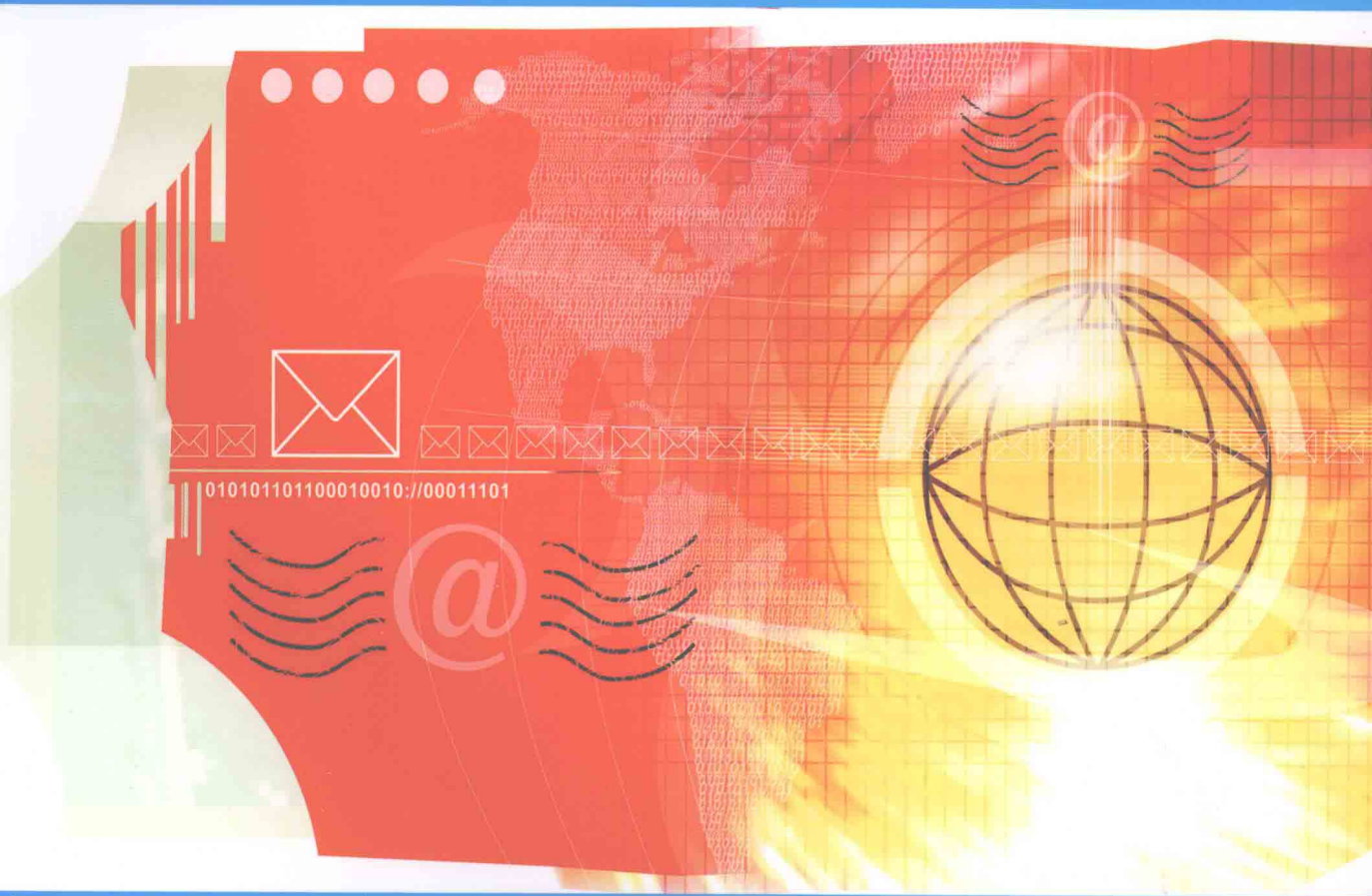
新世纪高职高专
软件专业系列规划教材

软件设计基础

新世纪高职高专教材编审委员会组编
赵从军　钟闰禄　编著



大连理工大学出版社

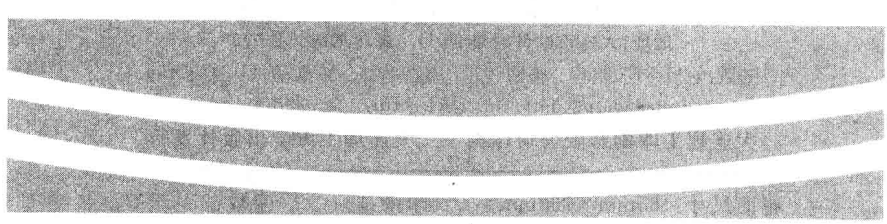
新世纪

新世纪高职高专软件专业系列规划教材

软件设计基础

新世纪高职高专教材编审委员会组编

赵从军　钟闰禄　编著



RUANJIAN SHEJI JICHU

大连理工大学出版社
DALIAN UNIVERSITY OF TECHNOLOGY PRESS

图书在版编目(CIP)数据

软件设计基础 / 赵从军，钟闰禄编著. —大连：大连理工大学出版社，2008.10
新世纪高职高专软件专业系列规划教材
ISBN 978-7-5611-4197-7

Ⅰ. 软… Ⅱ. ①赵…②钟… Ⅲ. 软件设计—高等学校：技术学校—教材 Ⅳ. TP311.5

中国版本图书馆 CIP 数据核字(2008)第 146746 号

大连理工大学出版社出版
地址：大连市软件园路 80 号 邮政编码：116023
电话：0411-84708842 邮购：0411-84703636 传真：0411-84701466
E-mail：dutp@dutp.cn URL：http://www.dutp.cn
大连理工印刷有限公司印刷 大连理工大学出版社发行

幅面尺寸：185mm×260mm 印张：12 字数：270 千字
印数：1～3000
2008 年 10 月第 1 版 2008 年 10 月第 1 次印刷

责任编辑：马 双 责任校对：白 俊
封面设计：张 莹

ISBN 978-7-5611-4197-7 定 价：21.00 元

总　序

我们已经进入了一个新的充满机遇与挑战的时代，我们已经跨入了21世纪的门槛。

20世纪与21世纪之交的中国，高等教育体制正经历着一场缓慢而深刻的革命，我们正在对传统的普通高等教育的培养目标与社会发展的现实需要不相适应的现状作历史性的反思与变革的尝试。

20世纪最后的几年里，高等职业教育的迅速崛起，是影响高等教育体制变革的一件大事。在短短的几年时间里，普通中专教育、普通高专教育全面转轨，以高等职业教育为主导的各种形式的培养应用型人才的教育发展到与普通高等教育等量齐观的地步，其来势之迅猛，发人深思。

无论是正在缓慢变革着的普通高等教育，还是迅速推进着的培养应用型人才的高职教育，都向我们提出了一个同样的严肃问题：中国的高等教育为谁服务，是为教育发展自身，还是为包括教育在内的大千社会？答案肯定而且唯一，那就是教育也置身其中的现实社会。

由此又引发出高等教育的目的问题。既然教育必须服务于社会，它就必须按照不同领域的社会需要来完成自己的教育过程。换言之，教育资源必须按照社会划分的各个专业（行业）领域（岗位群）的需要实施配置，这就是我们长期以来明乎其理而疏于力行的学以致用问题，这就是我们长期以来未能给予足够关注的教育目的问题。

如所周知，整个社会由其发展所需要的不同部门构成，包括公共管理部门如国家机构、基础建设部门如教育研究机构和各种实业部门如工业部门、商业部门，等等。每一个部门又可作更为具体的划分，直至同它所需要的各种专门人才相对应。教育如果不能按照实际需要完成各种专门人才培养的目标，就不能很好地完成社会分工所赋予它的使命，而教育作为社会分工的一种独立存在就应受到质疑（在市场经济条件下尤其如此）。可以断言，按照社会的各种不同需要培养各种直接有用人才，是教育体制变革的终极目的。

随着教育体制变革的进一步深入，高等院校的设置是否会同社会对人才类型的不同需要一一对应，我们姑且不论。但高等教育走应用型人才培养的道路和走研究型（也是一种特殊应用）人才培养的道路，学生们根据自己的偏好各取所需，始终是一个理性运行的社会状态下高等教育正常发展的途径。

高等职业教育的崛起，既是高等教育体制变革的结果，也是高等教育体制变革的一个阶段性表征。它的进一步发展，必将极大地推进中国教育体制变革的进程。作为一种应用型人才培养的教育，它从专科层次起步，进而应用本科教育、应用硕士教育、应用博士教育……当应用型人才培养的渠道贯通之时，也许就是我们迎接中国教育体制变革的成功之日。从这一意义上说，高等职业教育的崛起，正是在为必然会取得最后成功的教育体制变革奠基。

高等职业教育还刚刚开始自己发展道路的探索过程，它要全面达到应用型人才培养的正常理性发展状态，直至可以和现存的（同时也正处在变革分化过程中的）研究型人才培养的教育并驾齐驱，还需假以时日；还需要政府教育主管部门的大力推进，需要人才需求市场的进一步完善发育，尤其需要高职高专教学单位及其直接相关部门肯于做长期的坚忍不拔的努力。新世纪高职高专教材编审委员会就是由全国 100 余所高职高专院校和出版单位组成的旨在以推动高职高专教材建设来推进高等职业教育这一变革过程的联盟共同体。

在宏观层面上，这个联盟始终会以推动高职高专教材的特色建设为己任，始终会从高职高专教学单位实际教学需要出发，以其对高职教育发展的前瞻性的总体把握，以其纵览全国高职高专教材市场需求的广阔视野，以其创新的理念与创新的运作模式，通过不断深化的教材建设过程，总结高职高专教学成果，探索高职高专教材建设规律。

在微观层面上，我们将充分依托众多高职高专院校联盟的互补优势和丰裕的人才资源优势，从每一个专业领域、每一种教材入手，突破传统的片面追求理论体系严整性的意识限制，努力凸现高职教育职业能力培养的本质特征，在不断构建特色教材建设体系的过程中，逐步形成自己的品牌优势。

新世纪高职高专教材编审委员会在推进高职高专教材建设事业的过程中，始终得到了各级教育主管部门以及各相关院校相关部门的热忱支持和积极参与，对此我们谨致深深谢意；也希望一切关注、参与高职教育发展的同道朋友，在共同推动高职教育发展、进而推动高等教育体制变革的进程中，和我们携手并肩，共同担负起这一具有开拓性挑战意义的历史重任。

新世纪高职高专教材编审委员会

2001 年 8 月 18 日

前言

在接触到实际的编程语言(C,C++,Java,C#)之前,即在没有任何编程方面的基础知识的情况下,进行编程逻辑思维训练是必要的,通过编程逻辑流程图和伪代码达到目标,可以为掌握编程方法和应用程序逻辑等打下坚实基础。面对实际应用的复杂性,现代软件设计既要能解决复杂的实际问题,要求软件功能强大;还要保证很高的质量水平和优越的性能,要求软件组织在进行软件开发时采用适合自己的软件过程,随着软件组织的积累而不断完善。在软件过程的生命周期中,软件设计是以模型为中心,即软件设计是由模型驱动的。模型采用 UML 建模语言构建,它以面向对象的编程方法为基础,对初学者进行面向对象的编程逻辑训练,提高其研究问题、理解问题的能力,并能够表述出解决这些问题所需要的逻辑。

本书拟从软件过程的角度介绍软件设计所涉及的基本概念和思想。首先引入软件过程中软件设计的基本要求和涉及的领域知识,然后对软件设计的实现中涉及的程序代码逻辑进行阐述,讨论了程序编码的语言、方法、算法描述以及数据类型和程序控制结构编程逻辑。针对软件编程设计方法,描述了模块化程序设计的面向过程编程逻辑与面向对象的概念和基本特性。讨论了软件设计将会从以程序语言代码为中心的开发移植为以模型为中心的设计开发,同时,对模型驱动的建模语言 UML 进行了详细描述,还介绍了建模工具 IBM Rational Rose 与集成设计开发环境 IBM Rational Software Architect 的使用方法,通过案例演示了需求建模的过程与规范要求。

本书作为软件设计的入门基础,是学好其它课程的知识预备,章节安排如下:

第 1 章为概述,介绍软件设计的基本要求和领域知识,初步了解软件过程和 RUP、XP、CIM、PIM、PSM、MDA、CMM 等概念。

第 2 章为程序设计,介绍程序设计语言的种类、程序设计方法和算法描述,使读者能够对程序设计达到初步的认识,了解程序设计的基本概念及特征。

第 3 章为数据类型及数据运算,介绍信息的表示方式、常量、变量和运算符,使读者能够掌握常量和变量数据

新世纪

的分类，了解数值和字符数据类型的特点，掌握算术、关系和逻辑运算符的操作。

第 4 章为选择结构，介绍程序控制结构与二分支和多分支选择结构，使读者了解程序控制结构的概念和分类，掌握二分支和多分支选择结构的编程逻辑思维。

第 5 章为循环结构，介绍 While 循环、Do…While 循环和嵌套循环，使读者掌握运用循环结构的编程逻辑来解决问题。

第 6 章为模块化程序设计，介绍模块化程序设计的概念、表示以及模块的调用关系，使读者能够熟悉模块化程序设计的概念和表示方式，并且掌握模块化程序设计的嵌套调用和递归调用的编程逻辑运用。

第 7 章为面向对象的分析与设计导论，介绍软件开发生命周期(SDLC)与面向对象的概念，使读者能够熟悉软件开发生命周期及其模型，深刻理解 SDLC 中设计软件的面向对象概念，并掌握其编程逻辑运用。

第 8 章为建模语言 UML，介绍 UML 的目标、UML 图素和 UML 视图，使读者能够掌握 UML 的理论原理与扩展机制，进一步理解 UML 的图素符号、模型图和不同类型的视图，在此基础上，通过实例分析，加强了理论与实践的结合，抽象面向对象建模的逻辑思维。

第 9 章为建模工具 Rational Rose，介绍 Rose 的环境和不同模型建模，使读者能够熟悉建模工具 Rational Rose 的分析设计集成环境，掌握运用 Rational Rose 建立系统的用例视图、逻辑视图、组件视图和部署视图等不同模型的能力。

第 10 章为集成设计与开发环境，介绍 RSA(Rational Software Architect)的工作台环境和相应的功能，使读者能够创建 UML 项目和构建 UML 模型，初步掌握 RSA 的使用。

第 11 章为 Rational 统一过程，介绍 RUP(Rational Unified Process)的概念和结构，使读者能够了解 RUP 生命周期的四个阶段的动态结构以及 RUP 的核心工作流程与核心支持工作流程的静态结构，认识软件开发过程对于软件质量控制与管理的重要作用。

第 12 章为需求建模，介绍业务建模和系统建模的问题分析、使用用例和角色建立需求模型、设置系统边界和项目范围、精化系统定义、实现系统用例以及由用例生成测试用例等。使读者能够领会需求建模的过程，掌握需求建模的要领。

本书的第 2 章到第 6 章由钟闰禄编写，第 1 章、第 7 章到第 12 章由赵从军编写，全书由赵从军统稿。

由于时间仓促，加之作者水平所限，错误和疏漏之处在所难免，请读者批评指正。

所有意见和建议请发往：gzjckfb@163.com
欢迎访问我们的网站：http://www.dutpgz.cn
联系电话：0411-84707492　84706104

编　者
2008 年 10 月

目　录

第1章 概 述

本章目标

- 了解软件设计的基本要求
- 熟悉软件设计涉及的领域知识

通过本章的学习，读者能够对软件设计有一个基本的了解，对软件设计需要储备的技术知识及领域范围有一个初步的认识，为进一步的学习做好准备。

1.1 软件设计的基本要求

软件产业已经从较低级的软件实现中摆脱出来，进入了设计和营销的阶段。软件是一种产品，软件生产需要遵循软件生命周期的开发过程，而现代的方法学理论以及相应的过程实践奠定了软件生产过程科学管理的基础，其中包括 RUP(Rational Unified Process)和 XP(Extreme Programming)。

RUP 是由 IBM Rational 开发的过程框架。它是一种迭代的开发方法，基于六个经过行业验证的最佳实践。一个基于 RUP 的软件开发项目将经历四个阶段：起始阶段(Inception)、细化阶段(Elaboration)、构造阶段(Construction)、交付阶段(Transition)。每个阶段都包括一次或者多次的迭代。在每次迭代中，根据不同的要求或工作流投入不同的工作量。RUP 可以根据软件项目的大小和复杂度进行裁剪，适应于小型项目到大型项目。

XP 是一个用于小型项目中的以代码为中心的轻量级过程，它来自 Kent Beck 的创意，如同 RUP 一样，XP 也基于迭代。RUP 和 XP 具有不同的基本原理。RUP 是过程组件、方法以及技术的框架，可以将其应用于任何特定的软件项目，限定 RUP 的使用范围。而 XP 是一个具有更多限制的过程，需要附加内容以使其适合完整的开发项目。

以上的不同点解释了软件开发的一种选择观点：开发大型系统的人员使用 RUP 解决问题，而开发小型系统的人员使用 XP 作为解决方案。经验表明，大部分的软件项目都处于两者之间，应尽力寻找适用于各自情况的过程以及恰当级别。单纯地使用两者之一是不充分的，可以从 RUP 出发，在必要时以一组更强大的技术来扩展 XP。

软件设计的最终目标是要解决实际工作中用户的问题，而用户的问题体现为对软件的需求，它是软件设计成功的标准。软件的需求往往需要用户、业务分析人员和软件设计人员的共同参与才能完成。软件的需求包括三个层次，它们是业务需求、用户需求和功能需求(也包括非功能需求)。业务需求反映了用户对系统、软件产品高层次的目标要求，它们在软件项目视图与范围文档中予以说明；用户需求描述了用户使用软件产品必须要完成的任务，使用用例(实例)文档或场景脚本予以说明，用例和实例的含义相同，有

关用例的概念将在后续章节中详细介绍；功能需求定义了开发人员必须实现的软件功能，使得用户能完成他们的任务，从而满足了业务需求。软件需求各组成部分之间的关系如图 1-1 所示。

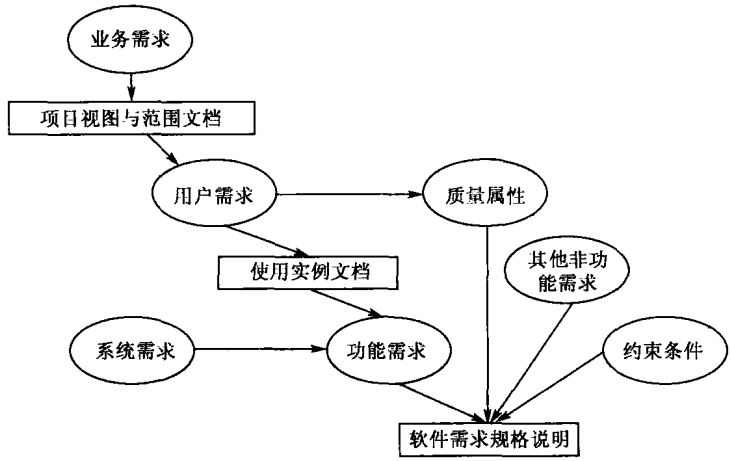


图 1-1 软件需求各组成部分

软件需求的实现，需要针对需求建立业务模型，确定角色和角色的用例，然后映射软件系统的设计模型，而模型的建立需要一种专用的建模语言。本书中将要介绍统一建模语言(Unified Modeling Language，简称 UML)的 UML2.0 规范。其中体现了模型驱动架构 MDA(Model Driven Architecture)的概念，它强调整个系统开发过程对软件系统的建模行为驱动，完成软件系统需求分析、架构设计、构建、测试、部署和运行维护工作。建模语言不仅仅是分析设计语言，更可作为一种编程驱动语言。MDA 通过抽象层次的不同，定义了计算独立模型(Computation-Independent Model，CIM)、平台独立模型(Platform-Independent Model，PIM)和平台相关模型(Platform-Specific Model，PSM)。

CIM 是一个抽象层次较高、独立于任何实现技术的系统模型，它着眼于操作环境中的系统以及系统需求的描述，而不关心系统本身的结构和功能实现细节，可作为业务模型和系统用例模型。

PIM 是一种系统分析模型，关注系统的整体架构实现，忽略与平台相关的内容，意味着不考虑与特定平台相关的细节，无需考虑特定的技术。

PSM 是一种系统设计模型，将 PIM 和与特定平台相关的细节结合起来，包含了具体平台的特定实现技术。PIM 可以转换为一个或多个 PSM，可以选择一种或几种平台技术实现系统。

通过支持 MDA 的工具，不同模型间可以通过模型转换技术(Model Transformation)实现相互转化。使用模型转换技术，可以将计算独立模型(CIM)转化为平台独立模型(PIM)，将平台独立模型(PIM)转化为平台相关模型(PSM)。如图 1-2 所示。

模型转换技术是实现 MDA 的关键。模型转换技术一般包括标记(Markings)和映射(Mapping)，映射包含了由一种模型向另一种模型转化的规约说明，而标记则用来在源模型中加入额外的信息，在转换时告诉映射如何将源模型中的特定模型元素映射到目标

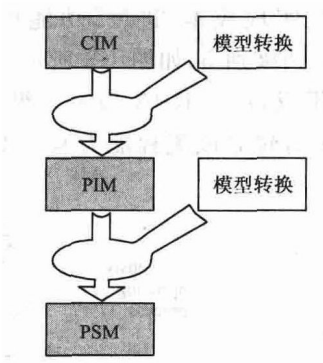


图 1-2　MDA 的模型及转换

模型。

本书将介绍一种设计与开发工具 IBM Rational Software Architect(RSA)，它是一个完整的设计和开发工具解决方案，实现了多项行业最新标准，提供了灵活的插件扩展机制。借助 UML2.0 技术，它实现了模型驱动的软件开发模式，可以帮助创建更加强壮的软件架构。同时，RSA 作为 IBM Rational 业务驱动软件开发平台的核心构件，提供了与需求管理工具、测试工具、配置和变更管理工具以及项目管理工具的完美集成，从而真正实现了企业内部的核心软件开发流程、开发平台和软件生产线。RSA 提供了模型之间的转换功能，可以从 PSM 模型转换为程序语言代码，其中包含了 UML 模型到 Java 程序代码的转换、UML 模型到C++程序代码的转换、UML 模型到 EJB 程序代码的转换以及 UML 模型到 C#程序代码的转换。

依据 MDA 的思想，它把以往以程序代码为中心的开发模式转变为以设计为中心的开发模式。MDA 为建立可重用的框架提供了一个很好的方式，可以作为设计重用的基本理念。利用 MDA 可进行通用软件(例如事务、目录服务)的模型设计，也可以用于特定业务领域的设计建模。可重用框架指的是将软件开发相关知识存储起来的框架，它不但包括了设计模型，还包括了软件开发过程和方法。软件开发是在这个框架之内进行的，而且不断地完善和改进。

开发出成功的软件，是软件设计的最终目标，而软件是通过代码体现的，如果没有最后的用于交付的代码，软件就无法成为软件。因此，必须保证软件过程能够产出代码，而且是优秀的代码。以 RUP 为基础，来定制自己的软件过程，删除不需要的活动。RUP 软件过程是经过精心的设计和实践的验证的，每个软件组织都结合自身的特点和具体软件项目的要求对其进行剪裁，形成自有特色的软件过程，而且随着组织的成长而成长，随着环境的变化而变化。对软件过程进行的改进过程称为软件度量，它需要一个长期的数据收集过程，可作为软件开发质量控制与进度控制的基础。衡量软件组织的软件过程成熟度的优劣，是通过软件工程协会 (SEI) 的能力成熟度模型 CMM(Capability Maturity Model)进行的，最高级为 CMM5。

CMM 是一个描述有效软件流程元素的框架，描述了一条从临时的、未成熟的流程向成熟的、规范化的流程演进的过程。CMM 覆盖软件开发和维护的规划、工程以及管理经

验。这些关键的经验提高了组织实现成本、进度、功能性和产品质量等目标的能力。CMM 有五个成熟级别：从级别 1 到级别 5，如图 1-3 所示。每个成熟度级别由关键流程领域(Key Process Areas，KPA)组成，每个 KPA 确定一组相关活动，当这些相关活动一起开展时，它们完成一系列被认为对建立该流程能力有重要影响的目标。表 1-1 说明了五个成熟等级的特点。

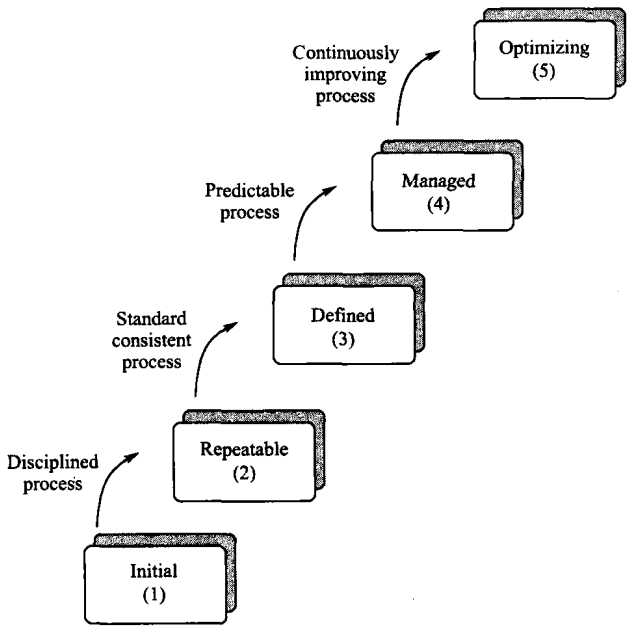


图 1-3 CMM 的演进

表 1-1　　成熟等级的特点

1 级：初始	软件过程被认为是特别的，有时甚至是混乱的。很少有详细定义的过程，而且成功取决于个人努力
2 级：可重复的	基本项目管理过程被建立起来，能够跟踪成本、时间进度和功能。必要的过程原则能够在有着相似应用程序的项目上重复早期的成功
3 级：清楚定义的	管理和工程活动的软件过程都被文档所记录和标准化，并被集成到一个标准的软件过程以备组织使用。所有项目都使用一个被认可的、专用版本的开发和维护软件的组织标准软件过程
4 级：良好管理的	对软件过程的度量和产品质量的详细信息被收集起来。软件过程和产品被量化地理解和控制
5 级：最优化	从过程和领先的创新思想及技术中获得的量化反馈激活了不断的过程改进

1.2 软件设计的领域知识

在一个软件的设计过程中，经过生命周期的不同阶段，涉及的领域包括业务建模、需求、分析与设计、实现、测试、部署、配置与变更管理、项目管理和环境。

业务建模包括业务架构模型、业务过程模型和域模型。业务架构模型以功能区域的形式描述，也称为业务组件模型，是企业的战略业务层部件，着重企业的核心能力，体现区别于竞争者的企业战略；业务过程模型体现活动及业务项的流程，表示企业的业务过程；域模型也称为概念模型，表示定义的业务术语和信息结构，提供一个可视化的业务因素之间的结构关系。

需求在上节中提到，它有三个层次，从业务需求到功能需求，意味着从业务建模到系统建模的转变。先建立业务用例模型，业务用例模型可以从业务建模的模型中导入，然后将其转化为系统用例模型。业务用例到系统用例的转化如图 1-4 所示。

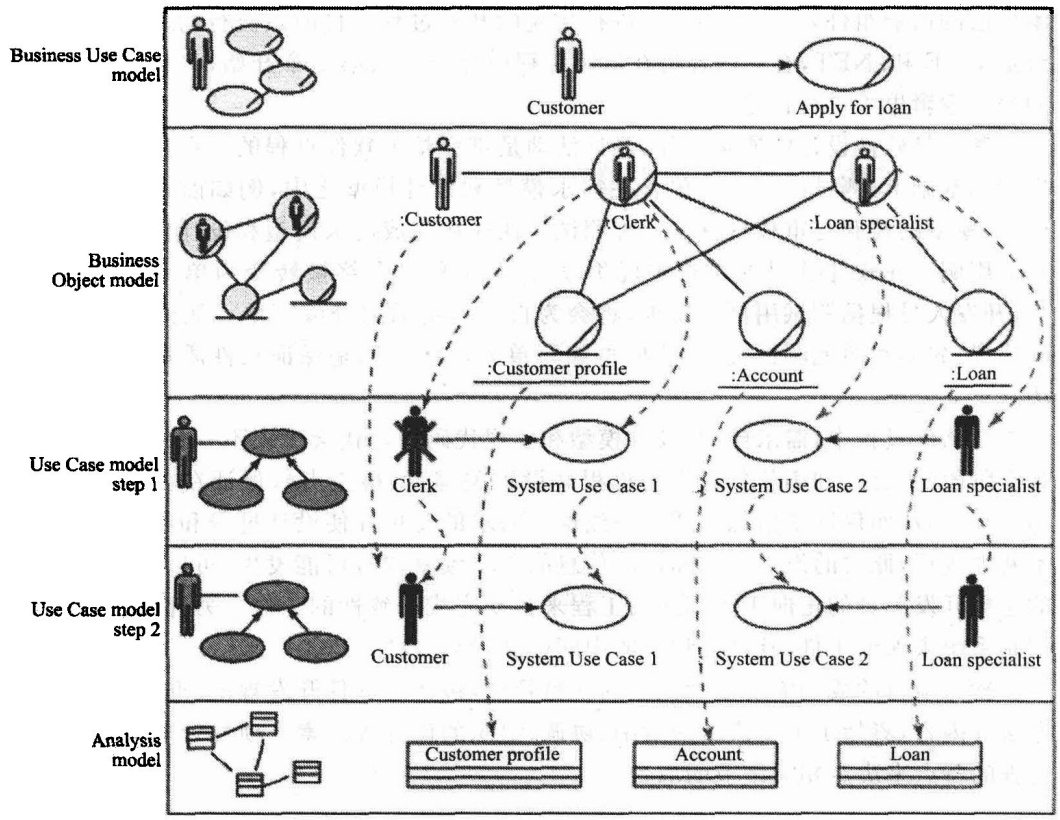


图 1-4　业务用例到系统用例的转化

图 1-4 中显示了业务模型中所找到的东西和系统用例模型中的东西之间的清晰映射，业务模型包含业务用例模型和业务分析模型。业务分析模型是业务用例模型的实现，并且拥有紧密的集成化和可追溯性。系统用例模型可以追溯到业务分析模型。业务分析模型可以追溯到业务用例模型。这时的模型体现为 CIM。

分析与设计通常指的是面向对象的分析与设计（Object-Oriented Analysis and Design，OOAD），有关面向对象的概念将在后续章节中介绍。在 OOAD 的基础上进行面向服务的分析与设计（Service-Oriented Analysis and Design，SOAD），这里涉及面向服务的体系结构（Service-Oriented Architecture，SOA）的概念，有关 SOA 的内容将在后续课程中介绍。分析与设计通过 UML 模型表示，UML 的分析模型通常体现为 PIM，而

UML 的设计模型通常体现为 PSM。

实现(Implementation)是通过编写程序代码完成 UML 模型的功能。代码是软件设计的最终目的。软件过程的每一个阶段所进行的模型变换都是知识转换的过程,知识转换的终点就是软件,软件的运行维护是通过代码体现的。在上节的内容中提到可重用框架,表明现代软件设计的发展趋势是重用,开发一个软件已经很少从最底层开始设计并实现代码编写了。使用各种各样的技术和平台,包括数据库、分布式体系、UI 机制、业务元素等等,PSM 模型涉及具体平台的特定实现技术,它可以转换为可执行的独立于平台的代码实现。因此,软件组织在实施可重用框架时,总是以具体技术开发平台为基础,积累自己的商业组件,并以此为中心进行相应的开发过程。目前最流行的两种软件开发平台是 J2EE 和. NET,有关内容将在后续课程中介绍。从第 2 章开始,将开始对代码实现的程序逻辑思维进行讨论。

测试是软件设计的质量保障,测试活动是贯穿整个软件过程的。在需求模型和设计模型的基础上,测试的信息都包含在需求模型和设计模型之中,例如前置条件和后置条件等,模型的校验与审核也算是一种测试。往往在完成需求模型和设计模型时同步完成测试用例。在软件过程进入到实现阶段时,测试信息最终被转变为单元测试用例的形式,开发人员根据测试用例的要求,将会为自己负责的部分编写单元测试代码,根据 XP 的建议,先得写单元测试代码,然后再编写单元功能代码是保证软件质量的一种有效方法。

在软件过程中,需求模型、设计模型和实现代码的迭代会出现不一致的情况,这涉及每个阶段的工件,利用软件过程的里程碑设置关键点(检查点),保证在关键点上的工件的一致性,从而保证了信息传递的一致性。需求的变更促使设计过程和编码过程的迭代不断地进行,此时的设计模型和代码信息的不一致就很有可能发生,可以利用 RSA 这样的建模开发工具的正向工程和逆向工程来自动完成一致性的工作。另外,可以使用版本控制系统来保证工件(软件过程中的中间产品)的一致性。

软件项目的成功需要科学艺术的项目管理,包括了软件开发规范、准则、实践、过程、方法等内容,囊括了项目成员的激励、协调、组织的设计等因素。项目管理需要根据软件过程的特点来决定相关细节的取舍。

习 题

1-1 简述 RUP 软件过程。

1-2 简述 XP 软件过程。

1-3 解释 CIM、PIM 和 PSM 的含义。

1-4 解释 MDA 的思想。

1-5 解释 CMM 每个级别的含义。

1-6 详述软件设计的领域知识。

第2章 程序设计

本章目标

- 了解程序设计语言的种类
- 了解面向过程与面向对象的程序设计方法
- 掌握算法的概念及特征

通过本章的学习，读者能够对程序设计达到初步的认识，了解程序设计的基本概念及特征。

程序设计(Programming)是指设计、编制、调试程序的方法和过程。它是目标明确的智力活动。由于程序是软件的本体，软件的质量主要通过程序的质量来体现，在软件研究中，程序设计的工作非常重要，内容涉及有关的基本概念、工具、方法以及方法学等。

当人们要利用计算机解决实际问题时，比如：数值计算问题，必须要让计算机按照人的规定完成一系列的工作，这就需要人与计算机之间的交流，这种交流能使计算机按人的意愿完成一系列的操作。要想完成这种交流必须在人与计算机之间有一种桥梁，这种桥梁就是计算机语言，又称为程序设计语言。使用程序设计语言编写的用来使计算机完成一定任务的一段“文章”称为程序，编写程序的工作称为程序设计。

2.1 程序设计语言

程序设计语言，是一组用来定义计算机程序的语法规则。它是一种标准化的交流技巧，用来向计算机发出指令。一种计算机语言让程序员能够准确地定义计算机所需要使用的数据，并精确地定义在不同情况下应当采取的行动。程序设计语言原本是被设计成专门使用在计算机上的，但它也可以用来定义算法或者数据结构。正因为如此，程序员才会试图使程序代码更容易阅读。

程序设计语言是人们根据计算机的特点以及描述问题的需要设计出来的。随着计算机技术的发展，不同风格的语言不断出现，逐步形成了计算机语言体系。人们总是希望设计出好用的语言，因此，计算机语言也经历了由低级向高级发展的历程。随着计算机技术的迅速发展，程序设计语言也经历了由低级到高级发展的多个阶段，程序设计方法也得到了不断地发展和提高。

计算机语言按其发展过程一般分为：机器语言、汇编语言和高级语言。

2.1.1 机器语言与汇编语言

机器语言是最底层的计算机语言。用机器语言编写的程序，计算机硬件可以直接识别。在用机器语言编写的程序中，每一条机器指令都是二进制形式的指令代码，即由一连

串的二进制 0 和 1 组合起来的编码，每一条指令规定了计算机要完成的某个操作。在指令代码中，一般包括操作码和地址码，其中操作码告诉计算机做何种操作，即“干什么”，地址码则指出被操作的对象存放在哪儿。

机器语言程序都是由二进制 0 和 1 组成的指令，程序编写起来非常繁琐，可以用“难学、难记、难写、难检查、难调试”来概括，尤其是用机器语言编写的程序完全依赖于机器，所以程序的可移植性差。由于用机器语言编写的程序直接针对计算机硬件，因此它的执行效率比较高，能充分发挥出计算机的速度性能，这也是机器语言的优点。

为了克服机器语言的缺点，人们对机器语言进行了改进，用一些容易记忆和辨别的有意义的符号代替机器指令，如：用指令助记符来代替机器语言指令代码中的操作码，用地址符号来代替地址码。用这样一些符号代替机器指令所产生的语言就称为汇编语言，也称为符号语言。

例如：为了计算表达式“9+8”的值，用汇编语言编写的程序与用机器语言（8086CPU 的指令系统）编写的程序如表 2-1 所示。

表 2-1　　汇编语言程序与机器语言程序举例

语句序号	汇编语言指令	机器指令	指令功能
1	MOV AL,9	10110000　00001001	把加数 9 送到累加器 AL 中
2	ADD AL,8	00000100　00001000	把累加器 AL 中的内容与另一数相加，结果存在累加器 AL 中（即完成 9+8 运算）
3	HLT	11110100	停止操作

从上表中可以看出，在该汇编语言程序中，以 MOV（MOVE 的缩写）代表“数据传送”，ADD 代表“加”，HLT（HALT 的缩写）代表“停止”等。这些符号含义明确，容易记忆，所以又称为助记符。用这些助记符编写的程序，可读性好，容易查错，修改方便；但计算机硬件不能直接识别，必须由一种专门的翻译程序将汇编语言程序翻译成机器语言程序后，计算机才能识别并执行。这种翻译的过程称为“汇编”，负责翻译的程序称为汇编程序，翻译出的程序称为目标程序，而翻译前的程序称为源程序，如图 2-1 所示。

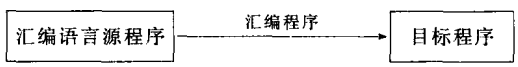


图 2-1　汇编程序的作用

汇编语言也是一种面向机器的语言，但比机器语言易读、易改，执行速度与机器语言相当，比高级语言快很多，所以直到现在仍在实时控制、实时处理领域中广泛应用。

2.1.2　高级语言

机器语言和汇编语言都是面向机器的语言，一般称为低级语言。低级语言对机器的依赖性太大，而且与自然语言相距甚远，不符合人们的表达习惯。

为了从根本上改变语言体系，必须从两方面下工夫：一是力求接近于自然语言；二是力求脱离具体机器，使语言与指令系统无关，达到程序通用的目的，这就要求程序的可移植性好。一个可移植性好的程序，应用在各种计算机和操作环境中都能运行，并得到相同的结果。因此，从 20 世纪 50 年代中期开始逐步发展出面向问题的程序设计语言，称