	20	5	20	5	20	5	20
访问订单	4	5	20	5	20	4	16	5	20
包装订单	5	5	25	5	25	5	25	5	25
订单发运	5	5	25	5	25	5	25	5	25
创建欠交（发现）订单	4	5	20	5	20	5	20	5	20
接收退还	4	5	20	5	20	4	16	5	20
更正客户账户	3	5	20	3	12	4	16	5	20
更新目录	5	5	25	2	10	3	15	5	25
创建特殊促销	3	5	15	0	0	2	6	5	15
开始保销制度	3	5	15	0	0	2	6	5	15
销售汇总	3	5	15	3	9	3	9	5	15
订单汇总	3	5	10	2	6	3	6	5	10
发运汇总	2	5	10	2	4	5	10	5	10
总计			285		213		235		285

图8-8　RMO的功能需求矩阵

图8-9所示为技术需求可能的权重因子和分数。对于已经建造的候选方案（如软件包或者是ERP系统）可以得出分数。然而对于定制方案来说（如自行开发的项目），这些分数成为新系统的目标。换句话说，由于尚未进行任何开发设计，所以定制方案的这些条目无法评估，但它们确实成为新系统的构建标准。为了对图8-9中的各种方案进行均衡比较，我们为“构建”方案赋上一些数值，这些值是“购买”方案的平均值（这些值用单号标示）。

或许这里最困难的部分就是确定权重因子。客户、系统用户和项目组都应该参与权重因子的确定。我们不仅要考虑每部分每条标准的相对重要性（通用的、功能的或技术的），还要考虑所有主要部分的均衡。换句话说，分级组必须确保通用需求对于功能需求的相对权重能够真正地代表客户的需要。

技术需求 评定标准	权重（5=最， 1=低）	方案1： 自行开发		方案2： 软件包#1+修改		方案3： 软件包#2+修改		方案4： 完全开发	
		原始的	扩展后的	原来的	扩展后的	原来的	扩展后的	原来的	扩展后的
能力性	5	3	*15	3	15	4	20	3	*15
编程错误	4	4	*16	4	16	4	16	4	*16
代码数量	4	4	*16	4	16	4	20	4	*16
文档	3	3	15	3	9	4	12	4	12
安装难易	3	5	15	5	15	4	12	4	12
灵活性	3	4	12	3	9	4	12	5	15
易长	3	4	12	4	12	4	12	4	12
用户界面友好性	4	5	20	3	12	4	16	5	20
总计			126		104		120		121

图8-9 RMO的技术需求矩阵

2. 做出选择

一旦需求确定，每个方案都可以用一个基于满足标准程度的原始分数来评估，原始分数的范围通常是从一个简单的二点计数到更精确的六点计数。例如，一个三点计数可以是：完全满意（2）、部分满意（1）、不满意（0）。一个六点计数可能是：最好（5）、优秀（4）、好（3）、普通（2）、差（1）、不合格（0）。计算每种候选方案每一个标准的权重，使用原始分数乘以权重因子，总分数是各个标准分数的总和，它决定了该分类中不同方案的等级。

RMO决定自行完成大部分的开发任务，正如图8-7所示，在通用需求上，前三名的方案得分非常接近，自行开发方案略高一点，在图8-8中，方案1和4得分相同，二者比方案2和3要好一些。在图8-9中，显示了技术需求，方案1，3，4非常接近。因此，从总体上来看自行开发方案确实有些优势。RMO的内部系统分析员和技术员在前面已经多次证明他们可以处理复杂系统。另外，这种方法可以使RMO建立自己的专业技术队伍，而且虽然选择自行开发，也不反对在必要时聘请专门技术人员。

RMO有足够的内部技术人员来开发系统的网络和数据库部分。基于Web的开发也将自行完成。但是RMO可能不得不雇佣几个专门技术人员。既然RMO希望拥有公司内部的专业技术人员，聘请几个新的专家也不失为一种可行的方法。

许多集成问题，如新的客户-服务器结构与主机系统的集成，可能变得非常复杂。RMO中有一些经验丰富的顾问人员，公司预计需要几个这样的人来管理项目的这个部分。

RMO职员现在已经准备好继续进行CSS项目，可行性的检查确认没有发现什么大问题。这个项目仍在计划和预算之内。公司内部的态度非常积极。由于有这些可用的额外资源，这个项目在技术上看起来也是可行的。

8.5 与供应商签订合同

在RMO事例中，自行开发是已选择的实现方案。为了得到必要的信息来评价其他方案，客户支持系统小组给每一个预期的供应商发送一个正式的RFP。

8.5.1 生成RFP

如前所述，建议需求（RFP）是一个发送给供应商的正式公文。它的基本目的是陈述需求并征求满足这些需求的意见。在政府交易中普遍使用RFP，甚至在工业中RFP的使用也相当普遍。项目经理主要负责完成RFP，评估提交的建议和在公司与供应商之间签订的合同。

建议需求（RFP）：一个规范化的文档，它包含系统需求的详细说明，发送给供应商并要

求供应商对提供硬件、软件和（或）支持服务进行投标。

通常RFP也是一个法定的公文，特别是在政府购买行为中，供应商要参照RFP中规定的信息和程序办事。也就是说，他们必须能够一致、完全地遵守所述程序的按预期进行投资。RFP通常被看做是提供一个协议，而供应商的响应则表示它接受这个协议。

一个好的RFP包括对一个组织的信息需求和必须满足的处理需求的详细解释。第4章中把信息系统需求定义为功能和技术需求组成。一个好的RFP将提供这两种类型的系统需要的详细解释。如果早期的项目假设表明，对于新系统来说最可行的选择是购买成套的解决方案或外包一个客户开发方案，则分析活动的主要任务就是开发一个RFP。当选择外部公司时，它将能确保在详细设计或定制开始之前获得深层的领域知识。

要开发和分发一个好的RFP，购买方必须做大量的分析，通常，这对内部有信息系统职员的公司来说不成问题。然而，对于内部没有信息系统职员的公司，决定处理需求就是一个很常见的问题。要产生一个有意义的RFP或评价各种购买方案也是很困难的。而一些小公司往往忽略这个问题，它们试图从一些无知的角度出发做出最好的决定。明智的方法是雇佣一个不参与开发的顾问，来帮助建立选择标准并决定提供解决方案的供应商。选择最终的供应商的标准同样也可以用于选择独立的顾问。

图8-10是一个普通的RFP的大纲。显然，每一个RFP都必须经过加工以适应组织的专门需要和工程的需求。RFP的第一部分包含下面的第一条和第二条描述了公司的背景信息及新系统的需要。接下来从第三条到第五条详细描述新系统所必须满足的全部需求。在这个例子中，我们可以将需求分成技术需求、功能需求和通用需求三类。第六条需要了解有关供应商的背景以及资历的信息。最后两条，即第七条和第八条，说明怎样提交建议以及如何对其进行评估。

RFP应该明确指出提交合法建议的程序要求。如有可能，应该在给出合法建议提纲的同时，对每一部分具体内容也给出一定的陈述。另外，RFP应该明确指出问题、建议发送和其他重要事件的最后期限。

需求建议目录表

I. 简介和背景
- A. 公司背景
- B. 工业商业标述

II. 需求概述
- A. 商业需求描述
- B. 期望的商业利润
- C. 系统需求概述

III. 技术需求概述
- A. 操作环境
- B. 性能需求
- C. 兼容、接口和兼容性
- D. 质量规格
- E. 扩展和增长需求
- F. 可维护性需求

IV. 功能需求概述
- A. 主要功能说明
- B. 输出信息说明
- C. 用户界面说明
- D. 可选功能与改进系统

V. 通用需求
- A. 维护与支持
- B. 文档和培训
- C. 培养发行
- D. 其他合同需求

VI. 需要了解的供应商和项目信息
- A. 需要工作和项目进度的说明
- B. 需要供应商的参考列表
- C. 需要项目人员的信息

VII. 提交建议的细节
- A. 时间上的要求
- B. 格式上的要求

VIII. 评估标准和过程
- A. 评估的期望时间表
- B. 对技术需求、功能需求及通用需求的评估方法

图8-10　需求建议目录表示例

需求陈述组成了RFP的主要内容。RFP的主体内容可以形式规范化并如前所述来陈述要点。

需求可以分为必要需求和可选需求（或者说是协商决定的部分）。这种分类是以前讨论过的列入优先地位的更正式的版本。RFP也应该明确指出评判标准，例如要被评估的类别及其相应的权重因子。

8.5.2 基准评价和选择供应商

一种评价外部供应商提供的系统质量的方法是在应用中观察，并安装在一个实验平台上并进行测试。然而，不管这种方法多么高效，都无一例外的是昂贵和主观的。数据格式、表格甚至平台可能都与已有的结构相异。如果系统复杂，人们必须经过培训才能使用它。此外，还必须有资源可以用来进行测试。虽然这种方法可能会相当昂贵，但它比由于错误的决策而付出的代价要小得多。

一些应用程序要服从较为严格的评价。这种评价称为基准评价。基准评价即在实际处理条件下用确定的硬件和系统软件对应用软件（或测试程序）进行性能评价。过去的几年，由于昂贵的硬件和软件系统结构以及安装的时间和费用等原因，基准评价执行起来经常很困难。目前，由于较廉价的硬件、流线型安装程序和供应商之间的激烈竞争，这些问题没有那么困难了。

基准评价：用某些标准对系统的评价。

测试应用中的系统的另一个方法是参观其他公司。有时供应商已经有安装在自己设备上的演示版本。作为一个潜在的购买者，你将有机会到供应商的工厂对系统进行测试。供应商也有正在使用这个系统的老客户，去了解这些老客户是个好方法。虽然不能用自己的数据测试系统，但可以观察其他公司如何使用这个系统。也可以与以前的客户交谈了解一些关于他们使用这个系统及与供应商打交道的经验。获得一些参考信息并与一些曾经和供应商打过交道的公司交流总是有益的。

通常，随着环境的改变，公司希望给软件添加一些功能。为此，公司必须自己升级或让供应商提供相应的升级与维护。组织应该确定供应商有持续的研究和开发水平。这种新功能在定制后怎样与现有的系统相互兼容？有没有一个积极的用户群体能够对供应商提供新的增加和修改意见？公司正在进行一项主要投资，从长远打算来考虑这项投资是十分重要的。在任何新系统中的最大的投资是长期的维护费用。好的供应商会通过提供升级支持和根据现存客户中反馈来的信息获得的新功能，从而帮助你平衡维护费用。

8.5.3 制订合同

一旦最终确定哪个建议能够提供最好的解决方案和价值，就可以制订合同了。合同的制订和协商通常需要项目经理、法律顾问及其他一些资深的主管一起合作完成。项目经理应该参与制订合同，以确保合同符合项目的需求，以及那些重要的性能和终止条件都被包含在内。

合同可以分成几种不同的类型，这使购买者和供应商都可能承担一定的风险。固定金额的合同使供应商的风险最大。当供应商假设项目延迟超限时对于采购公司来说是有利的。然而供应商通常设置较高的价位来弥补这种风险。成本加成本百分比合同使购买者承担了风险。实际上，由于供应商的收入直接来源于项目费用的比例，成本外百分比合同会刺激供应商投入更多的资金。中立的情况是两者同时承担一定的风险，叫做成本加固定费用或成本加激励合同。在这种情况下，当工程尽快完成时，供应商和购买者都获益。由于有固定的费用，如果项目尽快完成，供应商的利润空间就会变大。如果项目拖延，则其利润空间也就会相应地缩小。

8.6 提交结果并做出决策

在本章中所描述的调查和分析的结果通常被归纳成一个书面报告并向上级提交，同时要进行口头陈述。其听众是监督委员会的主管，他对该项目有决定的权力和投资的责任。报告和陈述的目的是提供必要的背景，以使做出正确的决定。

项目组（包括技术人员和用户成员）的责任是做详细的调查和计算，对所有的方案都做出正确的分析，然而最终选择哪个方案或哪几个方案的组合会由监督委员会的主管来决定。委员会不仅控制预算，提供资金，而且还负责公司的全局战略指导。

也许对于项目组来说，一个较困难的任务是用完整、正确且易于理解的简单方式来编辑、组织并提交方案和一些关键问题的。监督委员会通常由一流的企业领导组成。他们一般不是技术专家，但他们需要做出影响整个组织的决定。因此，提交结果是项目组将要做的最复杂的任务之一。这需要仔细地考虑以找到细节的平衡点。一个极端是技术细节太复杂，以至于监督委员会不能理解或不能跟上这种逻辑，从而变得不耐烦并失去信心。另一个极端是建议书没有足够支持的细节和逻辑。

不同组织之间的表达形式是不同的。有些公司需要非常规范的书面报告和口头表述。有些公司则不需要书面报告，只需要在客户（项目的出资人）和项目组领导的非正式讨论即可。组织越小越不正式，而大公司通常有标准的建议政策和程序。

文档和报告的格式随阅读者的要求会有很大的不同。附录D（附录D可以从www.course.com 网站上关于本书的页面上的Student Downloads链接下载。）给出了如何设计这种表述的一个详细描述。通常，书面文档与表述有相同的格式。

小结

本章解释的一些活动主要由项目经理负责。项目的焦点从发现需求转移到开发系统的解决方案。因此，描述的活动对系统转移侧重点来说是很重要的。显然，项目经理将对这方向的转移负主要责任。这些活动包括了附录A中所描述的8个项目管理知识部分中的7个。

在分析阶段的一个重要活动是在范围和期望的自动化水平的基础上优化系统需求。新系统的范围决定了系统要支持哪些功能。自动化水平是对所选择的功能自动化程度的一种衡量。高度自动化的功能需要有复杂的计算机系统，如专家系统用来帮助完成一些业务上的功能。

应用程序配置环境是指新系统运行的计算机硬件、系统软件和网络的配置。它决定了系统开发方案的限制条件。分析员必须定义一种环境或者多种环境的选择满足应用需求，符合组织的战略应用和技术框架计划。

另一个与优化需求相关的活动是决定可能的解决方案，然后如何从中选择一种可选方案。实现方案包含自行建立系统、购买软件包或成套软件解决方案，或与开发商签订合同建立（外购）等。当选择外包后，就要开发一个建议需求（RFP）并发送出去。然后评估RFP与期望需求的相符程度。在不同的解决方案中做出选择是一个需要慎重考虑的过程。评估时可能需要考虑以下因素，如建议的系统是否与功能需求和技术需求相符合，以及提交建议的供应商的信誉和成果记录等。

分析阶段的最后任务是开发推荐书和书面表述并提交给管理层以进行决策。分析阶段以后，将制定出一个关于方向、费用、可行性和项目其他部分的解决方法等因素的更具知识性的决策。系统分析员负责将分析阶段的结果整理成文档并以逻辑方式递交给负责分配资金决策的经理主管人员。

关键术语

application deployment environment	应用程序配置环境
benchmark	基准评价
development environment	开发环境
facilities management	设备管理
packaged software	软件包
request for proposal (RFP)	建议需求
turnkey system	成套系统

复习题

1. 什么是应用程序配置环境？为什么在考虑开发方法时它是很重要的？
2. 列举并简要描述当选择或定义配置环境时一个分析员要检验的特征。
3. 解释应用程序配置和开发环境之间的关系。
4. 解释设备管理的基础。
5. 解释自动化范围和水平的差别。
6. 构建－购买（build-versus-buy）决策的含义是什么？
7. 定义软件包方案。解释在软件包方案中什么是必须有的。
8. 什么是ERP？ERP方法对得到一种新的解决方案有何影响？
9. 什么是外购？它如何影响一个项目？
10. 给出基准评价的定义。为什么它在选择一个新系统时是有用的？
11. 什么是RFP？为什么在分析阶段的后期而不是早期开发它？
12. 通用需求、技术需求和功能需求之间的差别是什么？

思考题

1. 购买软件包方案的优点是什么？缺点或风险是什么？
2. 从头开发一个项目有什么优点？缺点是什么？
3. 外购一个开发项目有什么优点？缺点是什么？
4. 讨论在评估解决方案时开发一种正规的技术和特定标准的重要性。
5. 给定下列叙述，确定包含在这个系统范围的功能，并确定每一个功能的自动化水平。这个问题的目的是给你一个去创造性思考的机会，特别是在要确定每个功能的高层自动化选择时。

   会议协调者公司（CC）是帮助其他组织或公司协调和组织会议的公司。它提供如下服务：设计和打印小册子，处理参加人员的登记表，回答回答参加人员提出的问题，检查会议厅或宾馆客房的安全，计划会程之外的活动。CC从两种渠道得到业务：通过预先了解某个公司要举行会议或让公司直接与CC联系。联系建立之后，将会询问客户一些希望了解的基本信息：城市、日期、期望的参加人员数、价格范围、期望的外部活动。从这些信息中就可以准备一个标的。CC希望它的周转时间在五个工作日以下。每个项目分配一个项目经理，他从准备标的的员工处收集信息。如果需要，他可以从城市旅游中心查询信息。
6. 决定一个建议系统中功能需求的权重因子的重要因素有哪些？
7. 列出决定建议系统中通用需求和技术需求权重因子的重要因素。
8. 给出以下不同技术需求的矩阵，如图8-11所示，为一个小型的管道设备供应商的库存管理系统设计你自己的权重因子。正确调整权重因子，扩展原始分数到扩充列并计算总值。你选择哪一

个？正确使用你的选择：是严格按分数选择，还是还存在其他可能的因素要考虑？如何处理一个没有给定的分数。是给一个其他数的平均分数，选择最好的分数或猜测一个分数，还是干脆打0分？（原始分数使用6分制。）

类别	权重因子	方案1：自行开发		方案2：购买成套系统		方案3：购买软件包	
		原始的	扩展后的	原始的	扩展后的	原始的	扩展后的
能力性		5		5		3	
编程销售		5		4		4	
长期成本		5		4		5	
文档		4		4		3	
安装难易		5		5		4	
可维护		5		4		3	
用户界面友好性		5		5		5	
总和							

图8-11　最符合设备供应商的技术需求矩阵

实验练习

1. 假定高容量支付处理系统的配置环境由以下几方面组成：
   - 运行在IBM S/390主机上的OS/390操作系统下的DB2数据库管理系统
   - 运行在IBM zSeries 900主机上的Z/OS操作系统下的WebSphere应用服务器
   - 由Java编写由其他内部和外部系统执行的J2EE应用软件

   针对这个配置环境，研究可行的开发环境，描述它们的优缺点，并推荐一套特定的开发工具。
2. 对一个使用信息系统的组织做一次采访，询问软件系统的RFP的例子，确定RFP的各部分。与本章讨论推荐的组成部分做一个比较。
3. 从新闻文章或Internet资源，找一个安装了ERP软件包（SAP、Oracle或其他）的公司的例子。如果可能，取得整个项目规划的一个复本并分析不同的活动。把它们与标准SDLC做比较，做出这个项目的总预算。
4. 为RMO开发一个RFP以发送给不同的供应商。
5. 为RMO设计一个推荐实施方法，并为你的推荐书设计一个递交给高层管理人员的书面介绍。
6. 留心查看一些业务杂志（Software、CIO、Datamation、Infoweek等），找到曾对供应商做过评估的公司的例子，描述它们的方法并评论一下它们的优点和缺点。

实例研究

热带鱼销售的RFP

你已经学习了这一章，回过头去复习一下本章开头的实例：热带鱼销售。你的工作是为Robert和Bill提供一些关于如何评估不同RFP的特定的建议。

假设你可以建立一些矩阵来衡量不同提议的相对优势，评论一下严格依靠数字进行评估的适用性。换言之，假设Robert和Bill能创建标准和权重来衡量不同方案对各公司的利益。

1. 你认为可能累加所有的分数并且仅基于此做出决定吗？解释你的答案。

2. 除了在表中的权重标准，Robert和Bill在做出决定时还要考虑什么类型的因素？这些其他因素是否能向合格的标准一样对决策产生影响？

3. 如果有多个方案的分数十分接近怎么办？Robert和Bill可能还需要考虑其他什么因素？

房地产多编目服务系统

考虑在第5、6、7章中开发的房地产多编目服务系统的需求。假如你是项目经理，为该房地产服务公司所聘请的咨询公司工作，只负责进行调查和分析阶段的活动。

1. 假如系统用户和系统所有者非常希望"任何时间任何地点都可以访问系统。"讨论他们的要求意味着对系统范围造成怎样的影响。假定系统用户和所有者的优先权，你是否要准备一个如图8-2类似的表格？为什么？

2. 讨论任何时间、任何地点的需求对应用程序配置环境的具体含义。为了完成这一要求需要何种硬件、网络和软件结构？

3. 研究房地产多编目服务系统软件包和成套系统的可用性。搜索Internet及房地产贸易的杂志和网络站点，讨论选择软件包和成套系统优缺点。

4. 开发一个涵盖软件包、成套系统和定制开发系统的RFP。写出涵盖上述三方面的RFP的难点是什么？谁来参与评价RFP响应？

对落基山运动用品商店实例的再思考



不同的应用程序配置环境其实对RMO战略规划来说都是适用的。工作人员现在的考虑更倾向于使用微软Microsoft的解决方案，即微软的IIS作为Web服务器，然而，有Apache服务器的Linux系统提供另一种大型的服务器基础。考虑到RMO或许也能采用这种方案，进行如下工作：

1. 描述使用Apache/Linux的可行性配置。

2. 比较Microsoft和Apache/Linux的所占市场份额（从http://www.netcraft.com开始比较好）。

数据库是RMO另一个潜在的争论点。当前的决定是保留大型机和运行DB2，这是个非常高效的关系数据库。然而，另一种方案是采用Oracle数据库。在市场中，Oracle的地位也是很牢固的。为实现这两种给出的方案，操作如下：

- 比较两种方案在市场上所占份额；
- 列出每种方式的优缺点，DB2大型机的优缺点及Oracle运行在多处理器服务器上的优缺点。

关注Reliable Pharmaceutical Service



假定Reliable公司已经完成了一整套系统需求分析（在第4章－第7章中的案例研究中已经学到），现在，管理人员面临着选择系统范围和运营方法的任务。为了汇总候选方案，要准备如表8-10所示的表格，这个表格把需求分成功能子集，如果软件是定制开发的，那么评估每一个功能的设计和运行所需时间，并根据软件的复杂性、技术的完备性和需求的确定性对每个功能的风险进行分类。

高层执行经理已经对上述表格进行了评估，并且决定所有的功能都需要高优先级。该项目对于恢复赢利和维持市场份额非常关键。Reliable公司人人都记下本行业的技术曲线，它需要现代化的方法来减少成本和提供预期的服务水平。不幸的是，这些功能的交叉和相互依赖性使得只实现一个功能子集是很困难的。执行经理倾向于在一个项目中实现所有的功能，但是他们认为在一个项目中实现所有功能需要的时间太长了。

功　能	项目时间	风险
库存和购买	9个月	中等
完成订单（手动输入数据）	6个月	低
基于Web的订单输入	9个月	高
处方报警	12个月	高
账单	18个月	高

该系统的重要组成部分，如库存、购买和处方报警等同零售店和大型医院及卫生保健机构的内部药房使用的系统大致相同。但是在Reliable系统的需求中还是有些显著的区别存在，如订单输入、产品传送和账单等。只有少数的大型供应商和一些小型供应商专门研究药品系统。

假定管理者已经确定了进行系统开发或者需求获取的下列选项：

- 与供应商签订合同，要求修改处方软件系统包以适应Reliable的需求。
- 与供应商签订合同，购买处方系统的通用部分，并用定制开发软件的形式扩充系统，解决Reliable特有的需求。
- 与系统开发公司签订合同，订制系统，可能使用一些现成的组件来实现库存管理和处方报警。

Reliable公司的执行经理给你分派了如下任务：

1. 开发RFP大纲，描述执行经理所确定的上述选项。列举和简要描述每一个通用需求，技术需求和功能需求。

2. 假定你通过使用传统的或者面向对象的方法开发了一整套的分析文档（超过100页），这些文档是否应该包含在RFP中吗？为什么？

3. 为评估RFP响应开发类似于图8-6、图8-7、图8-8的矩阵。

4. 开发需要接收RFP的供应商的列表。

参考资料

Scott E. Donaldson and Stanley G. Siegel, *Cultivating Successful Software Development: A Practitioner's View*. Prentice Hall, 1997.

Ralph L. Kliem and Irwin S. Ludin, *Project Management Practitioner's Handbook*. American Management Association, 1998.

Sanjiv Purba, David Sawh, and Bharat Shah, *How to Manage a Successful Software Project, Methodologies, Techniques, Tools*. John Wiley & Sons,1995.

John J. Rakos. *Software Project Management for Small to Medium Sized Projects*. Prentice Hall,1990.

Kathy Schwalbe, *Information Technology Project Management, Fourth Edition*. Course Technology,2006.

Neal Whitten, *Managing Software Development Projects: Formula for Success*. John Wiley & Sons, 1995.

第三部分 系统设计任务

第9章 进入系统设计

学习目标

阅读本章后，你应具备如下能力：

- 讨论系统开发生命周期（SDLC）中设计阶段关键问题和协调问题
- 解释主要硬件和设计层次
- 描述各个设计阶段的活动
- 描述常见的开发环境和相应的应用程序结构
- 开发一个简单的网络图并评估项目资源需求

本章要点

- 理解设计要素
- 设计阶段关键活动
- 项目管理：协调项目
- 开发环境
- 应用程序结构
- 网络设计

FAIRCHILD PHARMACEUTICALS：一个生产系统的最终结构设计方案

James Schultz是Fairchild医药品公司的暑期实习生。他被安排参与一个正在开发的生产调度与控制系统的设计工作。两周前James加入进来时，这个项目已经进入分析阶段的尾声。

James为项目经理兼首席分析员Carla Sanchez工作。在过去了两周中James一直跟随着Carla完成了最后设计阶段的各项任务。他帮助Carla准备陈述材料以便向监督委员会和系统用户讲解。在很短的时间内，James收集了大量与项目有关的材料，但是一些细节问题和项目的总体方向尚未在其脑海中形成完整、连贯的概念。昨天，监督委员会停止了当天的工作，因此，今天早上Carla让James来她的办公室来讨论一下他下个阶段的工作安排。

James敲门问道："现在你有时间吗？还是我过一会儿再来？"

"我现在有时间"，Carla回答说，"进来坐吧！让我们从回顾昨天会议的结果开始，我会回答你提出的任何问题，然后我们将细化你在今后几周内的工作任务。"

James说："我有两个有关的问题。首先，哪些实现细节已经确定下来了？哪些还没有？在与用户与监督委员会的讨论中给我的留下的印象是，我们已经决定采用一个成熟的基于网络的系统了，然而基于网络的系统所需的基础设施目前我们还没有，是不是？"

Carla回答说："部分设施已经就位了，但大多数还需要设计、购买。在讨论这个问题之前，

我们还是听听你的另一个问题吧。”

James继续说：“好吧，那接下来我们需要做什么呢？也许你已经料到我的这个问题了。现在看起来好像有不少重要的任务需要着手开始，例如，选择合适的系统软件来支持Web服务，决定公司的网络需要进行哪些改动，还有就是设计数据库。但是，我怀疑自己忽略了一些重要的环节。此外，这些任务、决策之间相互依赖、联系得如此紧密，我都不知道从何处下手了。”

Carla笑了笑，然后说：“好的，看来你在所有的会议期间都非常清醒！你能发现自己对项目中的一些关键问题产生混淆，这一点真是难能可贵。在任何项目中，从分析阶段过渡到设计阶段都是重要的且存在不稳定因素的一步，我们这个项目也不例外。我们需要将详尽的用户需求转变成可以满足这些需求的系统的精确设计。你也已经看到了，许多重要的决策需要尽快做出，它们之间彼此重叠。此外还受限于时间、预算、现有系统、技术以及基础设施等一系列因素。”

James看起来放心了些，问：“那么接下来做什么呢？我从哪里入手呢？”

Carla回答说：“下一步任务叫做结构设计。这里我们将完成所有大的架构的决策，例如，使用哪些硬件来支持新系统，采用那种操作系统，如何存储访问数据，使用哪种语言和工具。有些问题已经在项目开始时简要地说明了，有些决策则已隐含在昨天监督委员会批准的开发环境和自动范围的选择中。我们现在要做的就是将问题罗列出来，确保它们之间以及它们与现有系统能力之间协调一致，并为这些问题各自分配相关的任务。”

Carla继续说：“昨天我花了一下午的时间把这些工作分成几个大类，它们包括硬件与操作系统、Web服务支持、数据库设计、应用软件设计和用户界面设计。我总结了目前我们已经做出的选择以及一些尚待讨论的问题。在本周剩下的几天中，核心人员将组成一个小组来研究各部分的问题并设计出系统结构。例如，我们将决定是通过扩展现有的数据库来支持新系统还是使用新的DBMS来重新开发数据库。到这个周末，我们将做出全部关键的决策，确保它们之间相互协调，并为处理各个部分做出人员分配以及时间进度的计划来。从此以后，各部分的工作便可以并行展开。Professor Chen告诉我你做了一份关于Web服务支持软件的独立研究报告，是吗？”

“是的”，James回答说，“我对比研究了使用CORBA、Microsoft.NET和Java 2 Web服务的通信协议以及基础设施需求。对每种技术我都做了深度调研，并访问了由每种技术支持的两个站点来考察它们的实际运行情况。”

“好”，Carla说：“那些知识我们用得着，因为我们需要决定新系统是否基于Web服务，如果是的话，采用什么样的基础设施以及使用何种开发工具就成为一个新问题。我想，跟着我在接下来的一两周内你会学到很多东西，那时我们将完成结构设计的工作，一旦我们开始进行细节设计任务，我们将根据你的兴趣和能力交给你一个合适的任务。那时将会有许多有趣的工作可供选择，还有大量的工作，够你忙上一两个月的。”

概述

第8章中，我们描述了完成分析阶段以及开始从分析阶段转化到设计阶段的一些活动。本章，我们将完成这一转化，并讨论与新系统设计相关的问题。在分析阶段，需要着重考虑的是系统做什么，即系统的需求，而设计阶段，我们的着眼点是系统如何构建，即定义系统的结构组件。很显然，确定义系统范围和确定需求优先级这样的活动在分析阶段就该完成，然而像定义应用程序配置环境和自动化程度等活动则应在分析阶段开始，并在设计阶段结束。因此，我们从上一章开始转向设计阶段。在本章则完全开始考虑如何进行设计。

本章是讨论设计的6章中的第一章。在本章中，我们将简要地描述所有阶段的活动并详细讨论第一个活动。在后面的章节中，我们将同时使用传统的和面向对象的模型与技术来说明其他设计阶段中的活动。

9.1 理解设计要素

系统设计是从构建新系统的角度来描述、组织、构造系统组件的过程。这个过程分为两个层次：一是结构设计，二是细节设计。系统设计就像建房子时所用的一套设计图纸。这些图纸包含了房子的各个不同组件，如楼层、墙壁、窗户、门、电线、管道以及其他小组件。系统设计和设计图纸做相同的工作，不同的是，这里所指的组件是新系统的组件。我们将设计并详细说明解决方案中的各种不同组件。

为了理解设计的组件，我们必须考虑以下两个问题：

- 系统设计需要哪些系统组件？
- 设计过程中输入输出是什么？

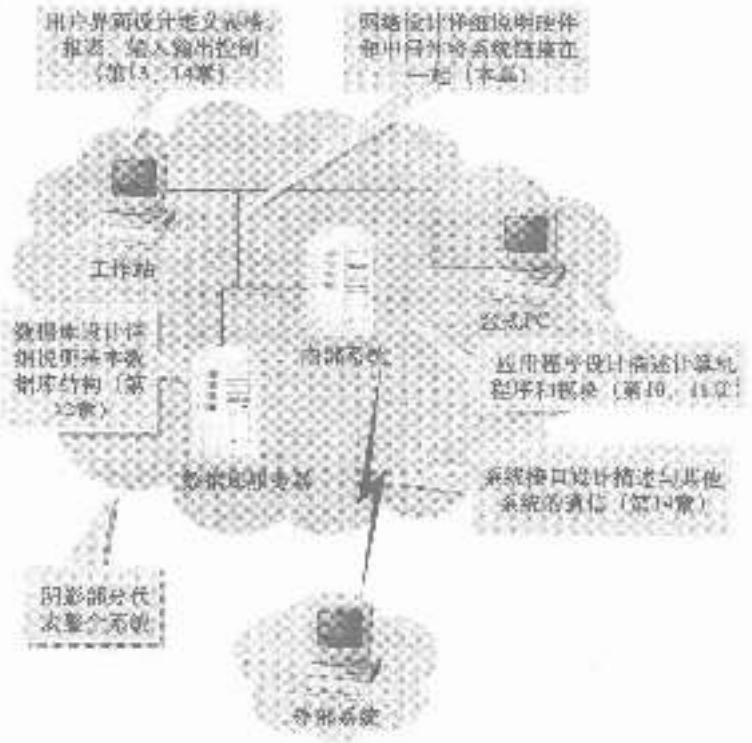


图9-1 系统设计所需的系统组件

9.1.1 设计的主要组件和层次

在进行设计时，要一下子设计出整个信息系统是很复杂的，因此分析员首先需要把整个系统分成几个主要的组件。图9-1描述了这些不同的组件是如何协调在一起的。其中的方块图标表示硬件，硬件里面是软件部分，云团表示一个完整系统，各种图标表示系统的各个部分，这些部分只有协调工作才能使系统运作起来。信息系统的专家们必须确保他们为用户开发出完整的解决方案。如果不能提出一个整体的、完善的解决方案，你们便不能成功地完成工作。

在下一节我们将会看到，SDLC设计阶段的各种活动能够把最终的系统分成各个设计组件。设计阶段的每个活动都将致力于设计出如图9-1所示的组件。

系统设计中的另一个重要概念是不同层次的设计。在分析阶段，我们在理解所有细节之前先定义了问题的范围，我们称之为自顶向下分析。在前面我们说过，分析方法有两种：自顶向下方法（例如，先定义范围，后定义细节）和自底向上方法（例如，首先建立数据流程图片段，再到中间层流程）。在设计过程中也采用同样的思想。

当你开始从事这个行业时，你会发现最高层设计有着许多不同的名称，包括架构设计、总体设计和概念设计。这里，我们使用架构设计这个术语。在架构设计中，你首先应该在进行细节设计之前明确解决方案的整个框架构造。设计细节也叫细节设计。这时，我们并不需要严格区分什么是架构设计，什么是细节设计，也不需要严格区分哪些文档或模型是属于架构设计还是属于细节设计，重要的是，要认识到设计应该以自顶向下的方式开始。从图9-1中，我们可以看到用这种方式进行设计时各部分之间的关系。

架构设计：对整个系统结构进行的广泛设计，也叫总体设计或概念设计。

细节设计：低层设计，包括具体的程序细节的设计。

对于整个系统，分析员首先要确立完整的应用程序配置环境。在确定路由器、防火墙、工作站等具体细节之前，就应该确认完整的体系结构需求和网络结构。这在第8章中已经讨论过。

对于应用程序，第一步是确立不同的子系统之间及其与网络、数据库和用户界面之间的关系。确定自动化的边界是前期设计的一部分。系统边界要确认哪些属于自动化系统的内容，哪些是需要人工进行的。确认好自动化的程度后，我们就可以开始设计了。这在第8章中曾做过解释。

对于数据库部分，第一步是确定所使用的数据库类型和数据库管理系统。一些关于记录结构和字段的细节可以确定下来，但最终的设计决定还取决于系统的体系结构。

对于用户界面，分析员首先设计基于主要输入、输出的进行用户对话的适用表格和结构。项目组还需要描述用户界面元素与应用软件、硬件设备的关系。最后，项目组还要开发出详细的窗体和报表格式。

9.1.2　从分析到设计

在分析阶段的活动中，我们建立了文档和模型。采用传统的分析方法，我们将建立事件表、数据流程图和实体-联系图等模型；采用面向对象的分析方法，我们将建立事件表，同时也建立一些其他模型，如类图、用例图和用例描述等。无论哪种分析方法，设计阶段的输入都是一系列在前期阶段创建好的文档和模型。

分析阶段，我们通过建立模型来表示真实的世界，以便理解所期望的业务过程以及在这些过程中所用到的信息。基本上来说，分析首先是分解，即把一个具有复杂信息的综合问题分解成易于理解的若干个小问题，然后分析员通过建立需求模型来对问题领域的知识进行组织，构造并编制文档。分析和建模需要大量的用户参与来解释需求，并验证建模是否正确。

设计也是一个建模的活动，它使用分析阶段得出的信息（即需求模型）来建立系统解决方案的模型。设计阶段所涉及的技术上的问题较多，所以它不要求太多的用户参与，但要求有更多的系统分析员和其他技术员参与其中。图9-2展现了从分析到设计的流程并指明了每个阶段的主要目标。

设计的原始定义表明：设计包含了对系统解决方案的描述、组织和构造。设计活动的输出是一系列实现这些目标的图和文档。这些图就是系统解决方案的各个方面的模型及其相应文档。与分析模型一样，结构化设计方法和面向对象设计方法的设计模型有些部分是相似的，但也有些部分也存在着极大的不同。

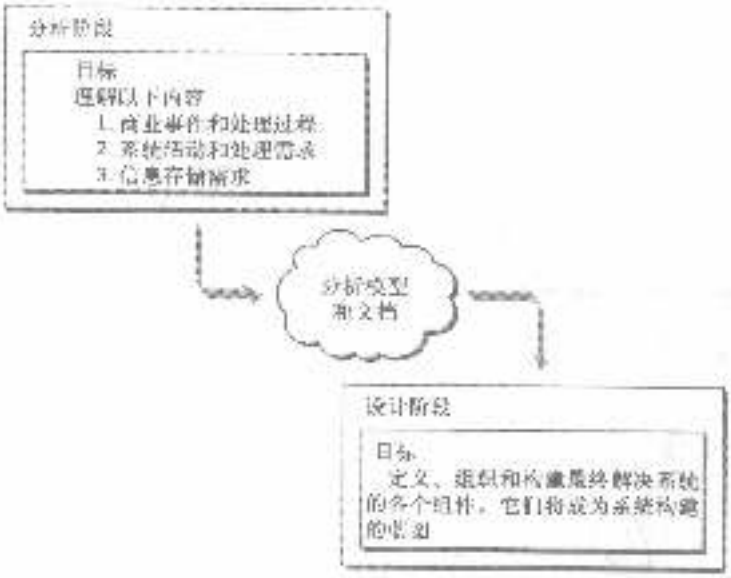


图9-2 从分析到设计的流程及各自的目标

图9-3复制了图5-42中的信息，并把它扩展成传统的结构化设计模型和面向对象设计模型。如图9-3所示，在分析阶段开发的模型能很好地满足设计阶段开发的模型——传统的分析模型满足传统的设计模型，面向对象的分析模型满足面向对象的设计模型。请注意，有些模型对两种方法都适用。

在进行数据库设计时，传统的方法通常会使用关系数据库模型，而面向对象设计技术既可以使用关系数据库模型，也可以使用新的面向对象数据库模型。在进行用户界面设计时，人机对话、表单、报表的设计技巧对两种技术而言是一样的。在数据库和用户界面的设计上，无论是传统的结构化设计方法还是面向对象的设计方法都使用了许多相同的技术。

然而，在应用程序结构设计领域，传统的结构化设计技术和面向对象的设计技术有本质的不同。结构化设计技术，包括分析模型和设计模型，使用"输入/处理/输出"的软件模型来描述系统组织结构已有多年。这些模型非常适合描述业务应用软件，它们中大部分依赖数据库和文件，并且不需要复杂的实时处理。这些模型最早是为支持应用程序设计的，使用COBOL和BASIC编程语言开发，也适用于其他语言，例如，C、FORTRAN、Pascal以及其他的面向事务的程序设计语言。

面向对象技术是一门新技术，在20世纪80年代后期才开始得到广泛的应用。它适用于具有实时性、交互性以及事件驱动的软件，如多任务的操作系统。由于当前许多的业务软件也是具有交互性和事件驱动的应用程序，所以面向对象的开发技术很快就成为应用软件开发的首选技术。

一个经常提到的问题就是：结构化设计技术和面向对象设计技术能混合在一起使用吗？换句话说，能在分析阶段使用结构化分析，而在设计阶段使用面向对象方法吗？反过来是否也成立呢？答案是：在某些时候，这两种技术是可以混合使用且能配合得很好的，譬如，在进行传统的结构化分析之后，可用面向对象方法完成界面的设计。但一般来说，对于应用程序设计，这样的混合效果并不是很理想，因为这两种设计技巧的基本原则有本质上的不同。使用传统方法进行的应用程序设计提供的是基于系统功能的体系结构，而使用面向对象设计方法进行的系统设计是基于一系列交互对象的体系结构的。

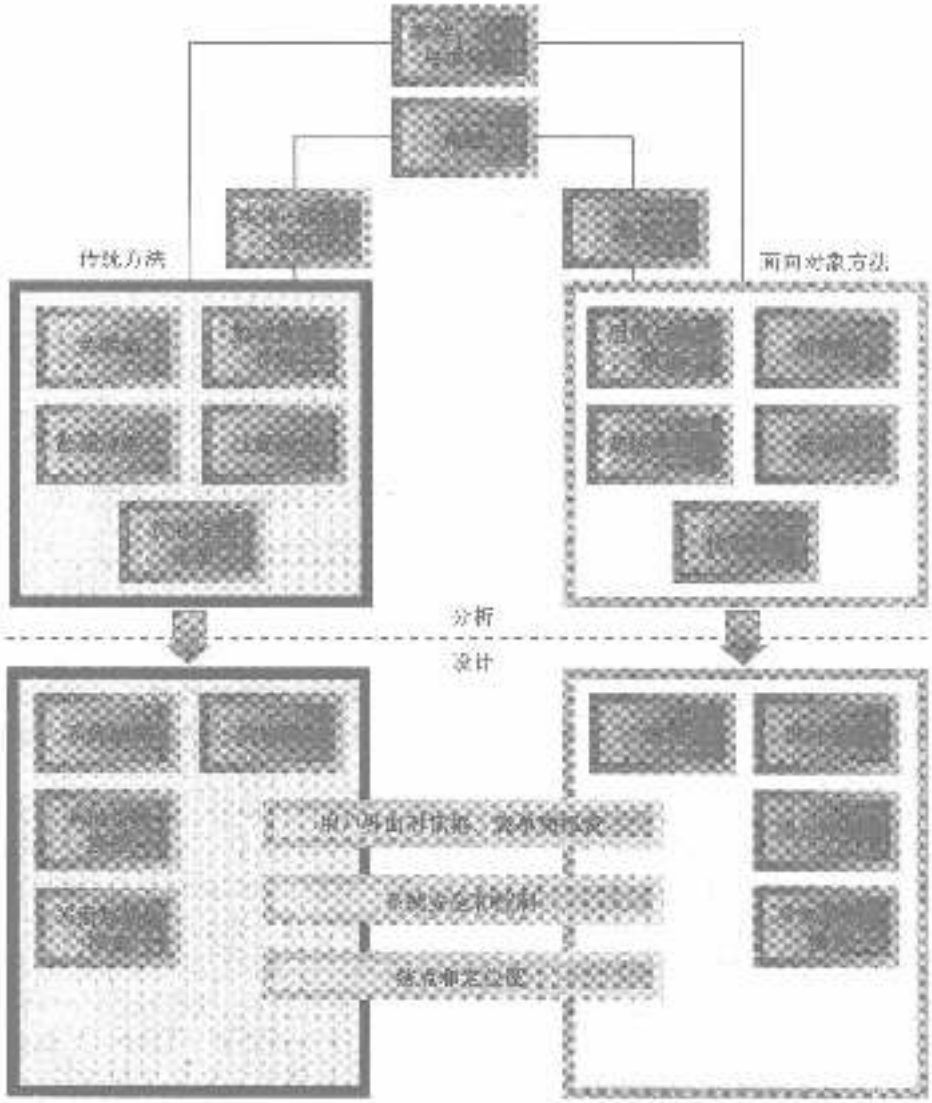


图9-3　传统的结构化设计和面向对象的设计模型

无论使用哪种方法，一旦分析员明确了系统主要组件，考虑好整体的架构设计，并且有了分析阶段得到的文档和模型，他们可以开始着手考虑如何设计系统了。下面我们转到设计阶段的下一项活动。

9.2　设计阶段的活动

SDLC设计阶段所确定的一些活动为设计过程提供了一个概貌。如前文所述，这些活动为图9-1中所示的每个组件提供了设计方案。在本章的下一节以及后续章节中会更详细地解释设计过程和每个设计活动环节。首先，我们看一下SDLC设计阶段的总体状况。图9-4描述了和这个设计阶段相关的一些活动。

设计阶段详细说明了系统是如何使用专门的技术工作的。一些设计的细节在进行系统分析的时候就可以开发出来，那时候，我们已经对各种方案进行了描述，但是我们需要了解更多的细节。有时候，设计工作和分析工作同时进行，系统设计中的一些活动也是同时进行的。

例如，数据库的设计和用户界面的设计通常一起展开。在SDLC中，我们也常常使用迭代的方法。因此，当我们在讨论设计阶段的活动时，这些活动有时会一次又一次地出现。就像在分析阶段的各种活动一样，设计阶段的每一个活动都可以被归纳成一个问题，如图9-5所示。



图9-4 SDLC的各个阶段以及设计阶段的活动

设计阶段的活动	关键问题
设计和集成网络	是否已经详细说明整个组织中系统的不同部分彼此如何通信
设计应用程序结构	是否已经详细说明每个系统活动实际中是如何由人和计算机来实现的
设计用户界面	是否已经详细说明所有的用户如何与系统交互
设计系统接口	是否已经详细说明系统如何与组织内外的所有其他系统一起工作
设计和集成数据库	是否已经详细说明系统怎样存储组织所需的所有信息？存储到哪里
建立设计规范原型	是否已经创建了原型以确保所有的详细设计决策被充分理解
设计和集成系统控制	是否已经详细说明如何确定系统运行正常以及系统维护的数据的安全性和保密性

图9-5 设计阶段的活动以及其关键问题

每一个这样的活动在最终的设计文档中都有自己专门的一部分，就像一座建筑的设计图有几份不同的文档一样，一个系统设计包也是由一系列专门用来详细说明整个系统的文档构成的。此外，与设计图在描述同一个实际建筑时必须一致、完整一样，不同的系统设计文档也必须相互结合为整个系统提供一个全面的说明。

9.2.1 网络的设计与集成

有些时候，新系统和一个新的网络需要一起实现。如果是这样的话，我们就有必要对网络进行设计了。然而，通常网络专家们会根据整体的战略规划来构建网络。设计者所选择的系统设计方案要适应已有的网络计划。所以，项目组并不是要设计一个网络，而是把新系统集成到现有的网络中去。

当在网络上对系统进行操作时，像系统的可靠性、安全性、吞吐量以及同步性这些重要的技术问题将会随之产生。同样，我们需要专家来提供一些技术上的帮助。在系统分析阶段所得到的系统需求会说明在哪些位置进行什么样的操作，因此，这些位置需要进行相互连接。技术需求（和功能需求相对）通常和这些通过网络的通信相关。

在本章的接下来的部分中，我们将提出在网络设计中需要注意一些关键问题。在实现"网络的设计与集成"时所必须回答的关键问题是："我们有没有详细地说明系统的各个不同部分之间是如何在组织内部进行通信的？"

9.2.2 设计应用程序的结构

设计应用程序的结构包括详细说明所有系统活动是如何完成的。在系统分析阶段过程中，这些活动已经作为逻辑模型被详细地描述过了，但那时我们并没有说明要使用那种专门的技术。一旦详细的设计方案被确定下来，我们就可以设计具体的计算机处理过程了。即物理模

型。这里，一个关键的问题是如何对系统自动化边界进行定义。这个问题在第8章中讨论过，它将人工操作和计算机自动操作区分开来。所创建的模型包括实际的数据流图、结构图、对象交互图以及其他的物理模型。

应用程序的设计方法以及创建的设计模型会因为系统的开发配置环境的不同而有所差异。举个例子，如果使用Visual BASIC作为编程语言，那么所开发出来的模型的类型和特点就将与使用COBOL语言所开发出来的有所不同。如果采用客户-服务器结构，则使用的模型将不同于采用了集中式结构的模型。如果采用面向对象的技术，各模型间的区别就更大了。另外，一些活动是由人而不是由机器来完成的，因此需要设计手工过程。

在实现"设计应用程序的结构"时所必须回答的关键问题是："我们有没有详细地说明每个系统的活动是怎样由人和计算机来执行的？"

9.2.3　设计用户界面

用户界面的质量是信息系统的一个重要方面。设计用户界面要确定用户将如何与系统进行交互。对大部分用户来说，这个界面是一个包括窗口、对话框和鼠标交互的图形界面。如今，交互方式日渐丰富，还包括声音、视频以及语音命令。用户的能力和需求相差很大，每个用户都以不同的方式和系统进行交互。此外，系统的不同部分可能需要不同的用户界面。所以，需要考虑许多种用户界面。随着信息系统的交互性和可访问性越来越强，用户界面正逐渐成为信息系统的一大部分。

分析人员必须记住，对用户来说，用户界面就是系统。用户界面不仅仅是个屏幕——它是用户在使用系统时所能接触到的一切。无论从概念上、感性认识上还是实际上讲都是这样的。因此，用户界面并不是系统的附属品。在系统的整个开发过程中它都需要被考虑进去。

用户界面的特点出现于开发过程的早期阶段。具体而言，即刚刚确定了系统需求的时候。因此，可以从对用户所要完成的任务的说明入手，开始对用户界面进行定义。在选择系统设计方案时，每种方案的关键问题就是用户界面的类型了。然而，对用户界面进行细节上的设计工作是在系统设计阶段发生的。

一些用户界面设计专家被请来协助项目组工作。这些专家可以称之为界面设计师，可用性顾问或者人性因素工程师。现在采用的可视化编程环境使应用程序的图形化用户界面的设计变得很容易。但是想要设计出友好的、直观的图形化界面仍然十分困难。

与用户界面设计相关的过程我们将在第13章中继续讨论。在实现"设计用户界面"时所必须回答的关键问题是："我们有没有详细地说明所有的用户如何和系统进行交互？"

界面设计师：界面设计专家，也叫可用性顾问或者人性因素工程师。

9.2.4　设计系统接口

没有绝对存在的系统。一个新的信息系统可能会影响许多其他信息系统。有时，一个信息系统提供的信息以后会被别的系统使用；有时，当系统运行时它们之间不断地交换信息。使系统之间能够共享信息的组件就是系统接口，我们对每个系统接口都要进行详细的设计。

从系统设计的一开始，分析员就必须保证所有的系统可以在一起良好地运作。有些系统接口与组织内部系统相连，所以分析员可以获得一些关于其他系统的信息。在某些情况下，新系统需要和组织外的系统相连接，例如，供应商站点或客户家中；在另外一些情况下，新系统需要和组织已经购买并安装的软件包相连接。使用目前各种各样的技术，系统接口可以变得很复杂。通常，这些接口设计工作需要由有非常专业的技巧的人来完成。

我们会在第14章中继续深入讨论与系统接口有关的话题。在实现"设计系统接口"时所必须回答的关键问题是："我们有没有详细地说明系统如何与组织内外的其他系统一起工作？"

9.2.5 数据库的设计与集成

为系统设计数据库是为一个关键的设计活动。在系统分析阶段所创建的数据模型（一个逻辑模型）在这里将被用来为数据库创建一个物理模型。有时候，数据库是传统计算机文件的集合，而更常见的数据库是一个包含几十张甚至上百张表的关系数据库。有时，文件和关系数据库在同一个系统中一起使用；有时，我们使用面向对象的数据库来代替关系数据库。

在设计数据库时，分析人员必须考虑一些重要的技术问题。许多在系统分析阶段确定的技术需求（与功能需求相对）要考虑数据库的性能需求（如响应时间）。许多设计工作都需要进行性能的优化以确保系统实际工作得足够快。设计数据库的另一个关键问题是必须确保新的数据库能与现有的数据库适当结合。

我们将在第12章中详细地讨论数据库设计的问题。在实现“数据库的设计与集成”时所必须回答的关键问题是：“我们有没有详细地说明系统如何并且在何处存放组织所需的各种信息？”

9.2.6 设计细节的原型

在设计阶段，不断地创建并且评估原型是很重要的。原型和界面设计有关，它可以用来验证包括数据库、网络结构、控制、甚至所使用的编程环境等的设计选择。因此，当分析员考虑所有的设计活动时，他们所想的是如何使用原型来帮助理解各种设计决策。在设计阶段，使用快速应用程序开发（RAD）方法开发的原型将发展成为最终的系统。认识到这一点非常重要。从这个角度来说，原型就是系统。

在实现“设计细节的原型”时所必须回答的关键问题是：“我们有没有建立原型来确保所有的细节设计的决定都能被充分地理解？”

9.2.7 系统控制的设计与集成

最后的设计活动包括确保系统有足够的安全措施来保护组织的资产。这些保护措施叫做系统控制。这项活动被列在最后并不是因为和其他活动相比它不重要，相反，它是一项至关重要的活动。它之所以被列在最后是因为，系统控制要考虑所有其他的设计活动——用户界面、系统接口、应用程序结构、数据库以及网络设计。

用户界面控制限制了授权用户对系统的访问。系统接口控制确保其他系统不会对本系统造成损害。应用程序结构控制用来确保交易记录和其他由系统执行的工作都能正确完成。数据库控制保护数据、防止未经授权的访问并防止由于硬件故障而造成的意外数据丢失。最后，网络控制用来保证网络间的通信得到保护，并且重要性日益增加。在现有技术的基础上，所有这些控制都应该被设计到系统中去，专家常常被请来做一些控制的工作。所有的系统控制需要进行仔细全面的测试。

关于控制的问题在几章中都提到了。我们将在第14章中更为清楚地描述它。在实现“系统控制的设计与集成”时所必须回答的关键问题是“我们有没有详细地说明如何保证系统正常操作以及系统所维护的数据的安全性？”

9.3 项目管理——协调项目

在开发项目时，设计活动的早期十分关键。开发的重点从发现需求转到寻找解决方案，同时项目的进程也发生了改变。协调好所有现在正在进行的活动，即使对于最好的项目经理来说也是一个挑战，因为这需要处理大量的细节问题和任务来保证项目按计划进行。附录A中的图A-4说明了许多在设计阶段所需要进行的项目管理任务。这些任务的大部分都包括监控进程以及协调现行的工作。

即使这时有一轮的分析实际上已经完成，仍有一些分析任务还需要进一步协调。每一个新系统都有必须集成到系统中的大量商务规则。例如，一系列关于销售佣金的商务规则包括：它在何时且是如何进行计算的，在货品返回时怎么处理，佣金何时支付，如何制订佣金计划表来鼓励销售高利润的商品等。要正确地开发佣金程序，这些商务规则必须都定义好。然而，如果管理人员还未对这些商务规则做出最后的决定，那又该怎么办呢？你不想因为这些悬而未决的考虑而阻碍整个项目进展，另一方面，你又想确保这些决定不会造成系统不可用。

此外，项目组成员，包括用户在内，最好能对新系统的潜力有所了解，他们可能希望通过调整商务规则来得到高水平的自动化和支持。这个要求对于公司来说很好，因为公司将从这个改进的系统中获益，但对于该项目来说却并不好，因为它需要扩大项目的规模并可能会延误整个项目。当面对这些对公司很重要的额外要求时，项目经理如何来控制项目的规模？

设计活动还需要多方的合作。由于设计任务数目的日益增加，项目被划分成块。通常，系统划分成了系统，而每个子系统都有自己的设计要求，项目组可能会被分成更小的组，以适应不同的子系统或其他设计任务。一些技术问题，如网络配置、数据库设计、分布式处理以及通信能力，对所有了系统都是相同的。其他问题，如响应时间、特殊输入设备等，也可以限制在特定的子系统中。协调和集成所有了系统的技术问题以及任何中间件问题，对整个系统来说是很重要的。最终，能否成功地开发出项目取决于整个设计组工作的协调性如何。

这时可能还会有两个小项目：一个是数据转换项目，另一个是测试实例的开发。在第15章中我们会更详细地介绍这些内容。在这里，提到它们的目的是想让大家注意，它们增加了项目管理的复杂度。

最后，在实施阶段的一些活动，例如编码，也应该在这时候开始了。实际上，设计和编码通常是同时进行的。当设计任务确定下来了，就可以立即开始编程了。因此，除了设计工作小组外，程序员组和程序员/分析员组也要加入到开发组中来。

如果参与项目开发的人员身处各方，这将进一步加大控制协调这些不同设计活动的复杂性。项目组的通信管理是能否成功协调整个项目组的关键，它将随着项目组人员的不断增加而日趋复杂。

对上述的复杂性，我们可以使用一些工具和技术来帮助我们协调项目的管理。

实践指南

经常召开会议协调设计活动，以保证整个小组成员步调一致并且清楚项目中的重要问题。 ■

9.3.1　协调项目组

协调不同项目组活动的最基本工具是项目进度表。随着设计阶段活动的开始，项目经理必须确定并估计与设计、实现相关的任务以及任何与目前需求相关的外部任务，并以此来更新项目进度表，同时还要对即将进行的比较重要的任务做出安排。在后面的阶段，要经常更新项目表，以保证项目是有组织进行的。

在分析阶段，经常是项目经理和助手进行项目管理。当项目扩展并形成了几个开发队伍时，项目的管理就会变得复杂了。一般来说，由关键设计领导者和实施小组组成的委员会来承担协调和控制方面的主要责任。每周，有时甚至是每天，都要召开进度会议。如果参加会议的人在较远的地方，那么还需要远程会议的支持。

9.3.2　RMO的项目组

随着RMO的客户支持系统项目进行到设计阶段，项目组引入了新成员，从而增强了实力。和早期的决策一致，RMO在这时产生了两个子项目：一个是数据转换，另一个是系统与验收

测试计划。为了使新成员融入项目组，Barbara Halifax对项目组的结构进行了重组。在分析阶段就一直在组里的那些人现在成为引导新成员快速入门的关键人物。短带的RMO备忘录中强调说明了这一时期项目的一些变化。

2007年5月5日

To: John MacMurty

From: Barbara Halifax

RE: 客户支持系统的设计

John，这是一个非常繁忙的时期！这里是一份已完成的和将要开展的活动情况简要报告：

上阶段所完成的工作

在法列登管委会可以继续进行设计的批准之后，我已经完成了项目进度表，它详细说明了我们从设计和实现阶段。我们又新加入5个人到项目组来，两位用户，一位资深设计师来协助我们进行系统设计。另外还有两位程序员分析员。

除了现在还在负责设计工作的主要项目组以外，我又组织了两个小组，一个用来确定数据转换的范围，另一个用来开发验收测试的例子。新加入的成员分别到小组中工作。

下阶段的计划

我们将要使设计完成架构设计。一旦架构设计完成了，我们就可以完成数据库的设计，接下来是用户界面设计并开始编写新的应用程序设计。

我也初始化了两个控制过程，为了稳定系统的范围，新的需求变化将被记录在活动请求日志上并等待委员会的批准。我们同样也召开组内情况会议以协调我们的设计活动。

问题、结果和公开条目

这段时间的主要问题是，在公开条目列表中有好几条需要最后确定下来的，就有我的人需要解决的技术问题，也有需要决定的商业需求定义。

我们正在努力中。

cc: Steven Decrfield, Ming Lee

9.3.3 协调信息

随着设计工作的进行，开发组产生大量关于系统的细节信息，模块、类、数据域、数据结构、表单、报表、方法、子程序、表格等，它们都需要详细定义。要跟踪这些信息需要进行大量的协调工作。不过，有两种工具能在这个处理过程中起到帮助作用。

最一般且广泛运用的技巧是用一个CASE工具记录和跟踪项目信息。大多数的CASE工具有一个用来捕捉信息的中心库。在第2章中，我们已学过有关CASE工具的知识。图2-22介绍了组成综合CASE系统的各种组件工具。CASE工具系统中的主要元素是保存信息的中心信息库。中心信息库不仅记录所有的设计信息，还正确地进行了配置，这样各小组能看到项目信息，以便了解项目组间的交流。图9-6给出了可能存在于一个CASE数据中心库中的各种信息组件。

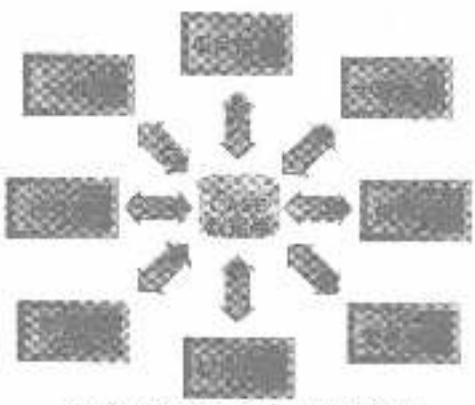


图9-6 CASE库中的系统开发信息

其他的电子工具也能有效地帮助小组进行交流和信息合作。这些工具和技术，通常是指计算机协同工作支持。它不仅记录最终设计信息，还帮助协同工作。在开发过程中，一些人往往需要共同合作。他们需要讨论和动态更新工作文档和图表。Lotus Notes就是经常使用协作工具之一。还可以使用其他的软件程序记录跟踪信息和版本信息来更新数据和图表，以便保留结果的变化历史。

开发项目的一个特别困难之处在于跟踪公开条目列表和未解决的问题。这里我们提到这个问题不是因为它需要使用一种新技术，而是因为它普遍存在。所有优秀的项目经理都会使用一些技术来跟踪这些条目。一个简单的方法是建立一个公开条目控制日志。在第4章中我们提出了公开条目控制列表的概念，并在图4-11中举了一个例子。它是所有公开项目的顺序列表，附有公开项目的职责和解决方案等信息，正如方框里的标题所示。通常，为了易于识别未公开的条目而加上阴影。这一技巧不仅在分析阶段，而且在整个项目开发过程都很有效。

实践指导

以公开条目形式列出未解决问题和疑难并经护该表。 ■

既然我们对设计阶段和项目管理的主要问题已经有了一个总体上的认识，我们就将话题转移到与结构和网络设计相关的任务上来。我们首先讨论一些与配置环境相关的技术，因为配置环境直接影响着架构设计；然后我们描述一些架构设计的常规方法以及与每种方法对应的软件设计、网络设计的问题。

9.4　配置环境

在第8章中，我们已经知道，定义配置环境是一项连接分析与设计的活动。配置环境包含了系统运行所需要的硬件、系统软件以及网络环境。本节将详细讨论一些常见的配置环境，在下一节中，将讨论设计模式和应用程序软件结构的话题。

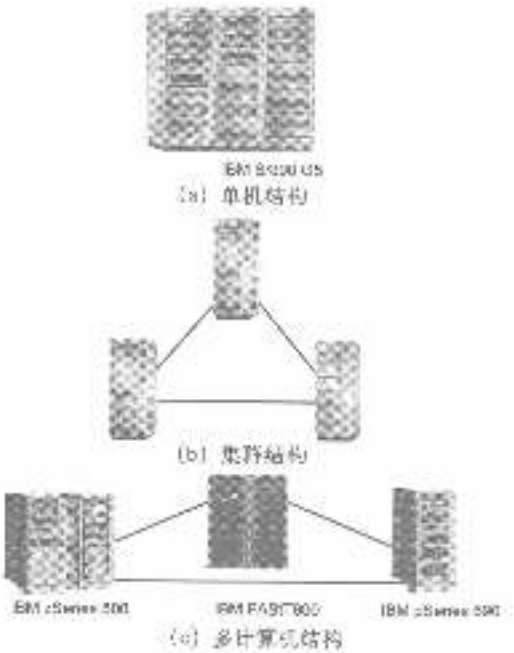
9.4.1　单机结构与多层结构

顾名思义，单机结构是指将所有的信息系统资源放在单独的一台计算机系统以及它直接所属的外围设备上，如图9-7a所示。它可以是独立的PC应用程序，但这里指的是大型主机应用程序。用户通过直接与计算机相连的简单的输入/输出设备与系统进行交互。单机结构要求所有系统用户都在这台计算机附近，其主要优点是简单性。相对而言，配置在单机系统中的信息系统设计、构造、操作以及维护都较为简单。

单机结构：只使用一台计算机来执行所有应用和关软件的结构。

然而，单个计算机的容量局限性使单机结构对于大型信息系统来说可能不切实际或不可用。许多系统非常之大，甚至是最大的大型计算机也不能执行所有所需的处理、数据存储和数据检索任务。对于这样的系统，需要采用另一种结构。

多层结构使用多个计算机系统以协作的方式来满足信息处理的需求。它可进一步分成下面两



(a) 单机结构
(b) 集群结构
(c) 多计算机结构

图9-7　单机、集群和多计算机结构

类结构。

多层结构：将应用和关键件或处理过程分配到多个计算机系统上的结构。

- 集群结构 使用一组（或一簇）计算机，它们通常由同一生产厂商生产，并且属于同一类型，如图9-7b所示。应用程序在运行时能被分配到一台最空闲的计算机上进行处理，这样，所有的机器能平衡分担处理负荷。实际上，这一组计算机可以当做一个大型的计算机系统。集群计算机系统一般彼此相邻，这样它们之间便可通过高速通信链路进行连接。
- 多计算机结构 同样使用多个计算机系统，如图9-7c所示。但它不像集群结构那样要求硬件和操作系统是相似的，每台计算机系统都各自分配了一套应用程序和数据资源，并针对其在组合系统中担当的角色进行了优化，如数据库服务器或应用程序服务器。

集群结构：一组可以协同工作，类似于一个大型计算机系统的同类计算机。

多计算机结构：一起链接在一起实现特定功能的不同类型的计算机。

9.4.2 集中式结构与分布式结构

集中式结构是指将所有的计算机系统都部署在同一个位置。系统集中式大型机一般用于大规模的处理应用，包括批处理和实时处理，这类应用常见于银行业、保险业和分类销售业。这些领域的信息系统通常有以下特点。

集中式结构：把所有的计算资源集中在一处的结构。

- 一些输入事务，不需要实时处理（例如，大批夜间从中央银行票据交换室发出的其他州账单）；
- 在线数据输入人员可以集中到一个地点（例如，集中于同一地点的一组电话订单转接员可以为地理上分散的客户服务）；
- 系统定期产生大量输出（例如，寄给客户的每月信用卡结算表）；
- 高速计算机之间产生大量的事务（例如，进行供应链管理的B2B处理事务）。

任何一个具有上述两到三个特点的应用系统都可以在集中式大型机上实现。现在，由于电子商务需要处理大量B2B事务，因此它日渐流行的趋势也给大型机处理注入了新的活力。

集中式计算机系统很少用散单独的信息系统硬件平台。大多数系统都有一些事务输入必须从地理上分散的地点接收并需要实时处理（如ATM机现金提取）。大多数系统也有一些输出源于远程请求并需要实时发送到远方（如州立机动车辆部门的保险政策查询）。因而，集中式计算机系统通常用散一个更大的信息系统中的一个或多个子系统，这种更大的系统包括在线处理、批处理以及地理上分散的组件。

现代信息系统的组件通常分布于多个计算机系统和不同的地理位置上。比如，公司财务数据可能存储在一个集中式大型计算机上。区域办公室里的中型机可根据存储在大型机中的数据来定期生成账目和其他报表。分布在不同地点的个人计算机可以访问和查看定期报表，也可以直接更新中心数据库。这种把组件分布到不同计算机系统和不同位置的结构叫做分布式结构。分布式结构依赖通信网络将地理上分散的计算机连接起来。

分布式结构：把计算资源分散在由计算机网络相连接的不同地点的结构。

9.4.3 计算机网络

计算机网络是允许在不同的用户和计算机系统之间通信的一系列传输线、设备和通信协议的集合。根据它们跨越的距离，计算机网络可以分为两类：局域网（LAN），典型的是在1公里范围之内，在一栋大楼或一个楼层内连接计算机；广域网（WAN），这个术语可以描述

超过1公里的任何网络，包括跨越城市、国家、洲，甚至整个地球。

计算机网络：用来共享信息和资源的传输线、设备和通信协议的集合。

局域网（LAN）：一个在本地区域内的计算机网络，例如，在同一幢大楼内。

广域网（WAN）：跨越如城市、州或国家这样大距离的计算机网络。

路由器：在网络内用来定向信息的一种设备。

图9-8所示的是为RMO设计的一个合理的计算机网络。每个地理结点由一个单独的局域网来服务。所有的局域网通过一个广域网连接起来。在一个结点上的用户和计算机之间通过局域网交流，地理位置上分散的结点之间的信息交流由两端的局域网和广域网完成。路由器将每个局域网与广域网连接起来。如果某个局域网上的信息被另一个局域网的用户或计算机访问，路由器就会扫描这些信息，并且把它们复制到广域网上。如果广域网上的信息被局域网的用户和计算机访问，路由器也会扫描这些信息，并且复制到局域网中。

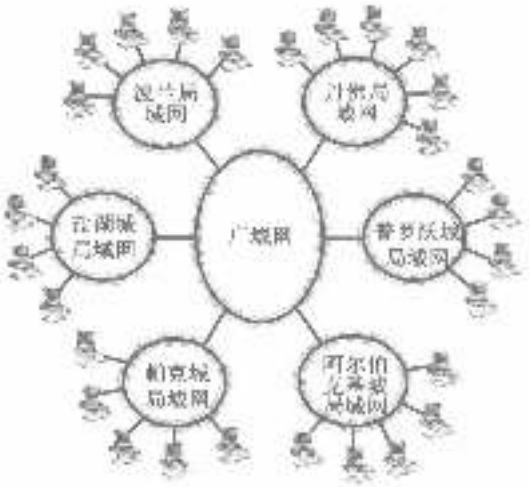


图9-8　一种可能的RMO网络配置图

建立局域网可以使用许多技术，例如，以太网技术是典型的用于实现局域网的技术，它们以相对低廉的成本提供从低等到中等的信息传输容量。像异步传输模式这样的广域网技术较复杂且较昂贵，但它们可以提供更好的信息传送能力和更高的可靠性。可以通过购买装置和租用远距离传输线路来建立广域网。广域网的建立和运营也可以转包给从事远距离通信的提供商，如AT&T或Sprint。

计算机网络在计算机系统和用户中提供普遍的通信能力。这种普遍的通信能力可用来实现多种服务，包括直接的通信（如电话服务和视频会议）、基于消息机制的通信（如电子邮件）和资源共享（如访问电子文件、应用程序和数据库）。如果有合适的硬件和足够的传输容量，一个单独的网络能同时支持所有类型的服务。

将信息系统资源分布到计算机网络中有多种方式。用户、应用程序和数据库可以放在同一个计算机系统中，同一个局域网的不同计算机系统中或不同局域网的不同计算机系统中。应用程序和数据库也可再细分并且每一个子块可以独立分布。

9.4.4　Internet, Intranet和Extranet

国际互联网（Internet）是一个全球性的网络集合，它们使用通用的低层网络标准协议TCP/IP互相连接（TCP/IP为传输控制协议/互连协议）。万维网World Wide Web（WWW）也简称为Web，是资源（程序、文件、服务）的集合，这些资源在Internet上可通过大量标准协议访问，这些协议包括：

国际互联网（Internet）：一个全球性的网络集合，它们使用相同的网络协议——TCP/IP。

万维网（WWW或Web）：是文件和程序等的资源集合，这些资源在Internet上可通过标准协议进行访问。

- 格式化链接文档协议，如超文本链接标记语言HTML，可拓展标记语言XML，超文本传输协议HTTP。
- 执行程序标准，如Java，Java Script和VB Script。

- 分布式软件和网络服务标准，包括公共对象请求代理结构（CORBA）、简单对象访问协议（SOAP）以及Java 2网络服务（J2WS）。

Internet是Web的基础，换句话说，Web的资源要在Internet上传送给用户。

企业内部互联网（Intranet）是一个专用网络，它使用Internet协议，但只限定一些内部人员使用（通常是一个组织和工作组的成员）。这个术语也可以描述为：一组专用的可访问资源，这些资源是通过一个或多个在网络上支持TCP/IP的Web协议进行组织和传送的。Intranet使用与Internet和Web相同的协议，但其资源限定为一部分用户使用。这种对资源的限制访问可通过几种方式完成，包括不公开资源名称、防火墙和用户/组账号名及密码。

企业内部互联网（Intranet）：一种专用网络，只限于一定数量的用户访问，并与Internet同样都使用TCP/IP协议。

企业外部互联网（Extranet）是一个扩展了的Intranet，它包括组织以及组织外直接相关的业务用户（如供应商、大客户、战略性合作伙伴）。Extranet允许各自分离的组织交换信息并进行合作，这样就形成了一个虚拟组织。一个广为使用的构建虚拟组织的方法是使用虚拟专用网（VPN）。专用网络是只对组织（或虚拟组织）内部的一些人开放的安全网络，它曾经意味着组织拥有并操作自己的网络线路或者租用专门的电话线，这就是说，它只对组织内的一部分人开放。VPN通过公共网络服务商来发送加密信息。

企业外部互联网（Extranet）：扩展到组织外部以方便信息流通的Intranet。

虚拟专用网（VPN）：建立在公共网络（如Internet）上的只对私人组织开放的安全且可控的网络。

虚拟组织：一个松散联系的人员和资源组，它们共同合作，就像一个组织一样。

9.5 应用程序结构

简单的配置环境（例如，带有视频显示终端的单一集中式计算机）对应于相对简单的应用程序结构。复杂的分布式、多层硬件网络结构则需要更加复杂的软件结构。本节描述常见的分布式、多层配置环境的应用程序结构，并讨论与每种结构相关的设计、决策问题。

9.5.1 客户-服务器结构

客户-服务器结构将程序分成两类：客户和服务器。服务器计算机管理一个或多个的系统资源并提供各种明确定义的服务；客户端连接服务器请求资源或服务，而服务器则响应那些请求。



图9-9 共享数据库的客户-服务器结构

客户-服务器结构是一种能用许多种不同方式实现的通用结构模型。在一台工作站上运行的客户应用程序能够与在另一台大型机上运行的数据库管理系统（DBMS）之间进行交互。一个例子很好地说明了问题（如图9-9所示）。应用程序通过网络向DBMS发送数据库访问请求。DBMS代表应用程序进行数据访问，并将查询操作的结果或更新操作是否成功的信息返回给应用程序。

服务器：在网络中提供服务的一个进程、模块、对象或一台计算机。

客户端：向网络中一个或多个服务器请求服务的一个进程、模块、对象或一台计算机。

在设计客户-服务器软件时，应注意以下结构上的问题：

- 将应用程序分解成客户和服务器程序、模块或对象；
- 确定客户和服务器各自运行的计算机系统；

- 描述连接客户与服务器的通信协议和网络。

实践指导

划分可由独立软件单元管理的资源和服务。

将应用程序分解成客户和服务器的关键在于正确地划分可以由独立软件单元集中管理的资源或服务，可集中管理的服务包括安全认证、授权、信用验证以及日程安排等。对于上述的每一种情况，服务会根据一个对客户不可见的数据库提供了一套明确定义的处理步骤，如取回、更新、批准等。如图9-10所示。



图9-10　客户、服务器和一个与服务器相关的数据库之间的交互

客户和服务器软件可以在任何计算机系统上执行，但通常把服务器软件部署在独立的服务器计算机系统上，并将客户软件安装到“靠近”终端用户的计算机系统上，如桌面工作站。图9-11给出了典型的订单处理应用程序的部署方式。信用验证、日程提交以及数据库服务器处理这些操作集中于中型机或大型机上，用户在工作站上运行的是客户程序的多份拷贝文件。

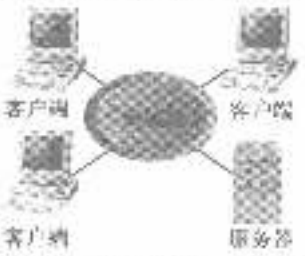


图9-11　多客户端与一个服务器之间的交互

客户与服务器之间在物理网络上通过定义好的通信协议相连接。在图9-11中，网络是局域网，并通过低层网络协议，如TCP/IP协议，来提供基本通信服务。但是，设计者还需要进一步指明上层的协议或语言。因为客户与服务器将通过这些协议或语言来进行请求、响应和数据的交换。有时候，例如，与DBMS进行通信，通过开放数据库互连（ODBC）的数据库连接使用标准的协议和软件（如结构化查询语言SQL）；但有些时候，设计者必须自定义确切的消息、响应的格式和内容。如果服务是由其他组织提供的（如信用验证服务），并且这些组织已经设计了相应的协议，这时应用程序设计者就要求客户端遵守这些协议。

客户-服务器结构的主要优点是开发的灵活性，它是一种把软件分配到连网计算机上的方法，提供了如下几点网络环境的内在优势。

- **位置灵活性**　可以在不影响系统其他组件的情况下移动系统的特定组件，从而可以反映组织的规模和物理位置等参数的改变。
- **可扩展性**　可以通过升级或更换核心软件运行的硬件来提高系统性能。
- **可维护性**　可以更新系统中某组件的内部实现而不影响其他组件的工作（例如，可以重写或更换信用验证程序，只要新软件仍然使用现有的客户-服务器协议）。

客户-服务器结构的主要缺点是引入了客户-服务器协议后所带来的复杂性以及通过网络连接所造成的潜在性能、安全性和可靠性方面的问题。集中式应用程序在独立的计算机上作为一个大型程序运行，它不需要客户-服务器协议，所有应用程序内部的通信都限定在一台相对安全、可靠、高效的计算机上。

对于大部分的机构来说，客户-服务器灵活可变的优点远远大于它的缺点。因此，客户-服务器结构及其新的变体成为大部分现代软件的主流设计结构。

9.5.2　三层客户-服务器结构

三层结构是广泛应用的客户-服务器结构的一种变体，它将应用程序软件划分成一系列独立于硬件环境和地理位置的客户与服务器进程，既可以由一个处理器来承担所有层的计算，

也可以将多个层的计算分配到多个处理器上。换而言之，这些层可以在一层或多层上。最常见的结构包含如下3层。

- **数据层** 它负责管理存储的数据，这些数据通常存储在一个或多个数据库中。
- **业务逻辑层** 它负责实现业务处理的规则和处理程序。
- **可视层** 它负责接收用户的输入，并将处理结果格式化输出。

图9-12显示了三层之间的交互情况。可视层作为业务逻辑层的客户，同时，业务逻辑层又作为数据层的客户。

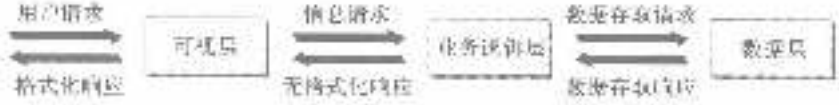


图9-12 三层结构

与前面的客户-服务器结构一样，三层结构也有与生俱来的灵活性。各层之间通常都是响应与请求的交互方式，这使得层与层之间相对独立，各层之间的实现细节彼此互不影响。其他层在何处实现，在那台计算机上采用何种操作系统并不重要，它们之间唯一需要彼此一致的是响应和请求的通用语言和一个有足够通信容量的可靠的网络环境。

多个层可以放在同一台计算机上，每个层也可以由独立的计算机来实现。复杂的层可以由两到三台计算机来实现。通过将层的功能分配给多台计算机或者在冗余计算机之间实现负载均衡，可以提高系统的处理能力。在出现故障时，服务器负载可以从一台计算机转移到另一台计算机上，这种冗余将增强系统的可靠性。总之，三层结构为现代企业提供了部署和重新部署信息处理资源的灵活性以响应不断变化的情况。

三层结构：一种客户-服务器结构，它将应用程序划分成可视层、业务逻辑层和数据层。

数据层：三层结构中负责和数据库交互的部分。

业务逻辑层：三层结构中包含实现业务规则处理程序的部分。

可视层：三层结构中包含用户界面的部分。

目前，无论传统的设计方法还是面向对象的设计方法，都广泛应用三层结构。作为客户-服务器结构的另一种模式，三层结构设计的关键任务是将应用程序分解成层、客户和服务器，与此同时，还要将它们部署到不同硬件平台上，并定义相应的物理网络和通信协议。

业务逻辑层是应用软件的核心部分，它是根据在分析阶段开发的需求模型设计出来的。我们在第5－7章中对此也有所叙述。例如，在传统的设计方法中，RMO数据流图中有关系统活动而定义的业务逻辑都会在业务逻辑层中以函数或处理过程的方式加以实现。由窗口或浏览器表单构成的可视层不会包含太多的程序代码。在面向对象的设计方法中，RMO类图（见图5-41）中的类会在业务逻辑层中实现，并由这些实现的类来完成用户的任务。无论哪种情况，业务逻辑层既是可视层的服务器，又是数据层的客户。尽管如此，业务逻辑层本身也可以再分解为多个客户和服务。三层结构通常都是由面向对象的技术和工具来实现的（见第11章），但它也可以由传统的设计方法和编程语言来实现（见第10章）。从这个角度上来说，三层结构是一种既适用传统方法，又适用于面向对象方法的主要的结构设计模式。

在本书中，第10章和第11章描述如何利用传统方法和面向对象方法进行可视层和数据层软件的设计，第12章描述利用数据层进行访问的数据库的相关内容；第13章描述与实现可视层软件相独立的用户界面设计的技术和基本原则，例如，通过视频显示界面元素的管理以及用户和计算机之间完成特定应用程序任务的对话框。

9.5.3　Web服务结构

Web服务结构是客户-服务器结构的另一种变体，它将软件打包成服务器处理程序，并可以通过Web协议读取获得。这些协议包括XML、SOAP、Web服务描述语言（WSDL）、通用描述、Discover和UDDI。图9-13显示了客户如何通过Web服务目录和Web服务交互。诸如服务器、服务名称、端口号、XML数据格式、安全性要求等Web服务的信息采用WSDL描述并发布在Web服务目录上。客户通过和Web服务目录交互，判定可以获得的服务以及如何获得。之后，客户通过SOAP和XML初始化到Web服务的连接。



图9-13　Web服务结构

Web服务结构：一种客户-服务器结构，它将软件打包成服务处理程序并使之可以通过Web协议获得。

信用验证服务，如图9-10所示，可以采用Web服务结构实现。信用部门可以实现一种或多种服务，它们可以通过SOAP在应用程序或者Web服务器上获得。信用部门将服务信息发布到一个或多个Web服务目录上，从而使客户可以发现并和这些服务进行交互。发布的服务描述应包括必需的输入（如信用卡名称、卡号、有效日期和使用金额等）和输出（如批准或拒绝，授权码等）。只要匹配服务目录中的WSDL描述，服务的内部实现将与客户无关。

Web服务结构为内部和外部客户获得软件服务提供了一种灵活机制，它被广泛应用于由多个组织和计算机提供的组合软件服务，并建立一个统一的系统的情况。比如，RMO可以在它新的在线订购系统上使用外部信用验证、运输、库存补充Web服务。RMO也可以Web服务方式构建一些内部功能（如查询库存数量或提交客户事务），使得它们能在不同应用程序和地方被轻松获取。

9.5.4　中间件

客户-服务器结构与三层结构需要专门的程序来实现不同层间的通信。实现这种通信接口的软件通常称为中间件，中间件连接应用程序的两端，并在它们之间传递请求和数据。实现中间件的功能有多种不同的方法。一些常见的中间件包括事务处理监视器、对象请求调度（ORBs）和Web服务目录。它们每个都有一套自己的协议来使信息系统的不同组件进行相互通信。

中间件：实现网络通信协议并帮助不同系统的互通信的软件。

当指定了客户-服务器之间以及内部层之间的通信协议以后，设计者通常会借助于标准框架。协议并将它们集成到中间件当中。例如，和DBMS的交互访问一般都使用标准的协议（如ODBC或SQL），并同时使用由DBMS供应商提供的或第三方的软件。第三方服务供应商（如信用部门、电子购物、竞拍）一般采用标准的Web协议，如HTTP或XML。许多行业，如医疗保健和银行等，都开发出自己的行业标准协议。

分布在多层、不同硬件平台上的复杂的面向对象软件依赖于基于分布式对象接口标准（如CORBA）的ORB。分布式的非面向对象软件依赖基于诸如DCE或Microsoft的COM+这些标准的不同的中间件产品。基于网络的应用程序需要依赖面向网络的协议（如Microsoft的.NET和Sun的J2MS）以及实现并支持这些协议的专门的中间件产品。在第16章中，我们会更加详细地讨论有关分布式对象、组件和基于网络的应用程序的协议和软件。

9.5.5　Internet和基于Web的应用程序结构

Web是客户-服务器结构的一个复杂的例子。Web资源既可以由一些专用的服务器来管理，也

可以当一些多用途的计算机系统来管理。客户程序使用一种或多种标准的Web资源请求协议向服务器发送请求。Web协议定义了有效的资源格式和请求资源与服务的标准方法。任何程序（不仅仅是网络浏览器）都可以使用Web协议，所以在普通的应用程序中也可以嵌入类似网络的功能。

Internet和Web技术对实施信息系统提供了一个极富吸引力的选择方案。例如，从RMO供应者那里购买产品的RMO采购人员会考虑数据输入和访问的问题。采购人员一般一年中要有几个月在路上奔波，通常一次就要几个星期的时间，因此，他们需要使用一些远程交互方式与RMO的供应链管理（SCM）系统联系来记录购买协议和查询库存状态。

提供这些能力的一种方法是设计客户应用软件和连接这种软件的私有网络。系统的主要部分应安装在RMO的服务器中，应用程序的客户部分（即数据录入部分）则安装在采购人员的掌上电脑中。采购人员远程登录系统并连接到执行应用程序的服务器上，进行数据库查询和数据添加。

对采购人员来说，实现远程访问的另一种方式是构造采用Web浏览器界面的应用程序。这种应用程序运行在Web服务器上，使用HTML或XML与Web浏览器通信。任何一台连接到Internet的计算机都可对它进行访问。采购人员不论当前身处何处，都可以在他们的掌上电脑上使用Web浏览器，并通过Internet服务提供商连接到应用软件。采购人员也可以从任何一台与Internet连接的其他计算机（如销售商、饭店亚间或像Kinko这样的复印中心的计算机）访问应用程序。

灵活是Internet方案的关键优势所在。通过Internet实施应用软件大大扩展了应用软件的可访问性，同时，它也不用在采购人员的掌上电脑上安装客户端软件。只需更新Web服务器端的软件版本就可以更新客户端软件。由于可以使用已有的网络标准和网络资源，开发和配置这种应用软件相对成本较低。定制软件和通过调制解调器的专用访问开发更为复杂，需要对许多自定义资源进行维护。

相对于传统的客户-服务器方法的应用程序结构而言，用Web，Intranet或Extranet实现应用有着许多优势，具体包括以下几个方面。

- 可访问性：Web浏览器和Internet几乎无处不在。Internet，Intranet和Extranet的应用对许多潜在用户（包括客户、厂商以及远离单位的工作人员）都是可访问的。
- 通信费用低：作为Internet骨干网的高容量广域网由政府投资建立。用户在骨干网上的通信是免费的，至少目前是这样的。专用局域网和Internet之间的联系可以用相对较低的费用从一些专用Internet服务供应商那里购买。实际上，公司可以把Internet当做一个廉价的广域网。
- 广泛的实现标准：Web标准已经众所周知，许多计算专业人员也已在使用中得到锻炼，服务器、客户端以及应用程序的开发软件都很容易得到，并且相对便宜。

由于Intranet和Extranet使用了Web标准，因此，通过Intranet和Extranet进行信息资源传送充分发挥了Web传输的各种优势。在许多方面，Intranet，Extranet和Web反映了客户-服务器计算向非定制技术的逻辑演变。以前由于费用和学习曲线的问题，组织极力回避客户-服务器计算，现在因为其在复杂度和费用上的大大减少，他们可尽情享受客户机-服务器结构所带来的一切便利。

当然，通过Internet和Web技术的应用传送也有其不利的方面，包括：

- 安全性　如果要破坏网络，Web服务器则是一个明确的目标，因为Web标准是公开的且人人皆知的。网络的大范围的互相连接，Internet的使用和Web标准创造了一个共同而又易被访问的目标。
- 可靠性　Internet协议不能保证每个用户最低的网络吞吐量或者一个消息只能被正确的人员接收。人们已提出许多标准来修正这些缺点，但还没有被广泛采用。
- 吞吐量　大多数家庭用户和许多企业用户的数据传输容量受模拟调制解调器的限制，它的速度低于56Kb/s。在高峰期间，Internet服务提供商和骨干网可能会超负荷。对所

有用户来说，这将导致响应时间延长。当访问大量资源时，需要很长一段延迟时间。

- 标准不稳定　Web标准变化很快。用户软件每几个月要更新一次。广泛使用的应用系统的开发者也进退两难：要么用最新的标准增加功能，要么用旧标准以保证与用户软件保持较大的兼容性。

对RMO来说，采用Internet方式的客户订单系统最主要的缺点是安全性、性能和可靠性。如果采购人员可以通过Web访问系统，那么其他用户也可以。虽然可以有许多方式控制对系统敏感部分的访问，包括用户账号或密码，但仍然存在破坏网络安全性的风险。采购人员的Internet连接点和介于连接点与应用服务器之间的Internet网络容量限制了网络的性能和可靠性。不可靠的和超负荷的本地Internet连接会导致应用系统不可用。RMO对这些因素还没有进行控制。

基于Web的应用程序结构设计中的关键问题与客户-服务器结构中存在的问题相似：定义客户与服务器处理进程或对象，将它们部署到不同的硬件平台上，并以合适的网络、中间件和协议相连接。然而对基于Web的应用程序而言，对中间件以及协议的选择范围要比其他形式的客户-服务器结构有更多的限制条件。

由于我们已经讨论了应用程序结构的通用开发方法，所以我们将集中精力来设计连接现代信息系统模型各部分的网络构造。

9.6　网络设计

目前，各个企业内部都在使用网络，因此，许多新开发的项目都包含了网络设计。对任何多层的系统来说，网络规划和设计的一些关键问题都必须在设计阶段的早期进行处理。这些问题如下：

- 集成网络既要有新系统也要包含现有的网络设施。
- 在系统分散的每一处都要描述处理活动和网络连接。
- 描述连接层与层之间的通信协议和中间件。
- 确保足够多的可用网络容量。

实践指导

咨询内部专家，以判定网络是否支持新系统且不破坏现有系统。 ■

9.6.1　网络集成

现代企业需要依靠网络来支持各种不同的应用。因此，新系统的主要部分必须和现有的网络相结合，同时还不能破坏现有应用程序。网络的设计和管理是一项技术含量很高的工作，许多企业都有自己固定的内部技术人员、承包商或顾问，由他们负责进行网络的管理。

新项目的系统分析员在进行网络设计时首先需要咨询企业的网络管理员，以确定现有的网络是否能对新系统提供足够的支持。有时，现有的网络有足够的容量，只需要进行很小的变动，例如，添加新服务器的连接线路或重新配置路由器、防火墙，从而使新的应用程序层可以进行通信。

然而对更大范围的变动进行规划就复杂得多了，这些变动包括大量的容量扩展、添加新的通信协议或修改安全协议等。通常，当网络管理员了解现有网络以及网络相关的应用程序工作方式后，他才会分配新的容量并对一些配置信息进行修改。此时，系统分析员在新系统中的任务是为网络管理员提供足够的资料和时间以保证系统的开发、测试和配置。

9.6.2　网络描述

在分析阶段收集的各种与位置相关的信息可以使用位置图（见图6-37）、活动位置矩阵

（见图6-38）以及活动数据矩阵（见图6-39）来汇录。在网络设计中，分析员需要扩充这些信息内容，扩充的内容包括信息处理位置、通信协议、中间件和通信容量等。

对一个特定的应用程序来说，可以采用许多不同的方法来描述其网络的基础结构。图9-14所示的网络图描述了RMO的客户支持系统的应用层是如何分布在不同的位置和计算机系统之间的。这张图从图8-5总结出核心的结构设计思想，并详细设想了应用程序软件在何处运行，服务器和工作站在何处部署，以及网络资源如何组织。

网络图：应用层是如何分布在不同的位置和计算机系统之间的模型。

图9-14中设想了服务器的部署位置，这通常需要咨询一下网络管理员。Web/应用程序服务器也有可能部署在盐湖城数据中心以外的地方，这样做有利于提高系统的响应时间，并减少专用WAN上的数据通信容量请求。然而，分散的服务器需要在多处进行管理，这无疑将会增加操作的复杂性和费用。在处理服务器部署位置、通信路由、网络安全选项等这些问题时，不仅需要考虑应用程序的要求，还需要考虑到整个企业的策略。

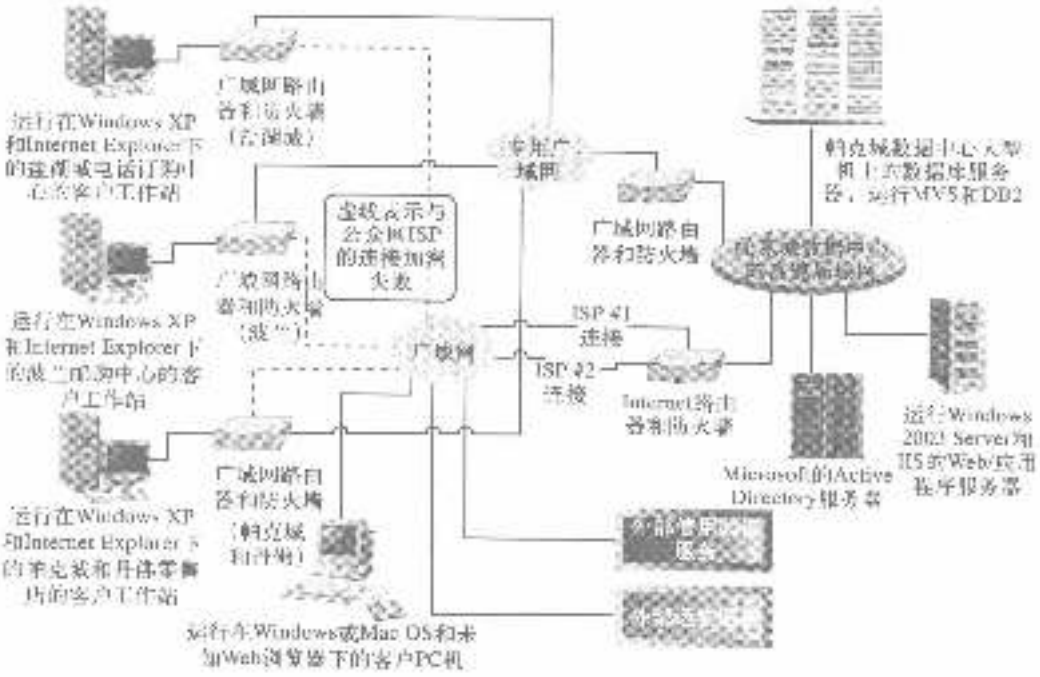


图9-14 RMO客户支持系统的网络图

9.6.3 通信协议和中间件

网络图也是确定通信协议和中间件需求的起点。例如，专用WAN连接必须支持处理Microsoft的Active Directory登录与查询所需的协议。如果这个WAN出现故障，信息则通过Internet在加密（VPN）连接上传送，因此那些连接也需要支持与专用WAN相同的协议。所有的客户必须能够发送HTTP请求，并可以接收活动的内容，如HTML表单及嵌入式的脚本。通过Internet，应用程序必须可以与信用验证以及运输服务通信。防火墙、路由器必须进行配置以支持工作站、客户PC机、Web/应用程序服务器、Active Directory服务器以及外部信用验证和运输服务之间的全部交互行为。转点城数据中心的LAN必须提供至少一种协议来支持主机和Web/应用程序服务器之间的数据库请求与响应。

9.6.4　网络容量

活动位置矩阵与活动数据矩阵中的信息是评估各种LAN、WAN和Internet通信容量需求的依据。图9-15复制了RMO活动数据矩阵（见图6-39）中的数据，包括两个活动（查询条目可用性和产生新订单）和三项数据实体（客户、库存条目和订单）。所有活动、数据实体与位置信息都需要类似的表。图9-15中对每种访问类型的数据大小以及每小时或每分钟的平均访问率与最高访问量都做出了评估。

活动与场所	客　户	库存条目	订　单
查询条目可用性（华盛顿电话订购中心）		R(125 bytes,25/min 平均，250/min 最高)	
查询条目可用性（纽约城零售点）		R(125 bytes,5/min 平均，15/hr 最高)	
查询条目可用性（月佛零售店）		R(125 bytes,5/min 平均，15/hr 最高)	
产生新订单（华盛顿电话订购中心）	C(500 B,2/min平均，10/min最高) R(500 B,8/min平均，80/ min最高) U(500 B,2/min平均，10/min最高)	R(60B,30/min平均，200/min最高) U(60 B,30/min 平均，300/ min 最高)	C(200 B,10/min 平均，100/min最高)
产生新订单（波兰邮购中心）	C(500 B,1/min 平均，10/min最高) R(500 B,4/min 平均，40/ min最高) C(500 B,1/min 平均，10/min最高)	R(60 B,15/min 平均，150/min 最高) C(60 B,15/min 平均，150/ min 最高)	C(200 B,5/min 平均，50/min最高)
产生新订单（纽约城零售商店）	C(500 B,1/hr 平均，5/hr最高) R(500 B,4/hr 平均，20/hr 最高) U(500 B,1/hr 平均，5/hr 最高)	R(60 B,15/hr 平均，75/hr 最高) U(60 B,15/hr 平均，75/hr 最高)	C(200 B,5/hr 平均，25/hr 最高)
产生新订单（月佛零售点）	C(500 B,1/hr 平均，5/hr 最高) R(500 B,4/hr 平均，20/hr 最高) U(500 B,1/hr 平均，5/hr 最高)	R(60 B,15/hr 平均，75/hr 最高) U(60 B,15/hr 平均，75/hr 最高)	C(200 B,5/hr 平均，25/hr 最高)

C=产生新订单，R=读入现存数据，U=更新现存数据，D=删除现存数据

图9-15　部分RMO客户支持系统的活动数据矩阵，更新了数据大小和容量

此时，由于在系统设计过程中，软件层、中间层通信对话框或数据库都还没有设计好，因此访问类型的数据大小只是一种基于经验的猜测。一旦这些部分开始进行更详细的设计或已经设计完成了，分析员就可以优化他们的评估，或者对真实的数据传送进行采样、测量。实际的数据传送容量不仅包括原始数据，还包含了通信协议。

小结

系统设计就是组织和构造系统组件来支持新系统运行（也就是编程）的过程。一个项目

的设计阶段包括了与新系统各种组件设计有关的各种行为，需要设计的组件有：应用程序结构、用户界面、系统接口、数据库、网络和系统控制。同时也可能需要原型来详细说明设计的某一部分或全过程。

设计活动的输入是图表或模型，它们建立于分析阶段。同样，设计的输出也是一组图表或模型，它们主要用来描述新系统的结构和各种编程组件的逻辑关系。输入、设计活动和输出在使用结构化方法或面向对象方法时是不同的。

设计活动开始后，项目管理就变得尤其重要。项目的进程开始产生变化，在设计的过程中，技术人员要加入进来，项目经理必须重视进度表。开发团队和设计活动。在开始编程时，编程人员又加入进来。同时，数据转换和测试数据开发也将开始运行。

设计应用程序结构可以分为两部分：结构设计和详细设计。结构设计使应用程序适合配置环境，包括硬件、软件和网络等。现代应用程序软件通常配置在分布式多计算机环境中，并且通常按照客户-服务器结构或其变体——三层结构或Web服务结构来进行组织。结构设计方案包括把应用程序分解为客户端、服务器端或层次，通过硬件平台分发软件，详细说明所需的协议、中间件和网络。

结构设计方案可以用一个网络图来表示。网络图描述了计算机和网络资源的组织和一些详细信息，如：需要什么样的协议，在何种计算机系统上运行何种类型的应用软件和中间件等。通过活动位置矩阵和活动数据矩阵来确定网络容量的大小，同时涵盖了对消息的长度和容量的评估。

关键术语

architectural design	结构设计
business logic layer	业务逻辑层
centralized architecture	集中式结构
client	客户端
clustered architecture	集群结构
computer network	计算机网络
data layer	数据层
detail design	详细设计
distributed architecture	分布式结构
extranet	企业外部互联网
interface designers	界面设计师
internet	国际互联网
intranet	企业内部互联网
local area network (LAN)	局域网
middleware	中间件
multicomputer architecture	多计算机结构
multitier architecture	多层结构
network diagram	网络图
router	路由器
server	服务器
single-computer architecture	单机结构
three-layer architecture	三层结构
view layer	可视层

virtual organization	虚拟组织
virtual private network (VPN)	虚拟专用网
Web services architecture	网络服务架构
wide area network (WAN)	广域网
World Wide Web (WWW)或Web	万维网

复习题

1. 系统设计的主要目标是什么？
2. 分析和设计的区别在哪里？列举SDLC中设计阶段的活动。
3. 为什么在设计阶段中项目管理非常重要？在设计阶段项目经理可以使用哪些工具？
4. 解释集中式和分布式结构的差别。
5. 解释集群式结构和多计算机结构在集中式系统中的差别。
6. 说明Internet、Intranet和Extranet的相同点和不同点。
7. 以开发客户-服务器信息系统为例，描述客户-服务器结构并列举出关键的结构设计问题。
8. 列举并简要描述三层结构中每一层的功能。
9. 中间件有什么作用？
10. 描述网络设计的过程。
11. 在网络设计过程中，系统分析员和网络管理员分别起什么作用？
12. 什么是网络图？它传达了什么样的信息？分析员从何处获取这些信息？
13. 分析员如何评估所需的通信容量？什么样的分析阶段模型用作输入？
14. 什么是Web服务结构？试举例说明该结构在业务业务系统中的潜在用途。

思考题

1. 讨论客户-服务器计算从文件服务器到多层应用再到基于Web的应用的演变过程。是什么导致了这种演变过程的发生？在未来5年或者10年内，你认为网络计算将处于何种位置？
2. 假定大宗支付处理系统的配置环境包括以下组成部分（这些假定的成分来源于第8章的第一个实验练习）：
   - DB2数据库管理系统，运行在IBM S/390主机的OS/390操作系统上；
   - WebSphere应用服务器，运行在IBM zSeries 900主机的Z/OS操作系统上；
   - 用Java编写的基于组件的应用软件，由其他内部和外部系统执行。

   这个系统的关键结构设计方案是什么？这个方案应该何时产生？由何人进行处理？简述结构设计方案生成以后的后续设计任务。后续步骤在多大程度上可以并行执行？
3. 在问题2的答案基础上，为结构设计方案开发一个网络图。

实验练习

1. 与中等规模或大规模开发项目的主要分析员一起讨论研究该项目从分析到设计的转变。如何以及何时来制订关键结构设计方案，如：自动化系统边界、网络设计和支持设施？谁制订该方案、是否要在项目的后期调整结构方案？如果需要调整，如何做？为什么这样做？
2. 编一个基于浏览器并使用TCP/IP标准的应用系统的例子，同时，解释它是如何工作的。并且给出界面样例和报告。列举每个中间件并描述其功能。列举所用的每个协议并指明该协议属于何种标准体系或者体系组。
3. 检查图9-14的RMO网络图。并注意信用验证和运输服务与外部服务提供商之间的连接。此处研究它们基于Web的在线服务能力，并且描述用于客户端与它们的服务进行交互的协议。

实例研究

房地产多编目服务系统

在第8章中讨论了针对应用配置环境的任何时间和任何地点需求的含义，并且也描述了为完成需求所需硬件、网络和软件构造的类型。假定通过详细说明一个三层结构来解决问题，使用运行Web浏览器的普通PC机实现可视层。画出代表你选择的解决方案的网络图。

现今，基于计算机的房地产编目必须包括图形数据，例如，除了属性的文本描述之外，还包括静态和动态图片。假定每小时访问10个、100个、1000个列表，那么在你的网络设计中，这种数据对数据通信需求分别产生什么影响？

对落基山运动用品商店（RMO）实例的再思考



在第8章中，针对运行了Linux和Oracle数据库服务器下的基于Apache Web服务器的RMO，要求你考虑一种配置方案。调整图9-14所示的网络图来反映这种配置方案。客户端工作站和客户PC机需要有些什么变化？中间件和通信协议需要有些什么变化？在位于帕克城数据中心的客户工作站和服务器中，评估所需的数据通信容量是否会发生变化？为什么？

关注Reliable Pharmaceutical Service



假定有与第8章的Reliable药品服务案例中相同的事实。同时假定你是开发团队的项目经理。你的公司是RxTechSys，它为零售商和医院药房开发和销售软件。同时它决定用Reliable的项目来扩大潜在的市场份额。RxTechSys和Reliable将合作开发新的软件。RxTechSys公司将向其他公司销售最终产品，同时，每销售一个软件，它就要向Reliable支付一次版权费用。

RxTechSys公司已经从事20年药品软件行业了。该软件的最新版本是基于Web应用的，它的开发平台是Microsoft的.NET平台。其主要功能包括：库存控制、购进、账单和处方通知。它们都独立地运行在.NET网络服务器上。为准备对应于Reliable公司RFP的组成部分，你决定RxTechSys公司现有的系统要适应Reliable公司的以下需求：

- 调整现有的基于浏览器的处方入口，使它能够在VPN网络中对来自多个客户的数据输入进行操作。这一显著的改变应归于数据容量的扩充和安全需求的增加。
- 操作订单的软件不得不重新编写。
- 由于现有的系统假定由同一个机构来管理所有患者的医疗保健（通过医疗补助和医疗保险，第三方机构可能参与进来），所以账单软件需要进行大范围的调整。
- 现有系统的其他组成部分可能只需稍微调整或不需要进行调整。

Reliable公司已经向你提供了完整的面向对象分析模型。在合同谈判阶段，你要验证这些模型的质量。你的任务是通过设计和实现推动这个项目向前发展。

Reliable公司同时也派出一名具有一定计算机经验的操作经理到你的团队全职工作，她有权指派其他你需要的Reliable的员工参与这个项目的工作。现在你已经有4位开发人员、两名资深设计人员，而且他们都参与了最新版本RxTechSys软件的设计。

开发一个设计方案并制订接下来4～6周的进度表（假定该项目预期时间为10月）。在接下来两周内，项目必须做出哪些设计决策？谁来决定它们？怎样设计？之后又如何开发处理——必须执行哪些任务？以何种顺序完成？你将怎样管理和控制项目？

参考资料

Robert Orfali, Dan Harkey, Jeri Edwards. *Client/Server Survival Guide, Third Edition*. Wiley, 1999.

第10章 传统设计方法

学习目标

阅读本章后，你应具备如下能力：

- 描述用传统方法设计应用程序结构的步骤
- 开发系统流程图
- 用事务分析和变换分析开发结构图
- 为结构化模块编写伪码
- 解释如何在传统方法中使用三层设计

本章要点

- 采用结构化方法进行应用程序结构的设计
- 自动化系统过程
- 系统流程图
- 结构图
- 模块算法设计：伪码
- 结构化应用程序设计与其他设计任务的集成
- 三层设计

剧院系统有限公司：新事物，旧事物

Bernard 关上办公室的门，激动地对他的同事Stana说："我不明白Jim为什么坚持让我更新系统流程图和结构图。我们应该彻底放弃这些，用面向对象（OO）设计方法从头开始。我在学校时是曾画过一些传统图表，但大部分时间我们还是学习面向对象的图表和技术。我觉得我在像是被叫去用铁锤和锯子装电脑。"

Bernard是剧院系统有限公司新招的MIS专业大学毕业生。这家公司主要面向美国中小型剧院销售财务报告软件并提供技术支持。Bernard被聘用时正值公司的一个升级项目从分析阶段转入设计阶段。尽管公司软件定期更新，不停地做些改进和加入新的特征，但是它仍然缺少一些现代的功能，如基于Web的界面和可升级的多层结构。

Stana，一名已经在这家公司干了近4年的员工，回答说："你必须记住两件事。第一，这里许多信息系统的员工并不很了解面向对象的分析与设计技术。我们的第一版软件以及它的分析与设计文档都是20世纪90年代开发的。之后所有的更新都在不断增加，所以我们没有必要销毁原来的设计模型从头开始。而且，十多年来，系统重要的大模块都一直保持不变。"

"第二，工具和任务之间的适应性也是问题。如果我们的目标是要开发一个新系统，而这个系统要涵盖小至零散的、大至全国性的剧院系统，而且能够随意扩展，那么我们基本上肯定会采用最新的分布式软件技术、面向对象程序设计语言以及最适合的面向对象分析和设计工具。同时，我们也会抛弃现有的大部分代码，从头开发整个系统。但是，我们当前的项目只要求用尽可能少的修改给基于C语言的系统换上前端网页浏览器界面。结构化设计模型很适合处理现存的C语言程序和功能。"

“那么我要怎样用结构化技术表示网络界面以及客户-服务器间的交互呢？”Bernard问。

Stana回答道，“这儿有点窍门。你可以把Web服务器当成一个应用软件程序的容器。而这些应用软件程序通过Internet或Intranet实时连接与Web浏览器进行通信。在结构化设计中，主要软件单元是程序和模块。所以在运行系统中，模块就是由许多基于菜单的前端包裹起来的C函数，这些函数被整合进负责处理各种事情的少量复杂程序中。你在这次更新中最重要的任务之一是把这些大程序分解成小的，并把现存系统中用于实现用户界面部分的函数从C代码中去除，改在网页代码中实现。剩下的函数就是一些应用逻辑，可以打包进一些小程序模块，从而用Web服务器脚本调用。每个小程序就是系统流程图中的一个框，用一个结构图描述。你一开始可以从现有的结构图中剪贴复制得到草图。”

Bernard开始松了口气，但一会儿又觉得还是有些迷惑，担心地说：“Jim会在这周末给我工作。我担心自己会犯什么大错误，那样他会觉得自己被错我了。你能在我见他之前帮我检查一些工作，给些指导吗？”

Stana冲Bernard一笑，让他放心，“Jim安排你和我在同一个办公室工作。尽管我有其他工程任务，但他已经跟我说过在你需要时帮帮你。软件开发要想成功，离不开团队合作，被解雇的往往都是那些从不寻求帮助的人。所以，我建议你在上午剩余的时间里设计出快餐店收银台的登录和确认模块，这样我们可以在午餐后坐下来讨论一下。”

概述

本章将描述传统软件设计方法。首先将概述结构化模型、模型开发流程以及相关术语，还将讨论如何用自动化边界信息注释数据流图。然后，我们探究如何使用系统流程图、结构图和模块伪码把从分析阶段模型得来的信息转换到设计模块中。之后，我们讨论如何把传统软件设计和其他设计阶段的内容整合起来。最后通过考察如何用传统方法设计三层结构做结尾。

如开篇案例所述，传统软件设计和结构化设计模型相对来说比较旧了。它们通常用于过程化程序设计语言开发的系统，并且很适于描述批处理和在线组件的系统。目前，大部分新系统是用面向对象程序设计语言开发的，因而传统的系统设计模型的受欢迎程度越来越低。然而，如此例中描述的一样，现在使用的许多较老的系统都是用传统的方法和模型进行设计和存档的，而且传统设计概念如耦合、内聚以及自顶向下则分支是面向对象设计方法的基础。所以理解这些概念是非常重要的。最后，传统模型有时候也适合新的软件开发方法和范例，如多层结构软件。所以，分析员应该充分了解传统的设计方法。

10.1 采用结构化方法进行应用程序结构的设计

应用程序结构由执行系统特定功能的若干个应用程序组成。应用程序的设计必然与数据库和用户界面的设计相关联。但为使本章易于理解，我们只把注意力集中到软件设计本身。

你也许已经使用第三代语言编写过业务软件了，像VB、C、COBOL或Pascal等。第三代语言是以模块的形式来组织的，这些模块呈树形层次排列。最顶端的模块叫做总模块或主模块，中层模块叫做控制模块，叶子模块（位于端点上的模块）是详细细节模块。它们包括了程序的大多数算法和逻辑。一个模块是执行某个功能的一小段程序代码。计算机程序是由一系列模块组成的，这些模块被编译成一个单独可执行的实体。稍后我们将详细介绍如何用结构图来描述程序的设计。

模块：计算机程序的可标识部分，用来完成某种特定的功能。

计算机程序：由一系列模块组成的可执行的实体。

在大系统中，单独一个程序往往不能完成所有需要的功能。有时，用一个程序完成实时

活动，而用另一个程序完成每天都要执行的周期性功能。其他的程序可能执行特殊的功能，如备份数据或产生年度财务报告。这些可单独执行的实体或程序构成了整个系统。

我们用系统流程图记录整个系统和各个子系统的结构。系统流程图标识了每一段程序及其存取的数据，也表明了不同程序、子系统、相关文件和数据库之间的关联关系。它记录了整个系统的体系结构。本章的后续章节中将介绍如何设计系统流程图。

系统流程图：描述系统内计算机程序之间所有数据流的图。

最后，项目组必须设计每个模块的内部逻辑。包含模块逻辑的内部算法通常用伪码来记录。如果你正在学习程序设计课程，在正式编程之前，你可能需要用伪码将算法写出来。伪码非常像第6章所描述的结构化英语，它与用来描述模块逻辑的结构化程序设计说明很相似。

伪码：与结构化编程类似的语句，它描述了模块的逻辑。

一般来说，分析员用自顶向下的方法进行设计，以数据流图、结构化英语描述的详细文档以及详细的数据流定义作为设计模型和开发文档的输入部分。图10-1说明了这个过程。分析员使用一种具有中间形式的DFD，称为带有自动化系统边界的DFD。自动化系统边界将系统设计的系统自动化部分从手工部分分离出来，从而也就确定了哪些部分需要包含在设计中。这种扩展的DFD正是图10-1中所示的设计模型的数据来源。

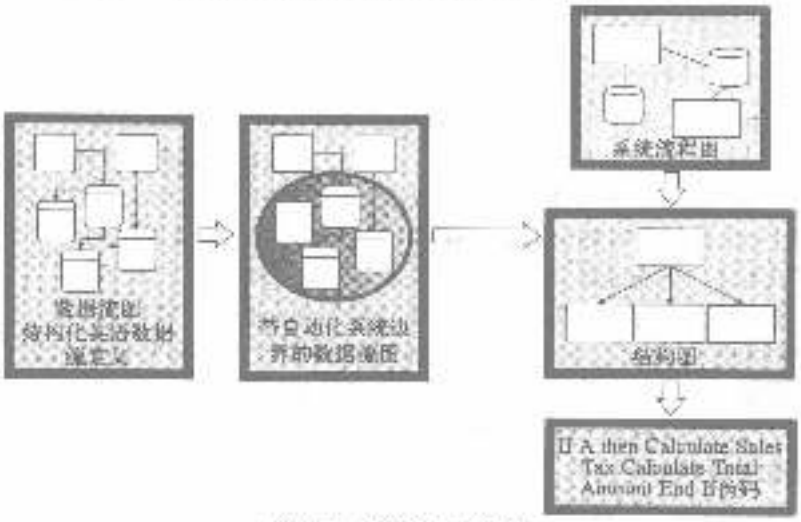


图10-1　结构化设计模型

下面章节将介绍图10-1所示的结构。首先，讨论自动化系统边界；接下来，剖析系统流程图的开发；然后，详细描述设计结构图的方法；最后，给出写伪码的格式和方法。

10.2　自动化系统边界

自动化系统边界将数据流图的处理划分成手工处理部分和系统处理部分。在系统分析阶段，我们分析这些业务事件，并描述这些事件的所有过程，但是没有区分哪些是人工处理的，哪些是自动完成的。但是，要开发计算机系统的设计，我们必须标识出哪些是要求系统自动完成的。

图10-2所示为一个包括自动化系统边界的典型数据流图。该图不仅说明了系统边界，而且也标出了程序界线。前者标识了整个自动化系统，后者将DFD划分成独立的程序。这个图是设计的第一步，它确认了程序是什么，以及这些程序中包含了哪些处理过程。在本书中，我们把程序定义为可独立执行的实体。

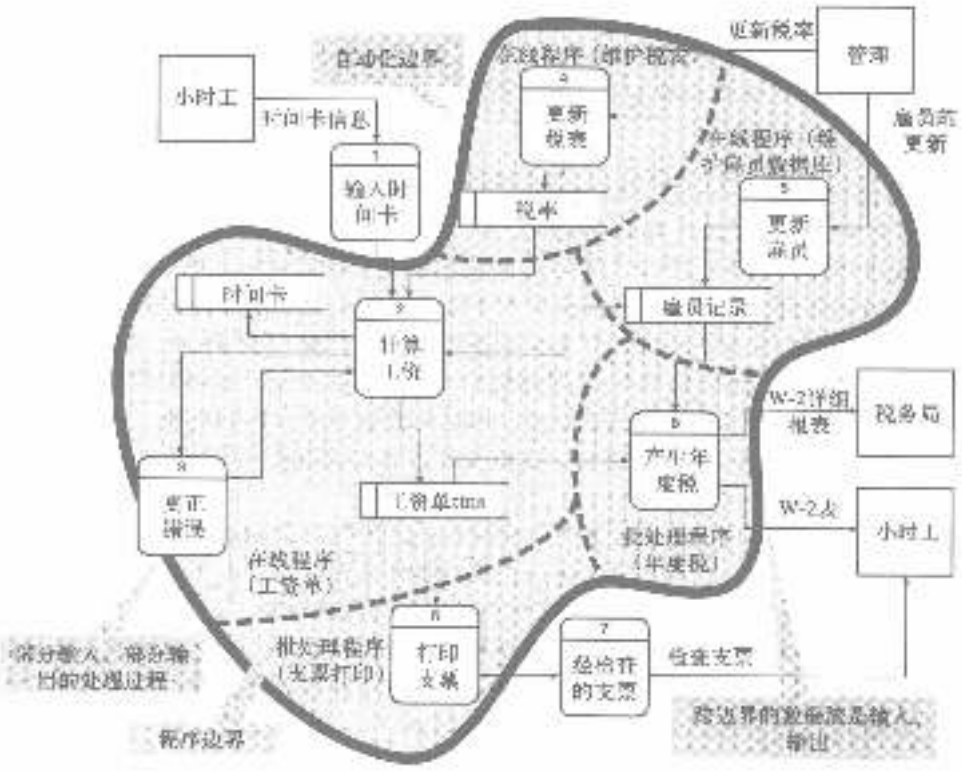


图10-2 带自动化系统边界的数据流图

处理过程既可以出现在系统边界内，也可以在系统边界外。系统边界外的处理是手工处理过程，例如，排序并审核文件资料，输入用户订单或查看收到的货物。边界内的处理过程能够以在线处理或批处理模式进行处理。通常来说，在线处理在每天的工作时间内都处于激活状态，批处理则每晚或定期激活一次。有些时候，系统边界贯穿一个处理过程，说明这个处理是一个中层或高层的处理，它需要一个更详细的图来对其进行分解，分解后的子处理过程可能有些在系统边界内，也有些在系统边界之外。

数据流可以在系统内部、外部，或穿过系统边界和程序边界。其中，穿过系统边界的数据流是格外重要的，它们代表了系统的输入和输出。换句话说，程序接口的设计（包括用户界面设计和与其他系统的接口设计）是由穿过边界的数据流定义的。在最终的系统中，这些数据流将是用户界面中的表单、报表，或是与其他系统进行转换的文件或远程通信。与此相似，穿过程序边界的数据流表示了程序间的通信。在最终系统中，这些数据流仍是程序之间转换的文件或远程通信。

图10-2是一个高层数据流图，它展示了工资单程序所有的主要处理过程。系统边界也可以画在每一个数据流片段中，进一步表明哪些是系统内部处理过程，哪些是系统外部处理过程，以及那些低层数据流穿过系统边界等更进一步的细节。

10.3 系统流程图

系统流程图是对组成一个完整系统的计算机程序、文件、数据库、以及相关手工处理的一种表示方法。可以根据一些相似的特点把处理过程分为程序组和子系统，这些相似的特点可以是时间间隔相同（如按月执行的处理）、存取数据相同（如更新员工信息的所有处理），

用户组织（如生成市场部报表的所有处理）等。这样产生的程序组和子系统有数据流、控制流、永久存储数据间的交互等复杂的依赖关系。在分析活动过程中，经常要建立系统流程图。例如，RMO客户支持系统的子系统在分析阶段就已定义好了（见第6章），同时分配给每个子系统的事件集构成了程序模块。

实践指导

系统流程图可以帮助建立应用程序结构文档，显示子系统、输入、输出和数据存储。 ■

系统流程图用图形的方式描述了哪些子系统是系统自动完成的，哪些需要人工的参与，还画出了数据流和控制流。系统流程图主要用于描述大的信息系统，这种大的信息系统可由独立的子系统和大量的程序块构成。它也用于描述需要执行批处理的系统（例如，处理银行交易、工资审核和有效账单的系统）。这类系统的一个普遍特点是，可将处理按特定的执行顺序分成离散的多步（例如，输入交易的验证、用交易数据对主文件的更新、周期性报告的产生）。许多批处理系统使用了扩展的文件，这些文件或者作为数据库的补充，或者干脆替代了数据库。

系统流程图最初广泛用于记录批事务文件信息处理的程序之间的处理过程和数据流。一般来说，在这些系统中，一个程序会产生一个每日事务处理文件，而另一个程序将处理这些事务并更新主文件。还有一个程序则用于产生系统需要的各种报表。

在今天较新的系统中，许多处理需要在事务输入时实时进行，这些系统通常要更新关系数据库而不是主文件。集中式数据库管理系统现在也包括许多以前只能由单个程序完成的处理。由于这些处理过程更加集中于某个程序或子程序中，所以流程图也变得更为简单。虽然如此，系统流程图仍可应用在这些较新的系统中。但是，因为今天并发的系统变得更为复杂和全面，所以仍然可从系统流程图中看出每部分是如何协调工作的。

现在，许多业务系统既有实时部分，也有批处理部分，例如，使用信用卡购物就是被验证的最简单的实时处理过程，因为它可能要求实时提交。然而，每个月的账目说明和用户付款则是典型的批处理过程。系统流程图可用来描述该类系统的整个体系结构并表明实时处理部分和批处理部分间的关系。

图10-3展示了系统流程图中的常用符号。在软件行业中，这些符号是很常用的，有时也会使用这些符号的一些变体。



图10-3　系统流程图中的常用符号

图10-4是工资系统的系统流程图，在图10-2中我们给出了它的数据流图。请注意，系统流程图标出了整个系统的文件、程序以及人工处理部分。通过对文件媒介、磁盘或磁带等的确认可以增加物理实现的描述。通常，实际中还会包含附加的系统功能和文件（如备份文件和历史文件）。虽然图10-2和图10-4的信息是很相似的，但它们所强调的重点却不同，系统流程图重点在于物理对象的实现（如可执行的程序、文件和文档等）。

从图10-4可以看到工资系统程序有4个输入、3个输出。输入分别为时间卡、税率表文件、职工数据库和更正信息。产生的输出包括错误报告、工资事务文件和经更新的工资历史文件。此外，两个程序分别负责维护税率表和职工数据库（这些程序均可独立运行），另两个程序分别负责打印账单和产生年度收入税报告。

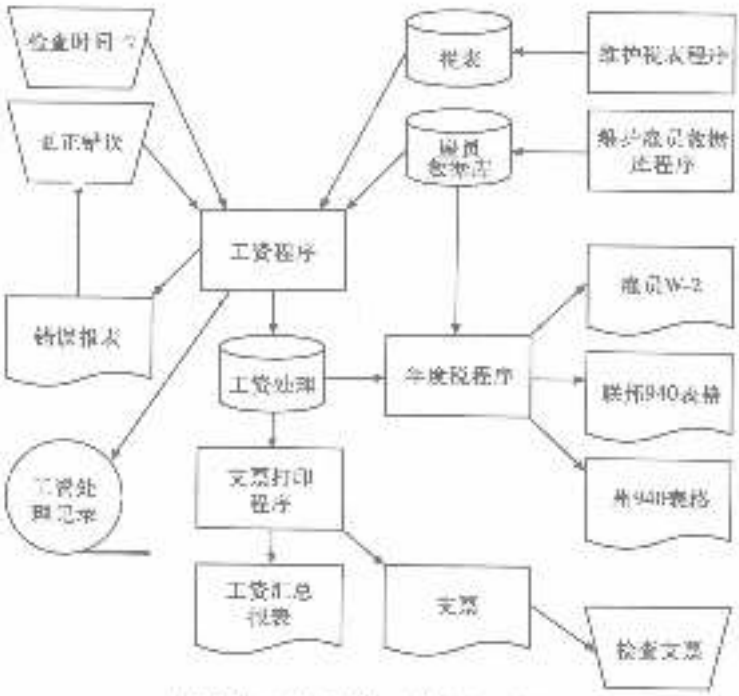


图10-4 工资系统的系统流程图样例

图10-5所示为RMO公司开发的系统流程图样例。其中，程序的4个主要部分与图6-10所示的数据流图中所定义的子系统相对应。在图6-12所示的订单输入子系统中，每一个子系统都包含自己内部事件的数据流图的片段。在系统流程图中，每个数据存储转换为相应的数据库文件。可以看出，创建系统流程图需要主程序步骤的结构设计、数据库的结构设计、主界面的标识以及主要输出的标识。

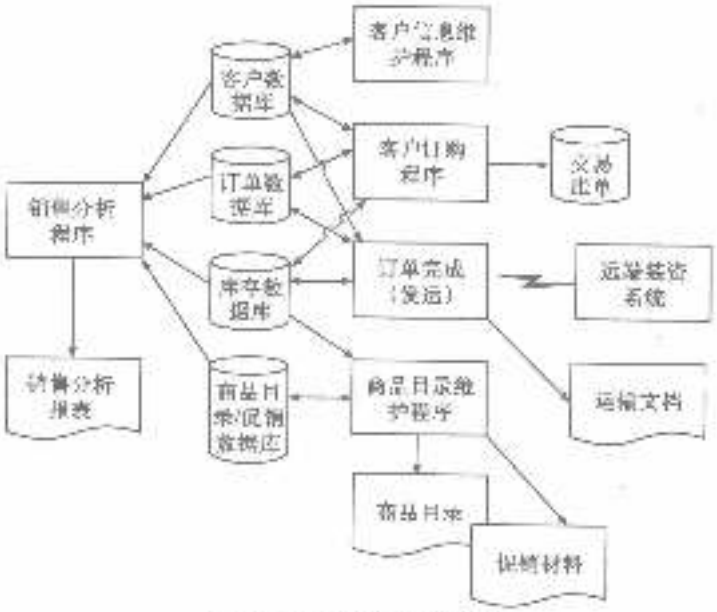


图10-5 RMO的系统流程图

图10-5中增加了一个在图6-10中没有出现的子系统。重新考虑第8章讨论的系统范围制订和自动化程度，RMO决定采用更高的自动化程度，以取得一张销售分析报告。在这个实例中，项目组定义了一个新的子系统而不是把报表增加到已存在的子系统上。

10.4　结构图

结构化设计的主要任务是自顶向下地分解新系统中给定程序所要执行的功能。系统结构图中每一个独立的程序都执行一系列的功能，使用结构图可以将程序功能有层次地组织起来。这一节首先解释结构图是什么，以及怎样理解结构图。我们将解释结构图是如何与在系统分析阶段建立起来的数据流图相关联的。最后，我们还将说明如何使用细化的数据流图来建立结构图。

结构图的层次描述了系统每部分的功能和子功能。例如，某程序有一个叫计算工资总额的功能函数，它可能的一些子功能函数是：计算基本工资、计算加班费和计算税费。在结构图上，我们用矩形框表示这些功能，每个矩形框代表了一个模块。

结构图：用来展示一个计算机程序模块间关系的层次图。

实践指导

使用结构图为系统流程图中的每个程序的模块化设计建立文档。由于每个高层模块通常都是基于某一活动或某一事件触发的用例的，因此传统结构图可以说是基于用例驱动的。 ■

结构图的基本组成部分是模块。模块用来标识一个功能。图10-6所示为工资系统计算工资总额模块的简单结构图。矩形所表示的模块是相对简单、独立的部分。高层模块是"控制"模块，它控制执行流。低层模块是"工作"模块，它包含实际执行功能的逻辑程序。在图10-6中，计算工资总额模块的功能只是按正确顺序调用低层模块来完成工资计算功能而已。



图10-6　计算工资总额模块的简单结构图

请注意结构图是如何简单直接地组织程序，使它完成计算工资总额功能的。在程序设计课上，我们知道模块化程序设计是编写程序的好方法，它易于理解和维护。将一个复杂的程序分割成小的模块会使程序的早期设计和维护变得更为容易。结构图的建立有一定的规则和方针，其关键是，程序是分层的，且模块按高内聚、低耦合的方式组织在一起。以后我们将更详细地描述好模块的一些特性。

模块之间的连线表示高层模块对低层模块的调用关系，其上的小箭头表示模块间传递的数据，以及各个模块的输入和输出。在结构图上我们还不能了解模块内部的内容。但我们想知道的只是模块以何种方式实现其名称所表示的功能，如何使用输入数据，以及如何产生输出数据。

图10-7所示为用来画结构图的常用符号。结构图中的矩形表示模块。一个模块可以表示一段代码，例如，用COBOL编写的一段或一部分程序。在其他语言中，模块典型地可表现为一个函数、过程或子程序。作为程序段的模块可以是子程序（如在FORTRAN和BASIC中），

程序段或子程序（如在COBOL中），过程（如在Pascal中）、函数（如在FORTRAN，C，C++中）。模块也可以是一个能独立编译的实体，如一个完整的C语言程序。带双竖线的矩形框代表一个现成的模块或在多处被用到的模块。是否使用带双竖线标记是可选的。

图10-7e中展示了一个高层模块对低层模块的调用。当一个模块激活一个低层模块以便执行所需的服务或计算时，就发生了程序调用。在不同的编程语言中，程序调用的方式是不同的，例如，在C和C++中是函数调用，在Pascal中是过程调用，在FORTRAN中是子程序调用。在每一次调用中，控制权由调用模块传向被调用模块，接着被调用模块执行一系列程序语句，当调用过程结束后，被调用模块立即把控制权返回给调用模块，程序紧接着执行下面的声明或指令。

程序调用：控制从一个模块转移到下一层模块以便执行一个需要的服务。

图10-7c还表示了数据是如何在模块间传递的。带空心圆的箭头叫**数据耦合**，表示输入和输出模块的数据。数据耦合可以是一个单独的数据项（如一个客户的账号），也可以是更高层次的数据结构（如一个数组，记录或其他数据结构）。结构图中每一层所用的耦合类型，往往与细节分层的有关原则相一致。也就是说，靠近结构图顶端的模块，它们的耦合往往使用高度集中的数据结构，而结构图底端的耦合往往是一些单独的数据项、标记及相对较小的数据结构。

数据耦合：在程序调用中模块间传递的单独的数据项。

带实心圆的箭头是控制耦合标志，在模块间使用的标识是表示某种结果的内部信息。我们经常使用低层模块产生的标识表示结果（如某记录经验证有效）。另一个常见的用途是标识文件已经到了结尾。

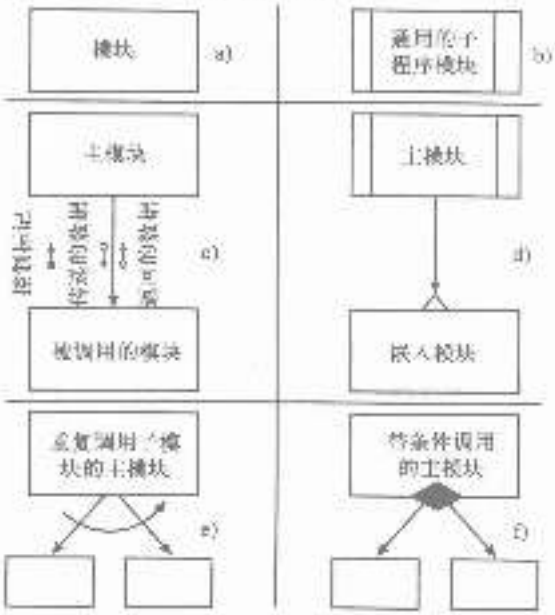


图10-7 结构图符号

图10-7d是从结构图中分割出来的低层模块，它一定包含在程序的某个调用模块里。这种分类技巧用于保证强调突出了模块实现的功能。图10-7e和图10-7f是程序调用的两种可能选择。在图10-7e中，我们用标识符来表示几个模块的迭代。在图10-7f中显示的是条件调用低层模块，即仅当某一条件成立时，模块才会被调用。

图10-8所示为一个更完整的工资计算系统视图，它包含了图10-6中最初计算工资总额的功能。要注意的是，图10-8所示的完整结构图是基于产生工资条时间已到这个临时事件引发的系统活动的，在工资系统的分析阶段，分析员确认结算职工工资这一事件每周末要发生一次，即系统每周运行一次。当然同时还要确认许多其他的事件。

结构化程序设计的基本思想是，每个模块完成且仅完成一项特定功能。最顶端的模块是主模块，它的功能是调用下一层的模块，把信息传给它们，并得到返回信息。每一个中间层模块的功能是控制它的下层模块的处理过程，这些中间层模块均有控制逻辑和错误处理逻辑（不是由低层模块处理）。在端点或叶节点上的模块包含执行程序功能的实际算法。这种程序设计方式将程序控制逻辑从业务逻辑算法中分离出来，使编程更为容易。

从高层模块到低层模块的箭头表示程序调用，调用的顺序总是从左到右的。请注意，结

构图在表示调用结构时仍具有严格的层次。一个低层的模块永远不会调用高层模块。主模块下的一个带弧度的箭头表示循环调用三个模块。换句话说，主模块有一个内部循环，依次循环调用同一层次上的三个下层模块。

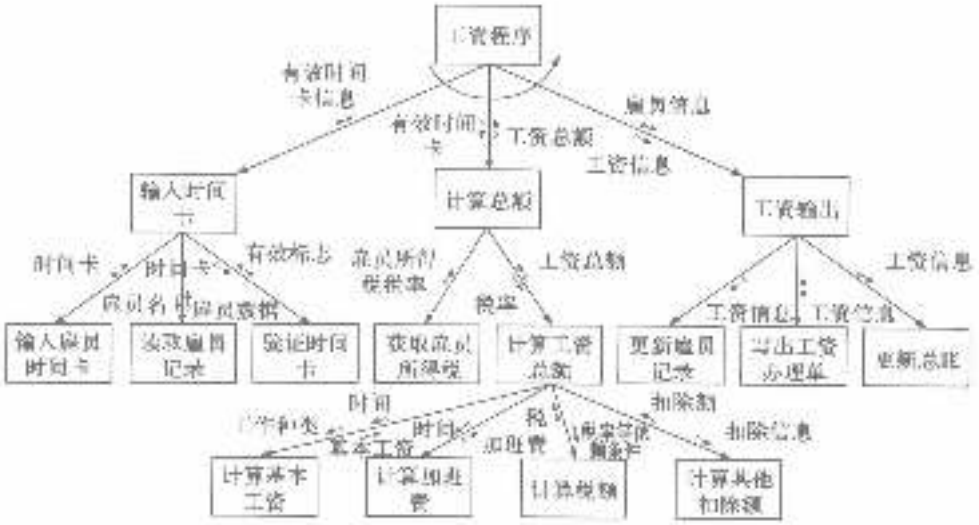


图10-8　整个工资系统的结构图

此例中，信息流往下流动并返回。通常，高层模块会要求低层模块提供某项功能，并向低层模块传递必要的输入信息，低层模块将返回所要求的信息，也可能返回一个标志作为控制信息通知高层模块。下层模块已成功完成了任务。观察图10-8中的输入时间卡子层，可以发现，它将职工时间卡信息传给了主模块，接着将包含职工名字的信息传到下一个模块。这个模块在读取职工信息后，将信息返回。最后，职工数据和时间卡信息被传到最右边的验证时间卡模块。它向上传递一个标志，表明验证成功或失败。如果验证失败，将显示错误信息并转入错误处理子程序。在实际的程序中，这些过程是很复杂的（尤其是错误处理部分），在这里我们并没有全部列出。

结构图中包括了从外界读取数据的模块。这些模块的设计必须与用户界面、其他系统的接口和数据库的设计相一致，这一点是相当重要的。结构图还必须与系统流程图相一致。在开发过程中，如果结构图发生了改变，项目组也要相应地更新系统流程图。

10.4.1　开发结构图

设计结构图的目的是为了给程序建立模块层次结构。树型结构的结构图有根模块和分支模块。一个子树就是从整个树中分割开的一个分支。当子树重新放回到原来的树上时，子树的根仅仅是整个树的另一个分支。为什么说这一点很重要呢？因为我们从中可以了解到，我们能先一块块地建立结构图，最后把它们组成一个完整的结构图。

图10-5是RMO客户支持系统的系统流程图。每一个主要的程序对应于按事件划分的一个子系统，每个程序有自己的结构图。然而，正如在图6-10中看到的一样，每个程序，即子系统也可包含若干个事件，每个事件对应于按事件划分的数据流图中的一个处理，而每个处理在基于事件表的数据流图片断中被详细描述。

有两种方法开发结构图：事务分析和变换分析。事务分析使用系统流程图和事件表作为输入建立树型结构的顶端模块，即产生主程序的主模块和第一层被调用模块。变换分析使用

数据流片段作为输入建立子树，为程序中的每个事件建立一棵子树。每棵子树的主模块对应于上层结构图的一个第一层分支。我们将分别介绍这两种方法。

事务分析：基于数据流图的结构图开发，用来描述多种事务类型的处理。

变换分析：基于数据流图的结构图开发，用来描述输入—处理—输出数据流。

1. 事务分析

在事务分析中，第一步是检查系统流程图和每个大程序块，如图10-5中的用户订单部分。在图10-9中，我们复制了图6-13中的内容，包括订单输入子系统接事件流分的数据流图，并由这个子系统5个事件派生出5个处理。这五大事务处理彼此不同且必须包含在子系统中，它们分别是：审核项目有效性、创建新订单、修改已存在的订单、产生订单汇总报表和产生事务处理汇总报表。

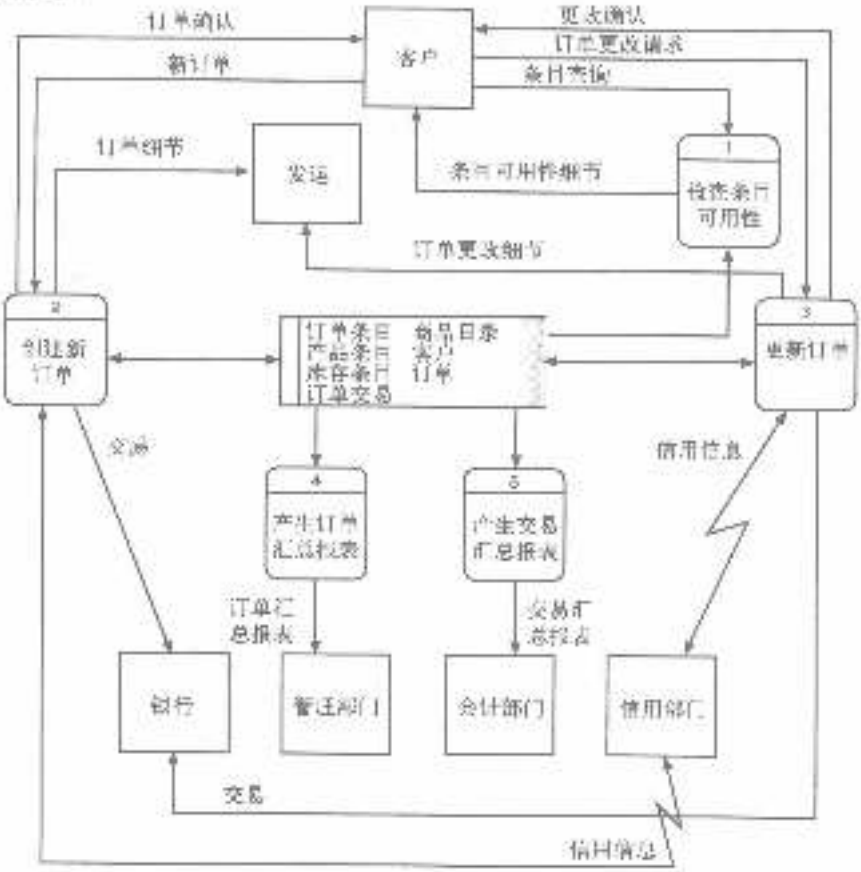


图10-9 订单输入子系统的事件划分数据流图

图10-10是这个程序的基于事务分析的结构图，如前所述，事务分析要识别每个程序必须支持的独立事务，并且必须为每一个单独的事务建立一个分支。其实，这个程序至少在最底层上，也就是用户显示界面的模块，它允许用户选择一种事务处理方式，然后激活相应的模块来执行这个处理。此图没有写出每个事务处理模块下的其他额外细节，每个以功能命名的处理模块都是一个主模块，主模块下的子树将根据该事件的DFD片断采用变换分析建立。

这个结构图的数据耦合非常少。实际上，所传递的唯一信息来自事务处理选择模块的事务选择。控制模块用这个信息来选择正确的处理模块，在这些处理模块下的子树将显示一个适当的界面来接收和传递所要求的细化的信息。

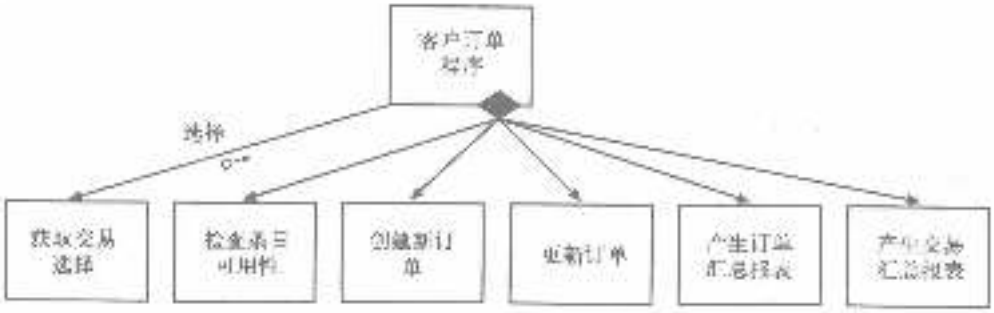


图10-10　客户订单程序的高层结构图

2. 变换分析

变换分析基于计算机程序将输入数据“变换”成输出信息的思想。用变换分析建立的结构图通常有三棵主要的子树：一棵输入子树获得数据，一棵计算子树执行算法，一棵输出子树显示结果。图10-8是一个用变换分析建立结构图的好例子，因为处理按照要求将输入的时间卡变换成工资单输出。要注意的是：数据流图片段通常按“输入—处理—输出”的模式建立，同时结构图把基于这些数据流图片断的处理变换为自顶向下结构的程序模块。

有时数据流图片断可分解为更细化的图，这些细化的图能提供比结构图更多的细节。图10-11～图10-14是RMO系统的变换分析的一个例子，图10-11是创建新订单事件的数据流图片断，图10-12是该事件用于变换分析的分解视图，结构图直接从数据流图开发而来，其基本思想是：结构图中的叶子模块来自于数据流图的处理细节，结构图的中间层主模块来自于数据流图的中间层处理——即分解得到低层模块的那些处理过程。因此，结构图的结构直接反映数据流图的层次和嵌套。此外，还必须开发额外的主模块来为结构图提供正确的结构。

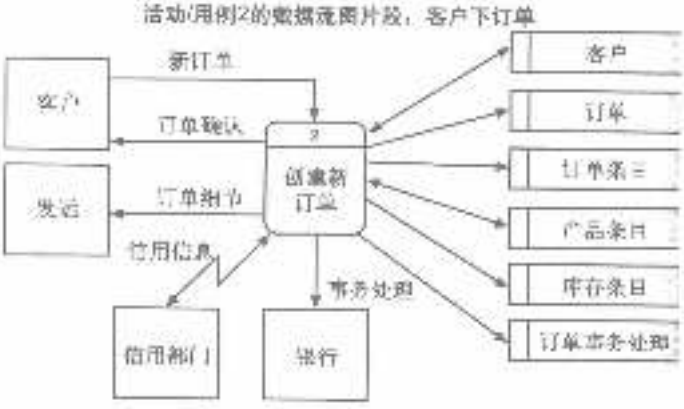


图10-11　创建新订单DFD片断

如上所述，用变换分析方法开发的结构图其一般格式为“输入—处理—输出”。从一个数据流图片断建立结构图的方法包括以下几步：

(1) 确定主要的信息流。这是从输入形式变换到输出形式的主要数据流。

(2) 找出输入流到输出流之间最基本的变换过程（见图10-13）。输入数据流称为传入数据流，输出数据流称为传出数据流，中心过程称为中心变换。

(3) 重画数据流图，将输入放在左边，输出放在右边，变换中心处理放在中间。如果这是一个被细化了的数据流图，则在图中加上父处理。不重要的数据流可以省略，以使图形更为简洁。图10-13就是一个重画的数据流图示例。

传入数据流：向一系列处理输入的数据流。
传出数据流：从一系列处理输出的数据流。
中心变换：在一个变换分析类型的数据流中的中心处理。

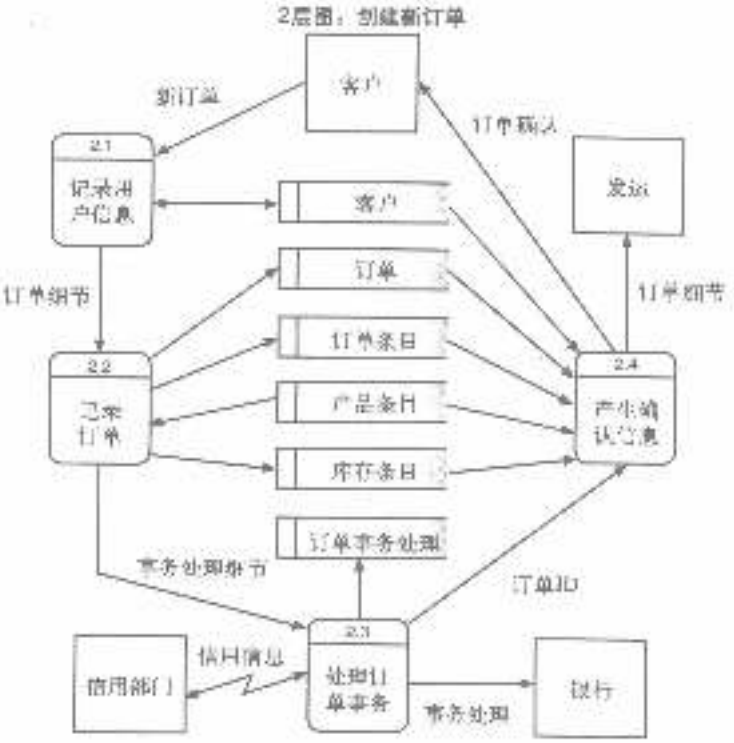


图10-12 创建新订单DFD的分解视图

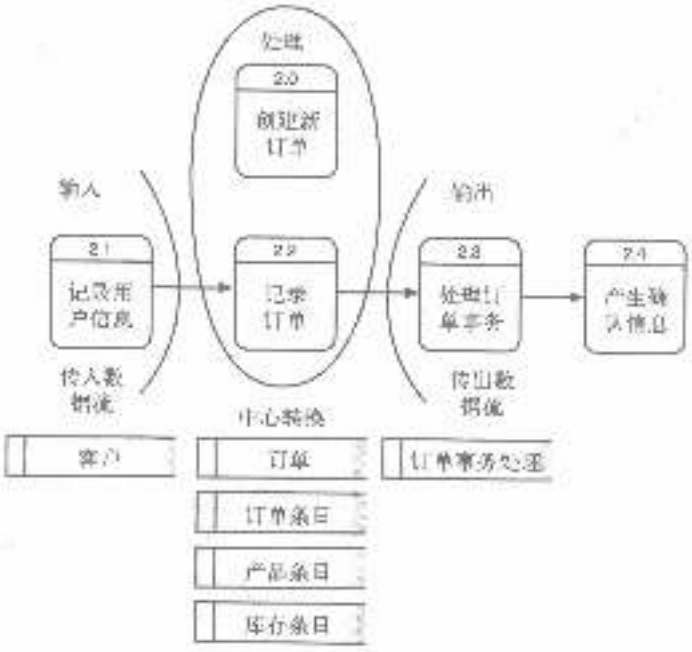


图10-13 创建新订单DFD的重组图示

（4）根据重画的数据流独立结构图的第一个草图，其中包括调用层次和必需的数据耦合。图10-14给出了一个这样的例子。

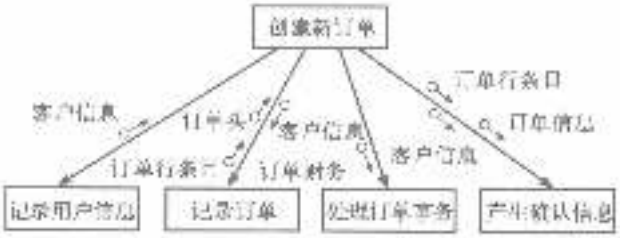


图10-14　结构图的第一个草图

（5）必要时，增加其他模块，以实现通过用户界面获得输入数据、读写数据存储、输出数据或报表的功能。通常，增加的这些模块是一些低层模块或实用模块。在读写数据存储的数据流的基础上增加一些适当的数据耦合。

（6）使用结构化英语和决策表，在其基础上加入其他所需中间模块间关系，如循环和决策标号。

（7）根据在下一节讨论的质量控制管理概念对结构图做最后的改进。

如图10-14所示，通过第4步，结构图的组织结构非常清楚地反映了产生它的数据流图。在第5步中，我们又增加了一些模块来扩展结构图的第一个草图，以便提供读写数据的模块。通常，这些模块在数据流图上没有对应的过程，所以在这一点上较少依赖数据流图信息，而较多地依赖良好的设计需求。图10-15是下一步（第5步）的结构图。

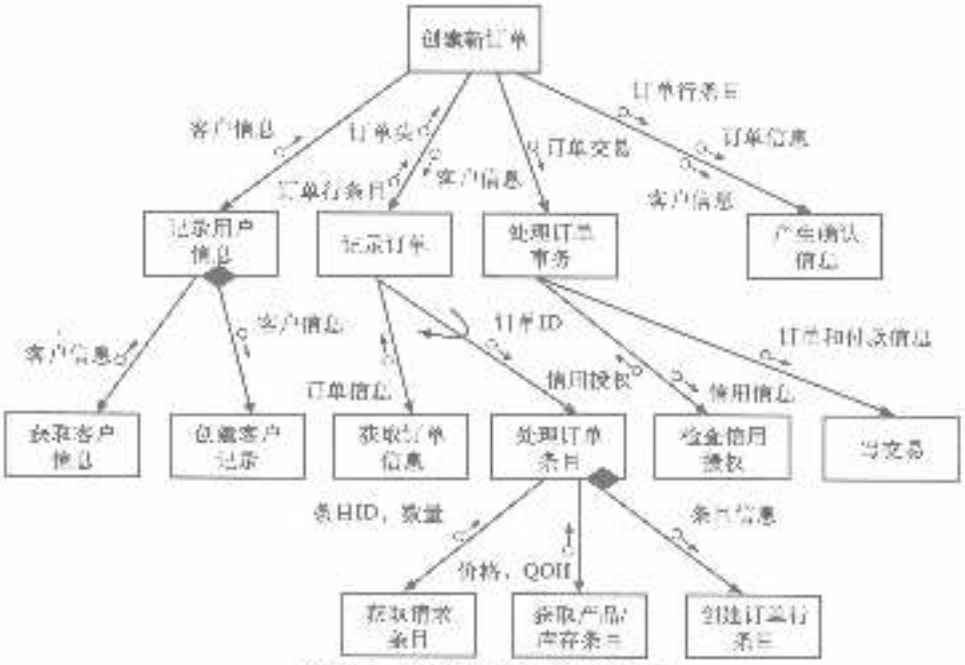


图10-15　创建新订单程序的结构图

比较图10-14和图10-15，图10-14表明了全部输入信息都来自最左边的模块——获取客户信息模块。从图10-15中可以看到，信息的存取分布到结构图的其他分支上，客户信息从最左边的分支处取得，但是关于订单的其他顾客信息从图中的第二分支处取得。虽然这种组织结

构不一定与数据流图完全符合，但它是一个更具逻辑性的结构图。增加这些存取数据的模块的确是一个设计过程，因为这是基于系统设计原则的对新部件的创建过程。

除了顾客输入数据的接口模块外，图10-15中还有其他的数据存取模块，用以取得产品和库存信息。这种获得数据的类型对应于数据流图中数据存储与处理间的数据流。在设计阶段，我们必须清楚地识别出对数据存储进行读写的模块。取得产品/库存项目模块也要加到结构图中，以使于读取产品信息。随着其他模块被添加到结构图中，数据耦合定义得更精确，它反映了更详细的设计结构。

在图10-15中，我们额外增加了循环调用和可选调用的符号。图10-15中，黑色菱形方块表示产生顾客记录调用的模块是可选的，实际上只有是新顾客时模块才被调用。结构图的一般形式为左边为输入，右边为输出。创建订单行条目的黑色菱形方块表明这样一种情况，即某种货物没有库存了，所以创建订单行条目是有条件的。

高层主模块，产生新订单模块以及它的模块对可以加入到图10-10所示的事务结构图中。图10-16表示的是一个处理过程。它包括用事务分析建立的顶层结构图和用变换分析建立的低层子树结构图。

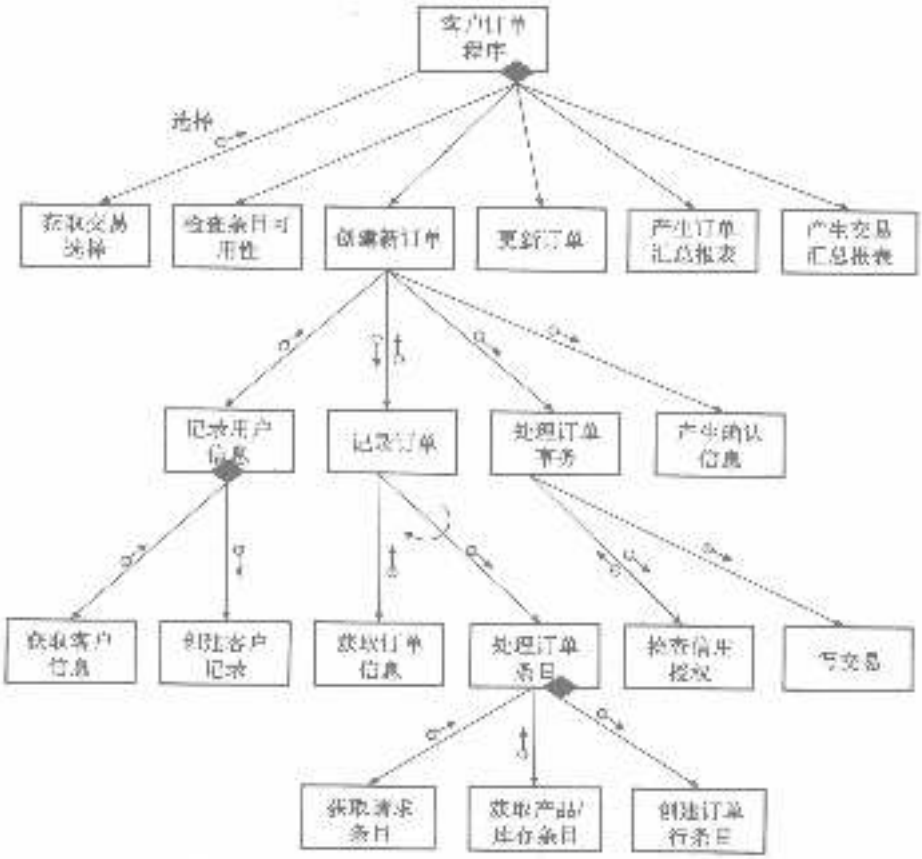


图10-16 合并后的结构图（没有给出数据耦合标志）

10.4.2 评价结构图的质量

从数据流图开发结构图的过程有可能是很复杂的。然而，我们可以采用一些规则和方法来测试最终结构图的质量。模块耦合和模块内聚是检测质量的两个标准。一般来说，我们期

要设计出高度内聚和松散耦合的模块。

模块耦合：模块与其他模块的相关程度，较好的耦合是数据耦合。

模块内聚：模块内部的凝聚程度。

耦合是指程序中一个模块与其他模块间的相关程度。我们的目标是使模块尽可能相互独立。独立的模块可以在任何环境下执行，它有一个严格定义的接口，包括一些预先定义好的数据字段，模块通过这些定义好的数据字段返回结果。模块不需要知道有哪个模块会调用它，实际上只要其他模块与这个模块的输入输出数据结构相符合，它便可由任何其他模块调用。最好的耦合是简单的数据耦合，换句话说，当调用模块时，仅传递一个特定的数据项集。然后执行这个模块并返回输出数据项。这类模块可以在需要执行该功能的其他任何结构图中重复使用。

内聚指在一个完成预定任务的模块中的所有代码的凝聚程度。具有高度内聚的模块只执行一个单一功能。模块中的所有指令都是这个功能的一部分，都是为这个功能服务的。低内聚的模块也可以完成多个、松散关系的功能。

值得注意的是，耦合及传递的特定数据的数目可以很好地表示模块的内聚程度。执行一个单一任务的模块往往是低耦合的，因为所有的内部代码使用相同的数据项。低内聚的模块往往有高耦合，因为相互有松散关系的任务经常对不同的数据项进行操作。因而，低内聚的模块经常由上层模块传递一些相互联系不大的数据项。

实践指导

耦合和内聚也是面向对象设计中的两个重要设计目标，即尽可能使对象间低耦合且每个对象内高内聚。■

在低层模块结构图中传递标记（flag）也是低内聚的一种表现，通常使用传递给模块的标记来选择将要执行该模块的哪部分代码。图10-17a便是一个低内聚的例子。项目组可以通过将模块分成几个小模块来提高内聚程度，如图10-17b所示。高层模块的代码通过标记来决定调用哪个子程序模块。

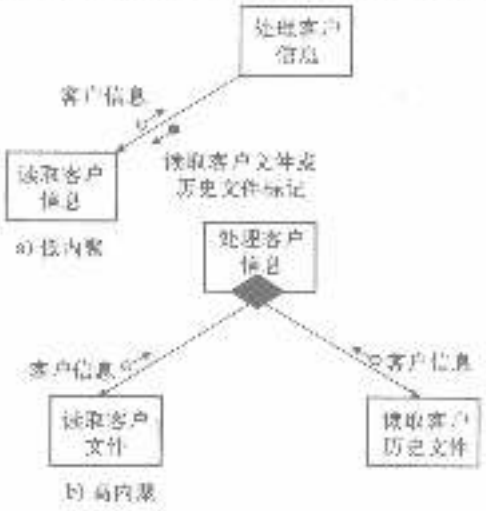


图10-17　模块内聚示例

10.5　模块算法设计：伪码

前面的两种模型——系统流程图和结构图，提供了整个系统的结构和每个程序的内部结构。设计的下一个需求是描述每个模块的内部逻辑。一般有三种描述方法：流程图、结构化英语和伪码。这三种方法在描述算法的逻辑时是相等的。流程图是使用方框、直线描述程序逻辑的可视化方法。在计算机发展早期，基本上只使用流程图来进行描述。然而，在今天，各种版本的伪码和结构化英语正逐渐取代流程图。在第6章中，我们已经学过了结构化英语。伪码是结构化英语的一种变体，更接近于程序设计语言。通常，我们用与目标语言相近的语句书写伪码。如果我们用COBOL写程序，就用类COBOL语法来写伪码；如果我们用Visual BASIC或C写程序，就使用类似于这些语言的语法书写伪码。

图10-18是关于工资系统逻辑的一个简单示例，其中包括产生工资模块、计算总量模块、计算工资模块以及计算税收模块的伪码语句。图中包含了结构化编程中的三种控制语句：顺

序贯顺序执行语句、条件选择即if-then-else语句，循环即do-until/do-while语句。

```
Payroll program
DoUntil No more time cards
     Call Enter time cards
     Call Calculate amounts
     Call Output payroll
End Until

Calculate amounts
     Call Get employee pay rates
     Call Calculate pay amounts

Calculate pay amounts

Call Calculate base amount
If (HoursWorked > 40) Then
     Call Calculate overtime amount
End If
Call Calculate taxes
If (SavingsDeduction=yes) or (MedicalDeduction=yes) or (UnitedWay=yes) Then
     Call Calculate other deductions
End If

Calculate taxes

Get Tax Rates based on Number Dependents, Payrate
Calculate Income Tax = PeriodPayAmount * IncomeTaxRate
If YTD Pay < FICA MaximumAmount Then
     Calculate EmpFICA = PeriodBasePay * FICAEmployeeRate
     Calculate CorpFICA = PeriodBasePay * FICACorpRate
End If
If StateTax Required Then
     Get StateTaxRate based on State, NumberDependents, Payrate
     CalculateStateTax = PeriodPayAmount * StateTaxRate
End If
If StateTaxRequired Then
     Get StateTaxRate based on State, NumberDependents, Payrate
     Calculate StateTax = PeriodPayAmount * StateTaxRate
End If
If OvertimePay > 0 Then
     Calculate OvertimeIncomeTax = PeriodOvertimePay * IncomeTaxRate
     Add to OvertimeIncomeTax to IncomeTax
     If YTDPay < FICAMaximum Amount Then
          Calculate EmpOvertimeFICA = PeriodOvertimePay * FICAEmployeeRate
          Calculate CorpOvertimeFICA = PeriodOvertimePay * FICACorpRate
     End If
     If StateTaxRequired Then
          Calculate StateOvertimeTax = PeriodOvertimePay * StateTaxRate
     End If
End If
```

图10-18 计算工资总额模块结构的伪码

10.6 结构化应用程序设计与其他设计任务的集成

在以上几节中，你已经学会如何利用数据流图中的信息建立结构图，其重点是抓住事务处理间的关联关系。用事务分析或变换分析建立的结构图是恰当的但可能不完全，正像Barbara Halifax在RMO备忘录里讨论的那样，通常在结构图完整建立前必须进行调整，加强用户界面、数据库、网络设计的集成，从而才能使之更全面，建立完整的结构图，因为我们会在后面的章节中详细讨论用户接口和数据库设计，在本章和下一章中仅简要讨论一下三层设计中需要的类型变换。

2007年5月19日

To：John MacMurty

From：Barbara Halifax

RE：客户支持系统的软件设计要注意问题

John，我们将在几天后完成所有的详细设计，这比预期的要所延后，但我想我们可以在实现阶段赶上。开发人员对话和其他用户界面的时间比预期的要长。这主要是因为那些结果和我们的最初期望并不十分匹配，我们需要对软件设计做一些修改，预计这些修改可以在周末前完成。

此外，在一次与Ann的预排中，我们发现原来计划将软件模块和分布到多层和多计算机上存在一些不一致性和潜在的网络瓶颈问题。我们已经适当地修改了应用程序结构并找出了将来网络方面，尤其是防火墙和路由器配置方面可能需要修改方案。Ann已经将那些更新安排在6月中旬进行，刚好在第一轮软件测试之前。

如有问题，请联系我们。

BH

cc：Ann Hamilton

用户界面应包括一系列输入、输出表单和报表。通常交互式用户界面是基于人机对话的，包括一系列的输入/输出表单。每个表单都必须显示并且用到的数据都包含在结构图的某个模块中。在建立这些表单的同时，我们应从以下三个方面评估结构图：

- 是否需要增加用户界面模块；
- 接口模块的伪码是否需要修改；
- 是否需要加入其他的数据耦合来传送数据。

在前面的章节中，我们已经知道实体-联系图（ERD）必须与数据流图中的数据存储相一致。数据存储与数据库表不需要一一对应，但数据存储中的信息必须能在数据库中找到。数据库的每一个表、每一个字段也必须在数据存储中表示出来。在设计阶段，项目组要使用同一种分析方法并对结构图进行适当的调整。

同样，对数据库的评价也分为三个方面：模块、伪码和数据耦合。如果目前正在使用一个数据库管理系统，则通常需要提供一个通用接口。设计者既可以通过调用数据库接口模块，也可以通过伪码中的嵌入式SQL（结构化查询语言）语句读取数据库。

最后，我们要检查结构图和系统流程图与现存的或计划的网络结构的一致性。因为结构设计通常和详细的软件设计同时进行，一般都是在合理的网络结构的假设上来开发系统流程图和结构图。然而，由于之后可能需要不断加入新的细节，系统流程图和结构图需要不断调整，这时就会产生新的问题。因此，详细设计的最后一步，即重新评估它与网络结构的一致性非常重要，特别是所需协议、容量和安全性等方面。

文本的线性特征要求必须以固定的顺序描述设计活动的具体细节，然而在实际的开发项目时，设计活动的顺序变化很大。有一些工程把详细设计，如软件本身、用户接口、系统结构和数据库设计，分配给不同的小组同时开发。如果在SDLC中使用迭代方法，那么每一次迭代都要完成详细设计的工作。另一些工程可能由于缺少个体的或特定的工程特性而更符合线性特征，对于较为符合线性顺序的工程，后期设计工作完成后要重新评估所有类型的早期详细设计决策。

10.7 三层设计

第9章中描述了三层设计及其对应用软件的划分，即视图、业务逻辑和数据访问层。结构图和系统流程图至少比三层结构并发方法早出现10年，但它们仍然可以用来描述基于三层结构的设计决策和软件结构。

图10-19给出了RMO客户订单程序的系统流程图。这个流程图按第9章中介绍的可视、业务逻辑和数据访问三层结构划分处理过程。层与层之间用流程图上定义好的协议进行通信，使得各层在需要时可以在不同的机器上创建。如第9章所述，协议的选择（如HTTPS、SQL）和开发环境的选择（如IE、Java组件和Oracle DBMS）在设计初没早期的结构设计中已经决定。用并发环境的详细说明注释系统流程图可以进行决策记录归档，并与项目的其他参与人员就重要的约束进行讨论。

然而，在软件层执行的地方系统流程图并不是必需的。例如，图10-19中，可视层运行在用户工作站上，但它并没有指出业务逻辑以及数据访问层的具体运行位置。它们可以运行在用户工作站上（一般不这样安排），也可以运行在一个独立的服务器上，或者各自运行在独立的服务器或服务器群上。除非加入了专门的位置信息，否则系统流程图应该按程序或模块组织，而不是计算机系统给出处理功能的分布情况。

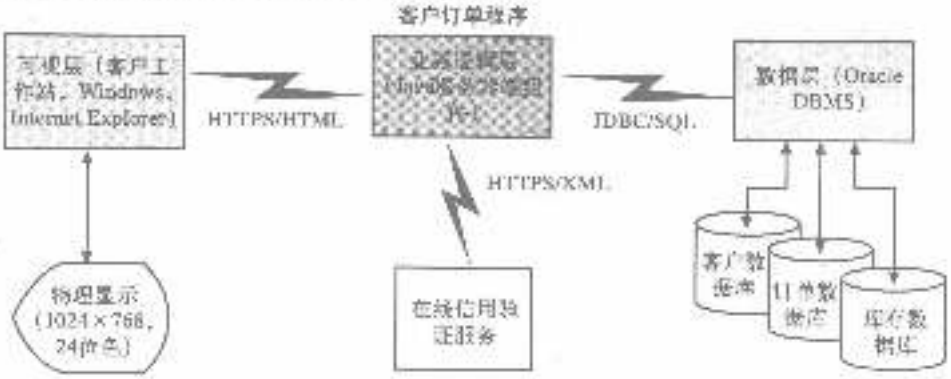


图10-19 客户订单程序三层结构的系统流程图

用三层结构的结构图来描述软件结构与本章前面介绍的用于并发环境和开发工具设计的例子大不相同。比较图10-15和图10-16所示的结构图。结构图是在应用程序功能达到全面的阶段开发的。换句话说，它们包含了处理所有应用程序任务的可操作模块，包括输入/输出（可视层），业务逻辑以及与存储在文件或数据库中的数据进行交互（数据访问层）。图10-16描述了一个可以用传统程序设计语言（如C或COBOL）来实现，并几在集中式硬件体系结构中展开的程序。

图10-20所示的是产生新订单的一个三层结构的结构图，这个结构图是基于面向窗体对话框的，这一内容将在第13章中讨论。黄色表示可视层模块，红色表示业务逻辑层模块，绿色表示数据层模块。图中给出了所有可视层模块，但只给出了用户窗体调用的业务逻辑层模块。其他可视层模块需要的业务逻辑层模块与此类似。数据层包含一个DBMS，因此业务逻辑层模块包含用来产生适当的数据库访问命令和处理响应的代码。

图10-21是RMO中用来查询、添加或更新结构图中所描述的客户数据的用户窗体。图10-22显示了怎样用VB程序设计语言完成用户窗体相关的代码来表示可视层。当用户单击Search（搜索）按钮时，执行事件处理程序btnSearch_Click()。输入窗体的用户ID号被传给表示业务逻辑层代码模块的函数。可视层不包含用户搜索的具体细节。

图10-23所示是业务逻辑层执行搜索现有客户功能的一个模板。它由btnSearch_Click()事件触发，调用RetrieveExistingCustomer()实现，它为数据库检索语句指定了代码插入点。注意这一功能只负责检索已有客户的数据，而不处理输入的新客户数据，也不处理新客户或已有客户数据的数据库更新。结构图中，客户窗体下的其他函数或过程都应放在业务逻辑层中实现。本例说明了可视层和业务逻辑层之间程序代码的一个清晰划分，尽管它们可能会在同一个机器上运行的同一段程序中执行。

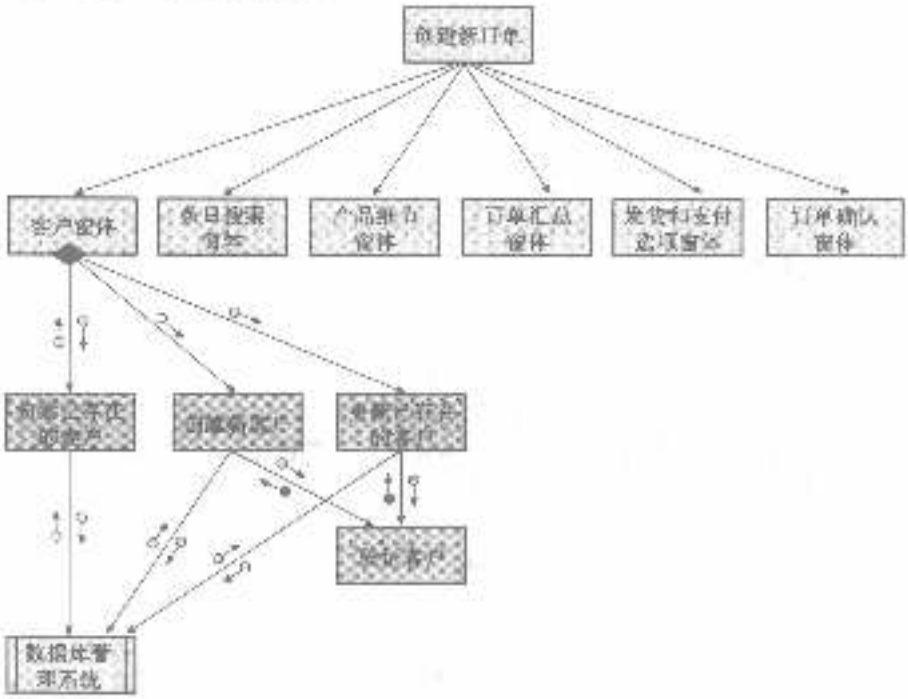


图10-20　创建新订单活动一层结构的结构图

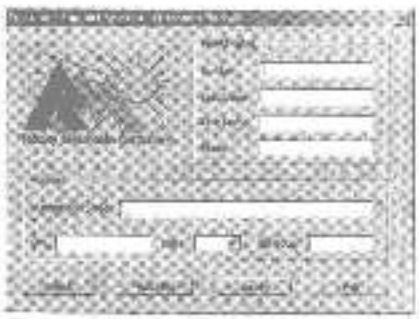


图10-21　RMO中布投、添加或更新客户数据的简单窗体

当把各层分布在不同的计算机系统上时，程序比集中式结构更专门化、多样化。分布式三层结构使用多个程序，而不是把用户界面、业务逻辑和数据访问模块整合进一个单一的程序和结构图中。有些层，如可视层，甚至不再是传统意义上的程序。例如，使用基于Web的HTTP用户界面时就是这样。用传统程序设计语言或面向对象程序设计语言编写的各个层通常是独立的程序。例如，在一个基于Web的系统中，结构图（见图10-20）最上面两层模块将被散成一系列网页来执行。第三层模块检索已存在的客户、创建新客户、更新已存在的客户可能是独立的程序，存储在应用程序服务器或Web服务器上，通过Web页面调用执行。因此，图10-20所示的结构图将被分解成几个更小的结构图。

注意，在不同的计算机系统上执行的各个独立程序不能通过系统流程图中代表数据耦合的函数或过程参数进行通信。不同层中的模块通过明确定义的协议在实时链接上进行通信。如图10-19所示，通信实体也在系统流程图中有所描述。层与层之间传递信息的准确格式和内

容必须在模块伪码或其他地方详细说明。

```
Public Class CustomerForm
    Inherits System.Windows.Forms.Form

 Windows Form Designer generated code

    Private Sub btnSearch_Click(ByVal sender As System.Object, _
    ByVal e As System.EventArgs) Handles btnSearch.Click

        'When btnSearch is clicked, call the code module
        'in the Business Logic Layer using customerID
        'and get back array of customer details

        Dim custID As String
        Dim CustomerDetails(6) As String
        custID = txtCustomerID.Text
        Try
            CustomerDetails = OrderEntryModule.RetrieveExistingCustomer(custID)
            txtLastName.Text = CustomerDetails(0)
            txtFirstName.Text = CustomerDetails(1)
            txtStreetAddress.Text = CustomerDetails(2)
            txtCity.Text = CustomerDetails(3)
            txtState.Text = CustomerDetails(4)
            txtZipCode.Text = CustomerDetails(5)
            txtPhone.Text = CustomerDetails(6)
        Catch
            MessageBox.Show("Customer not found")
        End Try

    End Sub
```

表10-22 图10-21所示窗体的VB代码

```
Public Class OrderEntryModule

    'This module contains functions and procedures that are
    'called from event procedures on forms

    'It represents part of the Business Logic Layer for shared
    'Order Entry processing code.

    Public Shared Function RetrieveExistingCustomer(ByVal anID As String) As String()

        'This function queries the database based on the
        'customer ID to search for an existing customer
        'and then it returns the customer details

        Dim customerDetails(6) As String

        'Insert needed code here

        Return customerDetails

    End Function

    'Continue with other order entry functions and procedures
    '...

End Class
```

图10-23 搜索已存在客户的VB代码模板

我们现在已经讨论过了用传统模型和方法进行软件的结构设计和详细设计。在下一章中，我们将讨论如何用面向对象的工具和技术完成这些设计任务。之后，我们将注意力集中到剩下的设计活动——数据库、用户界面和系统接口的设计上。

小结

采用传统的结构化方法设计系统，数据流图是主要输入。首先，加入系统边界来加强数据流图。设计人员画出系统边界草图，用以显示一个完整的系统。他也在DFD片断上画出边界以显示更低层的程序边界。

设计人员用一个或多个结构图描述各DFD边界内的处理过程。采用事务分析、变换分析或两者结合开发结构图。事务分析适用于处理多输入或多类型事务系统的上层结构图。变换分析适用于设计一个事务从输入到输出的变换程序。结构图还可以基于三层结构。这种结构可以通过层清楚地识别各模块，而且如果各层在多计算机系统中执行，还可以把结构图分解成多个小的结构图。

结构化设计也可以包含系统流程图和模块伪码。系统流程图表示程序、文件和手工处理步骤间的数据流动，从而提供整个系统的总体情况。系统流程图还可以描述多层系统中层与层之间的交互。模块伪码用来描述一个结构图模块的内部逻辑。

关键术语

afferent data flow	传入数据流
central transform	中心变换
computer program	计算机程序
data couples	数据耦合
efferent data flow	传出数据流
module	模块
module cohesion	模块内聚
module coupling	模块耦合
program call	程序调用
pseudocode	伪码
structure chart	结构图
system flowchart	系统流程图
transaction analysis	事务分析
transform analysis	变换分析

复习题

1. 解释模块和程序的区别与联系。
2. 引入自动化系统边界的目的是什么？如何开发自动化系统边界？
3. 系统流程图的作用是什么？
4. 在系统流程图中会使用哪些符号？
5. 引入结构图的目的是什么？
6. 结构图使用哪些符号？

7. 解释事务分析。
8. 解释变换分析。什么是中心变换?
9. 传入数据流和传出数据流有何区别?
10. 解释模块耦合和模块内聚。为什么这些概念很重要?
11. 描述三层结构图与在一个单独的计算机系统上执行的所有程序的结构图有何不同?

思考题

1. 给出如图10-24所示的数据流图。回答如下问题:
   (a) 画出系统边界;
   (b) 将数据流图划分成程序组件,例如,实时的、每月的、每日的、周期的等;
   (c) 根据程序组件的划分画出系统流程图。

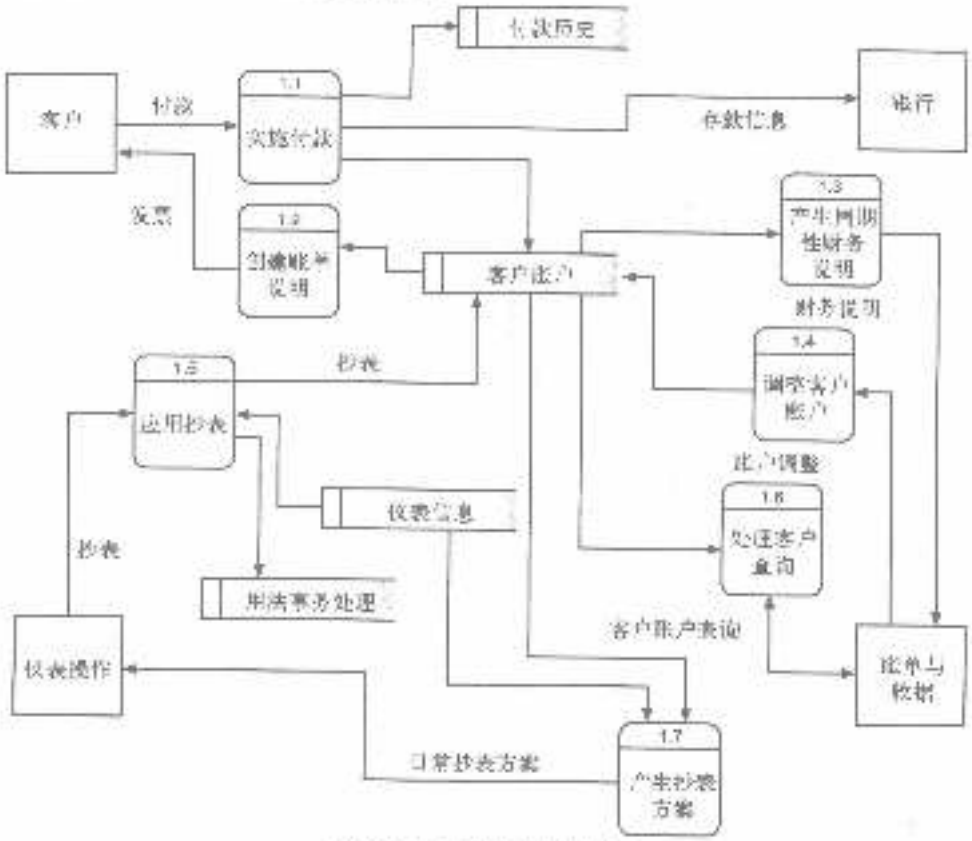


图10-24 电子公司客户账单

2. 按如图10-25所示的数据流图,用事务分析开发一个结构图。
3. 按如图10-26所示的数据流图,用变换分析开发一个结构图。
4. 将问题2、3的结构图整合成一个结构图。
5. 按如图10-27所示的数据流图,用变换分析开发一个结构图。
6. 完成图10-20所示可视层模块对应的业务逻辑层模块的开发。

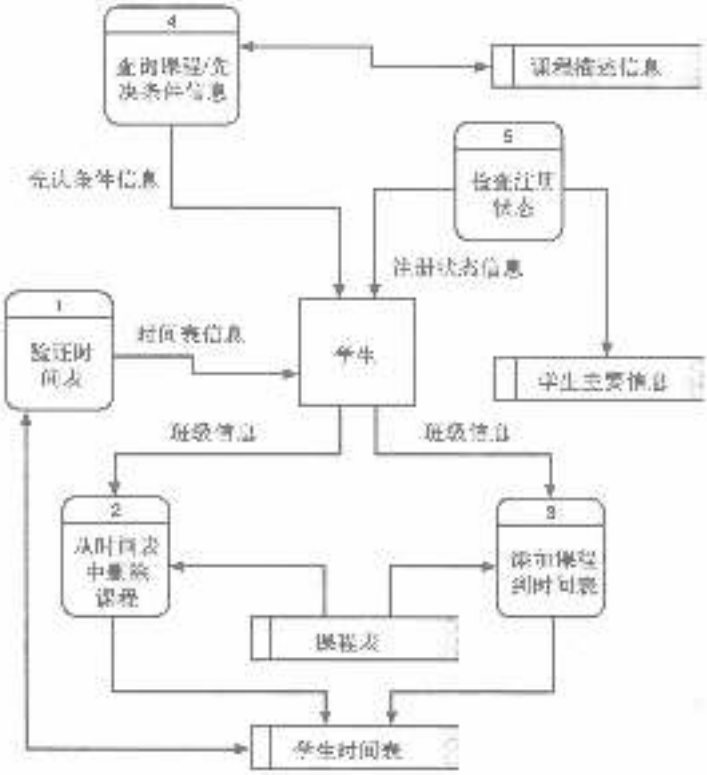


图10-25　学生注册程序

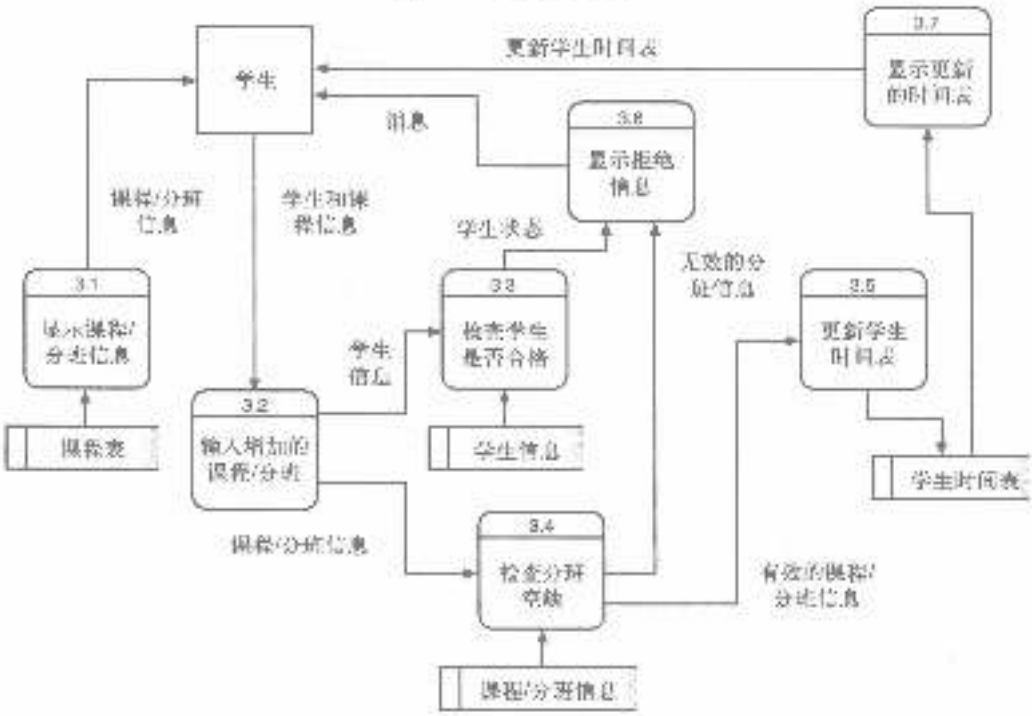


图10-26　添加课程到时间表的分解图

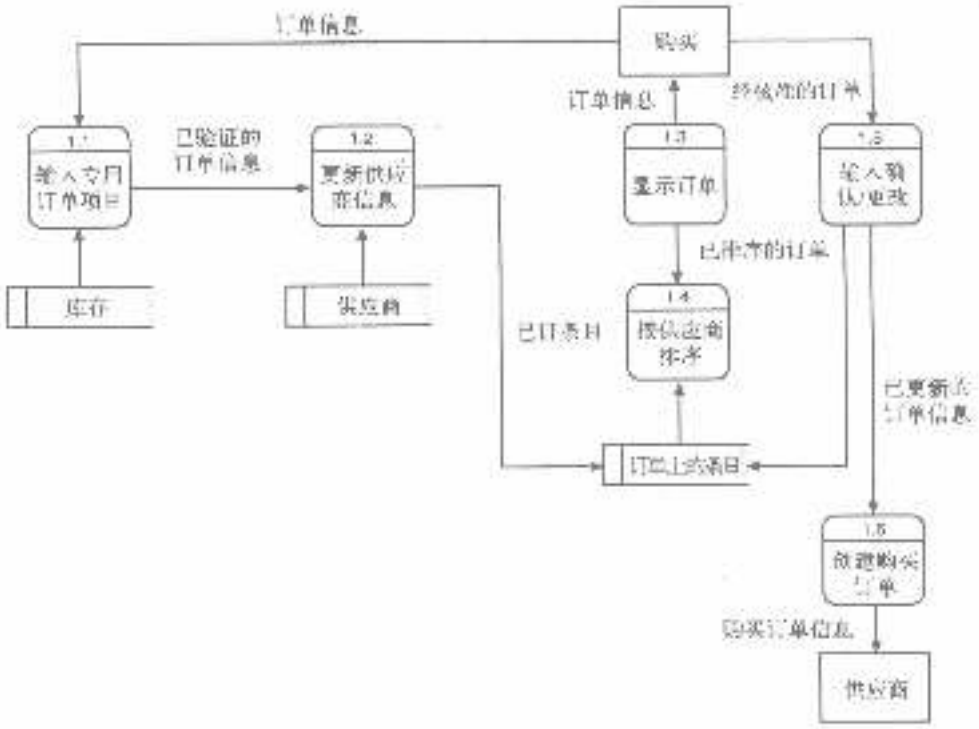


图10.27 专用订单购买流程

实验练习

1. 讨论传统结构化设计分层的本质。什么类型的系统和结构本身更倾向于一种分层结构？
2. 找一个用COBOL或BASIC编写的业务系统的例子，要求该系统具有层次结构并可能是用传统方法开发的。在Internet上查找COBOL、Microfocus COBOL或Visual BASIC等关键字。
3. 找一个使用传统结构化技术进行开发的本地公司，与IS部门人员进行一次面谈，尽可能多地收集关于该公司的信息。仔细研究并讨论这个公司的技术和SDLC方法。

实例研究

房地产多编目服务系统

参考第5章描述的房地产多编目服务系统的实例和你在第6章的实例研究中开发的DFD，为这个系统开发一个结构图。要求依照本章说明的步骤，且包含用于访问数据的所有模块。

对落基山运动用品商店（RMO）实例的再思考



回顾第8章和第9章中描述的关于RMO客户支持系统的配置环境和设计的决定以及本章中相关的传统设计模型。针对这个系统而言，用传统方法和模型进行软件设计与用面向对象的方法和模型进行软件设计各有什么优缺点？

关注Reliable Pharmaceutical Service



根据第5、6章中Reliable药品服务系统的描述以及你为第6章设计的数据流图，为系统开发一个系统流程图和一个结构图。要求按三层结构设计和开发系统。

参考资料

Tom DeMarco, *Structured Analysis and System Specification*. Yourdon Press, 1979.

Meilir Page-Jones. *The Practical Guide to Structured systems Design*, (2nd ed.). Yourdon Press, 1988.

Edward Yourdon, *Modern Structured Analysis*. Yourdon Press, 1989.

Edward Yourdon and Larry L.Constantine. *Structured Design: Fundamentals of a Discipline of Computer Program and Systems Design*. Prentice Hall, 1979.

第11章　面向对象设计方法：用例实现

学习目标

阅读本章后，你应具备如下能力：

- 解释面向对象设计的目的和目标
- 开发设计类图
- 根据对象职责准则和用例控制器开发交互图
- 开发作为系统设计核心过程的顺序图
- 开发作为系统设计一部分的协作图
- 用包图记录结构设计

本章要点

- 面向对象设计——程序分析和设计的桥梁
- 设计类和设计类图
- 交互图——实现用例和定义方法
- 实现多层设计
- 用协作图设计
- 更新设计类图
- 包图——将主要部分结构化
- 三层设计的实现问题

NEW CAPITAL BANK

尽管在项目开始的时候有一些问题，但是现在看来一切都在掌握之中。Bill Santora 是 New Capital Bank 的项目经理，负责开发一个集成客户账目系统，他刚刚和审查委员会的委员们完成了对新系统初步设计的技术审查。初步设计关注6个核心用例，它们作为系统最基础的部分，在第一次迭代中完成。

New Capital Bank 使用面向对象语言已经有一段时间，但开始使用面向对象分析和设计的技术却比较晚。Bill Santora 曾用面向对象的技术开发过一些系统，比如早期用统一过程（UP）和统一建模语言（UML）开发的导航系统。但是，这次的开发项目是他第一次遇到的完全面向对象的大型项目。

Bill Santora 把材料给他的上司Mary Garcia。Mary Garcia说："你的技术评审做得非常好。委员们只提出了很少的需要修改的部分。虽然我不是太明白面向对象技术，但是我还是看懂了你给我的材料和主要的功能。我很难相信你能用两个星期的时间完成这6个部分。"

Bill Santora笑着说："等等，只完成这6个主要功能的编码和运行并不意味着项目的结束。这个项目还需要一年的时间来完成。"

"但是两个月之后我们可以做出点东西来就很好。不单只有我对项目有信心，用户也愿意看到事情有所进展。"

"没错，别忘了我开始提出用UP来做这个项目的时候是多么的艰难！因为UP是一种迭代的方法，所以为以后的迭代制订计划是比较困难的。我花了很长的时间让大家相信这个项目

的风险不是很大。因为每次迭代只有6个星期的时间，所以在开始阶段就要展示一些东西。你不知道，设计通过评审之后，我的压力减轻了很多，大家做了很多工作确保设计的可靠性，我们觉得很有信心。能够得到认可是多么好的一件事。在接下来的两个星期里，我们还有一些关于新系统的基本工作要做。”

“采用渐增式的方法很有意义。我尤其喜欢你对每个用例设计做的详细的顺序图。支持每个用例的三层设计，你做得非常好。我能明白每个用例是怎样完成的。但是我还是不太明白先进的面向对象技术。我想当你证明用同样的基本设计既可以为我们内部银行出纳员，又可以为Web端的用户设计系统时，大家会前手为你叫好的。祝你成功！”

Bill Santora回应了Mary Garcia的祝贺，说：“类图的设计怎么样？您不觉得类图使得类和方法看上去更加明白吗？我们在组内讨论的时候都是用它们进行交流的，确实能帮助程序员写出好的、可靠的代码。”

“顺便问一下，你们实际和用户进行复审了吗？”Mary Garcia问。

“没有，我们在开发用例和创建用例描述的时候和用户联系比较紧密，我们还和用户共同开发了所有这些用例的原型，所以，我们不需要再向用户解释设计模型的细节，而可以对这些用例进行编码了。毕竟，在几个星期过后要给他们展示一些东西。然后，我们和用户有下一轮的会议，让他们给我们的工作提意见，我们可以开始下一阶段的迭代。我们需要他们的意见来开发下一阶段的用例描述。”

“我很期待看见第一阶段的成果。在项目的后续开发过程中可以测试这些核心功能是很有意义的，让我再次祝贺你。”Mary Garcia一边说，一边和Bill去吃午饭。

概述

回忆一下第2部分，面向对象分析是由两个目标组成的，即发现和理解。发现是由发现事实的活动组成的，比如和系统用户面谈；而理解是提取从被采访用户那里得到的信息并且构造相互关联的广泛的模型的过程。在第5章中，学习了如何识别问题的主要类和用例，它们提供了理解系统需求的基本信息。在第7章中，学习了如何通过详细描述、活动图和系统顺序图（SSD）扩展每一个用例和通过状态图定义对象行为完成面向对象分析需求模型。建立模型是理解用户需求的必不可少的部分。同时也影响着最终的系统。然而，要记住，建立分析模型的目的不是描述新的系统，而是更精确地理解系统需求。

第9章介绍了结构和详细设计方面的概念。你现在已经知道了系统设计要关注很多系统方面的问题，如网络设计、应用设计、数据库设计、用户界面设计和系统界面设计。本章和下一章的主要问题是如何开发面向对象设计模型，程序员要用这些模型来为系统编码。两个必须开发的最重要的模型是设计类图和交互图（顺序图和协作图）。你将学到如何为三层设计中的每一层开发设计类图，这三层包括：域层、可视层和数据访问层。设计类图扩充了分析阶段开发的域模型。交互图也扩展了分析阶段开发的系统顺序图。我们还将讨论如何将类关联到包图中，以说明其中的关系和依赖。最后，我们讨论一些优秀的设计准则，以及应用这些准则的方法。

11.1 面向对象设计——程序分析和设计的桥梁

如果我们把开发新的软件系统比做盖房子的话，那么分析就像是建筑师最开始的设计草图和方案图。这些草图描绘了房屋的主人想让卧室什么样，地板什么样，布局什么样，位置什么样等。然而，仅有这些草图，承包人是不知道该如何建造房子的，他需要更详细的信息。所以，建筑师需要利用草图制作出更详细的规划，称之为蓝图。蓝图详细规定了墙、地板和天花板应该是什么样，对电线和管道设备的说明，甚至该用什么样的材料。这些详细蓝图与

软件开发者构建的设计模型是类似的。

可以进一步类推，草图设计好之后，屋主可以向建筑师提出他们的要求，但是屋主一般不参加细节设计。屋主评估并纠正设计细节，但是墙内线路和管道的设计是由专家或建筑师指导承包人完成的。类似地，在软件设计中，细节设计规范主要是由软件设计专家完成的。用户只是偶尔参与——主要是为了对设计进行纠正。

那么什么是面向对象设计呢？它是一个建立一系列面向对象设计模型的过程，程序员利用这些模型对系统进行编码和测试。系统设计是用户需求和新系统程序设计之间必不可少的桥梁。建筑者如果没有蓝图，是怎样也不会建造出比狗窝或小屋更大的东西的。同样，系统开发者如果没有设计模型是不会开发出一个大型系统的。有的时候，学生做课程设计的时候需要开发个人的网页或小型的系统，所以，他们认为设计模型不重要。但是要记住，对于建造狗窝来说，蓝图可能不重要，但是对于复杂的东西，比如说房屋，蓝图是必不可少的。本章讲的应用软件的设计，只是面向对象设计的一部分。正如你在第9章中学到的一样，用户界面设计、联络设计、控制和安全设计和数据库设计同样也需要详细设计并开发出设计模型。

11.1.1 面向对象程序设计概述

要理解面向对象程序设计，我们首先要理解一个面向对象程序是如何工作的。面向对象程序由一系列协同完成某一任务的对象组成。每个程序对象之间有程序逻辑和一些必要的属性，这些逻辑和属性封装在一个单元中。对象之间通过互相传递消息来协同工作，它们共同工作来完成主程序的功能。

图11-1描述了一个面向对象程序是如何工作的。这个程序包括一个输入窗口对象，该窗口用来输入学生ID以及其他信息。当学生输入ID后，窗口对象会发出消息（消息2）给学生类，程序会产生一个新的学生对象（实例）。同时也转到数据库中取得学生信息，并把这些信息组成一个对象（消息3）。数据访问对象读取数据库，并将读取的值填充到学生对象中。该步完成后，新的学生对象会给窗口对象返回一个消息并显示在屏幕上，职员此时会输入个人信息的更新（消息4）。随后一些消息负责更新程序中的学生对象和数据库中的学生信息。

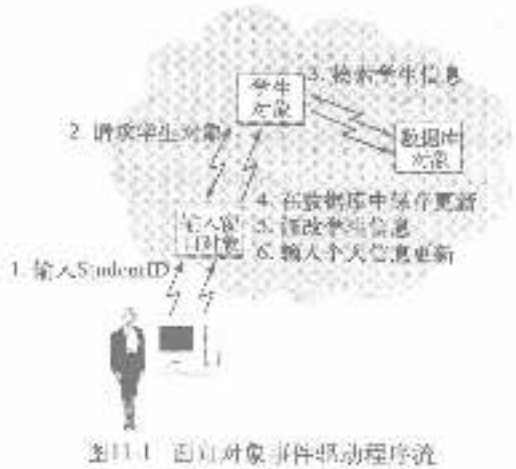


图11-1 面向对象事件驱动程序流

一个面向对象系统由一系列计算对象组成。每个对象都封装了它自身的数据和程序逻辑。分析员通过一个类来定义程序逻辑的结构和数据字段。类定义描述了一个执行对象的结构或模板。只有当程序开始执行时，对象才能存在。这个过程我们称之为类模块的实例化，或基于类定义所提供的模板生成对象实例。

实例化：根据类定义所提供的模板创建对象。

11.1.2 面向对象设计模型

面向对象设计的目标是确定组成用例的所有对象，如图11-1所示。这些对象包括用户界面对象，域类对象和数据库访问对象。此外，还需要一些额外的对象完成特定的功能，如用于完成登录验证的对象。将对象划分为几组叫做多层设计。设计的一个主要目标是确定各层的对象集，以及确定对象间的交互和消息。

顺序图和协作图是面向对象设计中最重要的模型。在第7章中曾介绍过，顺序图是交互图的一种。同样，协作图也是交互图的一种。在设计过程中，开发人员通过向系统对象添加用户界面对象、域类对象和数据库访问对象来扩展系统顺序图。换句话说，他们需要知道系统内部有些什么。在本章中，我们会花大量时间来学习如何开发详细的顺序图。图11-2是基于图11-1的一个简单的顺序图，其作用是更新学生信息。在顺序图中，与在第7章中学到的系统顺序图使用的符号是一样的。事实上，顺序图只是系统顺序图的一个扩展。在本章的后面部分，我们将详细讨论顺序图以及如何开发顺序图。注意：本章的目标是学习如何开发顺序图。



图11-2　更改学生姓名的顺序图

在本章中，你还将学到另外一种重要设计模型——设计类图。设计类图的主要目的是记录和描述构建新系统需要的类。它描述系统所有的类、类间的导航、属性和方法。设计类图是利用详细顺序图得到的最终设计的一个总结，并在编码过程中起着举足轻重的作用。图11-3所示为在分析阶段得到的域模型和设计类图。比较两图我们可以得知，设计类图的底部包含了该类的方法特征，同时为类的每个属性指定了类型。我们将在下一节中介绍图中所使用的各个符号所表示的含义。设计是通过开发顺序图从而将域模型转换成设计类模型的过程。本章，你会学习该设计过程。

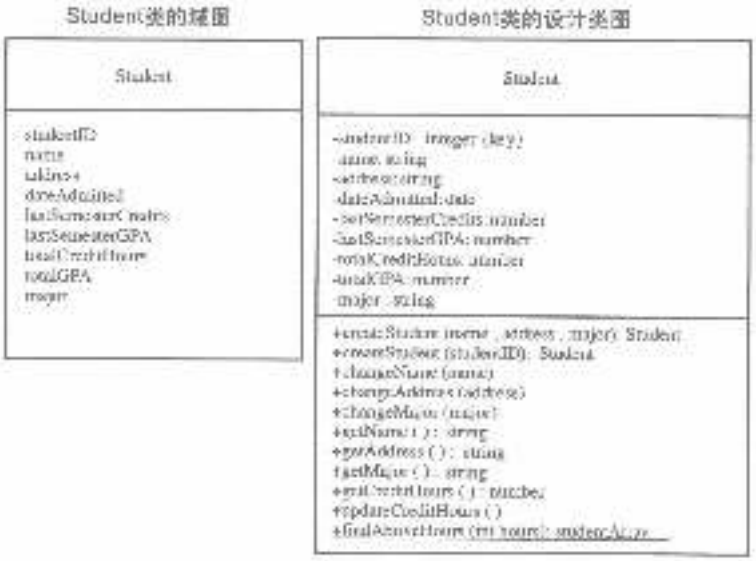


图11-3　Student类的域图和设计类图

系统的设计者要提供足够的信息，以使程序员知道如何定义那些类。在下面几节中你会看到，面向对象设计的主要部分是设计类图，交互图，对于有些类，还要设计状态图。比如，一个设计类说明有助于定义属性和方法，图11-4a所示是用Java语言实现的Student类的部分代码，而图11-4b所示是Visual Basic.NET实现的部分代码。再次回顾图11-3，我们可以从中看出Student的设计类是如何为图11-4的代码提供输入的。注意，类名、属性和方法名来源于设计类的符号。在设计类的时候，我们比较自由地缩减了第一个名字和最后一个名字来简化名字，并且把所有的地址成分组合到一起，称之为地址。如果程序员不知道是不是该把这些简化了的名字分开，那么设计者就不能进行这样的简化。类需要的其他代码可以从其他设计模型中获得，包括交互图和状态图。

```
public class student.
{
      //attributes
      private int studentID;
      private String firstName;
      private String lastName;
      private String street;
      private String city;
      private String state;
      private String zipCode;
      private Date dateAdmitted;
      private float numberCredits;
      private String lastActiveSemester;
      private float lastActiveSemesterGPA;
      private float gradePointAverage;
      private String major;

      //constructors
      public Student (String inFirstName, String inLastName, String inStreet,
            String inCity, String inState, String inZip, Date inDate)
      {
            firstName = inFirstName;
            lastName = inLastName;
            ...
      }
      public Student (int inStudentID)
      {
            //read database to get values
      }

      //get and set methods
      public String getFullName ( )
      {
            return firstName + " " + lastName;
      }
      public void setFirstName (String inFirstName)
      {
            firstName = inFirstName;
      }
      public float getGPA ( )
      {
            return gradePointAverage;
      }
      //and so on

      //processing methods
      public void updateGPA ( )
      {
            //access courses records and update lastActiveSemester and
            //to-date credits and GPA
      }
}
```

a) Java实现与Student类

图 11-4

```
Public Class Student

    'Attributes
    Private studentID As Integer
    Private firstName As String
    Private lastName As String
    Private street As String
    Private city As String
    Private state As String
    Private zipCode As String
    Private dateAdmitted As Date
    Private numberCredits As Single
    Private lastActiveSemester As String
    Private lastActivedGradeGPA As Single
    Private gradePointAverage As Single
    Private major As String

    'Constructor Methods
    Public Sub New(ByVal inFirstName As String, ByVal inLastName As String,
          ByVal inStreet As String, ByVal inCity As String, ByVal inState As String,
          ByVal inZip As String, ByVal inDate As Date)
        'firstName = inFirstName
        lastName = inLastName
        ...
    End Sub

    Public Sub New(ByVal inStudentID)
        'use database to get values
    End Sub

    'get and set accessor methods
    Public Function GetFullName() As String
        Dim info As String
        info = firstName & " " & lastName
        Return info
    End Function

    Public Property firstName()
        Get
            Return firstName
        End Get
        Set(ByVal Value)
        firstName = Value
        End Set
    End Property

    Public ReadOnly Property GPA()
        Get
            Return gradePointAverage
        End Get
    End Property

    'Processing Methods
    Public Function UpdateGPA()
        'read the database and update last semester
        'and to date credits and GPA
    End Function

End Class
```

b) VB.NET 实现的Student类

图 11-4 (续)

图11-5展示的是不同的设计模型各自所能使用的需求模型。位于左边的是在需求定义中得到的模型，其中有域模型类图、用例图、用例描述、活动图、系统顺序图和状态图。位于右边的是在设计阶段得到的模型，包括设计类图、交互图、状态图和包图。根据图中指向它们的箭头的数目可以看出，在设计过程中，交互图最为重要。

你可能已经注意到，设计类图和交互图之间的箭头是双向的。双向箭头意味着，交互图需要设计类图的一些信息，同样，设计类图也需要交互图的一些信息。稍后，我们还会讨论面向对象设计中其他模型和图，这些模型，如包图和部署图，对文档编制非常有用。

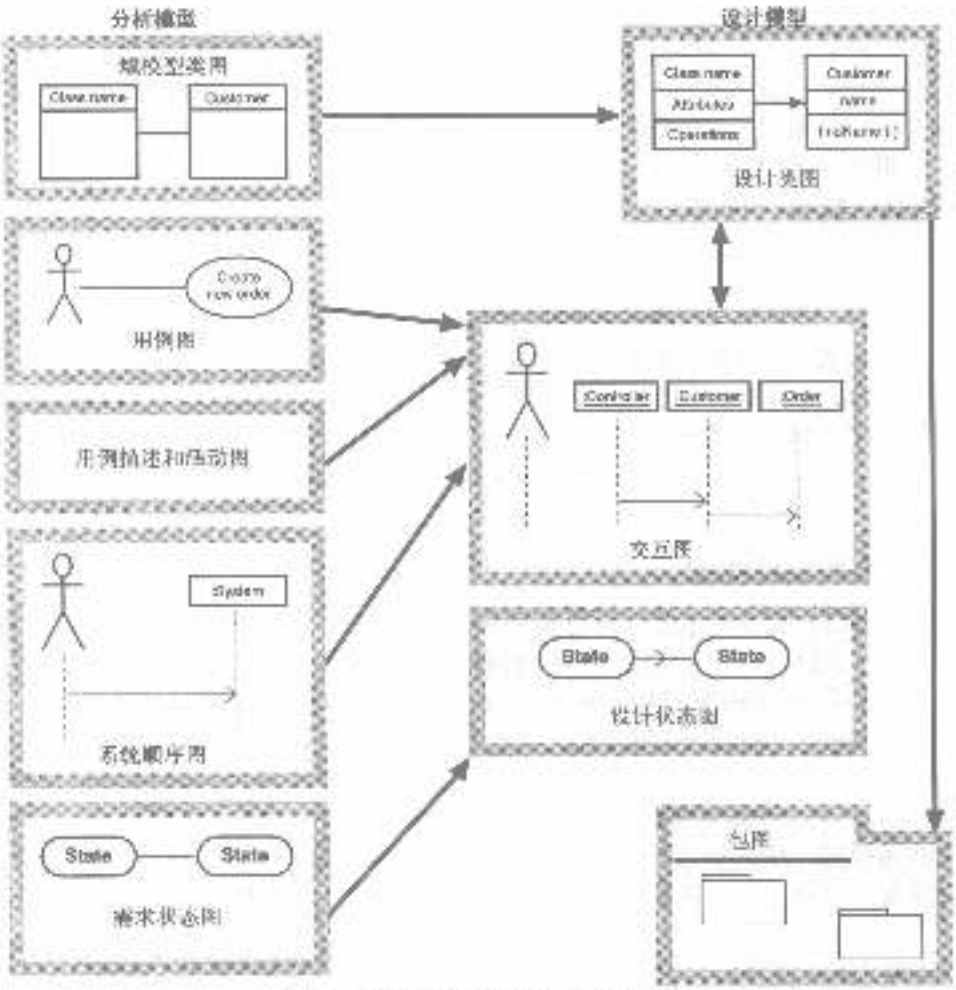


图11-5　带有各自输入模型的设计模型

11.1.3　面向对象设计过程

面向对象设计是模型驱动和用例驱动的。如图11-5所示，整个设计过程将需求模型作为输入，并生成设计模型作为输出。显然，我们需要一个方法来组织这项活动，并且其围绕用例展开。换言之，我们通过用例来开发设计模型用例。例如，每个用例都相应开发了一个交互图。一旦一组交互图设计完成，整个用例组的设计类图也就相应完成了。我们可以将这个过程划分为4步，如图11-6所示。

> 整个设计过程
> 1. 开发初步设计类图，显示导航可见性
> 2. 为每个用例开发顺序图完成用例设计
> （a）设计初步顺序图
> （b）设计多层的顺序图
> 3. 向类添加方法特征和导航信息
> 4. 将解决方案划分成适当的包

图11-6　设计步骤

首先为设计类图创建一个初步的模型。该模型中必须包括一些基本信息，如属性名等，以便进一步开发交互图。这一步是第2步的基础，而第2步是整个设计过程中最复杂的部分。

设计的第2步是开发交互图，即为每一个用例产生一个交互图。开发一个交互图需要多步，交互图确定了哪些对象一起工作以及怎样协同工作。首先，需要开发一个只包括域类的顺序图，随后添加数据访问类和可视层类，以得到多层的

解决方案。并发交互图是面向对象系统设计的核心。正如图11-5所示，交互图将输入模型使用了用例图、用例描述、系统顺序图和设计类图。我们称这些设计模型的最终开发结果为用例实现。在这个例子中，术语实现指的是对每个用例的详细系统过程进行说明，而不是实现用例，换句话说，是制定了软件的蓝图。因此不仅仅面向对象分析是用例驱动的，面向对象的设计也是用例驱动的。

用例实现：对每个用例的详细系统过程的说明。

设计的第3步是根据开发交互图时得到的信息，回过头来设计类图和并发方法名称。在本次迭代过程中，导航可见性和属性信息也做了相应修改。

设计的最后一步是用包图将设计类图分解成相关的功能。有几种分割系统的方法。一种方法是建立子系统。另一种方法是分层。在第9章中，已经学习了多层和多级结构。本章中，将学习如何将设计类图分解成多个包，来表示多层系统中不同的层次。我们关注的是基本的多层设计，包括可视层（用户界面类）、域层（来自问题域模型类图的问题域类）和数据层（数据访问类）。域层有时又叫问题域层和业务逻辑层。包图使我们可以从较高的层次去看待最终的系统。

11.2　设计类和设计类图

如图11-5所示，设计类图和详细交互图需要协同工作，它们的开发过程是同时进行的。设计类图的第一次迭代是基于域模型和设计准则的，初步设计类图用于辅助并发交互图。由于设计决策是在交互图的开发过程中制订的，所以结果又可以用来完善设计类图。

域模型类图揭示了问题域类和它们之间的联系。由于分析是一个发现的过程，所以分析人员一般不太关心属性或方法的细节。然而在面向对象程序设计中，类的属性必须被声明成public或private，每一个属性必须定义类型，比如character或numeric，在设计中，详细定义这些细节是很重要的。详细定义传递给方法的参数和方法的返回值也是很重要的。有的时候，分析人员还要定义每个方法的逻辑。我们还过集成来自交互图和其他模型的信息来完成设计类图。

开发者建立设计类图时，还要在以前的域模型的基础上增加很多类。要建立一个完整的面向对象系统，还要定义很多其他设计类。图11-1中的输入窗口对象和数据库访问对象就是很好的例子。这些类被定义后，设计师在各种类图中记录它们。系统中的类可以划分为几类，比如，用户界面类。有时候，设计师同样可以以了系统的方式来划分类。不管使用哪种方式，设计师都需要借助类图来记录他们的设计，因此类图的使用方式也会有所不同。下面我们介绍设计类图中使用的符号和在第一次迭代中用到的设计准则。

11.2.1　设计类符号

统一建模语言没有说明设计类表示法和域模型表示法的区别，但实际上二者是存在区别的。主要是因为设计模型和域模型的目标不同。域模型表现的是用户工作环境下的事物以及它们之间的联系。类不是特指软件类，但是一旦我们开始创建设计类图，我们就要定义软件类。因为在设计过程中要定义很多不同类型的设计类，UML使用了一种特殊的表示法，叫做构造型。它允许设计者为每一个类指明一个专门的类型。构造型将模型元素以特定的类型分类。它通过说明我们想要强调的特征来扩展模型元素的基本定义。构造型的表示是将类型的名称放到书名号中，像《control》这样。设计用例图的时候，你会接触到构造型。前面已经学过，参与者和用例之间的连线表明了一种关系，这些关系中有一种叫《includes》关系。

构造型：按照模型元素的特征进行归类的一种方式，用《》符号描述。

有4种类型的设计类被看做标准的设计类：实体类、控制类、边界类和数据访问类。图11-7展示了这4种构造型的表示法，设计类有两种类型的表示法。左边的表示类的矩形框给出了完整的符号。在每个矩形框中，类名上方是构造型，右边的圆形符号是这些构造型的简化符号，称为图标。我们会经常使用这些构造型图标，但是在大多数情况下我们更倾向于使用完整的符号。

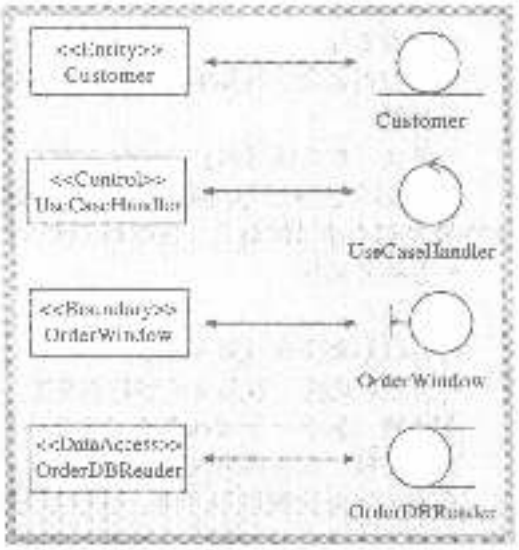


图11-7 设计模型中的标准构造型

实体类是问题域类的设计标识符。换句话说，它来自域模型。这些对象通常是被动的，因为它们只是等待事件的发生。通常，它们还是持久类。一个持久类在程序结束后仍然存在。换句话说，在系统关闭后，这些数据仍要存在。很明显，实现方法是将它们写入文件或数据库中。

边界类是存在于系统的自动化边界上的类。在一个桌面应用系统中，这些类可以是窗口类或所有和用户界面有关的其他类。

控制类是在边界类和实体类中间起协调作用的类。换句话说，它负责从边界类对象获取信息，然后发送给适当的实体类对象，就像是域层和可视层之间的一个交换机。

数据访问类是从数据库获取信息或向数据库发送信息的类。不是在实体类方法中包含数据库访问逻辑（包括SQL语句），而是设计一个专门用于访问数据库的数据访问层。

实体类：是问题域类的设计标识符。

持久类：程序结束后仍然存在的实体类。

边界类：存在于系统的自动化边界上的类，如输入窗口。

控制类：是在边界类和实体类中间起协调作用的类，在域层和可视层之间起开关控制的作用。

数据访问类：是从数据库获取信息的类。

11.2.2 设计类表示

图11-8所示为在图11-3中首次出现的设计类符号的详细信息。名字部分包括类名和构造型信息。底部的两个部分是关于属性和方法的更详细的信息。

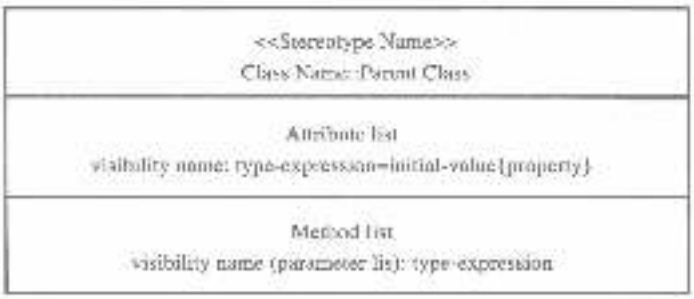


图11-8 设计类的表示法

分析员用来定义属性的格式如下：

- 属性可见性，可见性表示其他对象是否能直接访问该属性（"+"表示可见，"-"表示不可见）
- 属性名称
- 类型表达式（例如字符型、字符串型、整型、数字型、货币型或日期型）
- 初值
- 原型（在花括号内），比如{关键字}

第二部分包含了方法特征信息。方法特征显示了需要调用这个方法的所有信息。它给出了需要发送的消息的格式，包括如下内容：

- 方法可见性
- 方法名
- 方法参数列表（输入参数）
- 类型表达式（方法返回参数的类型）

可见性：表示一个属性是否可被外部对象直接访问（"+"表示可见，"-"表示不可见）。

方法特征：表示调用该方法所需的所有信息。

在面向对象的程序设计中，分析员通过使用完整标志来识别每一个方法。一些面向对象语言允许多个方法使用同一个名称，因此要将它们区分开来，就要使用参数表。在这些语言中，需要用方法名和参数表去调用相应的方法。例如，假设我们想通过客户ID或客户名来取得客户记录，那么就要建立两个方法，并为这两个方法起同一个名字，如："GetCustomer（CustomerID）"和"GetCustomer（CustomerName）"，当一个方法的名字相同，只是参数列表不一样时，我们称此为重载方法。要确定调用哪个方法，运行时，环境需要借助于传递的参数：如果传递的是一个数字，则调用GetCustomer（CustomerID），如果传递的是一个文本，则调用GetCustomer（CustomerName）。

重载方法：一个具有两个或更多参数列表的同名方法。

域模型属性列表包含了所有在分析活动中发现的属性。设计类图包含了关于属性类型、初值和原型的更多的信息。它也可以包含用来声明的构造型，Student类就是一个<<实体>>构造型。如图11-3所示，Student设计类图的第三部分包含了该类的方法特征。UML是一种通用的面向对象的表示方法，并不局限于某一种语言。所以，这种表示法与程序设计中方法的表示不完全一样。

比如，对于有程序设计经验的人来说，构造器的表示是CreateStudnt:Student(name, address, major)。构造器是用来建立类的新对象的方法。在很多程序设计语言中，构造器和类同名。然而，我们使用创建语句来跟踪在交互图中使用的消息名称。图11-3所示是另一个构造器。在第二个方法中，只有StudentID被传递进来。这说明一个学生只有ID信息，构造器要自己填充学生的其他信息。这通常要求访问数据库以获得其他字段的值。

图11-3中，用下划线标识的findAboveHours(int hours):StudentArray方法是一个比较特殊的方法。在面向对象方法中，类是创建个体对象和实例的模板，大多数方法适用于对象。但是，分析员常常需要检查类的所有实例。这种类型的方法称为类方法，它们用下划线表示，以示区分。在VB.NET里，这种方法称为共享方法；而在Java里，叫静态方法。这种方法是由类而不是由类的某一对象执行的。因为这些方法供类使用，所以它们不依赖于某一对象。如果需要，还可以访问所有对象的数据。在这个例子里，findAboveHours遍历类的所有实例，并返回所有总小时数大于输入参数的学生。

类方法：与类而不是类的对象相关的方法。

11.2.3 开发初步设计类图

开始设计时，我们开发一个基于域模型的初步设计类图。图11-9是在第5章中为RMO设计的域模型（见图5-41）片断。

可以通过扩展域模型类图的方法来构造初步设计类图，它需要两个步骤：（1）给属性添加类型和初值；（2）添加导航可见性。正如前面所提到的，面向对象的设计是用例驱动的，所以，我们先选择一个用例，然后只关注这个用例所涉及的类。从几个简单的用例入手，往往是一个不错的选择。因此，我们从“查询可用条目”这个用例入手。

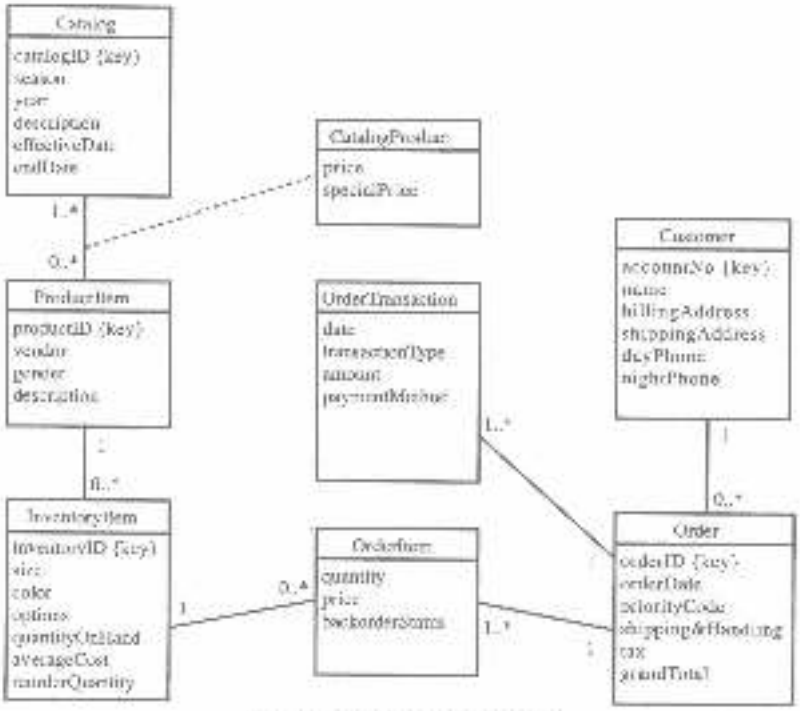


图11-9 RMO域模型类图片断

1. 属性细化

属性的细化很简单。设计者往往凭借自己的经验来确定属性的类型。在大多数情况下，属性对于外部对象来说是不可见的。属性前的“-”号意味着该属性是不可见的。每个类也需要一个新的部分来描述该类的方法特征。

2. 导航可见性

就像前面提到的，面向对象系统是相互作用的对象的集合。在设计阶段得到的交互图记录了对象之间的交互。但是，如果一个对象通过发消息的方式和另一个对象进行交互，那么第一个对象对于第二个对象来说，必须是可见的。这里我们所说的导航可见性，就是指一个对象能够看见另一个对象并与其进行交互的能力。在设计中，我们会用到两种导航可见性：属性导航可见性和参数导航可见性。属性导航可见性是指，一个类具有一个引用其他对象的属性，这种可见性是通过属性引用实现的；而参数导航可见性是指，一个对象作为参数被传递给另外一个对象，通常发生在函数调用的时候。有时，开发人员直接把导航可见性简称为导航或可见性，但是我们推荐使用导航可见性，以便与属性、方法的public和private可见性区分开来。

导航可见性：是一种设计准则，指一个对象可以看到另一个对象并与之交互。

图11-10所示的是Customer类和Order类之间的单向导航可见性。注意，在Customer类中有一个叫myOrder的变量，该变量指向某个Order实例。导航箭头表示一个Order对象必须对Customer类可见。在这个例子中，我们添加了myOrder属性来强调这个概念。

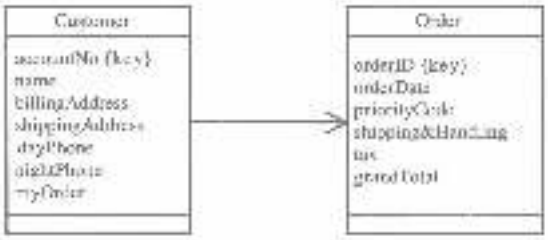


图11-10　Customer和Order间的属性导航可见性

你可能会心存疑惑：怎样确保myOrder属性引用正确的Order对象呢？一个办法是，让Customer对象创建Order对象。Customer类的构造器在创建新实例的时候便创建需要的Order对象。另外一个办法是，将Order作为方法的参数传递给Customer对象，例如，代码myCustomer.addOrder（anOrder）可以将一个特定的Order对象传递给一个特定的Customer对象。随后Customer对象将这个Order对象赋给图11-10中的myOrder属性。在这个例子中，导航可见性最初是参数导航可见性，然后又被提升为属性导航可见性。如果anOrder对象只是存在于一个临时或局部变量中，那么，该参数导航可见性不会提升为属性导航可见性。

作为设计者，你必须时时考虑导航可见性，因为只有依靠它才能确定对象之间的交互。调用一个对象的方法需要借助“.”号完成。设计的一个任务就是要说明一个对象对哪些对象有导航可见性。导航可见性可以是单向的也可以是双向的。比如，一个Customer对象可以看见一个Order对象。这意味着，Customer对象可以知道客户下了哪些订单。在程序里，Customer类用一个变量或变量数组来指向这个客户的一个或多个Order对象。如果导航可见性是双向，Order对象就也有指向Customer对象的变量了。而如果导航可见性不是双向的，Order对象就没有指向Customer对象的变量。在设计类图中，属性导航可见性用类之间的箭头表示，箭头指向可见的类。参数导航可见性用虚线箭头表示。

现在，我们为RMO的类图添加导航可见性。别忘了，目前我们开发的只是初步设计类图，所以在随后的设计过程中，我们可能还需要对图中的导航箭头做适当修改。在构建导航可见性时，需要解决的基本问题是：哪些类需要访问其他类以及能访问哪些类？下面是一些指导原则，不是必须遵守的。

* 一对多的关系。它说明了上级/下级关系通常是由上级到下级的导航，

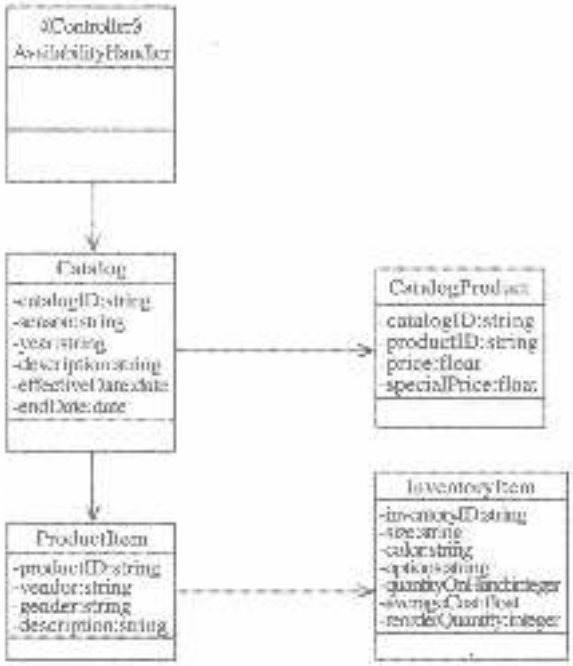


图11-11　RMO“查询可用条目”用例的初步设计类图

比如，从Order到OrderItem。而有的时候，这些关系组成了导航链的各个层次，比如，从Catalog到ProductItem再到InventoryItem。

- 强制关系。在这种关系里，一个类的对象在没有另一个类的对象的情况下是不能存在的。这种关系是从相对独立的类到不独立的类的导航，比如，从Cutomer到Order。
- 当一个对象需要获取来自另一个对象的信息时，必须有导航箭头指向对象本身或父类。
- 导航箭头可以是双向的。

图11-11是RMO的一个设计类图，其给出了上述两个步骤的结果。“查询可用条目”用例会用到4个类，分别是：Catalog类、CatalogProduct类、ProductItem类和InventoryItem类。正如上面的规则所述，图11-11显示了Catalog对CatalogProduct和ProductItem的导航可见性。Catalog、ProductItem和InventoryItem形成了一个导航链层次，自上向下导航。作为关联类的CatalogProduct，对Catalog也要可见。

图11-11中多了一个Availability Handler类。它是一个构造型并充当控制类的角色。我们会在下一节讨论控制器类。

这里要强调三点：第一，由为设计的细节是依靠一个个用例完成的，所以要确保交互图支持并实现最初定义的导航；第二，导航箭头要随着设计过程进行相应的修改，以保持和设计细节的一致；最后，要根据为用例创建交互图时制订的设计决策，为每个类添加方法特征。

11.2.4 设计模式和用例控制器

模式，又称模板，在日常生活中经常用到。厨师们使用他们的模式，即菜谱，制作出美味的菜肴；裁缝使用他们的模式，制作出合身的衣服；工程师们使用他们的模式，构建高楼、系统和其他各种产品。模式为解决问题而生。经过不断尝试，这些适用于特定问题的模式形成了一套解决方案，这些方案能够被反复使用。随着时间的推移，这些方案广泛流传，并最终成为被普遍接受的标准。

本章主要介绍两种标准的设计模式——用例控制器和三层设计。标准的设计模板深受广大软件开发人员的欢迎，因为其能加快面向对象设计的进度。这些模板的正式名称叫设计模式。1996年，Eric Gamma、Richard Helm、Ralph Johnson和John Vlissides合著的《Elements of Reusable Object-Oriented Software》一书的问世，使得设计模式成为被广泛接受的设计技术。该书的4个作者被称为Gang of Four(GoF)。在书中，作者们列举了23种最基本的设计模式。现在存在很多的设计模式——从低层的编程模式，到中层的架构模式以及高层的企业模式。Java和.NET这两个主要的企业级平台，都含有大量的企业模式，这在很多著作中都有论述。

设计模式：是使用良好的设计准则的解决方案模板。

图11-1中有这么一条用例消息。它源自于外部的参与者，然后到一个窗体类（输入窗体），最后到达一个问题域类。系统设计的一个问题是，明确哪些域类应接收输入消息，从而降低耦合度，提高内聚度，并使用户界面类和域类相互独立。在这种情况下，设计者在用户界面类和域层类中间加入一个协调类，这些类叫做用例控制器。例如，对于“查询可用条目”用例，将会产生一个叫做AvailabilityHandler的控制器类。用例控制器完全是系统设计者人为创建的一个类。有时候，这些类也叫做制品，表示某些事物是为了满足特定的需求而创建的。我们之所以把这些类称做制品，是因为工业中经常这样使用。随着设计的不断深入，你会发现需要创建很多作为制品的类，这些类不是基于域类的，它们只是为了完成执行用例。

用例控制器：系统设计者创建的，用来作为输入消息收集点的类。

制品：系统设计者创建的，用来处理必要的系统功能的类。

用例控制器相当于一个开关，它将接收的消息路由给正确的域类。实际上，用例控制器

是外部世界和内部系统的一个协调者。图11-1中存在一个位于系统边界的输入窗口对象和一个叫做学生对象的问题域对象。假如输入窗口对象要向一些对象发送消息，那该怎么办？这就需要引用所有的问题域对象。这样的话，输入窗口对象与系统之间的耦合度可能会很高，因为有很多的连接。因此，用一个用例控制器对象来管理所有的输入消息就能够降低用户交互对象与问题域对象之间的耦合度。用例控制器有其自身的业务逻辑，这些业务逻辑管理用例的消息流。这样，域类就能够只关注属于它们自己的职责，从而提高它们的内聚度。

在后面章节的所有例子中，我们会为每个用例定义一个控制器，这是一个普遍的做法。并且许多开发环境，如Java Struts等，会为每个用例定义一个控制器。当然，这就产生了许多制品。如果有100个用例，那么就有100个控制器制品。为了减少控制器的数目，开发人员通常将功能相似的几个控制器合并为一个。不管采用哪种方法，只要设计合理，都会形成一个好的解决方案。

在前面讲述的就是系统设计的思想，即在耦合度、内聚度以及类的职责间寻求一个平衡点。在接下来的章节中，我们会详细讨论这个过程。

实践指导

指定合适的控制器类作为域类的进入点。 ■

11.2.5　一些基本的设计准则

既然你已经了解了一个面向对象程序是如何运行的，也学习了设计类的表示方法，让我们来回顾一些指导设计决策的基本准则。在本章中，我们讨论面向对象设计的步骤时，会提到一些好的面向对象设计的准则。以下的基本准则对面向对象设计的每一部分都很重要。

1. 封装和信息隐藏

封装是一种设计概念，它说明一个对象是一个包含数据和程序逻辑的独立单元。每个对象内部携带自己的数据，并提供访问数据的方法。每个对象还提供一系列调用对象方法的服务。用这种方法设计软件的一个好处是，软件开发人员可以用搭积木的方式设计系统。几乎所有的工程规范都有像积木块一样的标准单元，它们可以组合形成最终的设计。被封装的对象等价于软件的每一个积木块。

封装：是一种设计准则，规定数据和程序逻辑包含在一个独立的单元中。

程序员依靠封装来实现对象重用。每一种面向对象的语言都有一些在系统中被反复使用的标准对象。这些对象的标准集提供了在同一系统中多次使用的基本服务——有时甚至是在不同的系统中多次使用。重用的一个最普遍的例子是为电脑或网络应用设计的用户界面。设计者通常重用相同的类来开发窗口及按钮、菜单、图标等窗口部件。有时问题域类也可以被重用。

对象重用：是一种设计准则，说明标准对象可以在系统中反复使用。

和封装相关的概念是信息隐藏，信息隐藏的意思是，和一个对象有关的数据对外部世界是不可见的。换句话说，对象的属性是私有的，要用一组方法来访问和修改数据。虽然这条准则主要是程序上的概念，并且主要有利于编程和测试，但它是一些重要准则的基础。如果对数据属性的访问是通过下面一节将要介绍到的含有方法的标准接口来实现的，那么系统中对象间的联系和耦合会好得多。

信息隐藏：是一种设计准则，指和一个对象有关的数据对外部世界是不可见的，要使用一组方法来访问和修改这些数据。

2. 耦合

耦合这个术语来自导航可见性。在前面的例子中，Customer对Order有导航可见性，也可

以说，Customer和Order是耦合的，或者是有联系的。现在，在整个系统的所有类中扩展可见性这个概念。耦合是对设计类图中的类与类之间连接关系紧密程度的定性的度量。一种比较容易理解耦合的方法是，看设计类图中导航箭头的个数，对系统来说，弱耦合比强耦合好。换句话说，较少的导航可见性箭头说明系统更易于理解和维护。

耦合：是对设计类图中的类与类之间连接关系紧密程度的定性的度量。

之所以说耦合是个定性的度量，是因为在系统中并没有一个特定的数字来定量地说明耦合。设计者必须对耦合有一定的理解——当耦合太强时，设计者要能够认识到，或者要能够知道什么样的耦合才是合适的。耦合是作为一个设计过程用例被评价的，如果每个用例设计都有合适级别的耦合，那么整个系统也会有比较合适的耦合。

我们再回顾一下图11-1，仔细观察对象之间的消息流。很明显，互相通信的对象必须有导航可见性，所以它们就是耦合的。输入窗口对象为了向学生对象发送消息，则它对学生对象必须有导航可见性，所以输入窗口对象和学生对象是耦合的。但是注意，输入窗口对象并没有和数据库对象连接，所以它们之间不是耦合的，如果我们设计一个系统使输入窗口对象也可以访问数据库对象，那么这个用例的整体耦合度将会增加——也就是说，会有更多的连接。这样做是好是坏呢？在这个简单的例子中，这也许不会成为什么问题。但是对一个有10个或更多用例的系统，无规则的连接和导航可见性会导致很强的耦合，使系统很复杂。

那么，为什么强耦合度不好呢？因为强耦合增加了系统不必要的复杂性，使系统很难维护。一个类的变化会波及整个系统。所以，有经验的分析员会尽量简化耦合，并降低对新系统设计的影响。

3. 内聚

内聚指的是一个类中各种功能的一致性。内聚是对一个类的功能一致性的定性度量。比如在图11-1中，我们希望学生类有这样的方法或功能，它能够输入学生信息，如学生的学号或名字。这代表了一个单一功能和一个高内聚类。但是如果同一个对象也有分配教室或分配教授的方法呢？类的内聚度就会被减弱。

内聚：是对一个类中功能一致性的定性度量。

低内聚的类有几个方面的负效应。第一，它们难于维护，因为它们有很多不同的功能，所以对系统内的变化非常敏感，容易产生联锁反应。第二，很难重用这些类，它们有很多不同的（通常是无关的）功能，因此，在其他环境下的重用通常没有意义。比如，一个有按键功能的按钮类经常被重用。但是，有按键功能和用户登录功能的按钮类却很少被重用。最后一个缺点是，没有内聚的类难于理解。通常，它们的功能相互交叉，逻辑非常复杂。

虽然内聚无法用一个度量系统来衡量，但是我们可以把类的内聚性分成极低、低、中等以及高内聚4种类型。我们最需要的是高内聚。一个极低内聚的类可以是这样的，它在不同的功能领域都有服务的任务，比如一个类既可以访问网络又可以访问数据库。这两种活动是完全不同的，而且是为了达到不同的目的，因此把它们放在一个类里就会造成低内聚。

举个低内聚的例子。这个类可能在相关领域有不同的任务，也可能完成数据库中所有表的数据访问功能。然而，更好的方法是用不同的类来访问用户信息、命令信息和库信息。虽然功能是一样的——就是说，这两种方法都访问数据库——但是传入和导出的数据类型是不一样的。所以，与整个数据库连接的类就不像只与用户表连接的类那样易于重用。

举个中等内聚的例子。这个类有联系密切的相关任务，比如一个记录客户信息和客户账号信息的类。可以定义两个高内聚的类，一个记录客户信息，比如名字和地址；另一个或一组类记录客户账号，比如余额、支出、信用卡信息和所有的财务活动。如果客户信息和账号信息是受限的，它们就被组合到一个中等内聚的类中。在系统设计中，中等内聚或高度内聚

的类都是可以接受的。

实践指导

经验丰富的设计者应尽量降低耦合度，提高内聚度。在设计过程中，应牢记这一点。 ■

4. 变量保护

设计中一条基本的准则是变量保护。变量保护指的是将系统中比较稳定的部分和需要改变的部分分开。事实上，在早期的用户驱动和事件驱动的系统中，业务逻辑总被放在可视层的类里，通常是输入窗体类里。比如早期的Visual Basic和PowerBuilder等开发图形用户界面。它们都将所有的业务逻辑放在可视层的类里。这样设计的问题在于，当一个界面需要更新时，所有的业务逻辑也需要重写。一个更好的实现是，解除用户界面逻辑和业务逻辑之间的耦合。这样的话，我们可以在不影响业务逻辑的情况下重写用户界面逻辑。换句话说，相对稳定的业务逻辑应与用户界面分离，进而能变量保护。

变量保护：是一种设计准则，将不易改变部分与易变部分分离。

同样，当业务逻辑需要添加新的类或方法时，又会如何呢？如果用户界面类与业务逻辑紧耦合，那么在用户界面类中会产生一系列的改变。然而，用户界面类可以将所有的输入消息发送给用例控制器，这样的话，对业务逻辑的改变就不会影响到控制器。这条准则几乎会影响到所有的设计策略，因此在所有的设计活动中应注意运用这条准则。

5. 间接

在面向对象设计中，间接是用来降低耦合和防止稳定部分被改变的一个准则。间接用于降低对象之间以及系统的组件之间的耦合度。这通过在对象或组件之间添加一个起链接作用的协调类完成。换句话说，操作不是从A直接到B，而是通过一个中间者C，A将消息发送给C。C再将消息传送给B。

间接：是一种设计准则，在两个类之间添加一个协调类以达到降低耦合的目的。

尽管变量保护的方法有很多，但是经常使用间接。添加一个协调对象可以把对系统的改变仅局限于该对象。间接同样是一条有助于系统安全的设计准则。许多公司在内网和外网之间设有防火墙和代理服务器，代理服务器可以接收和发送消息。代理服务器就像一个真正的服务器，它接收E-mail和HTML页面请求等消息。但是它并不是一个真正的服务器，它只是接收消息并将它们分发给客户。这样就可以设置安全控制以便保护系统。

11.3　实现用例和定义方法——顺序图设计

正如我们在本章的最开始提到的一样，开发交互图是面向对象设计的核心。用例的实现是在交互图的开发过程中完成的。实现用例的过程就是确定哪些类通过发送消息与其他类进行协作的过程。在设计时，要开发两种交互图：顺序图和协作图。用任何一种图都可以完成设计。有些设计者喜欢用顺序图，有些喜欢用协作图，我们首先讨论用顺序图进行设计，然后讨论用协作图进行设计。

交互图既是一种完成设计活动的机制，又可以用来记录设计的结果。在软件设计的时候，设计者开发诸如设计类图和交互图这样的图。这些图向程序员和其他开发人员说明了其结构和行为的细节。图本身并不是目标，相反，它们代表了基于优秀设计准则（如耦合、内聚和职责分离）的设计决策的结果。典型的例子是，设计者设计草图后，通过评价它们是如何反映优秀设计准则的来评价它们的质量。设计者为了提高质量和纠正错误可能会多次修改这些图。这些图就是存储设计者想法的一种方式，也是向开发人员展现最终设计结果的一种方式。

用例设计的一个任务就是确定用例需要哪些对象，以及各个对象要执行哪些功能。下面

介绍为对象分配职责的概念。

11.3.1 对象职责

面向对象开发的一个基本准则是对象职责的思想，即由对象负责实施系统过程。设计的一个主要活动就是定义对象类的任务。这些任务分为两类：认识和行动。换句话说，一个对象应该知道什么？一个对象应该做或者激起什么？

对象职责：是一种由对象负责实施系统过程的设计准则。

认识包括这样的任务，如认识自己的数据，认识其他的一起协作执行用例的类。很明显，一个类应该认识自己数据，存在什么样的属性，怎样维持这些属性的信息。还应该认识到哪里去找需要的信息。比如，在一个对象初始化的时候，构造器需要其他对象作为参数为其提供一些数据。一个对象应该认识到，也就是说，一个对象应该具有对向它提供信息的对象的导航可见性。比如，在图11-3中，学生类的第一个构造器不接收学生ID值作为参数。学生类负责根据它所知道的某些准则创建一个新的学生ID值。

行动包括一个对象所做的支持用例执行的所有活动。一些活动接收和发送信息。另一些对象负责实例化或创造完成用例所需的新的对象。类之间要相互协作完成用例的实施。一些类用来统筹协作。比如，对于图11-11中的“查询可用条目”用例，Catalog类主要负责处理用例的实施。其他类，如InventoryItem类，只负责提供它自己的信息。

11.3.2 “查询可用项目”用例的初步顺序图

首先，我们复习一下顺序图的结构和句法，然后按照一些规则和步骤来设计系统。在第7章中，在讲述如何建立一个系统顺序图的时候，我们已经对顺序图有了大致的了解。因此，现在理解顺序图，甚至自己设计一个，都应该得心应手。要记住系统顺序图是用来为一个用例或者一个场景记录系统的输入与输出的。它描述了系统与参与者所代表的外部世界之间的相互关系。系统本身被看做是一个叫做System的对象。系统的输入就是参与者传递给系统的消息，输出通常是回复的消息，表示数据正在返回。图11-12是图7-14的细化版本，它描述了顺序图的基本组成，并做出了解释。我们从一个简单的RMO典型用例“查询可用条目”开始。如图11-12所示，每个消息都有一个源头与目的地。在顺序图中，由于只有两个对象有生命线，因此，消息源与目的地就受到了限制。如果分析详细的顺序图的话，那么就要做出一些很重要的决策，即消息源对象以及消息目的地对象。

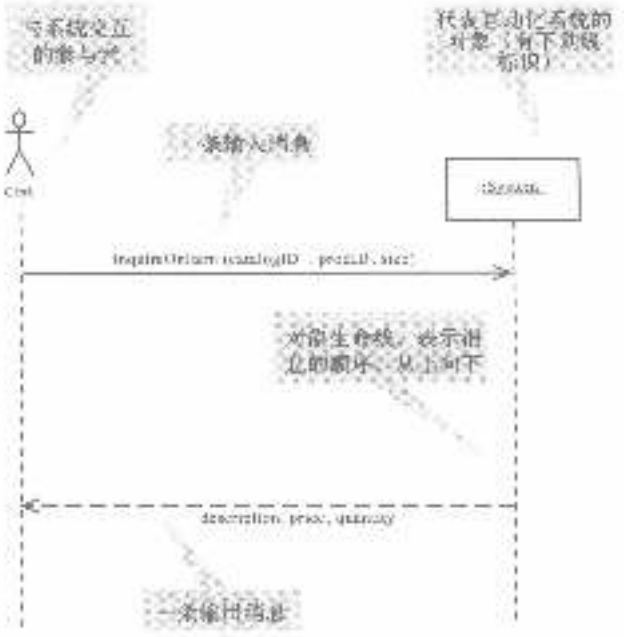


图11-12 “查询可用条目”用例的系统顺序图

在第7章中提到的输入消息的句法如下：

*[true/false condition] return-value := message-name (parameter-list)

而对于输出消息，我们通常只需写出参数表而无须加括号。

详细的顺序图使用和SSD相同的元素。但与SSD不同的是，系统中所有的内部对象和消息取代了:System对象。换句话说，对于SSD，系统被看做一个黑箱，我们不知道其内部处理过程。那么设计的目标就是打开这个黑箱，并找到那些在自动化系统中进行的内部处理。正如图11-1所示，我们需要确定实现用例或者场景的内部对象和消息。

基于图11-12中的系统顺序图，我们继续“查询可用条目”用例的详细设计。扩展系统顺序图的第一个步骤是明确哪些对象应该参与到用例实现中。系统顺序图指出有关每个条目的返回信息应该包括description，price以及quantity。从初步的设计类图中，我们可以看出，description从ProductItem中得到，price从CatalogProduct中得到，而quantity从InventoryItem中得到。因此，这三个对象就包括在初步顺序图中。如上文所述，我们仍需添加一个用例控制器对象。该用例的用例控制器为AvailabilityHandler。有一点需要注意，用例控制器对象作为用例所有输入消息的中心采集点，发挥了消息交换机的作用。它接收输入消息并把它们分发给合适的内部对象。部分设计就是要在设计类图中已经定义的导航可见性的基础上，决定如何分配这些消息。

下面我们使用图11-13中系统顺序图的元素来构建初步的图表。第一步需要明确用例所需的所有对象。首先，用用例控制器:AvailabilityHandler来替换:System对象。然后，添加其他需要包含在用例中的对象。在设计类图11-11中，我们看到从Catalog到ProductItem，再到InventoryItem之间存在层次关系。Catalog中也有到CatalogProduct的导航可见性。最后添加输入消息。在这个例子里只有一条消息。图11-13说明了这一步骤。

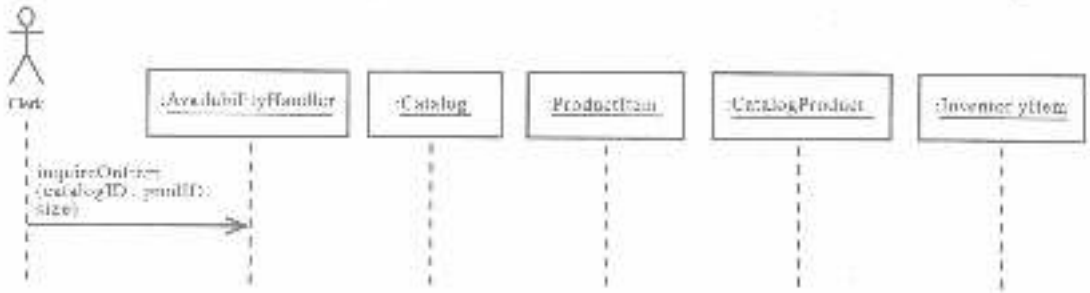


图11-13　“查询可用条目”用例包含的对象

下一个步骤需要确定对象之间的内部消息，以及确定每个消息的源头和目的地，从而收集到所有需要的信息。明确哪些消息是必要的，哪些对象会被激发。要基于前面提到的设计准则：耦合度、内聚度、职责和控制器。

图11-14显示了“查询可用条目”用例的完整的初步顺序图。正如前文所述，Catalog对象相对于其他含有所需信息的对象来说，处于导航层次结构的上层。因此，:AvailabilityHandler将输入消息传给:Catalog对象。因为:Catalog对象位于导航层次的顶层，所以让它收集信息并反馈给控制器是很有意义的。:Catalog对象将消息传送给:ProductItem以及:CatalogProduct，从而获得描述与价格。然而，:Catalog与:ProductItem并没有直接的导航可见性，所以它就发送一个消息给:ProductItem寻求帮助以获得数量。:ProductItem对象知道:InventoryItem含有数量信息。因此就发出请求并得到了数量信息。:Catalog收集完毕所有的信息后，再返回给:AvailabilityHandler，后者再将这些信息返回给职员。

在继续讨论之前，分析一下基于设计准则，即耦合度、内聚度、职责和控制器的解决办法。

用例控制器为内部对象和外部环境提供了联系。这就限制了外部环境与单一对象的耦合

度。分给:AvailabilityHandler的任务是捕捉输入消息，将它们分发给正确的内部域对象，以及给外部环境返回所需的信息。通过使用类似交换机的用例控制器，域对象与环境之间整体的耦合度就受到了限制。同时，:AvailabilityHandler也是高内聚的，它只有两个主要任务。

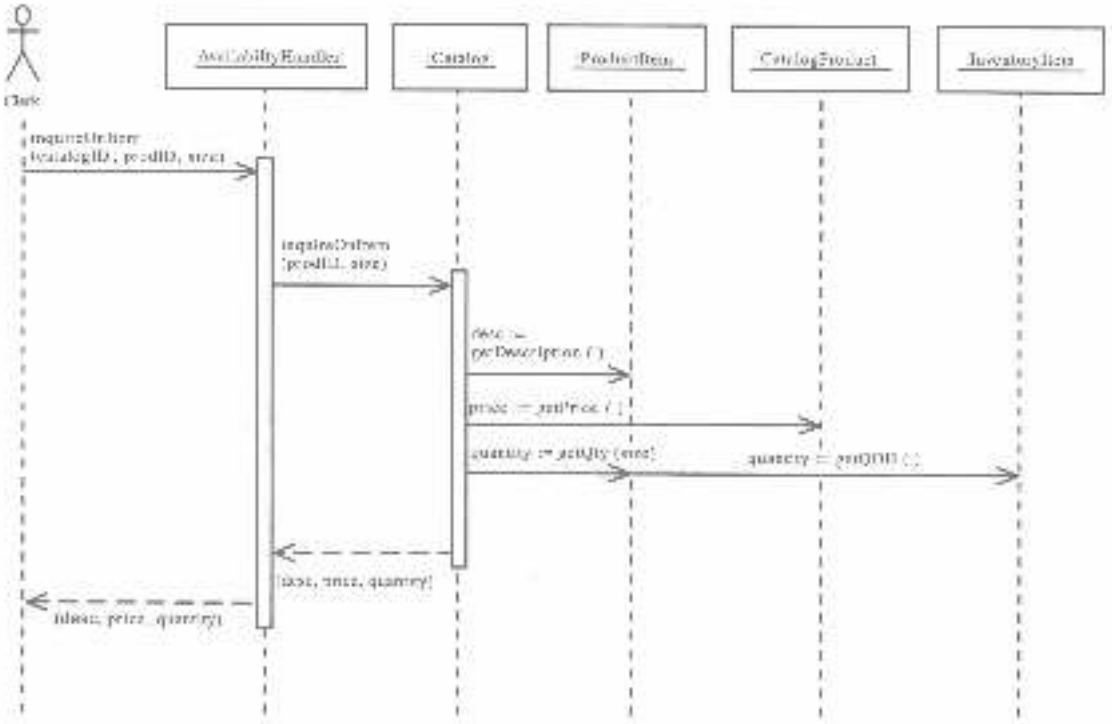


图11-14 “查询可用条目”用例的初步顺序图

分配给:Catalog的任务是收集产品分层结构中所有产品和库存信息。既然:Catalog位于顶层，给它分配这个任务是很合理的。:Catalog类具有高内聚度，至少在这个用例中如此。耦合度比较简单，在分层结构中呈现垂直分布。这样看来，任务和对应消息的分配似乎符合了一个好的设计应该遵循的准则。

设计中获得数量的请求应该由:ProductItem发送给:InventoryItem？还是直接从:Catalog发出？如图11-14所示，这个请求确实是逐层传递的，而且由于这个分层结构很可能为其他的用例所使用，因此这个设计是一个比较好的选择。更直接的解决方案需要更少的消息，但是会增加总体的耦合度。:Catalog和:ProductItem都需要导航可见性，因此就增加了耦合度。综上所述，图示的设计更好。

图11-14所示顺序图里面有一个新的标记。这个位于:AvailabilityHandler和:Catalog生命线上垂直的矩形框叫做激活生命线。第7章曾讲过，对象的生命线用一条垂直的虚线来表示。一个对象只能处于激活状态或者未激活状态。如果对象在执行一个方法，那么它就处于激活状态。反之，如果这个方法完成了，那么这个对象就处于未激活状态。处于未激活状态的对象停留在存储器中，等待其他消息，并且保留了内部数据的属性值。激活生命线代表着一个对象处于激活执行状态的时间。在图11-14中，当:AvailabilityHandler收到了条目查询消息时，它就变成激活的（开始执行方法），并保持这个状态直到它给职员返回一个响应信息。

激活生命线：顺序图中用以表示方法处于激活状态时间的垂直矩形框。

显然，对于一个初步的设计，我们将注意力主要集中在问题域类和基本的相互关系上。许多其他的细节还可以添加进来，以使解决方案更加完整，比如用户交互类、基于姓名的用户检索甚至检索客户的数据库等。这也说明了当前的设计只是个初步的设计。

11.3.3　顺序图初步设计的指南和假设

即使图11-14所示的例子比较简单，但仍然能提取一些任务来使读者学习使用顺序图设计用例或者场景。不过，该设计过程需要基于几点假设。

1. 指南

下面的任务不是顺序完成的，而是在建立顺序图过程中根据需要完成的。我们把它们看做三个分离的任务并确保完成。

- 接收每个输入消息，并明确由这个输入消息产生的所有的内部消息。明确消息的目标：需要什么信息，哪个类（即目的地）需要这条消息，以及哪个类（即消息源）提供这条消息，有没有任何一个对象因为输入而被创建。这种分析有助于定义为部消息、源对象和目的地对象。也就是说，要尽力去定义那些用来支持输入消息的类和内部消息。
- 在处理每个消息的时候，一定要明确会受之影响的完整的类的集合。即从域类图中找出该消息所涉及的所有对象。第7章讲述了用例的前提条件和后续条件。在前提条件或者后续条件中罗列的任何类都应该包含在设计中。其他一些类，即使不在前提条件或者后续条件中，如果满足下列条件，也应该包含到设计中去：被创建的类、创建用例对象的类、用例期间更新的类以及为用例提供所需信息的类。
- 此外，要充实消息的结构，即添加迭代、真/假条件，返回值和传递参数。传递参数应该参考域类图的属性。返回值和传递参数可以是属性，也可以是类中的对象。

完成以上步骤就能够实现一个初步的设计。修改与完善是必要的，但我们这里还只专注于用例中的问题域类。

2. 假设

开发初步顺序图的过程需要基于一些假设。下面列举了3个。

- 技术假设：在第5章中确定业务事件的时候，我们首次提到了这个假设，同样，此处也需要该假设。在这里，并没有包括用户登录以及检测网络可用性的步骤。
- 内存假设：你可能已经发现，前面我们已经假设必要的类位于内存中并可供用例使用。此处，我们不用担心对象是否在内存中生成这个问题，而在多层设计中，我们需要对该假设做适当修改，即需要包含在内存中创建对象的逻辑。

可以这样理解这个假设，内存是无限的或计算机的硬盘可扩展为内存，我们不必担心对象是如何进入内存的。面向对象的设计语言实现了这个假设。这些语言带有一个底层数据库，当程序员需要一个对象时，系统会自动将其从硬盘移动到内存中，当不再需要时，又将其移回硬盘。

- 异常假设：初步顺序图假设不存在异常，其中没有处理查询的产品为空的逻辑。当然，系统中还会有更严重的异常，如信用卡验证失败。许多开发人员最初只定义最基本的逻辑，以后才会添加其他消息和异常处理逻辑。同样，我们也采用这种方式，对初学设计的人来说，更应该如此。

11.3.4　“维护产品信息”用例的初步顺序图

在进行多层设计之前，我们再看一个稍微复杂的初步顺序图的例子。“维护产品信息”用例由几个场景组成。从其域模型可以看出，产品信息与库存信息息息相关。产品信息指的是诸如滑雪夹克等信息，而库存信息指的是每种产品的颜色、大小以及相应的库存量信息。因

此，对于每种产品，都会有多条库存信息。维护这些信息的场景有：在不带库存信息或带有库存信息的条件下，增加一种产品；添加某种商品的库存信息；更新产品信息；更新库存数量等库存信息。在设计过程中，我们可能会发现合并其中一些场景将更易于对象重用。但是此处，我们分别对待每个场景。下面我们为添加新产品这个场景设计初步顺序图，其中也包括为产品添加库存信息。

设计的输入是类图，用例的输入是系统顺序图。此处的域模型类图同上个例子是一样的，如图11-9所示，另外一个输入是来自设计活动的系统顺序图，如图11-15所示。在这两个图的基础上，我们构建出图11-16所示的设计类图。由于这个用例设计的域模型类很少，所以设计类图显得比较简单。但是，输入消息和输出消息增加了其复杂性。总之，虽然其复杂性与"查询可用条目"用例相当，但是它展示了几个新的设计准则。

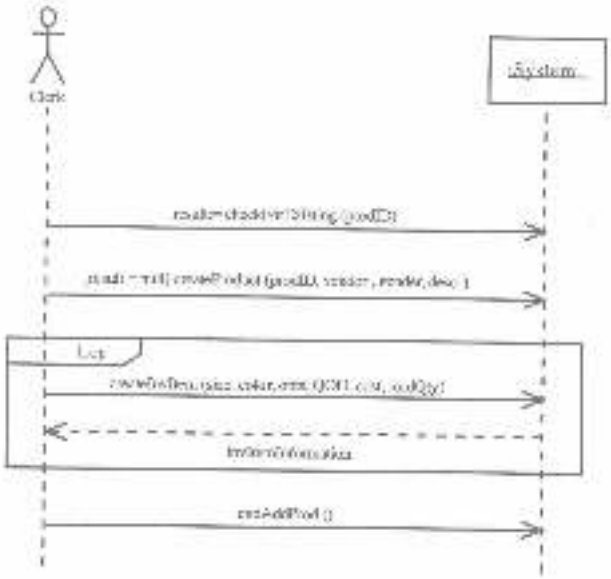


图11-15 "维护产品信息"用例的系统顺序图

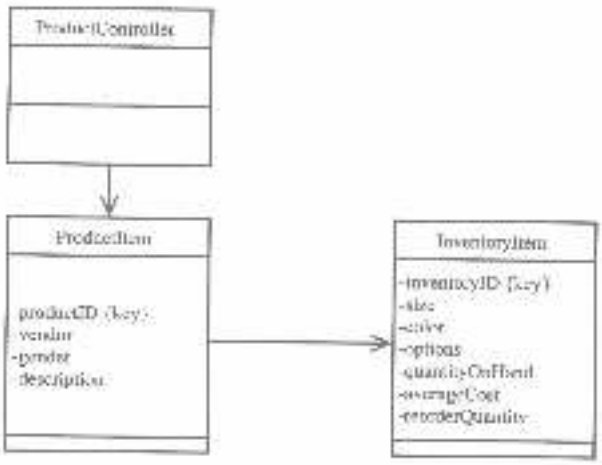


图11-16 "维护产品信息"用例的初步设计类图

回忆一下第7章，系统顺序图是在完整的用例描述和用户需求的基础上得到的。如图

11-15所示，用户输入productID检查该产品是否已经存在。RMO系统使用的是智能的productID，即组成productID的数字自身包含了生产线信息。productID并不是随机生成的，而是根据一定的规则生成的。

如果输入的productID对应的产品不存在，职员则需要输入创建一个产品所需的其他信息。这步完成后，职员再输入该产品各种型号和各种颜色的库存信息。不需要输入InventoryID，因为RMO系统会自动生成。当所有的库存信息输入完毕后，需要发送一个结束消息，告诉系统信息已经输入完毕。

在设计初步顺序图之前，我们先列出将会用到的输入消息和对象，如图11-17所示。根据前面所述，本例只需3个对象和4个输入消息。下一步需要接收消息并确定需要哪些内部消息。



图11-17　“维护产品信息”用例包含的对象

图11-18所示为设计结果。为了更好地理解该图，我们单独讨论每条消息。第一条消息checkForExisting(productID)，首先被ProductController接收，随后转发给ProductItem。返回给职员的结果是productID对应的产品是否存在。根据前面的假设，此时我们并没有包含异常处理。如果输入的productID已被使用，职员可以指定一个不同的号。

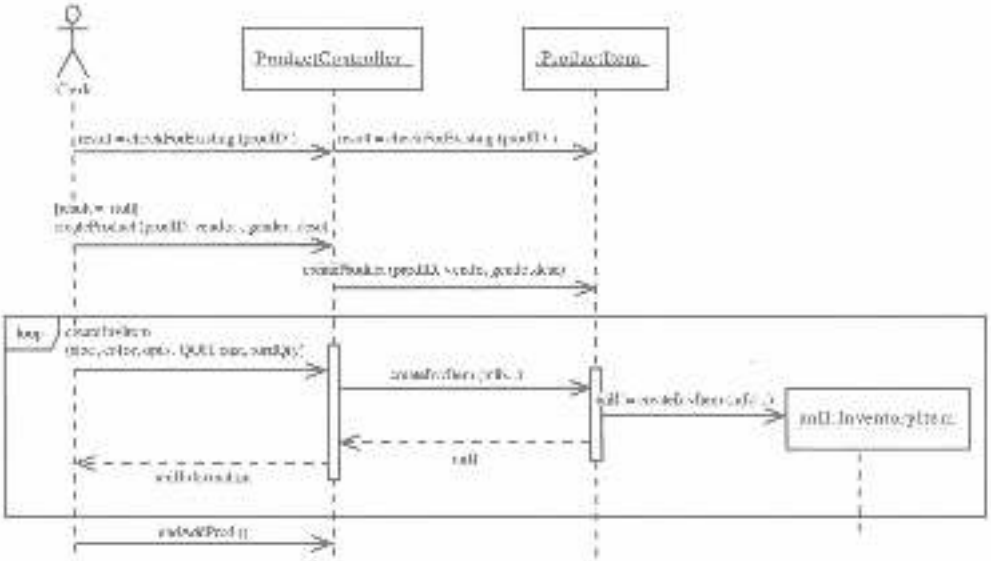


图11-18　“维护产品信息”用例的初步顺序图

第二条消息[result=null]createProduct(...)，演示了真/假条件的用法。这条消息只有在第一条消息返回空，即没有找到productID对应的产品时，才被发送。这条消息首先被控制器接收，然后转发给ProductItem类以创建新的ProductItem对象。

消息createInvItem(...)被包含在一个循环符号中。表示循环有两种方式，一种是在消息本身上面加一个“*”号，另外一种是用带边框的盒子。本例中带边框的盒子会更加清晰，因为该循环中包含多条消息。在:ProductController对象上，我们使用了激活生命线，用以显示该对象的方法的执行时间。:ProductController将消息转发给:ProductItem，它又发送消息激活:InventoryItem的构造器。

在这一系列的消息中，有两个新概念需要注意。第一个出现在:ProductItem和:InventoryItem之间，前者发送给后者的消息并没有指向生命线，而是指向了表示类的矩形框。这是为了强调这条消息激发了构造器方法。但是发送给:ProductItem的消息为什么没有直接指向表示类的矩形框呢？原因在于，是否发送激发构造器的消息依赖于第一条消息的结果。将消息指向矩形是可选的。

另外就是为新建的:InventoryItem对象取了一个名字anII。之所以这样做，是因为我们需要返回新对象的引用。给该对象取名后，:ProductItem根据名字将新对象的引用返回给:ProductController，:ProductController再返回给职员。

设计中还有一个问题：应该由谁激发:InventoryItem类的构造器函数？换言之，应该由谁负责创建新:InventoryItem对象。本用例中有两种选择——:ProductController对象或者:ProductItem对象。因为库存信息完全依赖于产品，即没有产品就不会有库存信息，所以让:ProductItem对象负责创建:InventoryItem对象是一个不错的选择。而:ProductItem不依赖其他任何类，所以可以让:ProductController负责创建新的:ProductItem对象。

最后一条消息endAddProduct(...)，只是用来表示用例已经结束，因此只需发送给用例控制器。

上面的两个例子讲述了为两种类型的用例进行初步设计的过程。这两种类型的用例分别是：一个叫“查询可用条目”的查询用例，一个叫“维护产品信息”的更新用例。两个用例都不复杂，却说明了一些重要的设计概念。在接下来的一节中，我们还将使用这两个用例来演示如何进行多层设计。

11.4 多层设计

初步顺序图只专注于域层上的类。然而，如前文所述，在系统设计过程中，我们还需设计用户界面类和设计数据访问类。本节就要扩展图11-14所描述的设计，将其变成一个多层的设计，包括一个可视层和一个数据访问层。我们先讨论数据访问层。

实践指导

添加数据访问层和可视层前，应确保域层的用例设计已经固定。 ■

11.4.1 设计数据访问层

职责分离准则是数据访问层设计背后的激发因素。较小的系统只有两层，即可视层和业务逻辑层。在面向对象的两层设计中，访问数据库的SQL语句嵌在业务逻辑层中。这意味着SQL语句被包含在问题域类的方法中。然而，对于更大、更复杂的、具有三层的系统来说，所创建的类的唯一任务就是执行数据库SQL语句，得出查询结果以及向域层提供信息。随着硬件及网络的日趋复杂，支持多层网络的多层设计诞生了。在多层设计中，数据库服务器位于一台机器上，业务逻辑作为一个服务器上，而用户界面在客户机的桌面上。这种新的设计思路不仅能提高系统的健壮性，而且能提升系统的灵活性。

职责分离：将系统划分成多个具有相似功能的类集的设计准则。

在第5章中，我们学习了如何建立域模型，从而描述事物，或者说是需要维护信息的实体。这个域模型有两个作用。首先，当然是为了开发新系统的数据库。第12章讲述了如何使用域模型来设计数据库。第二个作用就是确定组成新系统的内部类。显然，数据库表格与设计类之间有着密切的关系，因为它们都来自于同一个域模型。

在有关数据库的课程中，我们学习了如何使用结构化查询语言（SQL）在相关数据库中访问表格。在数据库里执行SQL语句就能用程序访问数据库的一个记录或者一组记录。面向对象程序在使用数据库时出现的问题之一就是编程语言与数据库SQL语句之间微小的差异性。举个例子，在数据库中，表格之间是通过使用外键连接起来的。例如，Order表中有CustomerID列，使得Order和Customer可以关系连接的形式关联在一起。然而在面向对象编程语言中，这个导航是反过来的。Customer类可能含有一组引用，指向计算机内存中正由系统处理的Order对象。换句话说，设计类没有外键。

面向对象编程语言与数据库SQL语言之间的差别从一定意义上决定了多层次设计的趋势。如果定义不同的类来访问数据库，得出利于计算机处理的数据形式，那么设计、编程及维护就变得更加容易了。定义不同的类，让它们各自完成自己的主要任务，而不是把业务逻辑与数据访问逻辑混淆起来。这是对类具有单一职责和高内聚度这个优秀设计准则的典型应用。

在这一章中，我们选择了一个简单的方法来完成设计，目的是教授基本的思想，而不要纠缠于数据库访问的复杂性中。假设每个域对象在相关数据库中都有一个数据表，使业务逻辑层与数据访问层关联起来有多种方法，不同的方法会导致不同的设计。一种方法是通过业务对象来创建与自己不同的业务对象。在每个业务对象的内部调用数据访问对象，从而获取必要的信息完成实例化工作。另外一种方法是给数据访问对象发送消息，然后数据访问对象读取数据库并完成新对象的实例化工作。两种方法都可行，并且都不失为好的设计。

图11-19展示了上面讲述的两种实例化对象的方法。在a部分，:Controller对象调用:Customer对象的构造器并给它传递一个custID参数，:Customer对象调用:CustomerDA对象并把自己的引用aC传递给它，然后:CustomerDA对象读取数据库，并填充aC相应的字段。在b部分，:Controller对象给:CustomerDA对象发送一个消息，给它传递一个键值或其他字段的值，:CustomerDA对象根据传递给它的值访问数据库，然后调用:Controller对象的构造器方法，并把实例化对象需要的数据作为参数传递给它。图中使用缩写的“info...”来表示这些消息。

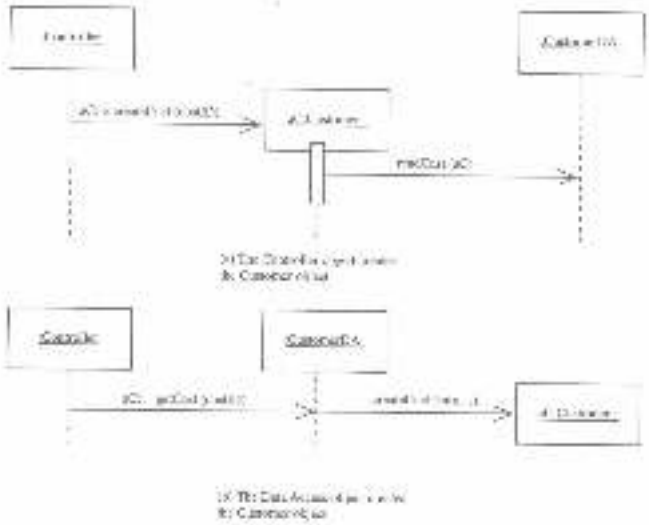


图11-19　访问数据库和实例化对象的两个方法

11.4.2 “查询可用条目”用例的数据访问层

要设计数据访问层，我们就不能继续假设对象是在内存中自动生成的了。现在，我们抛弃前面提到的内存假设。在用例设计里，我们必须添加额外的消息以从数据库获得信息来初始化对象的属性。我们采用图11-19中b部分所示的方法。

图11-20是包含所有数据访问类的用例设计图。因为最初的图中只包含4个问题域类，所以此处需要添加4个访问数据库的类。这么多的类和消息，可能会让人望而生畏。但仔细观察，你会发现数据访问中很多都是重复的模式。

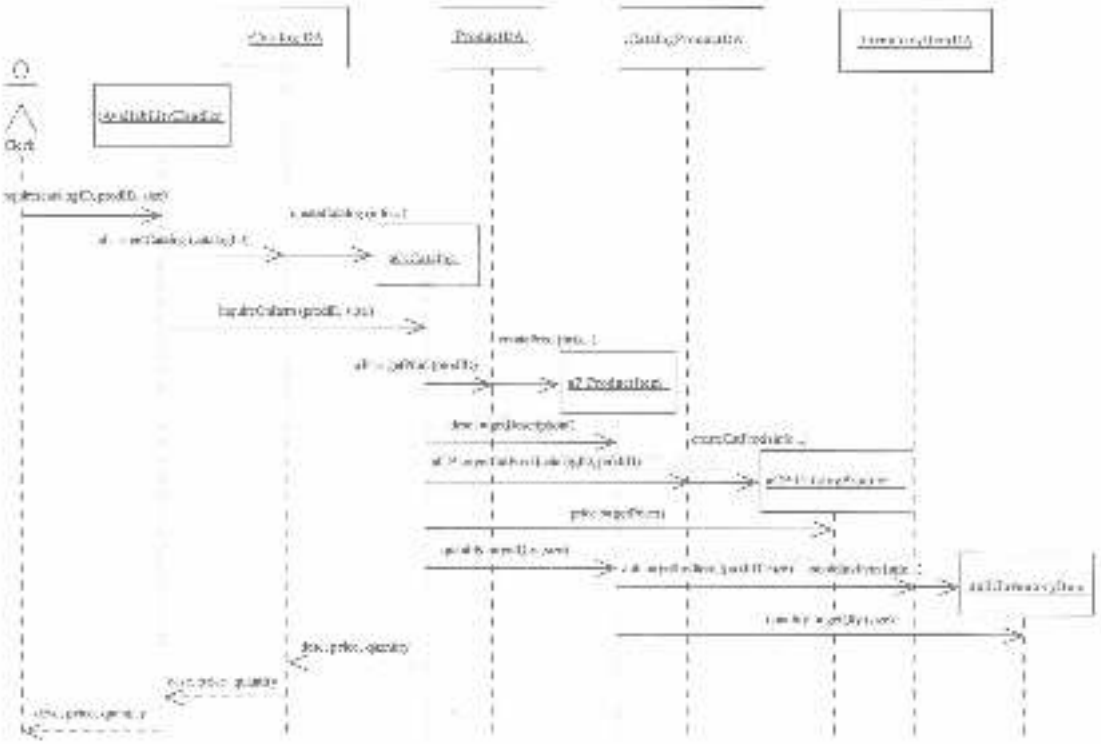


图11-20 “查询可用条目”用例的业务类和数据访问类

参考图11-14，该图是带有问题域类的初步顺序图。:AvailabilityHandler对象将消息inquireOnItem()转发给:Catalog对象。:Catalog对象从其他类中收集需要的信息，并返回给控制器。与上图唯一的不同在于，每个问题域对象在提供信息之前，必须是在内存中创建的，并且在创建过程中使用了来自数据库的数据。:AvailabilityHandler对象调用:CatalogDA对象，:CatalogDA对象读取数据库并创建:Catalog对象，同时返回一个该对象的引用给:AvailabilityHandler对象。

:Catalog对象的职责是收集相关信息，并把这些信息发送给数据访问对象:ProductDA和:CatalogProductDA，这两个对象负责初始化:ProductItem对象和:CatalogProduct对象。这些对象被创建后，会有消息发送给它们以搜集信息。:InventoryItem对象上也发生类似的事情，只是:ProductItem对象负责创建它们并从它们身上收集信息。

在设计过程中，有一个重要的问题需要考虑，即消息的源对象对消息的目的对象必须具有导航可见性，这样消息才能被正确发送。我们假设数据访问类具有全局的可见性，这在图中并没有显示出来（在编写类的过程中，可能需要借助工厂模式或单件模式以确保这种可见性）。问题域对象被创建后，要把该对象的引用返回给那些需要访问该对象的对象。例

如，:Catalog对象通过返回的id来引用:ProductItem对象。仔细观察图11-20，你会发现发送信息给其他对象的对象首先必须具有目的对象的导航可见性，这是设计中需要牢记的一点。

11.4.3　“维护产品信息”用例的数据访问层

图11-21是“维护产品信息”用例域层的扩展，它包含了数据访问层。第一条消息有微小的变化，它代表请求一个特定的数据，这里请求的是description信息，而不是请求一些较泛泛的信息。一个请求发送给:ProductDA对象，该对象访问数据库并返回请求的信息。在这个例子中，并没有创建:ProductItem对象，只是返回description信息。

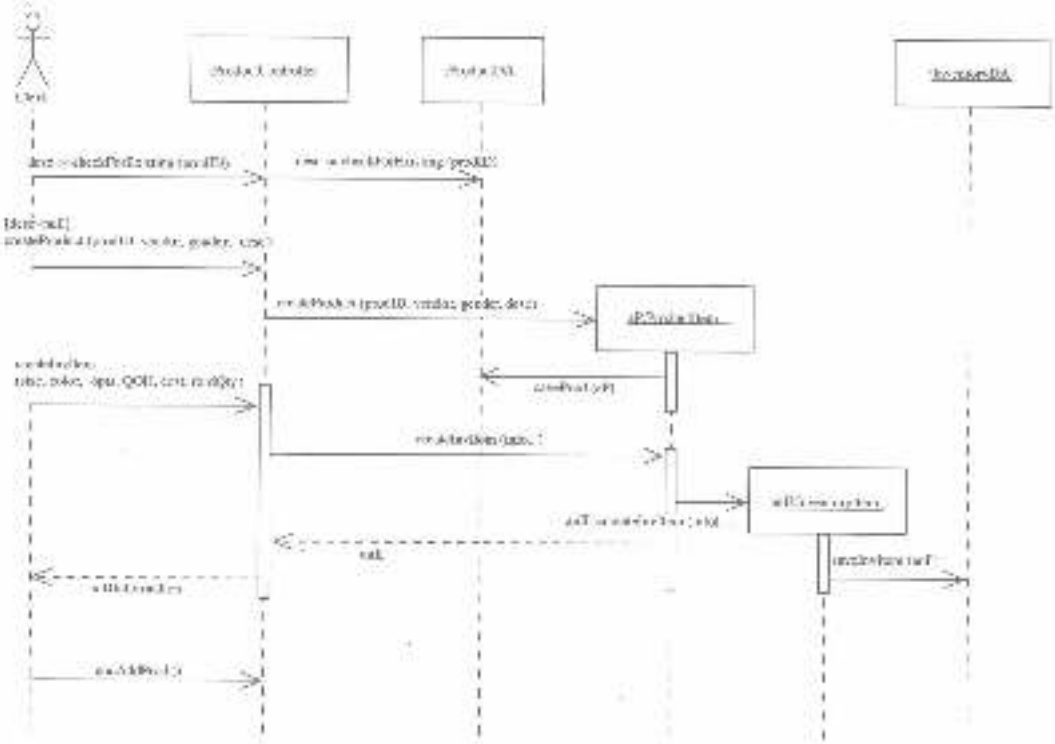


图11-21　“维护产品信息”用例的业务类和数据访问类

接下来的两条输入消息输入创建对象需要的信息。这个用例与上个用例不同，它利用输入的数据而不是来自数据库的数据来创建对象。对象首先被创建，然后数据访问对象将它们包含的信息存入数据库中。上个用例中域对象和数据访问对象之间的交互过程正好相反。发送给数据访问对象的消息是:ProductItem对象和:InventoryItem对象构造器的一部分。对象矩形与激活生命线相连表示活动和消息是构造器的一部分。

最后一条输入消息endAddProduct，只是用来表示用例的结束。因为数据已经存入数据库中，所以它是可选的。另一种选择是在没收到结束消息前，不把数据存入数据库中。换言之，首先创建:ProductItem对象和:InventoryItem对象，然后等收到endAddProduct消息时才把其数据存入数据库中。

正如前面所述，需要将数据存入数据库的域对象又叫持久对象，即使关闭计算机后，这些对象的状态依然存在。当然，关闭计算机后，数据访问对象便不再存在了，但它们传递的数据必须持久保存。因此，持久类用来指那些需要永久保存的对象或类。

11.4.4 设计可视层

多层设计的最后一步是添加可视层。对许多用例来说，可视层仅由一个简单的用户界面窗口组成。一些用例会稍微复杂一些，它们需要多个窗口来输入或显示相关数据。此处我们不深入讨论有关窗口类的概念，简单讨论用户窗口便足够了。显然，用户界面的详细设计会更加复杂。有关窗口类的详细讨论，如窗口控制和布局等，请参阅第13章和第14章。

在早期的交互系统和图形用户界面中，工具开发人员开发了许多语言和工具，这些语言和工具使利用图形用户界面（例如窗口和按钮）设计系统变得很简单。一些早期版本的语言，例如Visual Basic、Delphi和PowerBuilder，使建立交互的、事件驱动的和图形化的系统简单化。然而，在这些语言中，编程逻辑往往附着在窗口及其他图形组成部分上，如果将这些系统移植到其他的环境中（例如基于浏览器的系统），那么设计者必须重写整个系统。事实上，按照这种方法设计的系统很好地阐述了当与设计准则（高内聚类和职责分离）发生冲突时会出现的问题。如果一个类包含用户界面功能以及业务逻辑，并且二者混在一起，那么这个系统的升级维护就变得更加困难。

实践指导

不要将业务逻辑加入可视层类中。

随着面向对象编程的不断流行，设计工具也集成了面向对象语言编程和图形接口，设计系统变得更加容易。这些系统通常由几个具有单一职责的部分组成。除了编辑输入数据外，用户界面类不必包含业务逻辑，设计者可以建立多层系统，它具有更好的强健性，且更容易维护，并符合优秀设计准则。像Java和Visual Studio.Net之类的设计工具不仅能轻松地建立图形用户界面，而且能够创建复杂的问题域类。

在用例的对话设计过程中，作为可视层的窗口类被添加到顺序图中。在一般情况下，用一个输入表单来接收用例的所有消息。如果这些消息互不相同，那么每个消息可能需要自己的输入表单。然而在大多数的情况下，对于用例里所有相互关联的消息来说，一个输入表单就足够了。显然，从外部参与者发出的每个消息都要以某种方式送入到系统里，然后输出消息必须被显示出来。一种方法是通过一个窗口类来接收输入数据和显示输出数据。让我们再回到“查询可用条目”用例。图11-22描述了一个名叫:ProductQuery的窗口类，其可供职员访问。

向顺序图中添加用户界面类比较简单。在早期的设计过程中，如图11-14所示，我们假设一个消息直接从外部的参与者发给:AvailabilityHandler对象，而在现实中，数据通过键盘输入电子表单，接着用户界面窗口接收这些数据，形成一条消息，并将这条消息发送给:AvailabilityHandler对象。这样，要想添加用户界面对象，只需将它们放在参与者、职员和域对象之间，并插盖合适的信息就可以了。图11-22显示了给顺序图添加用户界面对象的过程。

添加用户界面类看起来似乎很简单，但实际上它的完成需要结合用户界面表单的详细设计（参见第13章）。例如，对于“查询可用条目”用例，是用两个表单——一个用于输入数据，一个用于显示数据呢？还是用一个表单——既用于输入数据又用于显示数据呢？在这个设计中，我们假设只有一个表单。在用户看到一个或两个备选方案前，设计都是暂定的。用例设计最初只关心业务逻辑层和数据访问层，而用户界面层在进行详细的用户界面设计时完成。

只在顺序图的基础上进行用户界面设计通常不是很有效。为什么呢？原因在于，用户在看见和接触最终产品前，他们也不清楚自己到底需要什么。因此，构建系统，尤其是用户界面部分，最有效的方法是让用户积极参与产品原型、模型和脚本的开发和使用。原型和模型相结合的方式是最有效的。

下面我们来讨论“维护产品信息”用例的可视层。我们刚刚提到过，确定用户界面类是

用户界面设计的一部分，该用例有4组消息：查询存在的产品、输入产品信息、输入库存信息和结束消息。我们假设用户选择使用同一个表单用于查询和输入。当输入的数据与系统已有的数据重复时，会弹出一个提示窗口；否则，表单继续接收数据。对于库存信息，一个表单用于输入，一个表单用于显示结果。最后一条消息endAddProduct()对应产品信息表单上的一个按钮，并非一个表单。

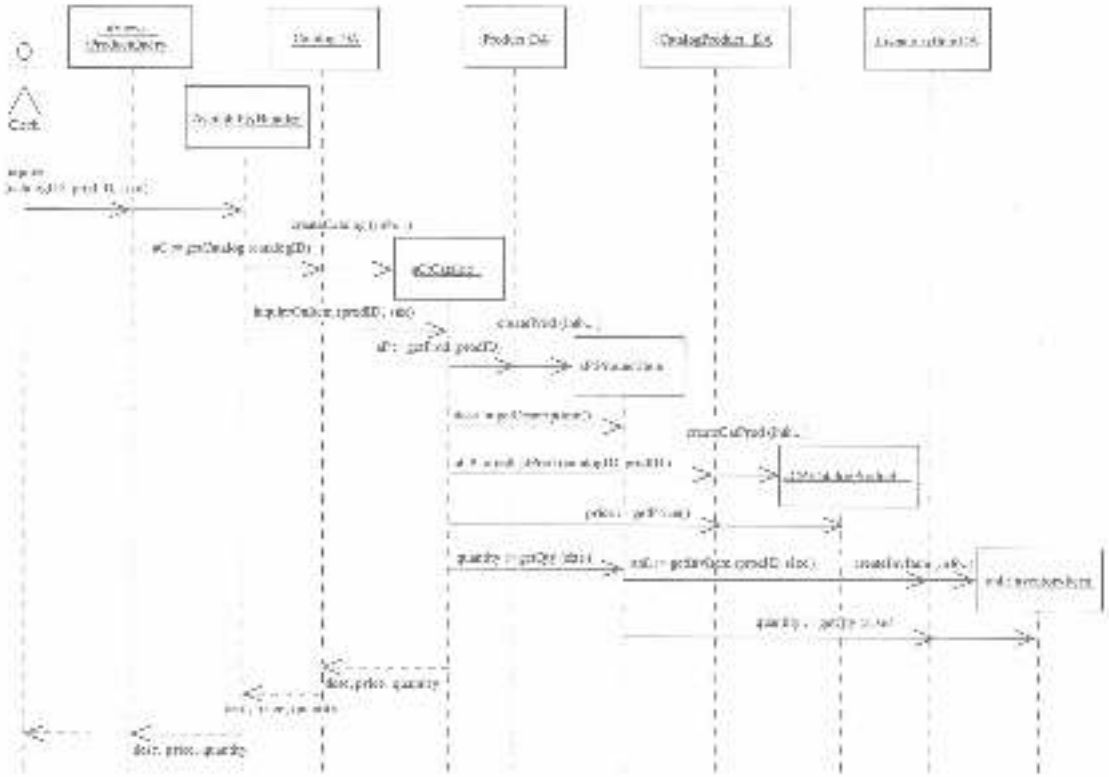


图11-22　含有可视层的“查询可用条目”用例

图11-23是图11-21添加可视层后的模样。基于用户界面设计的结果，添加了一些类。在该设计中，有两个用于输入数据和显示数据的窗口，以及一个消息窗口。

对于这两个用例，我们首先定义了业务层，然后是数据访问层，最后是可视层。在一般情况下，都是先设计业务层，但对于数据访问层和可视层的顺序，不同的开发人员有不同的选择。到底谁先谁后，常取决于日程安排和参与用户界面设计的用户数目。

下面我们的注意力将转向一个更复杂的用例——一个带有多条输入消息，并且这些消息会创建新对象的用例。同样，我们需要分步实现，首先在域对象的基础上设计用例，然后添加其他层的对象。

11.5　用协作图设计

协作图与顺序图都属于交互图，它们捕捉同样的信息。无论使用协作图还是顺序图，设计的过程都是一样的。使用什么模型来设计主要取决于设计人员的个人偏好。许多设计人员更喜欢使用顺序图来进行设计，因为用例描述和对话设计都是按照顺序步骤来进行的。而协作图则从更强调耦合的角度来审视用例。本节将对协作图做简单介绍。

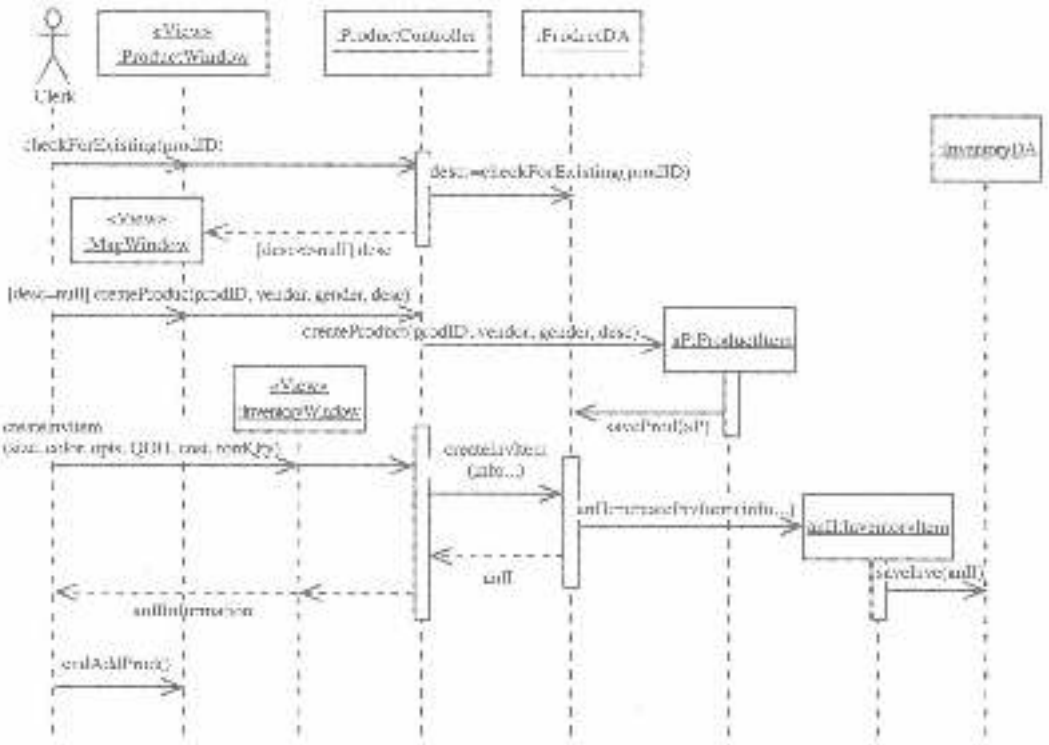


图11-23 含有可视表的“维护产品信息”用例

协作图所使用的有关参与者、对象和消息的符号与顺序图是一样的，但生命线和激活生命线不再出现，但使用了一个不同的符号——连接符号。图11-24描述了在大多数协作图中都会出现的4种符号。

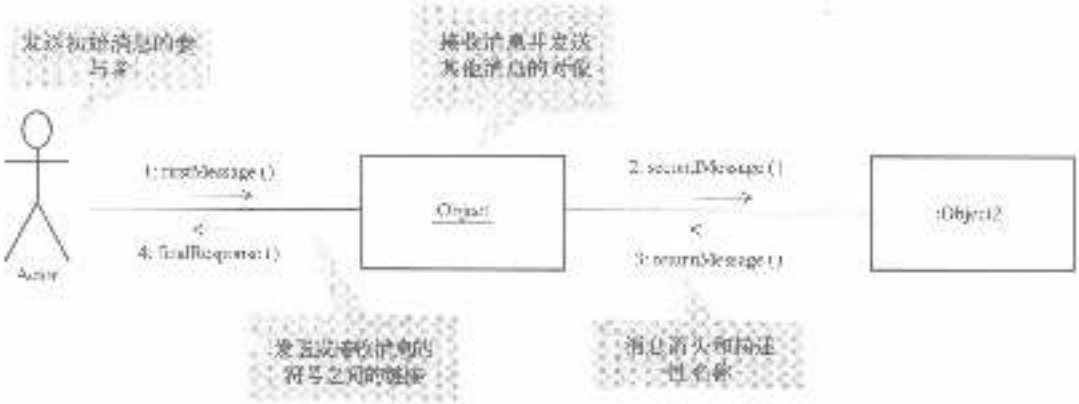


图11-24 协作图的符号

在协作图中，消息描述器（discriptor）的格式与顺序图中的格式稍有不同，由于没有生命线来表示场景中时间的流逝，因此，每个消息都按照顺序编上号来说明它们之间的次序关系。协作图中消息描述器的语法如下：

[true/false条件]序列号：返回值：= 消息名（参数列表）

如图11-24所示，冒号总是跟在序列号的后面。

在对象之间或在参与者与对象之间的连线表示链接。在一个协作图中，链接表示两个对象共享一个消息——一个发送消息，一个接收消息。连线本质上仅仅用于传递消息，所以你也可以把它们想象为用于传输消息的线路。

链接：协作图中用于在对象之间或在参与者与对象之间传递消息的符号。

图11-25是“查询可用条目”用例的协作图。该图仅仅包含域模型对象，而不含有可视层和数据访问层。使用协作图来进行多层设计与使用顺序图一样高效。

消息上的数字代表了消息的先后顺序。注意，编号为5和5.1的消息。这种使用“.”来表示层次性的方法只有在一些消息依赖其他消息时才使用。在这个例子中，最初的消息5: quantity := getQty(size)被发送给:ProductItem，然后:ProductItem转发一个相似的消息5.1:quantity :=getQOH()给:InventoryItem。第二个消息是第一个消息的结果，因此它就被编号为5.1，来表示与第一个消息的从属关系。有时候，缺乏经验的设计人员搞不清楚什么时候将消息编号为从属关系，什么时候将它们编成同级关系。举个例子，你可能认为这一系列的消息都是依赖于第一个发送的消息的，因此它们都应该从属于第一个消息。一个好办法是先为消息列个大纲，就像我们在写文章前列个大纲一样。大纲中处于同一级的消息都应该编为主要的消息。低级别的消息对应文章大纲中相互依赖的段落和标题。消息的层数（或者叫深度）可以根据所要描述的依赖性确定。

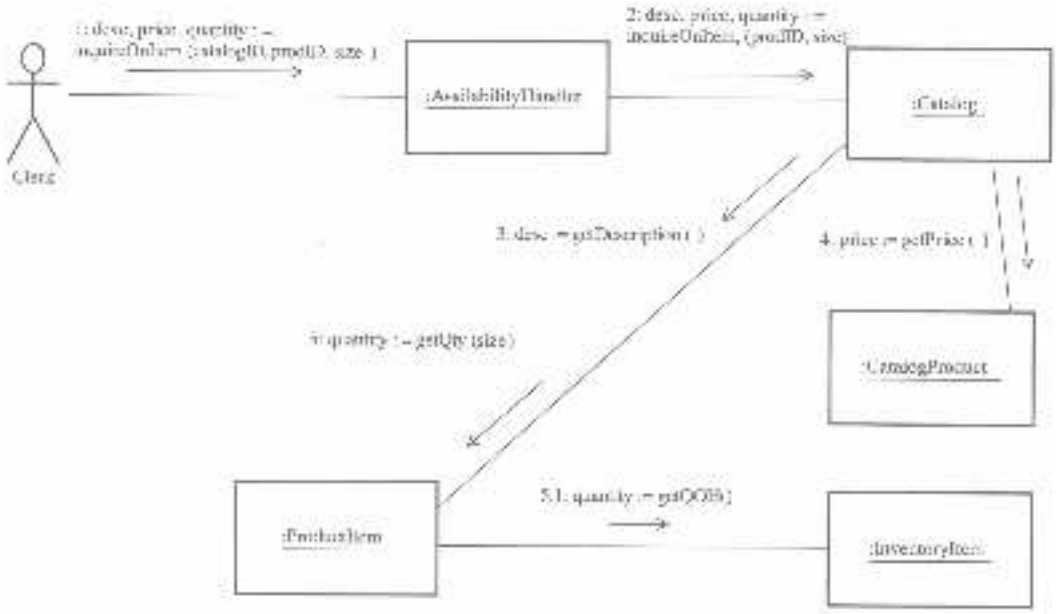


图11-25　“查询可用条目”的协作图

将协作图与顺序图相比较，很容易就可以看出，协作图主要关注对象本身，而绘协作图就能有效地统观协同工作的对象。同时你会发现，在顺序图中确定消息的顺序比较困难，你必须找到消息的编号才能知道它们的顺序。另外，要想很快地统观互相协作的对象，那么协作图就是一个很好的方法。

许多设计人员用协作图来草拟出解决方案，如果用例比较小而且不是很复杂的话，那么一个简单的协作图就足够了。但是，对于比较复杂的情况，就需要顺序图来可视化消息的流向及顺序了。通常，在同一个说明里面会夹杂两者：一些用例用协作图来描述，而另一些用例用顺序图来表示。系统开发人员应该学会使用这两种图形。

我们也可以使用一个更加简化的协作图。图11-7显示了用来表示对象的两个符号。图标样子的符号表示对象名。而符号本身说明了其构造型。不同的图标可以表示可视层、控制器、域层和数据访问层。图11-26就是“查询可用条目”用例多层设计的一个例子。这种画法不用消息就能表示互相协作的对象，或者和消息一起为完整的协作图提供速记符号。这些图标也可以用在顺序图中作为构造型的速记符号。

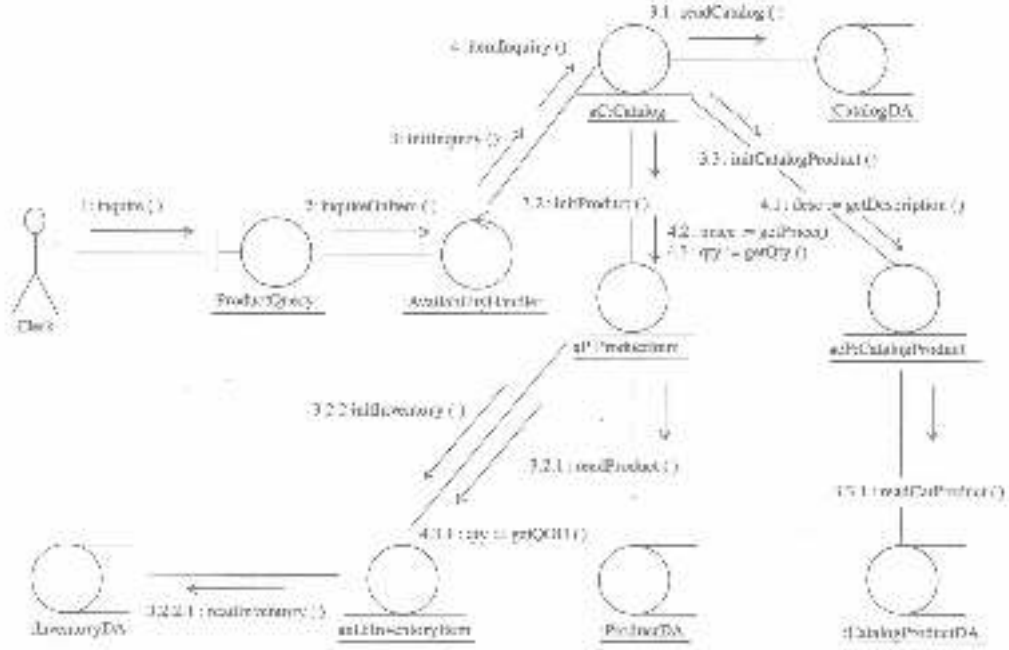


图11-26　使用图标符号的“查询可用条目”用例

11.6　更新设计类图

现在每层都可以开发一个设计类图。在可视层和数据访问层中，一些新的类必须要清楚。域层也有一些为用例控制器而添加的新类。

在图11-11中，我们为域层设计了一个初步的设计类图。那个时候没有开发出任何的方法特征。既然已经创建了顺序图，方法信息就可以添加到类里面去了。而且，在顺序图建立的过程中，导航箭头也要同时更新。

首先，我们添加方法特征。大部分类中主要有三种方法：(1) 构造器方法；(2) 数据读/写方法；(3) 用例特定的方法。构造器方法会创建新的对象实例。数据读/写方法读取或更新属性值。由于每个类都有一个构造器，并且大多数都使用数据读/写方法，因此把这些方法的特征包含到设计类图中是可选的。第三种方法一定要包含在设计类图中。下面的几个例子覆盖了以上三种方法，并展示了如何添加方法特征。

就像在顺序图中一样，每一个消息都有源对象和目的对象。如果一条消息发给了某个对象，那么这个对象必须准备好接收这个对象并初始化一些行为。这个过程无非就是调用对象上的某个方法。换句话说，顺序图上出现的每条消息都需要目的对象的一个方法。事实上，消息的语法很像方法的语法。因此，给设计类图添加方法特征的过程就是浏览每一幅顺序图并找到发给类的消息的过程。每条消息对应着一个方法。

我们通过一个基于ProductItem类的例子来说明如何添加方法特征。该例子使用的带有域模型类和数据访问类的顺序图是图11-20和图11-21。既然是为域模型类定义方法特征，那么我们就在添加多层设计之前使用顺序图。

图11-20中有3条消息发送给ProductItem。第一条是一个构造器，另外两条——getDescription()和getQty()用于请求信息。大多数设计类图都不列出构造器。众所周知，构造器是必需的，而在图中包括它们常常使图显得拥挤。第一条消息返回一个字符串，第二条消息返回一个整数。相应地，我们需要为这两条消息添加方法特征。图11-21中有一条请求ProductItem对象创建InventoryItem对象的消息。ProductItem对象负责创建InventoryItem对象，所以它接受请求并创建一个新的InventoryItem对象。因此，我们需要为ProductItem对象添加一个叫做createInItem()的方法。基于顺序图确定所有的方法特征后，最终结果如图11-27所示。



图11-27　带有方法特征的ProductItem类的设计类图

对域层中的每个类及用例控制器类执行上述同样的操作。图11-28所示的是添加方法特征后完整的域层对象，该图十分出色地记录了设计类，并将作为系统实现的蓝图。

我们还需要给域层类添加两个用例可视，加进附加的导航箭头可以对用例控制器可见的类进行记录。除此之外，在初步类图中定义的导航箭头对于这两个用例来说足够了。在进一步的用例并发中，我们会添加更多的导航箭头，如那些指向Order类、Shipment类和ReturnItem类的导航箭头。

11.7　包图——将主要部分结构化

UML中的包图是一个高层次的图。它使得设计人员可以将相关组中的类联系起来。前面几节讲述了三层设计，包括可视层、域层和数据访问层。在交互图中，每层中的对象都画在同一图中。但是，有些时候，设计人员需要记录不同层次中对象间相互关系的相同点和不同点——可能根据分布的处理环境不同对对象进行分组。将每层都表示为一个单独的包就能够捕捉到这些信息。图11-29分析了这些层次是如何记录的。

表示包的符号是带标签的矩形。包名通常显示在标签上，但是，对于高层次的视图，如果包内部不显示任何细节信息，那么包名也可以放在矩形框的内部。在这种情况下，属于这个包的类也放在矩形框的内部。

类根据它们所属的层被放在合适的包里面。在相关图中建立类的时候，它们就与不同层联系起来。为了创建包图，我们只需从每个用例的设计类图和交互图里提取信息。图11-29只是包图的一个部分，因为这些包仅包含本章所创建的用例交互图中的类。

包图使用的另一个符号是虚线箭头，它表示依赖关系。箭头的尾部连接着有依赖性的包，而箭头连接着被依赖的包。依赖关系在包图、类图甚至交互图中都有所应用。可以这样理解依赖关系：如果其中的一个元素发生了变化（被依赖部分），那么另一个元素（依赖部分）也一定会发生变化。依赖关系可以存在于包与包之间，或者包中的类与类之间。图11-29意味着可视层中的两个类依赖域层中的类。因此，如果Order类发生变化，那么OrderWindow类也应该去响应Order类的变化。然而，反过来就不一定了。可视层中的变化往往不能传递给域层。

依赖关系：包图、类图以及交互图中元素之间的一对关系，用于说明系统中哪些元素影响其他元素，使得设计人员可以跟踪变化的传递效果。

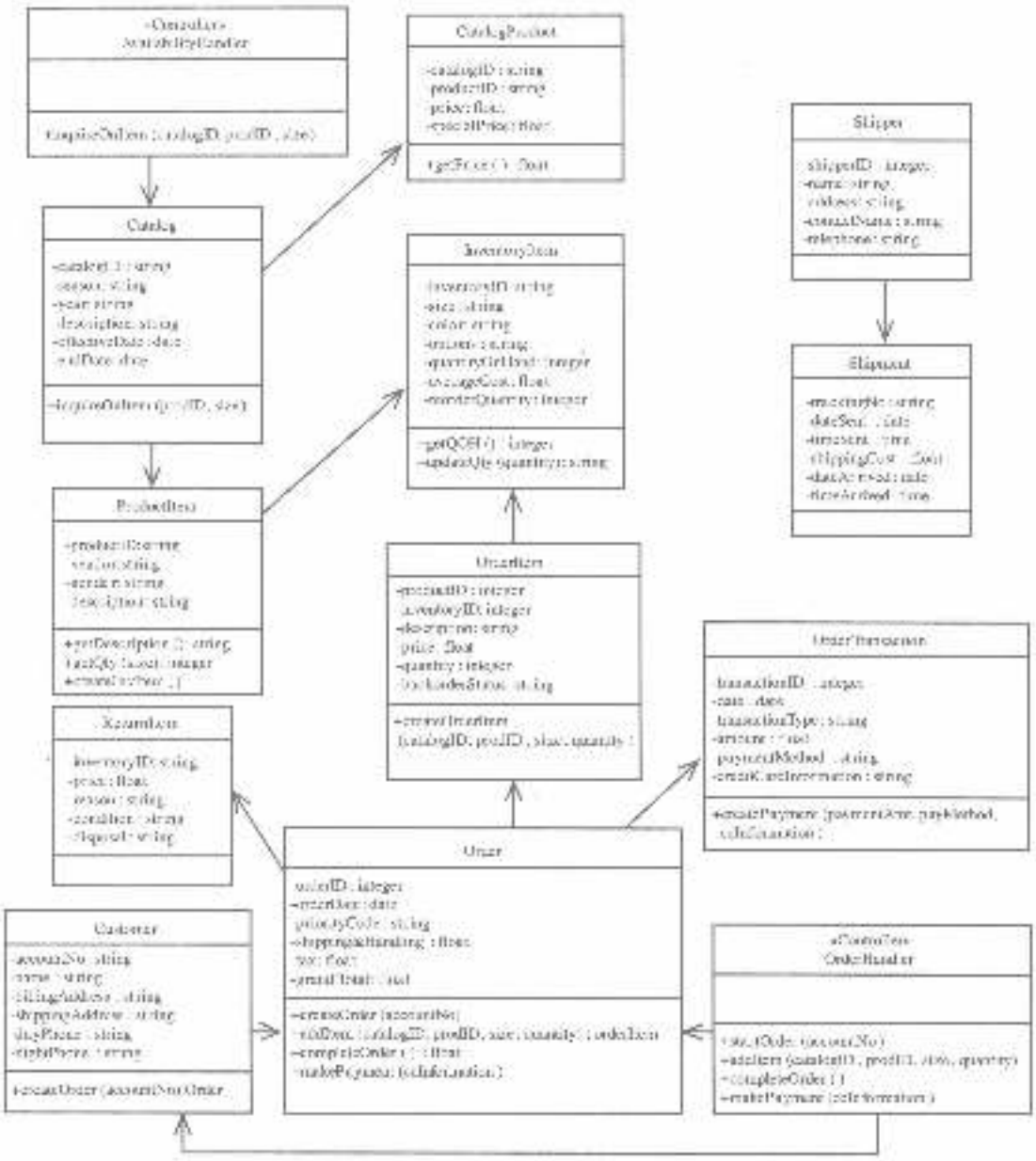


图11-28 更新后的域类设计类图

有关依赖关系的两个例子如图11-29所示，第一个是类与类之间，已经讨论过。另一个例子不是很详细，描述的是包与包之间的依赖关系。图11-29说明可视层和域层都依赖于数据访问层。因此，数据结构的变化，体现为数据访问层的变化，通常会引起域层和可视层的变化。

包图也能通过嵌套来描述不同层次的包。图11-30说明包和它们所包含的一些类，都是订单输入子系统的一部分。RMO系统可以分割成一些子系统。一种记录这些子系统的方法就是使用包图。这种方法的好处是，可以将不同的包交由不同的开发小组完成。依赖关系箭头可以帮助开发小组确定何时需要相互交流，以确保系统的完整性。

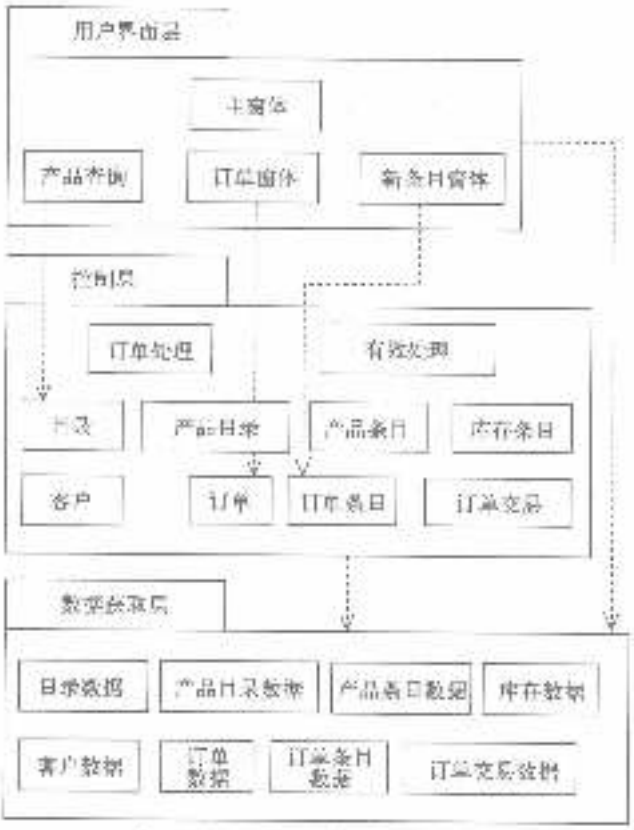


图11-29　RMO三层包图的部分设计

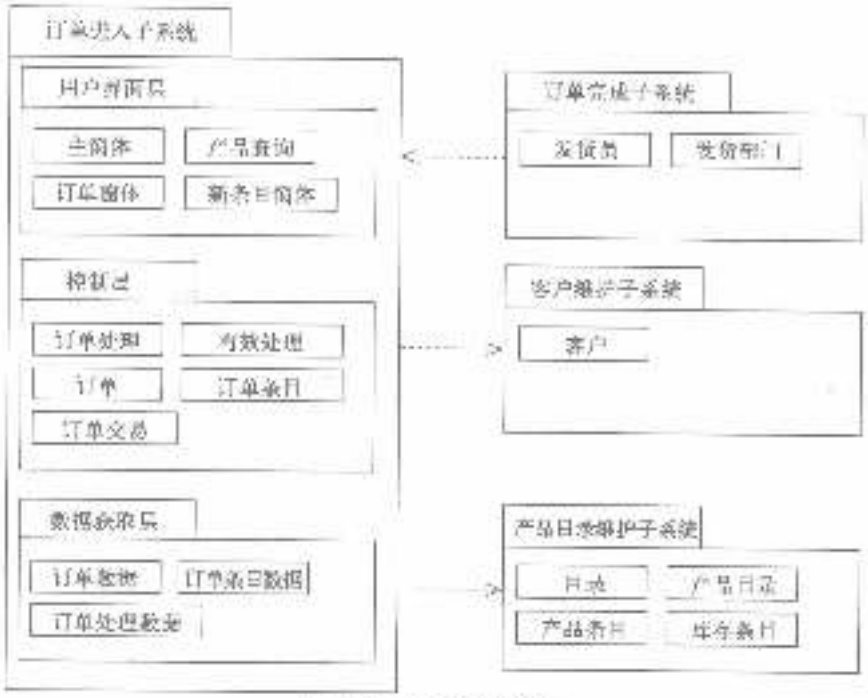


图11-30　RMO子系统包

如图11-30所示，Order Fulfillment子系统依赖于Order-Entry子系统，而Order-Entry子系统则依赖于Customer Maintenance子系统和Catalog Maintenance子系统。Order Fulfillment子系统可以使用在Order-Entry子系统中定义的Order类，或者发送来自Order-Entry子系统的消息。无论怎样，只要Order-Entry子系统中发生了变化，那么Order Fulfillment子系统就要做相应的修改。随着类被添加到单个包里，类中对象的使用就会决定依赖关系。举个例子，一旦类被分配给了包，那么依赖关系箭头也就被确定下来了。Order-Entry子系统显然要访问Customer类，因此它就依赖于Customer Maintenance包。

综上所述，包图可以用来表示各部分的相关性和依赖关系。在通常情况下，我们使用包图来关联类或者其他的系统组成部分，例如网络结点。前面的图说明了包图的两个应用：将系统分割成子系统，以及表示包的嵌套。

11.8 三层设计的实现问题

有了设计类图、交互图和包图，编程人员就可以开始创建系统的各个组成部分了。实现是指通过语言（Java或者VB.NET）编程来搭建系统。过去几年，人们开发出来功能强大的集成开发环境（IDE）工具来帮助编程人员搭建系统。这些工具包括：和Java有关的Jbuilder和Eclipse，和Visual Basic及C#有关的Visual Studio，和C++有关的C++Builder。这些工具从较高的层次上提供了编程支持，特别是在搭建可视层类如系统窗口以及窗口组件的时候更加明显。不幸的是，正是这些工具，使开发人员养成了许多不良的编程习惯。

轻松地建立图形用户界面窗体，以及代码自动生成所带来的便利，使得一些程序员将所有的代码都放在窗体里。每个窗体组件都自带几个事件，这些事件的代码是自动生成的。这样，程序员使用IDE就会很容易地建立一个窗体，工具自动生成类定义，而他们仅仅需要插入业务逻辑代码，不需要定义新类，也不需要其他的代码。许多这样的工具还含有数据库引擎。这样整个系统就可以只使用窗体类来建造。然而，这样走捷径以后是需要付出代价的。

这个方法存在的问题在于维护系统的困难性。图形用户界面类中散落的代码段很难定位和维护。而且，当用户界面类需要升级的时候，程序员也必须找到并升级业务逻辑。如果基于网络的系统需要升级到包括网络前端，那么程序员几乎要重建整个系统。或者，如果需要两个用户界面的话，那么所有的业务逻辑都需要编程两次。最后，如果不使用产生代码的工具，那么保持系统的通用性就是不可能的。随着新版IDE工具的发布，这个问题会进一步恶化，因为新版IDE与原有的IDE可能不兼容。许多程序员必须完全重写系统的前端，因为新版IDE工具产生代码的方式与原有的IDE不同。因此，我们建议未来的维分析员和程序员们在开发新系统的时候一定要遵循设计准则。

根据设计准则中对象职责部分，可以定义每一层的编程任务。如果按照这些规定来编写代码，那么新系统在其生命周期内就很容易维护。我们总结一下每层的主要任务。

可视层类完成以下任务：

- 展示电子表单和报告
- 捕捉输入，例如单击、滚动和键盘输入等事件
- 显示数据字段
- 接收输入数据
- 编辑并校验输入数据的合法性
- 将输入数据传送给域层类
- 启动与关闭系统

域层类完成以下任务：

- 创建问题域（持久）类
- 以适当的逻辑处理所有的业务规则
- 准备持久类以便数据库存储

数据访问层完成以下任务：

- 建立并维护数据库之间的连接
- 包含所有的SQL语句
- 处理结果集（SQL语句的执行结果），并赋给合适的域对象
- 适时断开与数据库的连接

小结

面向对象的设计在用户需求（表示为分析模型）与最终的系统（使用编程语言构建）之间架起了一座桥梁。用面向对象的方法来设计系统是一个技巧性很强的工作，它将分析模型转变为一组蓝图，程序员可以利用蓝图来编写代码。

设计是由用例来驱动的，即设计是在用例的基础上完成的。在设计过程中，两个主要模型是设计类图和顺序图。通过添加属性类型、可见性信息以及方法特征，域模型——域类图被转换成设计类图。顺序图是系统顺序图的扩展，通过驱动执行用例所需的内部进程来创建。顺序图明确了相互协作的类以及它们协作的方式，也明确了它们之间为了完成用例相互传递的消息。

为了得到一组正确的消息，并保证设计的优良性，必须应用一些面向对象的设计准则。这些设计准则包括封装、耦合、内聚、导航以及对象职责。封装是一个标准的面向对象准则，它确保数据字段位于正确的类中，以及有足够的方法来处理数据。耦合度准则会应用到所有类的集合中，表示类之间连接的程度。要成为一个优秀设计，连接程度越小越好。内聚度指的是每个单一类内部的属性，用来描述类的内聚程度。一个类如果含有一些方法，这些方法能够进行许多不同的处理过程，那么这个类就不满足内聚准则。换句话说，它发散了。导航指的是哪些类可以访问或者可以看见其他的哪些类。包含有太多导航连接的系统的耦合度就太强了。最后，对象职责准则能够帮助确定哪些类应该接收或者发送哪些消息。对象应该负责自己本身。但是它们也可能有其他的任务，那些其他的任务应该小心分配，这样系统才能坚持低耦合，类才是高内聚的。

一种设计可维护系统的方法是使用三层设计方法。三层设计，甚至是多层设计，根据类的主要任务把它们分成了不同的组。本章中讲述的三层设计由以下层次组成：可视层，由图形用户界面类组成；域层，由业务类组成；数据访问层，由能够访问数据库的类组成。对于系统来说，三层设计是一个既健壮又灵活的设计。

本章也讨论了使用协作图来完成设计的过程。协作图是顺序图的替代物，使用协作图还是顺序图主要是个人喜好问题。

最后，为了将组成部分，特别是类，分组成相关的项目，介绍了一种新的符号。包图由一个带标签的矩形框表示，矩形框的内部是分好组的类。包图就好像是拿起一个东西，将它们放入一个包或者容器中。包图的一个作用是将类分组成了系统，另一个作用就是可以显示出哪些类属于可视层，哪些类属于域层，哪些类属于数据访问层。

关键术语

activation lifelines	激活生命线
artifact	制品

boundary class(or view class)	边界类（或视图类）
class-level method	类级方法
cohesion	内聚度
control class	控制类
coupling	耦合度
data access class	数据访问类
dependency relationship	依赖关系
design patterns	设计模式
encapsulation	封装
entity class	实体类
indirection	间接
information hiding	信息隐藏
instantiation	实例化
links	链接
method signature	方法特征
navigation visibility	导航可见性
object responsibility	对象职责
object reuse	对象重用
overloaded method	重载方法
persistent class	持久类
protection from variations	变量保护
realization of use cases	用例实现
separation of responsibility	职责分离
stereotype	构造型
use case controllers	用例控制器
visibility	可见性

复习题

1. 哪三种模型在面向对象设计中应用得最多？
2. 为什么说设计是“用例驱动”的？
3. 有4个对标表者快捷方式用来描述不同类型的类，列举这4种图标，说出每一个都代表什么，并给出符号表示。
4. 列出出方法特征的元素。举出一个包含所有正确元素列表的方法特征。
5. 用来表示构造型的符号是什么？给出一个构造型类的例子。
6. 什么是导航可见性？它在UML中是什么样子？在编码中是如何实现的？
7. 什么是耦合度？为什么认为耦合度太强是不好的？
8. 低内聚类会出现哪些问题？
9. 什么是对象职责？为什么它在设计中是个非常重要的概念？
10. 用例控制器类的目标是什么？它代表哪一个设计准则？
11. 什么是设计模式？开发人员如何使用模式？
12. 什么是三层设计？在三层设计中，哪层层使用最多？
13. 为什么三层次设计是一个优秀的设计准则？

14. 进行三层设计的推荐方式是什么？或者说，层次是按照何种顺序来设计的？
15. 为了创建一个初步顺序图，必须遵守三个步骤。简要描述这三个步骤。
16. 简要描述顺序图与协作图之间的主要差别。
17. 描述在协作图中使用的消息符号。
18. 包图的目标是什么？使用了什么符号？举出一个例子。
19. 包图是如何说明依赖关系的？这意味着什么？
20. 列举可视层、域层和数据访问层中类的主要任务。
21. 基于Internet的系统与基于网络的系统有什么区别？

思考题

注意：练习1、2、3和4是基于第7章“思考题”中有关大学图书馆系统的练习1和2的解决方案而设计的。作为选择，你的老师可能会给你提供用例图和类图。

1. 图11-31所示为大学图书馆系统中“借书”用例的系统顺序图。完成以下习题：
   (a) 建立初步顺序图，它只含有参与者和问题域类。
   (b) 给(a)题中得到的图添加可视层和数据访问层。
   (c) 基于域类图和(a)、(b)两题的结果创建一幅设计类图。
   (d) 创建一幅包图，它描述了包含可视层、域层和数据访问层包的三层解决方案。

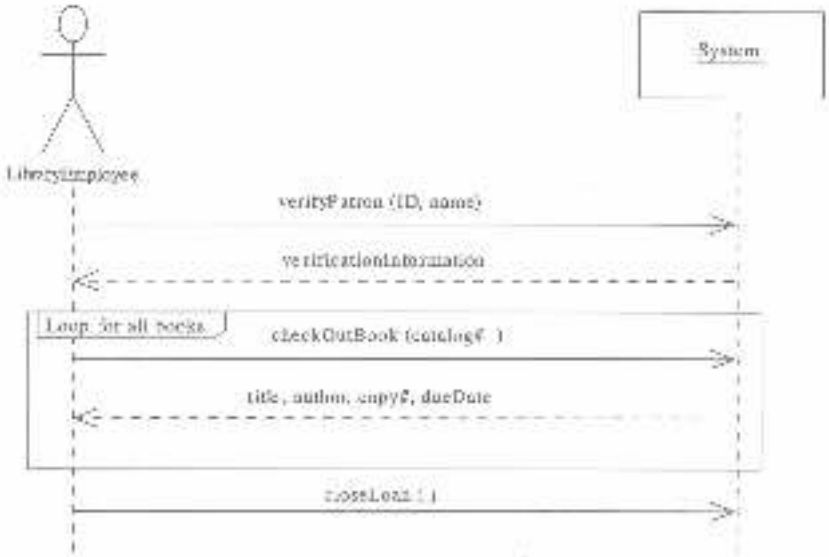


图11-31　“借书”的系统顺序图

2. 图11-32所示为大学图书馆系统中“还书”用例的活动图。完成以下习题：
   (a) 建立初步顺序图，它只含有参与者和问题域类。
   (b) 给(a)题中得到的图添加可视层和数据访问层。
   (c) 基于域类图和(a)、(b)两题的结果创建一幅设计类图。
   (d) 创建一幅包图，它描述了包含可视层、域层和数据访问层包的三层解决方案。
3. 图11-33所示为大学图书馆系统中“接收新书”用例的完整描述。完成以下习题：
   (a) 建立初步顺序图，它只含有参与者和问题域类。
   (b) 给(a)题中得到的图添加可视层和数据访问层。

(c) 基于域类图和 (a)、(b) 两题的结果创建一幅设计类图。

(d) 创建一幅包图，它描述了包含可视层、域层和数据访问层包的三层解决方案。

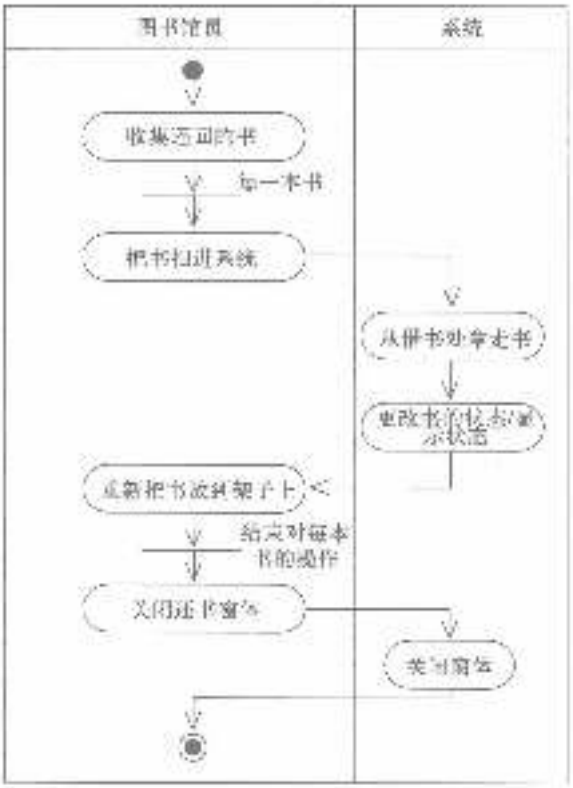


图11-32　"还书"活动主

用例名称	接收新书	
动员	接收新书	
触发事件	新购买的书籍到达	
简单描述	图书馆员计划采购一些新书，并发出了订单（关于这个用例），新书运来了。每本新书都被分配了一个图书馆目录号码。有些书只是已有书的副本，有些书是已有图书的新版本，有些书的书名是新的或者内容是新的。新书的信息被添加到系统里	
参与者	图书馆员	
系统相关者	图书馆员，图书馆管理员	
前提条件	无	
后续条件	存在书名，实物存本	
事件流程	参与者	系　统
	发发货单接收新书 对每本书都检查数目和目录号码，分配临时号码 3a. 如果是已有图书的副本，将图书馆总和目录号码输入系统 3b. 如果是已有图书的新版本，输入	 3a.1 准备号码更新目录，确实没有重复 3b.1 用新号码更新目录，确实没有重复

图11-33　"接收新书"用例的完全展开描述

事件流程	参与者	系　统
	图书情况、版本情况和目录号码 3c. 如果是个新名字，分配目录号码，分配图书副本号码 4. 给图书编号 5. 将图书放到小车上 每本书都要重复以上步骤（步骤3）	3c.1 核实目录号码没有重复
异常条件	重复的号码需要进一步确认，并重新分配目录号码	

图 11-33（续）

4. 将在习题1、2和3中得到的设计类匿解决方案结合起来形成一个设计类图。

注意：练习5、6、7和8是基于第7章"思考题"中有关牙科门诊系统的练习1和2的解决方案而设计的。作为选择，你的老师可能会给你提供用例图和类图。

5. 图11-34所示为牙科门诊系统中记录看病过程用例的系统顺序图。完成以下习题：

(a) 建立初步顺序图，它只含有参与者和问题域类。

(b) 给（a）题中得到的图添加可视层和数据访问层。

(c) 基于域类图和（a）、（b）两题的结果创建一个设计类图。

(d) 创建一个包图，它描述了包含可视层、域层和数据访问层三层次解决方案的包。

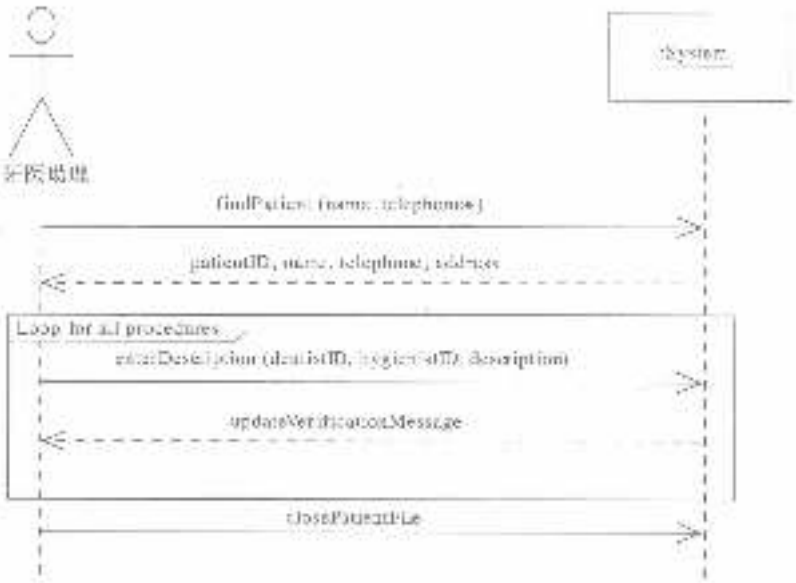


图11-34　记录看病过程的系统顺序图

6. 图11-35所示为牙科门诊系统中"输入新病人信息"用例的活动图。完成以下习题：

(a) 建立初步顺序图，它只含有参与者和问题域类。

(b) 给（a）题中得到的图添加可视层和数据访问层。

(c) 基于域类图和（a）、（b）两题的结果创建一个设计类图。

(d) 创建一个包图，它描述了包含可视层、域层和数据访问层包的三层次解决方案。

7. 图11-36所示为牙科门诊系统中"打印清单"用例的完全展开用例描述。完成以下习题：

(a) 建立初步顺序图，它只含有参与者和问题域类。

(b) 给 (a) 题中得到的图添加可视层和数据访问层。

(c) 基于域类库和 (a)、(b) 两题的结果创建一幅设计类图。

(d) 创建一幅泳道图，它描述了包含可视层、域层和数据访问层的三层次解决方案。

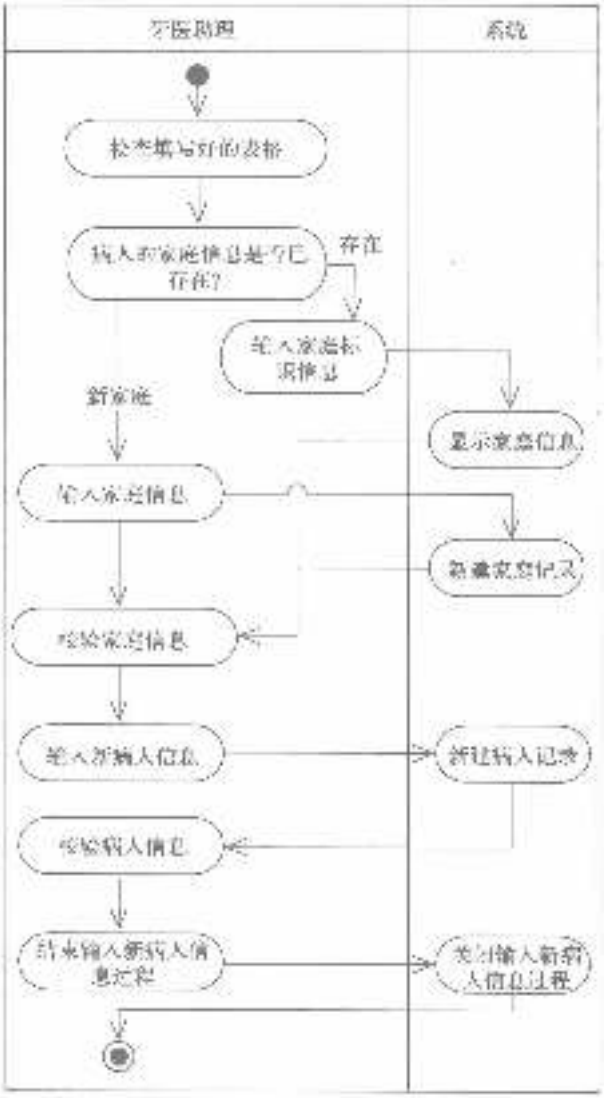


图11-35 输入新病人信息的活动图

8. 将在习题1、2和3中得到的设计类图解决方案结合起来形成一个设计类图。

9. 在第7章"思考题"的习题6中，已经为"高质量建筑供应"系统中的两个购买场景分别创建了活动图；在习题9中，创建了系统顺序图。基于上述活动图或者顺序图，以及后面域类图中的类，设计一个详细的协作图。只需包括问题域类。

10. 在第7章"思考题"的习题7、8和10中，已经为"将一辆新车加入一个已有保险中"创建了一个系统顺序图，还有一个类的列表。根据你创建的系统顺序图，设计一幅详细的协作图。只需包括问题域类。

用例名称	打印清单	
场景	打印清单	
触法事件	月末，打印清单	
简单描述	记账职员人工检查以确保已经收集了所有的过程。职员通过在系统中查看是否存在书面记录的方式进行批评调查，以确保所有的过程已经录入。最后，打印清单报表	
参与者	记账职员	
系统相关者	记账职员，开发	
前提条件	病人记录必须存在，过程必须存在	
后续条件	病人记录用上次的记账日期来更新	
事件流	参与者	系　　统
	1. 收集本月完成的有关过程的所有书面记录 2. 观察几个病人，确保过程信息已经输入 3. 检查已接收支付记录，核实支付已经输入 4. 输入月末日期，并请求清单 5. 核实清单的正确性 6. 关闭清单打印处理	2.1 显示病人信息，包括过程记录 3.1 显示病人信息，包括账户余额和上次支付事务 4.1 接收每个病人记录，找到未支付过程，分为高降和年级，列在报表上，计算并用公司支付或者保险支付逐步分解
异常条件	无	

图11-36　“打印清单”用例的完全展开描述

实验练习

1. 找到一个使用UML和面向对象技术的公司，采访IS成员，看看他们是如何使用UML的。了解使用域模型进行分析的方法。如果使用顺序图的话，他们是如何来设计新系统的。询问他们使用SDLC的情况，是使用预测方法还是自适应方法？
2. 找到一个使用Java开发的系统。如果可能的话，找一个既含有基于Internet的用户接口又含有基于网络的用户接口的系统。它是多层（三层或者两层）的吗？你能分开可视层、域层和数据访问层吗？
3. 找到一个使用Visual Studio.NET（或者Visual Basic）开发的系统。如果可能的话，找一个既含有基于Internet的用户接口又含有基于网络的用户接口的系统。它是多层（三层或者两层）的吗？你能分开可视层、域层和数据访问层吗？
4. 选择一种自己熟悉的面向对象编程语言，并找到支持该语言的程序设计集成开发环境（IDE）。测试一下从现有代码产生UML类图的逆向工程的处理能力。评价一下效果如何，模型使用得轻松吗？它能输入UML图，并产生类定义草图吗？写个报告，讲讲有关它的工作过程，以及它能生成什么样的UML模型？

实例研究

房地产多编目服务系统

在第7章中，我们为房地产公司的用例建立了用例图、类图和系统顺序图。在这些基础上，或者老师的帮助下，为域类图创建初步顺序图。接着，给顺序图添加可视层和数据访问层。通过输入属性，添加方法特征，将域类图转化为设计类图。

巡警罚单处理系统

在第7章中，我们为“记录交通罚单”和“确定当前时间”用例创建了用例图、类图和系统顺序图。在这些基础上，或者教师的帮助下，为域类图创建初步顺序图。接着，给顺序图添加可视层和数据访问层。通过输入属性，添加方法特征，将域类图转化为设计类图。

城市影碟出租系统

在第7章中，我们为“录入录象带”和“归还录象带”用例创建了用例图、类图和系统顺序图。在这些基础上，或者教师的帮助下，为域类图创建初步顺序图。接着，给顺序图添加可视层和数据访问层。通过输入属性，添加方法特征，将域类图转化为设计类图。

THEEYESHAVEIT.COM网站图书交换系统

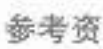
在第7章中，我们为“添加图书入库”和“记录图书顺序”用例创建了用例图、类图和系统顺序图。在这些基础上，或者教师的帮助下，为域类图创建初步顺序图。接着，给顺序图添加可视层和数据访问层。通过输入属性，添加方法特征，将域类图转化为设计类图。

对落基山运动用品商店实例的再思考



本章讲述了RMO两个用例——“查询可用条目”用例和“维护产品信息”用例的解决方案，并尝试为另外两个用例——“创建新订单”和“记录订单完成”设计三层的解决方案。使用这些用例设计的方法特征更多问题域的设计类图。在通常情况下，用来写报告的顺序图是非常有意思的。为产生订购完成报告用例完成三层次设计。由于没有这个用例详细的用户需求，只需列出用一个样本完成报告的提纲。

关注Reliable Pharmaceutical Services



在第7章中，我们为三个用例创建了用例图、域模型类图和详细的文档。在详细文档中，又生成了完整的规约和系统顺序图。基于这些信息和本章的一些准则，为每个用例设计一个三层结构。使用属性类型信息和顺序图中的方法特征来更新设计类图。

参考资料

Grady Booch, James Rumbaugh, and Ivar Jacobson. *The Unified Modeling Language User Guide*. Addison-Wesley, 1999.

Frank Buschmann, R. Meunier, H Rohnert, P. Sommerlad, and M. Stal. *Pattern-Oriented Software Architecture; A System of Patterns*. John Wiley and Sons, 1996.

E. Reed Doke, J. W. Satzinger, and S. R. Williams. *Object-Oriented Application Development Using Java*, Course Technology, 2002.

E. Reed Doke, J. W. Satzinger, and S. R. Williams. *Object-Oriented Application Development Using Microsoft Visual Basic .NET*. Course Technology, 2003.

Hans-Erik Eriksson, Magnus Penker, Brian Lyons, and David Fado. *UML 2 Toolkit*, John Wiley and Sons, 2004.

Martin Fowler. *UML Distilled Third Edition: A Brief Guide to the Standard Object Modeling Language*. Addison-Wesley, 2004.

Ivar Jacobson, Grady Booch, and James Rumbaugh. *The Unified Software Development Process*. Addison-Wesley, 1999.

Philippe Kruchten. *The Rational Unified Process, An Introduction*. Addison-Wesley, 2000.

Craig Larman. *Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and the Unified Process* (3rd ed.). Prentice-Hall, 2004.

James Rumbaugh, Ivar Jacobson, and Grady Booch. *The Unified Modeling Language Reference Manual*. Addison-Wesley, 1999.

第12章 数据库设计

学习目标

阅读本章后，你应具备如下能力：

- 描述关系数据库和面向对象数据库管理系统的异同
- 根据实体-联系图，设计关系数据库模式
- 根据类图，设计对象数据库模式
- 设计一个实现混合对象和关系数据库的关系模式
- 描述分布式数据库的不同结构的概念

本章要点

- 数据库和数据库管理系统
- 关系数据库
- 面向对象数据库
- 混合对象-关系式数据库设计
- 数据类型
- 分布式数据库

全国图书公司：设计一个新的数据库

全国图书公司（NB）新的网上订书系统的项目负责人与系统数据库管理员举行了一次会谈，就如何进行数据库的设计进行讨论。出席会议的有Sharon Thomas（开发初期便负责这个项目），Vince Pirelli（参与了大部分分析工作的承包商）和Bill Anderson（NB数据库管理员，项目开发初期没有直接参与）。Sharon主持会议并发言："项目一开始，我们计划使用现有的DB2数据库。Maria Pena（首席信息官，CIO）希望我们通过这个项目来试用在不久的将来要使用的新开发方法和工具，所以我们聘请Vince使用面向对象（OO）的方法进行系统分析。另外，我们也购买了一个Java开发环境并让我们的两名程序员上了一个星期的培训课。"

Vince补充道："我开发了一个传统的实体-联系图，这个图将作为设计新程序与现存数据库接口的基础。我查看了目前数据库的文档，新系统所需的大部分信息已经存在了。但我们还要加入一些字段并更改一些表格定义。既然我们在考虑设计和实现，我担心我们可能正在用一个过时的数据库管理系统，这会阻碍我们的新系统。"

Sharon补充说："我们决定用这个项目作为面向对象开发和实现的领航员，以便使项目更快完成。但如果仍用现存的DB2数据库，我不能保证是否真的能够实现这样的目标。"

Bill说："我知道你担心的是面向对象程序与关系数据库的接口问题。这听起来像试图将水和油混合起来，但实际上并没有那么不难，我们刚刚聘请了Anna Jorgensen，一个数据库开发人员，她具有编写与关系数据库交互的Java程序的经验。我已经让她看过新系统的类图和用例。她担保与DB2数据库的连接将会是简单而直接的。如果你们需要帮助，我可以安排她到你们的项目中工作几个月。"

Vince回答说："毫无疑问，接口是可以编写出来的，但它可能没有Anna想的那么简单。

面向对象程序的一些基本要素，例如继承和类方法，还有一些问题，这些在关系数据库中根本不能表示。这将使我们在设计和实现面向对象代码时不得不做一些艰辛的调整。我担心这会延长设计和实现周期，并给以后的系统升级带来更多的不便。”

“我做过一些调查，现在已经有了相当多的业务的面向对象的数据包可用。”Sharon说，“它们还没有适用于DB2的记录，但这个技术非常新。一个面向对象数据库管理系统最适合与Java结合，这会使我们有机会接触到其他一些新技术。”

Vince补充说：“由于OO数据库是直接建立在类图的基础上的，这也将缩短我们费用来完成设计的时间。”

Sharon继续说：“我想这个项目给我们过渡到下一代数据库软件提供了一个契机。”

Bill已被这些建议搞糊涂了，但他还是很快回答说：“你们是在让我支持两种基于完全不同数据库技术的冗余的数据库吗？管理已经超出了我的预算。为节省资金我已将服务器换成更便宜的，但这只是杯水车薪，没有什么大的作用。支持另一个数据库管理系统将会是一个大难题。我们需费用新的硬件将新的数据库管理系统与现有的数据库分离开来。我不能冒这个险。用一个有漏洞的软件去破坏现行的数据库。我们还要训练我们的人员熟悉这个新软件。此外，我们如何在两个数据库之间来回存取数据呢？”

Sharon先使气氛缓和一点，才做回答，说：“这无疑问，这是一个重要的任务，我不会忽视这个事实或让你的人员与设备达到极限。但现在我们有一个转向更新技术的机会，并且时机已经成熟。我已经和Maria讨论过这个问题，她也认为值得认真考虑一下。”

Vince补充说：“我可以用两种方法中的任何一种进行数据库设计。作如果我们用关系数据库，就不能为将来打下一个基础。如果需要，我甚至可以将它与一台老式IBM主机上的索引文件相连。但我们为什么不选择前进而要后退呢？”

概述

数据库和数据库管理系统是现代信息系统的重要组成部分。数据库为数据提供了一个公共仓库，使得整个组织可以共享这些数据。数据库管理系统为设计者、程序员以及终端用户提供复杂的存取、检索和管理数据的功能。如果没有数据库管理系统，共享和管理现代组织所需要的大量数据简直是天方夜谭。

在第5章中，已学习了构造概念数据模型，也学习了运用实体-联系图（ERD）进行传统的分析，以及运用类图进行面向对象的分析。为实现一个信息系统，项目组必须将概念数据模型转化为更加详细的数据库模型，然后在数据库管理系统中实现这些模型。

开发数据库模型的过程取决于概念模型的类型以及用于实现系统的数据管理软件的类型。本章将讲述关系数据模型和面向对象数据模型的设计，以及如何用数据库管理系统实现它们。我们将用RMO公司的例子来阐述在数据库设计时如何使用分析阶段收集来的信息。

12.1 数据库与数据库管理系统

数据库（DB）是被集中控制和管理的存储数据的完整集合。数据库一般存储几十或成百上千个实体类型或类的信息。存储的信息包括实体或类的属性（如姓名、价格、账户余额）以及实体或类之间的关系（如哪些订单属于哪些客户）。数据库中还存储数据的描述性信息，例如，字段名、对允许值的约束以及对敏感数据项的访问控制。

数据库（DB）：被集中控制和管理的存储数据的完整集合。

数据库由**数据库管理系统**（DBMS）来管理和控制。DBMS是一个通常与其他系统软件（如操作系统）分开购买和安装的系统软件。常见的现代数据库管理系统有Microsoft Access、

Oracle、DB/2、ObjectStore和Gemstone。

数据库管理系统（DBMS）：对数据库的访问进行管理和控制的系统软件。

12.1.1 DBMS的组件

图12-1给出了一个典型的数据库组件，以及它与数据库管理系统、应用程序、用户和管理人员的交互关系。数据库由两部分相关的信息存储组成：物理数据存储和模式。物理数据存储包含了信息系统生成和使用的原始比特和字节。模式包含了关于物理数据存储中所存储数据的描述性信息，包括：

- 访问和内容控制，包括特殊数据元素的允许值、多重数据元素之间的节依赖关系以及允许读取和更新数据元素内容的用户列表。
- 数据元素和数据元素组之间的关系（例如，一个从描述客户的数据指向客户所提交的订单的指针）。
- 物理数据存储组织的细节，包括数据元素的类型和长度、数据元素的位置、关键数据元素的索引以及相关数据元素组的排序。

物理数据存储：数据库管理系统用来存储数据库原始比特和字节的存储区域。

模式：对物理数据存储或数据库的结构、内容以及访问控制的描述。

一个数据库管理系统（DBMS）包括4个关键的组成部分：应用程序接口（API）、查询接口、管理接口以及一组基本的数据访问程序和子程序。应用程序、用户和管理人员从来不会直接访问物理数据存储，相反，他们使用在模式中定义的名字来“告诉”某个适当的数据库管理系统接口他们需要读/写的数据。数据库管理系统通过访问模式来确认所请求的数据是否存在以及该用户是否有访问该数据权限。如果请求是有效的，则数据库管理系统从模式中提取出所请求数据的物理组织信息，并代替提出请求的程序或用户使用该信息访问物理数据存储。

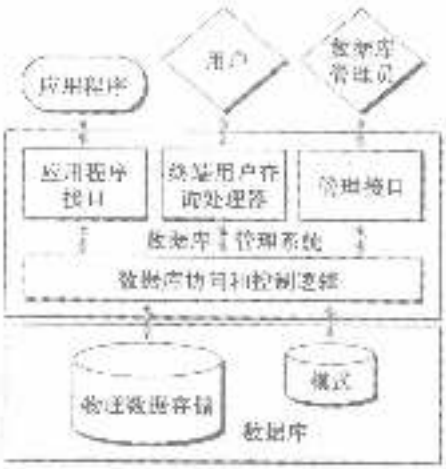


图12-1　数据库和数据库管理系统的组成，以及它们与应用程序、用户、数据库管理员的交互

数据库和数据库管理系统提供了一些重要的数据访问和管理功能，包括：

- 允许多个用户或应用程序同步访问；
- 无须编写应用程序来访问数据（即可通过查询语言）；
- 应用相同且一致的访问和内容控制。

由于上述和其他的一些原因，数据库和数据库管理系统（DBMS）在现代信息系统中得到了广泛的应用。

12.1.2 数据库模型

数据库管理系统自从20世纪60年代出现后，经历了一系列的变化。其中最重要的变化是用来表示和访问物理数据存储内容的模型。以下4种模型得到了广泛的应用：

- 层次模型
- 网状模型
- 关系模型
- 面向对象模型

层次模型出现于20世纪60年代。它用一系列组织成层次的记录来表示数据。网状模型也

把数据元素分成许多记录集合的组。但允许把这些记录组成更灵活的系统结构。近20年来，已经很少有人用层次模型开发新数据库了。然而，现在许多老的层次和网状数据库仍在使用，特别是在大规模的批量交易处理的应用当中。

本章的其余部分将会详细地讨论关系数据库和面向对象数据库模型的设计问题。基于面向对象概念的数据库管理系统是目前现行和新开发的系统当中应用得最广泛的。另外，本章不再介绍基于层次模型和网状模型的设计问题，因为信息系统专业的学生在以后的工作中几乎不会遇到基于这些模型的数据库管理系统。

12.2 关系数据库

关系数据库发展于20世纪70年代初，但由于对层次和网状数据库进行转换固有的不确以及成功实施关系数据库所需要的大量计算资源，使得关系数据库的采用经历了相当长的一段时间。随着计算机科学中其他理论的进步，数据存储和硬件处理的性价比特性过上新的理论花费了多年时间，关系数据库管理系统已成为现行数据库管理系统的主流。

关系数据库管理系统（RDBMS）是一个把存储数据组织成一种称为表或关系的结构的数据库管理系统。关系数据库的表与传统的数据表相类似，即都是包含行和列的二维数据结构。然而，关系数据库管理系统的专业术语与传统的表格和文件的专业术语有些不同。表中单独的一行叫行、元组或记录，表的一列叫字段或属性。表中的单一单元叫字段值、属性值或数据元素。

关系数据库管理系统（RDBMS）：在其中存储数据的数据库管理系统。

表：包括行和列的二维数据结构，也叫关系。

行：表的一部分，包含描述一个实体、关系或对象的数据，也叫元组或记录。

字段：关系数据库表的一列，也叫属性。

字段值：存储在关系数据库表的一个单元中的数据，也叫属性值或数据元素。

图12-2是用Microsoft Access关系数据库显示的一个表的内容。注意，表的第一行包含一系列的字段名（标题栏），而余下的每行都包含了一个描述某个特定商品的字段值的集合。还要注意的是，每一行都按同样的次序表示同样的字段。

实践指导

作为一个系统设计者，不管你是坚持传统的方法还是面向对象的方法，关系数据库和SQL是必须掌握的知识。

关系数据库的每个表都必须有一个唯一的关键字。关键字是一个字段或一个字段集合，它的值在表的各行中只能出现一次。如果只有一个字段（或字段组）是唯一的，那么这个关键字业称为表的主键。如果有多个唯一的字段（或字段组），那么必须从中选择一个作为主键。

关键字：在能够一对应关系数据库表中各行的字段。

主键：用来唯一标识关系数据库中表的某一行的关键字。

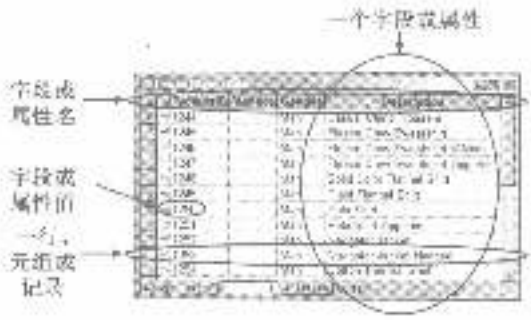


图12-2 某关系数据库表格的部分显示

关键字段可以是自然属性，也可以是人为定义的。例如，化学中一个描述元素数据的表格中直接把元素的原子量作为关键字段。然而遗憾的是，在业务中，要处理的信息很少有直接能用的关键字段。在关系数据库中，大多数的关键字段是人为定义的。你的钱包里可能就

有许多定义的关键字的例子，包括你的社会保险号码、驾驶证号、信用卡账号以及ATM卡号。有些定义的关键字是外部指定的（例如联邦快递公司的跟踪号码），也有许多是内部指定的（例如图12-2中的产品序列号）。定义的关键字可以保证是唯一的，因为插入新行时，用户、应用程序或数据库管理系统会给新行的关键字赋予唯一的值。

关键字是关系数据库设计的决定性因素，因为它们是表示表间关系的基础。关键字是连接一个表和其他表中的行之间的“纽带”，换句话说，是关键字将表格相互连接起来的。例如，考虑图12-3所示的RMO公司例子中的部分实体-联系图和图12-4所示的表。这个实体-联系图片断给出的是实体“产品条目”与“库存条目”之间可供选择的一对多关系。图12-4上面的表包含代表实体类型ProductItem的数据，下面的表包含代表实体类型InventoryItem的数据。



图12-4　两个表中数据之间的关系：InventoryItem表中的外键ProductID对应ProductItem表中的主键ProductID

实体类型“产品条目”和“库存条目”之间的关系由它们各自表的公共字段值表示。ProductID字段（ProductItem表中的主键）也以ProductID字段出现在InventoryItem表中，这时字段ProductID叫做外键。外键是另一个不同的（或外部）表的主键的副本字段。在图12-4中，InventoryItem表中作为一个外键存在的值1244决定了ProductItem表中的第一行中Vendor、Gender和Description的值，同时也指定号码在86779－86790之间的库存项。

外键：存储在一个关系数据库表中，同时又是另一个关系数据库表的主键值的字段值。

12.2.1　设计关系数据库

关系数据库设计可以从一个实体-联系图（ERD）或一个类图开始。这一节介绍如何根据一个实体-联系图来生成数据库模式，基于类图的模式建立将在本章的后面讨论。

从实体-联系图建立一个关系数据库模式，可采取以下步骤：

1. 为每个实体类型建立一张表。
2. 为每个表选择一个主键（如果需要的话，可以定义一个）。
3. 增加外键以表示一对多关系。
4. 建立新表来表示多对多关系。
5. 定义参照完整性约束。
6. 评价模式质量，并进行必要的改进。
7. 为每个字段选择适当的数据类型和取值约束（如果需要的话）。

在下面的12.2.2节中，我们将详细讨论上述每个步骤。

12.2.2 实体的表示

建立一个关系数据库模式的第一步是给ERD中的每个实体建立一张表。图12-5是RMO客户支持系统的ERD。其中有11个实体，并为每个实体建立了一张表。每张表中的数据字段要与对应的实体定义相一致。为避免混淆，表和字段名称应该与ERD和项目数据字典中的名称相同。图12-6中列出了表示ERD中实体的原始表的集合。

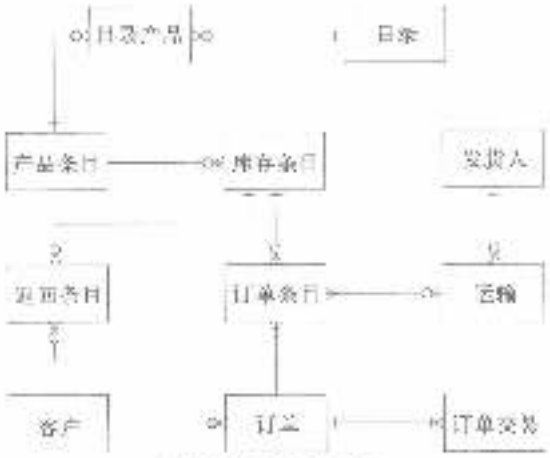


图12-5 RMO的ERD

为每个实体创建表后，设计者必须为每个表选择一个主键。如果表格已有一个字段或字段组可以保证为唯一的，那么设计者可选择这些键作为主键（如Shipment表中的"Tracking"字段）。如果设计者不能从已有的字段或字段组中选择一个主键，那么必须构造一个新键。它可以取任意名称，但该名称应该能够暗示这个字段是唯一的主键字段。典型的名称包括"Code"、"Number"、"ID"，也可以与表名结合起来，例如"ProductCode"和"OrderID"。图12-7给出了实体表，同时确定了每个表的主键。

表	属 性
Catalog	Season, Year, Description, EffectiveDate, EndDate
CatalogProduct	Price, SpecialPrice
Customer	AccountNo, Name, BillingAddress, ShippingAddress, DayTelephoneNumber, NightTelephoneNumber
InventoryItem	InventoryID, Size, Color, Options, QuantityOnHand, AverageCost, ReorderQuantity
Order	OrderID, OrderDate, PriorityCode, ShippingAndHandling, Tax, GrandTotal, EmailAddress, ReplyMethod, PhoneClerk, Call Start Time, LengthOfCall, DateReceived,ProcessorClerk
OrderItem	Quantity, Price, BackorderStatus
OrderTransaction	Date, TransactionType, Amount, PaymentMethod
ProductItem	ProductID, Vendor, Gender, Description
ReturnItem	Quantity, Price, Reason, Condition, Disposal
Shipment	TrackingNo, DateSent,TimeSent,ShippingCost, DateArrived,TimeArrived
Shipper	ShipperID, Name, Address, ContactName, Telephone

图12-6 表示ERD中实体的原始表的集合

表	属 性
Catalog	**CatalogID**, Season, Year, Description, EffectiveDate, EndDate
CatalogProduct	**CatalogProductID**, Price, SpecialPrice
Customer	**AccountNo**,Name, BillingAddress,ShippingAddress,DayTelephoneNumber, NightTelephoneNumber
InventoryItem	**InventoryID**, Size, Color, Options, QuantityOnHand, AverageCost, ReorderQuantity

图12-7 带主键（用黑体标出）的实体表

表	属　性
Order	**OrderID**, OrderDate, PriorityCode, ShippingAndHandling, Tax, GrandTotal, EmailAddress, ReplyMethod, PhoneClerk, CallStartTime, LengthOfCall, DateReceived,ProcessorClerk
OrderItem	**OrderItemID**, Quantity, Price, BackorderStatus
OrderTransaction	**OrderTransactionID**,Date,TransactionType,Amount, PaymentMethod
ProductItem	**ProductID**, Vendor, Gender, Description
ReturnItem	**ReturnItemID**, Quantity, Price, Reason, Condition, Disposal
Shipment	**TrackingNo**, DateSent,TimeSent,ShippingCost, DateArrived,TimeArrived
Shipper	**ShipperID**, Name, Address, ContactName, Telephone

图 12-7（续）

12.2.3 关系的表示

关系数据库的关系由外键表示，哪个外键应该放在哪张表中取决于所表示关系的类型。图12-5所示的RMO的ERD中包含9个一对多关系。在Catalog和ProductItem之间有一个多对多的关系，它由两个一对多的关系和相关实体CatalogProduct来表示。每个关系类型的表示规则如下。

- **一对多关系**——把“一个”对应的实体类型的主键字段添加到表示“多个”的实体类型的表中。
- **多对多关系**——如果没有其种关系的相关实体，则建立一张新表来代表这种关系；如果这种相关实体确实存在，则就使用这张表来代表这种关系。使用相关实体的主键字段作为这张表的主键。

图12-8表示的是图12-7数据表中的9个一对多关系的表示结果。每个外键表示了包括外键的表与用该字段作为主键的表之间的单个关系。例如，把AccountNO字段作为一个外键加入到Order表中，表示实体Customer与Order之间的一对多关系，把外键ShipperID添加到Shipment表中表示Shipper和Shipment之间的一对多关系。注意，当表示一对多关系时，外键不会成为它所添加到的表中主键的一部分。

表	属　性
Catalog	**CatalogID**, Season, Year, Description, EffectiveDate, EndDate
CatalogProduct	**CatalogProductID**, Price, SpecialPrice
Customer	**AccountNo**,Name, BillingAddress,ShippingAddress,DayTelephoneNumber, NightTelephoneNumber
InventoryItem	**InventoryID**, *ProductID*, Size, Color, Options, QuantityOnHand, AverageCost, ReorderQuantity
Order	**OrderID**, *AccountID*, OrderDate, PriorityCode, ShippingAndHandling, Tax, GrandTotal, EmailAddress, ReplyMethod, PhoneClerk, CallStartTime, LengthOfCall, DateReceived,ProcessorClerk
OrderItem	**OrderItemID**, *OrderID*, *InventoryID*, *TrackingNo*, Quantity, Price, BackorderStatus
OrderTransaction	**OrderTransactionID**,OrderID,Date,TransactionType,Amount, PaymentMethod
ProductItem	**ProductID**, Vendor, Gender, Description
ReturnItem	**ReturnItemID**, *OrderID*, *InventoryID*, Quantity, Price, Reason, Condition, Disposal
Shipment	**TrackingNo**, ShipperID, DateSent, TimeSent, ShippingCost, DateArrival, TimeArrived
Shipper	**ShipperID**, Name, Address, ContactName, Telephone

表12-8　通过增加外键属性（斜体）来表示一对多关系

图12-9通过更新CatalogProduct表，对图12-8中的定义进行了扩充。从而表示了Catalog和ProductItem之间的多对多关系。CatalogID和ProductID共同构成CatalogProduct表的主键，而原有的主键CatalogProductID就不再使用了。这两个共同构成主键的字段也是外键，CatalogID是从Catalog表中得到的外键，而ProductID是从ProductItem表中得到的外键。

表	属 性
Catalog	**CatalogID**, Season, Year, Description, EffectiveDate, EndDate
CatalogProduct	**CatalogID**, **ProductID**, Price, SpecialPrice
Customer	**AccountNo**,Name, BillingAddress, ShippingAddress, DayTelephoneNumber, NightTelephoneNumber
InventoryItem	**InventoryID**, *ProductID*, Size, Color, Options, QuantityOnHand, AverageCost, ReorderQuantity
Order	**OrderID**, *AccountNo*, OrderDate, PriorityCode, ShippingAndHandling, Tax, GrandTotal, EmailAddress, ReplyMethod, PhoneClerk, CallStartTime, LengthOfCall, DateReceived,ProcessorClerk
OrderItem	**OrderItemID**, *OrderID*, *InventoryID*, *TrackingNo*, Quantity, Price, BackorderStatus
OrderTransaction	**OrderTransactionID**,*OrderID*,*Date*,*TransactionType*,*Amount*, PaymentMethod
ProductItem	**ProductID**, Vendor, Gender, Description
ReturnItem	**ReturnItemID**, *OrderID* ,*InventoryID*, Quantity, Price, Reason, Condition, Disposal
Shipment	**TrackingNo**, *ShipperID*, DateSent, TimeSent, ShippingCost, DateArrived, TimeArrived
Shipper	**ShipperID**, Name, Address, ContactName, Telephone

图12-9 经过修改的表示Catalog表与ProductItem表之间的多对多关系的CatalogProduct表

12.2.4 加强参照完整性

既然我们已经描述了如何用外键表示关系，我们还需要描述如何限制这些外键的取值。参照完整性描述了外键值和主键值之间关系的一致性。每一个外键是另一个表的主键的参照。在多数情况下，数据库设计者希望保证这些参照是一致的。也就是说，一个表中的外键值也必须是相关表格的主键值。参照完整性约束是对数据库内容的约束，例如，“一张订单必须来自于一个客户”，同时“一个订单项必须存储在库存清单中”。

参照完整性：一个一致的关系数据库状态，其中每个外键的值也作为一个主键的值存在。

一旦设计者确定了主键和外键，当遇到以下情况时，DBMS会自动执行参照完整性规则。

- 当建立一个包含外键值的行时，DBMS会确保它在另一个相关表格中以主键形式出现。
- 当删除一行时，DBMS确保相关表格中没有外键与被删行的主键有相同值。
- 当改变一个主键值时，DBMS要求相关表格中没有外键与它有相同值。

在第一种情况下，DBMS拒绝添加包含未知外键值的行。在后两种情况下，数据库设计人员通常要决定如何执行参照完整性进行一定的控制。当包含主键的行被删除时，设计人员可以让DBMS删除其他表中含有相应关键字的所有行，也可以把所有相应的外键设为NULL。当改变一个主键值时，可以采取类似的操作，DBMS会将所有对应的外键值改为同样的值，或把外键的值置为NULL。

12.2.5 模式质量评估

在建立了一整套表后，设计者应该检查整个模式的质量，消除模式中现在所能找出的所

有问题，以免浪费后期所付出的努力。一个高质量的数据模型具有以下特点：

- 表中每行以及主键都是唯一的
- 冗余数据较少
- 容易实现未来对数据模型的改变

然而，对实现高数据库模式质量的方法很少。数据库设计是建模过程的最后一步，它同样依赖于分析师的经验和判断。在后面的章节中我们将介绍大量正式或非正式的评价模式质量的方法。没有一种方法是万能的，但它们的结合能保证设计出一个高质量的数据库。

1. 行和关键字的唯一性

所有关系数据模型都有一个基本的要求：主键和表中的行都是唯一的。每个表必须有一个主键，如果主键是唯一的，那么表中的每一行也一定是唯一的。程序中的数据访问逻辑的基础是，假设关键字是唯一的。例如，一个程序员写一个查看客户记录的程序，一般假设对一个特定客户号的数据库查询仅会返回唯一一行（如果该客户不在数据库中，则没有结果）。程序将围绕这个假设进行设计，如果数据库管理系统返回两条记录，那么该设计就失败了。

设计者通过考查关键字内容、可能的关键字值以及关键字管的指定方法来评价主键的唯一性。在这一点上，由于内部定义的关键字是系统自动产生的，所以相对来说比较容易评价，也就是说，使用自定义关键字的信息系统，可以通过执行适当的程序为新建的行指定关键字来保证其唯一性。

信息系统中的一些不同的程序能在数据库中产生新行是很普遍的。因此，每个这样的程序都应能给数据库中新建立的行指定关键字。然而，关键字唯一性的重要性要求这些功能必须一致地应用于整个信息系统之中。

因为关键字的生成和管理在信息系统中是一个普遍的问题，所以许多关系数据库管理系统提供自动生成关键字的功能。这样的系统通常会自动为定义的关键字产生一个特定的数据类型（例如Microsoft Access的AutoNumber 类型）。DBMS会给新产生的行自动分配一个关键字的值，并将该值传送给应用程序以备后续的操作数据库之用。DBMS中这一功能的嵌入使得信息系统开发人员省去了设计和实现自定义关键字软件模块的麻烦。

对于不是由信息系统指定的自定义关键字，必须对它在整个过程的唯一性和有效性进行仔细审查。例如，在美国，常常用社会保险号码作为职工数据库的关键字，因为美国政府十分重视该号码的唯一性，所以假设该号码具有唯一性是安全的。但对于这些号码的使用的其他一些假设还需要进行仔细的检查。例如，存储在数据库中的所有职员是否都有一个社会保险号码？如果公司在欧洲或南美开分公司，那又该怎么办？

对由非政府指定的自定义关键字需要进行更仔细的检查。例如，联邦快递公司以及许多船业公司为它们所处理的每次运输指定一个跟踪号。跟踪号在某时段一定是唯一的，但能够保证永远唯一吗（即它们是否被重复使用）？重复使用跟踪号是否会引起RMO数据库中主键的重复？并且，如果两个执行不同发货任务的发货人被指定同一个跟踪号，会发生什么事情？

在多数情况下，这些不确定性使得内部定义关键字成为保证安全性的长期策略。虽然内部定义关键字最初可能需要额外的设计和开发，但它们可以防止以后可能发生的资源剧变。在一个带有千兆位存储数据、成千上万个应用程序和查询语句的大型信息系统中，几乎没有什么变化能像数据库关键字的改变那样带来广泛而又灾难性的影响。

2. 数据模型的灵活性

在关系数据库最早的规范当中，数据库的灵活性和可维护性是主要目标。如果对数据库模式进行更改，对现存的数据内容和结构造成的影响很小，就认为这个关系数据库模型是灵活的和可维护的。例如，添加一个新的实体到模式中，不需要对现存的表进行重新定义；增

加一个新的一对多关系只要求给已存在的表添加一个外键，增加一个新的多对多关系，只需要在模式中添加一个单独的新表。

数据冗余在判断数据库或数据模型的适应性和可维护性方面起着重要的作用。众所周知，冗余的存储要求额外的维护。也就是说，如果数据存储在多个地方，那么对数据进行添加、改变或删除时，这些数据都要被找到并更改。同样，在存储数据的多个地方执行这样的操作比在一个地方执行操作更为复杂（并且更低效）。对存储在多处的同一信息，如果不能执行相同的更新、修改或删除操作，不一致的数据便自然产生了。然而如果信息只存储在一个地方，那么这种不一致性也就不会出现了。

关系数据模型故意多次存储关键字段（即冗余的），而对非关键字则只存储一次。作为主键的关键字只被存储一次，而当其次存储时，则是作为外键。因为主键和外键的对应是表示各表之间关系的唯一办法，所以有对冗余的关键字值是必要的。但是使用冗余关键字值也增加了对关键字段操作的复杂性。

关系数据管理系统通过参照完整性的约束来保证主键和外键之间的一致。但是，没有自动化机制保证其他冗余数据项的一致。因此，关系数据库中避免不一致的最好办法是避免非关键字段的数据冗余。数据库设计人员可以通过不将它们引入模式中来避免这样的冗余，但冗余的出现太容易了。如果冗余数据不知是出于什么原因引入了模式中，那么就需要有一个处理过程来识别并消除它。检测和消除冗余数据的最普遍方法是数据库规范化。

3. 数据库规范化

规范化是一个用来评价关系数据库模式质量的有效技术。它确定一个数据库模式是否包含了任何错误的冗余，并且定义特定的方法来减少这些冗余。规范化是基于函数依赖和一系列范式的概念的。

- 第1范式（1NF）：如果一个表没有重复字段或字段组，那么它是第1范式的。
- 函数依赖：函数依赖是指两个字段值之间的一对一关系，其正式声明是：对于字段B的任何一个值在字段A中有且只有一个值与之对应，则称A函数依赖于B。
- 第2范式（2NF）：如果一个表是第1范式的且每个非关键元素均函数依赖于整个主键，则称它是第2范式的。
- 第3范式（3NF）：如果一个表是第2范式的且没有非关键字段函数依赖于任何其他非关键字段，则称它是第3范式的。

规范化：通过最小化数据冗余来保证数据库模式质量的过程。

第1范式（1NF）：没有重复的字段或字段组的关系数据库表结构。

函数依赖：两个字段值之间的一一对应关系。

第2范式（2NF）：每个非关键字段函数依赖于主键的关系数据库表结构。

第3范式（3NF）：没有非关键字段函数依赖于任何其他非关键字段的关系数据库表结构。

下面我们将对这些概念做进一步解释。

第1范式　第1范式是对表格的一个结构限制，像图12-10中Dependent这样的重复字段在关系数据库的表中是不允许的。重复的字段组也是禁止的。实际上，因为关系数据库管理系统不允许设计人员定义一个包含重复字段的表，所以第1范式是不难实现的。

SSN	Name	Department	Salary	Dependent1	Dependent2	Dependent3	Dependent4
[illegible]	Mary Smith	Accounting	40,000	John	Alice	Jane	
[illegible]	Joe Jean	Marketing	50,000				
[illegible]	Frank Collins	Production	55,000	Joe	Julia		

图12-10　带有重复字段的职员表

函数依赖　函数依赖是一个很难描述和应用的概念。判断一个函数依赖的最普通的方法是，在表中选两个字段并把它们插入前面定义中的斜体部分。例如，考察ProductItem表中的ProductID和Description字段（见图12-4）。我们知道ProductID是一个内部定义的主键，在表中一定是唯一的。为了判断Description是否函数依赖于ProductID，在函数依赖定义的斜体部分用Description替换A，用ProductID替换B：

"如果对于每个ProductID值有且只有一个Description值与之对应，则Description函数依赖于ProductID。"

现在设想一想，对于ProductItem表中的所有可能存在的行，这个命题是否是正确的？如果正确，那么Description函数依赖于ProductID。只要能保证自定义关键字ProductID在ProductItem表中是唯一的，那么前面的命题也是正确的。因此，Description函数依赖于ProductID。

分析Description函数依赖于ProductID的一个不太正式的方法是记住ProductItem表代表RMO公司出售的某一特定商品。如果数据库中该商品仅有唯一的一个Description，那么Description函数依赖于表示商品的关键字ProductID。如果某个商品有多个Description，那么Description字段不函数依赖于ProductID。

第2范式　评价ProductItem表是否符合第2范式，我们必须首先判断它是否是第1范式。不包含重复的字段就是第1范式的，接着，我们要判断每个非关键字段是否都函数依赖于ProductID（即，在函数依赖定义中依次用各个字段替换A），如果所有的非关键字段都函数依赖于ProductID，那ProductItem表是2NF的；如果一个或多个非关键字段不函数依赖于ProductID，那么这个表不是2NF的。

当主键由两个或多个字段组成时，要判断一个表是否是2NF的，就更复杂了。例如，考虑图12-11所示的RMO公司系统的CatalogProduct表。这个表表示了Catalog和ProductItem之间的一个多对多关系，因此，表达这个关系的表的主键由Catalog的主键（CatalogID）和ProductItem的主键（ProductID）组成。这个表还包含两个非关键字段Price和SpecialPrice。



图12-11　简化后RMO的CatalogProduct表

如果表是2NF的，那么非主关键字段Price必定函数依赖于CatalogID和ProductID的组合。我们可以通过替换函数依赖定义中的同样来验证函数依赖，并判断下面的命题是否正确：

"如果对于CatalogID和ProductID的任一组合值有且只有一个Price的值与之对应，那么称Price函数依赖于CatalogID和ProductID的组合。"

分析上述语句的正确性是需要技巧的，因为你必须考虑CatalogProduct表中所有可能出现的关键字值的组合。一个较简单的解决方法是考虑表格中出现的基本实体。一个商品可能在多个不同的目录（即Catalog）中出现，如果在不同的目录中，它的价格不同，那么上述的命题是正确的。如果不论商品出现在哪个价目表中，它的价格总是一样的，那么这个命题是错误的且这个表不是2NF的。对于这个问题，答案没有定论，它取决于RMO公司对位于多个目录的产品的定价规则。

如果非关键字段只函数依赖于主键的一部分，那么非关键字段必须从当前所在表中移出并放在另一个表中。例如，考虑图12-12上半部分所示的另一个版本的CatalogProduct表。非关键字段CatalogIssueDate只函数依赖于CatalogID，而不是CatalogID和ProductID的组合。因此，这个表不是2NF的。

为修正这个错误，我们必须将CatalogIssueDate从CatalogProduct表中移出，并把它放入将CatalogID单独作为关键字的表中，因为图12-9的Catalog表使用CatalogID作为主键，所以

CatalogIssueDate应该加入到这个表中。如果Catalog表不存在，就需要我们创建一个包含CatalogIssueDate的表，像图12-12所示的那样。



图12-12 将一个1NF表分解成两个2NF表

第3范式 验证一个表是3NF的，我们必须考虑是否每一个非关键元素均函数依赖于另一个非关键元素。这对于一个大型表来讲是很麻烦的，因为，随着非关键字段的增加，要考察的数字对也快速地增加，当非关键字段数是N时，要考察的函数依赖数目是N（$N-1$）。要注意的是，对函数依赖的考察是两个方面的（即A相关于B，B相关于A）。

实际上，可以集中考虑以下两类问题从而简化寻找3NF的过程：

- 存放描述两个或更多实体的属性的表
- 可计算的字段

考虑图12-13所给出的简单表。假设AccountNo是主键，并且所有的客户都住在美国。因为有三个非关键字段，所以需要考察以下6个函数依赖：

- State是否函数依赖于Street Address?
- Street Address是否函数依赖于State?
- Zip Code是否函数依赖于Street Address?
- Street Address是否函数依赖于Zip Code?
- Zip Code是否函数依赖于State?
- State是否函数依赖于Zip Code?



图12-13 将一个2NF表转化成两个3NF表

在上述6个陈述中，前面5个的前者不函数依赖于后者，只有最后一个的前者是函数依赖

于后者的。在美国，所有的邮政编码（Zip Code）都按州划分。因此，每个Zip Code值对应唯一的State值（例如，87123总是在美国新墨西哥州中）。将这两个字段都包含到这个表中就造成冗余。例如，如果把100个居住在邮政编码为87123地区的客户的地址信息都存在表中，那么，87123所对应的新墨西哥州就被（冗余地）存储了100次。

事实上，这张表合并了两个实体的信息：Customer和Postal Delivery Area，每个实体都有自己的主键（Customer的主键是AccountNo，Postal Delivery Area的主键是Zip Code），同时也都有自己的非关键字段（Street Address和Zip Code是Customer的非关键字段，State是Postal Delivery Area的非关键字段）。

因为State函数依赖于Zip Code，所以这个表不是3NF的。为纠正这个问题，你必须把State从表中移出。State和Zip Code之间的关系必须保存在数据库的某个地方，否则，信息系统将不能产生完整的邮递标签。解决这个问题的办法是建立一个只包含State和Zip Code的新表（见图12-13）。在这个新表中，Zip Code是主键，State是唯一的非关键字段。这样一来，打印或显示一个完整的邮件地址的程序或方法必须通过CustomerAddress USA表中的Zip Code值才能在新表中找出对应的State值。

可计算字段中存放的值可以通过使用公式或算法来进行计算。这些公式和算法的输入是数据库中其他的字段。一些常见的可计算字段有Subtotals，Totals和Taxes。例如，考虑图12-9中Order表的GrandTotal字段和下面的公式：

$$\text{GrandTotal}=\Sigma\ (\text{Quantity}\times\text{Price})+\text{Shipping}+\text{Tax}$$

注意，公式中的所有输入并非都来源于同一张表（见图12-14），不像违反3NF原则的设计那样在一张表里存放多个实体，计算字段可以涉及多个表。Shipping和Tax存放在Order表中，Quantity和Price存放在OrderItem表的相关行中。计算某特定发货单的GrandTotal，需要通过OrderID这个外键在OrderItem表中找到所有与之匹配的行。

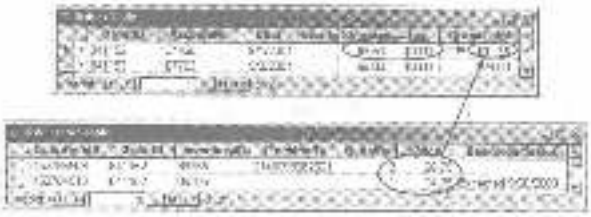


图12-14　从两张表的字段中计算出Grand Total

GrandTotal函数依赖于这4个字段的结合。这种计算性依赖也会造成冗余，因为公式中任何一个变量的改变（例如Shipping），最终都会导致计算结果的改变（例如GrandTotal）。

解决这类3NF冲突的方法很简单：从数据库移除可计算字段。移除可计算字段，并不意味着任何信息的丢失。例如，当某个程序或方法需要一个订单的GrandTotal信息的时候，它首先通过该订单的OrderID在OrderItem表中找到所有属于这个订单的OrderItem，然后累加每个OrderItem的Quantity与Price之积，最后加上这个订单的Shipping和Tax便可得到。

4. 实体-联系建模和规范化

实体-联系建模和规范化是关系数据库设计的补充技术。由RMO的ERD所产生的表（如图12-9所示）中并不包含任何1NF，2NF或3NF的冲突，这绝非偶然。实体的属性函数依赖于该实体唯一的标识符（主键）。多对多关系的属性函数依赖于两相关实体各自的唯一标识符。因此，当创建ERD时，分析师在决定哪个属性属于哪个实体或关系时，必须直接或间接考虑函数依赖问题。

实践指导

当需要维护和升级一个现有系统时，可以利用一个CASE自动生成工具将关系数据库转化为实体-联系图。 ■

现在我们将注意力转向如今广泛使用的第二类数据库——面向对象数据库。

12.3 面向对象数据库

对象数据库管理系统（ODBMS）是面向对象设计和编程范例的直接扩展，是为了存储对象并与面向对象程序设计语言交互而专门设计的。虽然也可以将对象存储在文件或关系数据库中，但使用专门为面向对象设计的数据库管理系统有许多的优点。这些优点包括对方法存储、继承、对象嵌套、对象链接以及程序员定义的数据类型的直接支持。

对象数据库管理系统（ODBMS）：以对象为基本实例存储数据的数据库管理系统。

作为研究原型，ODBMS最早出现在20世纪80年代。在20世纪90年代初期出现了它的业务雏型。目前，有多ODBMS有GemStone、ObjectStore和Objectivity等。ODBMS是可供采用面向对象工具实现的较新系统，尤其是科学工程应用选择的数据库技术。ODBMS有望在今后10年内逐步替代RDBMS在传统业务的应用。

由于ODBMS相对较新，所以对于指定对象数据库模式、很少有被广泛接受的标准。在20世纪末、21世纪初，对象数据库管理小组（Object Database Management Group）开发并完善了一个被称为对象定义语言（Object Definition Language,ODL）的标准。ODL是描述对象数据库结构和内容的语言。ODMG标准是Java数据对象（Java Data Objects，2003年被采用）标准的基础，同时也是ODMBS与C++、Small Talk语言交互的基础。在后面几节中，关于模式的例子都将用到ODL。

对象定义语言（ODL）：由对象数据库管理小组发布的一种标准的对象数据库描述语言。

12.3.1 设计对象数据库

依照下面步骤，从一个类图建立一个对象数据库模式：

1. 确定哪些类需要持久存储。
2. 定义持久类。
3. 表示持久类之间的关系。
4. 为每个字段选择合适的数据类型和值域（如果需要的话）。

我们将在下面的小节中对每一步进行详细的讨论。

12.3.2 类的表示

从数据管理的角度，对象可以分成两大类型。暂存对象仅存在于一个程序或过程的生命周期中，在一个遵循二层结构的设计中，用来实现用户界面的对象就是一个暂存对象（像一个窗口或一个弹出菜单）。暂存对象在程序或进程每次执行时产生，当程序或进程结束时消失了。

暂存对象：在实例化或方法调用中不需要存储任何属性值的对象。

当创建持久对象的程序或进程终止执行时，该持久对象不会消失。相反，它独立于任何其他程序或进程而继续存在。将对象状态存储在永久存储器中（例如磁盘或光盘）以保证对象在进程执行过程中一直存在。对象可以永久存储在文件或数据库管理系统中。

持久对象：在实例化或方法调用中必须存储一个或多个属性值的对象。

对象数据库模式包括每个需要永久存储的类的定义。ODL类定义可以从相应的UML类图中得到。因此，在UML中已定义的类只是重用于数据库模式定义。

例如，RMO客户类的ODL描述如下：

```
Class Customer{
   attribute string accountNo
   attribute string name
   attribute string billingAddress
   attribute string shippingAddress
   attribute string dayTelephoneNumber
   attribute string nightTelephoneNumber
}
```

这个ODL类定义对应于图12-9中的客户表。图12-15所示的每个类都会产生一个类似的ODL类定义。定义好每个类后，还需要定义类之间的关系。

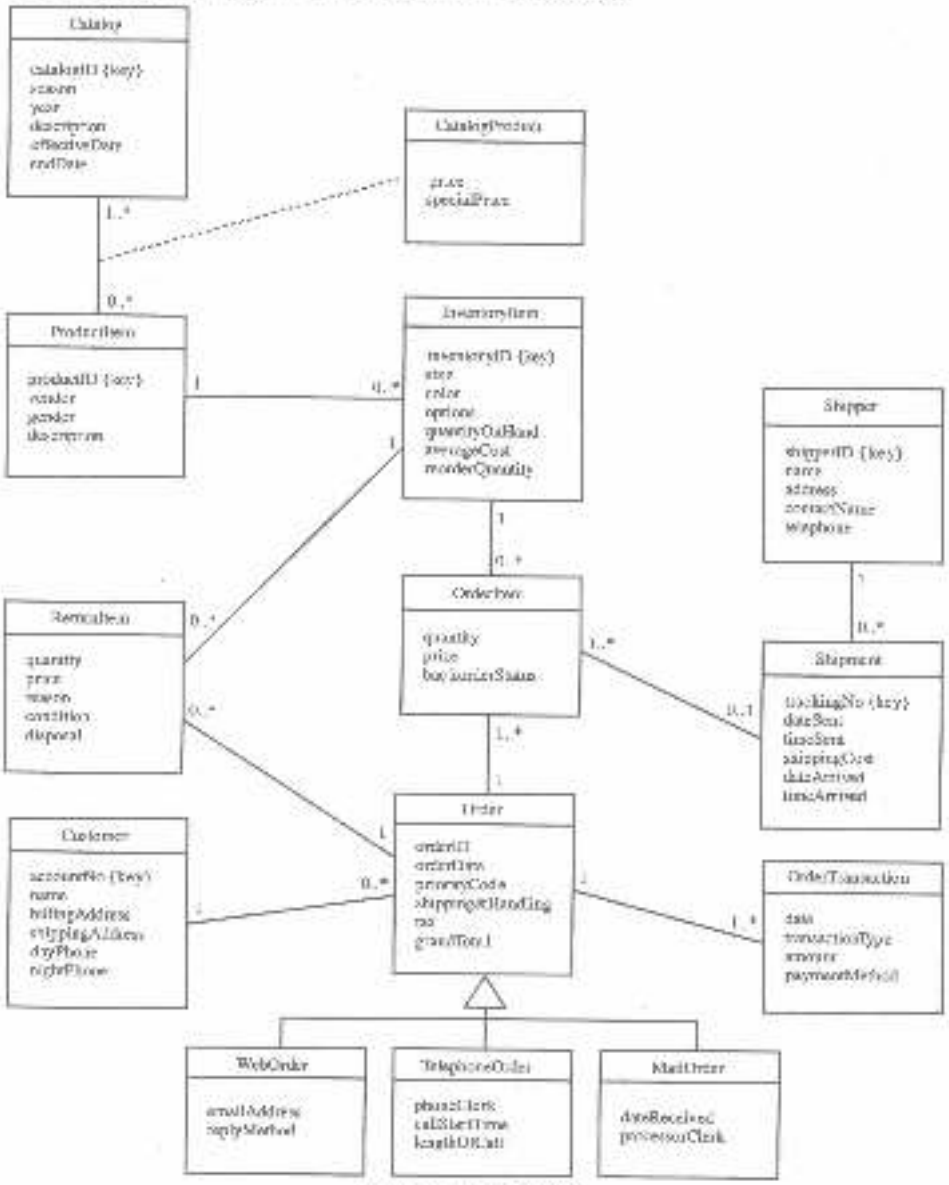


图12-15　RMO的类图

12.3.3 关系表示

在ODBMS中存储的每个对象都会被分配一个唯一的对象标识。这个标识可以是一个物理存储地址或在运行时能自动转换成物理地址的引用。不论哪种情况，每个对象的标识都可以存放在其他对象中以表示一个关系。

对象标识：物理存储地址或在运行时能自动转换成物理存储地址的引用。

ODBMS通过存储相关对象中一个对象标识来表示关系。对象标识是将相关对象连接在一起的纽带，正如第11章所描述的那样。例如，考虑图12-16所示的Employee类和Workstation类之间的一对一关系。每个Employee对象有一个属性叫computer，其中包含了分配给该雇员的工作站对象的对象标识。每一个Workstation对象有一个叫user的匹配属性，该属性包含了使用这个工作站的Employee的对象标识。



图12-16 用包含对象标识的属性表示的一对一关系

ODBMS用包含对象标识的属性来找到与其他对象相关的对象。这种从一个对象中提取对象标识并用它来访问另一个对象的过程有时被称之为定位。例如，考虑下面由用户提出的查询：列出分配给雇员Joe Smith的工作站的制造商。

定位：通过从另一个（相关）对象中提取对象标识的方法来访问对象的过程。

查询处理器可以通过在所有雇员对象中查找姓名属性为Joe Smith的对象来找到所要查询的雇员对象。它还可以通过使用存储在computer中的对象标识来找到Joe Smith的工作站。一个查询处理器还可以通过使用存储在User中的对象标识来回答相反的查询（列出被分配给某一特定工作站的雇员姓名）。一对匹配的属性对能使一个关系从两个方向进行定位。

表示关系的属性通常不是由对象数据库模式设计人员直接指定的。相反，设计人员通过声明对象间的关系间接指定它们。例如，考虑下面ODL模式语言的类声明：

```
class Employee{
  attribute string  Name
  attribute integer Salary
  relationship Workstation Uses
   inverse Workstation:: AssignedTo
}
class Workstation{
  attribute string manufacturer
  attribute string serialNumber
  relationship Employee AssignedTo
   inverse Employee:: Uses
}
```

关键字*relationship*用来声明一个类和另一个类之间的关系。Employee类和Workstation类有一个Uses关系。

Workstation类与Employee类有一个匹配关系叫AssignedTo。每个关系包含一个位于其他

类的匹配关系的声明，该声明由关键字*inverse*完成，而ODBMS就通过该关键字找到行为镜像的两个关系。

声明一个如上关系而不是建立一个包含对象标识的属性有两个好处：

- 由ODBMS承担确定如何实现对象间的联系的责任。实际上，模式设计人员已经声明了一个relationship类型的属性并由ODBMS为属性确定一个合适的类型。
- 由ODBMS承担维护参照完整性的责任。例如，删除一个Workstation会导致相关Employee对象的Uses链接被置为NULL或未定义。

ODBMS会自动产生用来表示已声明关系的属性，这些属性包含对象标识。但是关于标识是如何被实现和操作的细节，对用户和程序员来说是透明的。

一对多关系　图12-17给出了RMO中Customer类和Order类之间的一对多关系。一个客户（Customer）可以发出多个不同的订单（Order），但一个订单只能由一个客户发出。表示一个订单和一个客户之间的关系，需要一个单独的对象标识；表示一个客户和多个不同订单之间的关系，则需要多个对象标识，如图12-18所示。



图12-17　Customer和Order类之间的一对多关系

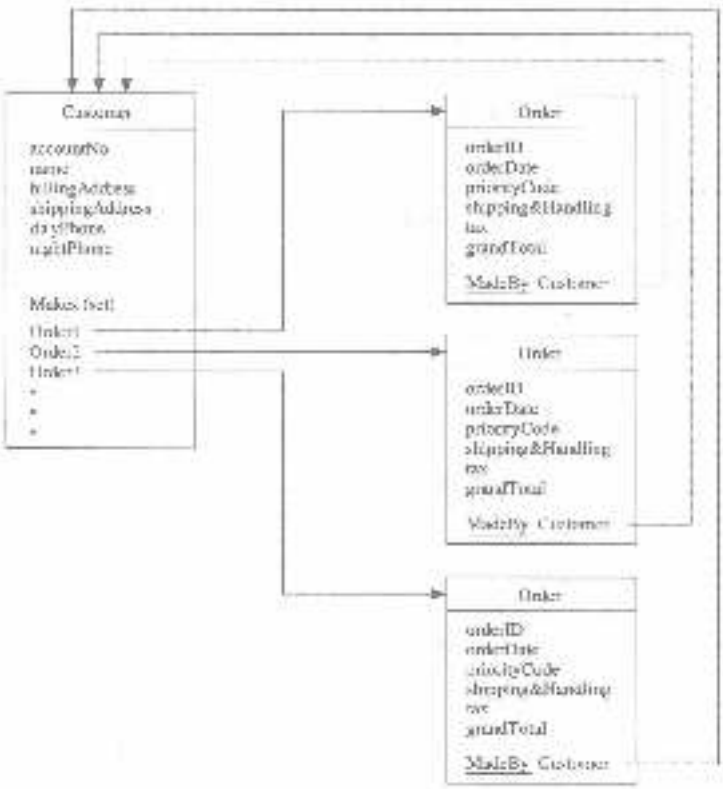


图12-18　用包含对象标识的属性表示的一对多关系

Customer类和Order类的部分ODL类声明如下所示：

```
class Customer{
  attribute string accountNo
  attribute string name
  attribute string billingAddress
  attribute string shippingAddress
  attribute string dayPhone
  attribute string nightPhone
  relationship set<Order>Makes
   inverse Order::MadeBy
}
class Order{
  attribute string orderID
  attribute string orderDate
  attribute string priorityCode
  attribute real shipping&Handling
  attribute real tax
  attribute real grandTotal
  relationship Order MadeBy
   inverse Customer::Makes
}
```

一个Customer对象和一组Order之间声明了Makes关系。通过将关系声明为一个集合，可以让ODBMS给每个Customer对象分配所需的Order对象标识属性来代表关系实例。随着关系实例的创建和删除，ODBMS将动态地添加和删除集合中的对象标识属性。

对象标识属性的集合也被称做多值属性，一个多值属性也叫做一个重复组，它是包含了零个或多个相同数据类型实例的属性。ODBMS通常都支持多值属性，而RDBMS却不支持，其原因是它违反了第1范式。

多值属性：包含零个或多个同一数据类型实例的属性。

多对多关系 多对多关系根据其是否含有属性，表示方式会有所不同。没有属性的多对多关系可由两个相关类的对象集属性表示，该对象集属性是类的一个多值属性，其值为指向相关类实例的对象指针，图12-19所示为Employee类和Project类之间多对多的关系。



图12-19 Employee类和Project类之间多对多的关系

```
class Employee{
  attribute string name
  attribute integer salary
  relationship set<Project>Workson
   inverse Project::Assigned
}
class Project {
  attribute string projectID
```

```
    attribute string description
    attribute string startDate
    attribute string endDate
    relationship set<Employee>Assigned
     inverse Employee:: WorksOn
}
```

表示含有属性的多对多关系稍微复杂一些。在RMO的类图中，Catalog类和ProductItem类之间存在一个多对多的关系，它们之间引入了一个叫CatalogProduct的类，如图12-15所示。在第5章中曾提到，关联类是用来存储多对多关系属性的类。

要想使用关联类来表示多对多的关系，必须改组图12-20所示的关系。Catalog 和ProductItem之间的多对多关系被分解成两个原始类和关联类之间的一对多关系，ODL模式的类声明如下：

```
class Catalog{
  attribute string season
  attribute integer year
  attribute string description
  attribute string effectiveDate
  attribute string endDate
  relationship set<CatalogProduct>Contains1
   inverse CatalogProduct::AppearsIn1
}
class ProductItem{
  attribute string productID
  attribute string vendor
  attribute string gender
  attribute string description
  relationship set<CatalogProduct>AppearsIn2
   inverse CatalogProduct:: Contains2
}
class CatalogProduct{
  attribute real price
  attribute real specialPrice
  relationship Catalog AppearsIn1
   inverse Catalog::Contains1
  relationship ProductItem AppearsIn2
   inverse ProductItem::Contains2
}
```



图12-20　用两个一对多的关系来表示多对多的关系

泛化关系　图12-21给出了从RMO的类图得到的一个泛化层次。WebOrder类、TelephoneOrder类和MailOrder类都是Order类的派生类。表示这些类及其关系的ODL类定义如下：

```
class Order{
  attribute string orderID
  attribute string orderDate
  attribute string priorityCode
  attribute real shipping&Handling
  attribute real tax
  attribute real grandTotal
}
class Weborder extends Order{
  attribute string emailAddress
  attribute string replyMethod
}
class TelephoneOrder extends Order{
  attribute string phoneClerk
  attribute string callStartTime
  attribute integer lengthOfCall
}
class MailOrder extends Order{
  attribute string dateReceived
  attribute string processorClerk
}
```

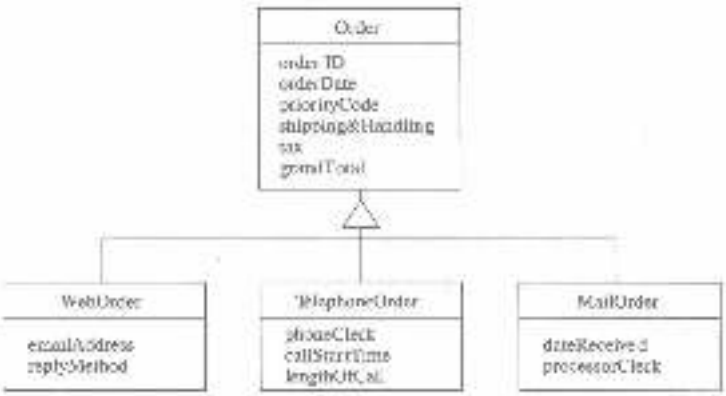


图12-21 RMO类图中的概略层次

关键字extends说明WebOrder、TelephoneOrder和MailOrder派生自Order。当存储在对象数据库中时，三个派生类的对象将继承Order类中所定义的所有属性、方法和关系。

关键属性

由于参照完整性是由对象标识实现的，因此关键属性在对象数据库中不是必需的。但是，关键属性在对象数据库中有多种用途，包括保证对象的唯一性并提供一种查询数据库内容的方法。ODBMS保证对象数据库中关键属性的唯一性。因此，声明一个关键属性可以防止两个对象拥有相同的键值。

除了关系数据库和面向对象数据库外，还存在融合了关系和面向对象这两种方法元素的第三种类型。下面我们将讨论这种混合类型。

12.4 混合对象-关系数据库设计

面向对象开发工具最早广泛应用于20世纪80年代中后期。在此期间，关系数据库管理系

继被广泛使用并达到一个成熟阶段。许多面向对象工具的开发者通过使用一个RDBMS来利用现存的RDBMS工具和知识存储持久对象状态。由于这是一个经济的方案（至多开发一个面向对象的工具），并且由于许多较新的面向对象系统需要对存储在现有关系数据库中的数据进行操作，因此很有意义。使用RDBMS来存储对象还没有一个通用的名字，所以我们打算自定义一个名称以便后文使用：混合对象-关系DBMS（或简称混合DBMS）。

混合对象-关系DBMS：一个用来存储对象属性和关系的关系数据库管理系统，简称混合DBMS。

混合DBMS方法是目前用来存储持久对象的最普遍的方法。设计一个混合数据库实际上是将两个问题变成一个。设计者必须开发一个完整的关系数据库模式，然后设计一组与其等价的类，以便在面向对象的程序中替代那些模式。后项任务比较复杂，因为设计者必须克服从面向对象的角度和从关系的角度存储数据的差异。

以下是存储数据的关系和面向对象视图的最大不同：

- 在RDBMS中，类方法既不能被直接存储也不能自动执行。
- 和RDBMS相比，ODBMS能表达更多的关系类型，包括继承关系以及整体-部分聚合关系，而在RDBMS中关系仅能用参照完整性来表示。
- 和RDBMS相比，ODBMS能表达更多的数据类型，可以定义新类来存储专用数据。

因为RDBMS先于面向对象范例被开发出来，所以它们不具有表示方法和继承的特性，访问数据库的程序必须在内部执行方法。因为不能直接表示一个分类层次，所以在RDBMS中不能直接表示继承。

虽然从关系的角度和从面向对象的角度存储数据有显著的不同，但它们也有很大一部分是重叠的。在第5章中，我们知道，一个系统中的"事物"在概念上能通过使用实体-联系图（关系数据库模式的基础）、类图（面向对象数据库模式的基础）或者两者的结合来进行建模。这两种表达方式有很大程度的重叠，包括：

- 将数据项聚合成实体或类；
- 定义实体或类之间的一对一、一对多和多对多关系。

这种重叠为在关系数据库表示的类和对象提供了一个基础。

12.4.1　类和属性

在RDBMS中，设计者可以通过定义适当的表来存储类和对象的属性，从而达到存储类和对象的目的。对于一个全新的系统，可以在类图的基础上设计一个关系模式，这跟在ERD基础上设计关系模式的过程是一致的。图12-22描述了OO、ERD以及关系数据库概念之间的对应关系。每个表对应一个类，每个字段对应类的一个属性，每一行则表示一个对象。

面向对象	实体-联系图	关系数据库
类	实体类型	表
对象	实体实例	行
属性	属性	列

图12-22　OO、ERD以及关系数据库存储数据的视图中概念的对应关系

每个表要选择一个关键字段（或字段组）。如前所述，设计者可从已存在的属性中选择或创建一个关键字段，也可以增加一个自定义的关键字段。表中的主关键字段需要保证表内的唯一性，并通过外键表示关系。

图12-23给出了一组关系数据库表，这些表用来表示例12-15中RMO类图中的类。图中给

新增了MailOrder、TelephoneOrder和WebOrder三个类和对Order类做了适应修改外，其他部分和图12-7中的定义是一致的。

表	属 性
Catalog	CatalogID, Season, Year, Description, EffectiveDate, EndDate
CatalogProduct	CatalogID ProductID, Price, SpecialPrice
Customer	AccountNo,Name,BillingAddress,ShippingAddress, DayTelephoneNumber, NightTelephoneNumber
InventoryItem	InventoryID, Size, Color, Options, QuantityOnHand, AverageCost, ReorderQuantity
MailOrder	MailOrderID,DateReceived,ProcessorClerk
Order	OrderID, OrderDate, PriorityCode, ShippingAndHandling, Tax, GrandTotal
OrderItem	OrderItemID, Quantity, Price, BackorderStatus
OrderTransaction	OrderTransactionID ,Date,TransactionType,Amount, PaymentMethod
ProductItem	ProductID, Vendor, Gender, Description
ReturnItem	ReturnItemID, Quantity, Price, Reason, Condition, Disposal
Shipment	TrackingNo, DateSent, TimeSent,ShippingCost, DateArrived,TimeArrived
Shipper	ShipperID, Name, Address, ContactName, Telephone
TelephoneOrder	TelephoneOrderID,PhoneClerk,CallStartTime,LengthOfCall
WebOrder	WebOrderID,EmailAddress,ReplyMethod

图12-23 带主键的表表（主键均以黑体表示）

12.4.2 关系

ODBMS用对象标识来表示对象间的关系，但是RDBMS并不建立对象标识，所以必须用外键来表示对象间的关系。RDBMS中的外键起着ODBMS中对象标识的作用，即外键提供了一个对象引用另一个对象的方法。

为表示一对多关系，设计者将关系中“一”侧类的主关键字段添加到“多”侧的类的表中。为表示多对多关系，设计者建立一个包含相关类的主键字段和关系本身属性的新表。请注意，表示对象间关系的方法同前面描述的表示实体间关系的方法是一致的。

图12-24通过增加表示图12-23所示关系的外键来扩展图12-15的定义。例如，Customer和Order类之间的一对多关系通过存储在Order表中的外键AccountNo来表示。Catalog 和ProductItem之间的多对多关系由包含外键CatalogID和ProductID的CatalogProduct表来表示。

除了表Order、MailOrder、TelephoneOrder和WebOrder（这些表将简单介绍）中的内容外，图12-24与图12-9是一致的。它们的相似不是偶然的，这源于RMO的ERD和类图之间的相似性。因此，从表示同一个基本实体的类图和ERD得到的关系数据库模式相似就不足为奇了，事实上，要是它们不相似倒奇怪了。甚至还可能预示着一个错误。

分类关系，例如Order、MailOrder、TelephoneOrder和WebOrder之间的关系，是关系数据库设计中的特殊的实例。同子类继承父类的数据和方法一样，表示子类的表从表示其父类的表中继承部分或所有的数据。这种继承有如下两种方式：

- 将所有的表组合为一张单独的表，包含所有类的属性，但不包含子类的自定义主键。
- 使用多个单独的表表示那些子类，并用父类表的主键来代替子类表的自定义主键。

上述任何一种表示分类关系的方法都是可接受的。

图12-9给出了使用第一种方式的Order表的定义。MailOrder、TelephoneOrder和WebOrder表中所有的非关键字段都被添加到Order表中。对于任何一种特定的订单，每一行的一些字段

值都将被赋为NULL。例如，一个表示电话订单的行在EmailAddress、ReplyMethod、DateReceived和ProcessorClerk的值为NULL。

表	属　性
Catalog	**CatalogID**, Season, Year, Description, EffectiveDate, EndDate
CatalogProduct	**CatalogID**, **ProductID**, Price, SpecialPrice
Customer	AccountNo,Name,BillingAddress,ShippingAddress, DayTelephoneNumber, NightTelephoneNumber
InventoryItem	**InventoryID**, *ProductID*, Size, Color, Options, QuantityOnHand, AverageCost, ReorderQuantity
MailOrder	**OrderID**,DateReceived,ProcessorClerk
Order	**OrderID**, *AccountNO*, OrderDate, PriorityCode, ShippingAndHandling, Tax, GrandTotal
OrderItem	**OrderItemID**, *OrderID*, *InventoryID*, *TrackingNo*, Quantity, Price, BackorderStatus
OrderTransaction	**OrderTransactionID**, *OrderID* ,Date, TransactionType, Amount, PaymentMethod
ProductItem	**ProductID**, Vendor, Gender, Description
ReturnItem	**ReturnItemID**, *OrderID*, *InventoryID*,Quantity, Price, Reason, Condition, Disposal
Shipment	**TrackingNo**, *ShipperID*, DateSent, TimeSent, **ShippingCost**, DateArrived,TimeArrived
Shipper	**ShipperID**, Name, Address, ContactName, Telephone
TelephoneOrder	**OrderID**,PhoneClerk,CallStartTime,LengthOfCall
WebOrder	**OrderID**,EmailAddress,ReplyMethod

图12-24　通过添加外键属性（用斜体表示）将关系信息添加到类表中

图12-24给出了用第一种方式表示继承的RMO实例的表的定义。三个子订单类型和父Order表之间的关系由三个子类表中公用的外键OrderID表示。请注意，每个表中的自定义关键字已经被取消了。因此，在每个事例中，表示继承关系的外键又是表示子类的表的主键。

12.4.3　数据访问类

在第11章中，我们已经学过怎样基于三层结构开发一个面向对象设计。在那种结构下，数据访问类担当存储在对象以及关系数据库中数据之间的桥梁。

图12-25展示了RMO问题域类ProductItem、数据访问类ProductItemDA以及关系数据库之间的交互作用。数据访问类含有能够添加、更新、查找及删除与类相对应表中的字段和记录的方法。数据访问类方法封装了把问题域类中的值复制到数据库中或把数据库中的值复制到问题域类中所需的逻辑。通常，这些逻辑是用一种编程语言（如C++或Java）编写的代码和用结构化查询语言（SQL）编写的嵌入式关系数据库命令的结合。

图12-25左下方是一段嵌入了SQL语句的Java代码，用来实现ProductDA中的find()方法。数据访问类的其他方法也通过类似的代码实现。

既然我们已经介绍了多种不同的设计数据库模式方法，现在来考虑存储在模式中的数据类型。

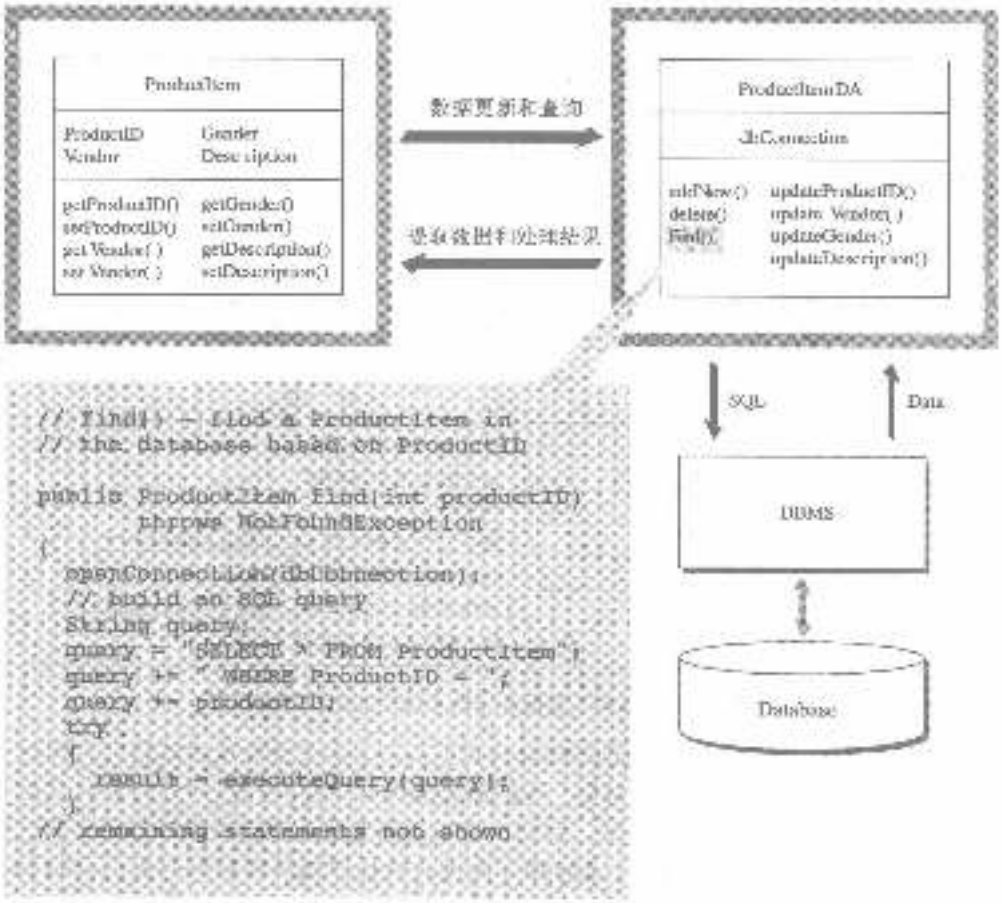


图12-25 问题域类、数据访问类以及数据库管理系统之间的交互作用

12.5 数据类型

数据类型定义了程序变量、对象状态变量、数据库字段或属性的存储格式和允许值。基本数据类型是由计算机硬件或程序设计语言直接支持的数据类型，它包括内存地址（指针）、布尔型、整型、无符号整型、短整型（一个字节）、长整型（多个字节）、单个字符、实数（浮点数）、双精度整数（两倍长度）、双精度实数。许多面向过程的程序设计语言（如C语言）以及大多数面向对象的程序设计语言，都允许程序员利用基本数据类型构造其他的数据类型。

数据类型：字段、类属性或程序变量的存储格式和允许值。

基本数据类型：由计算机硬件或一种程序设计语言直接实现的存储格式。

随着信息系统的日趋复杂，实现它们所需的数据类型也越来越多。现代数据类型包括日期型、时间型、货币型、音频流、视频图像、动画视频流、统一资源定位（URL或Web链接）。因为它们通常被定义为基本数据类型的复杂组合，所以这样的数据类型也叫做复杂数据类型。而由于这些复杂的数据类型可以由用户在分析和设计过程中或由程序员在设计和实现过程中定义，所以也被称做用户定义数据类型。

复杂数据类型：不能由计算机硬件或程序设计语言直接支持的数据类型，也叫用户定义数据类型。

12.5.1　关系DBMS的数据类型

设计者必须为关系数据库模式中的每个字段选择一个合适的数据类型。对许多字段而言，数据类型的选择相对来说比较简单。例如，设计者可以用一组固定长度或可变长度的字符数组来表示客户名和地址，库存量和商品单价可以分别由整型和实型来表示，而颜色可以由包含颜色名的字符数组或表示基本颜色红、蓝、绿强度的三个整数的集合来表示。

现代RDBMS不断增加新的数据类型来满足现代信息系统的需求。图12-26是Oracle RDBMS中部分数据类型的列表。Oracle 中的复杂数据类型包括：DATE，LONG和LONGRAW，LONG常用来存储大量格式化或非格式化的文本（例如，一个字处理文件）；LONGRAW用来存储大量二进制数值，包括编码后的图像、声音和动画。

类　型	描　述
CHAR	固定长度的字符数组
VARCHAR	可变长度的字符数组
NUMBER	实数
DATE	特定当有效性检验的日期和时间数据类型
LONG	最长可达2GB的变长字符数据
LONGRAW	不限制格式和内容的二进制大对象（BLOB）
ROWID	某一的6字节长的物理存储地址

图12-26　Oracle关系数据库管理系统中的部分数据类型

在将数据存入数据库时，现代RDBMS会对其进行许多有效性和格式的检查。例如，模式设计者可以规定要处理的数量不能为负，美国邮政编码必须为5位或者9位的数字以及一个包括URL的字符串必须以“http://”开头。这样，使用数据库的所有应用程序自动具享这些有效性和格式限制，程序变得更为简单，而且由于数据有效性逻辑不匹配而造成错误的可能性基本消除。当然应用程序仍需提供程序逻辑，以他之从试图添加“错误”数据中恢复过来，但它们确实从有效性检查中解脱出来了。

12.5.2　对象DBMS的数据类型

ODBMS通常提供一组与RDBMS相对应的基本和复杂数据类型。ODBMS也允许模式设计者定义格式约束和值域，但ODBMS提供了一个更为强大的方法来定义有用的数据类型和约束。模式设计者可以将一个新数据类型以及它的相关约束定义为一个新类。

类是一个很复杂的用户自定义数据类型，它将数据以及处理这些数据的过程（方法）结合起来。在许多面向对象的程序设计语言中，程序员可以扩展已定义的数据类型以设计出新的数据类型（类）。设计者可以设计特定的类来满足系统对数据类型的要求，因此以往需求和有限的数据类型之间的矛盾不复存在。对ODBMS来说，新数据类型的实例只不过是一个存储在数据库中的对象。

类的方法可以执行许多以前由应用程序代码和（或）DBMS本身所执行的类型和错误检查。事实上，程序员构造了一个“用户设计”的数据类型以及正确使用该类型所需的程序逻辑。DBMS不再直接管理复杂数据类型和存储在其中的值，它通过提取和执行存储在数据库中由程序员定义的方法来间接执行对该类型的有效性检查和格式转换。

面向对象工具如此广泛运用于非业务信息系统的一个主要原因是，它定义新的数据类型的灵活性。在工程、生物和物理等领域中，使用的数据往往比简单的字符串、数字和日期更复杂，面向对象工具允许数据库设计者和程序员针对某一特定问题自定义数据类型。

在数据库设计阶段，另一个必须要考虑的问题是数据存储和访问的地址。在如今的网络信息系统中，各组织还常使用分布式数据库。

12.6 分布式数据库

很少有组织将它全部的数据存储在单个数据库中。相反，数据通常被存储在多个不同数据库中，由多个不同的DBMS控制。使用多个数据库和DBMS的原因如下：

- 信息系统可能是在不同时期使用不同的DBMS开发的；
- 组织的各个部门拥有和管理自己的数据；
- 数据与使用它们的应用程序在物理上紧密联系，能提高系统性能。

12.6.1 分布式数据库体系结构

第9章描述了在网络环境中组织和计算信息处理资源的各种方法。分布式数据库服务有以下几种体系结构：

- 单个数据库服务器
- 备份数据库服务器
- 分区数据库服务器
- 联合数据库服务器

这些体系结构的组合结构也是可以的。

1. 单个数据库服务器

图12-27所示的是一个典型的单个数据库服务器的体系结构。一个或多个局域网中的客户共享一个单独计算机系统中的单个数据库。这个数据库服务器可以连接到其中的一个局域网上或直接连到广域网的主干线上（见图12-27）。直接连接到广域网上确保了不会由于与数据库服务器之间的网络传输而造成某个局域网的超载。

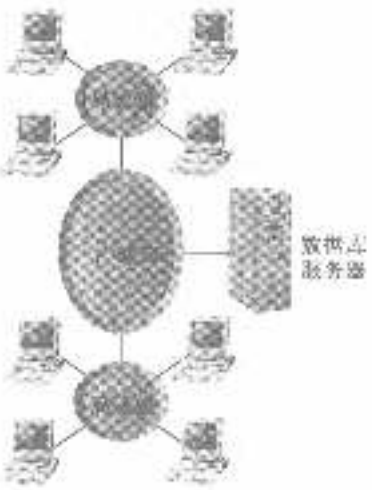


图12-27 单个数据库服务器体系结构

单个数据库服务器结构的最主要的优点是简单。它只管理一个服务器，并且所有的客户直接请求服务器。它的缺点包括对服务失败的敏感性以及网络和服务器发生超载的可能性。如果服务失败，单个服务器不能提供备份能力。一旦服务器无法使用（例如系统崩溃或硬件维护期间），所有依赖服务器的应用程序都将不能工作。因此，单个数据库服务器结构不适合需要一周7天，一天24小时运行的应用。

性能瓶颈可能会发生在单个数据库服务器或与服务器相连的网络段上。当交易数目增多时，单个数据库服务器的性能可能不足以对它所收到的请求做出迅速反应。为提高它的性能，设计者可以使用一个功能更强的计算机系统作为数据库服务器。但是在海量数据存储的今天，大型数据库的大小和交易量超过单个计算机系统的能力是非常常见的。使用超大型机可能会由于费用、系统管理或网络性能的考虑而不切实际。

对数据库服务器的请求和响应可能会跨越局域网或广域网的很长一段距离。数据库事务处理必须和网络上其他类型的网络传输（例如声音、图像和网站访问）竞争以获得可用的传输能力。因此，对一个远端服务器访问时的延迟可能来自于网络拥塞或从客户端到服务器的

传输延迟。

减少网络拥塞的一个方法是提高整个网络的容量。但这个方法的代价较高，并且常常是不可行的。另一个特别适用于提高数据访问速度的方法是在物理上将数据库服务器靠近客户端（例如，服务器和客户端在同一个LAN网段里）。这个方法使距离造成的请求和响应延迟减到最小并大大减小广域网的通信量。

当所有的客户端很集中时，将数据库服务器靠近客户端相对来说是件简单的事情。但当客户端很分散时，像今天的跨国公司，那又该怎么办呢？在这种情况下，没有哪个单一的数据库服务器位置能够同时提高所有客户的数据库访问能力。因此，相对较远的客户在对数据库进行访问时必须付出更大的代价。

2. 备份数据库服务器

设计者可以通过使用备份数据库服务器结构（如图12-28所示）来减小对远端数据库服务器访问时的延迟。每一个服务器放在和一组客户很近的位置，并对所需的数据保存一个单独的拷贝。客户端配置成与本地域网中的数据库服务器交互。消除来自广域网的数据库访问并且使得传输的延迟最小化，本地网络和服务器的性能可以根据本地的需要而进行独立的优化。

备份数据库服务器还使得信息系统具有更高的容错能力，可以设计应用程序对任何可用的服务器进行直接存取请求，特别是距离最近的服务器。当一个服务器不可用时，客户访问可以自动转向另一个可用的服务器。因此，当一个本地数据库服务器不可用时，访问可通过广域网改变方向。设计者还可以通过在客户端和备份数据库服务器之间放置一个事务服务器来获得负载平衡。事务服务器监视所有数据库服务器的负载并自动将客户请求转到负载最低的服务器上。

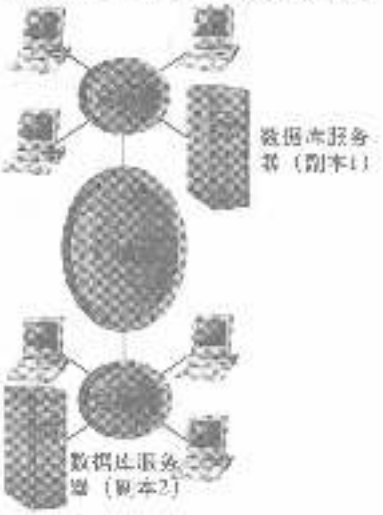


图12-28　备份数据库服务器体系结构

尽管备份数据库服务器有这些优点，但是也有一些不足之处。无论什么时候，使用多个数据库拷贝时，数据之间的不一致性就成了问题。当一个数据库副本中的数据更新时，到另一个数据库副本中相同数据的客户访问可能会收到过时的应答。为克服这个问题，对每个数据库副本进行更新时也必须定期对数据库的所有其他副本进行同样的更新。这个过程叫数据库同步。

数据库同步：保证两个或多个数据库副本之间一致性的过程。

设计者可以通过开发自定义的同步程序或使用DBMS中内置的同步程序来实现同步。因为自定义的程序开发困难，并且当数据库模式或数据库副本的数目和地址改变时，这些程序也要进行调整。所以很少使用。DBMS同步程序功能强大且灵活，但也比较昂贵。DBMS用来实现同步的方法间的不相容使得同时使用不同厂商的DBMS是不切实际的。

更新某个数据库副本和将更新操作传给其他副本之间的时间延迟是数据库设计时要考虑的一个重要因素。在最初的更新和对副本更新之间的这段时间里，访问过时副本的应用程序不会收到反映当前实际情况的响应。设计者可以通过减少同步延迟来解决这个问题。但越短的延迟意味着更频繁（或可能是连续的）的数据同步。同步会消耗数据库服务器相当大的一部分资源，而且相关数据库服务器要求大量的网络资源。合适的同步策略是费用、硬件、网络能力以及应用程序和用户对当前数据的需要之间的复杂平衡。

3. 分区数据库服务器

设计者可以通过将数据库内容划分到多个数据库服务器上来最小化数据同步的需求。图12-29给出了将一个假想的数据库模式分成两部分的一个划分，每一部分被不同的客户组访问。图12-30是一个分区数据库服务器结构，每一部分在一个单独的数据库服务器上。每组中客户和数据库服务器间的交互被限制在一个局域网中。

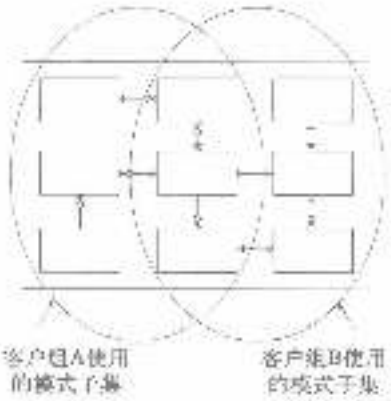


图12-29 将数据库模式划分成多个客户访问子集

仅当一个模式可以清晰地在客户访问组间划分时，分区数据库服务器结构才是可行的。首先，客户组必须要求访问明确的数据库子集（例如，市场数据而非生产数据）。此外，一个客户组的成员必须处于一个小的地理区域中。当一个客户组分布在多个地理位置（例如，三个地区中心的订单处理）时，通常要求把备份和分区数据库服务器结合起来使用。

很少将数据库模式分解成相互独立的子块。数据库中的一些部分常常被大多数有时甚至是所有的用户使用。这些部分就必须存在于每个分块中。例如图12-29中重叠的区域。这一区域中的数据库内容将存在于所有服务器上，并且它们的内容需要定期同步。因此，分区可以减少与数据库同步有关的问题，但它很少能完全消除这些问题。

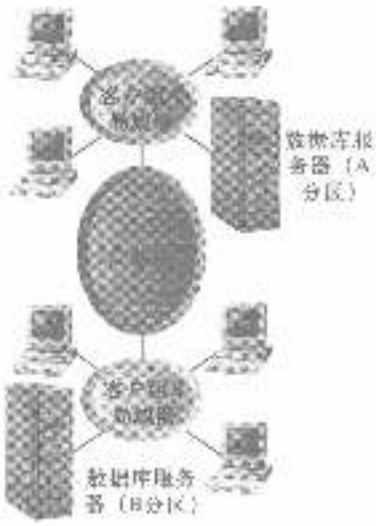


图12-30 分区数据库服务器体系结构

4. 联合数据库服务器

一些信息系统得益于如图12-31所示的联合数据库服务器结构的很好运作。这个结构一般用来访问存储在具有不兼容性的存储模型或DBMS的数据库中的数据。可以在一个联合的数据库服务器上建立一个单独且统一的数据库模式。这个服务器在应用程序和其他服务器上的数据库之间起中介作用。数据库请求首先被发送给联合数据库服务器，它会依次对下层的数据库服务器进行适当的请求。在系统给客户端返回响应之前，来自多个服务器的结果会被结合起来并重新格式化以满足统一的模式。

联合数据库服务器的结构可能会相当复杂。许多DBMS产品都可以用来实现这样的系统，但它们的代价很高，实现困难，且不易维护。联合数据库结构对计算机硬件和网络容量要求很高。但与应用程序直接跟所有下层数据库交互的实现和维护相比，它的代价和管理复杂度还是降低了。

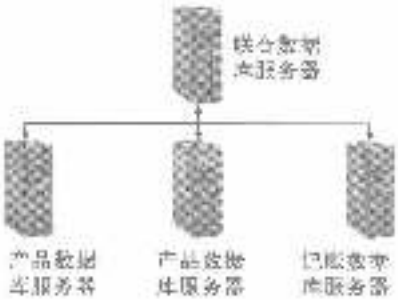


图12-31 联合数据库服务器体系结构

联合数据库服务器结构一般用来实现数据仓库。数据仓库是用来支持结构化或非结构化管理决策的数据集合。数据仓库一般从组织中运转的数据库和多个外部数据库中提取数据（例如政府、业务部门协会以及私人研究机构管理的数据库中的经贸数据）。因为数据来源于许多不相容的数据库，联合结构通常是实现数据

仓库的唯一可行办法。

数据仓库：用来支持结构化或非结构化管理决策的数据集合。

我们已经讨论过支持分布式数据库设计的基本问题。下面我们给出当给RMO新的客户支持系统做决策时这些是如何运作的。

12.6.2　RMO分布式数据库体系结构

设计数据库结构的出发点是收集地理上分散的用户的数据需求信息。RMO公司的这类信息有一些在分析阶段就已经收集好了（见图6-37、图6-38和图6-39），现总结如下：

- 仓库员工（波特兰、格瑞城和阿尔伯克基）要具有审核库存水平、查询订单、记录返回订单和订单实现以及记录订单返回的能力。
- 电话订购中心职员（盐湖城）要具有审核库存水平，建立、查询、更新和删除订单，查询客户账号信息以及查询目录的能力。
- 邮购中心职员（普罗沃）要具有审核库存水平、查询订单、查询目录信息和更新客户账号的能力。
- 直接客户访问（地点还没确定）要具有与电话订购中心职员同样的能力。
- 总部职员（纳克城）要具有查询和调整订单、客户账户信息，以及建立和查询目录、促销商品的能力。

RMO公司已决定使用纳克城数据中心现有的主机来管理它的数据库。因此，要求有一个广域网来将服务器连接到仓库、电话订购中心、邮购中心、总部和数据中心的局域网上。最后还要连接上用于直接客户订购的Web服务器。虽然这些Web服务器可能会放在一个已存在的位置上（例如数据中心）。

图12-32给出了为RMO设计的单个服务器数据库体系结构。这种结构要求广域网有足够的容量来满足来自各个结点的数据库（或其他）通信量。这种结构的主要优点是它的简单性。它不需要管理分区或数据库副本，只需要维护一个单独的服务器。其主要缺点是，对广域网容量的要求相对较高以及整个系统对服务器故障较敏感。

图12-33给出的是一个较复杂的候选方案。在每个远端结点上联合使用数据库分区和数据库备份的方法。每一个仓库有一份订单和部分数据库库存清单的本地副本。电话订购和邮购中心有较大的数据库子集的本地副本。公司总部依赖于数据中心的中心数据库服务器。

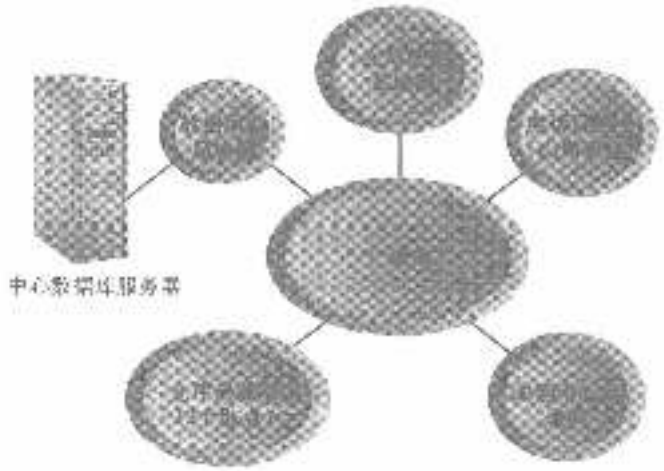


图12-32　一个为RMO设计的单个服务器数据库体系结构

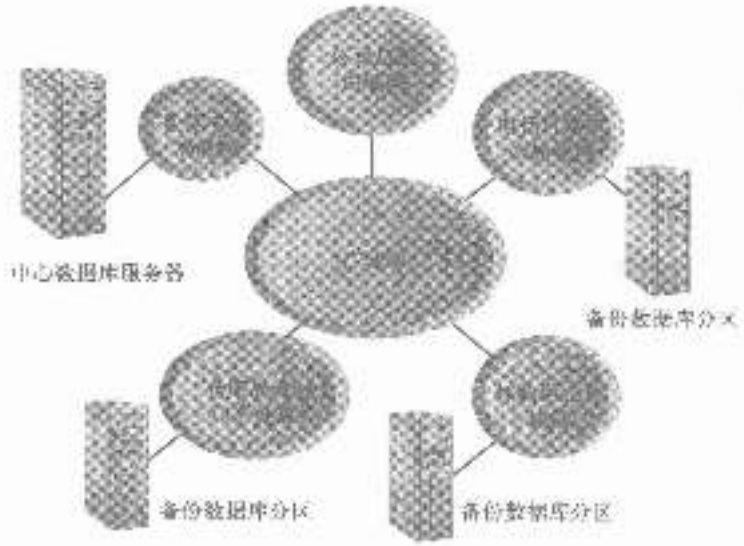


图12-33 为RMO设计的备份和分区数据库服务器体系结构

这种体系结构的主要优点是，它具有容错能力而且对广域网容量的需求有所降低。如果中心数据库服务器出现故障的话，每一个结点仍然能够继续工作。但是，在远端节点连续工作的同时，它们的数据库内容会渐渐偏离同步。当此必须执行同步策略进行数据库的定期更新和从服务器故障中恢复。这种策略可以因结点的不同而有所不同。

分布式结构的主要缺点是成本高和复杂。这种结构通过降低容量需求来减少广域网的费用，但是它增加了附加的数据库服务器的费用。附加服务器的购买、操作和维护的费用可能比提高广域网容量的费用更高。

那么，这两种结构中哪一种更适合RMO呢？答案取决于一些还没有被收集的数据和所期望的系统性能等问题。RMO的管理人员还必须确定系统性能和可靠性的目标。这种分布式结构将会提供更高的性能和可靠性，但这样做会使得费用显著增加。管理人员必须对额外的花费与期望的利益进行比较，考虑这种花费是否值得。

网络通信的附加数据用来精确地确定客户端和数据库服务器之间的局域网和广域网的通信需求，对每个结点要进行交易和查询量的估计。这些估计必须包括正常时期和高峰需求时期。这些数据可以在分析和设计阶段收集，可用来确定局域网、广域网和数据库服务器容量的最优配置。有一点要注意，对数据的分析以及对网络和数据库结构的实际设计是复杂的工作，它需要很高的专业知识和经验。在条件允许的情况下，应该在实际的运行环境下对以上估计进行测试和验证。Barbara Halifax描述了RMO项目中这样的一个测试。

2007年3月20日

To：John MacMurty

From：Barbara Halifax

RE：客户支持系统（CSS）数据库的基准测试

John，我们已经接近第二次迭代的结束，现在我向你提供一些关于数据库设计和测试的细节信息。你应该还记得，在第二次迭代中，我们在运行MVS和DB2的IBM服务器上创建了一个用于测试的数据库。尽管这个服务器很小且不能用于实际产品的数据库，但是用于开发和测试"输入订单"功能的数据访问逻辑已经足够了。在下次迭代中，我们会测试

其他功能，但常规是，这些功能的数据库模式和数据访问逻辑不应有显著变化。

目前，我们到了项目测试和部署的紧要关头。当供应链管理（SCM）系统设计完成后，项目小组提供了客户支持系统（CSS）所需的网络和数据库能力。但是，我们需要对核心的数据库结构进行实际的性能测试，看看我们的评估是否准确，详细计划如下。

各周计划：

4月2～6日　制作一个production数据库的备份，该备份用于SCM与CSS的联合测试。将SCM的production数据库中3个月内的数据转移到测试数据库中。

4月9～13日　修改数据库的模式，使其包含到目前为止所开发的新表。从原系统的客户数据库中提取10000条最近3个月的客户记录以及他们的订单记录到测试数据库中。

4月16～20日　修改CSS应用程序和Web服务器，使其能与主机上的测试数据库交互。在高级的平上做一个负载性能测试，在午饭前分析得到的结果。如果结果没有什么问题，那么，下午在系统负载和网络负载达到顶峰的时候，再做一个负载性能测试。

4月23～27日　仔细分析两次测试的结果，如果发现比较严重的问题，就对数据库和网络做相应调整，并在周五再次对其进行测试。

当然，我们希望4月20日那天一切顺利，这样的话我们就不需要4月27日的又一轮测试。当额外的软件部署到Web和应用服务器，以及主机上的测试数据库发生相应改变的时候，我们需要做负载测试。我们希望在接下来的几次迭代中，每3周便进行一次数据库测试。

BH

cc：Steven Deerfield, Ming Lee, Jack Garcia, Ann Hamilton

小结

大部分现代信息系统将数据存储在数据库中，并用数据库管理系统（DBMS）访问和管理数据。今天，关系数据库和DBMS得到了广泛的使用，但对象数据库和其DBMS也越来越流行。系统设计的主要活动之一是关系或对象数据库模式的设计。

关系数据库是一个存储在表中的数据的集合。关系数据库模式通常是从一张实体-联系图发展而来的，每一个实体表示为一张单独的表。一对多关系通过嵌入在实体表中的外键来表示，多对多关系通过创建额外的、包含相关实体外键的表来表示。

对象数据库以相关对象的集合存储数据。设计类图是设计对象数据库模式的起点。数据库模式定义了每一个类，而ODBMS将每个实体作为一个特定类的实例来存储。每个对象被分配了一个唯一的对象标识，对象间的关系通过存储相关对象中一个对象的对象标识来表示。

对象也可以存储在一个关系数据库中，对象属性和对象之间的关系——包括一对多、多对多和泛化关系都可以被表示，但是方法不能被存储，继承也不能被直接表示。

中大规模的信息系统通常使用分布在不同地理位置上的多个数据库或数据库服务器。备份数据库体系结构通常在不同地理位置的不同服务器上使用多个数据库副本。分区数据库体系结构使用不同用户子集附近的不同服务器的部分拷贝。联合数据库体系结构使用多个数据库（可能是不同类型的）和一个能够提供一致的数据库视图和单独访问点的专用的DBMS。

关键术语

complex data type　　复杂数据类型

database (DB)	数据库（DB）
database management system (DBMS)	数据库管理系统
database synchronization	数据库同步
data type	数据类型
data warehouse	数据仓库
field	字段
field value	字段值
foreign key	外键
first normal form (1NF)	第1范式（1NF）
functional dependency	函数依赖
hybrid object-relational DBMS	混合对象-关系数据库管理系统
key	关键字
navigation	定位
normalization	规范化
multivalued attribute	多值属性
object database management system (ODBMS)	对象数据库管理系统
object definition language (ODL)	对象定义语言
object identifier	对象标识
persistent object	持久对象
physical data store	物理数据存储
primary key	主键
primitive data type	基本数据类型
referential integrity	参照完整性
relational database management system (RDBMS)	关系数据库管理系统
row	行
schema	模式
second normal form (2NF)	第2范式（2NF）
table	表
third normal form (3NF)	第3范式（3NF）
transient class	暂存类

复习题

1. 列出数据管理系统的各个组成部分以及各部分的用途和功能。
2. 什么是数据库模式？它包含哪些信息？
3. 为什么数据库成为目前最受欢迎的信息系统存储数据的方法？
4. 列出目前常用的4种不同类型的数据库模型和数据库管理系统。现在，用得最广泛的是哪一种？
5. 针对关系数据库，简要定义术语行和域。
6. 什么是主键，主键允许重复吗？为什么？
7. 系统自动产生的关键字和自定义的关键字有何不同？业务信息处理中最常使用的是哪一种？
8. 什么是外键？关系数据库中为什么要使用外键？外键允许重复吗？为什么？
9. 列出将实体-联系图（ERD）转换成关系数据库模式的步骤。

10. 在关系数据库中，ERD上的实体如何表示？
11. 在关系数据库中，ERD上的一对多关系如何表示？
12. 在关系数据库中，ERD上的多对多关系如何表示？
13. 什么是参照完整性？当建立一个新的外键值时，当一行的主键被删除或改变时，如何实施参照完整性？
14. 在关系数据库中，哪些类型的数据（或字段）只能存储一次？哪些类型的数据（或字段）必须存储多次？
15. 什么是关系数据库的规范化？为什么一个第3范式的数据库模式被认为比一个未规范化的数据库模式质量要高？
16. 描述关系数据库规范化的过程，哪些范式依赖于函数依赖的定义？
17. 列出将一个类图转换成对象数据库模式的步骤。
18. 持久对象和暂存对象有何不同？分别举几个例子（至少一个）。
19. 什么是对象标识？为什么在对象数据库中要使用对象标识？
20. 在对象数据库中，类层中的类是如何表示的？
21. 在对象数据库中，类层名一对多关系是如何表示的？
22. 在对象数据库中，不带属性的多对多关系是如何表示的？
23. 什么是相关类？在对象数据库中如何用相关类来表示多对多关系？
24. 描述在对象数据库中表示泛化关系的两种方法。
25. 对象数据库中要求有关键字段或属性吗？为什么？
26. 描述基于同一基本实体的类图和实体—联系图之间的相同和不同之处。
27. 在关系数据库中，如何表示类图中的类？
28. 基本数据类型和复杂数据类型之间有何不同？
29. 使用RDBMS提供复杂数据类型的优点是什么？
30. ODBMS需要提供预定义的复杂数据类型吗？为什么？
31. 为什么整个或部分数据库需要在多个地方进行备份？
32. 简要描述下列分布式数据库体系结构：备份数据库服务器、分区数据库服务器和联合数据库服务器。
33. 当数据库内容有多个备份时，会引入什么额外的数据库管理复杂性？

思考题

1. 通用产品代码（UPC）是用于唯一标识在美国销售的商品的条形码数字。比如，在美国销售的这本教科书的所有副本，其封底都有同样的UPC条形码。现在考虑一下，如果RMO销售的所有商品在法律上要求有一个永久的UPC（例如衣服上的标签），那么该如何更改RMO的数据库设计？RMO的关系数据库模式又该怎么更改？
2. 假设RMO计划改变它的价格政策。如果同时流通两份或更多的价格表，那么价格表中各项和包的价格应该是一致的，随着时间的流逝，价格仍可以上下浮动，并且这些变化都会记录在数据库中并且打印在最近发布的价格表上，所有要订购的顾客，都会得到目前最低的价格或最新价格表中的价格。如何修改图12-9，以保证价格政策变化后，数据库是3NF的？
3. 假设RMO将开始对电话订购商品的顾客进行随机抽样调查，询问他们关于从竞争对手购买商品的情况。作为顾客回答几个问题的回报，RMO将在当前价格的基础上给这些顾客15%的优惠。为了存储和使用这些信息，需要对ERD和类图进行扩展，添加两个新的实体（类）和三个新的关系。新的实体是Competitor和ProductCategory。Competitor和ProductCategory之间存在一对

多的关系，已存在的Customer实体（类）和ProductCategory之间也存在一对多关系。Competitor有4个字段（属性）：Name、ProductCategory有4个字段（属性）：Description、DollarAmountpurchased、MonthPurchased和YearPurchased。修正图12.9中的关系数据库模式，使之将新的实体和关系包含进去。所有的表格要求是3NF的。

4. 假设RMO应面向对象方法扩展它的数据库。进一步假设，数据库设计者想要对图12-15中的类图做一些改变，特别是，他们要从其他具体的产品类中抽象出ProductItem作为一个父类。需要加入三个具体的类：ClothingItem、EquipmentItem和OtherItem。ClothingItem需要增加Color属性。相同的属性要从InventoryItem中删除。EquipmentItem也需要增加一个Color属性，但它不需要Gender属性。OtherItem既需要Color属性，也需要Gender属性。修正图12-24中的关系数据库模式来存储新的ProductItem的泛化层次。为每个具体的类建立一个单独的表。
5. 假设RMO将使用图12-9所示的关系数据库。进一步假设，将由位于意大利米兰的一个新的目录小组对目录进行创建和维护。为最小化网络耗费，目录组将使用一个连接到局域网的专用的数据库服务器。设计一个方案来划分RMO数据库。哪些表需要在目录的本地数据库服务器上备份。更新图12-33，给出一个新的分布式数据库结构。
6. 重新考虑在本章一开始就提出的全国图书公司（NB）案例。NB应该采用ODBMS来实现网络订购系统吗？为什么？

实验练习

1. 本章没有详细讨论网络数据库。但许多数据库书课本中都做了介绍。研究一下网络数据库模式以及指针表示记录间关系的使用方法。在网络数据库中使用指针与在对象数据库中使用对象标识在哪些方面是相似的？这种相似性是否表示对象数据库仅仅是旧的DBMS技术的一个换名版本？
2. 访问对象数据库管理组网站（http://www.odmg.org），收集关于ODMG标准现状的信息。
3. 回忆你所在学校的学生管理系统。确定所使用的是什么数据库管理系统。DBMS使用什么数据模式？如果DBMS不是面向对象的，找出正打算使用什么方案（如果有的话）将它移植到ODBMS。为什么要进行这样的移植？
4. 访问与RMO有类似在线产品目录的供货商网站（例如http://www.llbean.com）或计算机及相关配件的在线厂商（例如http://www.cdw.com）。浏览在线商品目录并注意包含其内的各种信息类型。构造一个用来存储所有在线目录信息的复杂数据类型列表。

实例研究

房地产多编目服务系统

查阅在第5章实例研究中的房地产多编目服务系统的描述，使用实体-联系图和类图作为该系统的起点。

1. 设计一个3NF的关系数据库模式
2. 设计一个ODL数据库模式

国家巡查罚单处理系统

查阅在第5章实例研究中的国家巡查罚单处理系统的描述，使用ERD和类图作为该系统的起点：

1. 设计一个3NF的关系数据库模式
2. 设计一个ODL数据库模式

计算机出版公司

短短10年之内，计算机出版公司（CPI）从一个小型的图书出版社发展为一个在传统图书、电子图书以及远程教育课件上占重要市场份额的大型国际化公司。CPI开发图书和课件的过程与其他出版商相似，但其他出版商的开发过程在快速的产品周期和产品形式的多样化上被证明是不方便的、缓慢的。

文字和艺术在以电子形式的开发上有很大的不同。它们之间格式的转换是很困难的且很容易出错。许多编辑步骤仍采用"纸和铅笔"来完成。关于一致性错误在书内和书与其他相关产品之间是很常见的。开发或修订一本书及其所有相关产品通常要花上至少一年的时间。

CPI的总裁决定开始一项战略规划来重新设计CPI开发图书和相关产品的过程。他们与Davis System（DS）结成了战略伙伴关系，一起来开发支持重组流程的软件。在化学和制药产品的软件开发上，DS有着丰富的经验，他们使用最新的开发工具和开发技术，当然也包括面向对象的软件和数据库。CPI希望新方法和软件可以缩短开发时间，降低开发成本。两家公司都希望在今后几年中可以把这个软件授权给其他的出版商。

一个联合分析组为这个软件规划出工作流和高层需求。分析组计划使用一个大型数据库来存放生产过程中各个阶段涉及的所有书和课件的内容。作者、编辑以及其他生产人员可以通过各种途径和数据库进行交互。这些途径包括传统的字处理程序和基于网络的接口。如果需要的话，可以正确地完全连续地进行各种形式的转换。所有内容的创建和修改都将是电子化的——除了供出售的印刷书以外，没有任何文本或美工会在纸上创建和编辑。

在生产过程的每个阶段，软件都要对内容进行记录和管理。对各种产品都一样的内容在数据库中只存储一次。产品内部以及产品之间的依赖性会被记录在数据库中。软件要确保对任何内容的增加或改动都会在与其相关的内容和产品上得到体现，而不管最终产品的形式如何。例如，第2章中的一句话引用了第1章中的一幅图，如果那幅图中的数据发生了变化，那句话也会被自动更新。如果书中添加了新图，那么这幅图也会被自动添加到相关课件的演示幻灯片中去。网站上相关的课件和学习材料会自动地反映章节后问题的答案的变动。

1. 将书本内容作为CPI数据库内容的模板，画出代表书本及其关键内容元素的ERD和类图。哪一个图能更准确地反映书本的内容？扩展你的图，使之包含相关的产品内容，例如：一套幻灯片，一本形如网站格式的电子书，一个基于网络的试题库。

2. 设计一系列可以用来存储书本、幻灯片以及网站内容的数据类型。图12-26所列出的关系DBMS的数据类型够用了吗？

3. ODBMS的哪些特点（优于或不同于RDBMS的特点）在实现CPI数据库时有用？给出如何使用它们的例子。

对落基山运动用品商店实例的再思考



在第5章的"对落基山运动用品商店实例的再思考"中曾让你考虑，RMO要实现自己的赊购账对建模所需的附加事物和关系。如果你当时没有完成的话，那么现在请完成这个练习，并同时更新相应的ERD和类图。接下来，完成下面的任务。

1. 根据你对RED所做的改变，更新图12-9中对RMO关系数据库的设计，并确保你数据库中的表都属于第3范式。

2. 为类图中所有新增类和关系写出ODL的模式规范说明。

3. 验证在你为问题1所更新的数据库设计中，哪些新类和关系都可以正确表示出来。

关注Reliable Pharmaceutical Services



使用你在第5章中设计的ERD和在第7章中开发的设计类图完成下面的任务。

1. 设计一个3NF的关系数据库模式。

2. 设计一个ODL数据库模式。

3. 从正反两方面讨论分布式数据库结构。一旦新系统完成，Reliable应该采用哪种（或结合哪几种）结构。

参考资料

Dirk Bartels and Greg Chase."A Comparison Between Relational and Object-oriented Database Systems for Object-oriented Application Development." http://www.versant.com/resources/control/white_papers/ODBMSvsRDBMS.pdf, 2001.

The Object Database Management Group Web site. http://www.odmg.org.

Robert Orfali, Dan Harkey, and Jeri Edwards. *The Essential Client/Server Survival Guide* (3rd ed.). John Wiley&Sons. 1999.

Peter Rob, and Carlos Coronel. *Database Systems:Design Implementation and Management*(7th ed.).Course Technology.2007.

第13章　用户界面的设计

学习目标

阅读本章后，你应具备如下能力：

- 理解用户界面和系统界面的差别
- 解释为什么用户界面是面向用户的系统
- 讨论以用户为中心的设计的三个原则的重要性
- 描述人机交互（HCI）技术领域的历史发展
- 描述有关人机交互的三个隐喻
- 讨论可视性和可供性如何影响可用性
- 在设计用户界面时应用对话设计的8条黄金规则
- 定义整体系统结构——菜单层次体系
- 将用户—计算机交互场景设计成对话
- 创建或非图示演示一段对话框出现的次序
- 利用UML类图，顺序图为对话设计建立文档
- 设计用于实现对话的窗口窗体和浏览器窗体
- 列出在Web设计时使用的主要原则

本章要点

- 输入和输出的识别与分类
- 理解用户界面
- 用户界面设计指导原则
- 对话设计的文档编制
- 设计Windows和浏览器窗体的指导原则
- 网站设计指导原则
- RMO对系统设计

Aviation Electronic的界面设计

Bob Crain近来一直很赞赏安装在Aviation Electronic（AE）的制造支持系统的用户界面。Bob是AE在中西部的制造厂经理，他负责生产销售在全球的飞机上使用的各类航空设备。这些航空设备为全体飞行机组人员提供指导和控制的功能，而且还为飞行员提供在飞行过程中所需要的最新安全保障功能。

制造支持系统涉及的制造过程包括产品生产、购买、零配件库存、质量控制、成品库存和销售等诸多方面。Bob在几年的时间里全程参与了该系统的开发工作，包括初期的规划和开发。该系统体现了他在制造方面所知道的几乎全部内容，开发该系统的信息服务小组人员完全依靠Bob的专家知识。Bob对于制造过程可谓轻车熟路。

最终的用户界面尤其令Bob满意。Bob坚持要求开发人员要考虑“走出盒子（outside the

box)”（突破思维模式）的设计思路，他不想要一个切蛋糕式的事务处理系统。他希望构造出的系统能够像制造过程中的一个合作伙伴，让人感觉到它真正地适用于用户所做的工作。毕竟工厂往往把可用性作为其所生产的设备的主要设计目标，制造支持系统不也应该像这么设计吗？

第一位项目经理根本就不讨论系统的可用性，他认为用户界面只是在控制功能开发将近结束的时候要添加的辅助部分。Bob一再要求撤换项目经理，于是信息服务部门委派了Sara Robinson来领导这个项目组。

Sara持有截然不同的见解，她刚刚接手工作就积极了解影响制造过程的事件以及用户需要系统提供哪些支持。虽然她有一组系统分析员从一开始就着手财会事务细节的设计开发工作，但她仍然一直强调从用户的观点出发来设计系统，侧重于如何方便用户的使用。Bob和Sara组织了有用户参加讨论系统设计方案的会议，甚至请用户到现场模拟与系统进行对话。这种方法就是“走出盒子”。

在另外一次会议上Sara使用了界面设计草图，并要求用户充实草图。用户将自己希望看到的信息和将要用到的选项添加到草图上。这些会议产生了许多新的想法。例如，许多用户并不是整天坐在办公桌前，他们需要更大的和更多的图形显示信息以便能在房间的其他地方看到屏幕内容。许多用户需要参考多种显示内容，并且希望能够同时看到这些信息。利用原形仿真来模拟操纵过程是完成这些功能最好的方法，经过用户的充实，界面草图真正反映了生产过程的实际流程。这些草图最后大多派上了用场，占了最终系统界面设计的一大部分。Sara和她的小组坚持每月都与用户见面，拿出更多的设计方案，征求更多的建议和意见。

当系统最终完成并安装时，由于许多用户一直参与了系统的设计工作，他们已经知道如何使用系统。Bob了解系统能做的全部工作，但他只使用他自己这一部分。他坐在办公桌前，只要单击屏幕上的“监视当前过程”按钮，制造支持系统就会给他整个上午工作情况的介绍。

概述

信息系统捕获输入和产生输出，并且输入和输出发生在介于信息系统与其环境之间的界面。系统界面处理那些需要少量人员干预的输入和输出，用户界面处理那些需要人员直接参与的输入和输出。本章内容将对上述两种界面进行比较，并将侧重于描述如何设计用户界面。接下来的第14章侧重于描述系统界面、系统输出和系统控制部分。

为系统设计用户界面是系统设计关键活动之一。当用户与计算机之间进行的交互执行一个任务时，设计用户界面就是设计输入和输出。本章内容侧重于用户和计算机的交互，即所谓的人机交互（Human-Computer Interaction，HCI）。对于每一项输入，开发者必须考虑用户和计算机的交互，并设计界面来处理输入。类似地，对于用户所需要的每一项输出（如联机报表），开发者也必须设计交互过程。因为人机交互更像是用户与计算机之间的一次对话，所以通常也把用户界面设计称为对话设计。

本章开头部分的内容主要讨论用户界面。我们将以用户为中心的设计、人机界面领域的发展和用于描述人机界面的一些隐喻作为背景。许多设计指导原则有助于确保系统的可用性，我们将讨论一些最重要的指导原则，包括基于Web开发的指导原则。接下来的内容是介绍如何编制对话设计文档，包括利用面向对象方法开发用UML对话。有关设计窗体和网页的指导原则也有所涉及。设计举例一直贯穿本章始末，包括RMO（落基山运动用品商店）的对话设计的例子。用户界面设计通常采用迭代的方法，即在每次迭代中设计几个用例，在

项目的初期，我们就应该对用户界面设计建立一个全局的概念，以便各个对话的设计能够相互协调。

13.1　输入和输出的识别与分类

系统的输入和输出是任何系统开发项目早期关注的内容。项目规划中列出了系统分析员在定义系统范围时所标识出的关键输入和输出。在系统分析阶段，系统分析员很早就讨论输入和输出，并经常与系统相关者一起确定系统的外部代理和参与者。这些外部代理和参与者一方面影响着系统利益，另一方面又依赖于系统所产生的信息。分析阶段产生的需求模型也强调输入和输出，例如：事件表中包含每一个外部事件的触发器，这些触发器就代表输入。输出被视为对于外部、状态和临时事件的响应。

13.1.1　传统和面向对象的输入和输出

在传统方法中，在关联图、数据流图（DFD）片断和详细数据流图中，输入和输出被表示成数据流。一项列出所有数据元素的数据流定义中详细描述每一项输入和输出。在设计阶段，当决定设计方案时，系统分析员根据所做出的决定会增加更多有关数据流的细节。例如，一项输入是被自动捕获还是由系统用户手工输入决定着系统设计的细节。第10章中曾经讨论过，这些细节必须与应用软件设计协调在一起。

在面向对象方法中，输入和输出被定义为进入和离开系统的消息。输入和输出分别对应于事件表中事件的触发器和响应。参与者为许多用例提供输入，并且许多用例为参与者提供输出。一个场景中交换的消息详细定义这些输入和输出，而且随着每一场景设计愈发细化，消息的内容也愈加确定。这些内容反映在交互图、设计类图的方法和状态图中。在第7章中介绍的系统顺序图给出了这些输入和输出。

13.1.2　用户界面与系统界面

无论是在传统设计方法还是面向对象设计方法中，系统设计的一个关键步骤是将每个事件的输入和输出划分到系统界面或用户界面中。系统界面中包含的输入和输出有很少的人工干预，它们可能由特定的输入设备（例如扫描仪）自动捕获的输入数据或者来自其他系统的电子消息，或是来自其他系统的已编译的批处理事务。许多输出被划分在系统界面中往往是因为它们主要向其他系统发送消息或信息，或者它们在无人工干预的情况下为外部代理或参与者生成报告、陈述或文档等内容。

系统界面：系统中包含少量人工干预的输入和输出部分。

用户界面包含了那些需要系统用户直接干预的输入和输出。一个用户界面能够使用户通过与计算机进行交互而记录一个事务。例如，一个客户服务代表为一位RMO顾客记录一份电话订单。有些时候在用户交互之后产生输出，例如在用户查询订单状态后相关信息会被显示出来。在基于Web的系统中，一个客户可以直接与系统进行交互从而得到信息，如下达订单或查询订单状态等。在RMO公司，Barbara Halifax的日常状态备忘录将会更新John MacMurty所做的客户支持系统用户界面设计中的某些活动。

用户界面：信息系统中需要用户交互的输入和输出部分。

在大多数系统开发项目中，系统分析员会将系统界面设计与用户界面设计分开，因为两种设计需要不同的专业知识和技术。但是对于任何系统组成部分的设计而言，都需要相当大量的协同工作。本章讨论用户界面的设计，下一章将讨论系统界面和系统控制部分的内容。

2007年5月12日

To: John MacMorty

From: Barbara Halifax

RE: 客户支持系统的用户界面设计

John，到目前为止我们一直都致力于用户界面原型开发，甚至在项目一开始时，我们就与不同的终端用户群体同工作。系统中基于Web的组成部分已经耗费了我们很多精力（自从可用性原型开发开始），我有一个开发小组致力于用户界面设计工作，另外一个小组致力于系统输出和软件的细化设计工作。将来我会投入更多精力到输出和软件方面的内容。

这里我只是想简短地汇报一下我们那部大部分交互对话设计工作的完成情况。我知道你已经看过有关关键对话设计的故事脚本并且已经做过由对象设计原型。我们已经进一步利用活动图和顺序图建立了对话文档，这些图将帮助我们解决界面中的技术问题，而且它们提供模板，能够帮助于确保对话之间具有一致性的视觉和感觉效果。

我们已经进行了设计方案的可用性测试并且召集了设计小组与用户的见面会。这些用户大多是电话订单客户代表和直邮报名参加Web软件设计的顾客。我们为这些用户准备了荣誉证书作为答谢。感谢你为系统原型所做的辅助工作。我会在下次工作会议之前进行检查。

BH

cc: Steven Deerfield, Ming Lee

13.2 理解用户界面

许多人认为用户界面是在开发过程临近结束时才开发和增加到系统上的。然而，对于交互式系统来说，用户界面远非如此。用户界面是当最终用户使用系统时所接触到的全部内容，无论从物理意义，感知意义，还是从概念意义上来讲，都是如此（见图13-1）。对系统的最终用户来讲，用户界面就代表了系统本身。



图13-1 用户界面的物理、感知和概念特征

许多系统开发人员，尤其是开发高度交互式的系统人员都同意这种观点，他们认为设计用户界面就是设计系统。为此，在设计过程中应该早地开始考虑用户界面。人机交互（HCI）这一术语一般用于指对最终用户及其与计算机交互的研究。

人机交互（HCI）：对最终用户及其与计算机交互的研究。

13.2.1　用户界面的物理特征

物理特征包括用户实际接触到的设备，即键盘、鼠标、触摸屏或数字键盘。但界面的其他物理部分，包括参考手册、打印文档、数据输入窗体等一些用户利用计算机完成任务时所涉及的内容。例如，洛基山运动用品商店的邮购订单输入职员在计算机终端前工作，利用打印形式的分类目录和手写的订单表格将各种订单输入系统。桌面空间、文档、照明以及计算机的终端硬件即组成了这位终端用户的物理界面。

13.2.2　用户界面的感知特征

用户界面的感知特征包括用户看到、听到、触摸到的所有东西（物理设备除外），用户所能见到的包括显示在屏幕上的所有数据和指令，如图形、线条、数字和文字。用户可能依赖于系统合成的声音，即使是用于表示击键和确认功能选择的蜂鸣声和咔嗒声。近来，计算机合成语音的出现使得计算机已经能够真正地与人交谈，并且利用语音识别软件，用户可以对计算机"说话"。用户可以使用鼠标"触摸"屏幕上的对象，诸如菜单、对话框和按钮，也可以在完成任务时用鼠标单击文档、图形或事务记录。

13.2.3　用户界面的概念特征

用户界面的概念特征包括用户了解的有关系统使用的所有内容，即用户正在操作的系统中所有问题域中的"事物"。系统所执行的操作以及随后的操作实施过程。要想使用系统，终端用户必须了解相关系统的所有细节，不仅包括系统内部如何实现，而且包括系统所做的工作以及如何完成任务。这些要了解的内容被称为系统的用户模型。大多用户模型是系统的逻辑模型，我们在本书的第5、6、7章已经学过。系统需求的逻辑模型要制订得非常详细，所以用户需要了解相当多的细节才能操作该系统。我们也可以回顾一下，系统分析员依赖最终用户的帮助来定义系统需求，而最终的系统需求由系统分析员从各种各样的模型中综合而来，用户对系统需求的了解已成为决定最终系统成功与否的根本性因素，而且如果用户对系统的了解包括部分界面内容，那么用户界面就不仅仅只是开发过程临近结束时才增加到系统中的一个部件。

实践指导

记住用户和用户界面也是系统的一部分。■

用户模型：用户对所使用系统了解的内容，包括用户正在模拟的问题域"事物"、系统所执行的操作以及随后的任务实施过程。

13.2.4　以用户为中心的设计技术

许多研究人员将他们的注意力集中到有关以用户为中心实施开发过程的创造性分析和设计技术上，因为他们已经认识到了用户界面对系统开发人员和系统用户的重要性。这些技术经常统称为以用户为中心的设计。以用户为中心的设计技术强调以下三个重要原则：

- 及早关注用户及其工作；
- 多次评价系统设计以确保其可用性；

- 使用迭代开发方法。

以用户为中心的设计：将用户放在开发过程中心地位的技术集合。

及早关注用户及其工作与本书中的系统分析方法是一致的：系统分析员必须理解和识别系统用户以及他们对系统的需求。传统开发方法更多地关注从业务角度出发的需求——系统要实现什么功能以及数据来源和目的是什么。可能是因为面向对象系统更加具有交互性，所以面向对象方法就更加关注用户和他们的工作，这通过分析参与者、用例及场景完成。正如第7章所讨论的那样，用户与计算机之间的自动化边界在系统需求模型建立的早期就要定义好。

以用户为中心的设计方法试图更深入地理解用户，但是他们都了解些什么？他们是怎样学习的？他们倾向于如何进行工作？是什么在激励他们？对用户及其工作的关注程度往往由所开发系统的类型的不同而不同。如果系统是一个预包装的直接面向最终用户的桌面应用系统，那么在该系统中需要高度关注用户及其偏好。

以用户为中心的设计方法的第二个原则是评价系统设计以确保其可用性。可用性是指学习和使用一个系统的容易程度。确保系统可用性并不容易——有太多不同类型的用户，他们又有不同的偏好和适应能力。对于某个人来说非常易学的设计特点有可能对于另一个人来说却非常难，如果一个系统要面对各种各样的最终用户，那么系统设计者怎样确保该系统所有用户用起来都能得心应手呢？如果系统的设计过于灵活，某些最终用户可能会感到失落；另一方面，如果系统设计过于死板，那么某些用户将会觉得倍受挫折。

可用性：学习和使用系统的容易程度。

在方便用户和易于学习方面，我们有更多的考虑，而这些概念往往是冲突的，过为易于学习的界面并不总是易于使用的。例如，拥有大量表单、对话框、广泛提示信息和指导信息的基于菜单的应用程序往往是易于学习的，实际上，这种系统是自解释型的。这种易于学习的界面适用于那些用户并不经常使用的系统。但如果是办公室职员整天都在使用的系统，设计的重点就应该放在界面的快速切换和灵活性方面，应该有包括快捷键、热键、大信息量屏幕显示等方面的设计。这里的第二种界面相对前者而言不易学会，但是一旦学会就显得非常好用，办公室职员（在管理人员的支持下）往往愿意花费较多的时间学会使用后一种系统以提高办事效率。

系统开发人员利用许多技术来评价界面设计以保证系统的可用性。以用户为中心的设计方法需要检测用户界面的各个方面。有些系统可用性测试技术收集进行统计分析的客观数据，用于与设计时的数据进行比较，有些技术用于收集那些关于用户感觉和态度方面的主观数据。为了评价用户态度，开发人员要进行正式调查、重点小组会议、设计演练、书面评价、专家评价、正式的实验室试验以及非正式的观察等。

以用户为中心的设计方法的第三个原则是使用迭代开发方法。即先做一些分析工作，接着做一些设计工作，再接着做一些实现工作，然后重复以上过程。每次重复之后，项目小组都要对当前系统进行评价。迭代开发方法始终以用户为焦点，在某一设计周期结束后重新审视用户需求并对新近完成的设计进行评价。这里所讨论的迭代开发方法既可以应用于传统方法，也适用于面向对象方法。

13.2.5 人—机界面研究领域

用户界面设计技术和HCI的研究领域起源于对人与机器的交互作用的一般研究，即人为因素工程或人体工程学。对人为因素的正式研究开始于第二次世界大战期间，当时宇航工程师研究战斗机驾驶舱控制器的不同排列方式对飞行员的影响。飞行员在飞行时要控制许多设备，并且飞行员与设备间交互的有效性是至关重要的。如果飞行员犯了一个错误（就是说他不能

正确使用某一种设备），飞机可能就会坠毁。飞行员出错即是宇航工程师所认为的"人为因素"，而人为因素导致的错误往往是无法避免的。

人为因素工程（人体工程学）：对人与机器的交互作用的一般研究。

这里有一个关于人为因素重要性的故事，它讲的是飞机驾驶舱的设计在经过微小变动后所造成的不同后果。设计者交换了节流阀和座椅弹射控制手柄的位置，结果导致了飞行员弹射次数急剧上升。在有压力的情况下，飞行员往往伸手去抓他以为是节流阀的东西，结果却把自己从飞机里弹射出去。最初，设计者把这一失误归于飞行员缺乏进一步的准确性操作训练。但是在加强训练之后，压力之下的飞行员仍然抓错操纵杆。显然，解决这种人为因素的关键应该是改变机器设计以适应驾驶员，而并不是要求驾驶员来适应机器设备。

因为是工程师在设计机器，所以人为因素领域首先与工程学发生了联系。但是习惯于精确规范和预计行为的工程师们常常因人为因素的存在而受挫。随着时间的推移，逐渐涌现出研究和总结人及其行为原则的专家，这些行为原则包括认知心理学、社会心理学、语言学、社会学、人类学以及其他众多学科，如图13-2所示。信息系统专家带着对人机交互研究的兴趣在计算机中引入所有这些原则。



图13-2　HCI研究领域

20世纪70年代，施乐公司在人机交互领域做出了重要贡献。施乐公司生产的高速影印机提供了大量的专门选项以及操作员可设定的支持能力。影印机的设计者认识到了使复杂机器便于操作员学习和使用的重要性。施乐公司的顾客希望操作员培训时间能够最小化，但操作错误可能会付出代价。例如，一个职员要影印大量文件，但却在设定细节要求时出错了，那么就造成浪费并且延误重要文件的发出。因此，施乐公司强调机器的可用性。

施乐公司建立了称为Xerox Palo Alto Research Center（Xerox PARC）的研发实验室，该实验室研究内容主要涉及影响人对机器操作的相关问题。作为这项投资的结果，施乐公司最后为影印机配备了触摸屏和菜单驱动的界面。界面上使用了图标对象，如文档、纸堆、订书钉和分类箱等。

Xerox PARC中心的研究和开发工作也包括计算机操作和面向对象程序设计。最早的纯面向对象程序设计语言叫Smalltalk，就是由Xerox PARC中心的Alan Kay设计出来的，并且它能简化交互用户界面的开发过程。20世纪70年代早期，Kay设想出一种被称为Dynabook的先进便携式个人计算平台（类似于现在的超轻笔记本计算机）。因为就当时的技术条件来说，Dynabook所需要的硬件还无法实现。所以许多研究人员认为这种计算机要生产出来至少需要30～40年的时间。Kay于是决定研究能在他所预期的未来硬件上运行的软件，这样就出现了Smalltalk。

Smalltalk包括了用于构成窗口界面关键部分的各种类——窗口、菜单、按钮、标签、文本域等。用于描述和构造这些界面的设计编程规则是与该语言的发展一起成熟起来的，而且是百分之百的面向对象。

由于Xerox PARC的工作，Xerox公司终于在20世纪70年代末开发出了个人计算机（Xerox-Star）并成功地推向了市场。Xerox-Star是第一批具有图形用户界面的通用个人计算机之一。虽然Xerox-Star在那个时代非常超前且过于昂贵，但是它的出现仍然被看做是计算机界具有里程碑意义的事件。Xerox-Star的关键特征是由位于施乐公司附近的一个名叫苹果的小公

可在20世纪80年代初开发出来的。Apple首先开发出了成为Xerox Star特征的Apple-lisa，其后是Apple Macintosh。Xerox PARC的研究工作对于面向对象程序设计、个人计算机以及用户界面设计产生了重大影响。

现在系统设计的面向对象方法变得更加具有影响力，像Xerox PARC这样的实验室所提出的先进设计理念和开发技术正日益集成到众多业务系统的开发方法中去。HCI领域发展起来了。有关这方面研究和实践的学术刊物、会议以及系列书籍已有很多。经过一些培训，大学生和研究生程度的开发人员都可能成为HCI领域的专家。

13.2.6 有关HCI的隐喻

市面上存在着许多有关HCI的看法，通常称为隐喻或类比。这些隐喻包括直接操纵隐喻、文档隐喻和对话隐喻三类。每一个隐喻都是对不同的概念给出一个类比，都是对用户界面的设计给出暗示。

1. 直接操纵隐喻

直接操纵假定用户与屏幕上的对象直接交互，而不是用原来的命令行输入方式。与用户交互的对象在屏幕上是可见的，用户可以用鼠标或方向键指向并操纵它们。最早能直接操纵的界面来自字处理程序，字处理程序允许用户在文档的相应位置直接键入文字。到了20世纪80年代初，IBM DOS PC计算机中的电子表格应用程序（最早是VisiCalc，接下来是Lotus 1-2-3）为用户提供了直接操纵的方法，用户在电子表格的单元格中直接输入数字、公式或文字。屏幕上的电子表格在概念上非常类似于从事会计和金融工作的人士所熟悉的纸质表格。由于这一相似性，人们很容易理解和使用电子表格，并且发现这些应用程序非常有利于提高他们的工作效率，因为该程序中包含有对电子表格进行自动计算的大量公式。这些早期能直接操纵的DOS应用程序是个人计算机取得成功的重要原因之一。即使没有图形用户界面，它们还是非常流行，因为它们使得用户与计算机的交互变得直接、自然和便利。

直接操纵：HCI的隐喻，告诉我们用户能直接与显示屏幕上的对象进行交互。

Xerox PARC开发的Smalltalk语言将直接操纵应用到屏幕上的所有对象，其中有些对象是界面对象，例如按钮、复选框、滚动条和滑动条控件；而其他诸如文档、进度计划、文件夹和业务记录这些问题域对象也都以对象方式显示在屏幕上供直接操纵。例如，其界面上可能有一个垃圾桶对象，如果用户想要删除一个文件，他只需用鼠标单击文档对象并将该文档拖到垃圾桶上即可。通过这样的直接操纵方式，用户向计算机发出删除这一文档的命令。

伴随着面向对象程序设计方法，直接操纵最终演变成桌面隐喻，即显示屏幕上排列着众多的通用桌面对象，例如记事本、日历、计算机和文件夹。现在许多桌面中还包含了电话、自动应答录音电话机、CD播放机甚至还有视频监视器。用户与任何这些对象进行交互就像是与它们所代表的现实世界对象进行交互一样（见图13-3）。现在最终用户期望包括业务信息系统在内的所有应用程序都能够像在桌面上的对象一样自然灵活。较大的显示器可以使多个桌面程序经合理的排列后同时显示给用户（见图13-3）。

桌面隐喻：一种直接操作方法，在那里显示屏幕上选择通用的桌面对象的排列。

2. 文档隐喻

关于界面的另一种看法是文档隐喻，其中的人机交互包括对电子文档的浏览和录入。这些文档非常类似于打印文档，但因为文档是电子形式的，更具有交互性。电子文档的组织不同于纸质文档，因为读者可以在电子版本的不同位置之间跳转。超文本文档允许用户单击某一链接然后跳转到文档的另一部分或者完全跳转到另一文档中。

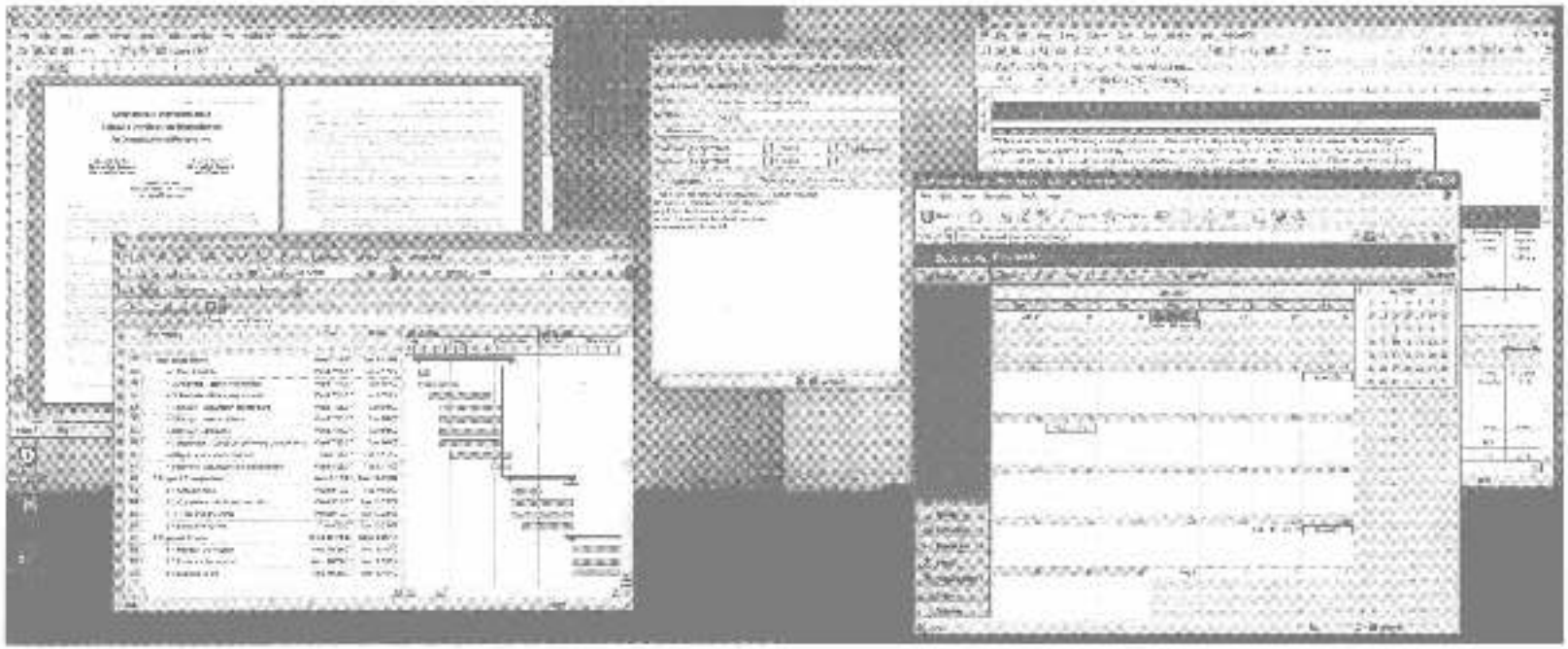


图13-3 在宽屏上显示的基于直接操纵的桌面隐喻

文档隐喻：HCI隐喻，实现了电子文档浏览和录入数据的人机交互。

超文本：超文本允许用户单击某一链接并且跳转到该文档的另一部分或者另一文档中。

大多数通用桌面应用程序可以创建并编辑电子文档，该电子文档并不局限于文本，其还包括了文字处理、电子表格、演示文稿和图形处理。所有这些应用程序都产生文档，其产生的任一类型的文档都能包含文字、数字和任何其他应用程序所生成的图形，从而构成所有相关介质类型的文档集合。超媒体将超文本概念进行了扩展，包括诸如图形、视频、音频等可以通过文档中用户导航操作而相互链接在一起的多媒体内容。

超媒体：对超文本进行扩展的技术，包括诸如图形、视频、音频等多媒体内容。

WWW是围绕着文档隐喻而组织的，网页上的所有内容被组织成相互链接在一起的众多页面，统称为超媒体（注：HTML即超文本标识语言）。其至通过选择页面文档中的信息就可以处理事务。作为描述和设计交互式系统的方法，文档隐喻和浏览器界面将继续对用户界面的设计产生影响。图13-4中的宽屏显示器有两个浏览器，每个浏览器包含一个超媒体文档。

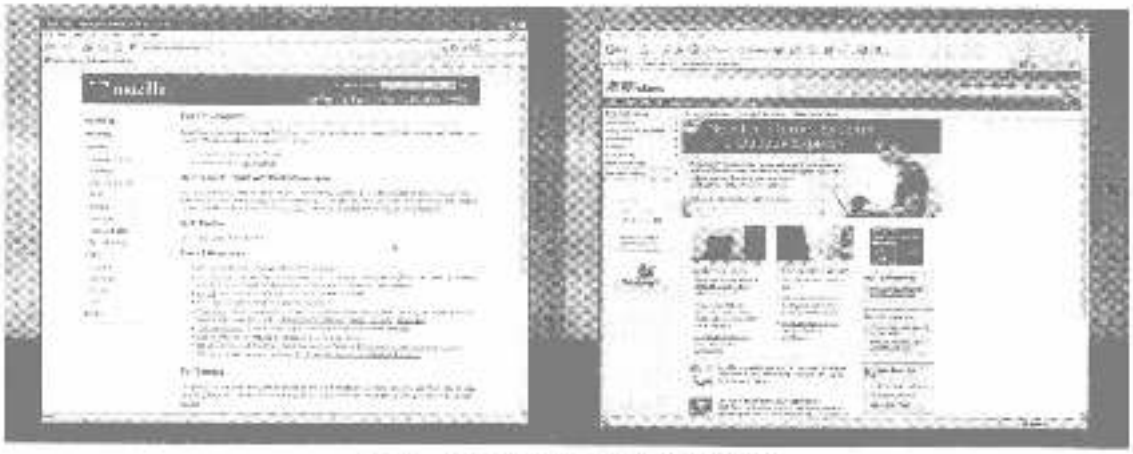


图13-4 网络浏览器中超媒体的文档隐喻

3. 对话隐喻

直接操纵和文档隐喻强调的是与用户交互的计算机对象。界面的另一种看法是对话隐喻，即用户与计算机的交互更像是在进行交谈或对话。用户界面设计常常被称做对话设计。与某个人进行一次交谈或对话，意味着参与对话的两个人中的每个人都要倾听对方意见并回答问题，你来我往，有序地交换信息。计算机“倾听”和“回答”用户的问题和意见，并且用户倾听和回答计算机的问题和意见，所以对话隐喻是对人机交互的另外一种理解。像直接操纵隐喻一样，对话隐喻以系统的面向对象观点为基础，因为通信是将一个对象的消息传递给另一个对象。图13-5表明用户和计算机之间是如何彼此传递消息而进行通信的。

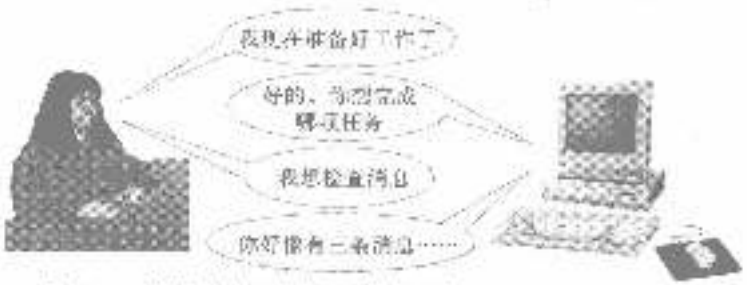


图13-5 对话隐喻表示用户和计算机之间通过传递信息进行交互

对话隐喻：HCI隐喻、人机交互更像是在进行交谈或对话。

考虑下面所描述的经理与其助手之间的对话：

经理：我出去的这段时间有留给我的消息吗？

助手：是的，你有三条消息，分别来自于Bob、Mary和Lim。

经理：Lim说了些什么？

助手：Lim在昨天晚上8:15留下一条消息，是有关下周一库存管理系统会议的，消息内容是："我们能把会议时间改在10:30吗？我可能因为误机会议而迟到。"

经理：我最好回复一下，告诉他修改时间不是问题。

助手：好的，我会告诉他的。你想了解下一条消息吗？

经理：Mary说了些什么？

助手：她在今天上午8:15留下有关午餐的消息，她说："午餐安排照旧，但Joe不能和我们共进午餐了"。

经理：好，不必回复了，就到这里吧，谢谢你。

助手：好的，你还有一条来自Bob的消息，我稍后会提醒你。

这段对话的内容是经理和助手之间关于消息的对话。问题是由经理提出的，问题的回答和接下来助手提出的问题看起来非常清晰自然。如果由自动电话应答服务来完成这个基本对话会有什么不同吗？（其中，自动电话应答服务响应声音命令且利用计算机合成语音来进行回答）有可能不同。如果利用计算机应用程序来模拟实现一位智能"秘书"会有什么不同吗？有可能不同，但基本对话的模式将是相同的，那就是：问题、回答、另一个问题、回答、并且可能要求对方澄清、对澄清的要求做出回答、最终的回答。

基本对话形式与典型的电子邮件应用程序是相同的，即使用户和计算机以不同的方式发送消息。用户通过选择某一菜单项来读一个新邮件。计算机列出所有新的邮件消息供用户选择。用户选择了一个消息后，计算机会将消息的内容显示出来。也许你会奇怪为什么说用户与电子邮件应用程序的交互与上面所演示的对话是相同的，但是实际上两者交换的基本信息和次序确实是一致的。

用户和计算机都发送信息，但双方由于各自所受限制而使用了不同的语言。用户不能理解隐藏在计算机内部的二进制代码，更不能直接进入计算机去逐一解释计算机用于表示二进制代码的电脉冲。对计算机而言的自然语言甚不能为人所用，计算机不得不适应用户，即以一种对用户来说很自然的方式提供消息，即用户能看懂和读懂的文字和图形。

同样，计算机不能理解复杂的语音信息、面部表情和形体语言等，这些对人类来讲很自然的信息形式。所以用户不得不适应计算机，通过单击鼠标、拖动对象、键盘输入文字等方式向计算机发送信息。计算机技术的进步使得用户通过更加自然的方式与计算机进行交互成为可能。但是目前典型的用户界面仍然要依赖鼠标和键盘。原因之一是，需要保持安静的办公环境以及保密要求。因此语音命令是否会在计算机应用环境中流行起来，其前景还不明朗。

用户界面设计的挑战是架构一个自然对话序列，它允许用户和计算机之间彼此交换用于完成某项任务的信息。于是设计者既要设计用户向计算机发送消息（用户语言）的语言细节，又要设计计算机向用户发送消息（计算机语言）的语言细节。

图13-6所列的是把前面提到的经理与助手之间的对话翻译成用户和计算机交互时所使用的语言。界面设计者使用一系列的信息图表和书面文字叙述来模拟人机交互。这是一种设计细节模型化的方法，其他的技术放在本章后续部分进行讨论。

	消息	用户语言	计算机语言
经理	我出去的这段时间有留给我的消息吗	单击在主菜单上的“读新消息”菜单项	
助手	是的，你有三个消息。分别来自于Bob、Mary和Lim		查找新消息并在列表框中显示消息标题
经理	Lim说了些什么	从列表框中取出Lim	查找所选消息的内容并在一个详细表中显示
助手	Lim在昨天晚上8:15留下一条消息，是有关下周库存管理系统会议的。内容是：“我们能把会议时间改在10:30吗？我可能因为测试会议而耽搁。”		
经理	我们最好回复一下，告诉她修改时间不是问题	单击在详细表单中的“回复”按钮 在消息中输入“好，没问题”，单击“发送”按钮	显示新消息表单并附上发送者的地址
助手	好的，我会告诉他的，你想了解下一条消息吗		显示“消息已发送”并且回到最初查找的新消息列表
经理	Mary说了些什么	从消息列表中双击Mary	
助手	她在今天上午8:15留下有关午餐的消息。她说：“午餐安排照旧，但Joe不能和我们共进午餐了。”		查找所选消息的内容并在一个详细表中显示
经理	好的，不必回复了。就到这里吧，谢谢你	单击“关闭消息”按钮 单击“关闭新消息”按钮	重新显示最初查找到的一个新消息表单
助手	好的，你还有一条又自Bob的消息，我稍后会提醒你		显示正在关闭“读新邮件”，并显示未读消息

图13-6 用于实现基于经理与助手之间对话的电子邮件应用程序的用户语言和计算机语言

13.3 界面设计指导原则

目前存在着许多已公布的系统开发人员界面设计原则。用户界面设计原则涉及面很广，包括从一般原理到特定规则的许多内容。本节将描述一些非常著名的有关用户界面设计的指导原则，本章的后续部分将介绍有关设计窗体和浏览器格式输入形式的设计原则和规则。一些系统开发组织采用界面设计标准，即该组织所开发的任何系统都必须遵守的一般原理和规则。设计标准有助于确保所有的用户界面都可用并且该机构所开发的所有系统都拥有相同的外观。

界面设计标准：某机构所开发的所有系统的界面设计都必须遵循的一般原理和规则。

13.3.1 可视性和可供性

Donald Norman提供了两项确保人机交互友好性的关键原则，即可视性和可供性。这两项

原则既适用于人机交互，又适用于其他任何设备。

可视性指的是控件对用户来讲是可用的，并且控件在用户动作之后应提供及时的反馈以示响应。例如，方向盘对于司机来讲是可见的。司机向左打方向盘时，方向盘的向左转动体现了对司机动作的响应。同样，按钮对用户是可见的，用户可以单击按钮。当按钮被用户单击的时候，它的形状发生变化表示按钮被用户按下去了。这就是信息反馈。有时候按钮以发出声音的方式进行反馈。

可视性：HCI的关键原则，规定所有控件必须是可见的，并且提供反馈信息指示控件对用户动作的响应。

可供性指的是所有控件的外观都要体现其功能，即控件的使用方式。例如，形状像方向盘的控件表明该控件是能够转动的。在计算机中，控件提供单击功能，滚动条提供滚动功能，列表中的列表项提供选择功能。Norman的原理适用于页面的任意对象，诸如先前图13-3和图13-4中给出的例子。

可供性：HCI的关键原则，规定所有控件的外观都应该体现和反映控件所实现的功能。

如果用户界面设计者保证所有控件都是可见的，并且都能体现其功能，那么界面就是好用的。许多用户都非常熟悉Windows界面和通用的Windows控件。然而，设计者在设计网页时要小心应用这些规则。许多网页中出现了新的控件类型，这些控件并不总是可见的，并且它们的效果也不如在标准Windows界面中显著。大多数对象是可单击的，但它并不直观表明它是否可单击，何时确认单击动作以及单击后会触发什么功能。比如，有时你单击一张图片会在浏览器中开启另一个新的页面，有时却什么也不会发生。

13.3.2　八条黄金规则

Ben Shneiderman总结出适用于大多数交互式系统的八条基本设计原则（有关Shneiderman的叙述参见本章末尾的题为“参考资料”部分的内容）。尽管这些设计原则是一般性指导原则而非特定规则，但Shneiderman仍将它们命名为“黄金规则”，希望借此说明这些原则对于提高系统可用性的关键作用（如图13-7所示）。



图13-7　设计交互式界面的八条黄金原则

1. 尽量保持一致性

设计外观和功能一致性界面是最为重要的设计目标之一。信息在窗体上的组织方式，菜单项的名称及其排列方式，图标的大小和形状以及任务的执行次序都应该是贯穿系统始末的。人有习惯性，一旦我们学会一种做事方式就很难改变。在操作计算机应用程序时，许多要采取的动作都以自动方式执行，我们不必考虑正在做什么。能够盲打的人不必考虑每一次按键，手指习惯性地自动击键。试想一下，如果键盘上两到三行的按键排列位置颠倒，对于习惯盲打的人会有什么后果，他们肯定用不好新键盘（当然也不会喜欢新键盘）。如果一个新的应用程序提供与众不同的操作方式，肯定会降低它的生产效率并且用户也不会乐意接受。

Apple Macintosh在20世纪80年代首先强调一致性的好处。苹果公司在Macintosh中提供了许多应用程序并制订标准用来规范开发人员设计其他应用程序。苹果公司声称，如果新的应用程序与原有应用程序保持一致性，那么新的应用程序将是很容易学会的。苹果公司还公布了相应的设计标准文档以便保证应用程序与Macintosh界面具有一致性。在这之后又出现了微软公司的Windows界面的类似例子和标准文档。

商务信息系统与最初基于Macintosh的桌面应用系统存在很大差异，有时应用程序恰恰需

要与最初原则的不一致性。例如，早期标准规定每项应用程序的菜单条都要包括文件、编辑和格式等菜单项。所有的面向文档的应用程序，例如字处理、电子表格和图形处理软件都需要这些菜单项。但是，许多商务系统不具有文件、编辑和格式等相关处理功能，但是其他的大多数指导原则和标准仍然适用。

相关研究也表明非一致性界面有些时候也是有利的，如果用户在分立的窗口中与多个应用程序进行交互，不同的可视化外观可能会有助于用户分辨不同的窗口。另外，在用户同时学习多个应用系统时，界面上的差别也有助于用户记忆不同应用程序的不同界面环境。基于以上原因所体现的不连贯性只是微妙而表面化的，应用程序的基本操作还是应该相同的。

2. 为老用户提供快捷键

整天使用某个应用系统的用户愿意花些时间学会使用快捷键操作方式。当老用户明确知道自己要做什么的时候，他们很快就对冗长的菜单选择次序和大量的对话框操作失去耐心。因此，快捷键的使用可以针对某一给定任务减少交互步骤，而且设计者应该为用户提供实用功能（例如宏），允许用户创建自定义快捷键。

有时整个界面是为熟练用户而设计的，而他们往往不对系统灵活性做太多要求。考虑一下落基山运动用品商店负责邮购订单输入的人员，他们每天的工作都是把顾客寄来的订单录入计算机中。这些用户需要简单、快捷、准确的界面形式。冗长的对话、大量菜单和大量窗体的系统会降低用户的工作效率。

3. 提供有效反馈

对用户所做的每一个动作，计算机都要提供某些类型的反馈信息。使用户知道相应动作是否已被确认。既然键盘单击对用户是有帮助的，所以单击也必须包含在操作系统中。如果用户单击一个按钮，按钮就应该改变外观并且可能会发出声音。

给用户的信息反馈也很重要。如果落基山运动用品商店邮购订单输入人员在订单屏幕上输入某一个顾客的ID号，系统应该进行相应查询以确保该顾客ID号的有效性，并且系统也应显示该顾客的名字和地址，使业务员确信该顾客ID号是正确的。同样，当业务员输入产品ID号时，系统应该显示该产品的相应描述。当业务员的注意力在邮购订单和计算机屏幕之间不断转移切换时，他要把系统提供的产品名及其描述信息与订单信息进行比较并保证使它们完全一致。这种确认方式与相应的自信感觉对于用户来说是非常关键的，特别是在他们整天与这一系统打交道的情况下。但是系统不能显示太多的对话框并要求用户做出回应，这样会降低用户的工作效率。

有时，反馈从另一个方面来帮助用户。RMO的电话订购办事员同样也需要邮购订单办事人员从系统中获取的信息，但他还需要额外的信息。电话订购的客户可能会问一些问题，给予电话订购的办事员的反馈信息会更细致、更灵活。我们将在本章的后面讨论为电话订购办事员所做的一些设计。

4. 设计完整的对话过程

系统的每一次对话都应该有明确的次序：开始、中间处理过程、结束。任意定义完好的任务都有开始、中间处理和结束三部分。因此计算机上的用户任务也有相同的操作感觉。正如前面所述的经理与助手之间的对话一样，如果用户正想着“我要查一查消息”，那么对话过程就以此询问开始。接下来是信息交换，然后结束。如果任务的开始和结束不明确的话，那么用户用的时候可能会很迷惑。另外，用户常常会一心一意地专注于某一任务，所以如果确认该任务完成，那么用户就会理清思路并转向下一项任务。

如果系统需求最初被定义为系统要响应的事件，那么每一事件触发某一特定处理过程，即预先定义好的活动，在传统的结构化方法中，每一活动是由数据流和结构化英语定义的。对于面向对象方法，每一项活动（用例）可能进一步定义为多个场景，每个场景都与一个事件流相联系。每一场景是一项预先定义好的交互过程，因此不管是在传统方法中，还是在面向对象方法中，事件分解使实现完整对话过程成为可能。

5. 提供简单的错误处理机制

用户出错是有代价的，既要花费时间改错，又会产生错误结果。如果零基正运动用品商店给顾客配送了错误的产品，那样代价相当大。因此，系统设计者必须尽可能防止用户出错。主要方法是限制可用选项和允许用户在对话框的任意位置都能选择有效选项。前面已讨论过的态足的反馈信息也有助于减少错误。

如果出错，就需要系统提供相应机制来进行处理。尽管第14章讨论的有效性技术可以用于找出错误，但系统还必须帮用户改正错误。一旦系统发现错误，错误消息应该特别说明出了什么错误并且解释要如何改正。错误消息不应该是指责性的，责备用户或者使用户感觉不舒服的做法不合适。

系统还应该简化错误处理。如果用户输入一项无效的顾客ID号，系统要提示用户并把先前输入的该顾客ID号显示在文本框中以供编辑。这样，用户能看到错误并编辑修改而不必完全重新输入。下面是一段用户输入顾客ID号后显示的出错信息：

```
The customer information entered is not valid.Try again.输入的客户信息无效，请重试。
```

这条消息并不解释在哪里出了错或者下一步要做什么。消息显示之后，系统接下来如果清除输入窗体内容并重新显示的话，结果会怎样？如果这样，用户则需要全部重新输入先前输入的相关内容。而且对于做错了什么还是一头雾水。这是因为对错误不做解释，而且要求重新输入，用户则无法判断出了什么错误。可以表达得更好一些的错误消息是：

```
The date of birth entered is not valid. Check to be sure only numeric
characters in appropriate ranges are entered in the date of birth field...
```

输入的出生日期无效。请检查并在日期输入栏中输入有效的日期。以附输入格式会重新显示，其他输入不变，只是输入无效数据字段中等待用户编辑改正。

6. 允许撤销动作

应该让用户感觉他们可以检查选项并且可以轻而易力地取消或撤销相应的动作。试验是用户学习使用系统的一种方法。这也是防止出错的一种方法：如果用户发现自己出错就可以取消该项动作。在棋子游戏中，只有参加游戏者的手指离开棋盘，相应的移动才告结束，用户在屏幕上用鼠标拖放对象也是同理。而且，设计者应该在所有对话框中都包含取消按钮，允许用户在任一步骤上都可以回退。最后，如果用户删除某些如文件、记录或事务等内容，系统会要求用户确认该项操作。

7. 提供控制的内部轨迹

有经验的用户希望有控制系统的感觉，系统响应用户命令，而不该让用户被迫做某事或者感觉到正在被系统所控制。系统应该让用户觉得是由用户在做决定。设计者可以通过提示字符和提示消息的方式使用户产生这种感觉。设计像前面提到的经理与助手对话那样的对话，这种对话会让用户觉得自己握有控制权。

8. 减轻短期记忆负担

人有很多限制，短期记忆是其中最大限制之一。本书前面部分已经讨论过，人在同一时间只能记忆7条信息。界面设计者不能假定用户能够记住在人机交互过程中一个接一个窗体或

看一个对话框提着为一个对话框的所有内容。

如果用户不得不停下来问："文件名是什么？顾客ID号是什么？产品描述信息是什么？"等等内容，那么就是系统设计给用户制造了太多的记忆负担。

牢记以上八条黄金规则，界面开发人员就能够保证用户界面的高效和有用。接下来我们将讨论关于编制对话设计文档的技术。

13.4 对话设计文档编制

现在，有许多技术可以用于帮助设计人员整体考虑对话设计及其文档。前面章节中已经讨论过，对话的设计基础是需要用户交互的输入和输出。这些内容主要用于设计菜单层次，其目的是对用户的对话过程进行导航。故事脚本、原型设计和UML图表等方法都可用于完成设计工作。

13.4.1 用例、子系统和菜单层次

输入和输出来自于传统方法中的数据流图或者面向对象方法的用例和场景中的数据。一般地，需要交互方式获取的输入项都需要进行对话设计。另外，用户所要的所有输出项也都需要对话设计。早期生分析过程中记录的事件是每一对话的基础，这则分为用户界面的非系统界面。

对话设计必须与其他设计活动同时进行。第10章的内容已经表明，子系统结构图（事务分析）包括系统交互部分的菜单结构的细节内容。另外，每一事件（DFD片断的转换分析）的结构图也包括与用户对话的细节内容。在面向对象方法的设计过程中，也在较早阶段关注对话设计，甚至在分析任务阶段。顺序图和协作图表中包含了对话涉及的细节内容。总体，菜单设计和对话设计不是孤立完成的。

从用户立场出发的完备系统结构可以用菜单来反映。每一菜单包含一个选项层次，其中的选项往往根据子系统或对象行为来排列。洛基山运动用品商店的客户支持系统包括订单录入子系统、订单完成子系统、客户维护子系统、目录维护子系统以及设计阶段增加的报表子系统。菜单的设计也是按对象排列的——顾客、订单、存货、装运等。每组菜单中很可能存在重复相关功能的菜单项。例如要查找完成订单，既可以通过客户菜单也可以通过存货菜单实现。

有时需要根据不同的用户类型提供多种菜单版本。例如，RMO的邮购订单处理人员并不需要过多选项，他们只是处理新订单。电话订购处理人员可能需要更多功能选项，但他们也不需要全部的系统功能。而且，有些功能选项只有经理才用到，例如管理报表和价格调整等功能。

菜单还应该包括不活动选项或使用在事件列表中的用例的选项，许多重要选项与第14章中所讨论的控制相关。其中包括某些场合使用的数据库备份与恢复功能以及用户账户维护等。另外，用户偏好功能允许用户自己定制界面。最后，菜单还应该包括帮助功能。

RMO客户支持系统中的所有引发对话的事件都可以以子系统方式分组列举出来。如图13-8所示。这些分组形成菜单层次的集合。另外，还有实用工具、用户偏好及帮助菜单层次。图13-8中的列表是众多可能菜单层次设计中的一种——这里只是刚刚开始。

依据以上菜单选项可以设计出相应的对话过程。在实现对话设计的所有选项后，设计人员能够定义面向不同用户的多种菜单结构。在设计过程中，菜单层次体系可以很容易地重新排列。

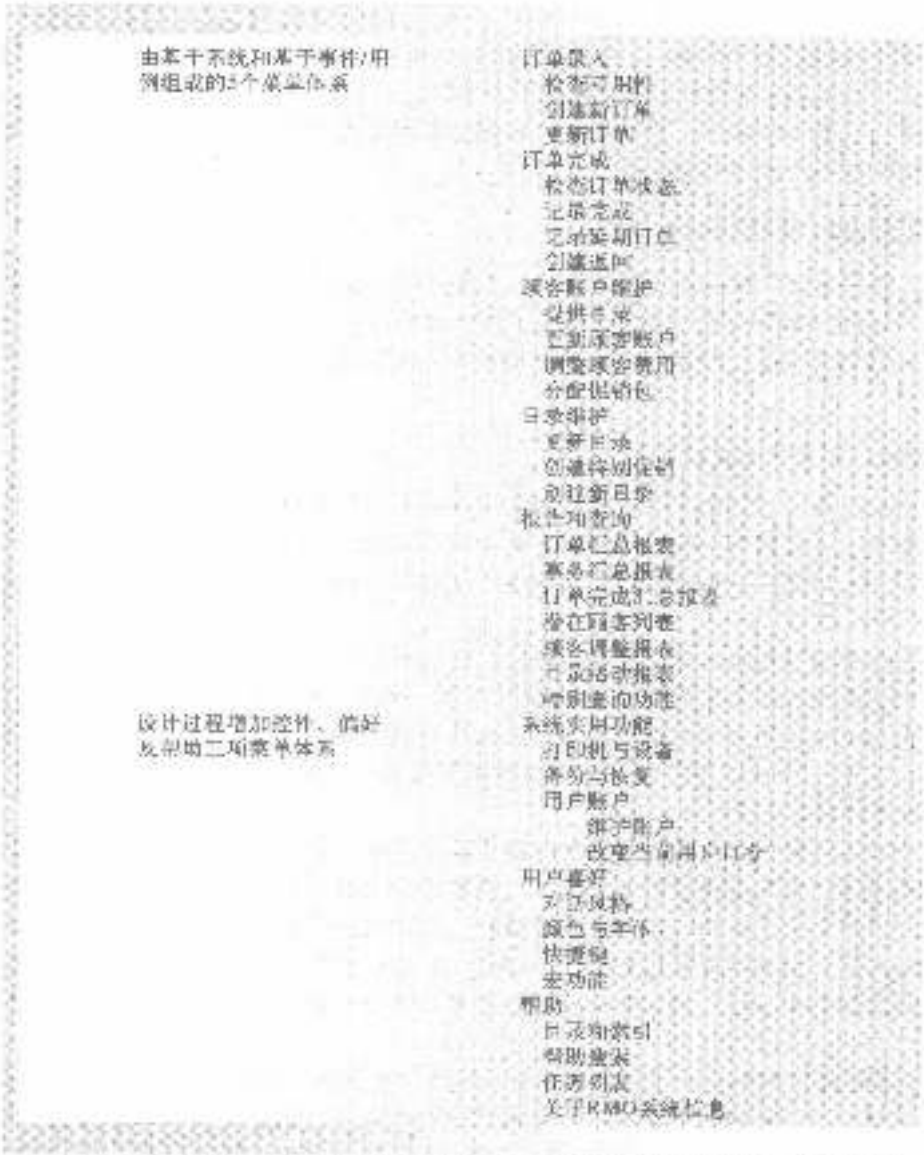


图13-8　RMO客户支持系统全部菜单层次（不是所有的功能对所有的用户都可用）

13.4.2　对话与故事脚本

一旦明确了所有用户需求的对话，就可以采用多种手段编制对话设计的文档。目前还不存在事实标准。一种方法是列举对话中的关键步骤并用书面一步骤中用户和计算机交互内容的文字描述。书写交互步骤的形式可以依据第4章中描述的活动图和第7章中的用例使用图格式，也可以更加自由。

设计人员也可以用本章前面描述的经理与助手对话的组织形式编制对话文档，记录用户和计算机如何表达自己，细节参见前面介绍的表13-1。有时设计者直接从对话框中抓取一系列的屏幕画面作为更具可视性的文档。想一想第7章结尾部分描述过的城市影碟出租系统的相关内容。顾客租借影碟是一个事件，它的对话可以用在职员和系统之间的下述对话描述：

系统：您打算选择城市影碟出租系统的什么选项？

用户：我想租借一些影碟。

系统：好的，那么您在出租服务中的顾客ID号是什么？

用户：201564。

系统：对的，您的顾客名是Nancy Well，住址是Silicon Avenue大道1122号，有两个小孩，名叫Bob和Sally，只能租借PG-13级影碟，对吗？

用户：对。

系统：您想借的第一张影碟的ID号是什么？

用户：5312。

系统：好的，这是1953年Barbara Stanwyck版的Titanic，租金是每3天1.00元，PG级，下一张影碟的ID号是什么？或者就借这些？

用户：9645。

系统：好的，这是2007年拍的Titanic，租金是每3天3.00元，PG-13级，下一张影碟的ID号是什么？或者就借这些？

用户：就借这些。

系统：好的，一共4元外加0.24元的税金，共计4.24元。付款完毕后请确认。

用户：确认。

系统：祝配Nancy Well，这些影碟必须在9日周三晚11时关门之前归还。

用户：好的，谢谢！

设计人员要通盘考虑用户与计算机对话过程所需要的屏幕画面次序。用户需要输入顾客ID号和若干影碟ID号。计算机必须查找并显示相关顾客信息和影碟信息。接下来计算全部费用并显示提示信息，以便业务员转告顾客影碟归还日期。

有一种屏幕显示技术叫**故事脚本技术**，即在对话中显示一系列的屏幕概略图。这些概略图不必包含体现基本设计概念的诸多细节。故事脚本可以用VB等可视化编程工具来实现，而用图形工具包绘制出的简单略图能够帮助设计者始终集中关注于基本的设计思路。

故事脚本技术：一种辅助对话设计文档的技术，显示一系列的屏幕概略图。

实践指导

在项目的早期，使用故事脚本为每个用例定义用户界面需求。向用户展示这些故事脚本，并要求他们提供反馈。■

图13-9显示的是城市影碟出租系统对话设计的故事脚本。系统拥有基于事件列表的菜单层次以及所需的控件。用户偏好和帮助等功能选项。系统使用一个窗体和几个对话框外加大量窗体显示信息来实现对话过程。注意，计算机所提问题显示在窗体底部的提示信息区域中，使用了几乎与手写对话内容中一样的语句格式。用户可以选择扫描或键入方式输入影碟ID号。提供给用户的信息显示在窗体中的标签上。系统提供的信息允许用户确认顾客身份，查看可能的限制，并且向顾客转达费用及归还截止日期等相关信息。换句话说，这个系统有助于用户与顾客进行信息确认，提供反馈信息，提供终止信息交互工作。

这些对话设计方法只是提供了一种工作框架，所得到的设计方案具有相当大程度的一般性。当工作原型构造出来后，许多细节内容仍然需要花费精力加以充实。在设计过程中回顾一下黄金规则以及其他指导原则将有助于提高系统的可用性。

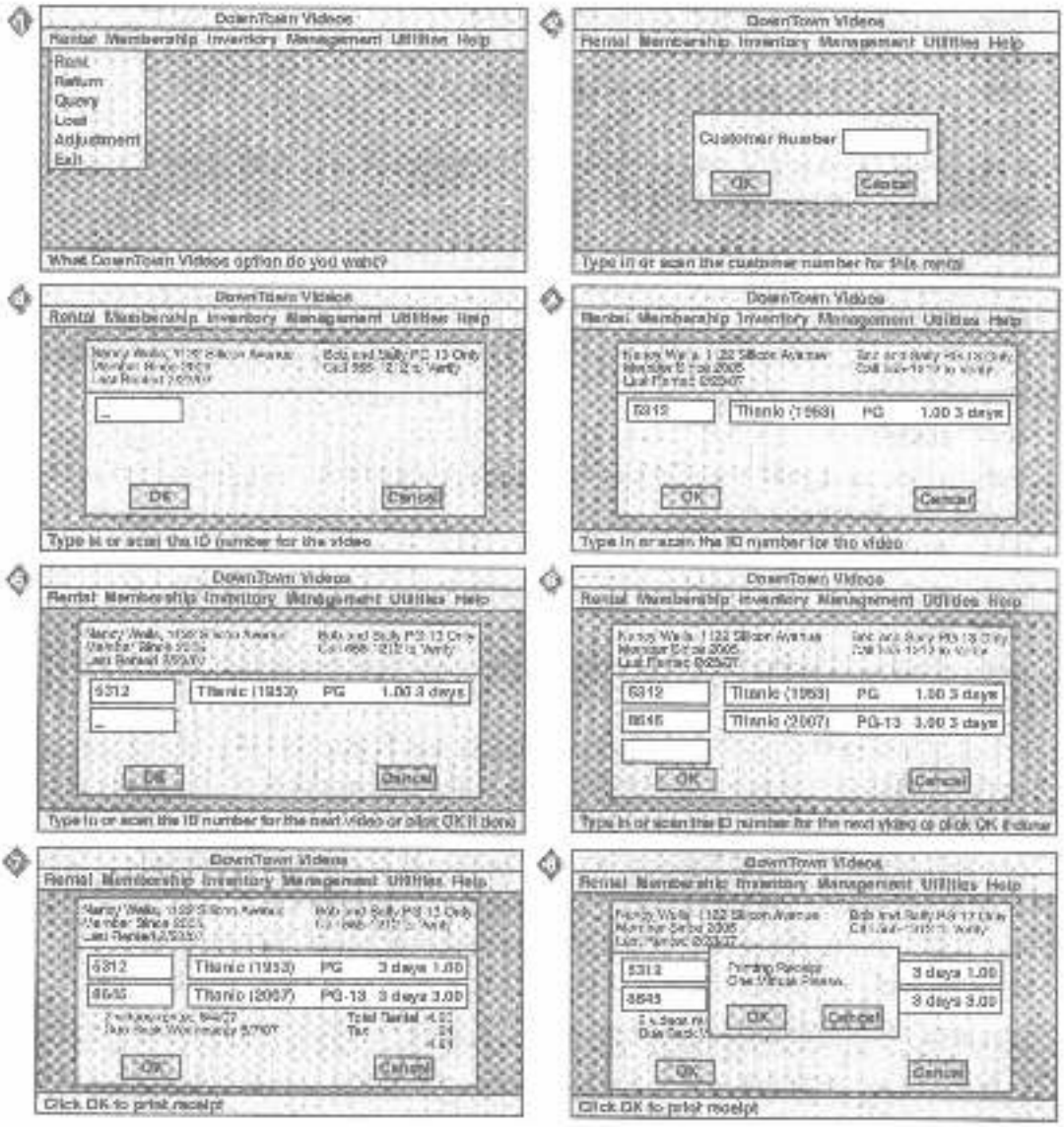


图13-9　城市影碟出租系统对话设计的故事脚本

13.4.3　用UML图表实现对话文档编制

面向对象方法提供特定的UML图表用于人机对话的建模。用例描述（在第7章中出现过）包括人与系统在交互过程中所遵循的步骤。活动图（在第4章和第7章中出现过）把人机之间的用例存为对话文档。两种图都能为对话提供人机交互所需模型。在面向对象方法中，对象之间来回传递消息，彼此按顺序“倾听”对方和发送反馈信息。人也可以发消息给对象并获得对象的反馈信息。第7章描述的系统顺序图就包括用户向系统传递消息和对象以消息形式反馈信息等。它主要显示了人与系统间的对话。由于系统顺序图是基于用例图中的顺序的，所以用例的对话设计要尽早开始，并且要经常改进。

在项目从分析阶段转向设计阶段的过程中，面向对象的方法把更多类型的对象添加到类

型和交互图中，具体实现参考第11章。这些对象类被打包成三层，包括用户界面类、问题域类和数据访问类。设计者可以将界面类和对象添加到这些图中，显示更多的关于人机对话设计的细节内容。这个设计过程在第11章中有图示。首先要决定用什么样的窗体来实现先前由非正式对话设计技术所描述的对话过程。接下来是对人机交互的场景顺序图进行扩展，显示用户（参与者）与窗体之间的交互。然后，用于构成窗体的用户界面类可以用类图来建模。最后，顺序图被进一步扩展以显示用户与构成窗体的特定对象之间的交互。

回顾一下RMO公司“查询可用条目”对话的用例，第11章中给出了这个用例的顺序图（图11-12），并且被扩展为一系列顺序图来说明该用例的实现过程。图13-10给出了该顺序图的一个版本，其中包括名为产品查询窗的窗体。这个窗体代表了三层设计中的用户界面层。该窗体位于参与者与用例控制者之间。参与者通过窗体与界面对象进行交互，参与者通过窗体上的界面对象向问题域对象传递信息，并获得问题域对象的响应。交互界面的需求模型展现了参与者与系统之间所需的消息。设计模型中增加界面是为了创造实现交互的物理模型。

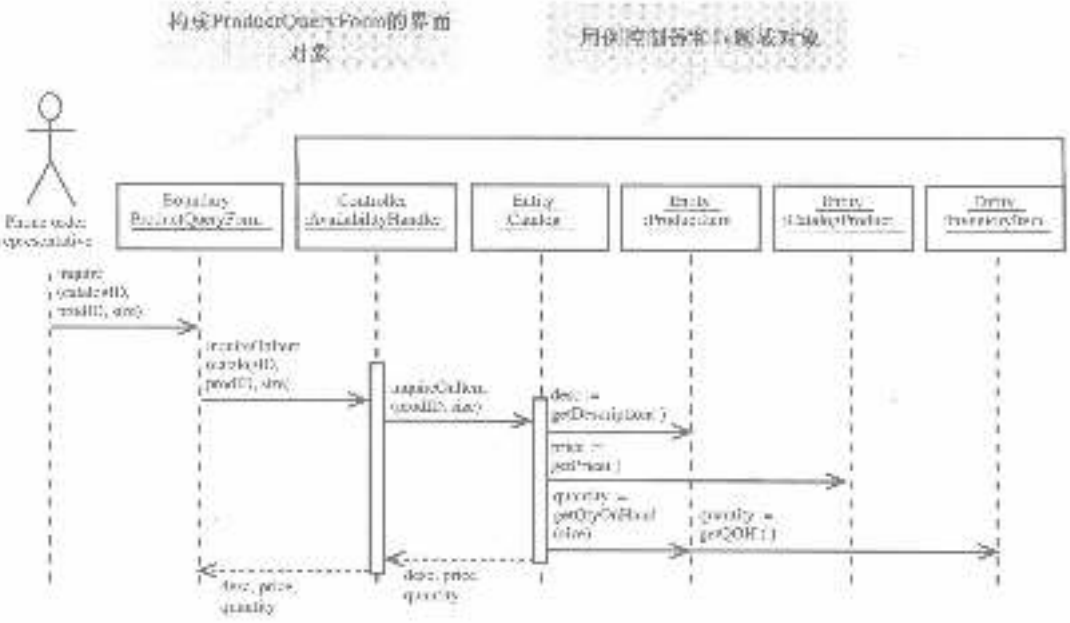


图13-10 增加ProductQueryForm的RMO“查询可用条目”对话的顺序图

在顺序图中呈现的窗体包含了特定的界面对象。图13-11所示为用类图显示了用于构造窗体的界面类。图13-11中的框架类代表了包含其他界面对象的基本结构。一个菜单栏与一个框架相连。一个菜单栏包含多个菜单，每个菜单中又包含多个菜单项。这些类之间的关系是聚集型关系，在类图中用菱形符号表示（有关聚集型关系的描述见第5章）。类图中还有其他类，包括列表类、按钮类和标签类，这些类都是框架的组成部分。本例主要用Java语言实现。框架能够“侦听”界面对象所发生的事件，例如单击菜单项或按钮。当框架“听到”事件发生时，框架的ActionListener（）方法就会被触发执行。

顺序图可用于用户和组成窗体的特定对象之间的消息，以及界面对象彼此间消息的建模。图13-12显示了经过进一步扩展的顺序图。这个模型强调窗体设计的细节，所以问题域细节可以忽略不计。顺序图中问题域对象间交互部分不做变动。界面对象仅仅是简单插到问题域对象和参与者之间。

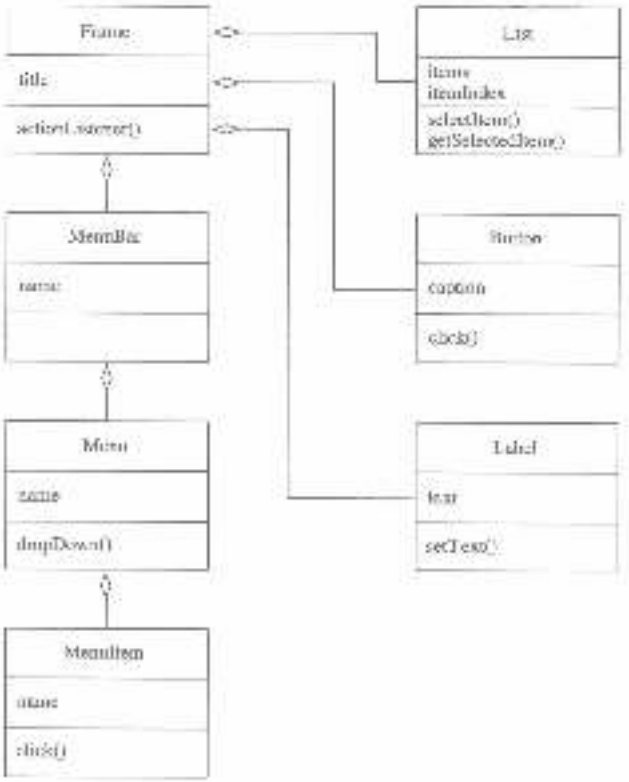


图13-11　构成类目搜索窗体的界面类的类图

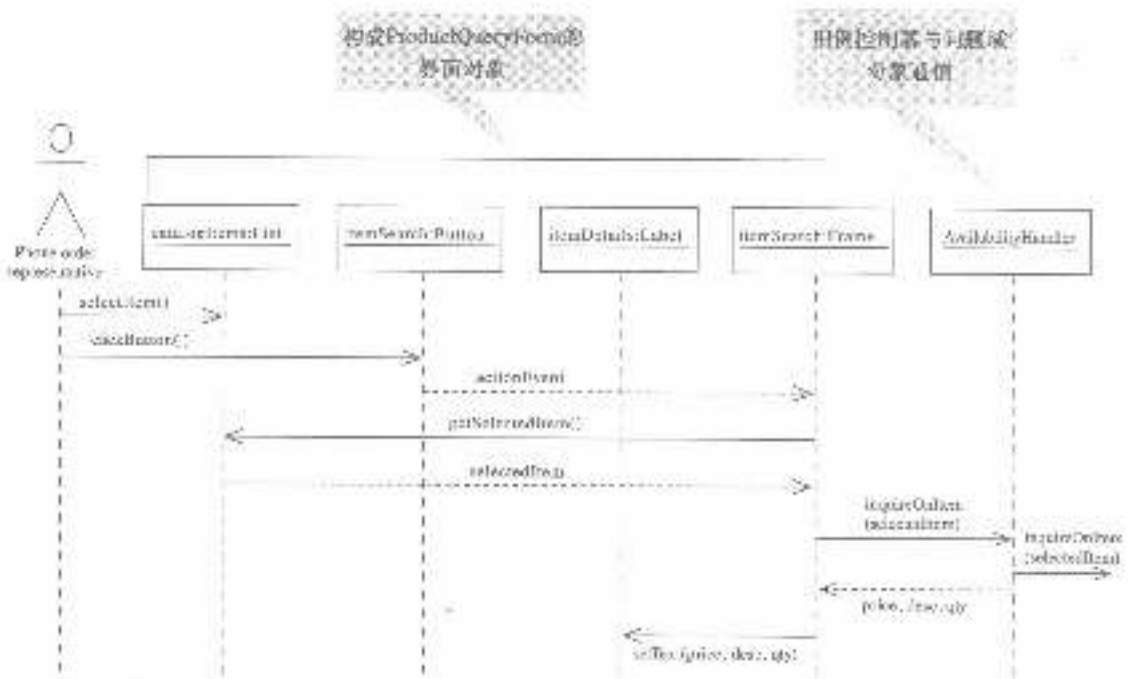


图13-12　构成“查询订用类目”对话ProductQueryForm的特定界面对象的顺序图（并非所有问题域对象）

与“查询可用条目”窗体交互的用户在列表中选择某一要查询的条目并单击搜索按钮。框架“听到”单击，问列表询问所选中的条目，并使用所选条目通过消息机制查询目录对象。该目录对象与其他的问题域对象进行交互。当目录对象返回所需信息时，框架告诉标签显示相关信息给用户。设计者不可能将每个窗体的设计细化到如此程度，本例只是举例说明在用户交互层是如何进行交互的。

13.5 设计标准窗体和浏览器窗体的指导原则

伴随用户界面设计的早期活动，分析员必须关注用户能够在屏幕上看到的窗体的设计工作。每个对话可能需要几个Windows窗体。为实现其可用性，我们必须细心设计每个窗体。目前几乎所有的新型商务系统都在交互式的Microsoft Windows、X-Windows（UNIX）或Macintosh环境中开发。无论哪种环境，窗体设计的基本原则是一样的。本节，如果我们提及窗体或窗口，实际上是指这三种环境中的任意其一。然而，无论在哪一种Windows环境之中，我们都应该注意区分标准窗体和浏览器窗体。

标准窗体是指那些利用功能齐全的程序设计语言编制出的窗体，例如Visual Basic、C++或Java语言。标准窗体的优势在于其很好的灵活性以及能够直接访问工作站的数据。另一方面，浏览器窗体是使用符合互联网规范的HTML或脚本语言编制而成的，例如VB Script或Java Script。任何一种浏览器都可以显示浏览器窗体。这种特性使得浏览器窗口可以在不同的平台上运行。采用Visual Studio.NET开发的浏览器窗体称为Web窗体，其灵活性可与Windows媲美。此外，利用活动服务器页面（Active Server Pages，ASP）或Java Servlet程序进行服务器端处理工作能够增强功能。浏览器窗体的优势在于同一窗体既可以服务于内部员工又可以应用于互联网上，因此许多公司将浏览器窗体作为其新系统的用户界面。

浏览器窗体：使用符合互联网规范的HTML或脚本语言编制而成的窗体。

在确认窗体的目标和与之相关的数据之后，系统开发者可以用多种原型工具来开发窗体。早期，开发者往往在编程之前花费大量时间在纸上设计窗体布局，但现在则利用更具效率的原型工具。这样，设计者在设计窗体内容的同时也设计了窗体的外观效果。这种方法还能够使用户完全参与到开发过程中来，由此而设计出真正的用户界面，提供很强的归属感和可接受性。

窗体的种类包括输入窗体、输入/输出窗体和输出窗体。输入窗体主要用于记录事务或输入数据，虽然窗体的某些部分用于显示来自系统的数据。图13-9中城市影碟出租系统故事脚本所用到的窗体就是一个输入窗体。输入/输出窗体一般用于更新已有信息。这种窗体显示某个实体的信息，例如顾客信息，并且允许用户键入新信息以及更新已有信息。输出窗体主要用于显示信息，其设计原则与第14章将要讨论的报表设计原则是一致的。输入窗体和输入/输出窗体关系密切，并且使用相似的设计原理。每一个窗体都包含了数据输入所需的一些控件。我们应该仔细分析这些控件的完整性，参见第14章内容。在窗体设计过程需要考虑的4个主要问题是：

- 窗体布局与格式化
- 数据输入项
- 导航与支持控件
- 帮助支持

13.5.1 窗体布局与格式化

窗体布局与格式化主要关注窗体的整体外观和感觉。每个人都可能遇到过输入窗体非常

不好用的系统——或者字体太小，或者标签让人看不懂，或者颜色令人不适，或者导航按钮不明显等。而另一方面，容易使用的窗体往往布局合理的，不同的信息容易便于用户查找和理解。确保窗体布局合理的方法之一是创建不同的设计方案后由用户来检验。用户会让你知道哪些特点是有用的，那些信息容易造成使用过程中的迷惑和误解。当你设计输入窗体时，应该考虑以下内容：

- 一致性
- 标题、标签和标志
- 文本框和按钮的组织与分布
- 字体大小，亮度对比和颜色

我们把一致性列在最前面，是因为它对轻松学习和使用的重要性，这一点先前已经讨论过。一些大型系统需要很多输入窗体，这些窗体通常由几组程序员/分析员共同开发。不同小组的人员在开发过程中往往缺乏交流，所以不同小组开发出来的窗体风格各异，从而导致系统的不一致性。为了避免这样的局面，系统所有窗体需要具备相同的感官效果，功能键的连贯使用、快捷键、控制按钮甚至相同的颜色和布局等都会使得系统更好用且更专业。设计者通过层叠样式表（Cascading style sheets）来保证Web窗体的一致性，通过模板来保证Windows窗体的一致性。例如，通过Microsoft Visual Studio.NET设计的模板作为设计中其他窗体的父类。

窗体上的标题，标签和标志有助于体现窗体的功能和用法。位于窗体顶部，含义清晰的标题使得混淆窗体的可能性降到最低。标签也应该是易于辨识和阅读的。

设计者也应该妥善安置窗体上的文本框。相关的文本框通常放在一起，甚至可用一个细线框将它们与其他文本框隔离开。设计者还应该仔细考虑Tab顺序。如果输入数据来自于纸张表单，Tab顺序应该沿用纸张形式中的顺序，即从上到下，从左到右。窗体中要留出空余区域以免给人文本框太拥挤的感觉，而且这也有利于不同域的区分和阅读。在正常情况下，按钮位于窗体底部，按钮放置位置的事实标准主要来自于Apple，Microsoft，Sun和Oracle等大型软件公司的标准。虽然业界相关技术不断发展更新，但是这些传统标准一直得以沿用和遵循。

强调字体大小，高亮显示和颜色等特征的目的在于使窗体便于阅读。不同字体的合理搭配，常规和粗体的交替使用，多种字体的巧妙运用以及恰当的背景色都有助于用户找到窗体上的重要信息或关键信息。但变化太多的样式可能使得窗体变得复杂而难以使用。但是，这些技术的合理使用能帮助用户更好地理解窗体的用途。比如说，关标题和总计可以设计成大一些的字体或粗体字体以示醒目，或者通过改变字体颜色来突出负值或缺失数额。而且字体颜色和背景色应该搭配协调以便于用户阅读，例如，绿色、黑色背景上的红色字体或者深色背景上的深色字体都不是明智的选择。因为有些有色盲症的用户往往不能区分红色与绿色或红色与黑色。



图13-13　RMO产品明细窗体，用于查找产品信息、选择大小和颜色并将其附加到订单上

RMO客户支持系统中的一个窗体如图13-13所示，该窗体用来查询产品信息并将查询出的产品添加到订单中。观察图中的标题和标签是如何布局的。图中的布局方式是从上到下的，并把相关的元素放在一起，导航和关闭按钮也很容易找到，并且不会影响其他数据的输入。

13.5.2 数据的键控与输入

任何输入窗体的核心任务是新数据的输入。在这里，我们的主要目的是尽可能地减少输入数据量。任何计算机已有信息或者由计算机生成的信息应该不必重新输入。常用选择列表、复选框、描述信息字段的自动检索等能够提高输入速度并减少错误。图13-13中的RMO产品明细窗体展示了几种减少输入数据量的方法。

目前有几种广泛用于窗体系统设计的数据输入控件。文本框是实现数据录入的最通用控件。文本框是能够接受键盘数据的矩形框。在多数情况下，在文本框旁会附带有描述文本框内容的标签。我们可以将文本框指定为单行模式，也可以将文本框指定为具有滚动能力的多行模式。对于单行模式，可以指定文本框能接受的字符的长度。

文本框的变体可以是列表框、微调框和组合框。列表框中包含一个可接受数据项列表。该列表是一组预定义的数据值，用户可以从中选择其一。列表可以显示在矩形框中或者以下拉列表形式出现。微调框是列表框的变体。微调框在文本框自身内部提供可能的数据值，两个微调箭头允许用户在众多数据值间滚动。组合框也包含可接受数据项的预定义列表，而且还允许用户输入不包含在预定义列表中的新值。列表框和组合框的使用实现了数据输入的击键次数最小化并且相应地降低了出错可能性。

文本框：接收键盘输入的控件。

列表框：包含可供用户选择的数据项列表的控件。

微调框：列表框的变体。在文本框中提供了供用户选择的数据项。

组合框：列表框的另一种变体。允许用户输入新值或者从列表中选择。

有两种输入控件以成组方式使用：单选按钮和复选框。单选按钮成组出现。用户每次只能选择其中一项。一旦用户选择某个选项后，系统将自动关闭其他所有按钮。因为所有可能的数值都要显现在窗体上，所以这种控件要求选项数目较少并且选项值要保持不变。复选框也是以成组形式使用的。但是，多个复选框之间不是互斥关系。用户可以同时选中多个复选项。图13-14所示为包含上述数据输入控件的一个窗体。



图13-14 输入窗体上的数据输入控件举例

单选按钮（选择按钮）：输入控件，用户可以从一组选项中选择其中某一个选项。

复选框：输入控件，用户可以从一组选项中选择不止一个选项。

浏览器窗体也有类似的控件。标准窗口输入和浏览器窗体输入的主要区别在于，标准窗体的数据域编辑操作方式是边输入边编辑，而浏览器窗体的编辑工作在整个窗体传送到服务器后才执行。但是随着浏览器程序的日渐复杂化，浏览器会提供越来越多的数据输入功能。现在，标准窗口输入窗体和浏览器输入窗体几乎具有同等的处理能力。

13.5.3 导航与支持控件

标准Windows界面提供几种用于导航和操纵窗口的控件。对于微软公司系列的应用程序，这类控件包括窗体右上角的最小化、最大化和关闭按钮，水平和垂直滚动条，位于左侧面板上的记录选择条和窗体底部的记录导航箭头等。为了维护系统一致性，要求尽可能地利用这些导航控件来实现界面设计。良好的用户界面设计还应该包括其他控件或按钮。你可以在窗体中放置按钮，使用户能转向其他相关屏幕去搜索和发现数据或者关闭窗体。浏览器窗体也提供导航和支持控件。浏览器中包含导航按钮和控件，所以该应用程序应该支持浏览器窗体

的使用。每个页面可能都包含属于它自己的导航按钮。

13.5.4　帮助支持

输入窗体的主要设计目的是使其具有直观性，即用户无须帮助功能即可使用。然而，即使是定义合理的窗体也可能令人产生误解，所以仍然建议系统带有在线帮助。当前的系统中存在三种通用类型的帮助信息：与窗体的使用流程相关的指导信息，帮助主题的索引列表，上下文相关的帮助信息。

大多系统都是供用于培训新用户的帮助信息。帮助信息可以按任务进行组织，一般是一个对话框一组相关窗体。每一个新系统都应该具有帮助主题的索引列表，用户可以通过关键字搜索该列表，或者像许多Microsoft软件系统一样通过帮助向导来检索帮助主题。帮助向导是一个简单的自动搜索程序，以用户输入的问题或语句作为搜索关键字，根据关键字搜索的结果返回给用户多个可选的帮助主题。

上下文相关帮助以帮助信息的索引列表为基础，但是触发方式不同。上下文相关帮助根据光标当前所在位置自动显示适合的帮助主题，换句话说，如果光标在某一文本字段中，并且用户请求上下文相关帮助，则该文本字段的帮助主题显示出来。

13.6　网站设计指导原则

网页设计原则源于标准窗体和浏览器窗体设计的指导原则和规则。目前许多商用系统，包括RMO公司的客户支持系统在内，都是采用这两种技术实现的。而RMO公司系统的订单处理功能其实就是RMO网站的一个组成部分。网站还往往用于内部人员交流，客户信息与服务，在线销售，派送和营销。网站可以实现与客户进行一周7天每天24小时的无缝交互。本节将讨论网站设计的原则和经验教训。详细的网站设计原则不在本书的讨论范围。许多优秀的书对网站设计原则进行了详细介绍，本章的“参考资料”部分将列举一些。

13.6.1　网页设计中的10种好的做法

Jacob Nielson是一位HCI研究人员，他目前正致力于网页设计的研究。同其他有用的指导原则一样，Jacob Nielson的研究重点放在了一般性问题上，包括“网页设计中的10种好做法”。

1. 将机构名称和标志放置在所有页面上，并建立标志与主页的链接。

2. 如果网页数量超过100，应该提供搜索功能。

3. 书写简洁的标题行和页面标题。这些内容有助于解释页面功能，而且对在搜索引擎列表上读取这些内容很有意义。

4. 构造页面的原则是便于读者浏览并帮助读者在匆匆一瞥中找到关键内容。例如，利用分组方式和副标题将大的篇幅分割成几个小的单元。

5. 不要将有关某一产品或某一主题的所有内容拥挤地塞满单一页面，而应该使用超文本来构造内容空间。即由一个开始页面提供概述信息，其他页面内容分别着重于某一个特定的主题。

6. 可以使用产品照片，但要避免在产品系统页面上混乱而繁杂地堆满照片，主要产品页面必须做到迅速地加载和执行功能，所以其中内容应该短小精悍。

7. 页面上准备放置小的照片和图像时，要利用相关增强图像缩影技术。创建原始图像缩影时，不是简单地缩小成看不清楚的小东西，而是在剪切图片、缩小尺寸的同时放大相关部分的细节。

8. 利用链接标题为用户提供链接内容预览信息。

9. 要保证所有重要的页面都能被残障用户访问到，特别是有视觉障碍的用户。

10. 工作方式总与主流保持一致。如果许多大型网站都按特定的方式工作，用户往往期望其他网站也采用类似的方式。

13.6.2 网站设计原则

由于网站涉及许多方面的内容，所以网站设计者从更广泛的角度来研究网站设计原则。在Joel Sklar的网站设计书中，作者认为设计者应着眼于网站设计的三个方面：（1）计算机媒体设计；（2）设计整个网站；（3）为用户设计。

1. 计算机媒体设计

网站要在计算机屏幕上展示，而不是在纸上展示，记住这一点很重要。虽然设计者有多种字体、颜色和布局方式可选，但网站的外观取决于其功能和组织机构的目标。超媒体使用户可以以非线性的方式访问网站，因此设计者应充分利用超媒体来组织信息。可以考虑以下5个原则。

- 精心设计网页的外观及感觉以充分利用媒体介质。
- 由于要保证具有相当广的技术范围内的可访问性，因此要使得设计具有可移植性。
- 要考虑低带宽，因为用户不会有耐心去等待网页加载。
- 规划好网页的展示方式，与可能易于访问，以便用户能够在网站中轻松浏览。
- 若在线展示的信息来自其他站点资源，需要对这些信息重新格式化。

2. 设计整个网站

整个网站必须有统一的主题和结构，主题应反映出公司想要传达的理念。例如，若网站的用户对象主要是成年的业务用户，则网页应该使用柔和的颜色、熟悉的业务字体和结构化的线型布局。若网站的对象是儿童，则网页应结合明亮的颜色和开放、友好的动态结构，以及简单有吸引力的画面。可以考虑以下4个原则。

- 精心设计网页的外观和感觉，以使和设计者想要表达的理念一致。
- 网页之间创建平滑的过渡，以便用户能清楚地知道自己所处的位置。
- 用网格线来设计每个网页，以便为相关的信息组提供可视化的结构。
- 在每页的信息组之间预留一定数量的空白。

3. 为用户设计

本章前面部分，我们讨论了以用户为中心的设计。把网站设计的重心放在用户和他们的需求上是很重要的。如果有某个特征让用户厌烦或分心，就将此特征去掉。有时判断网站的用户是谁是很不难的，但如果整个网站的目标和目的明确后，设计者便能做出更好的判断。可以考虑的一些原则如下。

- 设计网站的交互性，因为网站用户往往期望网站是交互和动态的。
- 使网页上的信息能吸引用户的眼睛，这一点很重要。
- 保持浅层次的分层结构，使用户不用进入太深就可发现详细的信息。
- 利用超文本使用户能在网页中浏览。
- 每页网页的内容多少，要根据用户的特征决定，但不要把网页弄得很凌乱。
- 为不同群体的用户设计网页，包括残疾人。

13.7 RMO对话设计

我们已经讨论了对话设计的概念和技术，现在我们可以讲述为RMO设计某个特定对话的设计过程，同时这也是RMO公司网站设计的一部分。

13.7.1　电话订购业务员的对话设计

RMO客户系统支持电话订购业务员处理客户订单这项业务。这项对话活动与客户创建新订单这一事件有关，更具体地，讲与电话订购业务员创建新订单的场景有关。系统中，与此相关的目标环境是Windows平台上的电话订购业务员的台式PC机。

系统分析模型是设计者的设计起点。这些模型或者是数据流图片段和相应活动的详细数据流图，或者是该场景的顺序图，依赖于设计阶段采用的方法。这些模型包含的4个基本步骤如下：

1. 记录客户信息
2. 创建新订单
3. 记录交易细节
4. 产生订单确认信息

如果采用传统开发方法，那么第10章已经介绍的结构图与这些步骤相一致。如果采用面向对象方法，这个场景的顺序图就要扩充至包含用户与系统进行交互的窗体。如第11章所述，不管哪种方法，对话框设计要把用户界面设计同活动与用例的设计过程相结合。

基于所需处理过程的顺序，基本对话可以书写出来，以比较细化的形式表现用户期望的对话流程。界面对象的细节要等到对话精化阶段再充实进去。通过电话下订单的新顾客可能会与电话订购业务员进行如下内容的对话。

计算机：客户要选择哪一功能选项？

用户：我要创建一份新订单。

计算机：好的，是新客户还是老客户？

用户：新客户。

计算机：好的，请输入客户姓名、住址、电话等信息。

用户：新客户是Ginny Decker，11980 Visual Blvd ……

计算机：好的，客户想订购的第一项物品是什么？

用户：女式长统靴，茶色，属于Spring Fling目录Hiking Supreme Line 品牌系列。

计算机：存货有这些尺寸和颜色，9码除外。

用户：好的，茶色，8码，一双。

计算机：价格是65.50美元，含税、邮费和手续费，还有其他需要吗？

用户：是的，雨衣，女式的，属于Spring Fling目录Run品牌系列。

计算机：存货有这些尺寸和颜色，蓝色的中、长号除外。

用户：好的，棕色小号一件。

计算机：价格是87.95美元，含税、邮费和手续费，还有需要吗？

用户：不用了。

计算机：客户打算如何托运货品？

用户：UPS两天航空。

计算机：客户是否使用信用卡或者其他付款方式？

用户：用MasterCard卡，卡号是××674-22-××××，有效期到2008年1月。

计算机：好的，订单完成。总结一下：客户姓名Ginny Decker，通过UPS两天航空托运；Hiking Supreme Line 品牌女式长统靴，8码，茶色，65.50美元；Run品牌女士雨衣，小号，棕色，87.95美元，总价153.45美元外加9.20美元税费和13.40美元运费和手续费，总计176.05美元，付款方式为MasterCard卡，卡号：××674-22-××××。

用户：谢谢！

处理这一对话时，设计者可以着手定义用户与计算机交互用的窗体。细化的窗体列表可能包括以下内容：

- 主菜单窗体
- 客户窗体
- 条目搜索窗体
- 产品信息明细窗体
- 订单汇总窗体
- 托运和支付选项窗体
- 订单确认窗体

有了窗体列表，设计者就可以定义从一个窗体到另一个窗体的交互流程的设计概念。如图13-15所示是用顺序方法的设计概念实现创建新订单的过程。主菜单窗体出现之后，先是出现客户窗体，用户在其中填写或更新信息，接下来显示条目搜索窗体，以便用户搜索一个条目。某一条目的产品细节信息被搜索并显示出来以后，条目搜索窗体会再次显示出来。当所需条目都选择出来以后，订单汇总窗体就会显示出来。设计者应该把精力集中放在对话窗体的突出部分而不是操心每个窗体的物理设计。考虑好每个窗体所需信息之后，设计者可以创建更详细的故事脚本，或利用VB等工具来构造窗体的原型。

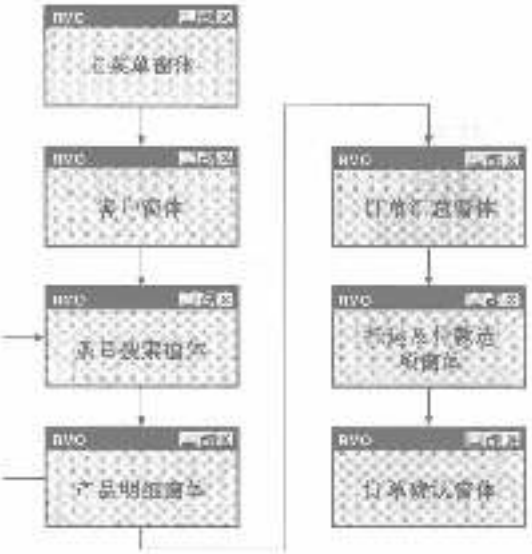


图13-15 用顺序方法的设计概念实现创建新订单的过程

第一次迭代过程的设计结果是非常有序的但不太合理。RMO的电话订购业务员对故事脚本和原型进行了评价，并且指出该顺序过于苛刻，因为其假定每次对话总是遵循相同的步骤次序。然而，在电话订购业务员与客户通话过程中，他们必须遵从客户的指引。有些客户直到订单完成并确认后才给出私人信息。有时客户要先知道订单总额或要求返回检查订单中某些条目的细节信息。但是，顺序设计总是假定客户信息最先给出。虽然顺序方法可能适合邮购订单业务员的操作，但电话订购业务员的操作却要求更高的灵活性。

实践指导

为用户界面编写故事脚本时，会发现许多窗体都可以重用。在项目早期便设计一些可以重用的窗体，这些窗体不仅可以节约精力，也可以提高系统的一致性。■

由于用户信息需求和灵活性的原因，项目组提出了第二个设计概念。这个设计概念以订单窗体为对话的中心，通过选项可以任意切换到其他窗体。每一个动作之后，订单窗体再次显示给电话订购业务员，其中附加了当前订单的详细信息。以订单为中心的设计概念如图13-16所示，

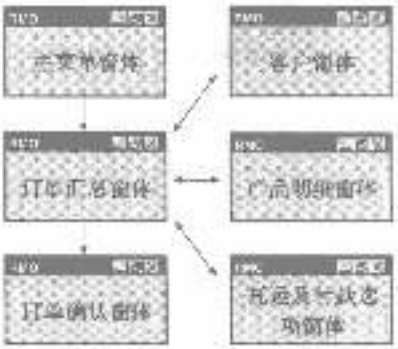
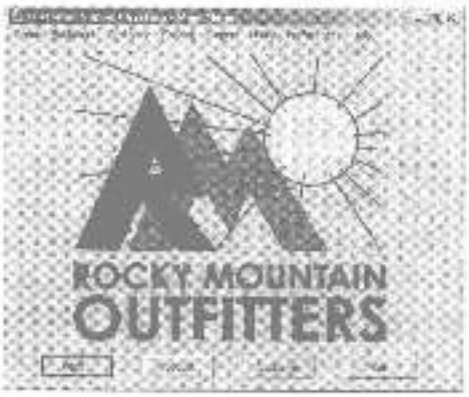


图13-16 用以订单为中心方法的设计概念实现创建新订单的过程

它既支持基本对话的次序，也允许在必要的时候做一定的调整。这种设计概念在对话过程中为用户显示更多有关订单的信息（如果用户需要这些信息）。

采用以订单为中心的设计概念后，项目组设计出具体的窗体。图13-17列举了部分窗体。用户从主菜单中选择“下订单”按钮，系统显示订单汇总窗体并给出了一个新的订单号。用户可以立即增加顾客信息（依据顾客号或姓名搜索老顾客信息或增加新顾客信息）。用户搜索所需条目并在产品明细窗体中查找该条目的相关细节信息。如果用户想订购该条目产品，就将其添加到订单中并在订单汇总窗体中重新显示。用户可增加另一条目到订单中，或者修改先前的条目或转去选择托运选项。这种灵活性和信息显示方式是电话订购业务员所期望的操作方式。要想获得最佳设计要经历几次迭代和用户评价的过程。



a）以订单为中心的方法进行对话设计后的主菜单窗体



b）以订单为中心的方法进行开始创建新订单对话设计后的订单汇总窗体



c）用户找到某个产品后的产品明细窗体



d）用户添加几种产品后的订单汇总窗体



e）已完成订单的托运和支付选项窗体

图13-17　以订单为中心的方法进行对话设计后的原型窗体

13.7.2 RMO网站对话设计

RMO客户支持系统中，“客户下订单”事件需要几种不同的对话设计（每一场景一种对话），包括刚刚讨论过的电话订购业务员的操作场景和邮购业务员的操作场景。而且系统另一关键目标是实现顾客使用万维网直接下订单，直接与系统进行交互。公司网站的完整设计已超出本书内容，但有些一般性的规则和指导原则仍适用于直接的基于万维网的客户订单的实现。

客户与计算机的基本对话与电话订购过程相同，但网站必须为客户提供更多信息，而且要更加灵活和方便使用。首先，顾客可能想要浏览有关产品的所有信息，所以，网页需要增加更加丰富的图形，用于显示不同颜色和样式的产品条目。虽然电话订货业务员也要求系统在屏幕上提供详细信息，以便能够回答顾客的询问，但在网络直接交互的方式中，顾客可能希望获得更加丰富的信息。同时信息也需要以不同方式在屏幕上显现。例如，电话订购业务员习惯于数据量大的信息显示，并且知道到哪里去查找所要的详细信息，但顾客需要在首次使用系统时就能够非常容易地定位到详细信息。

系统需要具备非常好的灵活性，因为与系统交互的客户有着各自不同的偏好。在电话订购对话中已经讨论过，对话步骤的先后次序需要具备灵活性——例如，有些客户想先浏览产品并且希望在输入私人信息之前选择产品条目，有些客户则希望系统提供查看以往订单及查看托运和付款方式等功能选项。如果一个客户打算做一些系统不允许的事情，客户会感到无所适从。与电话订购和邮购业务员的操作不同，如果客户在操作中受挫，客户就会直接退出登录而转到其他网站去购物。最终的系统必须做到容易学会和使用，做到用户想都不想就可以操作完成。最初的对话选项必须非常清晰明确，而且顺序过程一旦开始，所有功能都应该做到不言自明。不要期望客户会一直坐在那里接受网站的使用培训，也不要期望他们会查询提示信息或帮助（即使这些信息能够获得）。

正如前面已提到过可视性和可供性的设计原则一样，对于网页的控件来说，设计重点是要使控件能够反映其自身的功能和用法。大多数客户更关注操作的速度而不是花哨的图形或动画，然而设计新手往往以牺牲速度为代价而获得图形及动画效果。另外，因为网站反映了公司形象，所以设计的另一重点是让图形设计人员和市场专业人员参与到设计过程中来。一个考虑周全的可视主题是非常重要的，焦点信息分组和其他反馈技术也应该应用到设计过程中。

RMO的主页如图13-18所示，网页主要强调的是指导客户的交互，客户可以选择学习更多有关RMO的内容，通过电子邮件与公司联系，或者要求获得一份产品目录。网页中没有下订单的菜单选项，实际上，网站是向客户提供通过网络浏览RMO产品的机会。客户可以根据关键字或产品ID号搜索产品信息，也可以从列表中选择某一种产品，而且网站还提供每月特卖信息。当顾客发现了一些他们想要的东西，订单就会创建出来，并且在任何时刻客户都可以改变主意。



图13-18 RMO的主页

RMO网站使用了模拟购物车。顾客一旦找到他们想要的东西，接着报数量、选颜色并处理其他选项，然后向购物车中添加新的物品。顾客在任何时候都可以看到购物车，接着再去浏览商品并选择货品放进购物车。顾客在选购结束后付款并且系统对订单进行确认。图13-19中显示了在顾客选择了女士服装的导航

选项后链接到的产品明细信息页页。图13-20的内容显示了附带订单汇总的购物车。



图13-19　RMO公司网站的产品明细网页

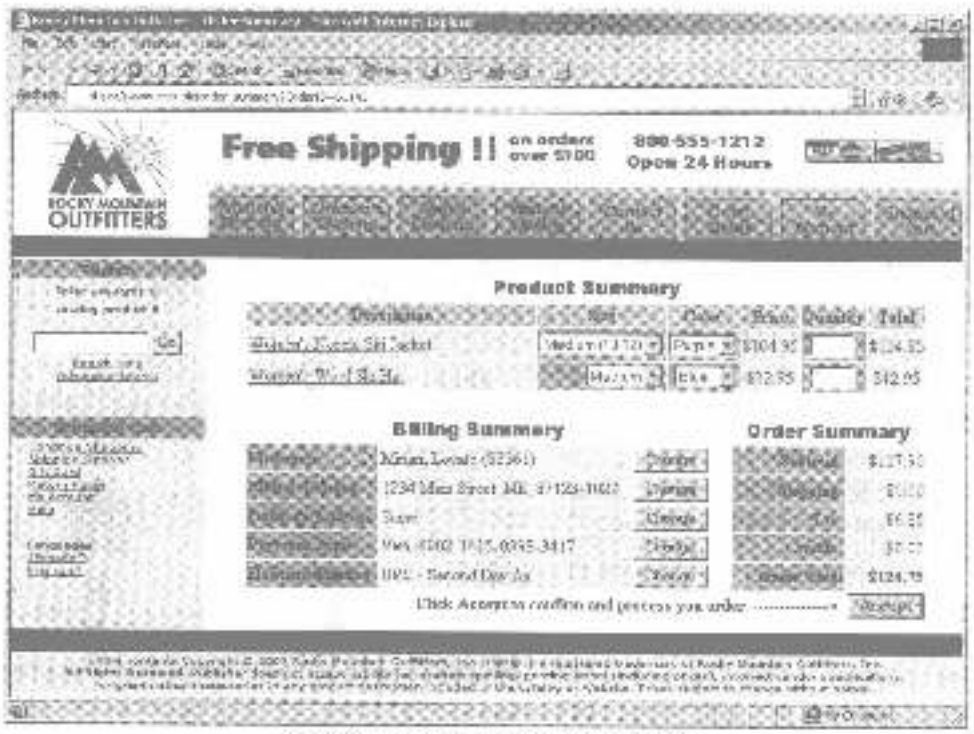


图13-20　RMO公司网站的购物车网页

小结

输入和输出可以分为系统界面和用户界面两类。本章内容侧重于用户界面，并且描述了用于用户与计算机交互设计——人机交互设计的关键概念和技术。在第14章中将描述系统界面，包括系统输出和系统控制。

从物理意义、感官意义和概念意义上来说，用户界面都是用户开始使用系统时所获得的全部内容。对用户而言，用户界面就是系统。用户使用系统（系统模型）所必需的知识包括与对象相关的信息和系统可用功能的信息，即与系统分析员在系统分析阶段辛苦得来的需求模型相关的信息。

以用户为中心的设计方法主要是指及早关注和重视用户及其工作，评价设计方案以确保其可用性，并且使用迭代开发方法。系统可用性是指系统容易学习和使用的程度。保证系统可用性的工作很复杂，因为易于学习和使用的设计方案往往是彼此冲突的。另外，系统的设计要考虑到系统可能要面对各种各样的用户。人机交互是作为源于人为因素工程学（人体工程学）研究的一个领域而逐渐发展起来的，所以人体工程学是研究人与机器间交互的一般原则。

描述用户界面的方式有很多种，包含隐喻、文档隐喻和对话隐喻。对话隐喻强调用户与计算机之间的交互，所以界面设计常被称为对话设计，通过许多途径可以获得界面设计指导原则和界面设计标准。Norman的可视性和可供性原则阐明控件应该是可见的，并提供反馈信息以表明系统正在工作，同时还要明确指示其相应功能。而Shneiderman提出的8条黄金设计规则强调系统设计要尽力保持一致性、提供快捷键、提供反馈信息、设计完善的对话、提供简单的错误处理机制、允许撤销、支持控件的内部轨迹以及减轻短期记忆负担等。

对话设计的最初工作是明确对话及其触发事件。另外在设计阶段还要增加用户偏好、帮助等其他对话功能，菜单层次要设计成能面向不同的用户。一旦对话设计出来，菜单层次就可很容易地重组。像编写剧本一样书写出来的对话次序能够帮助开发者找出对话过程中需要交换的关键信息。显示顺序的屏幕概略图或故事脚本可以表达设计要求，以便与用户一起检查设计，或者利用Visual Basic等工具创建对话原型。面向对象的方法提供UML模型来编制对话设计的文档，包括顺序图、协作图和类图。

对话中的每一个窗体都需要设计，并且要遵循有关窗体布局、选择输入控件、导航和帮助的设计原则，这些原则应用于标准窗体和以网络为基础的系统的浏览器窗体的设计。网站的对话设计几乎等同于其他对话，不同的是，用户需要更多信息和要求有更好的灵活性。补充了被应用到计算机媒体设计、设计整个网站及为用户设计的网站设计原则这部分内容。另外，因为网站反映了客户眼中的公司形象，所以图形设计者和市场专业人员应该参与到设计过程中。

关键术语

Affordance	可供性
Browser forms	浏览器窗体
Check Boxes	复选框
Combo Box	组合框
Desktop Metaphor	桌面隐喻
Dialog Metaphor	对话隐喻
Direct Manipulation	直接操纵
Document Metaphor	文档隐喻

Human Computer Interaction	人-机界面
Human factors engineering（ergonomics）	人为因素工程（人体工程学）
Hypermedia	超媒体
Hypertext	超文本
Interface Design Standard	界面设计标准
List Box	列表框
Radio Buttons（Option Buttons）	单选按钮（选择按钮）
Spin Box	微调框
StoryBoarding	故事脚本技术
System interfaces	系统界面
Text Box	文本框
Usability	可用性
User-Centered Design	以用户为中心的设计
User Interfaces	用户界面
User's Model	用户模型
Visibility	可视性

复习题

1. 为什么界面设计常被称为对话设计？
2. 用户界面系统包含哪三方面的内容？
3. 用于描述终端用户及其与计算机的交互的一般术语是什么？
4. 用户界面的物理意义方面的例子有哪些？
5. 用户界面的感知意义方面的例子有哪些？
6. 用户界面的概念意义方面的例子有哪些？
7. 用户界面作为开发过程中心的相关技术集合是什么？
8. 强调以用户为中心的设计方法的三个重要原理是什么？
9. 表示系统容易学习和使用程度的术语是什么？
10. 工程师感到较手的“人为因素”是什么？对人为因素问题的解决方法是什么？
11. 哪些研究领域对人机交互领域做出过贡献？
12. 哪个研究中心对于目前我们所使用的计算机的特征产生了深刻的影响？
13. 用于描述人机交互的三种隐喻是什么？
14. 屏幕上的桌面是描述人机交互的三种隐喻的例子中的一个吗？
15. 哪种类型的文档允许用户通过单击链接而跳转到文档的另一部分？
16. 哪种类型的文档允许用户链接到文档中的文本、图形、视频和音频部分？
17. 在设计系统界面的时候，开发者必须始终遵守的一般原理和特定规则是什么？
18. Norman提出的保证人机之间友好界面的关键原则是什么？
19. 列出Shneiderman 提出的8条界面设计黄金规则。
20. 什么技术显示了有关对话过程中所出现的显示屏幕的顺序顺序图？
21. 哪些UML图用于显示插入对话中的参与者和问题域类之间的界面对象？
22. 哪些UML图能够用于显示包含在窗口或窗体中的界面对象？
23. 用于商务系统中的三种基本窗口类型是什么？
24. 用于输入文本的输入控件（界面对象）是什么？

25. 用于从列表中选择数据项的输入控件有哪些？
26. 以成组形式出现的输入控件有哪两种？
27. 除了业务员使用的操作界面的可用性之外，顾客直接访问的网站界面时的三种系统可用性需求是什么？
28. 顾客在线购物时直接访问网站的流行模拟方式是什么？
29. 网页设计中三个指导性原则是什么？
30. 网页设计过程中的10种好做法是什么？

思考题

1. 想一想你用过的所有软件，哪些软件存在着易学和易用的冲突？
2. 访问一些网站并上网确用于导航和输入的所有控件。这些控件都显而易见吗？讨论一下控件可视性和可供性的差别。
3. 考虑一下人为因素的解决方案，是改变机器以适应人而不是改变人来适应机器。在你的日常生活中是不是还存在有待进一步改善的机器或系统呢？当前的Windows PC和Apple Mac在可用性上是不是已经尽善尽美？如果不是，哪些措施有助于进一步完善？我们是不是已看到可用性方面的突破，或者已经做出了许多重大进步？
4. 回顾一下城市影碟出租系统中顾客与计算机之间的对话。创建如图13-6所示的表格来表示如何将对话转换成用户语言和计算机语言。讨论将对话转化成表格形式时如何使用逆模型和创造物理模型。
5. 参考一下图13-6所示的表格。创建基于表格信息的电子邮件系统的故事脚本。
6. 阅读下列用户试图利用系统下订单的对话。从易于学习和易于使用的角度来评价对话。

   用户：我想订购产品。

   系统：好的。输入你的姓名和地址。

   用户：我的姓名是Timothy Mudd。一心大街北5139号，Los Angeles, CA 98210。

   系统：好。我们已记录所有信息。请继续输入。谢谢。

   用户：我想订购Acme钻孔压力机，需要用带可调的驱动带。

   系统：好。请继续。

   用户：我想要机器的蓝色橡皮脚底但不要脚制动器。

   系统：好。还有其他需求吗？

   用户：我想利用你们提供的特殊折扣优惠进行托运。

   系统：我们照办。

   用户：好的。就这些。

   系统：对不起，您的交易违反了312交易码，必须终止会话。

7. 参考图13-14所示的窗体，其中显示了数据输入控件，绘制UML类图来显示组成窗体的对象。
8. 复习Visual Basic提供的所有所见即所得的可视化控件，讨论控件如何满足系统的可视性和可供性要求。

实验练习

1. 复习一下在为落基山运动用品商店（RMO）设计新订单对话时所使用的顺序设计理念和以订单为中心的设计理念。考虑什么信息和导航选项是邮购业务员所不需要的。接下来，设计窗体使之应用于RMO邮购业务员处理邮购订单信息。或者绘制窗体概略图，或者使用Visual Basic等工具实现窗体。使用输入控件使得数据输入量最少。

2. 评价所在大学的课程注册系统。列举出与系统进行对话的基本步骤。系统存在哪些不利于实现易学易用的问题？系统以什么样的方式提供灵活性？哪些方式不能获得所需信息？系统是否提供太多的分散用户注意力的信息？
3. 评价所在的大学图书馆的在线目录系统（参见练习2中提出的问题）。写出用户与系统进行交互的对话过程并重新和改善该对话。包括故事脚本显示你的设计的适宜效果。
4. 找到一个整天使用的系统，询问一些用户，让他们说出系统的优缺点，并询问他们是如何学会使用系统的。用户使用系统时所犯的最典型错误是什么，以及如何避免和改正错误。是不是任何系统交互都是"自动"的？
5. 找一个提供顾客直接订购服务的网站。浏览一些产品的描述信息并注意对话和网页的设计。设计的优缺点是什么？依据可视性、可供性以及8条黄金设计准则来评价网站设计方案。依据网络设计过程中的十种好做法和十大错误来评价网站设计方案。

实例研究

21世纪大学系统

考虑用户与21世纪大学课程注册系统之间的对话。用户是需要注册课程的大学生。

系统：你想执行什么样的课程注册功能？

用户：预注册。

系统：好的。你的账号ID和密码是什么？

用户：我的账号ID是SATZ791。我的密码是xxxxxxxxx。

系统：好的，欢迎您返校，Kevin。您在7月7日最后一次修改日程。你想执行什么样的预注册功能？

用户：增加课程。

系统：以下是你目前可以预注册的课程信息。

（系统显示包括三门课程的课程信息列表）

因为你的专业是CIS，这些课程是所有不与你现有日程冲突的CIS专业课程（注意，先决条件未经核对）。

（系统显示CIS专业的可选课程列表）

你的选项如下所示。

- 切换到其他系的课程列表
- 重新显示课程列表，但忽略时间冲突
- 核对先决条件之后重新显示课程列表
- 提供所指示课程的细节
- 将所指示课程增加到你的日程中
- 取消

用户：我想切换到其他系的课程。

系统：好。你想切换到哪一个系？

用户：艺术系。

系统：以下课程是你可以目前注册的课程。

（系统显示包括一门课程的课程信息列表）

根据要求，这些课程是所有不与你现有日程冲突的艺术专业课程（注意，先决条件未经核对）。

（系统显示艺术专业的可选课程列表）

你的选项如下所示。

- 切换到其他系的课程列表
- 重新显示课程列表，但忽略时间冲突
- 核对先决条件之后重新显示课程列表
- 提供所指示课程的细节
- 将所指示课程增加到你的日程中
- 取消

用户：将第二项课程添加到列表。

系统：（系统继续对话过程）

1. 本次对话是逻辑模型还是物理模型？它能适用于一个电话注册系统吗？对话能由一个职员在桌面上手工完成吗？讨论以上内容。

2. 讨论此次对话设计对8条黄金设计准则遵循程度如何。

3. 通过系统交互完成对话实现学生添加课程。学生应该要求查看课程的详细信息，然后决定增加课程，包括学生对最终日程细节信息的查看要求。

4. 使用Visual Basic等工具设计和实现本次对话的故事脚本或原型，并且要尽可能地忠实于原对话。复制样本数据检验你的设计方案。

城市影碟出租系统

本章内容中有一个城市影碟出租系统故事脚本的例子。第7章中首次提出这个实例研究。故事脚本显示了影碟出租的对话。复习该案例并完成以下内容。

1. 使用Visual Basic等工具实现本章的故事脚本原型。

2. 为顾客归还影碟事件写出对话并创建故事脚本。考虑归还并且续租一张或多张影碟的情况。

3. 使用Visual Basic等工具实现故事脚本原型，接着要求几个人对它进行评价。讨论所给出的修改建议。

Waiters on Call系统

复习一下第5章中描述Waiters on Call系统的实例研究。即饭店的送餐服务。系统分析员找到至少14个用例，其内容已列举在案例中。

1. 依据列举出的用例为系统创建一套菜单层次体系。接下来为菜单补充菜单项，并提供基于事件的系统实用功能、用户偏好和帮助等。

2. 案例中列举的最重要的事件是顾客打电话来订餐。按照自然顺序写出用户和计算机之间的对话并且实现信息交换。

3. 绘制实现对话所需窗体的故事脚本。

4. 让若干人对设计方案进行评价，并讨论所有的建议修改内容。

5. 使用Visual Basic等工具实现最终对话的设计原型。

国家巡查罚单处理系统

复习一下第5章中作为实例研究介绍的国家巡查罚单处理系统。

1. 依据案例中列举的用例为系统创建一套菜单层次体系。接下来为菜单补充菜单项，并提供基于事件的系统实用功能、用户偏好和帮助等。

2. 按照自然顺序为“空乘递送新罚单”用例写出用户与计算机之间的“记录新票”对话，实现相应的信息交换。

3. 绘制实现对话所需窗体的故事脚本。确定使用列表框、单选按钮和复选框等输入控件，使

的输入数据量最少。

4. 让若干人对系统进行评价，并且讨论所有的建议修改内容。

5. 使用Visual Basic等工具实现最终对话设计原型。

对落基山运动用品商店实例的再思考



RMO的少数电话订购业务员对本章所提出的以订单为中心的对话设计方案不是完全满意。他们建议将设计方案进一步简化成只有一个窗体——订单汇总窗体。他们认为订单汇总窗体经过扩充可以显示附加信息而不必切换到单独的用户信息、产品信息或者运送信息的窗体。在不需要显示附加信息时，窗体可以压缩。他们认为这种方法对集中注意力更有好处，这样用户不必费力地从一个窗口注意到另一个突然出现的窗口，并且如此反复。交互窗体中可以将这两种选项都包含进来，允许用户根据个人偏好选择其中任意一种。

1. 绘制故事脚本，体现单一窗体设计理念，将顾客信息、产品详细信息和托运信息附加到扩充窗体中。

2. 使用Visual Basic等工具实现故事脚本原型。

3. 让一些人对设计方案进行评价，特别要对比本章提出的以订单为中心的设计方案。讨论相应结果。

4. 你还能描述或者实现其他的可选方案吗？

关注Reliable Pharmaceutical Services



Reliable Pharmaceutical Services系统包括以下三类用户：在Reliable办公室处理订单的用户，下订单的用户以及在客户卫生保健机构监控订单信息的用户。考虑关系到Reliable员工以及客户卫生保健机构员工的事件和用例。

1. 分别为Reliable员工和客户卫生保健机构员工设计两个不同的菜单等级。

2. 为客户保健机构员工的“下订单”用例写出用户与系统之间的对话步骤。

3. 通过粗略绘制网页交互顺序草图，创建“下订单”对话的故事脚本。使用像Dreamweaver或Frontpage这样的工具，实现这些网页的原型。

参考资料

Merlyn Holmes, *Web Usability and Navigation*. McGraw-Hill Osborn,2002.

Patrick J Lynch, and Sarah Horton, *Web Style Guide: Basic Design Principles for Creating Web Sites*. Yale University Press, 1999.

Deborch J Mayhew, *Principles and Guidelines in Software User Interface Design*. Prentice Hall, 1992.

Jakob Nielsen, *Designing Web Usability: The Practice of Simplicity*. New Riders Publishing, 2000.

Donald Norman, *The Design of Everyday Things*. Doubleday,1990.

Jenny Preece, Yvonne Rogers, David Benyon, Simon Holland, and Tom Carey, *Human Computer Interaction*. Addison Wesley,1994.

Ben Shneiderman, *Designing the User Interface*, (3rd ed). Addison Wesley, 1998.

Joel Sklar, *Principles of Web Design*, (3rd ed). Course Technology, 2006.

第14章　系统界面、控制和安全的设计

学习目标

阅读本章后，你应具备如下能力：

- 理解信息系统中的系统界面
- 根据应用程序需求定义输入和输出
- 为系统用户合理地设计对话和显示报表
- 解释完整性控制的重要性
- 识别输入、输出、数据和处理所需的完整性类型
- 理解影响信息系统设计与操作的安全性问题

本章要点

- 确定系统界面
- 系统输入设计
- 系统输出设计
- 完整性控制设计
- 安全性控制设计

Downslope滑雪用品公司：设计一个安全供应商系统界面

Downslope滑雪用品公司是一家中等规模的滑雪运动产品生产商。该公司以生产速降滑雪运动器材起家，因此有了现在这样的公司名称。但是，几年以后，它的业务已经从生产滑雪板扩展到滑水运动器材的生产。在公司的早期阶段，生产工艺相当简单。然而，随着先进生产原料的引进，如碳化树脂（carbon-laced resin）以及其他复杂化合物，生产变得相当复杂，需要对原料成分混合物的配比及其在烘焙炉中的温度耐受能力进行精确控制。为了保持滑雪板生产的连续性，Downslope公司对其购进的产品生产原材料的质量提出了非常严格的要求。最近几年，公司又不得不多次更换供应商以保证高质原材料的足量供应。

除了对产品的原材料进行质量控制之外，Downslope公司实施了一项改进的实时（Just-in-time，JIT）生产过程，这就意味着公司不再存储大量的生产原料。通常，在手头要存放能供应5天生产用量的原料，并且要求供应商能够至少在一周内为其补充原料。为了实现多种原材料的快速订货和交货，Downslope公司的管理高层决定允许其供应商访问该公司原料的库存数据库。由于所生产的滑板类型不同，不同类型的滑板原材料的消耗速度也不同。因此，Downslope公司着手于开发和安装一套与生产过程集成在一起的复杂库存管理系统。

Nathan Lopez是Downslope公司的系统开发项目经理，他很快意识到为公司的供应商提供关于数据库访问的系统界面比他预想的要复杂。他用了两个月的时间研究供应链管理的电子方式及其替代方法的可行性，现在他要向Downslope公司管理层汇报两个月的研究结果。

"我已经与所有供应商会面并确定了他们各自所需信息及其相应的格式。与预期的情况相同，他们想要的数据格式和我们所预期的相差很大。我只能将他们期望的数据格式加以限定并归纳为三种基本格式。起先，我们确信我们的供应商能够接受我们的输出设计。然而，现

在看来这不是一个好主意。如果我们的系统不具有灵活性，那么我们公司将很难增加或变更供应商。相反，如果我们正确地构造界面，并配备几种不同版本，那么供应商会更容易地访问到我们已授权的数据。”

“另一个重要问题是，数据和系统的完整性和安全性问题。例如，我们的生产过程与众不同，这是我们的竞争优势之一，如果某些有恶意的公司访问了我们的数据，可能会分析我们的使用模式，这样一来，他们不但会猜到我们所使用的原料而且还可能会解构我们的生产过程。为了保护我们的数据，我们必须做到只接受安全访问。另外还需要保证传送给供应商的数据在传输过程中和到达目的地后的安全性。这些安全性问题不仅与我们的系统有关，还与供应商有关。”

会议持续了很长时间，其间相当多的讨论是有关通过外部系统界面开放公司系统的机遇和危险的。公司董事会最终决定，Nathan应该再用几周时间研究具体情况并列出安全性问题及其解决方法的详尽方案。只有这些问题得以解决之后，开发项目才能继续开展下去。Nathan知道这种新型系统界面的开发将是一个难题。他希望能够找到所有现存问题的解决方法。

概述

大多现代信息系统包含外部输入和输出（I/O），而许多人或机构需要访问存放在信息系统中的数据。第13章中介绍了有关人机界面（HCI）和用户界面的内容，其中I/O作为用户与计算机交互作用的结果。但是许多系统的输入和输出操作不包含人的交互。许多这样的系统界面不与最终用户直接见面。但是系统分析师需要对已有系统、数据库和网络技术进行深入了解，以便综合考虑信息系统所需要的所有I/O需求。因此本章我们将讨论与用户界面分离开来的系统界面。

许多系统界面是针对外部代理的电子传输或纸面输出，包括报表、声明和账单等。这些输出需要针对特定的目的来进行确定和设计。通常，系统输出的质量是整个系统质量的标志，甚至是系统所属公司质量的标志。本章即讨论有关这些输出的设计。

由于输入和输出数量多并且变化多端，系统开发者需要设计和实现完整性控制和安全性控制，以便保护系统及其内部数据。如今越来越多的信息系统存在于开放环境中，所以对于系统控制的设计就显得越发重要。现在的信息系统作为网络的一部分可以为许多机构和个人提供广泛的信息访问机会，所以，系统设计时要考虑的一个重要方面就是，如何避免错误或欺诈性交易进入系统。完整性控制功能在数据输入和处理时进行数据有效性的验证，内部检查和交叉检查有助于保证对数据完整性的控制。本章内容将讨论完整性控制实现技术，以防止错误、欺诈以及操作误用的发生。

最后，尽管存在系统安全威胁，但业务机构和个人仍然是连接到Internet的公司最主要的关注对象。许多公司利用Internet作为市场和营销渠道，因此他们需要允许客户或潜在客户能够登录公司系统。同时又必须拦截入侵者和恶意黑客。另外，在电子商务事务中必须传递一些客户个人信息，例如信用卡号、金融交易信息等。本章的最后一部分将讨论安全性控制并阐述数据保护、数字证书和安全交易的基本概念。

14.1　确定系统界面

前面章节所描述的用户界面包括了需要系统用户直接参与的输入和输出。除此之外，还存在着许多其他系统界面。我们将系统界面广义地定义为不需任何用户干预或用户干预很少的输入和输出。属于系统界面的标准输出有：账单、报表、打印表格以及流向其他自动化系统的电子输出。属于系统界面的输入包括自动化输入和来自于非用户界面设备的输入。例如，

来自于自动扫描仪、条形码阅读器、光字符识别设备以及作为系统界面组成部分的其他计算机系统的输入。

在分析和设计信息系统时，用户界面往往被视为最首选的，因而也是最重要的界面。事实上，在对人机交互界面的理解，以及应用以用户为中心的设计原则来设计用户界面方面已取得相当大的进步。然而，目前以高集成度和互连性为特征的信息系统已逐渐地超出了用户需求，需要系统界面更快速、更有效、更准确并且全天候地处理输入和输出。

系统界面能处理输入，与其他系统进行实时交互，并且在最少人员干预的情况下分发输出。在系统研究和系统需求建模时，系统分析师必须细心寻找那些并不明显的系统界面。在设计系统时，设计人员应该考虑人机界面的替代方式以便自动地捕获输入和分发输出。信息系统中完整的输入和输出如图14-1所示。

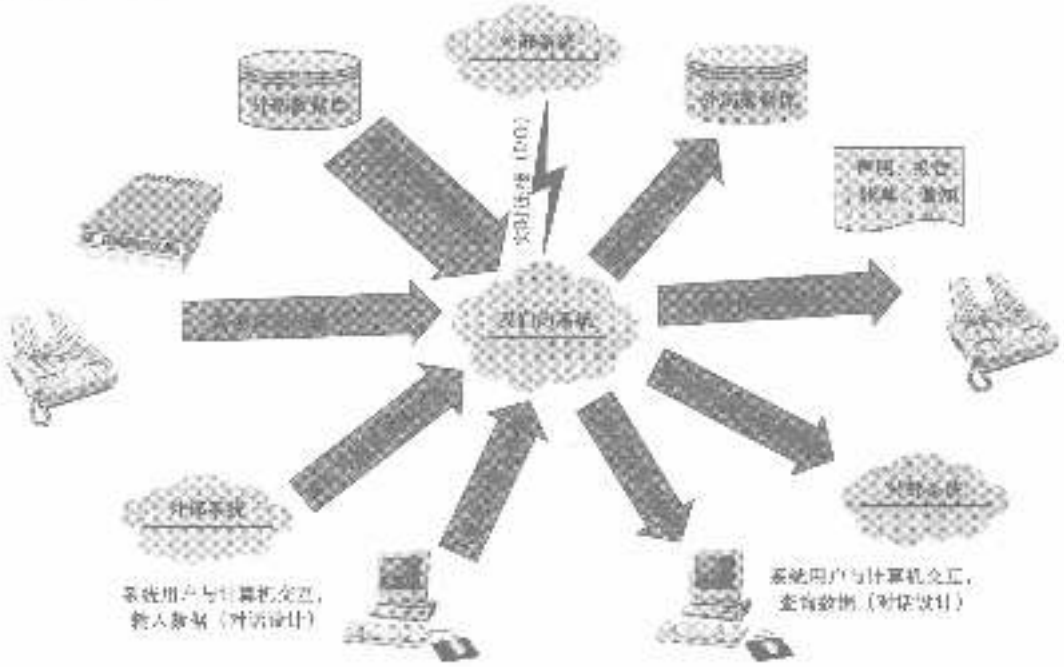


图14-1 信息系统中完整的输入和输出

下面的列表列出了系统界面种类，这将有助于确定I/O需求和设计可能性：

- 来自于其他系统的输入
- 高度自动化输入
- 来自于外部数据库的输入
- 流向外部数据库的输出
- 具有极少人机交互（HCI）的输出
- 流向其他系统的输出
- 实时连接（输入和输出）

实践指导

信息系统的相互链接和实时运行，使得系统界面越来越重要。尽可能使用能自动生成系统界面的工具和方法，因为这对系统的性能有重要影响。■

输入可以是直接来自于其他系统的网络消息。电子数据交换（EDI）和许多基于网络的系

统可以通过消息与其他系统实现集成。收到的消息触发系统进行处理活动的方式类似于用户与系统的交互方式。例如，在RMO公司的集成供应链管理和客户支持系统中，来自供应商的物品的到达会触发客户订货的托运操作。不需用户干预，并且作为结果，这项事务会马上被处理并且不会出错。在基于网络的系统中，一个单独的购物车订单输出应用程序可能发送一个消息给订单完成系统，以便处理新的订单。系统分析师通过确定系统范围来决定输入是什么以及内部消息是什么。

像扫描仪这样的高度自动化输入设备能够捕获许多系统输入。例如，从仓库中挑选的准备发货的货品应具有工作人员能扫描的可读标签。这种高度自动化处理代表了系统的一项输入。在某些情形下，当一件货品在传送带上移动时，一台扫描仪可能会记录输入——根本不需用户干预。以后我们会讨论专门的输入设备。

许多输入可能来自于外部数据库。例如，一个系统可以将事务记录在数据库中，有可能作为一批事务进行记录。另一个系统就能够分阶段地搜索这些事务并进行处理。例如，对于一个费用记账系统，在一定时间间隔内的费用记录都被存储在数据库中。另一个单独的记账和收款系统可能在之后的某个时间再来处理这些交易。因为有来自于其他系统的输入，所以这些输入是来自于外部数据库还是在记账系统中被处理就是一个关于范围的问题。

来自于外部数据库的某些输入可能发生在其他输入的处理期间。例如，处理信用请求之前需要核实用户的信用记录和就业状况。在收到信用请求时，处理该请求的工作人员必须依靠外部数据库的信息来确认是否批准该项请求。因此在"处理信用请求"活动期间存在着来自外部数据库的系统输入。

系统界面的输出方面与其输入方面是成镜像关系的。在系统产生大量详细数据的时候会产生流向外部数据库的输出。许多系统输出是在极少人员干预的情况下产生的。如生成报告，通过电子邮件发送给收件人或者打印和分发。用户不需要与系统交互就能自动获得报告输出。账单、通知、声明、费用信函等都是依赖类似于报表生成的方法生成的，而不需用户干预。这些输出可能通过电子方式发送出去或者被打印出来。发送到外部系统的、将触发处理工作的消息也属于系统输出。

有时候，系统输入和输出必须是实时连续。RMO公司的实时信用卡确认系统就是一个例子。RMO公司的系统没有采取访问外部数据库的方法，而是建立与其他系统的实时连接来接收输入并提供输出。因此一项实时连接既是系统输入又是系统输出，很像一个系统与系统的对话。从这个角度讲，实时连接与用户界面功能是相似的，只不过实时连接使用对话来输入数据并查询系统中的数据（参见图14-1）。

使系统输入正确的另一种机制是，通过直接连到其他系统的接口来实现。电子数据交换（EDI）是一种减少用户输入的方法。所有的输入信息，如购货订单、发票、库存更新和支付情况，都通过将交易由一个系统发送到另一个系统来完成。使用EDI，不同组织的系统之间常常发生这样的数据传递，而同样的规则也适用于同一组织内部的系统。

EDI的主要挑战是确定交易的格式。对于单一类型的交易很容易设计其格式，但对来自许多不同系统的多种类型的交易则不容易设计其格式。当许多公司想在一起工作时，其困难性和复杂性可想而知。例如，通用汽车公司是最早应用EDI的公司，它有数千个供应商，当然带来数千笔交易，且每笔交易都有不同的交易格式。更复杂的是，每个供应商可能通过EDI同数十或上百个顾客相联系，而这些客户仍有可能通过EDI相连。所以即使一笔单一类型的交易也可能需要一打或更多的规定的交易格式。建立并维护一个EDI系统需要很高的代价也就不足为怪了。即使如此，EDI系统仍然比采用文件交易格式更有效且更有影响，因为后者必须被打印和重复输入。

最近几年，基于超文本标识语言（HTML）的系统之间的标准通信方法有了进展。我们知道，HTML在文本文件中嵌入了开始和结束标识代码（如格式化）以定义其文本或图像的特征。HTML在其自身文档内嵌入了格式信息。因此，一个能读HTML的程序在读一个文档时，可以用内嵌的标记语言，准确地对文档进行格式处理。从计算的角度看，这个系统虽然不是特别有效，但它简单易懂。

可扩展标识语言（XML）是一种新的系统对系统的界面语言，它正在受关注。XML是对HTML的扩展，它也在文字信息中也嵌入了自定义数据结构。因此，包含数据字段的事务可以以XML代码的方式发送，以定义数据字段的含义。许多较新的系统正使用这种技术提供一个系统到系统的公共接口。图14-2说明了一个可用来在系统之间传送客户信息的简单的XML事务。

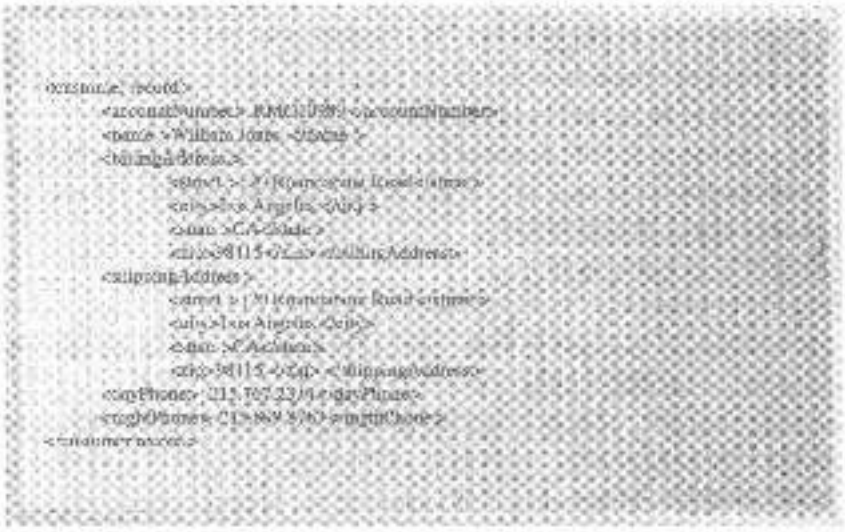


图14-2 基于XML的系统到系统的接口

XML和超文本标识语言（HTML）一样简单易懂。要想使XML正常工作，两个系统必须相互识别彼此的标识代码。建立一套完整的代码后，系统可以利用这些标识代码表示各种格式，这些格式能够被识别和处理。接收系统必须解析内部数据流并从标识代码中读取值。XML具有很好的可扩展性，不管有多少公司或交易类型，它都能应付自如。只要采用标准的标识代码，每个事务都可以有其自身的格式。

XML是在被称做文档类型定义文件（DTD）或称做XML Schema的独立的文件中定义的。许多产业和专门小组成立了标准委员会，用于定义标识代码集。普通业务、铁路零售业、新闻媒体和医药事务的标准代码已存在，目前，定义标准代码的小组有很多。

在第9章介绍了页面服务的概念，页面服务是基于XML的，所以业务交易可以通过互联网传送。实际上，XML的目的是充分利用互联网。

14.2 系统输入设计

当设计系统输入的时候，系统开发人员必须完成以下三个任务：

- 确定将要用做输入的设备和采用的机制；
- 确定所有的系统输入，并拟订一个包括所有数据内容的列表；
- 对于每个系统输入，确定哪些控制是必需的。

第一个任务，确定设备和机制，是对将信息输入到系统中的最新方法的回顾。在几乎所

有的业务系统中，一些输入被最终用户通过电子表单的方式来实现。在处处充满高科技的今天，向系统输入信息的方法有很多，其中包括各种类型的扫描、阅读和传输设备，这些设备更快、更有效，并且出错的可能性更小。

第二个任务，拟订必要输入的列表，提供应用软件设计与系统及用户接口设计的关联。正如前面所描述的，系统输入和接口设计必须集成到应用程序的设计当中，而这一步骤实现了此目的。

第三个任务是为正在开发的系统确定所需的控制点和安全级别。在对组成用户接口的电子表单进行详细设计之前，需要完成策略说明和控制要求。这些概念将在本章最后两节中阐述。

14.2.1　输入设备和机制

当分析师开始设计一个新系统的时候，他们通常假设所有的输入都可通过电子和图表的形式来实现，因为这些电子图表在今天的个人计算机和工作站上太普遍了。然而，伴随用户输入设计的开始，首要任务之一就是评价向系统输入信息的各种方法。

任何数据输入表单的主要目的是向系统输入新的、无差错的数据或无差错地更新数据信息。在这里，最重要的是不要出现差错。下面是有助于减少输入错误的几点经验：

- 获取的数据尽可能与原始数据接近。
- 尽可能使用电子设备和自动输入。
- 尽可能避免人工干涉。
- 如果信息可以从某个电子表单处得到，那么使用电子表单而不要重新输入这些信息。
- 在输入信息时，对数据进行检验和更正。

许多公司在设计系统时，在数据生成的地方获取数据。例如，原来销售人寿保险单的一个方法是让申请人和保险代理人填一个申请表，然后代理人将书面申请送到中心办公室并输入系统中。使用这种方法，可能会由于不能识别的书写、键盘输入错误、丢失字段等情况而发生许多错误。提交保险单的另一个方法是，代理人带上一个装有易于操作的电子表单的膝上型电脑，让申请人自己填写数据，或者从销售技术的观点来讲，更好的方法是让代理人输入数据而由申请人查看并确认信息是准确的、完全的。后者，申请人的信息是在申请人在场的时候被输入和确认的。事实上，甚至可以将一个便携的打印机连到膝上型电脑上，从而打印出完整的表单让申请人立即检查。这种方法明显降低了出错率，加速了整个业务过程和新保单的数据输入。

第二点和第三点经验，即数据自动输入、避免人工干预是紧密相关的，尽管使用电子设备不能自动地避免人工干预，但在许多事例当中，它们实际上就是同一个问题的两个不同方面。当开发者仔细考虑避免人工输入而使用电子输入媒体时，得到的系统就含有较少的电子输入表单，从而避免了一些常见的数据输入问题。数据输入的最常见的错误来源之一是，用户在输入字段和数字时出错。许多方法和设备可以不通过人工键入而获取数据。下面列出了几种普遍使用的输入设备：

- 磁卡片阅读器
- 条形码阅读器
- 光电字符识别阅读器和扫描仪
- 无线电频率识别标签
- 触摸屏设备
- 电子笔和书写板
- 数字化仪，如数字照相机和数字音频设备

实际，我们都见过一些电子输入设备。在杂货店里，使用电子扫描仪扫描通用产品代码（UPC）识别每一个货品并找到产品价格，自动称重机给产品称重并计算出价格；收银机读取你的信用卡，包括数量，客户和银行信息；新的自助结账系统几乎完全依靠自动化数据输入设备。

以前，具有法律约束力的契约要求具有书面合同和亲笔签名。而如今，新的法律法规允许数字化的合同和签名。现在使用信用卡进行采购时，采用数字签名来减少对书面凭证的需要。这种技术符合上述原则，信息以电子表单的形式作为数据源被获取，这种方式减少了许多错误来源。

下一个减少错误的原则是尽可能重复使用计算机中已有的信息。一些过时系统经常要求多次输入相同的信息，这不仅容易产生错误，而且会产生相同信息的多个副本，并且要求额外的控制和程序来确保各个副本是同步的。当发生错误时，很难知道哪一个副本是正确的。此外，当要求对数据进行修改时，必须对数据的所有副本同时进行修改。一个使用现有信息的例子是汽车出租系统，当出租一辆汽车时，要记录顾客、信用卡、汽车里程和油料信息。而当汽车被归还时，停车场的工作人员仅仅需要扫描合同的ID和输入返回时的汽车里程数和油料量，系统便可进行费用计算并为用户打印信用卡收据，且这些工作就在停车场完成。这种解决方案主要是为提供高级客户服务而设计的，它减少了许多数据输入的问题和错误。

通过不同的方法消除输入错误很关键，其中一个方法是可以用电子设备进行数据输入。但是，另一个潜在的问题是欺骗性信息的输入，对于欺骗性数据，有两个问题有待解决：存取访问控制和输入控制。首先，系统的访问权必须有所控制，只有经过授权的人和系统才能访问系统。今天，诸如指纹识别器、体温传感器、视网膜扫描仪等设备越来越多地被用于安全控制，其为传统的密码验证提供了额外的安全保障。其次，系统必须具有输入控制，以使欺骗性数据不能轻易进入系统。虽然完全消除潜在的欺骗性因素是不可能的，但精心设计输入控制将有助于消除风险。本章的后面部分会对安全设备和输入控制做进一步的讨论。

14.2.2 定义系统输入细节

这个任务的目标是确保设计者已确定了所有必要的系统输入并对它们进行正确说明。在前面的章节中，我们已经学过在分析活动和用例的基础上确定用户输入的多种方法。那些技术通过以用户为中心的设计方法来定义用户界面。本章的重点是通过分析在分析阶段建立的模型来确定系统输入，包括用户输入。至于分析和设计的其他方面，本任务也提供了一个定义检验以用户为中心的设计和分析模型中详细信息质量的机制。

分析师确定系统输入和用户输入的基本方法是确定穿越系统边界的所有信息流。结构化模型和面向对象模型的思想是一致的，但对每一种模型，其技术细节是不一样的。我们先从结构化模型开始。

1. 使用结构化的模型

在用结构化技术进行系统设计时，首要任务之一是定义自动化边界。从图10-2演变而来的图14-3是一个关于数据流图自动化边界的例子。以这个数据流图为基础的几项系统输入是：

- 时间卡信息
- 税率表的更新
- 雇员档案的更新

雇员文档的更新可能在图形化用户界面屏幕上完成。然而，时间卡信息和税率表的更新工作更可能是通过非图形的系统界面来完成的。时间卡信息常常是通过自动的雇员登记系统而收集到的。税率更新信息可能直接来自政府服务机构，并且可以以电子的形式提供。

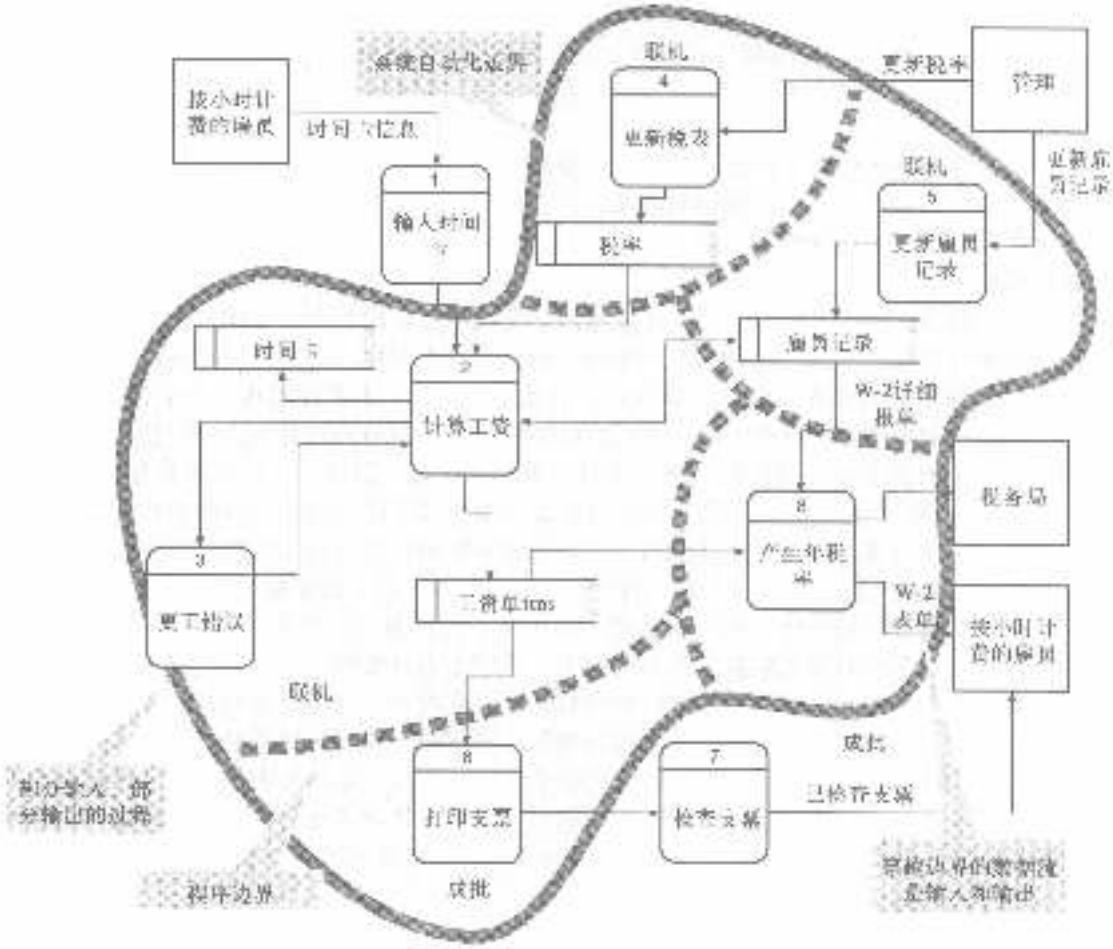


图14-3　系统级数据流图的自动化边界

尽管在高层数据流图上建立自动化边界并从那里找到输入是可能的，但是，从数据流图片段或更详细的数据流图开始往往工作效果更好。高层图通常不能提供足够的细节来识别许多数据流和系统输入。例如，图中的处理之一更正错误，被系统边界截断。因此对于解释哪些过程应该包括在自动化系统中，哪些数据流穿越了系统边界，就需要查看低层数据流图。

第6章解释了如何独立基于事件表中的事件的数据流图片段。对于更复杂的模型，可以通过查看每个数据流图片段并在每一片段上创建系统边界来定义系统输入。虽然具有自动化边界的高层数据流图给出了一个好的概述，但使用数据流图片段甚至每一片段的细化数据流图会更容易开展工作。

图14-4是带有垂直自动边界的RMO公司的“生成新订单”的细化数据流图。穿越边界的输入数据流被清晰地定义出来，需要的输入是新订单信息数据流和来自信用部门的实时链接。与信用部门的实时链接将是一个电子系统界面，用户界面的输入也将是新的订单信息。

设计者通过分析每个数据流图片段来确定必要的输入。数据流图上穿越边界的输入数据流对应于事件表中的外部事件的触发器。这个任务的结果是新系统高层输入的一个列表。图14-5是从数据流图片段得到的RMO客户支持系统的一个输入列表。为了得到这个列表，设计者需要分析图6-12中每个数据流图片段以及RMO系统中其他的数据流图片段。这个列表为新

有的系统输入和用户输入提供一个主控列表，但是，它并没有为设计输入提供足够的信息，所需的额外信息可从数据流定义、结构图和在第13章中讨论过的以用户为中心的设计活动（用户界面）获得。

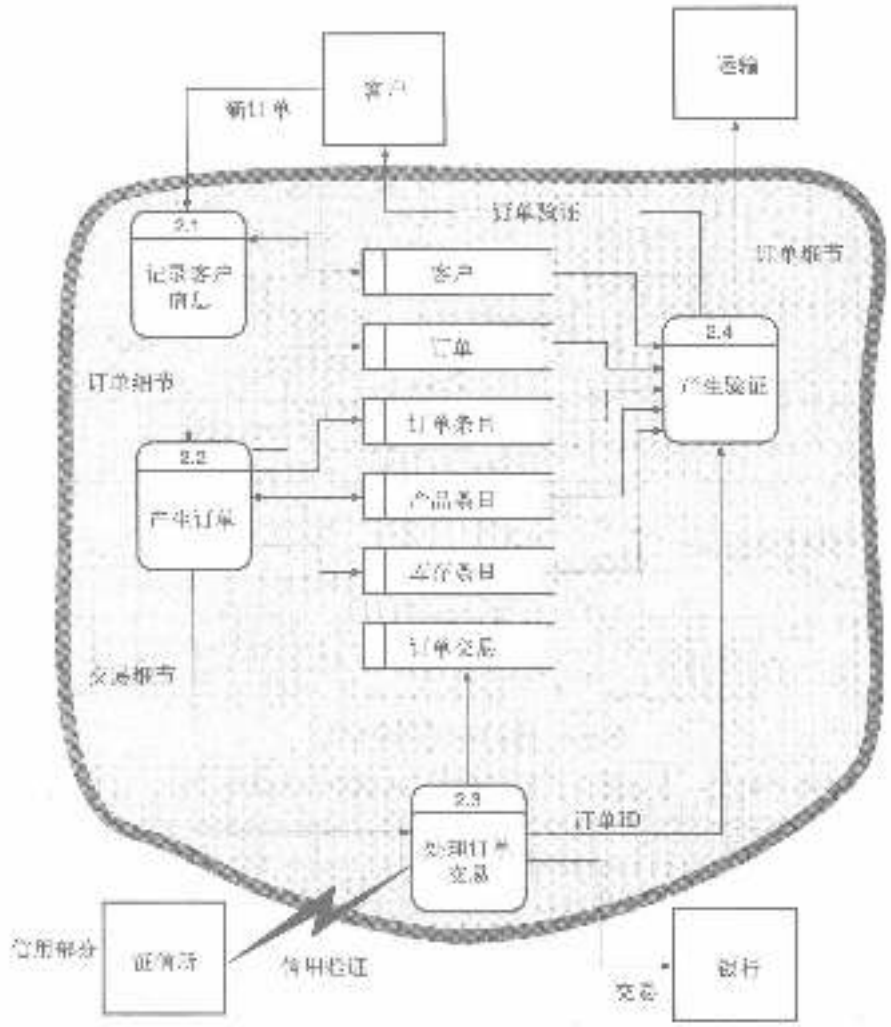


图14-4 与自动化边界的“创建新订单”数据流图

在设计结构时，设计者定义了各个程序模块以及和它们相关联的数据耦合。第10章讨论了每个数据耦合的详细数据内容的定义过程。数据流图上的每个输入数据流可以转化为结构图上的一个或多个物理输入。图14-6由图10-15演化而来，它定义了获得客户订购信息的输入模块，其中包括所订物品的详细资料。在这个图中，数据流图上新订单的数据流被扩展成结构图上4个单独的数据耦合。结构图定义了从外部系统获得数据的三个模块，这三个模块以及和它们相关的数据耦合是：“获得顾客信息”、“获得订购信息”和“获得信用卡信息”。换句话说，它需要三个模块来提供来自系统外部的关于顾客和新订单的所有信息。



图14-5 客户支持系统的输入列表

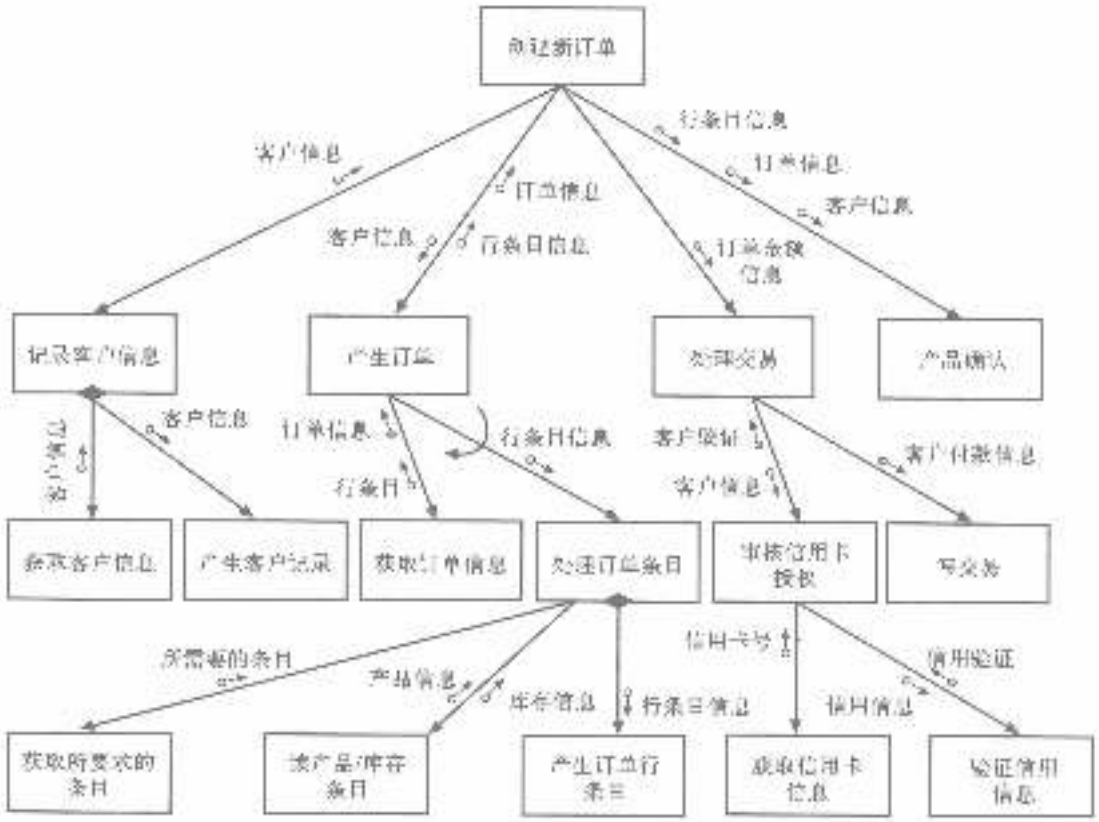


图14-6　创建新订单的结构图

下一步是分析每个模块，数据耦合，并列出每个数据耦合的数据字段。这个分析包括检查数据存储中的元素，从而确保数据存储中的所有元素建立在输入数据耦合的基础上。图14-7将图14-5扩展成既包含与数据流相联系的数据耦合，又包含与数据耦合相联系的数据字段。因为详细数据元素的确定是基于分析模型的，包括实体——关系数据模型。这就为第13章所阐述的方法提供了一个交叉确认，确保两种方法可以获得相同的数据元素是验证设计质量的有力手段。

图14-7的数据耦合列中所列出的由终端用户确定作为输入的每一项将成为电子输入数据表单或输入/输出表单的一部分。输入/输出表单允许用户输入空的键值并查询数据库以显示诸如产品描述的信息。这些与输入表单设计紧密联系的任务最好与应用程序结构设计任务结合起来。在仔细考虑人机交互后对它们进行进一步提炼。现在，让我们转向面向对象方法来看一下如何确定输入。

数据流	数据耦合	数据元素
新订单	客户信息	账号、姓名、订购地址、送货地址、白天电话、夜间电话
	订单信息	订购日期、优先号码
	行条款信息	产品ID、颜色、尺寸、数量
	信用卡号	信用卡号、终止日期
信用验证（实时）（鉴定）	信用信息（输出）	信用卡号、有效期、客户姓名、总额
	信用验证（输入）	接受/拒绝码、日期/时间、签字号、总额

图14-7　组成输入的数据流、数据耦合和数据元素

2. 使用面向对象模型

面向对象方法确定用户和系统输入的任务与利用结构化方法是一样的，不同之处在于，前者使用顺序图和设计类图，顺序图用于确定每条输入信息，而设计类图用于确定和描述输入参数。

图14-8是新金信息系统面向对象版本的一个简单顺序图（在图14-3中显示的是该系统的传统结构化模型版本）。在这个顺序图中，来自不同用例的摘录片段被组合起来，用以说明图14-3指出的主要输入。跨越系统边界的消息识别输入，既有系统输入又有用户界面输入。顺序图中这些输入的确定对第13章描述的以用户为中心设计方法所定义的GUI表单提供了交叉确认功能。跨越系统边界的三项系统输入是：

- 更新雇员信息——updateEmployee(empID, empInformation)
- 更新税率信息——updateTaxRate(taxTableID, rateID, rateInformation)
- 输入时间卡信息——inputTimeCard(empID, date, hours)

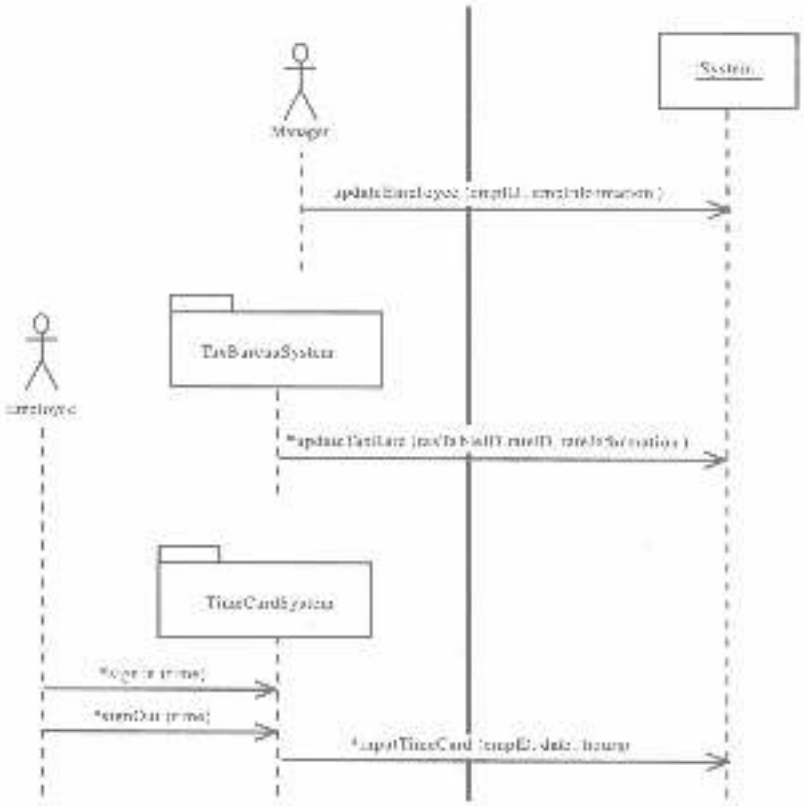


图14-8 新金信息系统用例的部分顺序图

实践指导

使用用例描述和系统顺序图定义需求时，着手考虑系统界面的输入和输出。 ■

第一个输入是GUI的一部分，在用户界面设计中已详细描述过。另外两个输入来自外部系统，并且不需要用户介入。来自税务局的信息或者以实时消息的形式发送，或者以CD或其他电子设备上的输入文件的形式发送。时间卡信息可以以各种格式进入系统，可以通过电子读卡器直接读取时间卡的方式，也可以通过子系统输入的方式。例如员工在下班打卡的时候，

时间信息便被录入。后面两个输入消息需要准确定义，包括传输方法、内容和格式。

图14-9是“生成新订单”用例的电话订购场景的系统顺序图的变体。原始顺序图参见图7-17。图中，订单职员和银行系统位于系统外部。订单职员到系统的消息属于用户界面，银行系统和系统之间的消息属于系统输入。在面向对象的模型中，参与者与带有内部对象的外部包之间的边界比结构化模型更清楚。在图14-9中，有4个从订单职员到系统的消息，一个从银行系统包进入系统边界的输入信息。这些消息以及其参数如下所示：

- 创建订单——startOrder(accountNo)
- 向订单中添加条目——addItemToOrder(catalogID, prodID, size, quantity)
- 完成订单——completeOrder()
- 执行支付——makePayment(paymentAmt, ccInformation)
- 返回确认——returnVerification(creditCard#, verificationCode, amount)

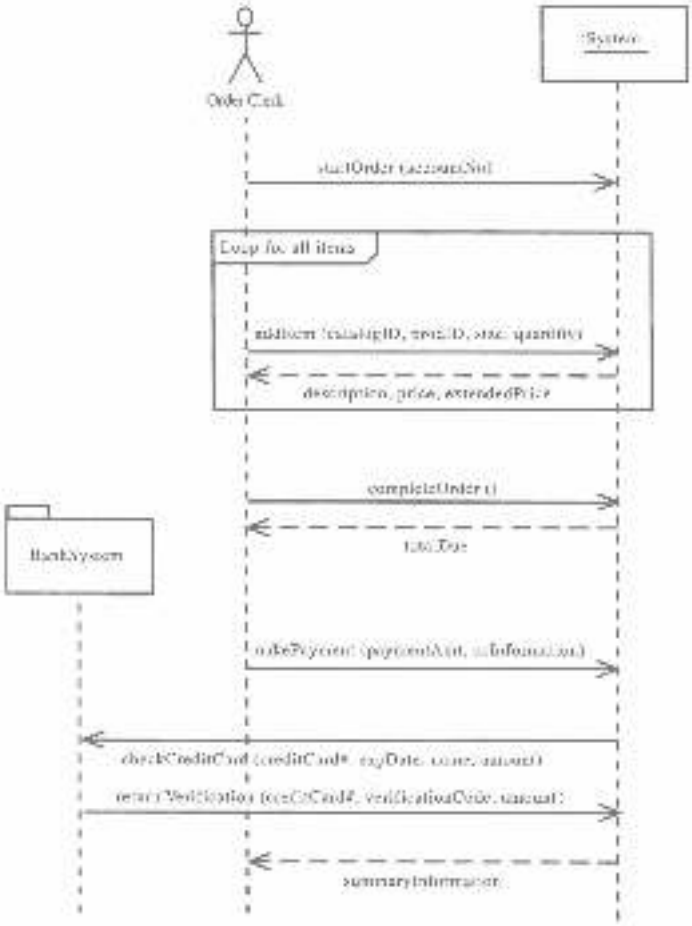


图14-9　创建新订单的系统顺序图

分析阶段的系统顺序图确定了从订购电话开始创建新订单的一系列步骤。在设计顺序图的同时，也设计了一个详细的对话框，用来协调用户和计算机的通信，这在第13章中做过说明。有一点需要指出，那就是顺序图提供了一个详细的用户和系统输入，这些输入为用例和相应的业务事件提供支持。

进一步分析消息可以获得该消息包含字段的一些信息。为了把消息分析得更加透彻，开

发者需要借助于类图。消息的参数必须与类图中的属性相一致，因为类的属性是有数据类型的，所以输入参数也应该有相应的数据类型。

图14-10是有关每个输入消息和消息所传送的数据字段的一个列表。在此例中，我们显示了同每个输入相联系的数据。同用户界面相联系的消息在设计用户界面时已被精确定义。在此案例中，只有系统输入被放置在列表中。这种分析不仅是确定系统输入详细信息所必需的，而且它也为分析提供了一个充分的检查。你会注意到，这个详细的表格有助于更精确地定义输入参数，这些详细信息常常会被回送到顺序图中，从而使得顺序图更充实。

Message	Data parameters
startOrder	accountNo
addItem	catalogID, prodID, size, quantity
completeOrder	—
makePayment	paymentAmt, creditCardt, expDate, name
returnVerification	creditCardt, verificationCode, amount

图14-10 RMO系统顺序图的输入消息和数据参数

14.3 系统输出设计

系统输出的主要目的是在正确的时间和地点，为正确的人提供相关信息。以往提供输出信息最普遍的方法是以打印文字的方式提供，尽管文本和表格仍然广泛使用，但诸如图表等新格式又为信息的表示、强调和归总提供了更多选择。

正如输入设计一样，输出设计的任务要求实现下列4个目标：

- 确定每个输出类型；
- 为应用设计所要求的特定输出制作一个列表；
- 提供必要的控制来保护输出的信息；
- 设计输出的布局并为其建立原型。

前两个任务的目的就是评价不同的输出信息候选方法，并设计出最合适的方法。在分析阶段，必要的输出报表通常作为系统需求建模的一部分被确定下来。输出设计的任务是：使输出与应用程序的模块（结构化技术）和方法（面对对象技术）相互协调。

第三个任务是评估信息的价值并为其提供保护。通常，组织机构对输入和系统访问实现了控制，但往往忽略输出报表也含有敏感信息。在14.4节中，我们将介绍几种所有输出都应包含的重要控制。

今天，用户可以通过使用工具和预先格式化的模板来设计他们自己的报表。这些报表，又称特定报表，并不是由程序员或分析师设计的，而是用户根据具体问题或需求查询数据所得到的结果。在第12章中，我们学习了关系数据库和结构化查询语言（SQL）的知识。许多系统提供一个简单的图形化工具，从而允许用户用SQL语言表示查询以生成特定报表。显然，在系统开发期间，设计师不可能设计出这样的报表。然而，系统中确实需要建立支持用户请求的工具和功能。报表设计活动是确定系统是否需要支持特定报表功能的好时机。如果需要，则增加该功能。

特定报表：程序员没有预先定义而在需要时由用户设计的报表。

14.3.1 定义系统输出的细节

定义系统输出细节的目的是确保设计者已经确认和定义了系统中每一个必要的输出。其

基本方法与系统输入的设计很相似。基于模型的方法利用事件表和其他模型来确定和定义输出的详细内容。尽管许多系统输入可能被定义为用户界面设计的一部分，但许多输出不需要动态的人机交互。例如，打印报表、返回文档或者简单的屏幕显示。因此，系统分析师必须依靠先前开发出的分析模型找出更多的系统输出。对于使用结构化技术的分析师来说，由从内部处理流向外部代理或外部处理的数据流来确定输出。对于面向对象技术，源自于内部类，而目的地是参与者或其他外部系统的消息，也是输出。

1. 使用结构化模型

确定输出的过程，实际上和从穿过系统边界的数据流建立输入列表清单的过程是一样的。在图14-3中的薪金信息系统的DFD包含了几个系统输出，这些输出一般不作为图形用户界面的组成部分。处理6——打印支票和处理8——生成年终税，含有三个在批处理执行中产生的输出。打印支票程序在特定的薪金发放期会打印出所有员工的薪金支票，类似地，年终税生成程序将打印W-2报表和一份详细报告。批处理报表也归属于系统输出类别之中。

在图14-4中，“创建新订单”的数据流图中有三个输出：客户确认信息、发货通知和发往银行的支付交易报表。和以前的任务一样，要建立一张数据流图输出表，确切地定义需要哪些报表并确定数据字段。为每个输出创建数据流，并在数据流图中记录这些数据流。图14-4中还有一项输出，它不是一个报表，而是发送给信用部门的要求验证信用卡的请求。

当系统分析师建立DFD输出、报表定义和数据字段的表格时，他们会在表格中增加两列内容。这新增的两列分别是：①产生报表所需的数据存储或文件；②产生报表的记录个数——是单条记录还是记录集。图14-11中的内容就是一个系统输出表格的例子。单条记录/记录集这一栏通常指明报表是在处理完成后立即打印出来还是在将来某个时刻成批打印。例如，来自图14-3的W-2报表将处理多个记录的报表打印。

为了证实结构图模块和输出报表的结构是一致的，分析人员再次查看数据耦合和报表数据需求。分析传送给模块的数据耦合和输出报表的数据字段，进一步检查生成报表的应用程序是否设计正确。

DFD数据流	结构图数据耦合	数据内容	必要的数据库文件	单条记录/记录集
订单确认	产生确认			单条记录
发货通知单	产生确认			单条记录
交易处理细节	书写处理			单条记录
信用验证（实时）	信用信息			单条记录

图14-11　基于传统结构化方法的系统输出表

2. 使用面向对象模型

在面向对象的顺序图中，由内部对象产生并发送给外部参与者的消息表示输出。图14-9中带有description、price和extendedPrice参数的消息便是一个输出消息。这个输出消息是由向订单（Order）对象添加条目（Item），即addItemToOrder（catelogID, procID, size）这个输入消息产生的。对于所有顺序图所生成的全部输出消息进行检查为分析阶段所确定的必要输出提供了一致性审核。

基于单个对象（或记录）的输出消息通常是对象方法的一部分。要为类的所有对象生成报表，需要借助于类方法。类方法属于类，而不是类的某个实例对象。例如，一张订单的客户确认信息是一个包括单个订单对象信息的输出消息。然而，要生成一周所有订单的汇总报表，类方法需要查看订单类的所有对象，然后输出只在本周内产生的订单对象的信息。

图14-9所示的顺序图中显示了四个输出消息：三个从系统到订单执行者，另一个从系统到银行系统包（BankSystem package）。每个消息都必须包含一列传送参数，尽管图中displaySummary ()消息没有显示出参数列表。输出设计阶段是分析师设计消息参数列表的好时机。与传统方法类似，图14-12是输出消息列表，需要的数据库文件或数据库表以及对象个数（单个对象或对象集合）也都包含在报表中。将其与图14-11进行比较，你可能会注意到，有些输出没有被列举出来。因为图14-12是由“创建新订单”单个用例派生出来的，所以表中未包含诸如发货消息等出现在其他用例中的其他消息。

输出消息	数据参数	数据库表	单记录/多记录
addItem()响应	Description，price，extendedPrice	CatalogProduct，ProductItem，Inventory，Order	单记录
CompleteOrder()响应	总付款额	订单，订单项	单记录
CheckCredit:Card()	信用卡号，预定日期，数额	顾客，订单交易	单记录
汇总信息——makePayment()响应	顾客名字，账单地址，货运地址，订单数目，日期，s&h，传真，总额	订单，订单项，顾客	单记录

图14-12 基于面向对象消息的系统输出表

14.3.2 设计报表、声明和返回文档

随着办公自动化和其他业务系统的出现，许多商务人士认为书面报表不再是必需的了。事实上，情况正好相反。业务信息系统使得信息的使用变得更加广泛，随之而来的是各种类型报表（书面的和电子的）的迅速增加。事实上，今天的信息系统面临的难题之一便是日益膨胀的信息，输出设计的一个最困难环节就是决定要提供哪些信息以及如何表示这些信息，从而避免产生大量令人困惑的复杂数据。

1. 报表类型

在学习分析师设计报表时所使用的不同格式之前，让我们先来讨论一下用户要求的四种输出报表类型：详细报表、汇总报表、异常报表和决策报表。

详细报表用来记录每天的业务处理过程。它们包含业务交易中的详细信息。有时，单个交易也许就需要一张报表，例如一张包含某一特定顾客订单细节信息的订单确认表格。有些详细报表也许列出了一组交易。这种报表的一个例子可以是所有过期账目的一个列表。这张报表的每一行包含有某一特定账单的信息。公司职员可以使用这张报表来查看哪些账目已经过期，并想好收回这些账目的措施。详细报表的目的是给公司人员提供工作文档。

详细报表：包含详细交易或记录的报表。

汇总报表通常用来对阶段性活动进行总结。这种报表的一个例子是对每天或每周的所有销售交易进行汇总，从而计算出销售总金额。中层管理人员通常使用这些报表来跟踪部门的业绩。**异常报表**也用于监视部门表现。只有当某个值超出其正常范围时，才生成异常报表；业务正常进行时，不会产生异常报表。当某些值超出期望值时，就生成异常报表向工作人员报警。这种报表的一个例子是生产线上用来列出不合格零件的报表。如果不合格率高于设定的阈值，那么就会生成一张报表。有时，异常报表定期生成，但只是着重说明其中超出范围的项目。因此，如果经常生产一些不合格零件，那么每天都可能会生成不合格零件报表。每

月把可收的门账列为过期账目。遗憾的是，总是有一些账目需要在这些报表中列出，因此这些报表总会定期生成。

汇总报表：对一段时间内或某些种类的信息细节进行摘要或汇总的报表。

异常报表：仅包含非标准或异常条件信息的报表。

组织中的高层管理人员通常使用决策报表来评估组织的整体健康状况和运行情况。同样，这些报表包括根据公司内部活动得到的汇总信息，也可能包括与全行业范围内平均水平业绩的比较。通过这种方法，管理人员可以评估自己公司的竞争力和薄弱环节。

决策报表：从通常用于战略决策的各种信息源得到的汇总报表。

在第1章中，我们讨论了系统分析师设计的各种类型的信息系统。某些类型的信息系统注重于生成特定类型的报表。尽管没有严格地要求每个系统只能生成一种报表，但是我们常常根据系统生成的报表类型对其进行分类。下一节将要介绍一些打印报表的例子。

2. 内部和外部输出

打印输出可以分为内部输出和外部输出。内部输出是为了组织或单位内部使用而生成的，我们在前面讨论的报表就属于这种类型。外部输出是为了组织外部人员的使用而生成的，它包括声明、通知及其他文档。生成这些报表需要使用高质量的图形和色彩，例如，银行每月的结算单、最新通知、订单确认信息、包装纸（如提供给RMO顾客的）以及保险单等法律文件。有一些外部文档被称为返回文档，因为提供给用户的文档中包含了可撕下来的一部分，而这部分用于系统以后的输入，例如，包含将与支票一起返回的付款存根的账单。上述的打印输出都必须仔细地设计。如今，高速的彩色激光打印机使得生成各种类型的报表和其他输出形式成为可能。

内部输出：为了组织内部的使用而生成的打印报表和文件。

外部输出：为了组织外部的使用而生成的打印文档，诸如声明、通知、套用信函和法律文档。

返回文档：一个外部输出，其中包含作为输入返回系统的一部分。

图14-13给出了一个外部输出的详细报表例子，这个报表来自与图13-20相类似的网上订单。好的用户界面设计要求，当一个顾客通过网络下订单时，系统可以把订单打印出来以便进一步确认。当然用户也可以通过浏览器的打印功能打印出网页，但这样做太浪费时间，因为这将打印网页上所有的图形和链接，只向用户显示与订单相关的、格式良好的确认信息，会显得更加友好。图14-13便是这样一个例子。该表是基于一个订单的，需要打印的数据记录包括顾客记录，订单记录和所订购的物品记录。格式化订单是为了便于阅读。不同模块的信息集中放在边界区域内。图14-13包含的信息非常全面，有当前的日期、订购的物品、付款明细和货运细节。

与网上购物订单不同，图14-14是一个内部输出的例子。与图14-13有几点不同：首先它是基于inventory数据库的所有记录的，而网上购物订单报表是基于单条记录的。这个报表还包括明细和汇总部分，有时也称控制变值报表。控制变值是一个可将报表分成多组的数据项。本例的控制变值数据项是产品号，即报表上的ID号。在输入记录过程中，任何时间遇到新的ID号，报告都将重新从新的控制值开始。详细部分列出了来自数据库的交易记录。汇总部分提供了总计信息和摘要信息。报表根据产品排序和显示。每个产品都列出了库存项，库存项显示了现有的库存数量。

控制变值报表：一个含有详细资料和汇总信息的报表。



Rocky Mountain Outfitters—Shopping Cart Order

Customer Name: Fred Meeting
Customer Number: 5747222

Order Number: 4673024
Today's Date: May 18, 2007

Shipping Address:

Billing Address:

Qty	Product ID	Description	Size	Color	Price	Extended Price
1	[illegible]	Jordan Men's Lumpman Team J	[illegible]	[illegible]	[illegible]	[illegible]
1	[illegible]	Wonderful Men's Lumpman Team J	[illegible]	[illegible]	[illegible]	[illegible]
1	[illegible]	Nike C.R.L – Fit Cart	[illegible]	[illegible]	[illegible]	[illegible]
1	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]

Shipping Information:

Payment Information:

Thank you for your order. It is a pleasure to serve you.
Check your mail mark for our weekly specials!

图14-13 RMO购物车订单报表

Rocky Mountain Outfitters — Products and Items

ID	Season	Category	Supplier	Unit Price	Special Price	Discontinued
RMO12557	Spring	Mens C	0201	$39.50	$34.96	No

Description: Outdoor Nylon Jacket with Lining

Size	Color	Style	Units in Stock	Reorder Level	Units on Order
Small	Blue		1500	150	
	Green		1500	150	
	Red		1500	150	
	Yellow		1500	150	
Medium	Blue		1500	150	
	Green		1500	150	
	Red		1500	150	
	Yellow		1500	150	
Large	Blue		1500	150	
	Green		1500	150	
	Red		1500	150	
	Yellow		1500	150	
Large	Blue		1500	150	
	Green		1500	150	
	Red		1500	150	
	Yellow		1500	150	

ID	Season	Category	Supplier	Unit Price	Special Price	Discontinued
RMO28187	All	Footwear	7555	$49.95	$41.88	No

Description: Hiking Walkers with Patterned Tread Durable Uppers

Size	Color	Style	Units in Stock	Reorder Level	Units on Order
7	Brown		1000	100	
	Tan		1000	100	
8	Brown		1000	100	
	Tan		1000	100	
9	Brown		1000	100	
	Tan		1000	100	
10	Brown		1000	100	
	Tan		1000	100	
11	Brown		1000	100	
	Tan		1000	100	
12	Brown		1000	100	
	Tan		1000	100	
13	Brown		1000	100	
	Tan		1000	100	

图14-14 RMO公司的库存报表

外部输出可以包括复杂的、多页的文档。众所周知的一个例子是，你收到的关于汽车保险结算的一套报表和声明。这个声明通常是一个多页文档，包括详细的汽车保险信息和费用、汇总页、返回的奖励支付卡和每辆汽车的保险卡。另一个例子是为每个雇员定制的多页信息性雇员福利报表。有时候，这些文档以带有特殊彩色显示部分或徽标的彩印形式来输出。图14-15是雇员福利计生存保护页表当中的一页，上面的文字都是一样的，只有数字会因人而异。

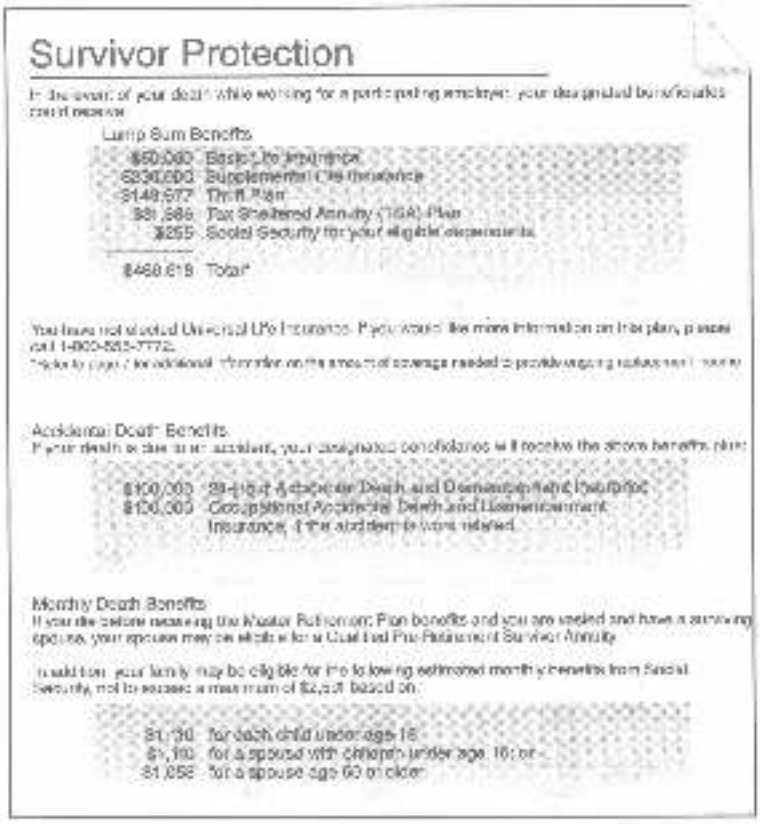
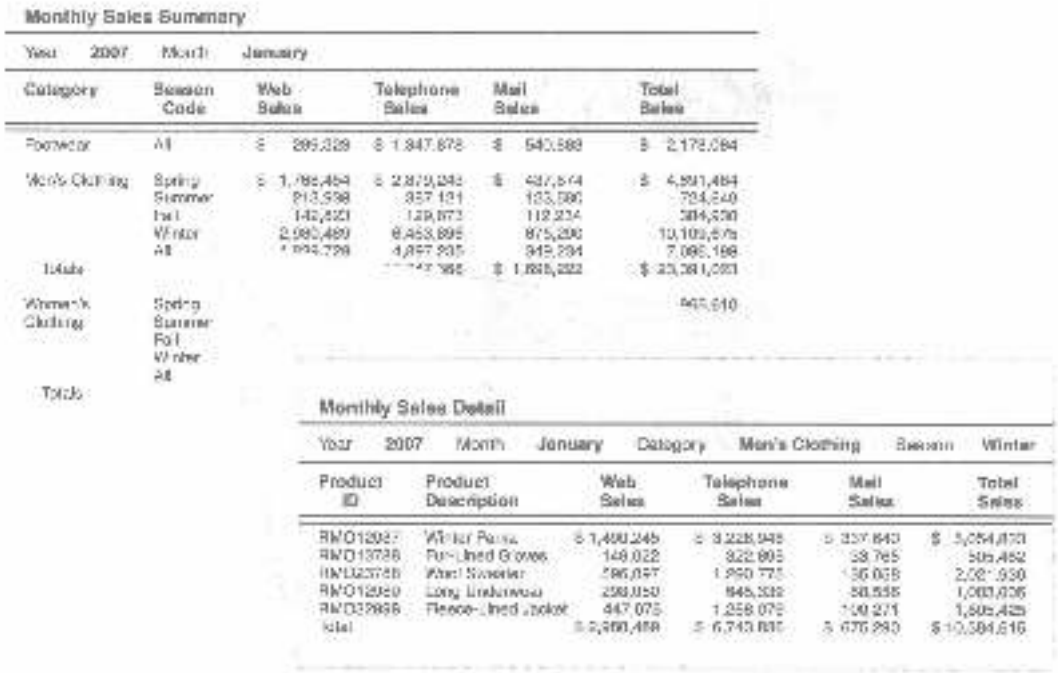


图14-15　雇员福利报表样例

3. 电子报表

信息系统可以使用各种类型的电子报表，每一种都服务于不同的目的，并且每一种都有各自的优点和缺点。电子报表在组织和显示信息上具有灵活性。在大多数实例中，模拟输出模拟打印报表的格式，只不过以电子形式显示，然而电子报表可以有多种形式来显示信息，例如，将详细和汇总部分一起显示，或将数据与图像一起显示，或以黑体和高亮形式显示，或动态改变数据组织结构及汇总部分，或包含链接相关信息的热链接。电子报表的一个重要优点是，它是动态的，它能满足用户在某些条件下的特殊需求。实际上，许多系统提供了特殊的报表能力，可以使用户悠闲地设计自己的报表。例如，电子表格可以链接到进一步的信息。所谓的下钻技术是指用户激活报表上的“热点链接”的能力，它告诉系统显示下一层、提供更详细的信息。例如，图14-16是一个订单产品库存的汇总评价报表。这个报表为每种产品系列提供了一个汇总评价。如果用户单击任何产品系列的热点链接，都会弹出带有库存项清单、当前库存数量以及每个库存项评价的详细报表。

下钻：将汇总字段与它所支持的详细资料连接起来，并使用户可以动态查看细节的能力。



Monthly Sales Summary

Year 2007 Month January

Category	Season Code	Web Sales	Telephone Sales	Mail Sales	Total Sales
Footwear	All	$ 295,329	$ 1,347,678	$ 540,589	$ 2,173,594
Men's Clothing	Spring	$ 1,786,464	$ 2,879,248	$ 427,514	$ 4,991,464
	Summer	913,598	957,151	133,590	724,549
	Fall	142,830	129,873	112,234	384,936
	Winter	2,980,469	6,453,895	675,200	10,109,564
	All	1,994,729	4,857,935	949,234	7,086,198
Totals			$ 15,278,102	$ 1,896,282	$ 23,381,083
Women's Clothing	Spring				945,610
	Summer				
	Fall				
	Winter				
	All				
Totals					

Monthly Sales Detail

Year 2007 Month January Category Men's Clothing Season Winter

Product ID	Product Description	Web Sales	Telephone Sales	Mail Sales	Total Sales
RMO12027	Winter Parka	$ 1,400,245	$ 3,228,948	$ 337,840	$ 5,054,833
RMO13726	Fur-Lined Gloves	148,022	922,895	38,765	509,482
RMO23758	Wool Sweater	286,097	1,950,775	136,038	2,087,930
RMO12959	Long Underwear	298,050	845,338	58,555	1,000,035
RMO32868	Fleece-Lined Jacket	847,075	1,258,079	100,271	1,505,425
Total		$ 2,980,469	$ 6,453,895	$ 675,200	$ 10,109,676

图14-16 可以下钻到明细报表的RMO汇总报表

另一种技术是链接，它使得用户可以将一个报表中的信息与另一个相关报表中的信息相互关联。这个概念，对于大多数经常浏览Internet网页的人来说都非常熟悉。在电子报表中，热点链接能提供一些相关和扩展的主要信息。同样的功能在业务报表上也非常有用。例如用来链接某个特定行业内主要公司的年度结算。

电子报表的另一个动态特性是可以从不同的角度来观察数据。例如，按照地区、销售经理、生产线、时间段来查看销售数据或者与上一个季度的数据进行比较，这些可能是很有益的。如果需要，电子表格可以生成不同的视图，而不是打印出所有报表。有时候，冗长的或复杂的报表会包括很多热点链接，这些热点链接可以链接到报表内容的不同部分。一些报表生成程序提供了电子报表能力，它包括所有Internet网页上提供的功能，包括框架、热点链接、图形甚至动画。

4. 图形和多媒体表示

数据的图形表示是信息时代最大的优势。允许数据以图形和图表的方式表示的工具，使得打印的和电子格式的信息报表更加友好。由于信息报表使商务人士能够预测趋势和变化，所以被越来越多地用于战略决策的制订。另外，如今的系统通常维护的是海量数据，远远超出了人们可以想象的程度。使用这些数据的唯一有效方法是对它们进行总结并以图形表格的形式表示出来。图14-17举例说明了柱状图和饼状图，它们是表示汇总数据最常用的两种方法。

近几年，随着多媒体技术的发展，多媒体输出也成为可能。如今，在屏幕上以图形和动画形式呈现信息并且突出部分带有声音描述已经成为可能。将视觉和声音输出结合起来是表示信息的一种强有力手段（当然，视频游戏将包含视觉、声音、触觉和嗅觉输出的虚拟现实技术推向前沿）。

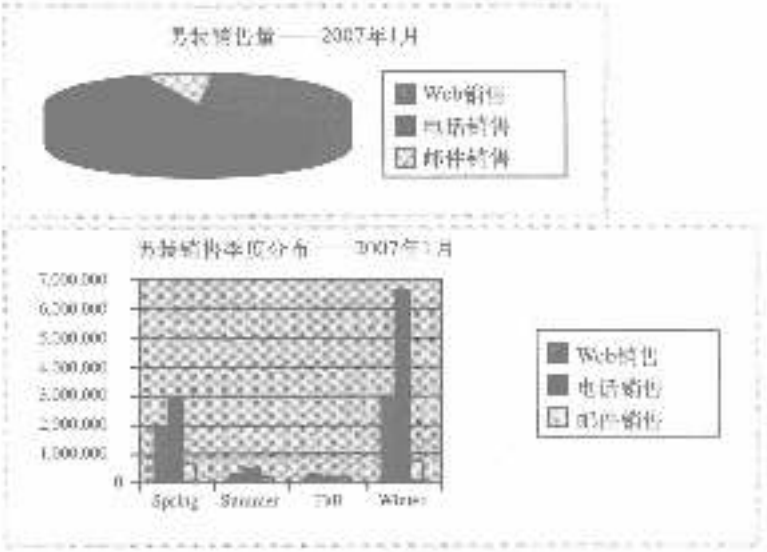


图14-17　柱状图和饼状图报表范例

随着系统输出设计的进行，评价各种表示方法是有益的。在系统中，设计报表能够提供一个全范围的报表定制方法。设计者应仔细分析每一个输出报表，确定输出的目标，选择最适合信息及其使用的报表格式。

前面的例子说明了RMO公司客户支持系统的一些报表。RMO系统中具有许多将数据呈现给有不同需求的用户的选择，在第4章中，RMO公司的组织图表显示，许多高层和中层经理对新系统很感兴趣。每个部门需要的信息不同，并且在大多数情况下，他们需要的数据格式也各不相同。在输出设计过程中，Barbara Halifax曾多次向她的小组强调这种灵活性的重要性（参见以下备忘录）。

2007年5月19日

To：John MacMurty

From：Barbara Halifax

Rf：客户支持系统的项目状况

John，按照你的要求，我写了这个备忘录，以便向你提交更新后的公司各部门所要求的信息报表。一项值得你关注的内容是，报表定义可能会揭示出一些我们未曾预料到的潜在信息需求，换句话说，存在着报表定义发生需求扩充这样的风险。

目前，我们基本上已经定义了所有的报表，我很高兴地向你报告，还没有发现哪个报表的定义引发数据库的结构改变或系统结构的改变。我们也接到几个新报表的需求以及相当数量的已有报表的急变新需求，例如，行销部门不仅要求打印报表，还要求在线报表并用带信息下钻链接，基础信息全部都在数据库中，所以我们已经能毫无困难地定义出报表。

你能在日程表的下一次指导委员会会议上说明一切均按计划进行吗？请代我向委员会各位成员表示感谢，感谢他们及时地向我们提出需求，使得项目进程能会预计日程并确保在无需定义的功能和数据结构拒绝户继续工作，我认为每个部门的信息需求均已满足。

BH

cc：Steven DeerField，Ming Lee

14.3.3 报表的规范化

目前存在多种输出格式可供选择，所以系统设计者在为用户提供输出时有更大的灵活性。选择一种格式并生成可用报表这一工作颇具挑战性，所以在设计输出报表的过程中必须牢记以下三个问题：

- 报表的目标是什么？
- 使用对象是谁？
- 媒体展示内容是什么？

在设计这些报表时，无怎么强调这些问题也不为过。在一些设计案例中，用户仅需要报表控制过程；而一些设计案例中，报表可能是决策中的关键因素。作为一个系统设计者，你必须事先确定报表的使用对象及这些用户如何使用报表。报表的内容和格式应该是基于报表的读者和使用者的。如果不考虑以上因素，你设计的报表可能会忽略掉关键因素或者会使设计的报表格式缺乏通用性。

设计者常常要确定报表的格式和细节级别。生成反映数据库中数据结构和格式的报表常常是很吸引人的。然而，经新的系统需要维护数据库中大量的详细资料。如果没有周密的考虑，设计人员很容易生成信息过载的报表。当提供的数据太多以至于用户很难发现并关注到重要的信息时，就发生了信息过载。许多人在网上搜索信息的时候也会出现这个问题。用户查找一条特殊信息的时候，搜索引擎会返回大量的查询结果。因此，需要仔细设计和表示输出，以防止类似事情的发生。

报表的格式也是很重要的，每个报表都应该有一个有意义的标题以说明报表的内容。像我们在前面讨论输出控制准确性时提到的一样，报表应该包含一个标题，其中既要包含报表生成日期又要包括基本信息的有效日期。有时这两个日期可能不同。报表还应该标记页数。在早期的系统中，当报表用连续的格式打印时，页码并不是很重要。然而在使用自动进纸打印机的今天，如果没有页码，用户很容易弄错各页的顺序，从而造成结果的误解。

标记和标题是为了确保正确地解释报表数据。图表应该附有坐标轴、度量单位和图例，以便标记得更清楚。如图14-14所示，注意报表中的标题和标记，以确保数据不被曲解。控制中断用于将数据分成易于引用的有意义片段，通过使用直线、黑体字和不同大小的字体，使得报表容易读懂。记住，任何报表的目的都是为用户提供有意义的信息，而不仅仅是数据，还要记住，要以一种易于阅读的格式来提供报表，报表设计就不是难事。

设计者常常假设报表只以标准的普通文件的形式打印出来，但是，这种假想是不正确的。正如我们所看到的，电子报表是展示输出信息的有效方法，而且电子报表的显示设备也越来越多，从标准计算机显示器到无线便携设备。设计者需要认真考虑，是否会有人使用非标准的设备访问这些输出，以及这些输出是否会通过有限的带宽传送。

14.4 完整性控制设计

信息系统控制构造于系统内部，用于保护系统及其内部信息的机制和程序。让我们描述几个场景示例来说明控制的必要性。

- 家具商店通过内部财务系统允许顾客使用信用账户购买商品。如果客户账号余额出现错误，那么我们如何保证只有经理一个人，即被授予修改余额权限的人，才能修改相应的错误呢？
- 一个可支付账目的雇员使用信息系统给供应商开支票，那么，信息系统如何保证支票的正确性以及将支票付给正确的供应商呢？系统如何确保没有人能通过给假供应商开具支票来进行欺骗活动呢？系统如何知道其一项付款是已被授权的呢？

- 许多公司拥有LAN或Intranet，那么，公司如何保护它的敏感性信息不被外部人员或恶意雇员所访问呢？
- 电子商务正呈现出指数级增长态势，并且许多公司已建成了电子商务站点。那么，一个公司如何保证其客户金融事务的安全可靠呢？一个公司如何保证它的系统和数据库不受互联网上黑客的侵入呢？
- 许多公司已经加入互联网并向销售为等外部雇员或客户和供应商提供在线访问机制。那么，公司如何保护它的信息系统不被病毒、蠕虫以及其他的恶意程序所攻击呢？

以上所有这些情形涉及了一般的业务和系统活动。因为信息是公司的最有价值的资产之一，所以新系统的开发者必须考虑如何保护和维持信息的完整性。如图14-18所示，许多地方需要利用安全方法和控制加以保护。有些控制必须被集成到所开发的应用程序和支持应用程序的数据库当中，其他控制来自于操作系统和网络。通常，集成在应用程序和数据库中的控制被称为完整性控制。来自于操作系统和网络的控制被称为安全性控制。本节将介绍完整性控制，后面一节将讨论安全性控制。

完整性控制：应用系统内部用来保护系统为信息的机制和程序。

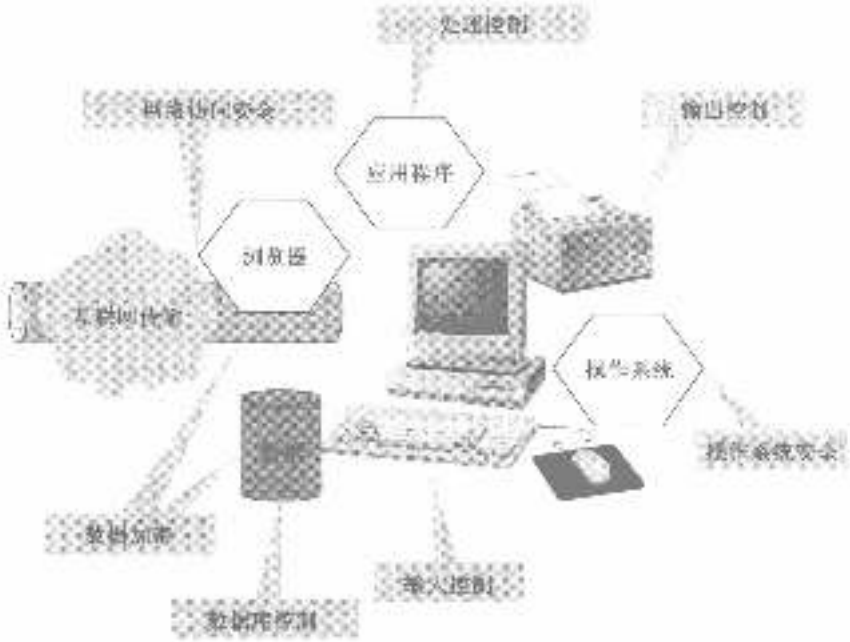


图14-18　安全和完整性控制点

通常在考虑完整性控制时，系统开发者往往将注意力集中在怎样避免应用程序系统出现问题和那些正要访问这些系统的雇员身上。这样，主要焦点是组织内部。完整性控制的主要目标是：

- 确保只发生适当并且正确的业务交易；
- 确保正确地记录和处理交易；
- 保护组织的资产（包括硬件、软件和信息）。

第一个目标是确保只发生适当并且正确的业务交易。其重点是识别和捕捉输入的交易。完整性控制必须确保包括所有重要的业务交易，即没有丢失交易，以及没有错误的和欺骗性的交易。

第二个目标是确保正确地记录和处理交易，它也关系到错误以和欺骗行为。控制需要进行检测，通知并警告用户数据输入错误以及在处理和记录数据过程中产生的系统错误。一个欺骗行为的例子就是，一个用户私自改变一笔有效交易的数额。

第三个目标是保护组织的资产，记录由于计算机冲突或系统崩溃丢失的信息。它也包括保护可能被恶意雇员或黑客破坏的计算机文件中的重要信息。

系统开发者常常太关注于软件设计，以至于忽略了开发必要的控制。因为计算机系统是如此的普遍，同时公司又是如此的依赖信息系统，所以没有特定完整性控制的开发项目极易招致破坏。系统可能会不断遭受错误、欺诈和欺骗行为，以至于最后无法使用。确保数据正确的主要控制点之一是数据输入点。

14.4.1 输入完整性控制

从共享的电子设备到标准的键盘输入，所有的输入机制中都使用了输入完整性控制。输入控制是用来减少输入错误数据的一种补充验证方法。例如，一个系统需要一定数量的有效输入数据，但一种输入设备不能保证输入所有必需的字段。出于这样的原因，有必要使用控制来检查输入的完整性。

老的计算机系统谚语“进来的是垃圾，出去的也是垃圾”讲的是输入控制，这里的目标是减少由于错误输入而带来系统的错误数据。在以前，为了保证正确的输入，最普遍使用的方法是将数据输入两次，这种技术称为验证。最早是为成批输入大量数据而开发的。一个人输入一个数据，第二个人再次输入这个数据，系统会验证这两个数据是否一致。如今，这种方法用得不是很多，因为审核数据需要许多高容量的处理。在线系统也会在数据输入时进行验证。下面给出一些现在广泛使用的控制技术。

- **字段组合控制** 检查所有字段的组合以保证输入的数据是正确的。例如，在保险公司的规则里，保险日期必须早于生效日期。
- **限值控制** 审核数字字段，确保输入数据是合理的。例如，销售额或提成的数目必须在某一具体范围内。
- **完全性控制** 确保所有必需的字段已输入。这种审核在输入时进行，所有控制所需的字段都要一一输入。例如，在一个保险表单中输入一个从属性，那么，这个人的生日也必须输入。
- **数据有效性控制** 确保包含代码的数字型字段是正确的。例如，银行账号必须是由7位数字和一个后缀校验位组成的8位号码。校验数字是在前面7位的基础上计算出来的。数据输入人员输入带有校验位的账号。系统对所输入的7位数字重新计算校验位。如果与输入的数据不匹配，那么说明输入有误。对于内部的表格和文件也需要进行在线验证。例如，当输入一个新订单时，会检验一下客户文件中的客户号。如果设计的系统可以从其他系统获取某些字段的值，这在一定程度上可以减少这种类型的控制。

字段组合控制：在一个多个字段数据的基础上对另一个字段的数据进行验证的完整性控制。

限值控制：验证一个字段的值过大或过小的完整性控制。

完全性控制：保证输入表单上所有必要的字段都被输入的完整性控制。

数据有效性控制：验证输入数据是否正确和合适的完整性控制。

14.4.2 数据库完整性控制

许多现代数据库管理系统包含完整性控制和作为附加控制层的安全性特征。在数据库级能够实现的5个主要的安全和控制领域是：

- 访问控制
- 数据加密
- 事务控制
- 更新控制
- 转储和恢复保护

1. 访问控制

访问控制是指用户获得数据访问的能力。操作系统一般以文件形式为基础应用安全和访问控制。数据库管理系统可以将这些控制应用到更细的等级。控制可被定义在诸如相互关联的一组表或对象、单个表或对象、或者单个字段或属性等模式的子集上。例如，不同的控制可以被应用到雇员表的姓名、社会保险号码、工资字段。同样，单个字段上也可能有读、写等不同访问控制。

访问控制：决定没有权访问系统及其数据的完整性控制。

数据库管理系统将安全控制访问信息存储在数据库模式中，并在每次读写数据时应用访问控制。当数据库管理系统执行安全控制时，数据库自动地为访问数据库的应用程序执行安全控制。有些数据库管理系统依赖于操作系统来识别正在试图访问数据的用户，这可使用户免受多次身份识别之苦；而有的数据库管理系统却独立于操作系统，实现独立的安全控制。

2. 加密

加密既可以用于数据库中的数据又可以用于数据传输，特别是借助公共线路进行的数据传输。数据库中的数据加密通常使用单键加密法。有关各种加密方法的内容将在14.5节中详细说明。

3. 事务控制

事务日志是记录数据库更新的技术。利用这项技术可以把对数据库所做的所有更新都以审计信息的方式记录下来，这些审计信息包括用户ID、日期、时间、输入数据和更新类型。其基本思想是建立一个数据库更新的审核跟踪记录，它能跟踪任何可能发生的错误和问题。在一些高级的数据库系统中，例如那些运行在服务器、工作站和主机上的数据库管理系统，都会将更新记录作为数据库管理软件的一部分。但是在一些小型的数据库管理系统中，特别是那些运行在个人计算机上的数据库管理系统，则不包括这项功能。所以设计小组必须将这些功能直接添加到应用程序中。

事务日志：对数据库的所有更新都予以记录的技术，记录信息包括谁、何时以及如何更新数据库等。

事务日志实现两个目标。首先，它可以防止欺骗性交易的发生。如果某人知道他所有的交易都会被记录下来，那么他就不会试图进行欺骗性交易。例如，如果一个人知道其ID号与每一次支付请求关联起来，那么，这个人就不可能请求一个伪造的支付。

事务日志的第二个目标是为错误的交易提供一个恢复机制。中等级别的记录系统维护所有更新设置，系统通过不应用这些错误记录而从错误中恢复过来。更复杂的系统不仅能够审核所有的更新记录，而且能够提供交易改变之前和之后的字段映像。这些复杂系统通常只应用在高度敏感和重要的数据文件中，但必要时，可以将它们作为一种重要的控制机制。

4. 更新控制

数据库管理系统支持多个应用程序的同时访问。这样，几个应用程序就有可能同时访问和更新一个记录或字段。数据库管理系统内部的更新控制提供了记录锁定机制来避免多个更新可能导致的冲突或互相改写。

另外，一些事务可能包含多个部分。例如，对于银行转账事务，它在增加一个账号存款

的同时，还需要减少另外一个账号的存款。延迟提交更新技术，即直到所有更新被确认才提交更新，可以保护数据不受复杂事务局部更改的影响。

5. 转储与恢复

转储与恢复程序可以保护数据库免于各种类型的灾难。许多数据库管理系统提供不同级别的转储和恢复。局部或增量式转储只记录两次完全转储期间的更新情况。完全转储定期地对所有数据进行存档，这些存档常常被放在远离数据库所在地且安全的地方，以避免受到诸如火灾、地震或恐怖攻击等灾难性的威胁。

对于时刻都在更新的系统而言，还有一个广泛使用的安全措施，便是数据库镜像或者回结镜像。这一技术转移整个数据库和发生的事务。显然，这一方法的代价相当昂贵。但当数据对组织的日常操作变得越来越重要的时候，它也变得越发重要。

14.4.3 输出完整性控制

系统的输出有多种形式，例如供其他系统使用的输出、打印报表和计算机屏幕显示数据输出。输出控制的目的是确保输出到达正确的目的地，并且这些输出是正确的、准确的、适用的和完全的。尤其重要的是，要保证具有敏感信息的报表可以到达正确的目的地并且不被未授权的用户访问。

1. 目标控制

在过去，大多数输出都是打印形式，分发控制台收集了所有在夜间处理的打印报表，并将它们分发给正确的部门和人员。这些控制台非常重要，因为一些报表含有敏感的、机密的信息，确保这些报表的安全是很重要的。一个控制良好的系统在打印报表的同时，还要在报表封面上打印目的地及路线信息。如今，企业通过给需要打印报表的地方配备打印机来实现与控制台相同的功能。打印出具有目的地和报头信息的封面不失为一个好主意。在设计的过程中，应该把目标代码和路由能力包括进去，便于对各个独立打印设备的报表分发进行处理。对这些报表访问的控制就成了一个物理访问。在设计过程中包含设计目的地代码和路由功能，它们被用来把不同的报表分配到不同的打印设备上。这种类型控制叫做目标控制。

目标控制：确保将输出信息输送到正确接收者的完整性控制。

提供给其他系统的电子输出通常有两种形式：在线的交易输出和具有成批数据输出的单个数据文件。每种形式都有自己的控制类型。如果系统提供在线交易，那么它必须确保每次交易都包括正确目的地的路由代码。两个系统需要一起工作来确保每笔交易都被正确地发送和接收。交易输出将包含确认码和校验位以允许接收系统验证交易的准确性。接收系统收到成功的交易后发出确认消息，许多这样的控制现在已经被加入到网络传输协议中。但在设计过程中，系统设计者需要注意网络和操作系统的能力，并在需要时予以补充以确保能够成功地接收数据。

输出数据文件控制会仔细识别文件的内容、版本、日期和时间。通常，系统在磁带或磁盘上生成一个数据文件，其他系统必须找到数据文件然后使用它。控制的主要问题是确保第二个系统使用正确的数据文件。例如，为了避免出现严重问题，我们需要确保星期五的交易不会运行两次，或者由于处理上的巧合，两个数据是同一天生成的，一个生成于上午，另一个在后半天。那么系统就必须使用这两个数据文件，如果第二个系统执行错误而需要重新执行，它必须能够重新找到执行时使用的正确文件。这种情况的控制文件通常含有两条比较特殊的记录，即开始记录和终止记录。这两条记录包含日期、时间、版本、记录数、控制金额总数和处理周期等具体信息。在系统设计过程中，必须预先考虑累计正确总量和产生必要的控制记录的问题。

计算机屏幕输出的目标控制的使用不像打印报表那样广泛。通常，计算机屏幕上信息的获得是由前面讨论过的用户访问权限控制的。然而，在有些情况下，目标控制对于哪些信息可以在什么终端上显示做出了限制。这对军用和其他安全领域的室内计算机终端是很有用的，并且对有权进入该范围的每个人提供了访问系统信息的权力。这些系统设计要求应用程序和网络安全控制系统之间紧密协调。

2. 完整性、准确性和正确性控制

输出信息的完整性、准确性和正确性主要是系统内部处理功能，而非任何一组控制。系统开发者要通过打印在输出报表上的控制手段确保其完整性和准确性。例如，每个报表都应有日期和时间标记，包括报表打印日期和数据有效日期。它们通常是相同的，但并不总是相同。尤其是当一个报表由于以前的错误而重新打印的时候更是如此。下列各项是应该被打印到报表中的控制：

- 报表打印的日期和时间
- 报表中数据的日期和时间
- 报表的覆盖周期
- 具有报表定义和描述的开始报头
- 目的地或路由信息
- 表格上的页数标记“第_页/共_页”
- 控制总数和交叉计算
- “报表结束”尾注
- 报表版本号和版本日期（例如，特殊打印的表格）

14.4.4　预防诈骗的完整性控制

前面几节介绍了几种完整性控制，以支持三种控制目标。许多技术都集中在预防错误和保护系统不受人侵上。然而，系统中具有合法使用权限的人对公司所进行的诈骗也是同样严重的问题。

诈骗问题在美国和全世界是很常见的。几乎每周，我们都可在报纸上看到诈骗事件和白领犯罪。由诈骗活动造成的经济损失是很惊人的，经济损失能达到上亿美元，甚至超过暴力犯罪和个人犯罪造成的损失。在过去几年中，因为主要负责人进行诈骗，已使一些大公司陷入亏损和破产境地。很显然，软件和系统控制不能完全消除诈骗。然而系统开发者可以利用系统发现出现诈骗的可能性，并采取控制措施去消除诈骗。我们在前面提到的控制方法，如输入控制、数据库控制和输出控制都是防止诈骗活动的关键组件。在设计系统时，可以考虑使用其他一些技术，以便更好地保护系统。

调查表明，在所有的诈骗案例中，有三种情况会导致诈骗活动：

- 个人的压力，如想维持奢侈的生活
- 经济问题，如某人的想法“我要还钱”
- 机会，如未验证的现金收入

完整性控制是指借助于足够的手工控制及对金钱财产的自动记录来减少和消除诈骗的可能。诈骗控制需要手工程序和计算机完整性控制两者共同完成，缺一不可。通过这种方法来减少诈骗的可能。系统开发人员需要同熟悉财会业务的用户紧密合作一起预防诈骗。

如果开发的系统不是一个财务系统，系统开发人员可能会认为不需要完整性控制。但诈骗的可能性存在于一切业务系统中。由于大多数业务系统能跟踪组织者的资产，一些人能控制这些资产并填写假额支票或伪造当事人。因此，几乎每个系统都需要某种类型的完整性控制。

图14-19包括了一些增加诈骗风险的主要因素。这个表不一定全面，但它为系统开发者提供了一个降低系统诈骗风险的框架。作为一个系统开发者，你应该同用户以及项目小组人员讨论以确保建立足够安全的完整性控制系统来降低诈骗风险。

造成诈骗风险的因素	降低风险的方法
职责分离	为口诗、批复以及消费的产生设计带有独立访问控制的独立电子表格
审计追踪不完善	包括事务日志 避免或者是严密控制绕过日志的手工覆盖能力
记录不完整	实施一个带有详细日志的综合数据库
监控不完善	用手工和自动程序来监控模式和禁止条件 包括异常报告 实施第三人审计功能
移动资产	包括一个易使用的图交叉接驳电物进计数器，带有自动记录功能
安全系统不完善	用附加程序和数据级安全措施来补充操作系统的安全特征 包括自动关机和锁定功能 包括分析访问模式的程序

图14-19 诈骗风险及预防技术

信息来源：表中信息由Brigham Young大学会计与信息系统学院的Marshall Romney博士提供。

我们已经对输入/输出的完整性控制进行了概述，现在可以将注意力转到安全性控制上来。

14.5 安全性控制设计

尽管前面提到过安全性控制的目的是保护机构的财产免遭任何威胁，但是其主要的焦点往往指向外部威胁。除了前面罗列的完整性控制的目的外，安全性控制还具有以下两个目的：

- 为用户和应用程序提供和维持一个稳定的、功能齐全的操作环境（通常是每天24小时，每周7天）；
- 在机构外部的传输过程中保护信息和事务数据。

安全性控制：通常由操作系统或环境提供的数据保护和恶意攻击防范机制。

第一个目的，即维持一个稳定的、功能齐全的操作环境。关注的焦点是一些能保护机构的系统免遭外部攻击的安全方法，例如黑客、病毒、蠕虫以及信息过载。今天的多数机构在它们的内部系统和互联网之间设有网关。每次机构人员与互联网进行通信时，都可能存在着影响系统安全的潜在威胁和一些会破坏机构的内部系统的不良访问。所以，减少和控制任何恶意访问有助于避免系统崩溃。

实践指导

与管理人员讨论时，随时都应该能回答"系统运行时，你能保证公司不会受到威胁吗"这个问题。 ■

第二个目的，即在机构外部的传输过程中保护信息和事务数据。关注的焦点是通过互联网发送和接收的信息。越来越多的机构利用互联网实现与客户和供应商之间的便捷联系。一旦一个事务从机构发送出去，它就有可能被中断、破坏或篡改。因此，安全性控制机制在信息从信息源到目的地的传输期间利用一些技巧来保护信息。

安全性控制可以在不同类型的软件中实现，包括网络和计算机操作系统、数据库管理系统或者各种应用程序。最常见的安全性控制点是网络和计算机操作系统，因为它们能够直接

控制系统资产，如文件、应用程序和磁盘。现代所有的操作系统都包含大量的安全特性，可以辨识用户、限制文件和程序访问、在分布式软件组件间安全传输数据。操作系统的安全性是大多数信息系统安全性的基础。

在某些场合，开发者可能直接在应用程序中实现安全性控制。当数据存储到文件中而不是数据库中时，开发者可能会基于单个数据项或记录定义他们自己的安全性控制。开发者还可以实现防止未授权用户执行某些特定操作的安全性控制，这些操作包括删除数据或者在可移动存储介质上建立转储拷贝等。

因为安全性功能的复杂性和重要性，大多数开发者避免在应用程序内部实现安全性控制。在大多数操作系统和数据库管理系统开发者队伍中，有一大部分人员专门负责开发和维护系统安全软件。对于应用程序开发者来说，他们很难投入充足的资源致力于实现准确且完整的安全性控制。这样，在一个典型信息系统开发项目中，与安全相关的实现任务经常被限制在或以操作系统或数据库管理系统的基础上配置一些安全软件。

14.5.1　系统访问安全

现代操作系统、网络软件以及Internet访问都需要有控制机制。这些机制可以实现由操作系统或网络管理的任何资源的访问控制，包括硬件、应用程序和数据文件。

系统访问控制是一种机制，它用来限制和控制用户能够使用计算机系统的那一部分资源。它包括对应用程序中某一部分或功能访问的限制，对计算机系统本身访问的限制，以及对某些数据访问的限制。

一个设计和实现良好的信息系统，可以利用嵌入系统的访问控制。这个方法的优点是，可以将一个一致性的访问控制集合应用到硬件平台或网络的所有资源上。因此，系统设计者可以实现一个单独的访问控制模式，并应用到每个资源或信息系统中。

系统设计者还可以在系统软件已有控制的基础上添加额外的控制。然而，设计并实现一个基于访问控制的有效应用程序，需要一定的专业技术。为了开发一个可靠且有效的访问控制，操作系统和网络软件开发者需要投入大量的精力和资源；而对于一个普通的机构来说，要投入同样的精力和资源，这是相当困难的，并且会得不偿失。由于这些原因，许多信息系统建立在系统软件已有的访问控制基础上。

1. 用户类型

系统开发者在设计访问控制时必须考虑不同类型的用户。图14-20演示了不同类型的用户及其对应的访问控制。以下内容将解释不同用户可用的访问控制。

在开发访问控制初期，设计者必须考虑这三类用户：未授权用户、注册用户和特权用户。未授权用户是指无权访问系统的用户，包括禁止访问系统的雇员，不再允许访问系统的以前的雇员以及黑客和入侵者等外部人员。访问控制系统必须有能力识别并拒绝这些人对系统的访问。

未授权用户：无权访问系统的用户。

注册用户是指经授权可以访问系统的用户。一般来说，根据能查看和更新内容的不同，可以建立各种级别的注册用户。在新系统的设计阶段，就应定义不同的访问级别。例如，一些用户允许查看数据但不允许更新数据，而其他用户则可以更新某些字段。对于其他级别的用户，新系统的一些功能和界面可能会被屏蔽。对系统设计者来说，最重要的一点是认识到可能具有多个级别的注册用户。授权是指决定一个用户是否被允许因某一特定目的而访问某一特定资源的一种处理。换句话说，授权就是决定用户是否应该是注册用户的。安全系统为每一个受保护资源保存着一个访问控制表。访问控制表是一个能访问资源的用户或用户组及

共访问类型列表。

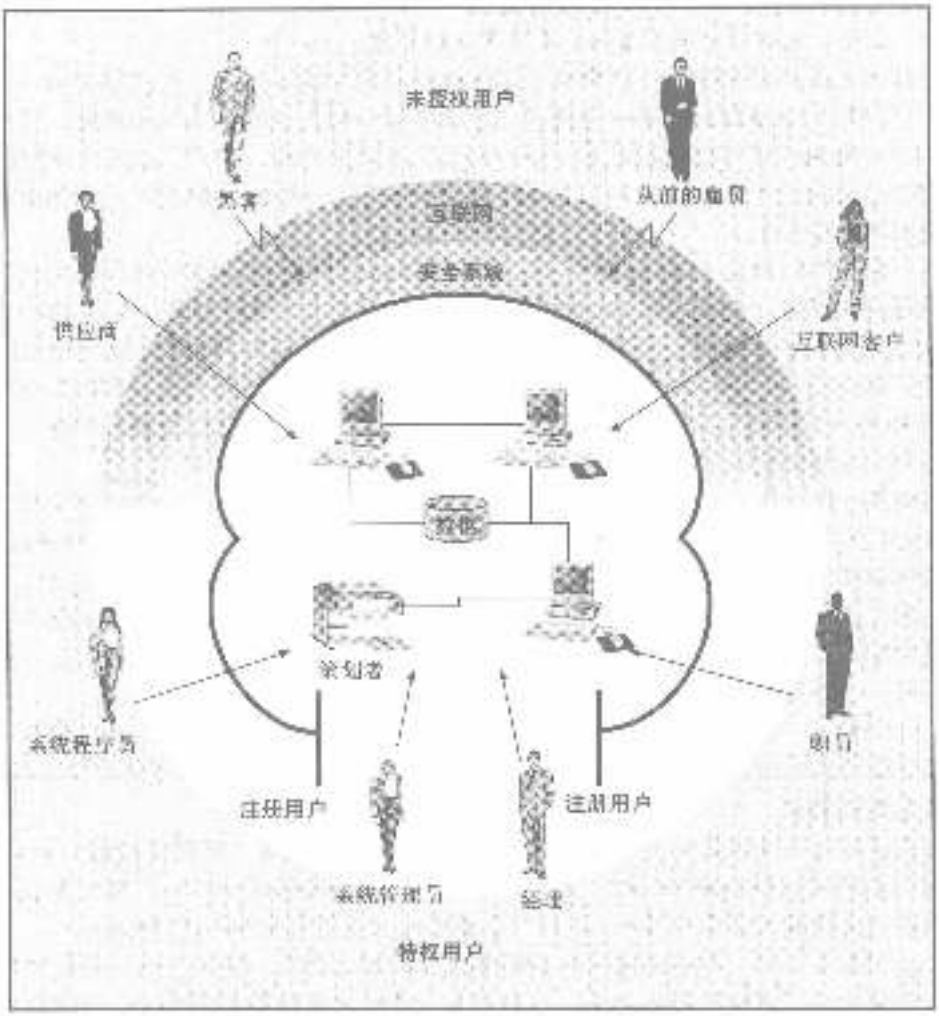


图14-20 计算机系统的用户和访问角色

注册用户：已注册或系统已知的并被授权可以访问系统某一部分的用户。

授权：决定一个用户是否被允许访问系统和数据的一种处理。

访问控制表：对系统和数据具有访问权限的用户列表。

特权用户包括能够访问源代码、可执行程序以及系统数据库结构的人员。这些人包括系统程序员、应用程序员、操作员和系统管理员，并且他们也有不同级别的访问权限。通常，系统程序员能够访问整个系统和全部数据。应用程序员只能访问应用程序本身，不能访问安全程序库和系统所用的数据文件。系统管理员能够访问所有的系统功能，并控制和建立各种级别的注册条例和注册用户。系统管理员通常还能利用软件程序来辅助访问控制并监视试图进行的访问。

特权用户：对系统有特殊安全访问权限的用户。

2. 密码与智能卡

鉴定是指识别用户（即授权用户或注册用户）的处理过程，这些用户需要访问系统敏感资源。鉴定是所有安全控制的基础，因为除非用户能被正确识别否则安全控制是无用的。在

许多操作系统中，鉴定过程需要用户键入用户名和密码。如果用户键入的密码与存储在安全数据库中的密码相匹配，则该用户通过了鉴定。

鉴定：识别，验证用户是否能访问系统的处理过程。

两种技术可用来定义密码。计算机可以随机地生成和指派密码，或者每个用户定义自己的密码。两种技术各有优势。第一种技术产生的密码一般较长并且具有随机性，但用户难于记忆。用户可能要用很长时间记忆像a3x7k69bis21这样的密码。用户自定义的密码比较容易记忆，但是这类密码不如前者复杂因此也不如前者安全。可以对密码的定义添加最小长度等限制来提高密码安全性。

当然，密码的问题之一是要记住密码。对巨型计算机来说有5～10个密码是常见的。虽然对所有的系统使用同一密码可减小问题复杂性，但一旦有人得知了密码，则所有的系统的安全性都会受影响。必须组织安全系统，从而使用同样唯一的标识符和密码就可以访问系统的所有资源。换句话说，只需一个用户ID和密码就能访问机构的不同系统。当用户不得不记住不同的ID和密码来访问不同的系统时，他们往往会将它们记下并贴在计算机附近，显然，这违背了用户安全验证的初衷。

智能卡是一种存储了安全信息的可机读的塑料卡片。这种卡片的信息可以用卡片扫描器读出，处理方式类似于在超市的收银台处刷信用卡。智能卡存储的是经加密的用户密码、指纹、视网膜或声音等信息。为了鉴定身份，用户扫描智能卡，接着输入密码或者提交指纹、视网膜或声音扫描，这样系统的安全性得到了增强，因为用户必须持有智能卡并且拥有适当的辨识信息才能通过鉴定。只有安全子系统知道密钥，这样可以防止潜在入侵者使用篡改过的智能卡。

智能卡：一种内部存储着安全信息的计算机可读的塑料卡片。

安全性的最后一步是确保系统记录每次试图对系统的登录，特别是不成功的登录。一次不成功的登录可能是用户忘记或输错了密码，但也可能是安全性攻击。这就需要进行调查。

3. 生物检测设备

身份验证也可以基于其他的个人特征，包括击键模式、指纹、视网膜扫描和语音特征等。当用户键入密码或其他击键序列时，每次击键的耗时和力度都是唯一的。有些安全系统既使用密码又使用击键模式来鉴别用户，这样可以防止他人滥用密码访问系统资源。

许多公司都在实验一种新的基于生物检测设备的安全形式。这种使用生物检测设备的关键是把人作为安全系统的密码或网关。这种更复杂的安全系统能扫描指纹、视网膜血管、声音。这些特征对每个人来说都是唯一的。随着具有高内存和逻辑电路的微型计算机芯片的出现，生物检测设备几乎可以嵌入到任何普通的计算机硬件中。与此同时，在这些微小的生物设备中，存放着完成指纹、手部静脉模式、视网膜、虹膜模式或完整的面部模式等复杂的模式匹配所必需的复杂逻辑。

生物指纹检测设备已被嵌入到鼠标、计算机键盘、小触摸屏等计算机部件中。其他一些生物扫描仪，如小型照相机，可以嵌入到计算机监视器中。当人注视监视器时，这些的设备可进行虹膜或面部扫描。图14-21所示的是一个带有嵌入式触摸屏的鼠标，可以用来检测指纹。还有一些鼠标在侧面安装了传感器，在使用过程中，拇指必须放在传感器上面，这样用户的身份才能通过鉴定。



图14-21　生物鼠标

资料来源：感谢萨斯卡彻温省政府的eyeDentify公司

基于生物检测设备的安全性还可以是多级的。在用户首次登录时，可以执行安全性检验。也可以在后面一个给定的程序执行一些较高层次的安全性检验，以获得一些附加的权限来访问某些特定的窗体或数据库记录。显然，每个个体必须被授权或与存储在所允许的安全级别中的信息相一致。

14.5.2 数据安全

除了需要控制对机构信息系统和内部网络的访问，数据本身的安全性控制也很重要。例如，用户ID和密码是必须保密的重要信息。通常，密码信息甚至要对系统管理员保密。系统管理员可以为用户指派新密码，但是他们不能读取或访问用户的当前密码。因此，如果一个用户忘记了密码，那么管理员可以为用户设置一个新密码。

许多其他类型的文件也要求保密，如：

- 金融信息
- 信用卡号、银行账号、工资信息以及其他个人数据
- 产品策划和规划信息以及其他关键任务的重要信息
- 政府和敏感的军事信息

有一些操作系统，特别是UNIX操作系统及其派生系统，其内部只有文件安全控制。每一个UNIX文件针对三类用户有不同的访问机制，这三类用户分别为：文件所有者、文件所有者的同组人员、其他用户。每个用户的访问级别又进一步分为三个等级：读访问、更新（创建、更新和删除）访问和执行访问。执行访问决定了文件是否是可执行的（例如，Windows中的.exe文件），安全等级决定了谁被允许执行该文件。

位于系统内部的数据需要保护，但是在机构外部传输的数据也可能遭到窃听和修改。随着电子通信的日益普及，越来越多的机构借助互联网传送和接收事务处理信息。在业务批发销售方面，客户浏览产品目录、订购产品、支付费用以及发货跟踪都是通过互联网进行的。在供给方面，机构通过互联网订购商品、查收、发送购货订单、开进行金融交易等活动。因为借助公共互联网传输这些信息，所以任何人使用工具监听和截取数据包后都能获得这些原始信息。

无论系统内还是传输过程中的数据，保证数据安全的主要方法都是数据加密。加密是将数据改变，使得未授权用户不能读取数据的一种处理。解密是将加密过的数据转换回初始状态的一种处理。存储在文件或数据库中的数据可以通过加密而保证它们不被窃取，通过网络发送的数据可以通过加密而防止其在传输过程中被偷听和窃取。获取加密数据包的行窃者或窃听者得到的是无意义的位串信息，这些信息很难或不可能转换回初始数据。

加密：改变数据，使得未授权用户不可读取数据的一种处理。

解密：将加密过的数据转换回可读格式的一种处理。

加密算法是一种用于数据加密或解密的复杂数学公式和处理方法。密钥是加密算法的一项二进制输入——通常是一个很长的二进制位串。加密算法基于加密密钥变换数据。加密数据又可以通过同一密钥或对应的解密密钥变换回初始状态。现有许多加密算法，包括数据加密标准（DES）和RSA Security所开发的几种加密算法，已经作为政府级或互联网级标准而获得广泛应用。加密算法必须保证所生成的加密数据若无密钥将很难或不可能被解密，而且随着密钥长度的增加，无密钥的解密过程难度也会增大。发送者和接收者必须使用相同或兼容算法。

加密算法：一种用于数据加密或解密的复杂数学公式和处理方法。

密钥：加密算法用于转换数据的二进制序列。

图14-22是一个对称密钥加密的例子，这里使用同一个密钥进行数据加密和解密。对称密钥加密涉及的一个重要问题是，发送者和接收者都使用同一个密钥，该密钥必须以一种安全性的方式被创建和共享。如果密钥和加密数据一起通过同一传输渠道传送的话，信息安全性就会大打折扣。还有，与多个用户共享同一密钥也可能增加密钥被窃取的可能性。

对称密钥加密：使用相同密钥进行数据加密和解密的加密处理。



图14-22　对称密钥加密

非对称密钥加密使用不同的但具有某种性的两个密钥来加密和解密数据，公共密钥加密是一种使用公共密钥进行加密而使用私有密钥进行解密的非对称加密形式。两个密钥是一一对应的，信息一旦被公共密钥加密，它只能通过私有密钥来解密，不能用加密密钥来解密。使用这种技术的机构往往广播其公共密钥，因此任何人都可以自由地获得该公共密钥。当某个实体——例如，从卖主手里订购产品的某个人——想要传送一组安全消息给卖主，客户就可能要将消息利用公共密钥进行加密处理再将加密消息发送给卖主。卖主接到消息后就会利用私有密钥对其进行解密。由于其他人没有私有密钥，所以其他任何人都不能解密该消息。

非对称密钥加密：使用一个密钥进行数据加密，而使用另一个不同的密钥进行数据解密的处理。

公共密钥加密：具有一个公开的密钥和一个保密的密钥的一种非对称密钥加密形式。

有些非对称加密方法能够进行双向加密和解密，即除了利用公共密钥对消息进行加密并利用私有密钥解密外，也能利用私有密钥进行加密并且使用公共密钥进行解密。注意，不但两个密钥仍然成对出现进行工作，而且消息可以通过加密/解密操作组合实现变换和还原。第二种技术是数字签名和数字证书的基础，其内容将在下一节阐述。图14-23演示了利用非对称加密方法进行加密的数据传输过程。

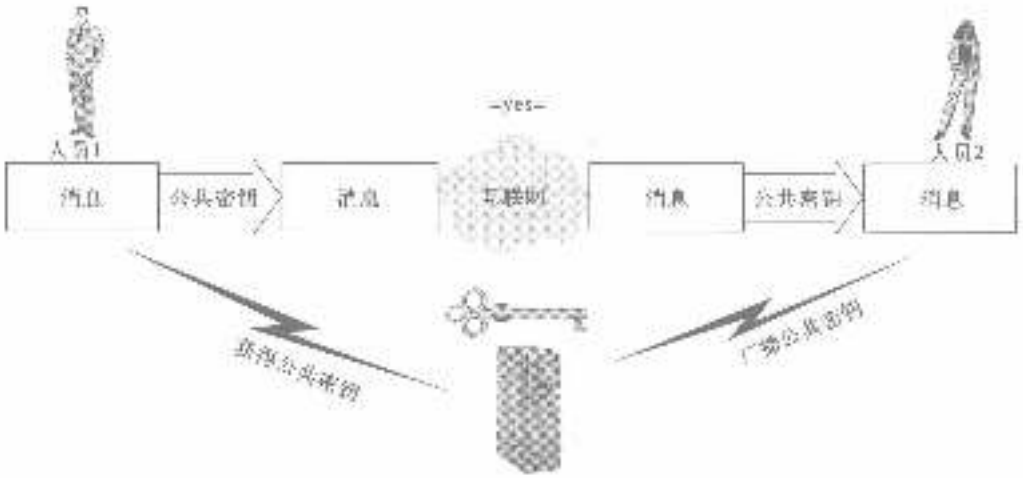


图14-23　非对称密钥加密

你可能会问："加密算法如何能做到单向处理（利用一个密钥）而不能按同一方式还原（利用同一密钥进行解密）呢？"这类算法的数学推导已经超出本书内容。然而，下面这个例子应该不难理解：乘法和因式分解。如果一个人给你两、三个数，甚至是很大的数，并要求

你将它们相乘，你能相当容易地完成任务。但是，如果一个人给你一个相当大的数并要求你将其因式分解（即找到乘积等于这个数字的原来的那些乘数），这就不太容易了，这项任务可能会花费你很多时间。基于这种单向数学特征的算法奠定了非对称密钥加密的基础。

14.5.3 数字签名和数字证书

信息加密是一种保证拥有密钥的两个实体安全地进行信息交换的有效技术。但是，你怎么才能确保通信另一端的实体正是你所期望的对象呢？数字签名是一项利用私有密钥对文档进行加密从而验证文档归属的技术。如果你拥有某个实体的公共密钥，并且该实体发给你一条已利用其私有密钥加密过的消息，你可以使用公共密钥进行解密，你可以确信与你通信的实体正是所期望的对象，因为只有掌握私有密钥的唯一实体才能对信息进行加密。利用私有密钥进行消息加密被称为数字签名。

数字签名：一项利用私有密钥对文档进行加密来验证文档归属的技术。

进一步分析举例，你可能会提出疑问："我怎么才能知道我所拥有的公共密钥是正确的公共密钥而不是假冒伪造的密钥呢？"换句话说，可能某个人正在假扮另一个人并传递假的公共密钥以便截获加密消息（如金融交易信息）并窃取信息。本质问题是要确保声称为某机构公共密钥的那个密钥的确是该机构的公共密钥。解决该问题的方法是证书。

证书或者数字证书，是经过第三方机构认证和加密过的机构的名称及其公共密钥（外加单位地址、网站URL、证书有效日期等其他信息）。许多第三方机构非常著名并作为鉴定权威机构而被广泛接受，如Verisign或Equifax等。事实上，这些机构非常出名以至于它们的公共密钥已经内置于Netscape和Internet Explorer等网络浏览器中。如图14-24所示，你能够知道正与你通信的实体身份是否属实，并且你确实拥有其正确的公共密钥。

证书或数字证书：被权威验证机构加密并用于广播机构名称及其公共密钥的文本消息。

想获得名称和公共密钥证书的实体要去认证机构购买证书。认证机构利用其自己的私有密钥对数据进行加密（标记数据）并且将数据返还给该实体。现在当某个人，例如一个客户，向该实体索要其公共密钥，该实体则向客户发送其证书。客户接收到证书并利用认证机关的公共密钥进行解密。此外，认证机构众所周知，所以它的公共密钥内置于所有用户的浏览器中并且不可能被假冒伪造。现在客户可以确认其与之通信的实体身份并利用该实体的公共密钥进行数据加密和传输。



图14-24 使用数字证书

这一场景的变体就是买方和卖方相互传送证书。每个参与者可以使用认证机构的公共密

钥对证书进行解密并从中摘录名称和地址信息。但是，为确保包含在证书中的公共密钥的有效性，证书要被传送到认证机构去进行验证。认证机构将包括公共密钥在内的认证数据存储在数据库中，并且通过与数据库内容的匹配来验证传送来的证书。

14.5.4　安全交易

安全的电子交易需要一套标准方法和协议进行鉴定，认证和实现数据的保密性与完整性。Netscape最早开发出安全套接层（SSL）协议来支持安全交易。尽管SSL协议后来作为互联网标准而被重命名为传输层安全协议（TLS），但是原先的名称SSL一直被广泛使用着。

安全套接层（SSL）：实现连接并传输加密数据的一种标准协议。

传输层安全协议（TLS）：SSL的更新版本。

TLS是保证在Internet中实现传送数据安全通道的一项协议。发送者和接收者首先利用常用的Internet协议建立连接，接着通过互相询问建立起一个TLS连接。接下来双方相互确认身份，主要是通过交换和验证前面已解释过的身份证书实现。此时，双方已经交换过公共密钥，接下来就可以进行安全的信息传递了。因为非对称加密过程困难并且效率很低，所以两个通信实体要共同遵循同一协议和加密方法，通常使用单密钥加密方法。当然，用于建立安全连接的所有数据都是通过公共密钥/私有密钥组合发送的。一旦加密技术被确定下来并且保密性单密钥已经传送，那么接下来的数据传输过程就要利用这一保密性单密钥。

IP安全性（IPSec）是保证信息安全传输的较新的Internet标准。IPSec是在网络协议栈的较低层次上实现的，因而可获得更快的操作速度。IPSec可以替代或补充TLS。这两种协议可以同时使用从而提供一种特别的安全方法。IPSec可以支持比TLS更安全的加密方法，但是这些方法还没有完全应用于Internet。

安全超文本传输协议（HTTPS或HTTP-S）是一项用于安全传输网页的互联网标准。HTTPS支持几种类型的加密方法、数字签名以及证书的交换和验证。所有的现代网络浏览器和服务器都支持HTTPS。它是基于网络的完备的安全方法。如果通过TLS或IPSec安全通道发送HTTPS文档可以加强其安全性。

安全超文本传输协议（HTTPS或HTTP-S）：用于安全传输网页的互联网标准。

安全性是现代基于网络的信息系统开发和部署过程中需要考虑的一个重要部分。幸运的是，许多成熟的工具和程序可以作为整体解决方案的一个部分而集成到新系统中。系统开发者需要具备安全意识，了解安全方法并且熟悉最新的安全工具和安全技术。

小结

本章首先讨论系统界面的确定，接着讨论了系统界面的设计。系统界面包括除去图形用户界面（GUI）中输入输出之外的所有输入和输出。

设计系统输入包括一个步骤：

- 确定设备和用于输出的机制；
- 确定所有的系统输入并开发出每一输入数据的内容列表；
- 决定每一个系统输入需要的控制。

为了开发系统输入列表，系统设计者使用在分析和应用程序设计活动中得到的图表。对于传统结构化方法，要使用DFD、数据流定义和结构图。对于面向对象设计方法，顺序图是主要的信息来源，同时借助设计类图，确保方法和产生输出的方法是完整的。

系统输出的设计过程与系统输入的设计过程大致相同，对于输出设计，可以利用DFD和顺序图确定流出系统的数据流和消息。新的技术提供了许多用图表、图形和多媒体等形式来

表示输出的方法。在决定使用哪种输出媒体之前，设计者应该认真考虑系统的使用对象和输出的目的。

接着本章又讨论了系统完整性控制。完整性控制的目标是：

- 确保只发生适当的和正确的业务交易；
- 确保正确地记录和处理交易；
- 保护组织的资产（包括信息）。

完整性控制的内容涉及了谁有权访问系统和数据库的各个组成部分。访问控制机制确定不同用户类型（无授权的用户，注册用户和特权用户）来保证系统的安全。附加的完整性控制则重于减少错误，防止诈骗和维护系统数据的正确性。

本章最后一节介绍了包含公共网络访问（主要是Internet）的系统安全基本概念。安全问题越来越重要，因此在开发新的信息系统时要考虑不同的技术。众多安全方法的基础是包含公共密钥和私有密钥的公共密钥系统。加密与公共密钥系统是数字签名、数字证书、安全连接和安全交易的技术基础。

关键术语

access control	访问控制
access control list	访问控制表
ad hoc reports	特定报表
asymmetric key encryption	非对称密钥加密
authentication	鉴定
authorization	授权
certificate, or digital certificate	证书或数字证书
certifying authority	验证权限
completeness control	完全控制
control break report	控制中断报表
data validation control	数据有效性控制
decryption	解密
destination controls	目标控制
detailed report	详细报表
digital signature	数字签名
drill down	下钻
encryption	加密
encryption algorithm	加密算法
encryption key	加密密钥
exception report	异常报表
executive report	决策报表
external output	外部输出
field combination control	字段组合控制
integrity control	完整性控制
internal output	内部输出
privileged user	特权用户
public key encryption	公共密钥加密

registered user	注册用户
Secure Hypertext Transport Protocol (HTTPS or HTTP-S)	安全超文本传输协议
Secure Sockets Layer(SSL)	安全套接层
secure control	安全性控制
smart card	智能卡
summary report	汇总报表
symmetric key encryption	对称密钥加密
transaction logging	事务日志
Transport Layer Security (TLS)	传输层安全协议
turnaround document	返回文档
unauthorized user	未授权用户
value limit control	限值控制

复习题

1. XML代表什么意思？解释XML和HTML的相似之处，并讨论XML和HTML的差别。
2. 比较一下使用DFD和使用顺序图定义输入的优点和弱点。你更喜欢哪种形式？为什么？
3. 解释系统边界。为什么用在数据流图中而没有用在顺序图中？
4. 在输入表单开发中，结构化能够提供信息中哪些是不能从DFD图中得到的？
5. 使用结构化方法如何确定数据字段？
6. 用UML和面向对象方法如何确定数据字段？
7. 解释输入表单的4种完整性控制。你最常见到的是哪一种？它们为什么重要？
8. 事务日志提供了什么保护措施？每一个系统都必须包含有事务日志吗？
9. 输出屏幕设计和输出报表设计要考虑的事项中有哪些不同？
10. 下钻的意义何在？试举例说明在报表设计中如何使用下钻方法。
11. 信息过载的危险何在？你能想到的避免信息过载的解决方案是什么？
12. 你认为哪些完整性控制应用到所有的输出报表上，为什么？
13. 信息系统中完整性控制的目标是什么？用你的话解释三个目标的含义是什么？为每一个目标举出一个例子。
14. 用于减少输入错误的4类输入控制是什么？描述一下它们各自是如何工作的。
15. 解释一下对数据库管理系统进行更新控制的意义。
16. 事务日志的基本目的是什么？Microsoft Access不包含自动事务日志。这是它的缺陷吗？还是说，事务日志并不是数据库完整性设计时考虑的重要内容？
17. 在打印出的输出报表上，报表打印日期和数据日期的区别是什么？
18. 安全性控制的两个主要目标是什么？
19. 解释三类用户访问权限。三类是确切的数目吗？能比三类少，或者比三类多吗？为什么？
20. 单密钥（对称）加密方法是如何工作的？它的优势是什么？它的缺点是什么？
21. 公共密钥（非对称）加密方法是如何工作的？它的优势是什么？它的缺点是什么？
22. 什么是数字证书？在安全子系统中授权验证扮演何种角色？
23. 什么是数字签名？它能告诉用户什么？

思考题

1. 本章描述了强调控制必要性的各种情况。在第一个场景中，一个家具店以赊购的方式销售它的商品。基于本章给出的控制描述，确定应该在系统中采用哪些控制，来确保只有具有正确权限的用户才可以修改客户的余额。
2. 在第二个场景中，一个可支付账目的雇员使用系统给供应商写支票。根据本章的信息，你将实施什么样的控制来确保支票写给正确的供应商，支票上的数量正确并且所有的支出得到必需的授权？如果不同的付款数量需要不同等级的授权，你将如何设计控制。
3. 公司的管理人员需要一份关于公司财政的专门的决策支持系统的报表。他们希望这份报表是基于过去几年的真实财务数据的。这种报表应有几个输入参数，以便管理人员能够根据过去的表现对将来的销售做"如果……，那么……"的分析。他们希望报表既可在线查看也可打印出来。你将采取什么样的控制手段以确保：① 只有被授权的管理人员才能索取这种报表；② 管理人员理解给定报表的基础（过去的和计划的数字）；③ 管理人员需处信息的敏感性并当做机密来对待。
4. 工资单系统有一个用于输入计时制雇员的计时卡信息的数据输入子系统。你将实施什么样的控制来保证数据是准确无误的？你将加入什么其他的控制来确保数据输入人员（可能是雇员的朋友）不夸大计时卡片上（已经由主管批准）的数据？
5. 基于第10章思考题3给出的DFD（图10-26），即"添加类到日程"，以及你设计的结构图，确定必要的输入输出界面集合，包括必要的数据字段。
6. 基于第10章思考题5给出的DFD（图10-27），即"特殊订单购买"，以及你设计的结构图，确定必要的输入输出。为每个界面和报表设计数据字段列表。
7. 图14-25描述了一个大学图书馆系统的两个用例的部分系统顺序图："借书"用例和"还书"用例。在此图基础上，依照图14-10和图14-12构造4个表单示：(1) 图书馆系统的输入，(2) 图书馆系统的输出，(3) 学生记录系统的输入，(4) 学生记录系统的输出。

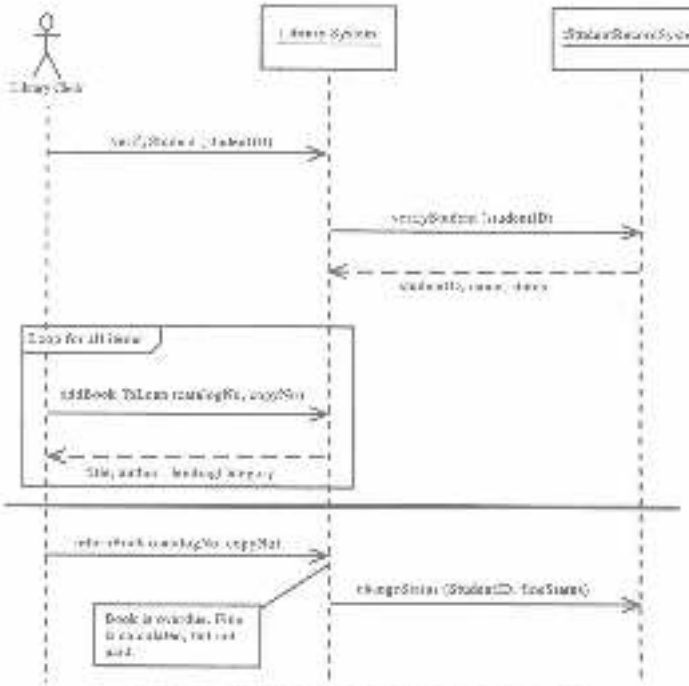


图14-25 大学图书馆系统的部分系统顺序图

8. 你所工作的连锁杂货店有大量客户。为了方便管理和加速结账，公司可能开发一个自助结账平台，客户可以核对自己的货物，并且使用信用卡或现金结账。你如何设计结账界面和设备？使用什么样的设备将使客户用起来简单、直接，使得价格输入正确并且现金或信用卡支付正确。换句话说，在结账台你将使用什么样的设备？在你的解决方案中，你可使用代表最新技术发展水平的解决方案或发明新的设备。

实验练习

1. 在网上查看电子商务站点（例如Amazon.com或者eBay）。评价一下用户界面的作用。系统中集成了哪些安全功能和控制，你发现了完整性控制所存在的潜在问题了吗？评价单个屏幕的设计。它的浏览和使用有多简单？你有什么建议使它们更容易使用。它们在降低数据输入错误方面的有效程度如何？
2. 检查本地商务信息系统（快餐厅、医生办公室、影碟店、食品店等），它其界面（如可能的话还有报表）的简易性和有效性进行评价。其中使用了什么样的完整性控制？界面使用起来简便吗？你将对其进行哪些改进？
3. 调查并找到一个正在建设或近期才建设好的系统。你或你的朋友所在的公司可能正在进行一项开发。另一个开发项目的来源是大学或学院。与其中一个开发者交谈，询问系统的完整性控制、界面设计的方法以及保证用户界面一致性的指导原则。询问输入和输出设计任务的数量和范围（例如需要多少界面和时间）以及设计界面和报表布局的方法（例如原型、CASE工具等）。
4. 如果你们大学使用Java，使用找到Jswing类库建立用户界面的系统。写一页关于Jswing类库、目的及其使用方法的描述。你的目标是说明你如何理解Jswing的概念以及在Windows环境中如何用它来建立窗口和输入界面。
5. 如果你们大学使用微软的Studio.NET，找出使用.NET类库建立用户界面的系统。写一页关于.NET类库、目的及其使用方法的描述。你的目标是说明你如何理解.NET的概念以及在Windows环境中如何用它来建立窗口和输入界面。
6. 浏览Internet，看你所能找到Pretty Good Privacy。它是什么？它是如何工作的？看你所能研究passphrase。它是什么含义？你可以从下列两个网址开始进行研究：http://www.pgpi.org和http://web.mit.edu/network/pgp.html。

实例研究

ALL-Shop大型超市

ALL-Shop大型超市是位于波士顿、纽约和华盛顿D.C等地的连锁超市。这个超市与其他像Wal-Mart、Kmart、Target等集团和零售商竞争。这些商店包括大型食品商店和家用织物、服装、汽车和家庭装修设备。总的来说，零售行业的利润非常小。食品这一部分一直很小，一般在5%～10%范围内。家用织物、服装等利润稍高一些，但是为了和Wal-Mart竞争，所有的利润都必须保持较低。

为了尽可能降低运作费用，ALL-Shop已经决定转向与它的供应商之间进行大量的电子数据交换（EDI）。ALL-Shop注意到，它的几个高级竞争者允许供应商管理它们自己仓库的库存水平。例如，卫生纸产品，如一次性尿布、卫生纸是用量大的产品，需要密切监控库存水平。ALL-Shop已经安装了先进的销售和库存系统用来跟踪各单项项目的日常活动（使用UPC码）。这些系统不但能获取各单项项目的日常活动，而且能维护活动数据仓库中的历史数据以支持在线数据分析。

ALL-Shop的第一步是使它的主要供应商有权访问它的日常销售和库存数据库。通过这种方法，供应商能够监控销售活动和检查库存，确保及时交货而使库存维持在适当的水平。系统也应允许

每个供应商访问和查看个人的支付状态和过去支付活动的历史记录。显然，所有这些信息必须由供应商来控制，而且供应商不可以查看其他供应商的信息。

1. 基于本章和前面章节中所学的知识，为作为参与者的供应商设计一个表示用例的用例图。即使这是一个系统到系统的界面，也可以把供应商系统认为是一个参与者。列出两列你认为对界面来说是必要的控制。第一列，标识出全部EDI接口可能需要的控制。然后，第二列为每一个表示的用例设计一套必要的控制。把你的分析建立在本章讨论的控制类型和完整性控制的三个主要目标的基础上。换句话说，你的任务是开发一个必要控制的说明，系统开发人员用它来确保ALL-Shop的财产和信息得到充分的保护。

2. ALL-Shop正在作规划，这个规划能使供应商可以访问数据仓库，这使他们能够分析过去的趋势并设计出促销策略，从而提高不仅仅是单个产品，而是整体的销售额。换句话说，ALL-Shop正在与它的供应商建立伙伴关系以扩大它在零售业市场的占有率。ALL-Shop管理人员主要关心的一个问题是，如何确保供应商以最大的安全性来对待这些信息，而不是用它来破坏ALL-Shop。因为供应商也同ALL-Shop实施竞争对手合作，那么如何能保证这些信息不被用来使它的竞争对手获益？

3. 你认为第一步对ALL-Shop来说是明智的一步吗？如果不是，为什么？如果是，那么应该用什么样的控制和契约方式来保护ALL-Shop？在第二例子中，你将看到对完整性控制重视不够将难以保护所有信息。在这个例子中要求对控制和控制的目标有更广泛的观察和理解。

房地产多编目服务系统

根据在第6章中设计的数据流片段和第10章中的结构图，设计一个输入的数据表以及相关的数据输入和每个输入的数据字段。同样建立一个所有必要数据字段的输出表。

THEEYESHAVEIT.COM图书交易系统

根据你在第7章中设计的顺序图和协作图，设计一个系统必需的输入输出列表，同时确定保证信息正确输入的必要控制。

城市影碟出租系统

使用你在第7章中设计的顺序图和协作图，为系统设计一个输入输出列表，连同必要的数据字段。

对落基山运动用品商店实例的再思考



RMO事件表列举了6个系统报表来作为新系统的一部分：

- 订单汇总
- 交易汇总
- 完成任务汇总
- 预期客户活动
- 客户调查
- 编目活动

对于以上各自报表，需要回答以下问题：

1. 确定每个报表应该包含的数据字段。
2. 用户希望每个报表能够回答的问题是什么？
3. 报表的所属类别是什么：详细报表、汇总报表还是异常报表？
4. 如何使用图形？下钻操作性能如何？

5. 你如何准备每一报表的实物模型，假定为打印输出还是在线输出？

6. 每个报表相关的输出控制是什么？

关注Reliable Pharmaceutical Services

医药公司面临的挑战是要跟上新药的发展和已有药品的变革。新药不断开发出来并获准上市。此外，普通药常常可以与品牌药相竞争。Reliable公司提供的一项服务就是找出处方中药品的最廉价的替代品来。节省费用的服务是客户卫生保健机构用来推广自己服务的一大市场优势。显然，这种服务在Reliable及其客户之间建立起了强大的信任。

为了跟上变化的步伐，Reliable预定了在线药品更新服务。这种服务提供了下列几种格式的更新，其中一个是XML文件。

1. 基于你在第11章中开发的类图设计的内容，描述一个XML输入文件的例子，此文件可以用来更新Reliable数据库中的药品信息。

2. 在前面章节中，用例描述中说明了无论何时需要填写并传送处方，将会为每个患者产生一个案例清单。基于类图中的数据，设计一个用例清单，需考虑在同一次处方递送时，病人会有多个处方的情况。

3. 每月Reliable会为每个客户保健机构产生一份报告。报告中将列出每个病人在本月中收到的处方。所有被填写的处方都被列出。每个处方列出下列信息：价格，病人医疗保险提供人的账单数量，医疗保险方支付数额，以及来自病人应支付的数额。设计这种月报告，同时确认并突出你认为适合这种报告的输出控制。

4. 在前面的章节中，你定义了用于收集来自客户卫生保健机构订单的输入窗体。回过来分析输入形式，并确定所有你认为对保证处方正确有必要的输入控制。要保证处方不出问题还可以有其他的程序或控制吗？说出你的建议。

参考资料

David Benyon, Diana Bental, and Thomas Green, *Conceptual Modeling for User Interface Development*. Springer-Verlag,1999.

Elfriede Dustin, Jeff Rashka, Douglas McDiarmid, and Jakob Nielson, *Quality Web Systems: Performance, Security, and Usability*. Addison-Wesley,2001.

Simson Garfinkel, Gene spafford, and Debby Russell, *Web Security, Privacy, & Commerce*. O'Reilly Publishing, 2001.

Anup K Ghosh, *E-Commerce Security: Weak Links, Best Defences*. John Wiley & Sons,1997.

IS Audit and Control Association, *IS Audit and Control Journal*, Volume I. 1995.

Brenda Laurel, *The Art of Human-Computer Interface Design*. Addison-Wesley,1990.

Ben Shneiderman, *Designing the User Interface: Strategies for Effective Human-Computer Interaction*. Addison-Wesley-Longman, 1998.

Donald Warren, Jr., and J Donald Warren, *The Handbook of IT Auditing*. Warren Gorham & Lamont, 1998.

Donald A. Wayne, and Peter B.B. Turney, *Auditing EDP Systems*. Prentice Hall, 1990.

第四部分 实现与支持

第15章 使系统可操作化

学习目标

阅读本章后，你应具备如下能力：

- 描述系统实施和支持活动
- 选择一条合适的途径来进行程序开发
- 描述不同类型的软件测试方法和使用每种测试方法的理由
- 列举出系统实施与转化的不同方法以及这些方法各自的优缺点
- 描述系统开发、维护所用的不同类型文档和程序
- 描述应用一个新系统所提出的培训、技术支持需求

本章要点

- 程序开发
- 质量保证
- 数据转换
- 安装
- 文档
- 培训与用户支持
- 维护和系统增强

TRI-STATE HEATING OIL公司：系统开始运行时的优先次序调整

星期一上午8：30，Maria Grasso、Kim Song、Dave Williams和Rajiv Gupta正准备进行一周一次的项目情况总结会。Tri-State Heating Oil公司已经在5个月以前开始开发一种新的用户订货和服务电话调度系统，预计完成时间是10周以后，但是这个项目的进展却已落后于计划。由于主要的用户对新的系统需求提出种种异议，使得系统的规模比以前要大得多，分析和设计阶段已经比预期多花了8周的时间。

Maria在会议的开始说："自从上次会议以后，因为单元测试结果比预期的要好，我们已经赢得了一两天的时间。上一周所有的开发方法均通过了测试，因此这周我们不必再检查这些代码是否有误。"

Kim说："我们不能太大意，上一次我们做过的一个面向对象的工程中所有令人讨厌的错误都出现在集成测试阶段，我们这周将完成用户界面所需的类，所以应该可以在下周的某个时间开始用事务类进行集成测试。"

Dave热情地点了点头并说："太好了！我们必须尽快完成对用户界面类的测试，因为我们计划将在3周以后对用户进行培训。我需要时间准备对用户培训的材料并且制订出最后的培训

计划。”

Rajiv回答说：“我们是否应该在系统的许多部分还处于开发状态时便准备早期培训计划，对此我没有把握。如果集成测试出现严重错误，我们要花很多时间去修改怎么办？还有，未完成的事务和数据库的类怎么办？我们能用这样一个在用户界面后面只有在一半的系统对用户进行培训吗？”

Dave回答说：“但是我们在三周以后必须开始培训，我们雇了12个临时工以便在新的系统上培训我们的职员。他们中有一半人计划将在两周后开工，其余人再过两周开工。现在要重新协商他们的开工时间已经来不及了，我们可以延长他们在这里的时间，但是推迟他们的开工时间就意味着我们要生怕他们不工作时也要给他们发工资。”

Maria大声说：“我认为Rajiv的顾虑是有道理的。系统刚刚完成和测试了一小部分，在三周以后要进行培训是不现实的，我们已经比原计划至少滞后了5周，在以后的几周内，我们也无法夺回多于4、5天的时间。我已经进行了调查，可以对一部分剩余的代码进行调整，从而使得用户培训的关键工作可以优先进行，有一些过程可以推迟一些进行。Kim，你能重新安排一下你的测试计划，而先处理所有的交互式应用吗？”

Kim回答说：“那我得回办公室，看看这些程序是否可以，我暂时同意，但是我需要几个小时的时间来确认一下。”

Maria回答说：“好吧，我们继续开会。假设重新整理编码并完成了测试，能为5周后的培训提供一个可用的系统吗？过一会儿，Kim有了确定的消息后，我会用电子邮件通知大家。我还要安排与CIO人员会面，通知他们关于临时雇员开销的坏消息。”

会场沉默了一会，Rajiv问道：“我们还需要考虑什么别的问题吗？”

Maria回答说：“让我再想一想……还有用户手册、硬件交付和安装、操作系统和数据库管理系统的安装、网络升级和对分布式数据库访问的加强测试。”

Rajiv笑着对Maria说：“你小的时候一定是个魔术师。把项目的所有方面都包含进来发表意见，这很好，管理部门为这个付你钱了吗？”

Maria咯咯笑着回答说：“我有时的确认为自己是个魔术师，如果管理部因此付我钱，那么项目一结束我就可以退休了。”

概述

本章着重讨论系统开发生命周期（SDLF）的实施与支持阶段的活动。在系统移交给用户之前的一系列活动叫做实施，系统投入使用之后所涉及的活动被描述为支持，图15-1列举了每个阶段的活动。



图15-1　实施和支持阶段的活动

实施和支持活动常常被认为是简单而枯燥的工作——它们不像分析和设计活动那样吸引人，这种情况类似于建筑学（Architecture）和建筑物（Construction）两者之间的区别。一个建筑师因为设计了一个新的建筑而获得了许多荣誉，尽管他的工作结果最终只是一张蓝图。

然而，在建筑蓝图勾画好以后，要将蓝图变为现实的建筑物还要有大量的工作要做。在这一点上与开发信息系统是相同的。

实施阶段所耗费的时间和资源比系统开发生命周期的前期阶段所耗费的要多。实施阶段需要大量人员完成实施任务——尤其是构造软件和测试。另外，不同实施活动是高度相互协调的。实施阶段的项目管理复杂程度最大，因为有那么多的人员和活动需要协调。RMO项目进度备忘录描述了项目管理复杂度以及必须完成的许多任务。

信息系统是现代组织的生命力。因此，支持这些信息系统是一个组织内部的最重要的工作之一。支持活动可以确保系统及其用户功能在系统初装多年以后仍然能够有效地运行。许多组织用在现有系统维护和技术支持活动上的花销比再建一个新系统的还要大。

2007年7月10日

To：John Blankens

From：Barbara Halifax

RE：客户支持系统实施里程碑

实施小组在本周初建立制订了一个开发控制进度表。截止到11月完成系统非常困难，但我认为我们应该由一个可行的计划。应用软件编程和测试的工作从现在开始，到10月初结束。有关其他活动的关键里程碑和目标日期些括：

9月1日	完成硬件和操作系统安装
9月15日	完成DBMS安装
10月1日	完成数据库初始化
10月20日	完成性能测试和调整
10月25日	开始用户培训
11月1日	开始与现有系统的并行操作
12月1日	终止现有系统

最后的日期假定并行操作发现不严重问题。并且在11月的财务验收测试期间高层管理停止活动。如果必要，我们将继续并行操作直到假日购物匹季结束，尽管那时会需要大量加班或雇佣临时人员。

制订进度表时很大胆，只留出一点儿时间用于改错，开发组将在每周一和周四上午8点开小组会，检查近期测试结果和即将开始的工作。如果可能的话，我希望您能参加所有的周一小组会，以便监督我们的进程并向小组成员强调新系统的战略重要性。

BH

15.1 程序开发

开发一个复杂的系统本来就是一个困难的过程。试看一下汽车制造的复杂性，必须制造和购买成千上万种零部件，一部分零件装配成小的子系统（如仪表装置、刹车系统等），这些子系统又组装成大的子系统，最终组装成一个完整的汽车。零部件和子系统必须经过制造、测试，接着交给后续的装配工序。有成千上万的单个生产步骤，每一步的实施，按期完工、花费和产品的质量都取决于其之前的所有处理步骤。

程序开发在许多方面类似于汽车的制造过程。要求和设计说明书早已确定，剩下的是复杂的生产和装配过程，这个过程必须确保有效地利用资源，在最短的工期内提交质量最好的

产品。但是不同于汽车制造的是，这个过程不能只设计一次，然后根据它制造成千上万个相似的单元，相反，对每一个新的工程而言，软件制造过程都是一个重新开发的过程，以便适应这个工程的独有特点。

许多人一想起系统开发，主要想到的是编程。编程不是开发阶段的唯一活动，但是显然它是最重要的活动。它的重要性体现在以下几个方面：

- 所需资源
- 管理的复杂性
- 系统质量

程序开发耗费的资源多于系统任何其他的开发活动。程序开发（包括单元测试）至少占用了全部开发劳动的1/3。它在系统开发计划中所占比重为1/3～1/2。由于程序开发要花费大量的资源和时间，我们理应精心地计划和管理它。

15.1.1　系统实施的顺序

程序开发所要做的一个最基本的决定就是确定程序组件的开发顺序。以下是几种可能的开发顺序：

- 输入、处理、输出
- 自顶向下
- 自底向上

对项目的需求和限制，每个工程必须采用其中的一种或几种方法的组合。

1. 输入、处理、输出（IPO）的开发顺序

这种顺序是基于一个系统或程序的数据流的，先开发包含外部输入的程序和模块，再开发处理这些输入数据的程序或模块（即把输入转换为输出），最后开发产生输出结果的程序和模块。

输入、处理、输出（IPO）的开发：一种程序开发顺序，先开发输入模块，再开发处理模块，最后开发输出模块。

因为对于结构化的设计和编程，一个分析员可以通过检查系统流程图和结构图来决定IPO的开发顺序。例如：如图15-2所示的工资单系统的流程图。“维护税表”和“维护雇员数据库”程序是获取和修改输入数据的部分，所以应该首先开发它们。“工资单”程序结合了输入和处理数据，应该在下一步开发。“检验打印”和“年终税”模块将产生系统输出，所以它们最后开发。

图15-3显示的是工资单程序的结构图。分析员可以按照IPO的实施顺序，把它们分成输入模块、处理模块和输出模块。如果这个结构图是由转化分析得到的，那么模块应分别组织为传入（输入）模块、中心转化（处理）模块和传出（输出）模块（关于结构图的组织参见10.3.3节）。对这个程序而言，首先开发结构图中的传入模块（如图15-3所示，包括叶两千片），接着就是转化模块（如图15-3所示，包括计算总量），最后是传出模块（如图15-3所示，包括输出工资单）。

IPO的开发顺序也可以应用于面向对象的设计和程序开发中。关键是要分析数据的相互依赖性——即某些类或方法获取或者产生的数据是另外一些类或方法所需要的数据。类中关于数据依赖性信息来源于程序包图表。类图类型能指导订单决策的执行。

例如，图15-4中的程序包图表显示出客户和目录维护子系统既不依赖并且不依赖于其他两个子系统中的任何一个。而订单输入子系统依赖于客户和目录维护子系统。而订单执行子系统则依赖于订单输入子系统。

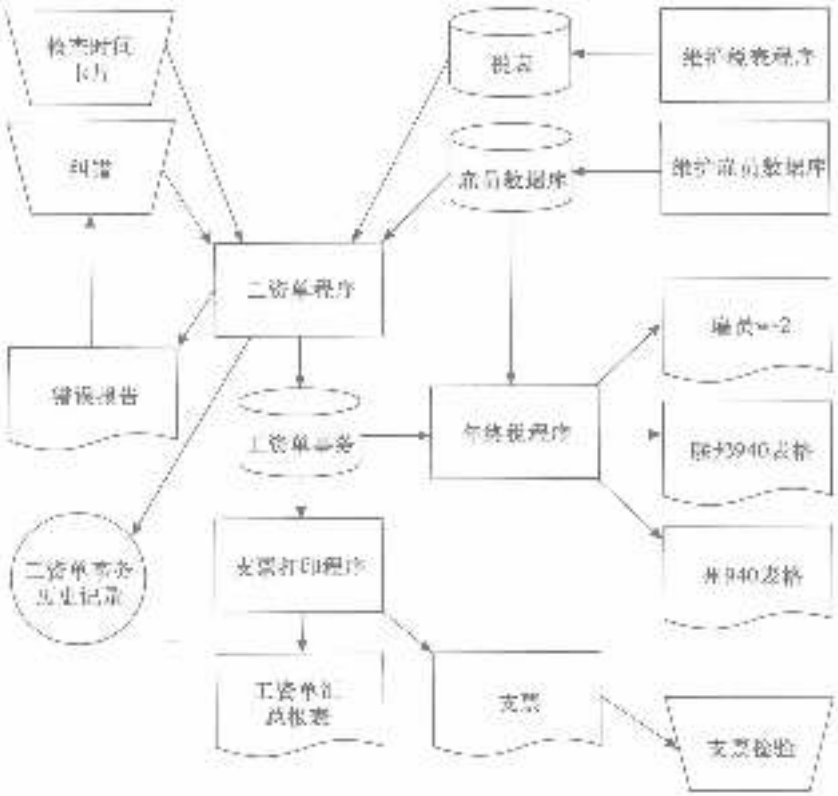


图15-2 工资单系统的系统流程图

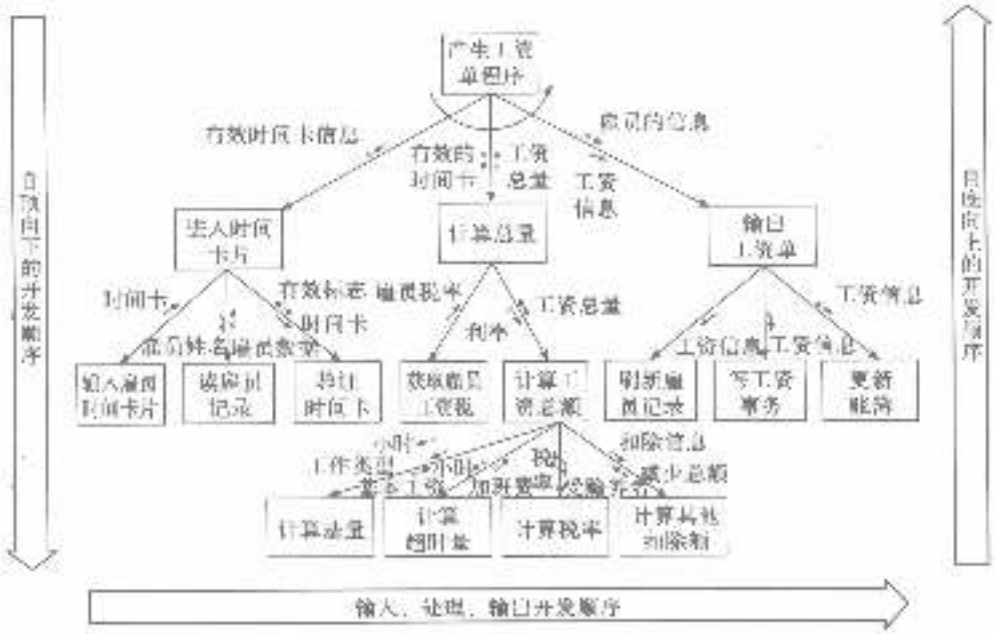


图15-3 图15-2中工资单系统的结构图

存在于程序包（子系统）中的数据依赖性暗示了嵌入类之间的数据依赖性。因此，客户

类、目录类和程序包类与RMO系统中其他的类之间无数据依赖关系，按IPO的开发顺序，应该首先实施这三部分。

IPO开发顺序的主要优点是简化了测试。因为最先开发输入模块和程序，它们可以用来为处理模块和输出模块输入测试数据，减少了编写专门程序来生成测试数据的必要性，从而加快了开发的进程。

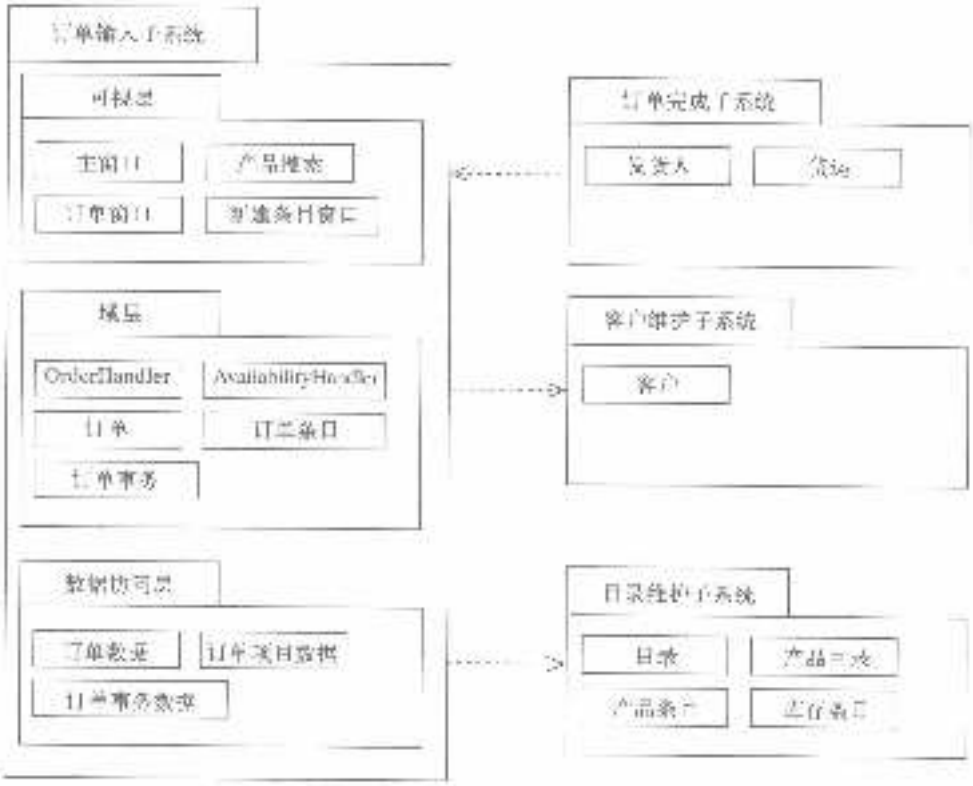


图15-4　4个RMO子系统的程序包图

IPO开发顺序还有一个优点，重要的用户界面部分在早期就已着手开发。用户界面部分比起系统的其他部分而言，更容易被要求修改。尽早开发这部分可以使测试活动提前，用户也可以尽早对系统进行评估，一旦需要改动，仍然有充足的时间来完成。尽早开发用户界面对其他相关活动也具有带头作用，如用户培训和编写文档等。

IPO开发顺序的一个缺点是，输出部分的滞后实现。输出程序对于测试面向过程的模块和程序十分有用，分析员可以通过人工检查打印的报告和显示的输出结果来发现处理过程的错误。IPO的开发方法直到系统开发后期才进行这部分的测试。但是，分析员可以经常使用查询程序来产生替代的输出结果或是报告一个数据库管理系统的写能力。如果这样的输出能够尽早尽快定义出来，那么将大大克服IPO输出部分滞后实现的缺点。

2. 自顶向下和自底向上的开发顺序

自顶向下和自底向上的术语来自于结构化的设计和结构化的编程。这两个术语描述了模块的实施顺序与其在结构图中的位置有关。例如，如图15-3所示，自顶向下的开发方法开始于结构图中最上层的模块，而自底向上的开发方法开始于结构图中最下层的模块。

自顶向下的开发方法：先实现结构图中的上层模块的开发顺序。

自底向上的开发方法：先实现结构图中的下层模块的开发顺序。

自顶向下和自底向上的程序开发方法也可以用于面向对象的设计和编程。虽然面向对象时表描法的可视过程不像结构图表那么清晰。关键问题是方法的依赖性——哪些方法调用哪些其他方法。在面向对象的子系统或类中，方法依赖性可以根据第11章介绍的导航可见性来衡量。

例如，考虑图15-5所示RMO订单输入子系统的三层设计部分。包和类之间的箭头显示了导航可见性需求。可视层中的方法调用域层中的方法，域层中的方法接着再调用数据访问层中的方法。自顶向下开发方法首先实现可视层中的类和方法，然后是域层，最后是数据访问层。自底向上开发方法的实现顺序正好相反。



图15-5 一个三层OO设计的包图

依赖性方法也可以源于顺序图表。例如，在图15-6中，依赖性方法源于对象之间从左到右的消息流图。把图向右旋转90°就成了自顶向下和自底向上可视化分析类似于结构图。自顶向下开发方法以产品搜索、供货处理器、目录数据和目录开始，自底向上开发方法以库存条目、库存数据、产品目录和产品目录数据开始。

自顶向下开发的主要优点是程序总是可以运行的。例如，如图15-3所示自顶向下的开发由一个顶层模块和骨架的二个子模块（存根模块我们将在15.2.2节中做进一步的探讨）组成。这一套完整的模块可以被编译、链接和执行，虽然这时的程序还不能实现什么功能。图15-5中三层设计的自顶向下开发最开始是由可视层类和域层类的虚构型（或存根程序）组成的。

一旦顶层的模块被实现了，实施活动就转移到结构图或程序包图中的下一个层次。每个模块或类都被实现后，下一个更低层次的剩余模块或类的存根程序将被加入。在开发过程中，每一阶段的程序都应该是完整的（就是说它应该能够被编译、链接和执行）。然而，随着开发的深入，越来越体现出程序的复杂性与可实现性。

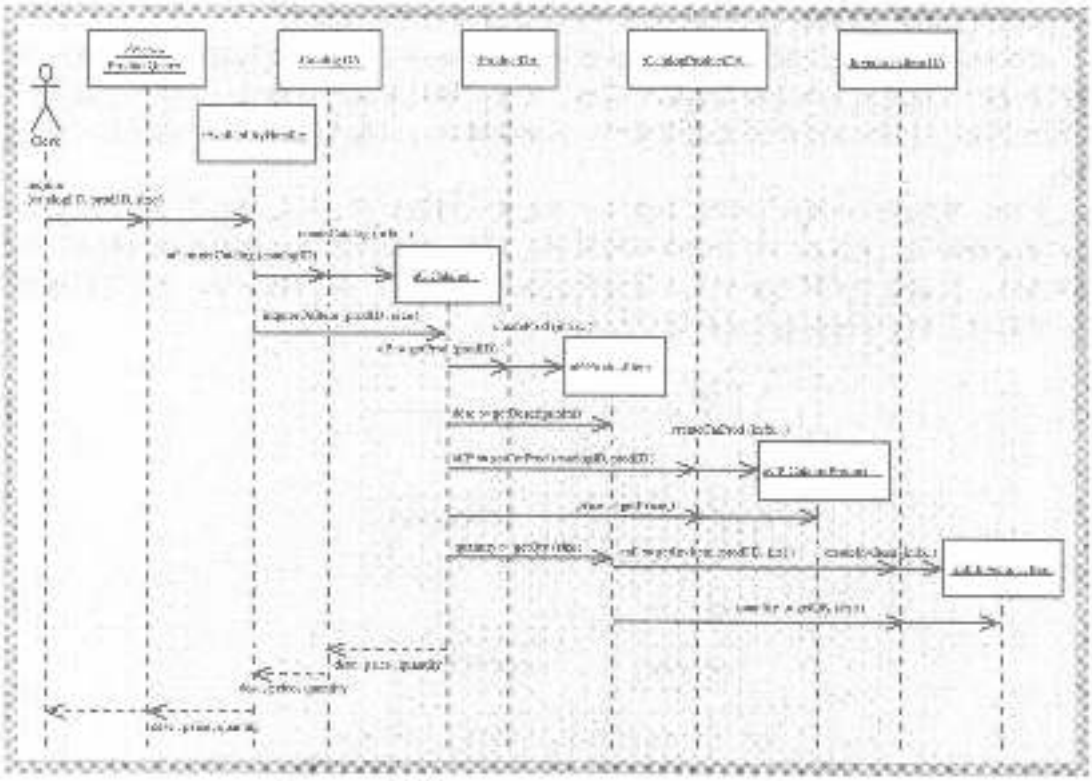


图15-6　事件“Look up Item availability”的顺序图

自顶向下开发方法的主要缺点是在系统构造初期不能很有效地利用程序员的人力资源。在大量的模块和方法被同时开发之前，开发工作只是进行两三个重复性的活动。但是，如果能够快速地完成开始阶段的重复性编程，那么此方法的不利因素将减至最小。

自底向上开发方法的主要优点是，许多相关程序员可以立即投入开发工作。另外，底层模块的编写是最复杂和最困难的，所以，尽早开发那些底层模块可以为开发和测试活动争取到更多时间。遗憾的是，自底向上开发方法要求编写大量的驱动程序来测试底层模块，这给开发和测试程序添加了许多麻烦（我们将在15.2.2节中做进一步的探讨），而且，整个系统直到最顶层模块被实现后才算完整。因此，系统整体测试也相应地被推迟了。

3. 其他开发顺序的考虑

IPO、自顶向下和自底向上的开发方法仅仅是程序开发规则的起点，还必须考虑其他的因素，其中包括用户反馈信息、培训、文档和测试。用户反馈信息、培训和文档都依赖于系统的用户界面。尽早地实施用户界面，可以使开发进程中的用户培训和用户文档的编写工作早些开始，而且还允许用户针对界面质量和可用性提出反馈意见。注意，如果在分析和设计阶段用户界面的原型已被构建完毕并确认，那么界面的构建实质上是提前了。

在统一过程工程中，用例是开发工程划分迭代的主要基础，在大多数UP工程中，开发者对单个迭代选择相关用例集，并对这些用例完成所有需求设计、实现和测试活动。用例集的选择基于很多因素，包括最小化工程风险、有效地使用非技术员工，或提前安装系统的某些部分。比如，不确定需求或高技术风险的用例作为早期的迭代。

在决定开发顺序的时候测试也是一个重要的考虑因素。随着单个的模块或方法的实现，也必须对它们进行测试。程序员们必须尽早地发现程序中的错误，因为随着开发进程的推进，

要找到这些错误越来越困难，修改错误的开销也越来越大。找出软件中易受错误影响的部分和找出软件中可能产生影响整个系统的重大错误的部分同等重要。这些部分不论存在于基本的IPO、自顶向下和自底向上方法中的任何部位，都应该尽早地构建和测试。

测试和构建是高度集存的，因此，在任何一项活动开始之前，通常要制订一个包括构建和测试两方面内容的正式计划。构建和测试计划包括许多细节，如：

- 开发顺序
- 测试顺序
- 用于测试模块、模块群、程序和子系统的数据
- 验收标准
- 人员的安排（构建和测试）

测试将在本章的后面详细讨论，但是现在要牢记构建和测试是联系起来的。它们的相互依赖性和复杂性使得我们有必要制订一个正式的计划并把计划和实际工作形成有规律的对比。

15.1.2 框架开发

构建一个大型的面向对象的系统时，通常的做法是建立一个对象框架（或一系列的基类），它包括大多数或全部的应用程序中所包含的域和数据访问层类。例如，为银行构建一个面向对象的账目维护系统，开发者会设计和建立一系列的类，来代表消费者和不同类型的银行账户（如储蓄、支票、存折等）。

基类可以在系统的许多部分和不同的应用程序中被重用。正因为这样，它们是系统的关键组成部分，基类中的错误将会影响到系统的每一个程序。另外，后期对基类的更改将会使整个系统发生重大变化。

最早采用基类是想减小更改错误所带来的影响。基类的开发任务往往分派给最优秀的程序员，并且对于基类测试比其他类更加全面彻底。尽早地全面测试基类，可以保证在编写基于这些基类的其他代码前，就能发现基类中的错误和问题。

15.1.3 基于小组的程序开发

程序开发通常是一个小组的程序员一起工作。使许多程序员同时开发系统的不同部分，可以缩短开发周期。然而，成组开发项目也会带来一些管理上的问题：

- 编程组如何组织
- 特定的小组或成员如何分派任务
- 成员与小组间的交流与协作

这里有许多开发小组的组织方法，一些常用的组织模式包括：

- 平等合作
- 首席开发者
- 协作专家

表15-7总结了不同类型小组的特征以及各个团体所适合承担的项目和任务。

一个平等合作小组包括的成员有以下特征：他们有着大体相同的技术、经验和相似的专业背景。尽管他们被分派了不同重要性和复杂程度的任务，但都被视为是平等的。决策需要大多数人同意才能通过。成员之间经常交换信息以形成决策意见。

平等合作小组：由有着大体相同技术、经验和相似专业背景的人员组成的开发小组。

一个有首席开发者小组类似于一个小的军事单元，一个被指派的领导行使着一些职责，包括技术咨询、小组协作和任务分配等。在这种类型的小组中，成员间的交流比平等合作小

组要少得多，尽管首席开发者会向各个成员征求意见，大多数的重大决策还是由首席开发者来决定。

小组类型	小组特征	任务和项目的类型
平等合作	成员的技术水平 专业的重叠 基于多数人意见的决策	实验 创造性的解决问题
首席开发者	像军队中的作业班一样的组织 一个领导做出所有重大决策	定义好的目标 定义好的完成途径
协作专家	各不相同的技能和经验 技术专业最小限度重叠 领导主要是一个行政管理者 基于多数人意见的决策	诊断或实验 创造性和综合性的解决问题 技术范围涉及极广

图15.7　不同类型开发小组的比较和总结

首席开发者小组：小组只有一个领导者，所有的重大决策都由他做出。

协作专家小组类似于平等合作小组，但是它的成员在技术和经验方面有着很大的差异，重叠的方面很少。这种小组成员通常来自于不同的组织单位。可能也会指定一个负责人，但此负责人只是行使行政管理职能（如制订计划、协调并与外界人员打交道）。虽然成员在自己所从事的工作领域中是专家，但技术决策通常由集体一致来决定。在大型开发项目中，协作专家小组可能只作为流动人员在其他小组中帮助处理一些复杂的问题。

协作专家小组：小组成员在技术和经验方面差异很大，很少有重叠。

小组的一些组织原则是所有开发项目和组织结构的基础。其中之一就是保持小组规模相对较小（不超过10人），较大的组织机构工作起来效率不高，因为大的群体内部的交流和协调工作比较复杂。如果某个项目组超过了10人，最好将其分成几个较小的组（一般每个小组是5个人）。每个较小的组承担此项目的各个相对独立的部分，并且每组要有一个人负责与其他组的协作和交流，这样只有一个接触点，简化了小组间的交流，使得每一个小组只需要完成特定的功能。

小组的另一个组织原则是小组的组织结构应该与任务和工程的性质相匹配。如果定义好了实施任务，并且这个任务对于成员的知识和技术可行性没有限制，则最好组织成首席开发者的组织形式。在这样的环境下，有首席开发者的小组常常工作的效率很高。

要求实验性或高创造性的工作最好由平等合作小组和协作专家小组来承担。因为小组人员之间那种更开放式的交流合作关系特别适合于那些要求产生和评估大量观点的任务。成员们在技术、专业和经验上的接近使得对每个观点的评估更加全面。

专家协作小组非常适合承担技术跨越范围很大的任务，他们也适合综合问题的处理（如诊断和修改存在于复杂系统的错误）。然而，成功取决于真诚协作的过程，这一点对技术和经验有较大差别的成员来说，有时是不容易做到的。

成员的技术要和所承担的任务大体相匹配。数据库管理系统、用户界面和数值算法显而易见要求相应的技术匹配。小组也需要非技术性的技能和特性，包括产生新观点的能力，集中众人意见的能力，具体管理能力以及与外界打交道的能力等。项目经理在项目开发早期应规划好执行方案，以避免项目延迟和个人分配任务不恰当。

15.1.4　源代码的控制

开发小组需要一些工具来帮助协调他们的编程任务。源代码控制系统（SCCS）是一种自

动工具，用来跟踪记录源代码文件并控制对这些文件的改动。SCCS把工程的源代码文件存储在一个仓库中。SCCS就像一个图书管理员——完成登记和检查手续，跟踪记录每个程序员拥有哪些文件，确保只有授权过的用户才有权访问这个仓库。

源代码控制系统（SCCS）：一种自动工具，用来跟踪记录源代码文件并控制对这些文件的改动。

程序员们对此仓库中的文件可以做以下操作：

- 以只读方式访问文件
- 以读/写方式来访问文件
- 对文件进行修改

当程序员只想检查而不改动代码时（例如，检查某个模块与别的模块之间的接口），他可以以只读方式来访问文件；当程序员想改动代码时，他可以以读/写方式来访问文件。在同一时间，SCCS只允许一个程序员以读/写方式来访问文件，在另外一个程序员能够以读/写方式访问该文件之前，此文件必须已经被放回到仓库中。

图15-8显示了Microsoft Visual SourceSafe的主界面。主界面上显示了来自于RMO客户支持系统的不同的源代码文件，一些文件正在被程序员访问。对每一个以读/写方式访问的文件，界面上将列出正在访问的程序员的名字，访问的日期，以及此文件所在的位置。每个被访问的文件都有一个红边的边框和检查标志。

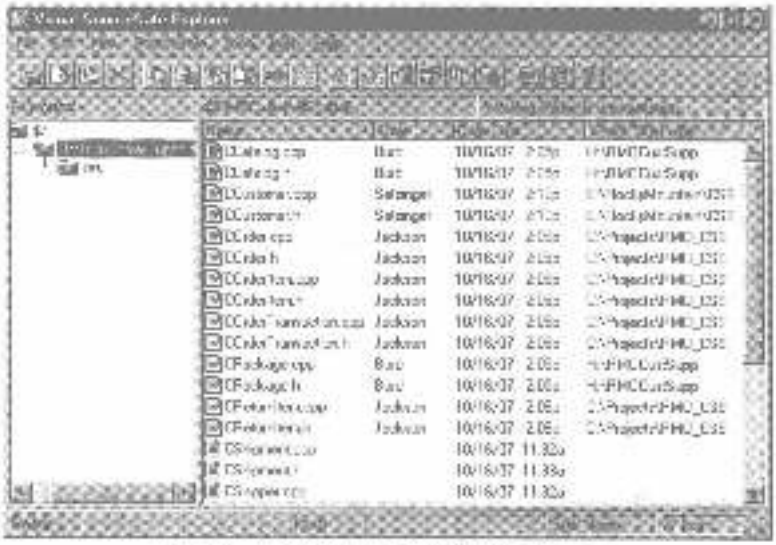


图15-8 由源代码控制系统管理的工程文件

SCCS禁止多个程序员同时更新同一个文件，从而防止了对于源代码改动的不一致性。当多个程序员合作开发程序时，对于源代码的控制是绝对必要的。它防止了程序改动的不一致性，并且能够自动地在程序员和小组之间进行协调。这个仓库也可以作为通用工具来实现备份和恢复操作。

15.1.5 版本

中大型规模的系统是复杂而不断变化的。在实施过程中，变化发生得很快，而以后就变化得慢了。系统的复杂性和易变性使得管理上产生了许多问题——尤其是在测试和支持活动中。系统总是在变化的，因此在这样的环境下，测试结果总是有问题的。等找到了错误所在，

引起错误的代码可能早被移动、替代或删除掉了。由于相似的原因，支持工作也变得复杂起来。当系统安装在用户的计算机系统上时，技术支持人员需要知道系统的状态，才能对所出现的问题做出反应。

为简化测试和支持的工作，复杂系统的开发、安装和维护是在一系列的版本中进行的。对终端用户，在被开发的系统中有多个版本以及在不同的开发阶段有更多的版本的情况并不常见。在开发过程中生成的系统版本叫做测试版本。测试版本有一些特定的特征，代表了最终完成系统过程中坚实的一步。测试版本提供了一个静态的系统映像和用于评估工程进展的检查点。

α版本是一个未完成的但是已准备好了接受严格测试的系统。多个α版本是根据系统的大小和复杂性来定义的。α版本的生命周期很短（通常只有几天或几周）。

α版本：一个未完成的但是已准备好了接受严格测试的系统。

β版本是一个足够稳定的系统，可以接受终端用户的测试。一个β版本是经过一个或多个α版本测试完毕，确认已知错误都被改正之后产生的。终端用户通过使用β版本实践真正的工作来测试它。β版本必须更加完整并且比α版本产生重大错误的机率要小。β版本将经过数周或数月典型测试。

β版本：一个足够稳定的系统，可以接受终端用户的测试。

对用户发布的能够长期使用的系统版本叫做**产品版本**、**发布版本**或**产品发布**。尽管在传统意义上，软件系统几乎不可能在此期间完成，但产品版本通常被认为是最终的产品。最小产品版本（有时叫做维护版本）则可提供纠错和对已有特征进行较小改动。主要产品版本则增加了许多新功能，也可能是旧版本的全部重写。

产品版本、产品发布或发布版本：正式对用户发布或可投入使用的系统。

维护版本：对已有功能提供纠错和较小改动的系统更新。

图15-9显示了RMO客户支持系统的一系列的测试和产品版本，在图15-10中对每种版本都给予了描述。这个系统是以两种产品版本的形式发布的——1.0版本和2.0版本。每个产品版本在发布之前都经历了一次或多次的α测试版本和β测试版本。每次都对以前的那些版本添加或升级了一些功能，并修改了一些错误，1.1版本就是1.0版本的一个维护版本。从时间线上可以注意到，2.0版本包含了1.0版本去维护需要所做的变更。用未来产品版本的测试版本来覆盖旧版本，这一点是典型的特征。



图15-9　RMO客户支持系统的测试和产品版本的时间线

版本跟踪是很复杂的。每个版本都需要唯一地标识出是给用户还是给测试者。在Windows下运行的应用程序，其版本信息在标准的帮助菜单的“关于”选项中可以得到，如图15-11所示，寻求技术支持或者想报告错误的用户可以根据这个特征向测试者或技术支持人员报告系统的版本。

控制同一个系统的多个版本需要有精密的版本控制软件，SCCS中通常都具有所需的版本控制功能。程序员和支持人员可以选取当前的版本或任何

以前的版本来执行、测试或修改，修改以后将保存成一个新的版本，这样可以使历史映像保持原样。



图15-10 RMO客户支持系统的各个版本的描述



图15-11 典型的Windows应用程序About框

beta版本和产品版本只要被安装在用户的机器上，就必须对其进行备份。备份的版本是用来测试未来的错误报告的，例如，当用户报告1.0版本出错时，支持人员将从存档中调出此版本并安装，力图重现用户出过的错误，反馈给用户的信息也是针对1.0版本而言的，即使最近的产品是更高版本的。

实践指导

为了帮助程序员和支持人员，把分配版本号和用户展示给他们。 ■

15.2 质量保证

对任何商务过程或系统而言，质量是信息系统所关注的主要问题。质量保证（QA）是保证信息系统满足用户、技术人员和管理人员最低质量标准的过程。QA有时就是在程序代码中纠错，但是这种理解是狭隘而不完全的。QA包括一系列的活动，贯穿于整个SDLC（软件开发生命周期），这样做有利于从一开始就尽可能地检错纠错，建立一个正确的系统。在工程的早期就进行综合质量保证活动，可以在许多编码中避免错误，并且可以确保所开发的系统切实满足用户和组织的要求。

质量保证（QA）：保证信息系统满足最低质量标准的过程。

在分析阶段，QA活动主要集中于明确系统需求中的分歧和矛盾；在设计阶段，QA活动主要集中于使系统满足需求，并使得设计决策正确，以实现尽可能无错的程序；在实施阶段，QA活动主要是测试。然而，在许多工程中，设计阶段和实施阶段相互重叠，因而，设计的质量保证活动也常常综合到测试活动中去。

在设计和实施过程中，QA活动常常会被行了事。归纳起来，有以下原因：

- 进度的压力使得各个阶段紧紧相连，在只追求速度的情况下，常常忽略了QA活动；
- QA活动要求开发人员公开其工作，以便别人全面检查和提出批评，很多人并不情愿这样做；
- 许多人认为测试人员总是带来坏消息，他们错误地认为没有消息就是好消息，而忽略测试就是避免坏消息的一种方法。

这里有一种简单办法可以阻止人的惰性和进度压力对QA活动的影响，那就是在项目开始时把QA活动列入计划中，并坚持不放弃它。不论会对进度及预算产生什么影响，质量标准必须明确，具有可度量性，对不符合标准的产品必须做出修改。这种QA方法从高层到底层都需要一个组织承诺。不幸的是，高层管理人员常常是缩减QA活动以加快项目进度的来源。

在开发过程中建立QA的另外一个关键因素是，建立一个开放的、平等的、在独项目参与者之间互相尊重的环境。职员们必须能够接受建议和批评，并且愿意对别人提出建议和批评。如果允许恶意的批评和肆意的指责存在，那么QA活动将不会有效地发挥作用。

随着开发阶段的推进，纠错的开销不断增加。错误最好在分析设计阶段就能测试到，这样错误就不会转移到程序代码中，在实施阶段的早期修改程序错误要比在阶段后期验收测试时修改容易，最糟糕的情况是系统已经投入运行后再去纠错。总之，在整个开发生命周期中，为QA付出努力是值得的。

15.2.1　技术复审

大多数程序员都曾有这样的经验，他检查不出来自己程序的错误，因为"看不到它们"，但是当他的源代码摆在另外一位程序员面前时，那位程序员立即就会发现他的错误。最普通的例子如：关键字拼写错误，暗形的if条件语句，在源代码中使用非法字符。这种错误在各种层次水平的程序员中都存在。

技术复审可以公开设计和构建的过程，为别人提供发现错误和提出建设性意见的机会，技术复审存在的范围不一，可以在组织和组织之间进行，也可以在同一组织的不同项目之间进行，技术复审的形式有正式的，也有非正式的。

技术复审：一组开发人员对设计和构建细节进行正式或非正式的复审活动。

遍查是一种技术复审形式，它是指由两人或多人来审查模型或程序的正确性和完整性。实施阶段可以使用遍查的方法，在分析和设计阶段也常使用。在设计和实施阶段，遍查是一种技术复审，是两个或多个开发者为评估和提高项目质量而进行的对于设计或实施的审查活动。典型的情况是，其中一个开发者在遍查前已经做出了有关的模型或模块，由他本人阐述以此为基础的设想和操作，然后由别的参与者进行评论和提出建议。

检查是一种更为正式的遍查活动。参与者在开会之前就要浏览，分析相关材料。审议材料包括要检查的代码，有关的模型（如结构图和类图等）和对可能遇到的错误类型的注释。小组开会之后往往要形成一个标准文件。

检查：一组开发人员对设计和构建细节的正式复审，每个开发者都有明确的任务。

在开小组会时，参与者们扮演不同的角色，包括提交者、质疑者和书记员。提交者（通常是本模块和代码的开发者）对要被检查的材料做总结，然后质疑者提出他们事先发现的错误和所关心的问题。所有的错误和问题都由大家进行讨论。参与者讨论解决问题的策略，书记员则记录所有的错误和相应的解决策略。

遍查和检查是很重要的QA过程，因为它们可以在编写代码之前就检测出错误。研究表明，技术复审可以达到以下目的：

- 可在测试之前减少由5－10种因素导致的错误的数量；
- 节约了大约50%的测试开销。

技术复审可以减少开发费用并缩短开发周期，这是因为大量的错误不会遗误到在编码、测试、诊断和修改阶段。

测试和技术复审都可以发现50%—75%的错误，有些错误可以很容易就被任意一种方法发现，但是有些错误采用某一种方法很难被发现，而采用另外的方法则变得很容易发现，因此，这两种方法结合起来比单独使用更加有效。

实践指导

综合技术复审和测试可以使效率最大化。 ■

15.2.2 测试

测试是对产品进行检验以确定其缺陷的过程。程序员必须已经完成了此软件，并且明确缺陷的标准，才能对软件进行测试。开发人员可以通过复审产品的结构组成或运行产品的功能，并检查运行结果来对其进行测试。本节着重讨论后一种类型的测试，过程如图15-12所示。

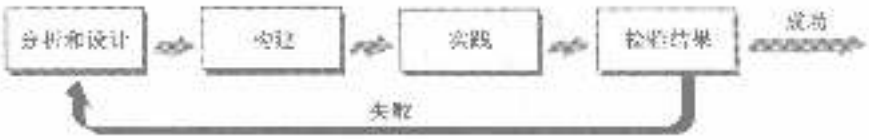


图15-12 软件测试的一般模型

1. 软件测试

一个信息系统是软件各个组成部分的综合体。各组成部分可单独测试，也可进行集成测试，或者整个系统进行系统测试。对组成部分单独进行的测试叫做单元测试，包含多个单元模块的测试叫做集成测试，对整个系统进行的测试叫做系统测试。每种类型的测试在本节的后面都有详细论述。

这三种类型的测试在SDLC的特定阶段是相互联系的，如图15-13所示。系统测试就是检查整个系统的性能，涉及技术和用户需求方面的问题。这些需求在SDLC的设计阶段就已经确定了。在高层设计过程中，系统被分割为几个高层模块，而且每个模块的结构设计也已经被确定了。集成测试就是测试软件的相关联的组成部分的性能，低层次设计只需关注单个模块的内部结构。单元测试就是对软件的每一个组成部分进行单独的测试。

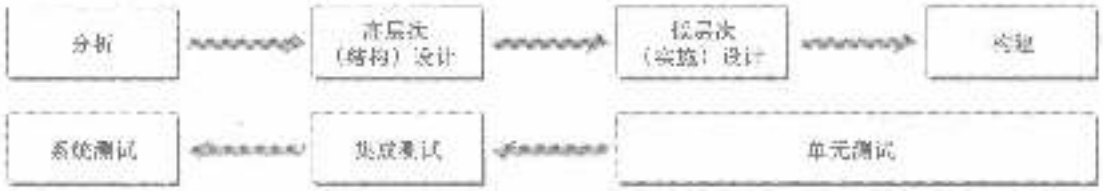


图15-13 SDLC各个阶段之间的关系以及不同类型的测试

因为每个层次的测试都与SDLC的特定阶段相联系，如图15-14所示，所以，测试工作贯穿于整个生命周期。每种类型的测试计划产生于SDLC的相关阶段，一旦完成了测试计划，便可以进行具体的测试。但是，直到系统的相关部分都已完成后，才能进行测试工作。

开发测试的一个关键部分是确定测试实例和测试数据。测试实例的正式描述如下：

- 开始状态

- 软件必须响应的一个或多个事件
- 期望得到的响应或结束状态



图15-14　SDLC的各个阶段和每个阶段可进行的测试工作

测试实例：对开始状态、软件必须响应的一个或多个事件，期望得到的响应或结束状态等内容的正式描述。

开始状态和事件都由一组**测试数据**代表。例如，系统的开始状态也许会代表一组数据库的内容（如存在某个特定的客户以及该客户的订单）。事件可以代表一组可输入的数据项（如客户的账号和用于查询订单状态的订单号）。所期望的反应可被描述为行为（如显示某种信息）或存储数据的确定状态（如一个取消的订单）。

测试数据：用于测试一个或一组模块乃至整个系统的一组开始状态和事件。

准备测试实例和测试数据是一段枯燥而且耗时的过程，在程序或模块级，每一条程序指令都必须至少被执行一次。要确保所有指令在测试中都被执行过，这是一件非常复杂的事情，幸运的是，已经有基于数学技术的自动工具来生成完整的测试实例。请参阅“参考资料”部分中Watson和McCabe的论文。该论文对这个话题进行了全面的讨论。

在准备测试实例时可以利用分析阶段文档。如果系统是由面向对象技术分析和设计的，就会开发出一个测试实例和场景的完整集合。因此，开发者必须为每一个用法实例和场景准备测试实例。应该为每一个场景准备许多既代表正常又代表异常处理情形的测试实例。

传统分析模型与测试实例之间的对应关系不太明朗。开发者可以使用数据流图和事件表作为准备测试实例的主要原则。开发者应该为每个事件准备多个测试实例，并且保证数据流图中的每个过程都应该由至少一个测试实例进行演习。

2. 单元测试

这是在与其他模块进行集成测试之前，对单个代码模块进行测试的过程。单元测试有时也叫做模块测试。事实上，单元测试可以用在结构化或面向对象的软件中。我们称之为“单元”的可能会是函数、子程序、过程或方法（在本节后面我们用术语“模块”来代表这些程序结构中的任意一种）。单元还可以是相联系的模块组成的较小的“组”，它们通常作为一个“组”来执行。单元测试的目的是在单个模块组成大的软件单元（如程序、类和子系统等）之前，尽可能地找出并改正其中的错误。当许多的模块组成大的软件单元后，再进行检错和纠错就变得非常困难和代价昂贵了。

单元测试或模块测试：在与其他模块进行集成之前，对单一的代码模块或方法进行的测试活动。

设计出来的模块很少是孤立运行的，相反，大量的模块是被组装起来作为一个整体来运行的。模块可能调用其他的软件单元来执行任务，也可能被别的模块调用。这种关系在图15-15中很容易看出，尽管它也存在于面向对象软件的方法中。例如，模块“计算工资总额”就被模块“计算总额”调用，而它又调用了结构图中另外的三个模块。

如果计算工资总额的模块要单独进行测试，那么要求有两种类型的测试模块。第一种类型的模块叫做驱动程序。驱动程序用来模仿模块的调用行为。驱动程序能实现以下功能：

- 为测试的函数设置输入参数值；

- 调用要测试的模块，并把输入参数传递给它；
- 接收被测试模块的返回参数，并打印或显示它们。

驱动程序：一种为单元测试而开发的模块，用来模仿尚未开发的模块的调用行为。

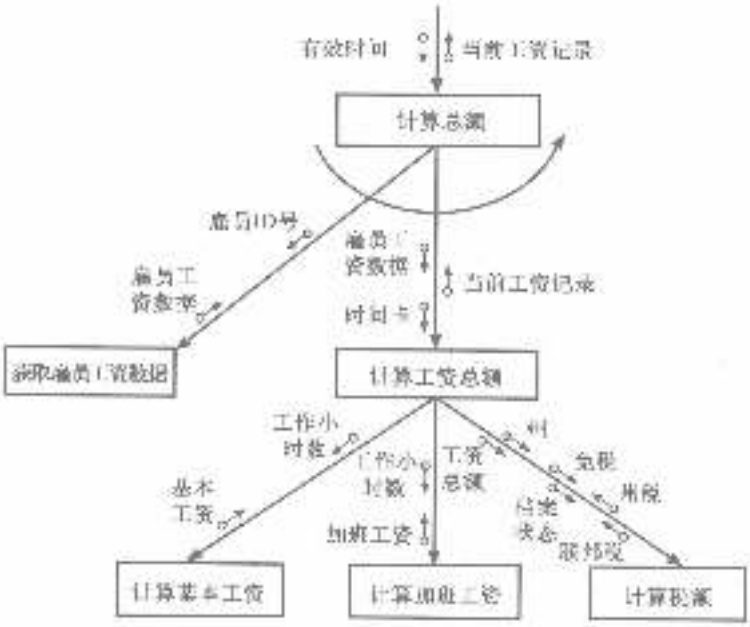


图15-15 计算工资单的部分结构图

图15-16显示的是一个简单的驱动程序，用于测试“计算工资总额”模块。更复杂的驱动模块也许会使用存储在文件或数据库中的成百上千的模块输入数据及其正确输出数据。驱动程序将循环进行数据测试，反复调用“计算工资总额”模块，检查它的返回值与期望值之间的异同。对于任何不符的地方，打印或显示警告信息。

使用驱动程序使得被调用的次级模块能够在编写调用模块之前得到测试。驱动程序在自底向上的开发模式中得到广泛的应用，因为在父模块开发之前，子模块（或方法）已经开发完毕并通过了单元测试。

另外一种用于完成单元测试的测试模块叫做存根程序，可模仿一个尚未开发的被调用模块的行为，“计算工资总额”模块的单元测试要求三个存根程序模块，显示在图15-15中的下部，存根程序是相对简单的模块，通常只有一行或两行的可执行代码。每一个用于测试“计算工资总额”模块的存根程序可以作为一个声明来实现，无论输入什么参数，它都返回一个常数值。图15-17对这三个存根程序模块分别给出了示例代码。

存根程序：一种为了测试而开发可模仿一个尚未开发的模块的执行或行为的模块。

存根程序适用于自顶向下的开发模式。实际上，自顶向下的开发模式经常是从为程序或者类中的每个模块或方法编写存根程序开始的。随着开发过程中每个模块或方法代码的完全实现，这些代码逐渐取代了模块或方法中的存根程序。

3. 集成测试

集成测试是测试一组模块或方法的性能。集成测试的目的是要发现单元测试不能发现的错误，这些错误可能来源于以下一些问题。

- 接口不兼容。例如，一个调用模块传给子模块一个错误数据类型的变量。
- 参数值。模块传入或返回的值是不符合要求的（如价格的值是负数）。

- 运行例外。因为资源需求冲突，模块产生诸如“内存空间不够”或“文件正在使用”的错误提示。
- 意外的状态交互。两个或多个模块相互作用的状态产生了复杂的操作失败（如一个订单类的方法能够处理除了某一个之外的所有可能的客户对象状态）。

集成测试：测试一组模块或方法的性能。

```
module main()

// Driver Module to Test CalculatePayRecord()
{
   //  Declare Module Parameters

   record EmployeePayData   {
      integer EmployeeIDNumber;
      boolean SalaridEmployee;
      real PayRate;
      char[1] State;
      integer FillingStatus;
      integer Exemptions;
   }
   record TimeCard   {
      integer EmployeeIDNumber;
      date StartDate;
      array[7] of real HoursWorked;
   }
   record CurrentPayRecord   {
      real BasePay;
      real OverTimePay;
      real FederalTax;
      real StateTax;
   }

   // Set Input Parameter Values

   EmployeeData.EmployeeNumber=123456789;
   EmployeeData.SalaridEmployee=False;
   EmployeeData.PayRate=32.15;
   EmployeeData.State="NE";
   EmployeeData.FillingStatus=1;
   EmployeeData.Exemptions=5;
   TimeCard.EmployeeIDNumber=123456789;
   TimeCard.StartDate=[3/7]/2006;
   TimeCard.HoursWorked[0]=0.0;
   TimeCard.HoursWorked[1]=0.0;
   TimeCard.HoursWorked[2]=0.5;
   TimeCard.HoursWorked[3]=1.5;
   TimeCard.HoursWorked[4]=9.0;
   TimeCard.HoursWorked[5]=8.0;
   TimeCard.HoursWorked[6]=9.0;

   // Call Tested Module

   call CalculatePayAmount(EmployeeData, TimeCard, CurrentPayRecord);

   // Print Results

   print(EmployeeData, TimeCard, CurrentPayRecord);
}
```

图15-16　用于测试“计算工资总额”模块的驱动程序模块

有一些是非常普通的集成测试的错误，但是还有许多其他可能的错误和原因。

一旦集成错误被检测出来了，就要追究到底是哪个或哪些模块产生了错误。测试执行人员通常也负责确定产生错误的原因。一旦确定了产生错误的模块，就要求这个模块的程序员重写代码以便改正错误。

结构化软件的集成测试是比较简单的，但是不一定容易实施。许多结构化的模块只被一个父模块调用，另外，大多数结构化模块的内部并不存储参数的状态。每次模块被调用时，

其内部变量通常被初始化为相同的值。这些特征使得测试人员（通常使用自动测试工具）能够产生测试用例和数据，来检验被测软件所有可能的控制路径。随着被检测到的控制路径的增加，也越来越有信心检测到隐藏的严重错误。

```
Module CalculateBasePay(HoursWorked,BasePay)

// Stub Module

array[7] of real HoursWorked;
real BasePay;
{
    BasePay=1000.00;
    return;
}

Module CalculateOvertimePay(HoursWorked,OvertimePay)

// Stub Module

array[7] of real HoursWorked;
real OvertimePay;
{
    OvertimePay=175.00;
    return;
}

Module CalculateTaxes(GrossPay,FilingStatus,State,Exemptions,FederalTax,StateTax)

// Stub Module

real GrossPay;
integer FilingStatus;
char[2] State;
integer Exemptions;
real FederalTax;
real StateTax;
{
    FederalTax=175.00;
    StateTax=75.00;
    return;
}
```

图15-17 用于测试“计算工资总额”模块的存根程序模块

与之相反，面向对象软件的集成测试复杂得多，并且不容易理解。在一个面向对象的软件程序中，没有明显的等级结构。面向对象的程序包括一系列相互作用的对象，这些对象在执行过程中产生或被销毁。对象的交互和控制流是动态而复杂的。

使面向对象的集成测试变得复杂的因素还包括：

- 方法可以（而且经常）被许多其他方法所调用，而且，这个调用方法可能分布在许多类中；
- 类可以从其他类中继承方法和状态变量；
- 具体的被调用方法是根据消息参数的数量和类型在运行中动态决定的；
- 对象可以在两次被调用之间保持其内部变量值（即对象状态）不变。对两次相同的调用，对象的响应可能有所不同，其原因是，第一次调用后或两次调用之间的状态变化。

这些因素的综合，使得确定一个理想的测试顺序十分困难，这些因素还使得预测一组相互作用的方法和对象的状态十分困难。因此，对面向对象的软件进行集成测试要比结构化的软件复杂得多。处理这些难题的方法和技术不在本书讨论范围内，可参考“参考资料”中关于对面向对象的软件进行测试的讨论。

系统测试用来测试整个系统或独立子系统的性能。系统测试通常最先由开发者和测试人员来操作，以确保没有明显的故障，确保系统能够满足用户的需求。后期的测试由用户来完成，以确定此系统的确能够满足他们的要求。如果系统被开发时有许多迭代程序，则在每层迭代程序的最后进行程序测试，以确定有意义的事件，如性能问题，必须在下一层迭代时被

处理。

系统测试：确定整个系统或独立于系统的性能。

Build和Smoke测试是日常执行的一种系统测试，系统被完全编译和连接（Build）后进行一组测试，以检查在一个明显的路径上是否存在故障（Smoke）。日常测试通常与迭代和快速并发过程相联系。但是，如果采用自顶向下的开发模式，日常测试也可以用在更为传统方式的项目开发过程之中。

Build和Smoke测试（日常测试）：日常执行的一种系统测试。

Build和Smoke测试是很有价值的，因为它们能够对重大问题做出快速反应。修改或增加代码所引起的任何问题和现象，都将会在日常测试过程中表现出来。日常测试可以确保迅速发现错误，并且能够很容易地找到错误的源头。如果不经常测试，那么这种好处就逐渐消失了，因为当更多代码发生了变化的时候，就越来越难以找到错误的源头。

4. 有用性测试

有用性测试是确定一个模型、方法、类、子系统或者系统是否满足用户需求的测试。因为有许多类型的需求，包括功能性的和非功能性的需求。有用性测试的许多类型在不同的时间来执行。

最通常的有用性测试类型是评估功能性需求和用户接口的质量。和用户相连接的系统部分决定系统是否满足期望的功能和用户接口是否很容易使用。随着用户界面的开发，需要不断地进行有用性测试以提供快速的反馈。这种反馈是为了改进界面和改正任何潜在的软件组建中的错误。

性能测试是一种确定系统是否能够满足时间方面性能要求的系统测试，如相应时间和吞吐量。响应时间需求确定了对查询或更新的软件响应所允许的预期的或最大的时间限制。吞吐量需求确定了每个小时或每分钟内必须处理的预期的或更少的查询和交易数目。

有用性测试：确定一个模型、方法、类、子系统或者系统是否满足用户的需求的测试。

性能测试：确定系统是否能够满足时间方面性能要求的系统测试。

响应时间：对查询或更新的软件响应所允许的预期的或最大的时间限制。

吞吐量：每小时或每分钟内必须处理的预期的或更少的查询和交易数目。

性能测试可以作为单元测试或集成测试的一部分来执行，但更多的是在系统测试中进行。性能测试通常比较复杂，因为它们涉及很多程序、子程序、计算机系统和网络结构。它们要求大量的测试数据来模拟正常负载或最大负载下的系统操作。诊断和纠正性能测试的故障也非常复杂。必须首先找到瓶颈和运行不佳的组建。纠正行为必须包括应用软件调试或重新执行、硬件或系统软件的重新配置以及运行不佳组建的升级与代替。

验收测试是一种系统测试，以确定系统是否满足用户需求。验收测试是在系统移交给用户之前进行的最后一轮测试。在大多数的开发项目中，验收测试是一种非常正式的活动。当一个新系统建立或从外部引进时，验收测试的细节有时存在于RFP和采购合同之中。

验收测试：一种系统测试，以确定系统是否满足用户需求。

谁来测试软件

在测试过程中有许多参与者，他们的数量和所扮演角色是根据项目的规模和其他特点而决定的。具体的参与者包括：

- 程序员
- 用户
- 质量保证人员

程序员一般首先负责测试自己代码的单元测试，然后与其他人的程序模块进行集成。在

一些组织中，会委派一个测试伙伴来帮助程序员测试他们自己的代码。在集成测试之前，程序员也要负责测试其伙伴的代码。由不同的程序员来检查代码，常常能够发现更多的错误。

测试伙伴：负责测试另外一个程序员编写的程序代码的程序员。

用户主要负责β测试和验收测试。当β版本开发完毕后，它们被分发到一组用户手中测试一段时间（几天，几周或几个月）。这些志愿者不是理想的随机用户，但还是经常使用志愿者测试软件。β测试的志愿者要比普通的用户具有更多的计算机知识，并且对故障的容错能力比普通用户好。这些特点导致反馈的信息往往都是高质量的问题，而缺乏对其他小错误的反馈。

验收测试通常是由用户和信息系统的开发或操作人员合作完成的。验收测试的严格和重要性要求其参与者要覆盖非常宽的用户层次范围（从数据录入人员到拥有该系统的管理人员）。尽管信息系统人员能够进行系统安装并能维护系统功能，但最终还是由用户来决定是否接受这个系统。

在一个大的系统开发项目中，通常成立一个独立的质量保证小组或组织。QA小组负责除单元测试和验收测试之外的所有方面的测试。其活动和责任主要包括：

- 制订测试计划；
- 进行集成测试和系统测试；
- 收集和组织用户对于α版本和β软件版本的反馈信息，并且确定系统设计和实施中需要进行的改动。

为了维护其客观性和独立性，QA小组通常直接向项目经理或者长期的信息系统经理汇报工作。

15.3 数据转换

一个可操作系统需要一个充实而易用的数据库来支持正在进行的处理。例如，RMO的订单输入子系统依靠有关目录、产品、客户，以及已经订单的存储信息。实施人员必须确保这些相关信息在订单输入子系统可操作时刻已经存在于数据库中。

系统启动时需要的数据可以从以下来源获得：

- 正被替代的系统中的文件和数据库
- 手工记录
- 组织内其他系统的文件和数据库
- 正常系统操作期间的用户反馈

15.3.1 重用现有数据库

大多数新的信息系统用来代替或增强现有手工或自动化系统。在最简单的数据转换形式中，新系统直接使用旧系统的数据库，稍加改变或不改变数据库的结构。重用现有数据库相当普遍，因为重新创建新数据库的难度和开销很大，尤其是在目前的企业应用系统中，一个单独的数据库通常支持多个信息系统。

虽然旧数据库在新的或升级的系统中被普遍重用，但经常需要对数据库内容进行某些改变。典型的改变包括增加新类或实体和新属性或关系及修改现有的属性或关系。现代数据库管理系统（DBMS）通常允许数据库管理员修改现有的充实且易用的数据库结构。简单的变化，例如增加新的属性或改变属性类型，完全可以利用DBMS来实现。

15.3.2 重新装载数据库内容

对数据库结构的更复杂变化可能要求在变化之后重新装载数据。在那种情形下，实施人

员必须开发程序以便在数据库修改之后改变数据。图15-18显示了重载数据的两种可能方法。第一种方法是先初始化一个新数据库，然后将旧数据库的内容复制到新数据库中，转换程序将存储在先前数据库结构中的数据转换到新修改的数据库结构中。

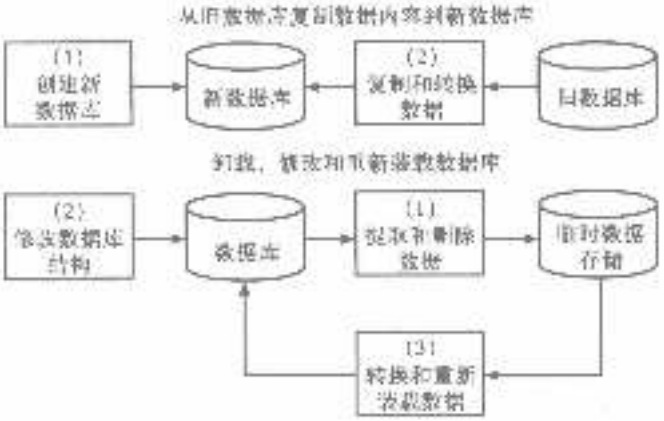


图15-18　结构修改之后重新装载数据库内容的两种方法

第二种方法是使用一个程序或者DBMS功能来提取并删除现有数据库中的数据，然后将其保存在一个临时数据存储中，接着修改数据库结构并利用另一个程序或数据库的功能进行修改后数据库的重新装载工作。第一种方法比第二种方法简单，但是它需要有足够的存储空间以便同时容纳新、旧两个数据库。第二种方法可用于数据存储空间不足以容纳两整套数据的情况。

许多DBMS提供了丰富的导入功能以便从现有数据库、文件或扫描文档中提取和装载数据。DBMS开发者提供这些功能是因为系统开发者更乐意采用具备从其他来源方便导入数据的功能的DBMS。如果DBMS导入和导出功能不足以实现数据转换，那么开发者必须构造一次性使用的转换程序。虽然转换程序并不是可操作系统的组成部分，但是它们必须以等同于可操作软件的形式进行构造和测试。

15.3.3　创建新数据库

如果正被开发的系统是全新的或者要替代一个手工系统，那么初始数据来源于手工记录或者来源于组织内其他自动化系统。来自手工记录的数据的输入可以使用正在为可操作系统开发的相同程序来进行。在这种情况下，数据输入程序通常需尽可能早地开发和测试。初始数据输入可以作为用户培训练习。另外，来自手工记录的数据还能被扫描到光字符识别程序中。接着利用数据转换程序或DBMS的导入功能将数据输入到数据库中。

某些数据已经被存储于组织内的其他自动化系统中。例如，当实施一个新的订单输入系统时，有些产品数据可能已经存在于一个加工计划和控制系统中，并且有些客户数据可能已经存在于一个现有的记账系统中。将这些数据复制到新系统中类似于从旧数据库重新装载数据到修改的数据库中或者备份数据存储。

图15-19显示了一个复杂的数据转换过程，它从各种数据来源中提取输入数据，利用手工输入、光字符识别、转换程序的混合方法进行数据输入。这种复杂程度的数据转换过程经常在一些大型开发项目中出现。

在某些情形下，可能系统操作起始于部分或全部为空的数据库。例如，一个客户订单输入系统不必要求现有客户信息事先装载到数据库中，客户信息可以在其第一次下订单时或基

于电话订单输入人员与客户之间的一次对话的信息增加到数据库中。这种遵循式增加数据的形式降低了数据转换的复杂度，但代价是初始事务处理变慢。

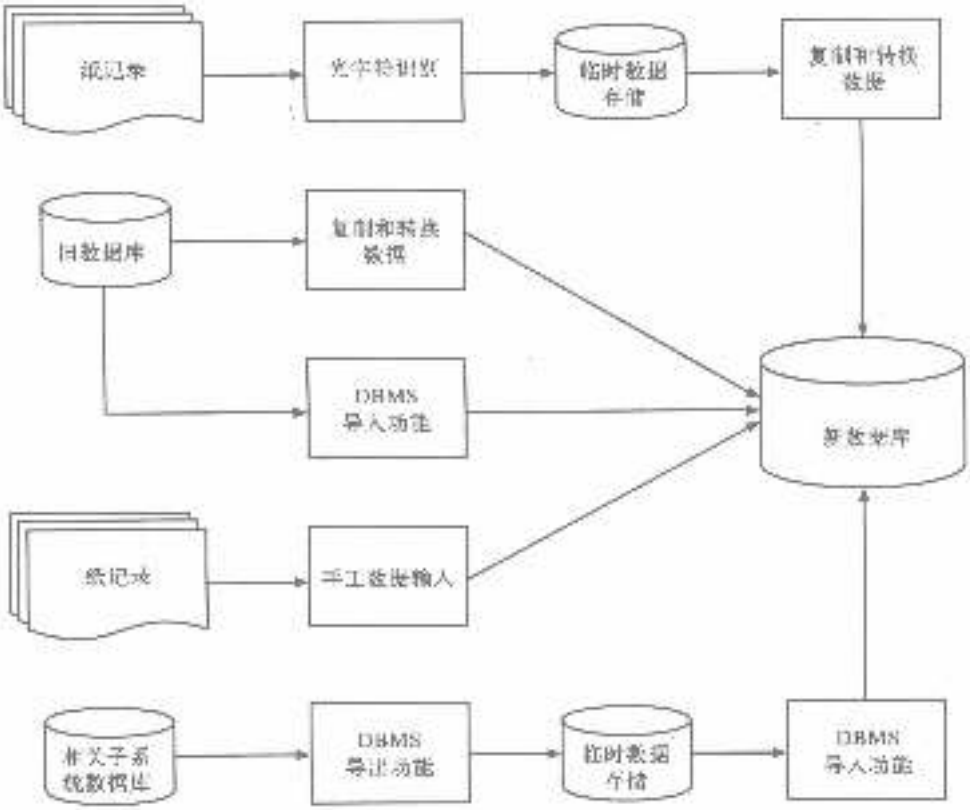


图15-19 复杂数据转换举例

15.4 安装

一旦一个新系统开发和测试完毕，必须安装和付诸运行。安装系统并使之运行也是一个复杂的问题，这是因为存在着许多冲突限制，包括费用、客户关系、雇员关系、复杂的后勤及全部风险。在计划安装时，有以下重要问题需要考虑：

- 并行运行两个系统所带来的开销
- 对于新系统的检错与纠错
- 可能会扰乱公司和它的信息系统正常运作的潜在因素
- 对人员进行培训并使客户熟悉新程序

不同安装途径代表着不同费用、风险和复杂度的交易方式。最常用的安装途径包括：

- 直接安装
- 并行安装
- 阶段安装

每一种途径有不同的优点与弱点，但没有一种途径是适用于所有的系统的。每种方法详细描述如下。

15.4.1　直接安装

在直接安装中，安装新系统并使系统快速进入运行状态。当有重复的系统时，立即关闭。直接安装有时也叫做立即接入。当新系统安装测试完后，新旧两个系统只同时运行很短的一段时间（通常是几天或者几周）。图15-20显示了直接安装的时间进程。

直接安装或立即接入：一种使系统迅速运行起来的安装方法，并要立即关闭任何重复的系统。

直接安装的主要优点是它的简单性。虽然新旧两个系统不能并行运行，那么就没有太多的后勤管理问题，耗费的资源也少得多。直接安装的缺点是它的风险性，因为旧系统不能并行运行，所以在新系统运行失败的情况下就没有备份。风险的大小取决于系统的特性，在系统失败时工作区的耗费及系统不能使用或非最佳状态所带来的损失。

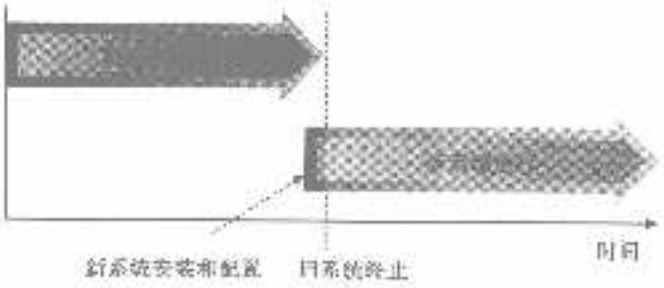


图15-20　直接安装和接入

直接安装典型地用在以下一种或两种条件下：

- 新系统还没有代替旧系统
- 可以忍受停机几天或几周

如果不满足以上的任何一个条件，就采用并行安装和分阶段安装的方法，以最大限度地降低风险。

15.4.2　并行安装

并行安装可以使新旧两个系统在很长的一段时间内同时运行（数周或数月）。图15-21说明了并行安装的时间进度。旧系统将一直运行到新系统被全面测试，确信无错和可以独立运行为止。从实践看来，并行运行的时间通常是预先确定好的并且是有限制的，以便尽量减少双系统运行带来的耗费。

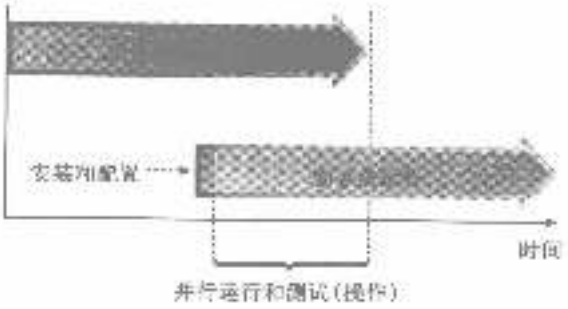


图15-21　并行安装和运行

并行安装：一种使新旧两个系统在很长的一段时间内同时运行的安装方法。

并行安装的主要优点是系统失败带来的风险和副作用较小。如果两个系统同时运行（即用所有的数据并训练所有的函数），那么旧系统可以作为新系统的备份，新系统的任何失败所造成的损失都可以从旧系统中得到恢复。

并行安装的主要缺点是资源耗费大。在两个系统同时运行时，要为两个系统都分配资源。在并行运行中，会带来许多与之相关的额外开销，包括：

- 雇佣临时员工（或者给已有人员增加临时任务）
- 为计算机设备和员工增加额外空间
- 增加了管理和后勤工作的复杂度

除非新系统的运行开销确实比旧系统少，否则两个系统同时运行的花费是单独运行旧系统的2.5～3倍。

当系统失败产生的后果较严重时，采用并行安装的方法是最好的。并行安装确实降低了系统失败带来的风险，这种风险的减少对“苛求的任务”是非常重要的，如客户服务、生产控制、基本财务功能和许多形式的联机交易等。很少有组织能够承担重要系统停止运作所造成的损失。

对于一些情况来说，实行系统完全并行运行也许是不切实际的。原因包括：

- 一个系统的输入数据对另外一个系统可能是不可用的；
- 新系统可能与旧系统使用同样的设备（如计算机、输入/输出设备、网络），这样有可能造成新旧两个系统都得不到足够的运行资源；
- 没有足够的操作和管理人员来同时运行两个系统。

如果完全的并行运行是不可能的或不可行的，那么可代之以部分并行运行。运行并行模式可能的部分包括以下几种：

- 在其中一个系统中只处理输入数据的一个子集，子集可由事务类型、布局或抽样（如每10个输入事务进行一次抽样）决定；
- 仅仅运行处理功能的一个子集（如，刷新账目的历史记录，但是并不打印出每月的账单）；
- 可综合运行数据和功能子集。

部分并行运行总是伴随着可能存在未被检出的严重错误或问题的风险。例如，带有部分输入的并行运行增加了检测不出那些和未经测试的数据有关的错误的风险。

15.4.3 阶段安装

在阶段安装中，系统被安装并在一系列的步骤或阶段后投入运行。每一阶段都为运行的系统增加一些组件或功能。在每一阶段中，系统都要经过测试以确保为下一阶段做好准备。分阶段安装可以和并行安装结合使用，尤其是在新系统将要取代现有系统的运行阶段。

阶段安装：经过一系列的步骤和阶段使新系统投入运行的安装方法。

图15-22显示的是在独立的阶段存在直接和并行安装的阶段安装。新系统取代了两个现有系统。安装分为三个阶段。第一阶段是直接取代其中的一个现有系统。第二和第三阶段是取代另一个现有系统的并行安装的不同部分。

实施阶段安装的方法不是唯一的，诸如具体阶段的组成及它们的安装顺序等安装细节随着系统的不同而有很大的差别。这些具体情况决定了安装过程要划分的阶段个数、安装顺序，以及新系统与现有系统并行运行的部分。

阶段安装的主要优点是降低了风险。风险降低的原因是，单个阶段的失败带来的问题要比整个系统失败所带来的问题少。阶段安装的主要缺点是增加了复杂度，将安装分为几个阶

段会产生更多的工作和里程碑，从而使得整个过程更加复杂，每一个单独的阶段又存在许多更小的问题需要解决。如果系统是因为太大或太复杂而不能一次完成安装，那么采用分阶段安装降低风险的意义就显得更为重大。尽管这种安装会使得管理和协调多个阶段的复杂度有所增加。

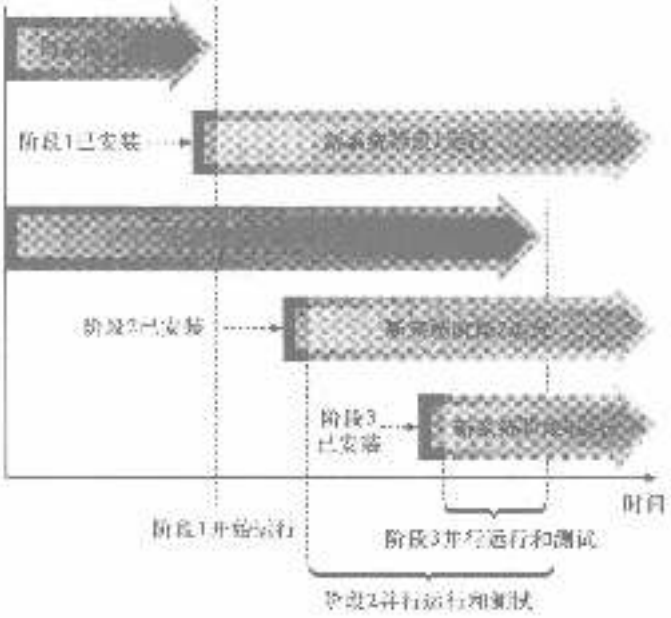


图15-22　带有直接接入和并行运行的分阶段安装

当一个系统很大，很复杂并且由一些相对独立的子系统组成时，分阶段安装的方法是非常有用的。但如果子系统不是相对独立的，那么定义一些独立的安装阶段就会很困难甚至是不可能的。系统的规模和复杂程度可能太大以至于不能够"立即完全"安装。在这种情况下，别无选择，就只能使用分阶段安装的方法了。

实践指导

为了减少风险，只要有可能就结合阶段和并行安装。 ■

15.4.4　人员问题

安装一个新系统对整个组织机构的人员安排有较高的要求。安装要求有一个时间安排，要求快速学习并适应，有较大的压力。在制订计划时，应该预料到这些问题，并且采取适当的措施来缓解这些问题可能带来的影响。

新系统的安装过程使信息系统的人力资源得到了最大限度的利用，许多任务必须在很短的时间内完成。这个问题在并行安装时显得尤为明显，因为必须操作新旧两个系统。在通常情况下，开发方和用户方的支持人员必须临时重新分派任务以提供充足的人力资源去运行两个系统。但重新分派的行为将会减慢别的正在进行的项目的进度，同时减少了对其他系统的支持与维护活动。

在安装过程中，必须雇佣一些临时的合同工，有两种人员是特别有用的：

- 在软硬件的安装和配置方面有经验的人员
- 对旧系统的运行有经验的人员（可以接受培训）

安装所需要的技术在组织机构内是比较短缺的，在这种情况下，雇佣一些软硬件人员投入到安装工作是很有必要的。软硬件专家可从人才市场或信息系统咨询公司找到并签约。

在并行安装过程中，雇佣一些临时人员有以下几点好处：第一，提供了运行两个系统所需要的人力资源；第二，解放了正式职员，使其从事培训和新系统的运行方面的工作。临时人员一般提前几个月雇佣，培训他们操作旧的系统，如果新系统出现了问题，那么可以延长这些雇员的合同期，直到旧的系统被安全替代为止。

另外一个必须考虑的人员问题是雇员的生产率。所有的新系统对于用户和系统操作人员来说都有一个学习过程，在安装之前都要经过培训，但不论培训是多么成功，用户和系统操作者在系统安装完后必须经历一段时期（通常是几个月）才能达到最佳熟练程度。在那段时间内，对人力资源的要求将比较高，员工的压力就比较大。

15.5 文档

在系统实施过程中，准备文档的工作很重要，但也经常被忽略。文档告诉用户如何操作和维护一个系统。文档还提供了关于未来改进与重新实施所需的信息。

在最近的几十年里，系统文档和用户文档的性质都有重大变化。这些变化是由技术的快速进步引起的。在20世纪80年代之前，多数文档打印在纸上并被装订起来或者用活页夹夹起来。现在的标准是自动文档，自动文档的形式包括：

- 电子手册，例如Microsoft Word文档或Adobe Acrobat文档
- 超链接文档，例如网上浏览用的文档格式，这些文档是嵌入链接到网上的
- 联机帮助——说明性的文本、图片、导航和嵌入在应用程序中的定义
- 嵌入式文档，例如手册、指导书以及保存在一张CD上的多媒体演示介绍并且作为应用整体的一个部分
- 电子系统模型——格式化并存储于图片文件中的文本或图片格式，例如GIF、JPEG、和Visio
- 专用工具系统模型——用系统开发工具开发的模型，例如集成开发环境、DBMS和CASE工具

可以将电子手册分布于一些标准的格式如Adobe Acrobat和Windows帮助文件中。选择一种标准格式以便确保用户能够用适当的软件查阅所需文档。超链接文档允许用户在相关的主题间快速浏览。大多数的电子文档格式允许超链接，使用网页的格式可以使许多用户共享同一份文档拷贝。网页形式使得对于文档的改变变得更加简单，因为它只需要更新服务器上的文档即可。嵌入式文档使用户能够在应用期间访问信息并提供上下文敏感帮助等特征。

电子和专用工具系统模型主要适合于软件开发人员使用，可以将一般的模型格式（如普通文本和GIF图像）格式化为任何类型的电子格式。专用工具模型必须由特定的软件工具来支持（例如，由CASE工具产生的模型就要求在CASE的环境下浏览）。然而，有一些开发工具允许其模型输出为其他格式（例如，Acrobat和Microsoft Word）。

文档可被大致分为以下两种类型：

- 系统文档——描述系统功能、结构及构建细节；
- 用户文档——描述用户如何使用和维护系统。

系统文档：描述系统功能、结构和构造细节，用户是维护人员和未来的开发人员。

用户文档：描述如何使用和维护系统，用户是终端用户和系统操作人员。

系统文档产生于整个SDLC，是每个生命周期阶段和活动的结果。用户文档产生于SDLC的实施阶段，开发小组不可能很早就编写出用户文档，因为用户界面和系统操作的许多细节

还不确定，而且这些细节在开发的过程中还可能发生变化。

15.5.1　系统文档

系统文档的一个主要作用是，为设计和开发人员提供相关信息来维护系统或对系统进行重新实施。由于这个原因，多数或者所有文档都是随着分析、设计和实施的活动产生的。图15-23显示了SDLC的三个阶段及每个阶段文档的产生和修改情况。虽然后期的文档要比早期文档的使用频率高，但是每个阶段的文档对未来的维护和升级都是十分有用的。

生命周期阶段	系统文档	
	传统方法	面向对象的方法
分析	实体-联系图 数据流程图 过程描述 数据流和元素定义	类图 用户情况 活动图 顺序图
	事件列表	
设计	系统流程图 结构图	设计类图 界面图 协作图 软件包图 状态图
	模块或程序的伪代码 数据库规划图	
实施	程序源代码 数据库规划说明 测试数据	

图15-23　生命周期阶段和每个阶段产生的系统文档

源代码是最经常使用的文档，因为它与系统的可执行软件直接相关。直接对二进制代码进行改动是非常困难且代价昂贵的，所以为了改变系统的功能，修改源程序代码并进行重新编译是唯一比较现实的方法。系统改变后，需要测试数据对其进行测试，系统的一部分发生改变后，用旧的测试数据重新进行测试，有助于确定这些改变是否会损害系统的其他部分。

使用源代码作为文档是很困难（而且效率较低）的，因为它是全文本化的，并且不好理解。一些重要类型的系统信息——诸如程序如何交互和程序要满足用户的什么需求——通常无法在源代码中体现出来。在评价一个系统设计或功能上的重大改变，以及追踪经过共享数据从一个程序传递到另一个程序的错误时，这样的信息是很有用的。在分析和设计模型中可以得到执行这些任务所需的信息。

设计模型比分析模型用得多一些，因为设计参数的变化比系统需求要多。例如，需要涉及模型而非分析模型的维护的变化，包括在新硬件上重新使用已有的程序或数据库、单个程序纠错、对已存在的分布式系统性能的优化。这些变化都会引起设计模型（例如系统流程图或包图）的相应变化，而不会影响分析阶段的模型。

当用户需求发生变化时，分析模型随之改变。例如，增加一个新的事务或新的子系统，将会改变数据流图和类图。其他的分析阶段的模型诸，如实体-联系图、事件列表、用例也会发生变化。

系统文档需要进行积极的管理，它必须以易于访问的位置和形式进行存储，以便在需要

进行系统维护时可以方便地检索到，或者一旦发生变化便于更新。在一个有许多大型信息系统的组织中，文档管理是一个非常正式的过程。在大的组织当中，有专门的人员负责文档的管理与恢复以及加强文档标准化的工作。

系统文档维护失败会使系统的价值遭受损害。文档不充足的系统是难以维护的，进而会增加系统过早老化和需要重新实施的可能性，维护文档可以延长生产资本的使用期。

系统文档对于系统本身来说是多余的，任何系统文档所包含的内容在对系统的检查中都能得到。例如，程序员通过检查描述数据库的SQL语句来确定关于数据库的实体和联系。程序员通过检查程序源代码，可以确定传统程序中的模块结构或面向对象程序中的类，即使得不到程序源代码，也可从可执行代码中确定程序结构，尽管这个过程非常复杂。

随着系统的变化，其文档也必须随之更新，如果文档不更新，那么系统就会出现不一致，而且这些文档对将来的设计和维护程序员来说没有什么用处。将文档集成到系统的安装之中，可以减少这种不一致性，因为当系统升级后，系统能够自动地更新这些文档。一些工具——尤其是CASE和反向工程工具——能够简化文档并有助于确保其准确性。

使用CASE工具，系统可以从设计模型中自动建立系统，并且由CASE工具来保存。设计模型在分析模型的基础之上也会自动建立，若要对一个系统做出修改，程序员需修改分析和设计模型，然后重新生成已安装的系统。CASE工具会自动维护这种存在于已安装的系统和修改后的模型之间的一致性，因此，如果只是模型发生了改变（而不是源代码或可执行代码），模型和系统总能保持一致。

反向工程工具可以从源代码中产生系统模型。例如，这样的工具可以从面向对象的程序中生成类图，也可以利用过程化语言编写的程序生成结构图。如果反向工程工具功能强大和具有高可靠性，足以产生所有类型的系统文档，那么就没有必要维护一个分离存储的文档了，源代码本身就是文档，利用反向工程工具可根据要求生成各种形式的文档。

CASE工具和反向工程工具是高度专业化的工具，对操作环境（例如，编程语言、数据库管理系统和操作系统）有专门的要求。它们一般价格昂贵，并需要一段时间的学习，导致了它们并不像想象中那样常用。因此，对于许多系统来说，还是必须单独维护系统文档。

15.5.2 用户文档

用户文档对最终的用户提供系统运行支持。它主要描述了系统操作的规程，其中包括一些功能，如数据录入、产生输出和定期维护。其主要内容如下：

- 软件启动和关闭；
- 执行一个特定功能时的按键、鼠标、输入的命令序列；
- 实现一个特定的事务处理所需的程序函数；
- 常见错误和处理方法。

为了方便使用，用户文档包括内容表、程序或系统的目标或功能的大体描述、术语表和索引。

现代系统的用户文档几乎都是电子版本的，并且通常是集成到应用程序之中。大多数的现代操作系统都提供标准工具来支持嵌入式文档。图15-24显示的是在一个典型的Windows应用程序中的电子用户文档。左边的窗格显示了内容表，并且能够通过单击屏幕上方合适的选项卡来访问一个索引或搜索引擎。右边的窗格显示了用户文档的单个网页。

用户文档是非常重要的组织资源。遗憾的是，许多组织在其内部开发系统中缺乏详细的高质量的用户文档。其原因如下：

- 认为经过培训的程序员能够检查源代码，确定系统是如何工作的，并且在需要的基础

上培训用户；

- 认为在系统的实施过程中接受过培训的用户能够非正式地将他们的知识教给未来的用户；
- 资源和专门技术需要开发文档，而且它们要保证是最新的。



图15-24　Windows帮助显示样本

正如前面一节所述，源代码是一种较差形式的系统文档。基于类似原因，甚至可以认为它是一种糟糕的用户文档。源代码提供了一个系统如何一条指令接一条指令地进行工作的细节，但它不能提供有关系统各部分的交互情况和整个系统在具体的环境中如何运行的信息。在源代码中补充其他形式的系统文档，可以有效地改善这种状况。但是即使这样，仅仅根据系统文档来弄清系统工作的方法较慢且容易出错。

如何使用系统的知识和系统本身一样都是非常重要的，最初的培训结束后，这些知识便存在于用户的脑海中，但是并不能保证这些知识能够得到维护或有效地传递给后来的用户。雇员的流动、重新分配任务和其他一些因素，使得人与人之间有关操作知识的直接交流越来越困难，并且这种行为并不一定会发生。与此相反，书面方式或电子方式的文档则显得更容易得到而且更持久。

要开发好用户文档要求有专门的技能和相当的时间和资源。在要求高但资源有限的情况下，开发文档的技术就要求语言清晰简洁，制作出的演示图片令人印象深刻，组织的信息便于学习和获得，并且与非专业人员进行的交流有效。开发工作耗费时间，只有通过复审和测试过程才能得到良好的效果。遗憾的是，准备用户文档的技术人员常常缺乏一种或多种必要的技能。由于工作进度超标或者忙于实施完工，演示时间、复审和测试经常被忽略或敷衍了事。

15.6　培训与用户支持

好的文档有助于简化所需的培训，也有助于降低对支持请求的频率。但是一些安装前培训和安装后的支持活动还是很需要的。记住，用户也是系统的一部分！没有培训，用户掌握起操作知识就会很慢，出错率会很高，系统的使用效率也会降低。培训可以使得用户在系统投入运行之后很快成为高工作效率的熟练用户，支持活动保证用户在系统安装之后长时间内保持良好的工作效率。

这里用户可分为两种——终端用户和系统操作员——他们的任务是编制文档、培训和支持活动。终端用户每天都在使用着系统的功能，系统操作员执行管理功能并负责系统正常运行的日常维护工作。表15-25显示了系统中每个角色所从事的具有代表性的活动。在一个小型系统中，一个人可能会同时扮演两种角色。

终端用户活动	系统操作员活动
创建记录或事务	修改数据库内容
生成报表	备份数据库
导入或导出数据	启动或关闭系统
查询系统状态	备份数据存档
恢复存档数据	软件安装或升级

图15-25 终端用户和系统操作员的典型活动

培训和支持活动随目标客户的不同而不同，影响培训活动的客户特点包括：

- 系统的使用频率和持续使用时间
- 需要了解系统事务的运行环境
- 现有的计算机技能和熟练程度
- 用户数量

一般情况下，终端用户频繁而长时间地使用系统，而系统操作员不经常与系统交互，往往交互时间也较短。终端用户使用系统是要解决特殊的事务问题或实现具体的事务过程，而系统操作员通常是计算机方面的专家，但对此系统所支持的事务过程的了解有限。终端用户的计算机技能水平差别很大，而系统操作员的计算机水平一般都很高。终端用户的数量远比系统操作员的数量大得多。

对终端用户的培训必须强调实用性而且是在具体的事务环境中运行系统。例如，订单输入，库存控制或财务等。如果用户对那些过程还不熟悉，那么培训内容必须包括这些内容。差异较大的用户技能水平和经验要求培训活动要包括实践练习、答疑和单独指导。自学培训材料能够满足一些需要，但是复杂系统一般至少需要几次面授培训。对于较大数量的终端用户来说，成组培训的方式是可行的，也可以先培训一些素质较高的终端用户，再由他们对其他用户进行培训（即对培训人员进行培训）。

当系统操作员不是终端用户时，对系统操作员的培训可以采取非正式方式。经验丰富的计算机操作员和管理员能够通过自学获得他们所需的知识，因此不必进行正式的培训。当然，如果需要，对少量的系统操作员也可采取单独的面授方式。

决定正式培训的最佳开始时间可能是比较困难的。一方面，相对较早地在实施阶段开始培训活动可以保证用户有足够的学习时间，但另一方面，较早地开始培训活动也可能会使用户和培训者都受到挫折，因为此时的系统不稳定也不完整。对于终端用户来说，试图掌握一个在不断变化、可能出错或崩溃的系统是非常容易遭到挫败的。

理想的情况是，系统的交互界面已经完成、安装好测试版本并且排除故障以后，再进行培训活动，到了项目结尾的关键时刻，培训的耗费较大，通常都要做出很大牺牲。培训材料一般都是在交互界面比较合理稳定时被制订出来，用户培训在此之后也会尽早开始。如果交互界面尽早开发，并且采用自顶向下的模块开发方法，那么就可以尽早进行培训工作。

进行中的培训和用户支持活动

用户支持包括系统安装并运行后的培训和用户帮助。这些活动中的某些部分同安装前的培训活动一样。例如，对新用户的培训活动必须根据人员调整周期性地进行。其他活动如进修培训和帮助台操作是单独进行的。

有很多方法可以提供用户支持，包括：

- 联机文档和问题解答
- 常驻专家
- 帮助台
- 技术支持

近几年，联机文档和问题解答成了一种流行的支持方法，越来越多的网站提供这些支持。这种支持大都集成到应用中。联机支持的目的是对人力支持的需求最小化，在需要的时候就可以得到联机支持。要达到这种目标，要求支持文档综合性强并要易于使用。

常驻专家是最常用的用户支持方式，他们的支持活动一般是非正式的。常驻专家一般是信息系统人员、业务人员，为其他用户提供技术支持的用户。常驻专家的职位通常都是非正式的，只要其显示出高超的计算机技术水平和软件知识，很快就可以成为常驻专家。随着时间的推移，所有其他用户遇到问题或难题时，都会首先向他请教。

帮助台是信息系统的永久组成部分，它为终端用户提供大范围的软件系统支持。负责帮助台的人员是经过安装、运行和解决应用软件（包括经常使用的商用软件如Word）问题培训的。一个帮助台往往是系统与用户的中心接触点，负责帮助台的人员要处理大量的用户问题。但那些需要进一步帮助的人就得求助于技术支持了。

在信息系统维护中，技术支持一般作为专门的功能或部门，它属于维护范畴，因为用户支持、变化要求与系统错误报告之间的关系很密切。如果帮助台负责人员不能解决用户的问题，那么说明发现了错误或者在用户需求和系统能力之间存在差距。如果出现的问题是系统错误，那么维护人员应立即查找原因并改正错误。不严重的错误和未能满足的用户需求都要引起维护人员的注意。在任何一种情况下，技术支持都是用户和维护活动之间的桥梁。

15.7　维护和系统增强

电子电器工程师协会（IEEE）和美国国家标准协会（ANSI）把软件维护定义为对软件产品的修改。修改至少要完成以下内容中的一项：

- 纠错
- 改善软件性能或其他属性
- 使软件适应变化的环境

软件维护：在软件发布之后为了纠错，改善软件性能和属性或使软件产品适应环境变化而对软件进行的修改。

维护实际上包括软件交付使用后对系统进行的所有修改活动，除非软件被完全取代或舍弃。

在大多数组织中，用于对现有系统维护的费用至少与开发一个新系统的费用一样多。现有系统是组织的财富，必须积极地进行管理，保持它的价值和实用性。从这个意义上讲，软件维护与其他类型的资产（如设备）维护是相似的。

维护涉及一些改动——为了适应新环境，为了满足用户的新要求，也为了解决出现的问题。但是对软件进行改动是有风险的，对一个处于运行状态的系统做出改动要比对处于开发状态的系统做出改动困难得多。当一个改动导致一个处于开发状态的系统崩溃时，并不会有用户向帮助台请求支持，也不会立即产生经济上的影响。但是这种改动如果出现在一个处于运行状态的系统中，立刻就会对用户、客户和组织造成很大的影响。

处于运行状态的系统如果无法修改会造成灾难性的后果，因此，软件维护与新系统开发大不相同。新系统的开发一般发生在相对开放的环境中，这时是希望有变动的，可以为外尝试

新的方法，也能容忍系统中存在的一些风险。相反，维护活动一般比较保守，只允许必要的改动，强烈反对有风险性的行为。

维护活动包括：

- 跟踪修改需求和错误报告
- 实施一些改动
- 监视系统性能，并实施一些改动以提高系统的工作能力或改善系统性能
- 升级硬件和系统软件
- 更新文档以反映出维护活动所做的改动

维护与新系统开发包括许多相同的活动，包括分析、设计、构建、测试和编写文档等。然而，对于这些活动的实施在许多方面又不相同，包括范围和细节、触发事件和实施上的限制。每一种维护活动在以下章节中会予以详细描述。

15.7.1 提交改动申请和出错报告

为了控制因改动带来的风险，大多数组织对所有运行的系统采取正式的控制程序。设计正规控制是为了确保在实施之前，能够充分地描述、考虑和规划潜在的变化。典型的改动控制程序包括：

- 标准的改动申请表
- 改动控制委员会对申请进行复审
- 设计和实施的广泛计划

图15-26显示的是一个改动申请表格，是由用户或系统所有者填写的，并把它提交到改动控制委员会进行审议。改动控制委员会审议这些改动申请，并评估对现有硬件、软件、系统性能、安全和操作预算的影响。改动控制委员会的建议以如图15-27所示样式记录下来。被批准的改动添加到预算、规划和实施等问题待定的列表中。

改动请求				
请求日期	2/1/2004		☐	纠错
请求人	Wen Hoa Chang Comptroller	改动类型	☒	修改
目标系统	Customer Accounts Refunds		☐	新功能
改动（或错误）描述 根据最近公布的联邦法律，美国支票格式有所改动，新的格式上在现有路线号码的右侧为安全条码按验数据保留一个位置。 法律要求新的检验数据要自2005年1月1日后（包括1日）起在支票上打印出来。 我们目前利用的一块区域要求打印彩色安全符号。这种安全符号将会按移动成去除，并将增加安全条码与验数据。				
改动请求ID	2002.11	审查日期		
接收时间	2/2/2004		2/7/2004，0930-1100	
审查成员	W Chang (Comptroller),R. Brooks(IS Operations), J. Hernandez (IS Security) G Weeks (IS Change Coordinator)			

图15-26 改动申请表样例

可以利用标准的改动申请表对系统存在的错误提出报告，但是许多计算组织因为需要立即纠错而采用不同的程序和方式。缺陷报告有多种来源，包括终端用户、计算机操作人员或者信息系统支持人员。缺陷报告一般提交给一个人或组织以便记录日志并采取进一步的行动。

改动审查			
改动请求	2007.11	审查日期	2/7/2007
优先级	☐ 首要	☒ 必要	☐ 可选
需要验证现行打印机的能力，验证其是否可在要求区域内打印出安全条码			
续的要求 需要修改数据库以存储带有其他检验信息的安全条码 与支票程序必须修改，以便能够产生能打印安全条码			
性能要求 无			
运行预算 无			
其他要求 无			
处理	☒ 赞成	☐ 反对	☐ 弃权
原因			
最近实施日期	12/31/2007		
再评估日期	n/a	签字	

图15-27　改动审议表样例

15.7.2　实施改动

改动的实施就是一个缩小了的系统开发周期形式，要进行许多同样的活动，尽管在范围上有所缩小或取消了某些活动。实质上，维护改动是一种进一步的项目开发过程，因为这时用户和技术需求更进一步地得到理解。因此，一般不必再实施分析阶段的活动。

改动计划包括以下活动：

- 确定系统的什么部分需要改动
- 保护实现改动所需的资源（例如人员）
- 安排设计和实施活动
- 为改动后的系统制订测试标准和测试计划

设计、开发和操作人员要复审文档以确定更改的范围，对现存系统的测试标准和计划也可作为测试新系统的出发点。改动或增加功能后只需更新测试计划。修改后的设计与测试计划和数据一起存档，留待以后改动工程时使用。

如果改动相对简单，设计可以与规划相结合。对复杂的改动，要有一个独立的设计阶段。为了实施提出的改动计划，如果有必要，可以对现存系统设计进行评估和修改。修订后的设计和测试计划，数据一起存档，留待以后改动工程的时候使用。

实施活动一般是在运行系统的拷贝版本上进行的。产品系统指的是用户天天使用的系统版本，而测试系统指的是实施改动后的产品版本的拷贝。测试系统的开发和测试可在单独的硬件或冗余系统中进行。测试系统只有在经过完全和成功的测试后，才能成为可运行的系统。

产品系统：用户天天使用的系统版本。

测试系统：实施改动后的产品版本的拷贝。

实践指导

在测试系统部署到一个产品系统之前，测试所有的变化。 ■

15.7.3 计算基础结构的升级

计算机硬件、系统软件和网络必须进行周期性的升级，原因包括：

- 软件维护版的发布
- 软件版本的升级
- 系统性能的不断下降

像应用软件一样，系统软件（例如操作系统和数据库管理系统等）必须周期性地纠错和添加新功能。系统软件开发者一般在一年之内发布几次维护版本。近年来，维护版本的发布频率有所增加，部分是基于Internet的软件发布，在一些情况下（如检测病毒软件和操作系统安全子系统），每周升级一次或者更为频繁。

像系统软件内部产生改动一样，系统软件升级会有风险。当软件升级后，原来与旧系统协调运行良好的应用程序软件也许会出错。因此，系统软件升级版在应用到操作系统上之前，要经过周密的测试才行。在许多情况下，为了减少风险，维护和版本升级被忽略了。升级并不会立即带来好处，除非与系统软件相关的错误经常出现。操作系统的维护通常遵循这样的原则："如果没有动静，就不要管它"。

实践指导

如果一个操作系统没有故障，就不要去装它。 ■

一般，为了增加系统容量或提高系统性能才对基础结构进行升级。事务处理容量的增加或使现有硬件和网络支持新的系统可能会使系统性能降低到令人难以接受的程度。基础结构升级与其他改动的实施是类似的，主要差别是如何使系统性能得到升级。

用户或信息系统人员的输入信息可能暗示着系统性能需要升级，但是要经过全面的调查研究才能确定是否有必要升级和到底哪些部分需要升级。计算机和网络的性能是复杂而高技术的，看似是性能的问题有可能与硬件和网络的性能无关。如果问题的原因是硬件和网络的原因，那么必须确认具体问题并且选择适当的升级方式。

性能问题需要有复杂的诊断确定一个最佳途径来找到问题的根源，应该由具有丰富技术背景并且了解所有相关过程的人员来做诊断。大型信息系统组织的职员有可能有这样的技术，但是许多组织必须依靠合同人员或顾问来诊断这些性能问题、推荐修改方案并安装或配置这些工具和方法。这种技术代价很高，但却可以防止组织浪费大量的资金去购买一些并不真正需要的硬件或网络资源。

小结

在设计之后和系统交付使用之前的系统开发活动一般称做实施，实施活动很复杂，因为它包含了许多相互依赖的活动，包括编程、质量保证、硬件和软件安装、文档编制和培训。实施活动难于管理，因为必须对活动进行适当的排序而且必须不间断地监视活动过程。实施活动是存在风险的，因为它需要大量的时间和资源，而且它对系统的影响关系到一个组织的日常运作。

编程和测试是两个相互依存关系最强的实施活动，必须按照使开发资源最少并且检测系统与纠错的能力最大的规则来构建软件组件。遗憾的是，这两个目标通常是有冲突的。因此，一个程序开发计划是可用资源、可用时间与在系统安装之前进行检测和纠错的愿望之间的一

种折中方案。

数据转换、安装、文档和培训通常是在程序开发之后的活动。它们之间是高度相互依存的，互为一个安装了的、带有文档的系统是完成培训的先决条件，并且系统启动需要有一个充满内容的数据库。人力资源的利用和直接受影响的人员数量，通常在这些活动中达到顶峰。

支持活动发生在系统投入使用以后。支持活动帮助用户实现对整个系统的充分利用。维护和系统增强活动确保系统以最高的效率工作，而且确保在最小程度扰乱组织的情况下实施所需改动。对大多数系统来说，用于支持和维护活动的资源多于用于开发系统的资源。由于对于资源的高度需求和更大的操作风险，支持和维护的实施通常很正式并且需要精心管理。

关键术语

acceptance test	验收测试
alpha version	α版本
beta version	β版本
bottom-up development	自底向上开发
build and smoke test	日常测试
chief developer team	首席开发者小组
collaborative specialist team	协作专家小组
cooperating peer team	平等合作小组
direct installation, or immediate cutover	直接安装，或即时接入
driver	驱动程序
input, process, output (IPO) development	输入、处理、输出（IPO）开发方法
inspection	检查
integration test	集成测试
maintenance release	维护版本
parallel installation	并行安装
performance test	性能测试
phased installation	分阶段安装
production system	产品系统
production version, release version, or production release	产品版本、发布版本或产品发布
quality assurance (QA)	质量保证
response time	响应时间
software maintenance	软件维护
source code control system (SCCS)	源代码控制系统
stub	存根程序
system documentation	系统文档
system test	系统测试
technical review	技术复审
test case	测试用例
test data	测试数据
test system	测试系统

testing buddy	测试伙伴
throughput	吞吐量
top-down development	自顶向下开发
unit testing, or module testing	单元测试或模块测试
usability test	有用性测试
user documentation	用户文档

复习题

1. 列出并简述程序开发顺序的三个基本排序方法。每个方法的优缺点各是什么？
2. 自顶向下和自底向上开发顺序的概念是怎样应用到面向对象的软件中的？
3. 描述组织编程小组的三种方法。每种方法最适合于哪种类型的项目或开发活动。
4. 源代码控制系统是什么？为什么当多个程序员生成一个程序或系统时需要这样的一个系统？
5. 给出术语α版本、β版本和产品版本的定义。在一个α版本成为一个β版本或一个β版本成为一个产品版本时，有没有定义好的用于决策的标准？
6. 列出并简述实施阶段的QA活动。注意不是软件测试。不执行非测试性QA活动对于软件测试的影响是什么？
7. 好的测试实例的特征是什么？
8. 给出下列术语的定义：验收测试、集成测试、系统测试和单元测试。这些测试一般按什么顺序执行？每种类型测试由谁执行（或者评价每种测试的结果）？
9. 什么是驱动程序？什么是存根程序？与它们联系最紧密的分别是什么类型的测试？它们分别最可能使用什么样的开发顺序？
10. 哪些因素使得测试面向对象的程序比测试结构化的程序更复杂？
11. 列出用于初始化一个新系统中数据库的可能数据来源。简述用于数据库初始化数据装载的工具和方法。
12. 简述直接、并行和阶段安装方法。每种安装方法的优缺点是什么？
13. 为什么在系统实施的后期通常需要额外的人员？
14. 终端用户和系统操作员之间的文档有什么不同？
15. 为什么系统文档对系统本身来说是冗余的？系统文档冗余的表现是什么？
16. 列举已支持维护活动所需要的文档类型。哪种类型的文档使用最为频繁？哪种类型的文档使用最不频繁？
17. 终端用户和系统操作员之间的培训活动有什么不同？
18. 实施一个维护性的改动与新系统的开发的差异如何？它们的相似性如何？
19. 为什么系统软件的升级可能不被安装？不安装它们的代价是什么？

思考题

1. 研究图10-5所示的落基山运动用品商店系统流程图。开发一个基于IPO开发顺序的初开发计划。哪些程序难于分类为输入、处理和输出？一个IPO开发顺序的直接应用是否适用于这个系统？如果不能，应该对这个初开发计划做哪些变化？
2. 本章讨论了自顶向下和自底向上的面向转换的结构图的开发顺序。这些开发顺序是否也能应用于如图10-10所示的面向事务的结构图呢？如果可以的话，该怎么做？
3. 描述使用自顶向下和自底向上的开发顺序开发的软件的测试过程。哪种方法需要的测试资源最少？在使用每种开发顺序的情况下，哪种类型的错误可能最早被发现？根据所需的测试资源和

在测试过程中发现严重错误的能力，哪一种开发顺序最好？

4. 假定像你对问题1的回答中所描述的那样开发洛基山运动用品商店客户支持系统，有14个人可用于编码和测试，那么，什么规模和类型的小组最适合于这项计划？

5. 考虑使用一个全程生命周期CASE工具仅以电子模型的方式为系统编写文档。优点是显而易见的（例如分析员改变了反映新需求的模型并且自动生成了一个更新的系统），有什么缺点吗？提示：这个系统或许要维护10年以上。

6. 一旦系统投入使用，一些类型的系统文档（例如，在分析阶段开发的模型）很少有人查看，那么不保存这种文档类型的优点和缺点是什么？

实验练习

1. 假定你和5个同学负责开发如图10-16所示的创建新订单的子系统。请制订一个开发和测试计划以使编写并测试所需的模块，并假定你有3周的时间来完成所有任务。

2. 在你的一个编程或系统开发类中实施一个正式的QA过程（首先要得到你导师的允许），组织一个学生小组来实施一个检查过程，让某个成员或所有成员准备有关其所写代码的演示材料，并在小组会召开之前把这些材料分发给小组成员。如果可能，建立一个用于测试每个人的代码的伙伴系统，根据代码开发所需的时间和最终的产品质量做出评分。

3. 检查典型的个人或办公产品包附带的终端用户文档。例如诸如的Office系统。根据本章描述的分类方法来分析这些产品中提供了哪些类型的文档。在内容和格式上与用于业务应用的文档有什么不同？

4. 与一个计算中心或者信息系统管理者讨论由于千年虫问题而引发的硬件和软件测试的测试过程。其中执行了哪种类型的测试？哪种类型的小组用来开发和实施这些测试？

5. 写你所在学校或工作单位的一个终端用户进行安装，或者为近期安装或发布的业务应用，编写文档并提供培训。需要提供哪种类型的培训和文档？用户认为培训充分吗？用户认为文档有用且完备吗？

实例研究

HudsonBanc记账系统升级

两个相邻地域的地区银行合并成HudsonBanc。两个银行都有信用卡操作和大约30年的内部开发和升级的记账系统，这两个系统执行相似的功能，都在IBM大型主机上按批处理模式运行操作。合并两个记账系统被认为是能迅速节约成本的首选方法。

HudsonBanc启动了一项计划来研究怎样合并这两个记账系统。升级任何一个系统的方案都被取消了，因为现有的技术相当落后，升级的费用估计会很高。HudsonBanc决定应该购买或者建立一套新的系统。管理部门倾向于购买方案，因为使一个新购买的系统上网能够更便宜、更快一些。他们准备了一项RFP，收到了许多反馈意见，经过几个月的分析和研究以后，他们选择了一家卖主。

新系统的硬件在一个月内就安装好了，随后的一个星期安装了软件。10%的客户账号作为随机的数据复制到了新系统中。新系统与旧系统并行工作了两个月，为了节约完成复制所花费的成本，新系统进行计算但没有实际打印记账报告。付款数据输入了两个系统，用于升级并行客户账号数据库，人工检查了复制的账户记录以确保它们是相同的。

在第二遍的记账循环测试之后，新系统可以投入生产了。所有的客户账号在4月中旬都移植到了新系统中，老系统在5月1日停止使用，新系统接管了所有操作。问题立刻就出现了，系统不能处理大量增加的事务容量，数据输入很慢，支付很快备份了几个星期。系统不能正确处理某些

类型的交易（例如，逾期付款费用的更正和征信）。通过人工检查最近转移的账户记录，发现大约有50 000个账户错误。

人工检查花了差不多6个星期来纠正这些不正确的账户，更新了处理所有交易类型的功能。6月20日，公司尝试打印出50 000个更正的客户账户的记账数据，但系统拒绝打印超过30天的事务信息。咨询的咨询顾问和经验商得来的结论是，修正这个30天的限制需要1个多月的工作和测试。最后认为，以手工输入方式进行账户调整及其后30天以内记账是解决这个棘手问题的最快且风险最小的方法。

清除备份日志花了两个月时间，这期间，寄出了许多不正确的账单。客户支持电话线路一直在超负荷运转，从其他地区调来25个人，并增加了额外的电话线路以便提供充分的客户支持能力。系统开发人员被重新安排到IS帮助台来帮助清理记账备份日志多达三个月的时间。联邦和州调节当局介入调查这些问题，HudsonBanc同意对客户最近二个月的账单策略不收利息。建立付款调整又进一步加剧了备份日志和人员问题。

1. Hudson Banc在新系统中使用了哪种类型的安装方法？这是正当的选择吗？

2. 可以怎样避免操作问题？

城市影碟出租系统

使用你在第11章中为城市影碟出租系统设计的类图，开发一个实施和测试计划。并说明实施类与其方法时采用的顺序以及在集成测试中将要测试的方法组和类。

对落基山运动用品商店实例的再思考



假定现在是2005年4月底，客户支持系统（CSS）项目的分析阶段将近完成。设计阶段按计划在7月15日完成。实施阶段按计划在11月1日完成。RMO想要在假日销售高峰期间——大约从感恩节到圣诞节——使用新的CSS。RMO希望在即将推出的新目录中通告用户可以上网下订单。新目录的邮寄工作按计划安排在6月15日、9月1日、10月31日和12月10日。

1. 描述与规划新的CSS实施以及通告网上订购能力有关的风险。要记得，新的CSS将替代目前的电话和邮件订购系统并且能够处理网上订单。关于测试、安装和客户通告，RMO应该体现多大的保守性？过于保守的代价是什么？

2. 应该开发什么样的依靠策略（如果有的话）？决定是否使用新的CSS来处理假日订单的最后期限是何时？

3. 开发一个安装规划和进度表。基于前期的风险分析判定你的方法和时间表。

4. 分析培训需求并开发一个培训计划和进度表。如何进行培训，数据转换和测试活动的重叠和组合？对使用网络订购系统的用户的培训和支持如何？

关注Reliable Pharmaceutical Services



使用你在第10章中为可靠性药效服务系统开发的结构图，开发一个实施和测试计划，并说明模块实现顺序和集成测试中所测试的模块组。

参考资料

Robert V. Binder, *Testing Object-Oriented Systems: Models, Patterns, and Tools*. Addison-Wesley, 2000.

Barry Boehm, *Software Engineering Economics*. Prentice Hall, 1981.

Robert G. Ebenau and Susan H. Strauss, *Software Inspection Process*. McGraw-Hill, 1994.

William Horton, *E-learning Tools and Technologies: A Consumer's Guide for Trainers, Teachers,*

Educators, and Instructional Designers. John Wiley & Sons, 2003.

William Horton, *Designing Web Based Training: How to Teach Anyone Anything Anywhere Anytime*. John Wiley & Sons, 2000.

William Horton, *Illustrating Computer Documentation: The Art of Presenting Information Graphically on Paper and Online*. John Wiley & Sons, 1991.

International Association of Information Technology Trainers(ITrain) Web site— http://itrain.org/.

Edward Kit, *Software Testing in the Real World: Improving the Process*. ACM Press, 1996.

David Kung, Jerry Gao, Pei Hsia, Yasufumi Toyoshima, Chris Chen, Young-Si Kim, and Young Kee Song, "Developing an Object-Oriented Software Testing and Maintenance Environment," *Communications of the ACM*, volume(1995-10, 38(10):75-87).

Steve McConnell, *Code Complete*. Microsoft Press, 1995.

Arthur H. Watson and Thomas J. McCabe, "Structured Testing: A Testing Methodology Using the Cyclomatic Complexity Metric." NIST Special Publication 500-235, National Institute of Standards and Technology, September, 1996.

http://hissa.ncsl.nist.gov/HHRFdata/Artifacts/ITLdoc/235/mccabe.html.

第16章　系统开发中的当前趋势

学习目标

阅读本章后，你应具备如下能力：

- 解释自适应方法开发与敏捷
- 列出并能描述统一过程开发方法的特征
- 列出并能描述敏捷软型的特征
- 比较和对比极限编程与Scrum开发方法的特征
- 解释模块驱动结构的企业级开发中的重要性
- 描述框架、组件、开发过程，以及它们在系统开发中的影响

本章要点

- 软件原则和实践
- 自适应方法开发
- 模块驱动结构——通用解决方案
- 框架、组件和服务

VALLEY REGIONAL医院：衡量一个项目的进展

Valley Regional医院的技术副院长Claire Haskell安静地听着Henry Williams讲述关于新的病人记录系统的进展报告。Henry是为这个医院开发病人记录系统的项目团队的项目经理。Charlie Montgomery和Jason Smith都是项目的赞助商，Charlie Montgomery负责病人的信息和记录，Jason Smith负责软件开发。几个月前，Jason和Henry与Claire见面并告诉她用一种叫做敏捷开发的新开发方法来开发这个项目。他们已经和Charlie对话，并且Charlie也已经同意试着用敏捷开发方法来开发。Claire已经批准这个工程和他们的需求，尝试用新方法来开发，尽管她对敏捷开发方法知道得很少。

在他的介绍中，Henry一直讲着团队在一起工作得有多好和他们都很有兴趣。尽管Claire对团队很好地工作感到高兴，但是她想知道更多的细节。她想知道这个系统是不是在按计划进行，并且成本是否在预算之内。耐心地听了20分钟以后，她再也等不住了，她要求Henry直接拿出时间表并汇报团队进展的情况。Henry概略地说了一下时间表，但是这无济于事——这个时间表没有像分析、设计和编程这样的里程碑。相反，她看到了其他的术语：迭代、用户故事和重构。

Claire对这一点显得很恼怒。因此她转向Charlie并直接说："从你的观点看，工程的确切进展到底怎么样了？"他的回答让她感到很惊讶。

Charlie说："我和管理者对我们见到的演示感到很高兴，我们对验收测试的系统质量也很满意。从我们目前所见到的来看，这个系统真是我们所需要的。但是考虑到时间表，我不能确定整个系统是否能按时交付，虽然我这样认为，但是我并不参加到每天的开发中。"

Claire感觉好点了，至少到目前为止，这个系统是我们所需要的。但是她还是希望能得到项目经理的保证："Henry，难道我们要延期完成吗？这个系统需要按时完成的。"

Herry回答："到目前为止我们的进展符合时间表，并且一切看起来都很顺利。但是我不能给你一个传统的时间表（有主要里程碑的时间表），不过这有接下来两个月的滚动时间表。"

Claire并不满意。她要求Herry会议结束后留下来和她私下交流。她变得很焦急，说："Herry，对于这个项目我们需要更多的责任。我看唯一的办法是我们经常碰面来监督进度。我要一个项目剩余部分的严格的时间表，并且这个时间表要在周一早上放到我的办公桌上。我给你三天时间来制订时间表。然后我要你从现在开始每周能见我一次，这样可以确保项目按计划进行并在交付日期前完成。"

尽管他对Claire的建议感到不愉快，但是他还是勉强接受了。

概述

在前面的章节，已经介绍了许多为解决业务需求而开发健壮的信息系统所必不可少的概念和技术。你已经学到了所谓的软技术。这些软技术与项目管理、迭代、信息收集和表达相关。你也学到了与解决问题、建立需求模型和设计新系统相关的硬技术。你也许知道传统的方法或面向对象的方法，或者两者都知道。你也学到了许多关于项目、迭代开发和实现替代的重要概念。总之，你已经学会了系统开发有用的知识，获得了着手为企业和其他组织开发信息系统的许多工具。

正如你通过信息系统所学到的那样，信息系统的规则是动态的、不断变化的。几年前广泛使用的工具和技术，许多现在已经消失或者被新的方法所替代。另外，现在系统的范围很广，构成的系统包括整个企业范围内的、分布式的、互动式的和相互连接的，系统还支持台式机和基于网络的计算，并且还能运行在各种计算机和移动设备上。更复杂的信息系统需要一套新的编程语言和工具，随着部分规则的不断演变，信息系统也需要创新，并创造新的系统开发技术，也就是说，用新方法来支持系统开发。

在第2章中我们介绍了预测性和自适应开发方法。过去预测性开发方法曾占主导地位。如今许多开发者创造了不同的、更多的自适应方法来开发系统。在这一章中，我们将介绍组织和进行系统开发的最新方法，包括预测性方法和自适应方法。

- 在第2章中介绍过的4种方法都是自适应的，包括：统一过程开发方法（UP）、敏捷开发、极限编程和Scrum方法。这4种方法有共同的思想，但是特征各不相同。
- 介绍企业系统集成思想的模型驱动结构（MDA）。MDA适用于预测性方法和UP、敏捷开发方法、极限编程和Scrum方法。
- 介绍关于对象框架和组件的基本思想。这两种方法对增量开发提供了额外的支持，加速了开发速度并提高了系统的质量。

本章只是介绍性的内容，这些有可能激发你的兴趣，你也有可能更想要去研究这些方法和更多的细节，如果这样，将会对你自己的专业开发和系统开发领域的改进有所帮助。

16.1　软件原则和实践

过去的50年见证了计算机领域的巨大进步。计算能力继续以惊人的速度提高。另外，出现了各种各样用于计算的设备，包括带有数码相机的手机、手提笔记本电脑、网络电话、嵌入计算芯片的设备和零售商店中带有无线电波的ID芯片。尽管许多客户对大规模的系统不太理解，但是其驱动了许多社会上基础性的活动。这些活动如在银行和其他支持银行业和信用卡业的金融机构之间的金钱交易。在交易背后，支持链式的管理系统激发了基于全世界的销售和发明的平均水平的产量。承运人和托运人之间的运输计划和信息共享使得人们和商品以最小的代价进行交流和流通。今天我们称之为普适计算，意思是说，计算机技术在生活中无

发不在，几乎影响着生活中的各个方面，正如一个计算和系统问题集的解决，并没有包括整个的新需求一样，业务和消费者计算问题还在不断地追求和改善。由于这个趋势，专家们对信息系统的长期的发展目标是有信心的。工业的下个目标是努力保持这样的趋势。

普适计算：目前在生活的各个方面使用计算机技术的趋势。

因此，我们怎样去面对工业和社会的许多需求？很显然，没有单一的解决方案和一个技术能够满足所有的需求。解决方案是由成千上万的人在一起工作并解决各个小问题而产生的。然而，许多规则和实践能够促进工业的发展，其中许多来自致力于计算科学与原理的计算机科学领域，其他应用到每天的业务问题中的计算原则来自信息系统的规则。当然，在这两规则之间有很大的重复，这可以比做科学家和工程师之间的关系——比如，化学家搞研究，化学工程师来实现他的研究成果。人们在计算机科学、信息系统、决策科学和数学规则上的合作能加快推进我们社会技术的不断发展。

在讨论当前趋势之前，我们先回顾一下。如果你想要用一两段话来概括这本书的内容，你会怎么概括？从这本书的前些章节中你学到的中心思想是什么？我们希望你会包括以下两点：

1. 你已经学会了怎么建立模型——模型用于捕获和解释需求与解决方案。就一个模型的广泛定义来说，也包括编写代码的过程。

2. 你已经学会建立一个解决方案所必需的过程或步骤——包括管理和指导系统开发项目的过程。

在这一章中，我们通过描述在建模和开发过程中目前的趋势来重点关注两个主要的领域。另外，我们还讨论支持这种趋势的流行的方法和技术。

首先，在讨论这种趋势的详细内容之前，我们应该先来回顾一下5种重要的软件规则和实践：

- 抽象
- 模型和建模
- 模式
- 重用
- 方法

16.1.1 抽象

抽象就是我们从许多事实和现象中提取出核心规则的过程。当学到去确认一个抽象类（没有例子）的时候我们学习这个规则。一个抽象类作为一个存储其他类所隐含的共有属性和方法的容器。在以用户的需求来建模时，你也需要学会抽象思考。抽象思考是一项很难学的技术。很多人是通过实际的例子来学习新概念的。然而，当你的思维能力很强的时候，你就可以抽象的思考了。

抽象在计算领域是很重要的。计算机领域很多先进技术的发展都是由于计算机科学家能够抽象思考且不断提高抽象能力。比如早期开发者使用集成语言来编写系统，这是一种很基础的机器语言，然后，他们通过更加抽象的思考而发明了编程语言，比如Fortran、CORBOL、C、Java语言和VB。因此我们必须抽象地思考机器语言和语言编译器的特征。思考用户的需求导致了模型和图表的发明，并用其来表达这些需求。再次，这过程需要对一个好的模型特征进行抽象思考并且发明类图表、顺序图表以及它们的属性。今天，我们用更抽象的思考来定义原模型。原模型是描述其他模型的模型，比如，你能够归纳和描述任何类图表组件的类图表吗？当我们讨论模型驱动结构的时候，你会发现，在计算机和系统开发的知识主体中抽象是一个很重要的思想。

16.1.2　模型和建模

模型和建模是第二重要的软件规则。在这本书的前面你已经学到了模型是什么，并且通过整个过程实践建模。一个模型就是真实世界中事物的抽象，代表属性的特殊集合。开发者建模有两个重要的理由。首先，必须了解过程。通过定义和解释它的重要特征来思考。建模可以使我们的思想具体明朗以使我们变得更加精确。在很多例子中，直到我们开始为它建模时才能理解它。第二，使用模型能使我们需要记住的想法文档化并且能够将这些想法传递给他人。在本章的后面，你会发现目前的趋势是着眼于在当前开发中怎样更好地使用模型——通过建模。

16.1.3　模式

第三种软件规则——模式——与抽象与建模紧密相关。一个模式是一个已知问题的或一个可以应用于某个问题的标准解决方案。当我们开始更抽象地思考和建模的时候，起初看起来似乎很难的问题，事实上用概括的眼光来看有相似的特征。在问题和解决方案中，模式变为一种形式。在第11章中，我们提供了许多关于设计模式的思想。我们注意到不同的工业中有标准的模式来解决重复的问题。比如，银行系统可以作为记录处理金融事务的基本目的。为此设计模式的存在不但是系统结构的需要，同时也是整个系统的需求。设计模式可以用来解决问题，为此其被广泛的接受。我们在这一章中不再重复讨论模式，但是你需要意识到。其在提高系统质量和加速系统开发中是一种驱动力量。

16.1.4　重用

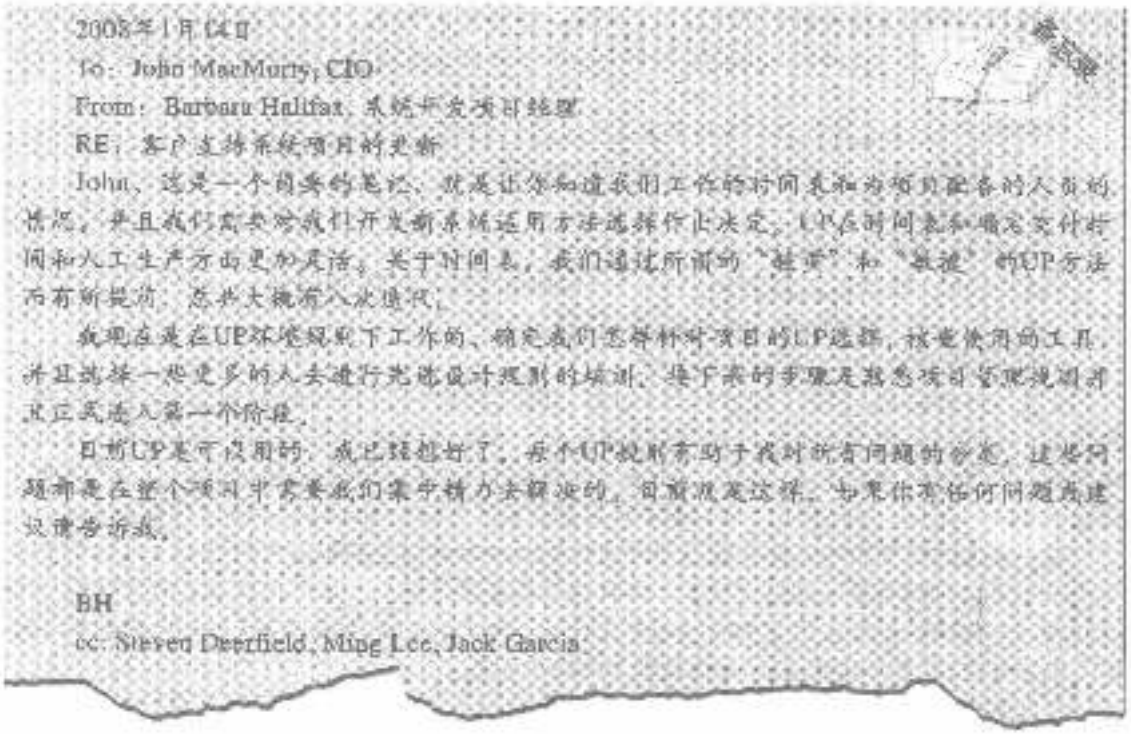
第四种软件规则——重用——是以前的一种规则，但是现在不太适用了。因为开发者已经发明了模式，他们正在建立可以重复使用的标准解决方案和组件。通过重用，如今的开发变得更加多产。比如，为Windows平台开发一个全面的用户界面，几乎所有的开发者都使用表格、按钮、菜单、下拉框、文本框和单选按钮的标准库。如果我们在界面中放置一个按钮都要重写代码来显示一个按钮，这将会浪费很多的时间。在代码层，Windows上界面是重用的例子。开发者也要更抽象地思考并发明更高层次的可重用组件。如今的专家指出，新系统是通过把操作系统、通信系统和应用集成到一个单一的系统中而建立的。因此，如今的系统开发者经常把组件集成到完整的解决方案中。重用在技术上是一种驱动力，比如网站服务、.NET和企业资源计划系统（ERP）。当我们讨论组件库和框架时，我们会讨论重用的更多细节。

16.1.5　方法和过程

第五种（最后一种）规则——方法——在这本书的前面也做了介绍。我们指的方法是过程的集合——包括规则、指导方针和技术——它被定义为系统如何建立、项目开发怎么管理。近年来，系统开发者实验了许多不同的方法。在第2章中，我们解释了这些方法可以被分为预测性方法和自适应方法。现在我们讨论在第2章中介绍过的4种系统开发的自适应方法：

- 统一过程开发方法（UP）
- 敏捷开发
- 极限编成（EP）
- Scrum

在对John Blankers关于Rocky Mountain Outfitter的客户支持系统的日常报告中，Barbara Halifax重新概括了其团队用于系统开发的基本方法——基于自适应的统一过程方法以及他们的进展（见Barbara备忘录）



2008年1月14日

To：John MacMurty，CIO

From：Barbara Halifax，系统开发项目经理

RE：客户支持系统项目的更新

John，这是一个简要的笔记，就是让你知道我们工作的时间表和为项目配备的人员的情况，并且我们需要对我们开发新系统运用方法选择作出决定。UP在时间表和确定交付时间和人工生产方面更加灵活。关于时间表，我们通过所谓的"轻重"和"敏捷"的UP方法而有所提高，总共大概有八次迭代。

我现在是在UP环境规则下工作的，确定我们怎样杯对项目的UP选择，核定使用的工具，并且选择一些更多的人去进行完选设计规则的培训。接下来的步骤是熟悉项目管理规则并且正式进入第一个阶段。

目前UP是可使用的，我已经想好了。每个UP规则有助于我对所有问题的分类，这些问题都是在整个项目中需要我们集中精力去解决的，目前就是这样。如果你有任何问题或建议请告诉我。

BH

cc：Steven Deerfield，Ming Lee，Jack Garcia

16.2 自适应开发方法

系统开发的自适应技术允许考虑不确定性。在着重于新应用的开发项目中，用户需求经常不能很好的理解并且细节上很难描述。因为新系统的范围很难确定，如果可能的话，系统分析员也很难制定一个详细的项目规划。这种情况下实施一个系统开发项目最好的方法是早期确定主要目标和随着项目进展制定详细工作计划。

两种力量促使增加对自适应方法的兴趣。首先，如在第3章所学到的，系统开发中很小的成功都是很不容易的。软件开发是困难的，并且成功的是很难记清的。在第3章中我们讨论了在改善过程和增加成功率中好的项目管理技术的重要性。另外，专家和开发者已经发明了各种不同的提高成功率的自适应方法。

自适应方法之后另一种力量是如今多变的业务环境。以前比较稳定的环境主要依赖于控制成本和加紧内部程序的管理。相反，如今业务的成功依赖于灵活性和对市场变化做出快速反应的能力。需要12个月、24个月或者更长时间定义系统需求是一种死板的系统开发过程，这种开发过程不能够灵活地适应快速变化的步伐。但是更新的是，自适应方法允许业务需求发生重大的变化。

从理论上来讲，开发的任何项目，不管是物理项目还是软件项目，都是随着过程完成的。需要控制每个过程以确保其进度。过程控制可以分为两类：预测型和经验型。预测型控制定义为在细节上对过程的监测。如果一个过程偏离了计划，那么详细的步骤——比如某个工作打破了结构——描述可以用来控制它。当很多计划可以提供详细的细节时，预测型控制效果更好。然而，对那些不可预测的过程，增加太多的细节和太多的控制只会使问题恶化，试着去控制不可控制的东西将会导致时间和金钱的浪费。

相反，经验型控制描述的是可变的、不可预测的过程。这个过程是通过当变化发生时及

时处理和用最好的方法来纠正错误。换句话说，经验型控制基于特定的环境监测进展并且及时的纠正错误。因为许多软件的开发项目包含了很多的不确定性，经验型过程需要对它们做更好的选择。

所有的自适应方法使用经验型控制并有它们自己的规则。然而，它们有一些共同的特征：

- 在开始的分析、设计和文档化过程中缺少经验
- 更关注增量开发
- 项目团队中涉及更多的用户
- 减少仅用于近期工作的详细计划。下一阶段可能需要高级的计划
- 将工作分成不同的时间段来加紧时间控制
- 更多地使用可以自组织的工作小组

首先，自适应团队不能花很多时间来分析、设计和文档化，因为这些活动是达到目的的一种手段，它们只是用于写可执行代码的简单工具。其次，快速写代码的唯一办法是分成小块来完成和以增量式开发系统。最后，快速写代码要求用户准备好一个项目团队——变成项目开发团队的一部分，并且和开发者一起工作来创立解决方案，因而这个方案适合业务的需求。

减少详细的计划和把工作分成不同时间片的时间表有助于控制团队的进展。每个增量定义为一个可以加到系统中的特定功能，并且这个功能要足够小，从而可以制订有意义的详细时间表。这样团队就能在规定的时间内完成任务。随着开发者经验的增多，他们就能熟练地把工作分成在3或者4周的短时间段内就能完成的小任务。

在自适应方法下，坚持用时间段的方法，每个团队为每个增量制订他们自己的时间表、自组织，这样他们的工作将会更有成效。唯一的额外的时间表只有在要求和其他工程合作时才要强加制订。

在接下来的部分，我们将探讨4种自适应方法：统一过程开发（UP）、敏捷开发、极限编程和Scrum方法。我们从UP开始介绍，因为许多人认为它是基础的自适应软件开发。尽管许多开发者认为UP是基于预测型和自适应型方法之间的。后面讨论的更高级的自适应方法常和UP作比较。

16.2.1　统一过程开发

统一过程开发方法（UP）是一个面向对象的开发方法。其由Rational软件公司提供，现在是IBM公司的一部分。UP是由Grady Booch, James Rumbaugh, Ivar Jacobson开发的——这三位先驱也是统一建模语言（UML）的开发者——UP是他们试图去定义为系统使用UML和描述一种新的自适应的系统开发生命周期的完整方法。在UP中，项目开发过程和开发方法是同步的。

现在UP被广泛认为是面向对象开发的标准的系统开发方法，并且许多它的变体也在使用中。原始的UP版本定义为开发过程的每个步骤的详细活动集。最近较多的版本是同一系列的简化方法，其具有更少的活动和交付。

如先前讨论自适应方法（包括UP）都是基于迭代方法的开发。在第2章（见图2-7）中讲过，每次迭代都可以看做一个小项目，在小项目中需求定义为基于任务分析，系统组件设计和通过编程与测试实现这些组件，至少是部分实现。然而，在自适应开发中最大的问题之一是每次迭代的重点是什么。换句话说，项目早期迭代与后期迭代的对象和重点是否相同？UP通过将一个工程项目分成4个主要阶段来回答这个问题。

1. UP阶段

UP的一个阶段可以看做一个目标或工程项目特殊部分的重点。UP生命周期的4个阶段是起始、细化、构造和交付，如图16-1所示。

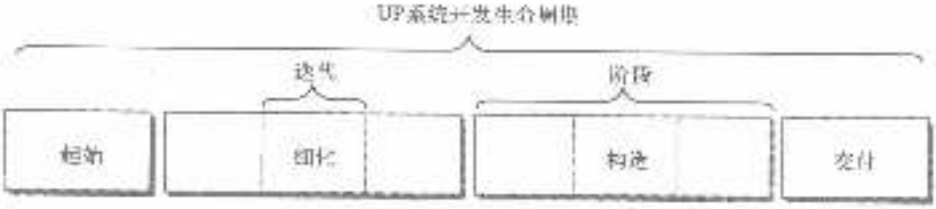


图16-1 UP系统开发生命周期

UP生命周期的每个阶段描述的是此时此刻项目团队成员和他们活动的重点和对象。因此，4个阶段为项目规划和时间跟踪提供了普遍使用的框架。在每个阶段，规划的几次迭代允许团队根据问题或环境变化做出灵活的调整。4个阶段中，每个阶段的项目团队的重点或对象的简单描述如图16-2。

UP阶段	对　象
起始	开发一个系统的近似版本，使用业务案例，确定范围，并且生成粗略的成本估计和时间表
细化	提炼版本，确认和描述所有需求、缺陷范围，设计和实现核心结构和功能，解决高风险，并生成现实的成本估计和时间表
构造	迭代方式实现剩下的低风险、可预测和容易的部分并准备部署
交付	完成beta测试和部署，为此用户有一个可以工作的系统并获得期望的结果

图16-2 UP阶段和对象

起始阶段 在任何一个项目规划的阶段中，在起始阶段，项目经理会为新系统开发和定义一个版本，用于展示他是如何改善操作和解决存在的问题的。项目经理为新系统使用案例意味着新系统的利益高于成本。系统的范围必须确定，因而可以明确系统能够完成什么事情。确定范围包括确定系统的许多关键需求。

通常起始阶段是在一次迭代中完成的。在任何一次迭代中，都对系统的一部分进行设计、实现和测试。当软件开发完成时，团队成员必须保证系统的版本仍然和用户期望一致或能按计划工作。通常这一点被证明以后，就可以抛弃原型了。

细化阶段 细化阶段通常包括几次迭代，并且通常早期的迭代完成确认和定义系统的所有需求。因为UP是自适应开发方法，项目的工作开始以后，需求是可以演变和变化的。

细化阶段也完成分析、设计和实现系统的核心结构。系统中能引起最大风险的部分首先被确认和实现。开发者不知道要付出多大的努力才能完成这个项目，直到开发者确定知道风险最大的部分怎么完成。在细化的结束阶段，项目经理应该有更实际的成本估计和时间表，同时项目的业务案例可以确定。记住，系统关键部分的设计、实现和测试是在细化阶段完成的。细化阶段并不完全等同于传统SDLC的分析阶段。

实践指导

确保不要把UP阶段和SDLC的瀑布方法相混淆。细化并不完全等同于传统的分析阶段。 ■

构建阶段 构建阶段包括继续设计和实现的多次迭代。系统的核心结构和高风险部分已经完成。现在工作的重点转向系统常规的数据维护功能和完成帮助与用户参考功能，团队也开始规划部署系统。

转化阶段 在转化阶段包括一次或者多次迭代，其包括最终用户接受测试和β测试，并且

系统已准备好可以使用。当系统投入使用之后，也需要支持和维护。

2. UP规范

如前所述，UP的4个阶段通过指出某时某刻项目团队的重点来定义项目工程的顺序。为了使迭代开发可以管理，UP定义了在每次迭代中使用的规范。规范是与对开发项目某个方面有贡献的一系列功能相关的活动。UP规范包括业务建模、需求、设计、实现、测试、部署、设置和变化管理，项目管理与环境。每次迭代通常包括所有规范的活动。

规范：合起来对UP项目开发过程有贡献的一系列功能相关的活动。

图16-3显示了UP规范是怎么样包含在每次迭代中的。典型的计划需要4周的时间。每个规范在曲线下面阴影部分的面积大小表示包括在迭代中每个规范的相关工作量。每次迭代之间的工作量和性质是不相同的。比如，在第二次迭代中，重点是业务建模和定义需求，而更少的时间用在实现和部署上。在第五次迭代中，则相反，重点放在实现、测试和部署上，而更少的时间放在建模和定义需求上。但是，大多数的迭代中包括所有的规范中的工作。

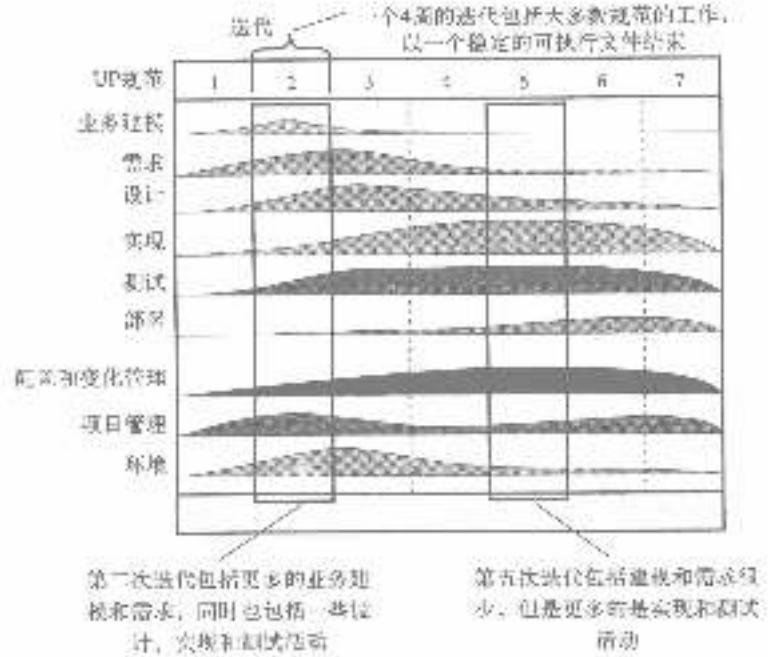


图16-3　每次迭代中UP规范使用的不同量

图16-4显示了整个UP生命周期——阶段、迭代和规范。图中包括所有关键的UP生命周期的特征，并且对理解典型的信息系统开发项目的管理很有用。

图16-4说明了阶段是怎么样包括每个规范的活动的。但是发生在每个规范里的详细活动是什么？规范可以划分成为两大类：系统开发活动和项目管理活动。6个主要的UP规范开发如下：

- 业务建模
- 需求
- 设计
- 实现
- 测试
- 部署

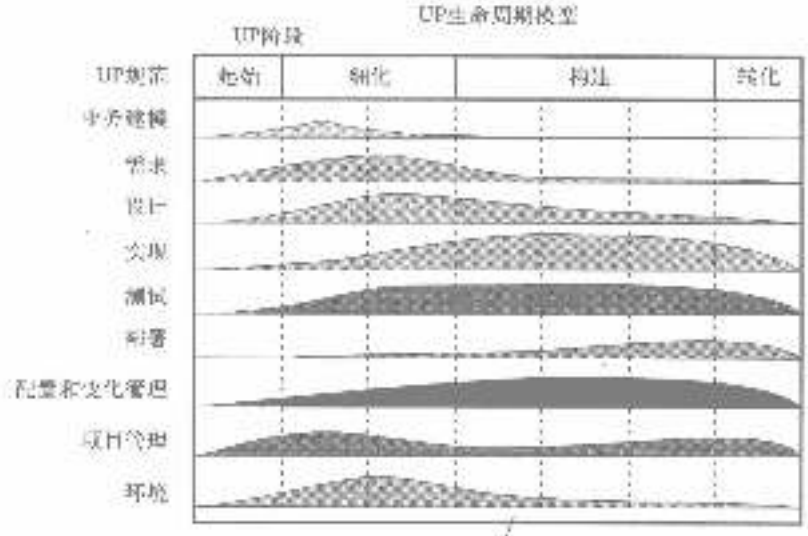


图16-4 阶段、迭代和规范的UP生命周期模型

每次迭代类似于一个小项目，完成系统的一个小部分功能。对每次迭代，项目团队必须了解业务环境（业务建模），定义系统的这部分必须满足的需求（需求），为系统的这部分设计满足其需求的解决方案（设计），集成计算机代码使这部分能工作（实现），对系统的这一部分进行全面的测试（测试），然后把完成和通过测试的这部分让用户使用（部署）。

为规划和控制项目必要的另外三个支持规范：

- 配置和变化管理
- 项目管理
- 环境

在整个项目生命周期中，9个UP规范都会用到，但是很难区分。比如在起始阶段只有一次迭代，在起始阶段的迭代中，项目经理可能制作一个模型来展示系统环境的一些方面（业务建模规范）。系统的范围是通过定义系统的许多关键需求和用例来描述的（需求规范）。为提高技术的灵活性，系统技术方向需要设计（设计规范），编程（实现规范）和测试（测试规范）。另外，项目管理者为处理项目的变化而制订计划（配置和变化管理规范），按计划工作和成本/利益分析（项目管理规范），选择UP阶段、迭代、转化和工具以满足项目需求（环境规范）。

细化阶段包括多次迭代。在首次迭代中，团队工作重点是主要类的细节和迭代用例（业务建模和需求规范）。同时他们可能完成所有用例来确定范围（需求规范）。在迭代中，用例是通过创建设计类图表和迭代图表来设计的（设计规范），使用Java和VB.NET编程（实现规范），全面的测试（测试规范），团队继续接受培训，关于他们完成的UP活动和使用的开发工具的培训（环境规范）。

到现在，项目进展到构建阶段，大多数用例已经在开始的几个阶段设计和完成。项目的重点转向对每个用例满足其他技术性能和可靠性需求，最终设计和实现。这些需求通常是很普通的并且是低风险的，但是它们对系统的成功是很关键的。重点要放在设计系统控制和安全，实现和测试这些方面。

UP作为一种系统开发方法必须根据开发团队和特定的项目需要进行裁剪。对那些可转化

过程和使用正式的程度必须做出选择。有时项目需要正式的控制和控制，其他时候就不需要太正式。UP需要不断根据项目进行裁剪。

实践指导

确保成功的UP裁剪适合项目。 ■

16.2.2　敏捷型开发观点和敏捷建模

快速变化的市场迫使业务对新的机遇做出新的反应。有时新的机会出现在实现其他业务需求之中。为了生存，业务必须敏捷。敏捷性——能够快速改变方向，甚至在项目进行了一半时——是敏捷开发的里程碑。在未知和快速变化的环境中，敏捷开发是一种观点和一系列软件开发的纲要。它为特定的开发方法（如UP）提供理念。每种方法中敏捷的程度可以不同。比如，我们认为UP有某种程度的自适应。许多UP项目采用敏捷的观点，其他可能用得会更少。

关于敏捷开发，敏捷建模是关于怎么样建模的观点，即哪些地方要求正式、详细，那些地方可以概要、简略。图16-5说明了敏捷开发观点、专门的自适应方法和敏捷建模的使用之间的关系。

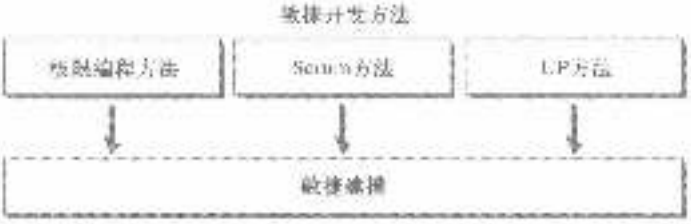


图16-5　使用敏捷建模的自适应方法

1. 敏捷开发观点和价值

"敏捷软件开发宣言"（看"参考资料"部分）确定了4种基本价值，它代表了敏捷运动的核心观点。这4种价值是：

- 快速响应变化于原有计划
- 个体和交流凌驾于程序和工具
- 执行软件凌驾于综合文档
- 用户合作凌驾于合同协商

注意，列表中左边的优先权高于右边。系统开发中的人员，不管是团队成员、用户或系统相关者，对一个确定的敏捷开发项目都需要接受这些优先权。采用敏捷开发并不一直是容易的。

在敏捷运动中，一些业界领导者创立了混序这个术语来描述敏捷项目。混序来源于两个单词——混乱和有序。列表中的前两个价值——快速响应变化于原有计划，个体和交流凌驾于程序和工具——这是产生混乱的原因。但是他们认为，软件开发本来就有未知和不可预测的因素，所以当然会有一些混乱，开发者需要接受这些混乱，但是也需要使用后面讨论的具体的方法把有序添加到混乱上，使项目进行下去。

混序：用于描述自适应项目，该项目既有混乱又有秩序。

经常想要控制开发团队、要求有详细的计划和时间表的管理者，以及与系统相关者都要努力接受这种不严格的观点。然而，敏捷观点却是一种有效的方法。在项目进度表中变得更加灵活，随着项目的进展来计划队制订计划和工作。

敏捷开发的另一个重要价值是客户需要参加到项目团队中，他们不是和开发团队单下为一

此开发任务来讨论开发规范，然后做他们自己的事情。相反，客户需要和技术团队合作并成为其中的一部分。因为软件开发贯穿于整个项目，客户需要参与到确定需求和测试组件中来。

同时，合同也采取一种完全不同的方式。固定的价格和固定的传达没有意义。合同有好多种合作方式，例如，如果项目以增量方式来衡量，结果没有进展的话，用户可以选择取消合同。敏捷开发项目的增量交付是通过将系统分解成多个小任务来完成的，而不是文档或说明书。

对于敏捷开发来说，模型和建模是很关键的。因此，我们接下来解释敏捷建模。在建模的原则与实践方面阐述了许多核心的价值。

2. 敏捷建模原则

你的第一印象可能是，敏捷方法就意味着很少建模或者不用建模。敏捷建模（AM）不是更少的建模，而是为实现正确的目标在恰当的细节水平建立恰当的模型。在本章的开始部分，我们确认了建模的两种原因：（1）为了理解你所建的东西；（2）在系统的解决方案中重要部分之间的通信。AM是由支持建模的这两种原因的原则和一系列实践组成的。AM并不是命令去建立那个模型或者这些模型的正式程度。相反，是帮助开发者沿着他们的模型的轨迹进行开发——使用模型而不是把建模作为一种目标。AM的基本原理说明了开发者开发软件应该具有的态度。图16-6概括了敏捷建模的原则。下面我们讨论这些原则。



图16-6 敏捷建模规范

实践指导

确保你的建模只是一种方法而不是把它作为一种目标。所有的模型都要有自己的目的。 ■

软件开发作为你的首要目标 软件开发的首要目标是开发出高质量的软件。工作是最初主要的衡量方法是软件能够工作，而不是系统需求和说明书的中间模型。建模只是一种方法而不是一种目标。任何一种活动，如果不能证明对软件产品的最终目标有贡献，则应该受到质疑和避免。

下一步的成果作为你第二个目标 只把目光放在软件能工作上可能会被自己击败，为此开发者必须考虑两个重要的对象。首先，需求模型对开发设计模型是有必要的。因此不要认为如果这个模型不能用于写代码就是没有必要的。有时在编写最终代码之前一些中间步骤是必须的。其次，尽管高质量的软件是首要目标，代码的长期使用也很重要。因此一些模型对支持维护和系统的增强也是有必要的。当然，代码是最好的文档。但是许多结构的设计很难轻易从代码中看出来。认真对待其他的软件产品对高质量软件的长期使用是有必要的。

最小化建模活动——少而精 只创建那些必要的模型，通过这些工作就足够了。这个原则不是进行草率工作和不充分分析的借口。创建的模型应该清晰、正确和完整。但是不要创建不必要的模型，同时要保证每个模型尽可能简单。通常，最简单的方案就是最好的解决方案。复杂的方案往往难以理解和维护。然而，我再次强调，简单并不是不完整的理由。

拥抱变化和增量改变 敏捷建模的潜在观点是，开发者必须灵活并对变化做出快速反应。一个好的敏捷开发者要自愿接受、甚至拥护变化。把变化看做是正常的而不是异常，监视变化并且把变化集成为一个模型。接受变化的最佳方法是增量式开发。做很小的一步并且把问题分解成几个小问题。增量地改变模型，然后证明其合法性并确保其正确。不要试图去完成一个大版本的所有事情。

目的模型　我们前面说过，建模的两个理由是：理解你建的是什么，以及系统解决方案各重要部分之间的通信。确保你建的模型支持这些理由。有时开发者通过以下几点声明来证明其建模的正确性：（1）开发方法来决定模型的开发；（2）有人想要模型，即使这人不知道它为什么重要；（3）一个模型可以替代面对面的讨论。为你开发的每个模型确定一个理由和一个听众。开发的模型要足够详细来满足理由和听众的要求。顺便说一句，听众可能就是你自己。

建立多种模型　UML和其他建模方法一起有多种模型来表达手边问题的不同方面。为了成功一理解或交流——你要对要求的解决方案的不同方面建模。不要去开发所有模型，确保模型最小化，但是要开发足够多的模型来确保表达所有的问题。

建立高质量的模型并快速得到反馈　没有人喜欢草率的工作。这是错误的思想和错误的引进。模型中避免错误的一种方法是快速得到反馈。反馈来源于用户，也来源于技术团队成员。其他人可能有很好的观察力，看问题与鉴别解决方案有不同的方式。

重点放着内容上而不是表达上　有时项目团队可以获得先进的用例工具。用例工具可能很有用，但是有时它们是分散的，因为开发者花费很多的时间在制作漂亮的流图上了。使用工具是明智的，需要建立一些模型用于交流或符合同，甚至处理期望的变化和更新。

公开交流的方式，相互学习　所有自适应方法强调团队工作。不要把你的模型保护起来，其他的队友会有很好的建议。不需要掌握问题或模型的所有方面。

熟悉模型并知道如何使用模型　成为一个敏捷开发者并不意味着你没有技术。你要有更多的技术去知道模型的优劣，包括什么时候和怎么样使用模型。专家级建模开发者应采用前面提到的简易、高质量和多种模型开发的原则。

适应特定的项目需求　因为每个项目存在唯一的环境，所以项目各不相同，包括不同的用户、系统相关者和团队成员；要求不同的开发环境和部署平台。让模型和建模技术适应业务和项目的需求。有时模型简单而富有信息。对于其他的项目，可能要求更加正式复杂的模型。敏捷建模者要有能力适应每个项目。

3. 敏捷建模实践

接下来的实践支持前面提到过的AM规范。AM的核心在于实践，它给实践者具体的建模技术。图16-7总结了敏捷建模实践。下面我们讨论实践的每一步。



图16-7　敏捷建模实践

迭代和增量建模　记住建模是一种支持性的活动，而不是软件开发最终的结果。作为一个开发者，你应该不断创建小模型帮助你理解和解决问题。初级开发者通常对选择哪种模型有困难。你应该不断学习关于模型的知识并扩大你的知识面。UML有一个大的模型集，其包含了许多分析与设计领域的知识。然而并不只是模型才有用。许多开发者还使用来自传统结构方法的数据流图和分解流图。模型是一种工具，作为一个专业建模者，你应该有大量的工具集用来建模。

团队工作　如图16-5所示，AM支持不同的开发方法。在所有方法中，其中一条是，开发者要在2～4个人的小团队中工作。另外，用户也要参加到团队中来。比如，假设手中的任务是理解采购订单怎么创建。对的AM实践表明，有合适的人员一起工作，包括团队

成员和用户。即使在电子白板上也能开发出详细的过程模型。其他小组可以为该模型拍一张数码照片，并将照片发布到项目小组的服务器上。这样模型便公开了，没有人能一个人占有它。所有人都可得到它。如果后来需要修改它，可以用软件注释并重新放置。一种可选的方法，在模型变成固定的文档时使用画图工具来开发模型，如笔记本和投影仪的图形开发工具。这种过程并没有像电子白板一样灵活，但是它生产出更加固定的模型。在任何情况下，模型被再次发布，以让所有人使用、评论和更新。

简洁 前面介绍的采购单例子说明了简单并且容易支持的方法。开发者应创建一些模型来帮助他们理解和解决一些难题。在采购单的例子中，模型的重点放在业务过程和用例上。在开始的迭代中，开发者应该把注意力放到典型的过程中，其没有太多的变化。然后在后面的迭代中可以增加额外的条件，安全性和控制要求，以及其他细节。

有效性 在建模的时候，团队可以着手准备好的解决方案并写代码了。这样可验证模型的合法性。简单支持常见的有效性验证。当可以用简单的模型用代码来证明其是合理的时候，就不要再去创建多而复杂的模型了。

文档 许多模型都是为解决一个具体问题而创建的临时工作文档。随着代码的演化和改善，这些模型就过时了。不要试图去更新，而要抛弃它们。如果将它们公布出来并及时更新，这样每个人都会知道它们在历史上的决策和进展，但是现在不能和代码同步。只有当它们坏的时候去更新这是一条准则，它告诉我们浪费时间去保持临时模型是没有必要的。在第一次迭代中，当许多开发的模型同时存在时，它们必须保持一致，然而随着开发的进行，许多模型将变成与其他模型没有关系的工作文档。请注意，项目的目标是开发软件而不是漂亮的模型。只有当它们坏时更新——也就是说，工作团队在没有相关信息的情况下不能有效地工作。

动机 记住建模的基本目标。只有当模型有助于你理解一个过程、解决一个问题或者需要去记录和交流一些东西时才去创建。例如，在设计阶段，团队成员要做出一些设计决定。为了方便对这些决定进行交流，团队张贴一个简单的模型然后公布它。模型对把决定文档化和确保大家有一个共同的理解与参考点是一个非常有效的工具。再次强调，模型只是用于交流的简单工具，而不是最终目标。

既然我们介绍了基本的观点、规范和敏捷开发实践，那么我们就要转向应用敏捷观点的两种方法：极限编程和Scrum方法。

16.2.3 极限编程

极限编程（XP）是在20世纪90年代中期创造出来的，是一种自适应、敏捷的开发方法。极限（XP）这个词有时使人们认为它是一种全新的方法，并且认为拥护XP的开发者是激进的。然而，XP是试图实现最佳的软件开发并且扩展它们到极限。极限编程：

- 被证明是工业上的最佳实践与重点关注对象；
- 联合这些最优实践（强烈的形式）到一种新的方法以产生一种比各部分之和还要好的结果。

图16-8列出了XP的核心价值和实践。接下来，我们首先阐述XP的4种核心价值，然后阐述12种主要的实践，最后，我们描述XP项目的基础结构和用XP开发软件的方法。



图16-8 XP的核心价值和实践

1. XP核心价值

XP的4种核心价值——交流、简单、反馈和鼓励——驱动着它的实践和项目活动。你会认为对任何开发项目，前3个为最佳实践。再思考一下，你也应该看到第4个对任何项目的驱动价值。尽管它不能清楚地陈述。

交流　引起项目失败的一个主要原因是缺少在恰当的时候与恰当的人公开交流。有效的交流不仅包括文档，也包括口头讨论。设计XP的实践和方法以确保进行公开、频繁地讨论。

简单　即使开发者一直主张使用简单的方案，他们也不能一直跟从自己的想法。XP包括的技术加强了这规范并使其成为系统开发的标准方法。

反馈　认为简单、频繁、有意义的反馈是最佳软件开发。功能和需求的反馈来自用户，设计和代码的反馈来自开发者，关于满足业务需求的反馈来自客户。XP把反馈集成到开发的各个方面。

鼓励　当开发者面对做正确的事情或是弃不好的代码重新开始等艰难的选择时，需要鼓励。他们有很紧的时间表内总是得不到鼓励，将会导致大的错误。设计XP实践以使给开发者"这样做是对的"这样的鼓励变得容易。

2. XP实践

XP的12种实践支持刚才说的基本价值。这些实践与本章先前提到的敏捷规范一致。

规划　许多人描述XP为光荣的黑客或20世纪60年代曾使用过的古老的"代码和安装"方法。这是不正确的。XP并不包括规划。然而，XP是一种自适应方法，认为在开始时，你并不需要知道一切。如早期所说的，XP拥护变化。XP规划着重的是很快制订一个概要的计划，然后定义使其更清晰。这反映了敏捷开发观点，即变化比详细计划更重要，这也同个人——和他的能力——比细化过程更重要的观点一致。

XP规划的基础是用户开发的活动集，一个活动简单地描述了系统需要做什么。XP并不需要使用用例这个术语，但是用户活动和用例表达了相似的意思。规划包括两个方面：业务问题和技术问题，业务问题由用户和客户决定，而技术问题是由开发团队决定的。规划，尤其是在项目初始阶段，包括一系列的活动（来自用户）及对能力、风险和每个活动工作的独立性（来自开发团队）的评估。在敏捷开发中，项目要求用户的积极参与，而不是只要求他们在说明书上签字。

测试　软件的每个新的部分都要测试，并且每一种方法都包括测试。XP要求在编程完成之前，要对编写的每个活动首先进行测试以加强总体测试。有两种主要的测试类型：单元测试，其测试一个小单元的代码；功能测试，其测试软件的业务功能。开发者编写单元测试，用户编写功能测试。任何代码，在集成到正在开发的系统库之前，必须通过测试。首先编写测试代码，XP自动调用这些代码并不断执行。总之，可创建一个测试用例库，这样，当需求发生变化和代码需要更新时，测试就可以自动并快速地重新运行。

两人一组程序设计　相比于其他任何实践，本实践是使XP闻名的重要原因之一。其替代了简单的要求一个程序员监督另一个程序员的工作。两人一组程序设计把编码工作分成两半。首先一个程序员把重点放在设计和核查算法上，而另一个程序员编写代码。然后他们互换工作，都考虑设计、写代码和测试。XP依赖于综合而又不断的检查。有兴趣的是，研究表明，两人一组编码确实比一个人编程效率更高，花很多时间去写起始代码，但是长期的质量更高。错误可以在早期快速发现。两个人对系统的每一部分都很熟悉。由两个大脑来设计决策，且减少"快而脏"的缺点。在两人一组程序设计环境下，代码的质量一直很高。

两人一组程序设计：两个程序员共同进行设计、编码和测试的XP实践。

实践指导

XP使用两人一组程序设计开发高质量的代码，并且把整个编程过程变有效。■

简单设计 反对者说XP忽略了设计，但是这不正确。XP符合敏捷建模规范，其通过避免"开始就有很多设计"的方法。相反，它重视设计以至于可以在很小的部分中也进行设计。同其他方法一样，设计必须伴随着代码和测试的不断检查以保证设计的正确性。

什么是简单设计？它就是用尽可能少的类和方法完成想要的结果，并且不需要重写代码。完成所有的事情经常是一种大的挑战。

重构代码 重构是改进代码的一种技术，而不要改变代码要做的事情。XP程序员不断重构他们的代码，在增加任何功能的前后。XP程序员要检查他们的代码是否实现了用简单的设计和简单的方法表达了同样的结果的目标。重构产生高质量、健壮的代码。

重构：检查，重构和重建系统的一部分以实现更高的质量。

集体所有权 这个实践要求所有的队员都有新思想。在XP中，每个人对代码都要有责任。没有人可以说："这代码是我的"，某人可以说："我写了代码"。每个人都能拥有它。集体的关系允许任何人修改任何代码片段。然而因为在一次变化前后，如果程序员看到某些地方需要变化，他们可以进行单元测试来确定其正确与否。变化没有破坏什么。这个实践支持团队概念，即开发者一起开发一个系统。

持续集成 这个实践拥护XP软件增长的思想。每次或更长时间把通过单元测试的代码集成到系统中。持续集成容易突出错误并且使项目提前。在项目的后期集成大量的代码块的传统方法经常导致大量的重复工作和时间浪费。因为开发者要试着去找出错误。XP持续集成实践避免了这种情况。

现场客户 同其他自适应方法一样，XP项目要求能够对业务功能和范围做出决定的用户参加进来，即有交流的核心价值。这个实践可以使项目提前。如果客户没有准备好为这个项目提供人员支持，这个项目不可能成功。

系统隐喻 这个实践是XP中用来定义一个结构独一无二的有趣的方法。它回答了这些问题：系统是怎么样工作的？它的主要组件是什么？为回答这些问题，开发者要为系统确定一个隐喻。比如Big Three Automaker Chrysler的工资发放系统是通过生产线隐喻建立起来的，其系统组件使用生产线这个术语。Chrysler的每个人都了解"生产线"，因此工资发放交易可以以同样的方式处理——开发者以一个基本的交易开始，然后应用到不同的过程中。当然，隐喻应该容易理解，或者开发团队的人员对此非常熟悉。一个系统隐喻可以引导成员朝着一个版本工作并且帮助他们理解系统。

小发行版 小发行版可以让用户进行功能测试，有时甚至可以进行生产性的使用。与软件增长的整体观点一致，小且频繁的版本为用户提供了升级的解决方案并且使他们包括在项目之中，这也有利于其他实践，比如立即反馈和持续集成。

一周40小时和编码规范 最后两个实践是对开发者应该怎么样工作设置规范。开发者确切的工作时间并不是问题，真正的问题是不能让团队的每一个成员因劳累致死。没有任何项目可以进行随意的编码练习。开发者应该根据标准进行编码和文档化。XP使用以基于经验控制的自适应过程相称的工程规范。

3. XP项目活动

图16-9显示XP系统开发方法的概况。XP开发方法被分为三个等级：系统（外环）、发布（中间环）和循环反复（内环）。系统级活动在每个开发项目中只发生一次，一个系统以多阶段形式交付给用户被称为发布，每次发布的系统是一个能完成系统用户需求子集的完全功能系统。一个发布的系统在至多几周或几个月的时间段内被开发和测试。多个发布系统被分到

多个循环反复过程中。在每一次循环中，开发者编写代码并测试一个发布系统的某一特定功能子集。这些循环经历几次或几周的编码和测试。

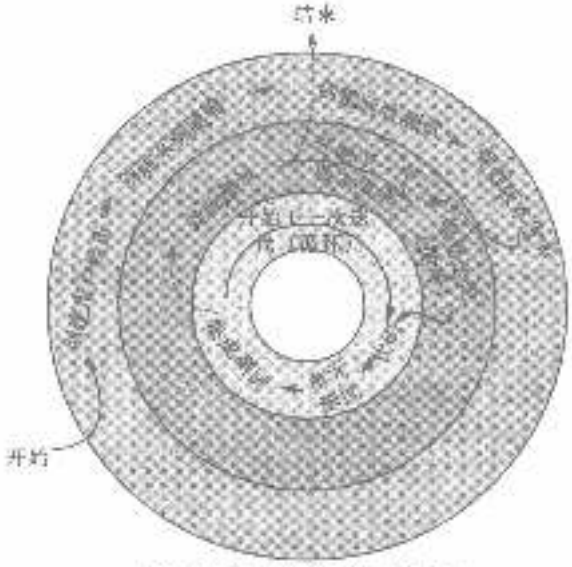


图16.9　极限编程系统开发方法

最早的XP开发活动是创建用户活动，类似于OO分析中的用例。一组开发人员和用户将系统所支持的所有用例进行快速归档。接下来，开发者创建一个类图，代表用户经历中兴趣对象。在XP方法中，类图被称为系统隐喻。

然后开发者和用户为每个用户经历创建一组验收测试。只有经过相应的用户经历验收测试的发布系统才被认为是完成了的。最后的系统级活动是创建以一系列发布系统为中心的开发计划。第一个发布系统支持用户经历的子集，并且后续的发布系统增加对另外用户经历的支持。每个发布系统被交付给用户并执行实际工作，这样就提供了额外的一级测试和反馈。

第一发布级活动是规划一系列循环。每一次循环侧重于一个小的（可能仅仅一个）系统功能或用户经历。循环的规模小，允许开发者在几天之内对其进行编码和测试。一个典型的发布系统是使用几次循环或几打循环开发而来的。

一旦规划迭代完成，发布工作将从第一层迭代活动层开始。代码单元被分配到多个规划小组。每个小组开发和测试各自的代码。XP开发方法提供了首次测试方法去编码。测试代码在系统编码前被编写出来。当编码模块通过单元测试，它们将结合成一个大型单元进行整体测试。（测试工作在第15章中详细讲述过）。当一个迭代工作通过迭代测试后，发布工作将在下一次迭代开始。

当一个发布系统的所有循环都完成以后，发布系统要经受验收测试。如果一个发布系统的验收测试失败，开发小组就会返回到循环级进行修改。通过了验收测试的发布系统将交付给用户，并开始下一个发布系统的工作。当最后一个发布系统的验收测试完成，开发项目就结束了。

16.2.4　Scrum

Scrum是另一种自适应方法。它类似于将世界的橄榄球再拿回来重新进行比赛的系统。橄榄球运动员聚在一起，然后裁判扔出球。Scrum队员用腿把球传递给正在等待的队员。橄榄球运动与系统开发方法之间有很多相似之处：两者都要快、自适应和自组织。Scrum隐含的基本

思想是对当前形势做出尽可能快又准确的反应。Scrum可以被描述为一种基于经验过程控制的软件开发方法。图16-10展示了Scrum软件开发过程。有三个重要的概念来描述Scrum：(1) Scrum理念；(2) Scrum组织；(3) Scrum实践。

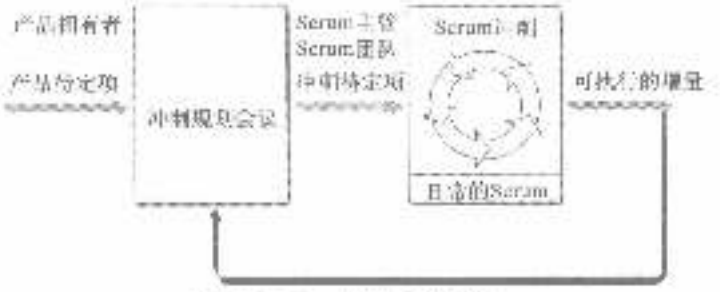


图16-10 Scrum软件开发过程

1. Scrum理念

Scrum理念是基于先前描述的自适应开发规范的。Scrum方法是对频繁变化的动态环境的适应，这样的环境是指用户可能不知道他的确切需求也可能不断改变优先处理的事情，在此环境下，变化如此之多以至于项目不能进行且永远不能完成。Scrum方法尤其适用于这样的环境。

实践指导

在动态程度高的环境中，即用户可能频繁改变优先处理的事情时，考虑使用Scrum方法。 ■

Scrum方法主要关注团队的开发水平。它是一类社会性的工程，其更多的是强调个人能力而非开发过程。并且描述团队的开发者怎么样一起工作来建立软件的一系列子项目。这种观点的关键是在自组织和工作过程中尽全力去控制一个团队。软件是增量式开发的，并且根据经验把控制强加上去——把重点放在能够实现的事情上。

Scrum项目的基本控制机制是应该列出系统所包含的和所涉及的所有事情的清单。这个清单叫做产品待定项，包括用户功能（如用户用例）、特征（如安全）和技术（如平台）。产品待定项清单根据项目和消费者的目前需求不断地被赋予优先权，并且只有少数的优先权高的事务可以同时工作。

产品待定项：Scrum项目中将要完成的用户需求的一个优先权列表。

2. Scrum组织

影响一个项目的三个主要组织因素是：产品拥有者、Scrum主管和Scrum团队。

产品拥有者就是客户，也就是购买开发结果的人。但是，产品拥有者有另外的责任。请注意，在敏捷开发中，用户和客户都要参加到工程中来。在Scrum方法中，产品拥有者维护产品待定项列表。最终系统中包括的任何功能必须在开始就放在待定项中。因为产品拥有者支持那个列表，所以任何需求都要得到产品拥有者的同意。在传统的开发项目中，开发团队首先确定他们的观点和确定其他活动并定义需求，在Scrum项目中，主要的客户控制着需求。这迫使客户和用户一开始就要参加到工程中来。如果产品拥有者没有创建待定项，则什么事情也完不成。

产品拥有者：所建系统的拥有者。

Scrum主管增强了项目实践并帮助团队完成工作。在其他方法中，Scrum主管类似于项目经理。然而团队是自组织的并且没有完整的时间表，所以Scrum主管的责任并不大，和传统项目一样，他的重点是进行交流和制作进展报告。但是Scrum主管并不需要制订时间表和分配任

务，这些事情当团队自己来做。Scrum主管职责之一是扫平障碍，可以使团队顺利工作。换句话说，Scrum主管是一个润滑剂。

Scrum主管：负责Scrum项目的人，类似于项目经理。

Scrum团队是一个开发小组，典型的是5～9个人，他们在一起开发软件。因为项目很大，所以工作要分成几部分，分到各个更小的团队，如果必要，Scrum主管可以促使团队合作。

Scrum团队：Scrum项目工作的团队成员。

Scrum之队在一个具体的时间内，设置一个能完成的目标，然后团队可以自组织并把工作分给成员。一个小团队很容易围坐在一张桌子旁决定需要完成什么需求，并且有团队的成员接受部分的工作。

3. Scrum实践

Scrum实践是一个项目怎么进行的机制。当然，Scrum实践是基于Scrum观点和组织的。其基本的工作过程是冲刺。并且其他所有活动都支持它。

一个Scrum Sprint是实现一个具体功能并在30天内能完成。冲刺开始时，团队人员聚在一起用一天的时间来制订计划。这个时间内，团队决定冲刺的主要目标。这个目标来自产品待定项列表中的几项。团队决定多少个优先级高的事务能在30天内完成。有时，优先级低的事务也被加进来作为额外的任务。

Sprint：一个时间起制的实现一部分具体功能的子项目。

当团队同意一个目标并且从待定项列表中选择好了事务后，就开始工作，然后冲刺的范围就定好了。没有人可以改变它——不管是产品拥有者还是其他用户。如果用户发现新的需求要加入，则添加到待定项列表中，为下一个冲刺准备。如果团队发现不能完成目标中的任务，他们可以缩小冲刺的范围，然而30天的时间是不能变的。

冲刺的每一天，Scrum主管举行一次全队员参加的例会，目的是汇报进度。这个会议只有15分钟或者更短的时间。团队成员回答下面几个问题。

- 上次例会后做了什么（最近24个小时）？
- 下次例会前做些什么？
- 工作中遇到什么困难？

此类会议的目的是简单汇报问题，而不是解决问题。作为每天工作的一部分，会议之后，团队成员要合作来解决问题。Scrum主管的一个主要责任是注意到障碍并看到障碍被清除。一个好的Scrum主管能很快清除障碍，Scrum主管也要为队员排除各种干扰，这样队员就可以自由地工作了。团队成员与用户对话以获得其需求，并且用户也加入到冲刺过程中。然而用户不能改变已经在工作的待定项列表，也不能改变放在待定项列表上的待定项的范围。

在每个冲刺结束时，产出一个可以接受的结果。最后半天来回顾制订的会议和进度，并且确定下个冲刺需要改变的需求。这段时间内的活动——规划、冲刺、日常Scrum会议和Scrum回顾——此过程已成为团队容易遵守的模板，经过良好定义的模板有助于Scrum项目的成功。

16.2.5 项目管理和自适应方法

在本章开始的Valley Regional医院的例子显示，系统开发的自适应方法有时会挫败那些习惯于在项目开始就有一个全面的时间表并按照时间表跟踪进度的主管们。但是自适应方法中的项目开发者为项目创建时间表作为进展。这种方法对那些没有用过的人看来可能是混乱和无法控制的。有些人会认为提倡自适应开发方法就抛弃项目管理，但事实上项目管理是自适应方法不可分割的一部分。如第3章中所说的，项目管理是所有自适应方法不可分割的一部分。

每次迭代可看做自己设置项目管理任务的小组。

实践指导

不管你使用什么方法，好的项目管理技术都很重要。它们是自适应软件方法不可缺少的一部分。

在第3章中，我们也定义了几个项目成功的标准，包括如下所列的清单：

- 清晰的系统需求定义
- 主要用户参与
- 有上层管理者的支持
- 全面和详细的项目规划
- 实际的工作时间表和里程碑事件

我们回顾一下项目管理的8个主要领域的知识，从而看看自适应项目的项目管理是怎么变化的。

在自适应方法中，时间管理有很大变化，因为项目是在不确定的条件下进行的，所以项目团队不能试图制作一个全面、详细地项目时间表。然而我们所看到的是，时间和时间表很难管理。首先，一些方法中，如Scrum方法，每个周期必须一个固定的时间段。在其他方法中，如XP或UP方法，迭代的长度更加灵活。因此对使用自适应方法的项目管理来说，实践管理仍旧是一项很重要的技术。一个重要的成功因素（实际工作时间表）在自适应方法中比在纯粹的预测型方法中更加明显。

项目范围管理也发生了彻底的变化。在预测型开发中，项目经理的主要责任之一是控制项目范围。项目经理最困难的任务是确保需求正确和看到用户的参与，并且阻止范围无限的增长。和这种顶部控制观点相比，自适应方法则是使用户参加到团队中并让他们对项目负责。Scrum方法中的待定项列表是用户负责的。一次迭代或子项目的范围是允许变动的，要通过可控的机制，包括要征得客户同意，项目范围才能改变。然而这种方法潜在的问题是，项目迭代可以无限地进行下去。因此，自适应项目的范围控制包括控制迭代次数。对一个UP项目，细化迭代可以被停止，而团队可以继续去做实现、测试和部署工作。范围控制仍旧是必要的，它有很多不同的方式。

项目时间和范围是相互依存的。项目范围变化影响时间表和要求完成项目的时间。对于一个成功的项目，监测和控制仍旧很关键。在XP迭代或Scrum的子项目中，为保证项目按计划进行，监测和控制仍旧是需要的。例如，Scrum方法，如果项目是自组织的，那么团队成员需要承担项目管理任务。因此，项目仍旧可以用指标来衡量进度并且预测完成日期。

在自适应方法中，成本管理也很重要。项目总成本可能远超预计的要多。因为完成项目的时间未定。执行主管可能对缺少完整的项目时间表和总预算而感到担忧，因此团队需要通过加强迭代控制和范围控制来让他们放心。

因为在自适应方法中用户加到了项目并发的各个方面，所以项目交流管理也是很重要的。因为在每次迭代中有需求要确定、定义、实现和测试，所以定义需要得到用户支持。口头交流和合作工作是定义业务需求的主要方法。对自适应方法，必须要能进行整个团队的交流。

项目质量管理一直是自适应方法的重点，并且事实上有许多工具可以让项目经理和团队来确保一个高质量的系统。有两种主要的技术可以使用：首先，测试贯穿整个过程——从开始写测试计划到检查与集成；其次，给系统重构分配时间，这样可使代码简单、可靠。项目需要尽最确保是够的时间来完成每次迭代中的重要活动。

项目风险管理在自适应方法中加强了。前几次的迭代需要实现系统中高风险的部分。因

此项目团队和客户要尽早发现那些可能破坏项目或导致难以克服的障碍。

和其他方法一样，在自适应方法中，人力资源管理也具有挑战性。在预测和自适应方法中，对的人力资源管理强调能自我管理的小团队。它们的主要的区别是，自适应技术为每次迭代或子项目有内部的为团队自组织的机制。因此在自适应方法中，项目经理很少进行控制。

项目支持管理对所有的项目都要一样的对待，如购买的组件集成到一个项目中，证明采购组件的质量，满足合同内容等问题必须解决。

接下来我们转向新的思想，即在大组织和企业中集成所有系统的开发工作。刚刚讨论过的自适应方法和敏捷概念都着重于一个组织内的单个项目。模型驱动结果着重企业级的活动，而不是单个人的努力。它描述了模型的使用和敏捷建模不同的建模实践。

16.3　模型驱动的体系结构——通用解决方案

UP方法是目前用面向对象模型和工具开发软件的主流方法。中型和大型组织的基本问题之一就是如何构建企业级系统，这些系统能够一致地工作或者至少能够交流。企业组织经常会有遗留下来与基于UNIX系统相耦合，与Windows平台系统耦合的主机系统。每一个这样的系统都有自己的操作系统、交互标准、中间件系统和专门的业务应用系统。今天许多企业组织面临的问题是如何让这些系统一致工作。

中间件包括以下服务。比如消息和电子邮件、HTML服务器、字典和域名服务器、数据库和数据服务器、组建注册、事务和事件处理机制。中间件一般都基于一些准则，这些准则定义了消息模式、消息内容、安全性、应用程序和服务器互相识别的方式。对于一个大的组织来说，独立一个单独的中间件平台是很困难的，因为不同的组织和分配通常有各自的需要，这就导致了它们采用了大量的基于不同准则的中间件方法。今天应用最广泛的一些中间件环境基于这样一些准则，比如CORBA、Java 2阅读服务、XML、SOAP和.NET，这些环境将在后面的章节解释。

系统开发者必须回答的问题是，如何从这些不同的企业级中间件系统中捕获和抽取信息，并会独立地使用它们。换句话说，一个企业如何独立地定义任何一个供销商或平台的体系结构需要呢？

当对象管理集团（OMG）提出的一种方法是模型驱动结构，由多于800个公司和组织的联盟组成的OMG有与语言，平台和卖主独立的互操作标准。一些标准是OMG自己开发和创建的，这样的一个例子是公共对象请求代理结构（CORBA），它描述了企业级界面的标准的通信结构。其他标准是和其他组织一起开发的。在整个内容我们讨论过的UML是有第一个被Grady Booch,Ivcar Jacobson和James Rumbaugh建议作为标准的，它作为标准和纳入OML过程被批准。

模型驱动结构（MDA）是由OMG首先提出的，其建立在抽象、建模、重用的规则和为公司提供用于帮助软件拆解和扩大企业级系统的额外的工具模式的基础上。MDA为在企业中做的所有系统开发工作确定和分类提供组件。MDA的精髓是“为一个变化的世界选择结构”。当我们看到MDA的细节时，你会意识到许多它的概念。MDA有助于描述你在整个课程中学到的概念。

图16-11是一个描述软件开发的典型过程的图表。这个模型应用到预测型过程和自适应、迭代过程中。在预测型过程中，开发团队收集所有的需求，完成全面的设计和编码。在自适应方法中，在多次迭代中，流则重复多次。注意图表右边的矩形。顶部描述性内容的矩形框代表了描述用户需求的文档。这些文档通常是便条、概要、提纲及其他没有组织过的业务过程和用户活动的描述。

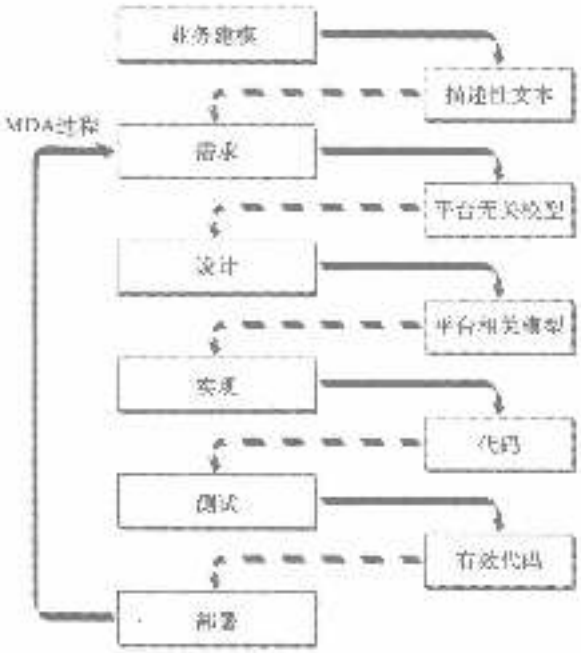


图16-11 软件开发和MDA

图表中的第二个矩形框叫做平台无关模型（PIM），是关于怎么样建立相独立商务模型的信息。一个PIM的例子是UML类图表。一个UML类图表描述系统需求信息，不论它是否使用关系数据库（如SQL）或者分层数据库系统（如IBM的DB2）来完成。

平台无关模型：一个描述对任何部署平台都没有影响的系统特征的模型。

第三个矩形框叫做平台相关模型（PSM），它提供详细的信息，包括计算机平台和实现的具体情况，因此，比如PSM包含一个关系数据模型，其显示了SQL服务数据库中的数据表，主码，外码和单个领域的类型信息。它提供了怎么样实现一个PSM的细节。

平台相关模型：一个描述包括部署平台需求的系统特征的模型。

其他几个矩形框是编码中具体实现的模型。你可能会问，"MDA的作用在哪里？到目前为止，它仅仅是在一些我们已经知道的过程加上一个好听的名字而已。"MDA所做的就是提供一个机制，通过这个机制，我们可以从当前的每个系统中提取关键的特征和信息，也可以把它们联合到一个PIM中。如我们前面所说的，大多数组织有多种运行在不同的平台上和不同编程语言的系统。为描述这些系统，个人的PSM也用到了，且它们都各不相同。PIM中摘取的信息可以使组织来分析联合的PIM去发现技术中哪里存在重复、不一致和冲突。另外，设计新系统需要符合现有的系统。

一个组织使用MDA策略，它首先必须有一个公共的系统模型和描述PIM的语言。PIM建模语言的一个关键组件是UML，我们已经在本书中学过UML。

其他需要的组件是从代码到PSM和从PSM到PIM的传输标准集。实际上，如果可以自动传输，那就更加理想了。然后可以使用自动工具来阅读存在的代码，生成PSM，分析PSM和生成PIM。这些工具都齐好之后，就可以协助转向其他方向——从PIM到PSM到代码。

OMG的目的是帮助定义这些转换的标准，支持这些活动的具体工具的创造，通常是由专门创建和出售工具的业务公司完成的。

图16-12阐述了这个过程怎么发生的。图中显示了不同的元模型的类型。回顾一下前面的

内容。一个元模型是用来描述其他模型的特征的模型，为建立能自动从一个模型到另一个模型转变的工具，开发者需要用精确的数学术语来描述每个模型。这种数学描述是一种叫做元模型的特殊模型类型。

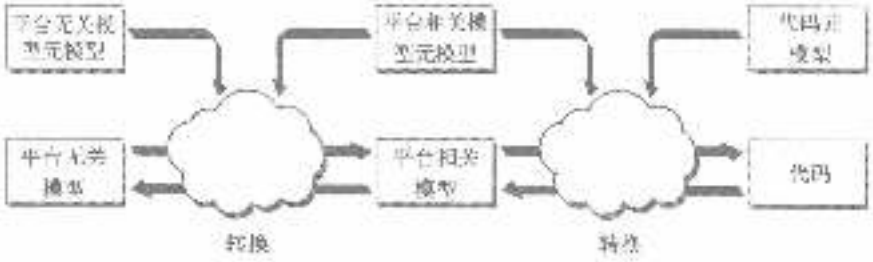


图16-12　元模型和PIM、PSM和代码之间的转换

举个例子，如果我们去描述一个UML设计类图表，我们会说它是由一些框和线组成的。一个框代表一个类。每个框有三个组件，框顶部是类名，中间是一列属性，底部是一些方法。线代表关系，每根线都连到一个框上。每种连接都有表示连接重数的数字。图16-13显示了一个描述UML设计类图表的部分类图表元模型。



图16-13　UML类图表的部分元模型

企业组织可以用每个模型的元模型和定义的转换集来自动转换模型。OMG是到目前为止第一个发布的用于定义这些模型和转换的MDA标准。这种倡议一直在继续，许多组织对标准做出贡献，一些业务公司开始创建自动的工具。

MDA原始定义是为了使用预测型开发方法。自适应方法把文档和建模简化了。那么这两种方法是不兼容或冲突的吗？

MDA主要优点是给出了整个企业级系统的结构一个蓝图。任何新系统需要适合现有的数据和平台配置。用自适应方法开发系统时，和已有的框架保持一致是有意义的。因此，自适应开发方法可以很容易MDA的使用。一个重要的观点是，在合适的条件下应用各种技术来最大化生产和最大化组织的利益。

16.4　框架、组件和服务

许多不同类型的系统都包含相似的功能。例如，在现代软件中，图形用户界面（GUI）就非常普遍，许多GUI的特性，如下拉菜单、帮助屏幕、对象的拖放操作在大多数的GUI应用程序中都有所使用。其他的一些功能，例如查找、排序和简单的文本编辑，对许多程序也相当常见。更高的功能如信用卡认证和装运创作都在许多不同的应用中使用。

重用软件以实现这样的通用功能已经应用了数十年了。但是如果使用老的编程语言，这

些重用将是笨拙和麻烦的。面向对象包括两种强大的技术：框架和组件，这些都支持软件重用。现代网络和标准提供了另外一种软件重用方法——基于网络的服务。

16.4.1 对象框架

对象框架类型是被指定设计用来在各种程序和系统中进行重用的类的集合。对象框架以预编译库或者能在新程序中嵌入可修改的程序源代码的形式提供给开发者。在对象框架中的类有的时候叫做基类。基类被组织成一个或者多个有继承关系的层次。程序员通过从现有的应用程序中派生来开发特定的应用类，然后程序员增加或者修改类的特性和方法使基类的后代满足特定的应用需求。

对象框架：被指定设计用来在各种程序和系统中重用的类的集合。

基类：对象框架中的类。

1. 对象框架类型

针对各种编程需要，人们已经开发了许多对象框架，包括：

- 用户界面类——针对图形用户界面通用对象的类，例如窗口、菜单、工具栏、打开文件和保存文件对话框。
- 通用数据结构类——针对通用的数据结构，例如链表、复数、二叉树，以及像查找、排序、插入删除这类与处理有关的操作。
- 关系数据库接口类——是指一些允许用面向对象编程建立数据库表、向表中增加数据、从表中删除数据或查询表中内容的类。
- 用于特定应用领域的类——针对特定应用领域设计的类，例如银行业、薪水发放、库存管理和货运等。

典型通用目的对象框架主要包括前三个范畴的内容。这个范畴里的类可以在广泛的领域内被重用。特殊应用的对象框架为特定的行业和应用类型提供一个类的集合。特殊应用的对象框架常常是由第三方软件商设计的，作为通用目的对象框架的扩展。一些大的组织已经开始开发自己的应用框架。而小公司通常不这么做，因为这样做所需要的资源十分可观。开发一个应用框架或者公司特定的应用框架通常需要长达几年的努力。这些努力可以通过在未来新系统的开发中重用框架得到回报，也可以通过简化现存系统的维护得到回报。但是，回报是在遥远的将来，与当前创建框架的资金相比，就意义不大了。

2. 对象框架对于设计和实现任务的影响

当决定是否使用对象框架的时候，开发者需要考虑几个问题，也就是说，对象框架在几个不同的方面影响系统的设计和开发过程：

- 框架必须在细节设计开始之前选定；
- 系统结构设计必须符合框架关于应用程序结构和操作的特殊假定；
- 设计和开发人员必须接受培训以便有效地使用框架；
- 可能需要多个框架，尽早进行兼容性和集成性测试。

利用对象框架进行系统开发的过程，从根本上讲就是一个改写的过程。框架为应用程序结构提供了一个模板和一系列具有通用能力的类。系统的开发者改写这些通用的类以适应新系统的特殊要求。必须尽早选择框架以便设计者能够理解框架引入的应用程序结构，知道为了得到所需类需要对通用类进行改写的程度，以及哪些类不能通过利用基本类得到，必须重新构建。

在面向对象的系统开发中典型的有三个对象层次（视图、事务逻辑、数据），视图和数据通常是由基本类派生而来的。用户界面和数据库访问是对象框架得到最强有力支持的领

成，而且要是重新开发也将十分麻烦。据不完全统计，系统80%的代码与视图和数据类有关。因此，从基类中构建视图和数据类将很容易得到代码重用的显著好处。而且从基类中改写视图类也会提供附加的好处，就是可以确保整个系统和系统内应用程序的用户界面之间具有相似性。

框架的成功使用需要大量有关类结构和程序结构的预备知识。这就是说，设计者和程序员在成功地使用框架之前需要对框架十分熟悉。因此，框架应该在项目中尽可能早地选择，开发者必须在使用框架开发新系统之前接受框架使用训练。

16.4.2　组件

除了使用对象框架外，开发者也常常组件来加速系统的开发。组件是一个标准的，可以互换的，已经装配完成的，能随时使用的，定义好的连接客户或其他组件接口的软件模块。组件可以是一个单独的可执行对象，也可以是一个互相作用对象的集合。另外组件也可以是封装在一个面向对象界面里的非面向对象的程序或系统。用非面向对象技术实现的组件仍必须具有对象化的行为特性。换句话说，它们必须实现一个公共的界面以隐藏实现的细节，并对消息进行响应。

组件：标准的、可以互换的、已经装配完成的、能随时使用的、定义好的连接客户或其他组件接口的软件模块。

组件由标准化的可以互换的软件组成。它们与类或对象不同，因为它们是二进制的程序（编译和链接过的），而不是符号程序（源代码），这点区别很重要，因为这使得组件比源代码程序更容易重用和重新实现。

例如，考察一下大多数字处理软件都应用的语法检查功能。语法检查功能可以作为一个对象开发，也可以作为一个子程序开发。字处理程序的其他部分可以通过合适的代码结构（例如，一个C++方法的激发，或者一个BASIC子程序的调用）来调用这个子程序或者对象。在程序的编译和连接后，语法检查功能的源代码就与字处理程序的其他部分集成在了一起。然后就可以把可执行程序交给用户了。

现在考虑早先的语法检查功能可能面对的两种变化：

1. 另一个字处理程序的开发者想把已经存在的语法检查功能集成到他们的产品中；

2. 语法检查功能的开发者发现了可以更正确、更快速地实现这个功能的新方法。

为了把已经存在的功能集成到新的字处理程序中，必须向字处理程序的开发者提供语法检查功能的源代码，然后开发者在他们的字处理程序中正确地调用语法检查功能，最后再把这个混合的程序编译连接后交给用户。

那么，这样的情景有什么问题么？理论上是没有的，但是在实际操作的时候问题却很多。语法检查程序的开发者只能通过源码向其他开发者提供他们的功能模块，这样就涉及了许多潜在的知识产权和软件侵权的问题。更为重要的是，字处理程序的开发者为了要升级内嵌的语法检查程序，就必须重新编译、重新连接整个的字处理程序，然后再把修改后的二进制程序发送给用户，安装到他们的计算机上。这是相当昂贵和费时的过程。而发送二进制的语法检查程序将消除或者最大程度地减少这些问题。

基于组件的软件设计和构造方法可以解决这两个问题。像语法检查程序开发者这样的组件开发人员能够以现成的二进制组件的方式提供他们的产品中。像字处理程序开发者这样的使用者能够简单地把组件插入他们的产品中。升级一个组件不需要重新编译、重新连接、重新发送整个应用程序。很有可能出现的情况是，已经安装在用户计算上的应用程序可以在每次运行开始的时候通过互联网查询升级站点，然后自动地下载、安装已升级的组件。

以这种观点来看，可以认为基于组件的开发是另外一种形式的代码重用。但是结构化设计、面向对象的继承方式、客户-服务器体系结构都以不同的方式进行代码重用，基于组件的设计和构造与众不同之处如下。

- 组件是具有可执行的代码单元的可重用软件包，重用源代码的方法是结构化设计和面向对象的继承。
- 组件作为可执行对象，公布一个公共接口（也就是说，一系列的方法和消息），对其他的组件隐藏（封装）它们方法的实现。它的客户-服务器体系结构不必依赖于面向对象的原理。基于组件的设计和构造是客户-服务器体系结构向纯粹的面向对象形式的演化。

组件为系统的设计和构造提供一个非常灵活的方法。仅仅通过获取和插入一个适当的组件集合，开发者就可以设计和构造一个新系统的许多部分。把功能、程序和系统当做组件的集合来设计和构造将使这些部分的实现更具灵活性。基于组件的分析和设计方法在实体物品（例如，汽车、电视和计算机硬件）的生产中作为规范已经有几十年了。然而，作为一种设计和实现信息系统的可行方法，组件技术才刚刚开始出现。

16.4.3 组件标准和基础结构

组件的互操作性需要开发相应的标准。例如，考虑一个典型的IBM兼容机的显示器，显示器的电缆末端的插头要遵循标准接口，插头上每一个连接部分都承载着一定类型的已定义电信号。很多年前，计算机和显示器生产商定义一个描述插头的物理形式以及每一个连接器承载的信号类型的标准。有了这个标准作保证，任何显示器都可以和兼容计算机协同工作。

组件需要标准的支持机构。例如，显示器单元自己本身不具有电能。因此，它们不仅需要标准的电源插头，也需要一个基础结构为插头提供电能。组件还需要从基础结构获得特定的服务。例如，一个蜂窝电话需要蜂窝电话服务提供商为其安排特定的传输频率与临近的蜂窝无线塔进行传输。当用户移动的时候，还需要在无线塔之间传递信号，并且需要建立与其他用户的连接，通过公共电话网中继从其他人的电话传来的话音。所有的蜂窝电话都需要这些服务。

软件组件对于标准也有相似的需求。组件可以硬性地连接在一起，但是这样就失去了灵活性。只有当组件可以通过标准的基础结构去动态地发现其他组件并与它们建立连接的时候，灵活性才有可能增强。

在最简单的系统中，所有的组件都在同一操作系统的控制下，在一台计算机上运行。当组件位于使用不同操作系统的不同的计算机上，并且可以从一个位置移动到另一个位置的时候，连接就更复杂了。在这种情况下，需要一个独立于硬件平台和操作系统的网络协议。事实上，即使所有的组件都是在一个计算机上运行的，使用一个网络协议也是值得的，因为这样可以保证这个系统应用在不同的环境下——从单个的计算机到计算机网络。

现代的网络标准已经在很大程度上解决了使用通用硬件和通信软件连接组件的问题。因为网络互联协议几乎已经是通用的标准，所以它为组件之间传递消息提供了一个现成的方法。互联网的标准也被用来为同一台计算机上运行的两个进程传递信息。然而，单独使用互联网标准还不能充分满足组件连接的需要，其缺少的部分是：

- 合法的消息和响应的格式定义与内容定义；
- 唯一确定互联网上的每一个组件，以及组件之间消息路由的选择方法。

为解决这些问题，一些机构已经开发或继续修改组件开发和重用标准。

1. CORBA

通用对象请求代理结构（CORBA）是由对象管理组（OMG）开发的，对象管理组

（OMG）是计算机软件和硬件供应商的联合组织。CORBA是作为一个标准平台和独立语言标准设计的，CORBA标准的核心是对象请求代理（ORB）和用于组件通信的internet ORB间协议。组件用户连接ORB服务器，定位组件并明确其能力和界面需求。在组件和用户间发送的消息通过ORB来确定发送路线，此处，ORB执行一些必需的转换服务。

通用对象请求代理结构（CORBA）：是由对象管理组织（OMG）开发的用于软件组件连接和交互的标准。

对象请求代理（ORB）：一个提供组件目录和通信服务的CORBA服务。

internet ORB间协议（IIOP）：一个用于对象和对象请求代理器通信的CORBA协议。

2. COM+

组件对象模型+（COM+）是由微软开发的组件应用性标准。它被广泛地应用于Windows下的应用软件，以及基于微软互联网信息服务器（IIS）和事务服务器（Transaction Server）的三层分布式应用中。大多数微软办公系列，如微软办公自动化系统，就是通过一组COM+协作而构成的。

COM+组件是在Windows注册表中通过个人计算机系统来注册的，此注册表把COM+组件限制在微软操作系统中运行。一旦组件通过注册表相互定位，它们就可直接通过网络协议或微软内部进程通信设备进行通信。

组件对象模型+（COM+）：由微软开发的软件组件连接和交互的标准。

3. 企业级JavaBeans

Java是由Sun Microsystems公司开发出来的面向对象程序设计语言。大多数人已经听说Java与能在网页上运行的applet小程序有关，Java与其他面向对象程序设计语言几点重要差别在于：

- Java程序可以被编译成对象代码文件，这种文件能够在多种硬件平台上的多种操作系统下执行；
- Java语言标准包含一个扩展对象框架，被称为Java开发工具包（JDK），jdk中包含着涉及gui、数据库操纵和网络互连方面的类。

JDK定义了许多类以及支持组件开发的命名惯例。其中一个类能够使Java对象从其本身内部状态转换为能够存储或在网络上传输的字节序列。其他类允许组件枚举Java对象的内部变量，命名惯例允许组件按惯例给出操纵变量的方法的名字。实现了所有需要的组件方法并同时遵循方法命名惯例的类对象被称为JavaBean。

JavaBean：实现了所有需要的软件方法并同时遵循了作为JavaBean标准的方法命名惯例的类对象。

企业级JavaBean（EJB）：可以在服务器上执行并利用CORBA与客户及其他组件进行通信的JavaBean对象。

一个企业JavaBean对象是可以在服务器上执行并利用CORBA与客户及其他组件进行通信的JavaBean对象。EJB对象在JavaBean对象之上提供的额外能力包括：

- 多组件事务管理
- 多个组件打包成较大的运行单元
- 在关系型和对象数据库管理系统中存储和检索复杂对象
- 组件和对象访问控制

16.4.4　组件和开发生命周期

购买和重用组件是加速系统开发的一个可行方式。涉及组件的开发情景有以下两种：

- 使用购买的组件形成一个新开发的或者重新实现的部分或全部系统
- 组件可以在新开发或者重新实现系统中进行设计和配置

每一种情景对系统开发有不同的实现方式，下面分别讨论：

1. 购买的组件

组件会改变项目的建立活动，因为它们影响系统实现的方法。购买和使用组件一般比构造等价软件更加便宜并且花费时间较少。购买的组件还能解决开发者不容易解决的或解决起来代价高昂的技术问题。

寻找合适组件的工作必须早在分析阶段就开始，但是寻找工作必须在开发人员已经明确了用户需求，能够评价用户需求与组件能力是否匹配的时候才能启动。当开发者购买整个软件包时，组件能力与用户需求之间很难不匹配精确。这样，开发者可能需要在可利用组件的基础上进一步明确用户需求，尤其是在开发项目时间很短的时候。

组件是在基于诸如CORBA和EJBs等标准这样的扩展性基础结构中进行操作的。许多系统软件包实现了每一标准的关键部分。这样，选择一个组件并不是简单地选择单独的应用程序模块。开发者必须还要选择兼容的硬件和系统软件来支持组件。

购买组件对某一特定基础结构的依赖性对于开发活动来说有几种实现方法，包括：

- 购买组件所需要的标准或支持软件必须成为技术需求定义的一部分；
- 组件的技术支持需求限制了体系结构设计时的选择；
- 提供组件服务的硬件和系统软件必须在实现阶段的测试活动开始之前就已经被获取、安装和配置；
- 组件及其支持基础结构必须在系统交付后得到维护。

许多开发项目，尤其是大型开发项目，可能使用来自于不同厂商的组件，这将会引起兼容性问题。组件寻找和选择过程必须仔细考虑兼容性问题，因此经常会去掉某些选择或改变对其他选择的期望。可能不得不在系统分析阶段末尽增加初步的测试活动，以便在应用组件及其支持结构完成系统体系结构设计之前验证组件的性能和兼容性。支持和维护也更加复杂，这是因为系统的重要部分并不在系统所有者或内部IS人员的控制之下。

2. 系统性能

基于组件的软件通常配置在一个分布式的环境中。组件通常零落地分布在客户机和服务器上，遍布整个局域网和广域网。遍布计算机和网络的组件带来了一些性能方面的问题。系统的性能依赖于组件所处的位置（即组件拓扑）、组件所处地点的计算机硬件能力、连接计算机的网络通信能力。而且系统的性能也依赖于其他应用和诸如电话、电视、传统的客户与服务器的交互这样的通信对系统的需求。

分析和控制计算机与网络性能的细节已经超出了本书的范围。但是当计划构建和实施一个分布式的组件系统的时候，开发者应该意识到性能问题。这些问题必须在系统的设计、实施和支持阶段仔细地考虑。

开发者应该采取的确保系统具有良好性能的步骤如下：

- 检查基于组件的设计以评估网络模式以及对计算机硬件的要求；
- 检查已有服务器和网络设施能力以决定它们是否有能力承担组件间的通信；
- 在开发和测试之前升级网络和服务器的能力；
- 在开发阶段测试系统的性能，做出必要的调整；
- 在系统安装之后持续监视网络性能以检查出现的问题；
- 重新调整部件、升级服务器能力和升级网络能力以面对变化的条件做出反应。

实现这些步骤需要对计算机和网络技术有全面的了解，并且熟悉已经存在的应用、通信

需求、基础设备的能力和配置。在实际中应用这些知识解决问题是一个非常复杂的任务，通常由经过良好训练的专家完成。

16.4.5　服务

Internet和快速网络时代使得有不同的名字可以描述软件重用的新方法，包括网页服务和面向服务的结构（SOA）。组件在被编译的时候插到一个应用中或者在执行之前组件被动态或者静态的连接到应用中。和组件不同，应用是在执行期间通过Internet或私人网络和服务连在一起的。与对象框架和组件一样，服务依赖于一系列的标准，这些标准对软件设计、开发和执行有很重要的作用。

1. 服务标准

服务标准已经从如COBAR和EJB等分布式对象标准演变到如SOAP、.NET和J2WS等标准。服务标准和早期的分布式对象标准主要差别是：必须编译和连接到一个可执行应用的信息量的减少及更多的依赖于基于网页的数据交换标准，如XML。

简单对象访问协议（SOAP）是基于已有的网络协议的服务标准，包括超文本传输协议（HTTP）和可扩展标记语言（XML）。对象之间的消息被编码到XML中，使用HTTP传输，这使得对象可以放置在网络的任何位置。因为SOAP组件使用XML通信，所以它们可以很容易地被嵌入到使用网页浏览器界面的应用中。复杂的应用可以通过由网络通信的多重SOAP组件来建立。

简单对象访问协议（SOAP）：通过Internet使用HTTP和XML通信的组件标准。

图16-14显示了一个应用和使用SOAP消息通信的服务。SOAP的编码/解码器和HTTP的连接管理都是一个SOAP程序员工具中的标准组件。应用也可以嵌入到使用网页服务来提供SOAP消息服务的脚本中。

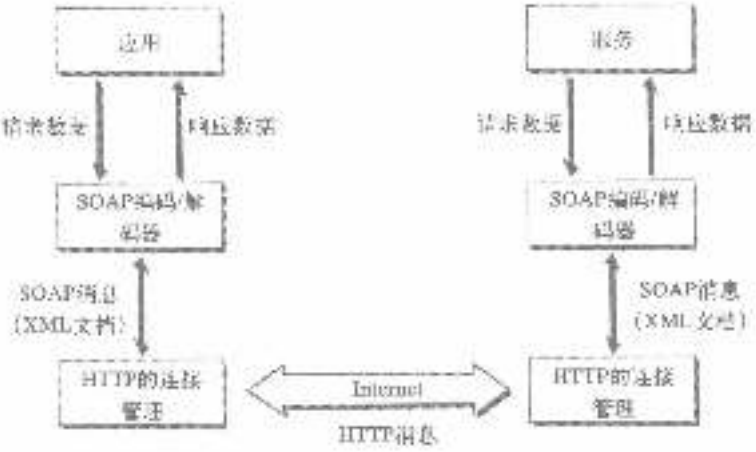


图16-14　使用SOAP通信

Microsoft.NET是基于SOAP的服务标准。.NET应用和服务通过SOAP协议和XML消息通信，这些应用和服务安装在微软的网页/应用服务上并且依赖于Microsoft's Active Directory的各种命名、位置和安全能力。这些应用和服务可以用多种编程语言开发，包括VB和C#。

Microsoft.NET：基于SOAP的微软的服务标准。

Java 2网页服务（J2WS）是用Java实现的应用和服务。J2WS扩展了SOAP和其他几个标准，定义了在应用和服务之间实现通信的特定的Java方法。尽管Java只是一种编程语言，它支

持如网页服务等基础组件和非专利安全软件。

2. 服务和开发生命周期

服务对开发生命周期的影响与对象框架和组件的影响是并行的。

- 额外的服务必须在项目的早期确定——它们的实现细节制约着后面的设计和开发任务；
- 服务标准和基础设施必须在项目早期选择好；
- 应用和服务设计必须与关于服务标准施加的程序结构和操作的特定假设一致；
- 设计和开发人员被训练成能有效地使用服务标准；
- 开发者必须认真考虑网络设计和应用/服务组件部署以确保足够的性能和安全性。

小结

本章的重点是系统开发的新方法、新技术。本章开始回顾5个重要的软件开发原则：抽象、模型与建模、模式、重用和方法。这些原则组成了面向对象开发的基础。系统开发者使用这些原则发明新的、唯一的方法来开发系统。

软件开发中重要的活动趋势之一是自适应方法。统一过程开发方法（UP）就是一个自适应方法的例子，它也影响着其他演进的方法。自适应方法背后的思想是软件项目需要敏捷和灵活，因为商务世界是不可预测的和快速变化的。"敏捷软件开发宣言"描述了4个软件项目的原则：

- 快速响应变化高于按计划进行
- 个体和交流凌驾于程序和工具
- 执行软件凌驾于综合文档
- 用户合作凌驾于合同协商

敏捷开发中另外一个核心概念是敏捷建模。它建议怎么建造模以及建模需要在开发项目中进行的指导思想。这种方法的根本是要记住：模型只是一种方法而不是最终目标。因此敏捷建模观点中，建模是一种工具——比如为了理解用户需求或设计一个具体的功能——而不是很重要的、详细的正式图表。

具体的开发方法，如极限编程和Scrum方法，都支持自适应的原则。XP的核心是要首先编写系统测试代码，程序员要结伴工作来设计、编码和测试软件。因此当一个功能被完成时不仅是设计和编码，也要经过核查和测试。

Scrum方法定义一个在4周内能完成的目标。在4周的冲刺中，项目团队被保护起来不受外部影响，从而可以完成既定的目标。所有重要需求的产品列压出客户维护，只有在两个冲刺时间之间才能改变团队的工作。

模型驱动结构（MDA）是OMG的一个建议，它为庞大的组织集成所有的软件和企业的所有软件开发提供了技术。这一点上，模型驱动结构是原则规范和思想的一个集合。要使用MDA，就要有工具开发商开发出具体的方法。MDA定义不同层次的模型，包括平台无关模型（PIM）和平台相关模型（PSM），它们可以为企业级的系统开发提供一个全面的观点。MDA是一个新开发都要做的一个框架，因此组织就可以去维护一个集成的、一致的操作环境。

软件重用是加速开发的基本方法。它已经有了很长的历史，目前由于面向对象编程、对象框架以及基于组件的设计和开发的出现，软件重用技术获得了更大的成功。对象框架技术可以通过继承来重用已经存在的软件。对象框架提供可重用的源代码库，并且可以用继承的方式把它们快速改变成适应新的应用需求和操作环境的代码。

组件是可重用的以分布对象方式运行的可执行代码单元。它们可以被插入已经存在的应用程序中，或者捆绑在一起形成一个新的应用程序。像软件重用的概念一样，组件的概念也

不是新提出来的。但是支持基于组件的应用的标准和基础结构却是最近才出现的。因此，组件目前刚刚开始进入软件开发技术的主流。

关键术语

chaordic	混序
Common Object Request Broker Architecture(CORBA)	公共对象请求代理结构（CORBA）
component	组件
Component Object Model Plus (COM+)	组件对象模型+
discipline	规范
Enterprise JavaBean	企业Java组件类对象
foundation classes	基础类
Internet Inter-ORB protocol(IIOP)	Internet ORB间协议
Java 2 Web Services(J2WS)	Java 2 Web 服务
JavaBean	Java组件类对象
metamodel	元模型
Microsoft.NET	微软.net
object framework	对象框架
object request broker(ORB)	对象请求代理
pair programming	两人一组程序设计
Platform-independent model(PIM)	平台无关模型
platform-specific model(PSM)	平台相关模型
product backlog	产品待定项
product owner	产品拥有者
refactoring	重构
Scrum master	Scrum主管
Scrum team	Scrum团队
Simple Object Access Protocol(SOAP)	简单对象访问协议
sprint	冲刺
ubiquitous computing	普适计算

复习题

1. 确认软件开发中驱动目前形势的5个重要原则和实践。对每一种作简要的解释。
2. 试探许多公司使用自适应方法开发的驱动因素是什么？
3. 解释预测型控制过程和经验型控制过程的不同。
4. 列出自适应项目的6个主要特征。
5. “敏捷开发宣言”的内容是什么？对每一条做出解释。
6. 混序是什么意思？在开发项目中混序的作用是什么？
7. UP的4个过程并指出每个过程的对象是什么？
8. UP开发的6个规范是什么？
9. UP的三个支持规范是什么？
10. 列出敏捷建模的主要规范。
11. 为什么极限这个词包括在极限编程的名称里？

12. 列出XP的核心价值。
13. 列出XP实践。
14. 在Scrum项目中的产品待定项是什么？
15. 解释Scrum冲刺是怎么工作的。
16. 解释使用敏捷方法时项目时间管理和项目范围管理的差别。
17. PIM是什么？PSM是什么？它们之间的关系是什么？
18. 模型驱动结构潜在的好处是什么？
19. 元模型是什么？元模型是怎么使用的？
20. 对象框架是什么？它与组件库有什么不同？
21. 面向对象的哪几层是最有用的组件？
22. 什么是软件组件？
23. 为什么软件组件直到现在才广泛使用？
24. 组件通过什么方法使得软件的构建更快？
25. 什么是服务？服务和组件的不同是什么？服务和组件的相似点是什么？
26. 面向服务结构基础基于什么样的标准？

思考题

1. 分析在你最近的编程和软件开发中使用的编程语言和开发工具的功能。它们能够在个人计算机上完成单用户软件的发展型原型开发吗？它们有能力在多用户、分布式、面向数据库和高安全性操作的环境中实现发展型原型开发吗？如果它们被应用于基于工具的开发，什么样的需求由于不符合语言和工具的能力而将被牺牲？
2. UP是由Rational公司首先开发出来的，现在Rational公司已被IBM收购。在IBM的网站上，可以通过IBM/Rational找到关于UP工具的任何信息。浏览IBM的网站和其他网站（如敏捷建模网站），对UP和敏捷建模进行比较。报告你的发现。
3. 考虑XP方式的一般性的基于区队程序设计方法以及它所采用的原则，即允许任意程序员在任何时候修改任意代码。其他开发方式和程序设计管理技术都不遵循这一原则。为什么？换句话说，这项原则有可能存在哪些消极含义？XP方式如何使消极含义最少？
4. 访问敏捷联盟网站（http://www.agilealliance.com/home）和敏捷建模网站（http://www.agilemodeling.com），找一些有关敏捷开发中的项目管理文章。确定在敏捷开发中你认为项目管理比传统方法中更困难的几点，同时找出敏捷项目中项目管理更容易的几点。
5. 本章讨论了使用敏捷开发技术的好处。列出并解释不适合采用的敏捷开发技术（如XP和Scrum）的条件。
6. 访问联盟网（http://www.omg.org），并评论关于SOAP标准的最新进展。考虑可以添加何种新性能，以及新性能对于SOAP标准的复杂性和基础结构需求有什么影响。
7. 根据以下几方面比较对象框架和组件与服务结构：系统安装前修改的难易、系统安装后修改的难易，代码重用节省的费用。哪一个方法对于单一的应用系统（例如某个公司的的分布式管理系统）可以获得更大的收益？哪一个方法对于通用的软件系统（例如电子表格或者防病毒程序）可以获得更大的收益？
8. 分析计算机硬件的基于组件的设计与结构（例如个人计算机）和计算机软件的基于组件的设计与结构有什么相似之处和不同之处？计算机软件可以获得像计算机硬件一样的插拔兼容性吗？你的回答是否依赖于软件的类型（例如，系统或者应用软件）？计算机硬件和软件预期寿命的差异会影响基于组件技术的应用性和价值吗？

实验练习

1. 与你所在学校的同学或者公司的同事谈谈一个最近由于开发缓慢而被迫放弃的项目。这个项目采用了什么样的开发方法？一种不同的开发方法可以使开发过程加快吗？
2. 找一个你身边的用UP（UP的变体也可以）和一些其他自适应方法作为开发方法的公司。学习他们是怎么使用UP和怎么使用UP规范与实践的。
3. 找一个你身边的在使用敏捷规范开发软件项目团队中工作的人。他们是怎样进行使用敏捷开发的团队训练的？这个组织是怎么样采纳这种方法的？成功的感觉是什么？他发现比较难使用的方面有哪些？
4. 考虑使用一个利用现代特性的系统（例如，基于网页的界面、学位进度的即时报告、长期学位计划课程自动注册）代替目前学校使用的学生指导系统。现在考虑怎样的一个系统怎样应用基于工具的方法实现。考察备选工具，如Visual Studio、PowerBuilder和Oracle Forms。如果做为开发决定的原因，决定（对每一个工具）哪些需求将要被放弃。
5. 检查一个现代编程环境的能力，例如Visual Studio .NET, IBM WebSphere Studio, Borland Enterprise Studio的能力。它提供了引象框架或者组件库了吗？成功地使用开发环境是否需要一种特殊形式开发途径？成功地使用开发环境是否需要一种特殊的开发方法？
6. 检查一个复杂的终端客户软件包，例如Microsoft Office。在哪些方面（如果有的话），软件使用了基于组件的软件开发？

实例研究

Midwestern Power Services（MPS）

Midwestern Power Services（MPS）为中西部的4个州提供天然气和电力服务。像大多数的能源企业一样，MPS在未来的几年里将要面临联邦和州对其进行的重组。联邦对这个行业的重组已经初露端倪，但是对于未来有什么规范或者限制还没有定型，MPS服务的两个州对于这个行业的重组已经开始立法。另外两个州的立法工作不久也将展开。工业将经历一场大动荡，是由几家California的电力公司电力短缺引起的重大问题，可能会突然的崩溃。这些规范严重影响着所有的业务领域，包括会计、记录保管、电力购买、分销协议，以及客户消费和计费。

重组方案总要仅仅通过监控分销部分来提高电力和天然气领域的竞争，最终的重组形式还没确定，它的具体结构可能因州而异。

MPS想要为重组而在修改系统方面带头。直接受到影响的有二个系统：一个是采购大宗天然气的系统，一个是采购大宗电力的系统。最后一个是针对同时需要天然气和电力服务的客户计费系统。计费系统目前没有把供应和分销责任进行分离，与天然气和电力的购买系统也没有直接的联系。MPS的总账系统也要受到影响，因为过去它已计算MPS自己的发电业务。

MPS决定重新构建它的财务、购买、计费系统以符合重组提案的框架：

- 客户计费系统将要明确天然气和电力的供应和分销费用，供应价钱将要由能源的批发商决定，收入将要按给对应的公司（例如分销部分归MPS，供应部分归批发商）；
- 为批发供应商建立一个新的计费系统以获得每个客户的收入，然后客户通过MPS向供应商付费，日常的付款将要依照客户目前的付费情况自动完成；
- MPS自己的发电业务将要被重构为一个与其他能源供应商相似的单独的利润中心，从选择MPS作为其电力供应商的客户那里得来的收入将与发电成本相配。

MPS目前的系统完全是内部开发的。总账系统和天然气采购系统是基于主机模式的，开发于20世纪90年代中期，从那时起已经做了很多的改变。所有的程序都是使用COBOL语言写成的，使用

DB/2数据库（一个关系型数据库）存储数据和进行数据管理，共有大约50 000行的COBOL源代码。

计费系统在20世纪90年代中期曾经被重新编写过，从那时起，做过特别修改。这个系统运行在一系列使用UNIX操作系统的服务器上。使用一个最新版本的Oracle数据库（一个关系型数据库）进行数据存储和数据管理。虽然使用了一些C语言和Oracle Forms，但是大多数的程序是用C++语言写成的，共有大约80 000行C和C++代码。

MPS有一个网络，它主要用来支持终端对主机通信、互联网连接，并支持微型计算机进行打印机共享和文件共享。计费系统依赖这个网络进行服务器间的通信。支持计算系统和购买系统的主机也连接在这个网络上，虽然该连接主要是用来备份数据和软件到远端计算机。公司已经通过浏览器界面进行电话客户支持和账单报表的实验，然而还没有实现或者安装过功能强大的基于网页的系统。

MPS目前处于系统升级规划的早期阶段，还没有决定应用那些技术或者开发方法，也没有决定是升级一些单独的部分还是替代整个系统。完成所有系统修改的目标时间是从现在开始的3年时间，但是公司正在寻找方法以缩短整个进度计划。

1. 描述使用UP、XP和Scrum开发方法进行系统升级或者开发新系统的优缺点。如果不进行系统重构而只进行系统升级，优缺点会发生改变么？优缺点会随着系统变化而变化么？如果会，对于每一个系统该使用什么样的开发方法？

2. 基于组件的开发方法适用于任何系统的开发吗？如果是，什么样的工具最合适。对于每一种工具，指出由于工具能力的局限哪些需求将要被牺牲。

3. 假定定制开发的软件将会替代所有的系统。对象框架对于完成这次替代有价值吗？可以获得一个来自第三方的特殊应用框架吗？为什么？

4. 假定定制开发的软件将会替代所有的系统，应该使用基于组件的设计和开发吗？为什么？MPS有足够的技术和人员去实现基于组件的系统吗？如果不能，缺少什么样类技术和人员？

对落基山运动用品商店实例的再思考

现在，你已经学习了本书的内容，你可以根据学到的开发方法和技术知识为RMO公司的客户支持系统（CSS）做出更深一次的选择。回顾一下图3-8中给出的CSS开发项目图，以及第2章和第3章末尾的“RMO的再思考”部分。你可能还需要看一下从第2章和第3章中有关RMO的其他材料，然后回答以下问题：

1. 考虑一下本章讨论的在系统开发时自适应开发方法的选择问题。CSS项目的什么特征适应于预测开发方式？什么特征适应于UP开发方式？什么特征适应于更灵活开发方式？哪一种方法最适合CSS开发项目？

2. RMO应该为新的CSS购买组件吗？如果购买，应该在何处寻找组件？使用组件的决定如何影响分析、设计和实现阶段？如果使用了购买的组件，内部构造的系统部分也应该按照组件结构方式来构造吗？决定追求基于组件的设计和开发方式，必须采用面向对象的分析和设计方法吗？

关注Reliable Pharmaceutical Services

再研究第2章和第3章的Reliable Pharmaceutical Services案例，结合用本章所学的知识，回答以下问题：

1. 本章描述的开发方法中，哪一个方法最适合该项目？为什么？用你选择的方法，制订项目的第一个6周计划。

2. 在为Reliable所开发的系统中，组件扮演什么角色？基于哪一种组件相关的标准有影响吗？为什么？

参考资料

Agile Alliance Web site. http://www.agilealliance.com/home.

Scott W.Ambler, *Agile Modeling: Effective Practices for Extreme Programming and the Unified Process*. John Wiley and Sons Publishing, 2002.

Ken Auer and Roy Miller, *Extreme Programming Applied:Playing to Win*. Addison-Wesley Publishing Company,2002.

Kent Beck, *Extreme Programming Explained: Embrace Change*. Addison-Wesley Publishing Company, 1999.

Ivar Jacobsen, Grady Booch and James Rumbaugh, *The Rational Unified Process*. Addison-Wesley, 1999.

Anneke Kleppe, Jos Warmer and Wim Bast, *MDA Explained: The Model Driven Architecture: Practice and Promise*. Addison-Wesley Publishing Company,2003.

Craig Larman, *Agile and Iterative Development: A Manager's Guide*. Addison-Wesley Publishing Company, 2004.

Scott M.Lewandowski, "Frameworks for Component-Based Client/Server Computing." *ACM Computing Surveys*, volume (1998 3, 30(1):3-27).

"Manifesto for Agile Software Development." the Agile Alliance. http://agilemanifesto.org.

Pete McBreen, *Questioning Extreme Programming*. Addison-Wesley Publishing Company,2003.

Stephen Mellor, Kendall Scott, Axel Uhl, and Dirk Weise, *MDA Distilled: Principles of Model-Driven Architecture*. Addison-Wesley Publishing Company,2004.

Ken Schwaber and Mike Beedle. *Agile Software Development with Scrum*. Prentice-Hall, 2002.

Steve Sparks,Kevin Benner, and Chris Faris." Managing Object-Oriented Framework Reuse." *Computer*, volume (1996-9, 29(9):53-61).

经典推荐



算法导论（原书第2版）



编译原理（第2版）



自动机理论、语言和计算导论（原书第3版）



分布式系统：概念与设计（原书第4版）



数据库系统概念（原书第5版）



软件工程：实践者的研究方法（原书第8版）

延伸阅读



计算机组成与设计
硬件/软件接口（原书第3版）
作者：David A.Patterson,John L.Hennessy
ISBN：978-7-111-20214-1
定价：75.00元



计算机体系结构：
量化研究方法（英文版·第4版）
作者：John L. Hennessy, David A. Patterson
ISBN：978-7-111-20378-X
定价：79.00元



现代操作系统(第2版)
作者：Andrew S. Tanenbaum
ISBN：978-7-111-18511-X
定价：55.00元
第3版将于2009年出版



编译原理第2版
作者：Alfred V. Aho,Monica S.Lam,
Ravi Sethi,Jeffrey D. Ullman
ISBN：978-7-111-25121-7
定价：89.00元

好书推荐

















CENGAGE Learning

Supplements Request Form (教辅材料申请表)

Lecturer's Details (教师信息)			
Name: (姓名)		Title: (职务)	
Department: (系科)		School/University: (学院/大学)	
Official E-mail: (学校邮箱)		Lecturer's Address / Post Code: (教师通讯地址/邮编)	
Tel: (电话)			
Mobile: (手机)			

Adoption Details (教材信息) 原版□ 翻译版□ 影印版 □			
Title: (英文书名) Edition: (版次) Author: (作者)			
Local Puber: (中国出版社)			
Enrolment: (学生人数)		Semester: (学期起止日期时间)	

Contact Person & Phone/E-Mail/Subject:
(系科/学院教学负责人电话/邮件/研究方向)
(我公司要求在此处标明系科/学院教学负责人电话/传真及电话和传真号码并加盖公章)

教材购买由 我□ 我作为委员会的一部分□ 其他人□姓名： 决定。

Please fax or post the complete form to (请将此表格传真至):

CENGAGE LEARNING BEIJING
ATTN : Higher Education Division
TEL: (86) 10 82862096/95/97
FAX : (86) 10 82862089
ADD: 北京市海淀区科学院南路2号
融科资讯中心C座南楼12层1201室 100080

Note: Thomson Learning has changed its name to CENGAGE Learning