项目 1

初识软件工程和项目

所谓项目是指在一定的资源约束下，为创造独特的产品或服务而进行的一次努力。项目有大小之分，一个大的项目是由若干小的项目构成的。通常一个大型项目的开发具有特定的目标和整体性，并且具有独特性，能够一次性完成而不可逆转。本书从项目一开始就引入了“学生信息管理系统”的分析和调研，其开发过程贯穿全书，使读者在学习软件工程基础知识的同时，接触到实际的软件工程项目开发案例。

概括地说，软件工程课程就是分析、设计、实现与维护软件系统的一组规范，它指导着软件开发人员以工程化、项目化的手段规范地开发高质量的软件。

你想知道软件开发中的项目是什么样的吗？软件开发及其发展经历了哪几个阶段？软件危机主要有哪些表现以及产生的原因何在？软件工程又是什么样的课程？软件工程的原则是什么？软件工程的目标及面临的主要问题是什么？软件的生存周期包括哪几个阶段？常见的软件开发模型和建模工具有哪些？这些问题将在下面逐一予以介绍。

学习目标

- **理解并掌握**：项目、程序、软件的概念，软件危机产生的原因，软件工程的定义、目标和原则，软件生存周期各阶段的特点和内容。
- **熟悉**：软件生存周期模型，包括瀑布模型、快速原型模型、螺旋模型、喷泉模型等；学会 Visio 2007 的安装和初步使用。
- **了解**：软件发展的 4 个阶段及其特点、软件危机的产生及其表现形式；软件工程的开发方法和常用建模工具的使用方法。

1.1 软件工程入门

在读者学习完计算机的高级语言等前导课程之后，往往习惯性地认为软件工程课程是教给大家如何开发软件的，而开发软件就是编写程序，但这样的认识是片面的。因此，要学好软件工程课程首先要对软件及其发展过程有一个正确的认识；其次要明白软件危机的概念及表现，熟知产生软件危机的原因及其解决途径；最后还要了解软件工程定义的产生，理解并掌握软件工程的 7 条基本原理。

1.1.1 软件及其发展

1. 人们对软件的认识

在学习软件工程课程之前，只有对软件工程课程有一个正确的认识才能学好这门课程。首先要理解以下两个正确观点。

1)开发软件不等于编写程序

事实上，开发软件应该完成的工作远远多于编写程序应该完成的工作，编程仅仅是开发软件所应完成工作的一部分，具体的软件开发工作包括如下几个方面。

(1)为了开发出一个符合用户需要、质量合格的软件，软件工程师必须首先弄清楚用户面临的问题是什么，也就是要明确软件的“主攻”方向。

(2)进行可行性研究，分析用户面临的问题是否有行得通的解决方案。为避免浪费资源，仅在该软件的开发是可行的前提下，才能进行实质性的开发工作。

(3)进行需求分析，通过与用户的反复交流，弄清楚用户对该软件的具体需求，这些需求是进行软件设计的依据。

(4)一般地，在编写程序之前需要先进行软件设计，大型软件的设计工作分成两个阶段进行，即先进行总体设计(又称为概要设计)，再进行详细设计。

(5)编写程序实质上是把设计结果翻译成用某种程序设计语言编写的程序，首先是算法设计(算法是指完成指定功能的解题步骤)，然后再用程序设计语言(例如，C++、VB、Java、VC++、C#等)表达该算法。

(6)程序编写出来之后，还需要经过严格的测试过程(需要的工作量通常占软件开发全部工作量的 40%～50%)，软件确实符合用户需求而且质量合格，才能交付给用户使用。

2)错误做法会导致软件危机

开发软件不等于编写程序，但是，迄今为止，仍然有不少人错误地认为开发软件就是编写程序，或者主要就是编写程序。人们之所以有错误的认识并在开发软件时采用了错误的做法，主要归因于在计算机系统发展的早期阶段“开发软件”的个体化特点。所谓软件危机是指在计算机软件的开发和维护过程中所遇到的一系列严重问题，这些问题绝不仅仅是不能正常运行的软件才具有的，实际上，几乎所有软件都不同程度地存在这些问题(关于软件危机的具体表现将在后面介绍)。

2. 软件的发展

1946 年，世界上第一台电子计算机 ENIAC 诞生，之后伴随着硬件的更新换代，计算机

程序从硬件中分离出来，从而逐渐形成了软件技术的概念。经过 60 多年的发展演变，人们对软件已经有了更为深刻的认识，在此过程中，计算机软件的发展经历了如表 1-1 所示的 4 个阶段。

表 1-1　软件发展的 4 个阶段

年　代	阶　段	软件的范畴	主要程序设计语言	软件工作范围	需 求 者
20 世纪 60 年代之前	程序设计	程序	机器及汇编语言	程序编写	程序设计者本人
20 世纪 60～70 年代	程序系统	程序及说明书	高级语言	包括设计和测试	少数用户
20 世纪 70～90 年代	软件工程	产品软件(项目软件)	高级语言系统、数据库	软件生存期	市场用户
20 世纪 90 年代之后	现代软件工程	项目工程	面向对象可视化设计语言	整个软件生存期	面向所有用户

(1)第 1 代(20 世纪 60 年代之前)——程序设计阶段。

这个阶段的软件生产方式是个体手工方式，使用机器语言和汇编语言编程，难读、难写、编程效率低，但追求编程技术和程序运行的效率。本阶段硬件的特点是价格高、存储量小、可靠性差；软件的特点是受制于硬件的发展，程序规模小，不重视程序设计方法，编程者和使用者一般为同一人(或同组人)。

(2)第 2 代(20 世纪 60～70 年代)——程序系统阶段。

这个阶段的软件生产方式是作坊式集团合作生产，但仍然是“个体化”开发方法，没有形成规模化生产；软件由开发小组开发并进行系统升级和维护；使用高级语言，开始提出结构化方法。软件内容由程序及其说明书构成。本阶段硬件的特点是价格降低，速度、容量、可靠性明显提高；软件的特点是多用户、实时系统出现，程序员数量猛增，但由于软件本身的特点以及软件开发工具和方法不适应规模大、结构复杂的软件产品化开发，导致形成尖锐的矛盾，开始出现软件危机。

(3)第 3 代(20 世纪 70～90 年代)——软件工程阶段。

这个阶段的软件生产方式是工程化生产，使用高级语言系统、数据库、开发工具、开发环境、网络及分布式开发等，软件开发成为一门新兴的工程学科，即软件工程。本阶段硬件的特点是向超高速、大容量、微型化发展；软件的特点是以软件的产品化、系列化、工程化、标准化为特征的软件产业发展起来，产生了软件工程化可遵循的设计原则、方法和标准，有专职的维护人员，开发技术有进步，但未获得突破性进展，未完全摆脱软件危机。

(4)第 4 代(20 世纪 90 年代之后)——现代软件工程阶段。

这个阶段的软件生产方式走向了项目工程生产，一些新技术开始出现，如面向对象技术、强大的桌面系统、专家系统、嵌入式系统、并行计算、人工神经网络、网络计算机等，本阶

段的特点是面向对象可视化技术在许多领域中迅速取代了传统的软件开发方法，软件开发技术逐渐摆脱了软件危机。

3. 软件的分类及特点

1)软件的定义

软件是计算机系统中与硬件相互依存的部分，它是包括程序、数据及其相关文档的完整集合。程序是按事先设计的功能和性能要求执行的指令序列。计算机程序包括源程序和目标程序。数据是使程序能正常操纵信息的数据结构(即数据的组织形式)。文档是与程序开发、维护和使用有关的图文材料。从应用的角度，可以通俗地按如下所列理解。

(1)软件＝程序＋数据＋文档。

(2)面向过程的程序＝算法＋数据结构。

(3)面向对象的程序＝对象＋类＋继承＋消息。

(4)面向构件的程序＝构件＋构架。

2)软件的分类

软件的分类反映了软件开发所需面对的不同背景。具体有以下几种划分方法。

(1)按软件的规模划分。

软件的规模是指软件项目可量化的结果，通常采用代码行(LOC)的数量或耗用的时间、人力等来度量。一般划分为微型(500 行以下，1～4 周，1 人)、小型(2 000 行以下，半年，1 人)、中型(5 000～50 000 行，1～2 年，2～5 人)、大型(5～10 万行，2～3 年，5～20 人)、超大型(100 万行，4 年以上，100 人以上)软件。

(2)按软件工作方式划分。

按软件工作方式，软件可分为实时、分时、交互式和批处理软件等几种。

- 实时处理软件必须满足严格时间约束条件，是对当前时间、当前任务所做的实时处理、反馈和控制过程的软件。例如，工业过程控制软件、卫星实时监控软件等。
- 分时软件是多个用户在各自的终端上按时间片轮换的方式同时使用一台计算机，而每个用户都感到好像自己独享一台计算机一样。
- 交互式软件是能实现人机通信的软件。用户通过交互方式工作能进行人机对话，联机控制程序可以处理执行任务，也可以产生一个任务让其他设备或软件完成。
- 批处理软件是一次可以执行多条指令的软件。

(3)按软件应用功能划分。

- 系统软件。系统软件是与计算机系统硬件紧密交互，协调计算机系统各部分工作必不可少的软件。例如，操作系统、设备驱动程序及数据库管理系统、通信处理程序等。
- 支持软件。支持软件是协助使用者开发软件的工具性软件。例如，程序编译器、自动化测试软件、系统分析辅助工具及软件开发管理工具等。
- 应用软件。应用软件是为特定目的的服务和用途而专门开发的软件。例如，商业信息处理软件、工程和科学计算软件、智能产品嵌入软件等。

(4)按软件服务对象的范围划分。

- 项目软件。项目软件是软件开发机构受特定用户委托而开发的软件。例如，商品管

理系统、空中交通管制系统、军用防空指挥系统、生产过程控制系统等。一般情况下，项目软件是在合同的约束下开发的。为了争取软件开发合同，软件开发机构必须重视质量管理，而软件开发机构的技术实力、开发经验以及社会信誉等也相当重要。

- 产品软件。产品软件是软件开发机构直接为市场开发的软件。例如，字处理软件、多媒体播放软件、游戏软件、教育软件等。产品软件的功能、性能、价格和售后服务对开发机构参与市场竞争有重要影响。

3)软件的特点

软件具有如下一些特点。

(1)软件是一种逻辑实体，而不是具体的物理实体，因此它具有抽象性。

(2)软件的生产没有明显的制造过程。对软件的质量控制，必须立足于软件开发方面。

(3)在软件的运行和使用期间，没有像硬件那样的磨损、老化问题。

(4)软件的开发和运行往往受到计算机系统的限制，对计算机系统有不同程度的依赖性。

(5)软件本身是复杂的。迄今为止，对软件的开发尚未完全摆脱手工方式。

(6)软件的成本相当昂贵，其随时间的推移而逐年上升，如图 1-1 所示。

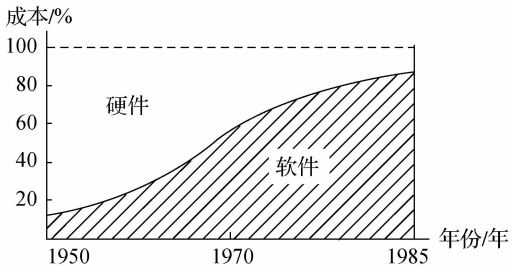


图 1-1　软件的成本随时间变化图

(7)相当多的软件工作涉及社会因素，从软件文档的书写规范，如 GB/T 8567—2006 计算机软件文档编制规范可以看到这一点。

1.1.2　软件危机

1. 软件危机的概念及表现

软件危机(software crisis)是在开发和维护计算机软件的过程中所遇到的一系列严重问题。概括地讲，主要包含两方面的问题，即如何开发软件以满足用户对软件日益增长的需求和如何维护数量不断膨胀的已有软件。

软件危机主要表现在如下几个方面。

(1)对软件开发成本和进度的估计常常很不准确。

一般地，实际成本比估计成本有可能高出几倍甚至十几倍，实际进度比预期进度拖延几个月甚至几年的现象并不罕见。这种现象降低了开发组织的信誉。开发组织为赶进度和节约成本所采取的权宜之计往往又损害了软件产品的质量，从而不可避免地引起用户的不满。

(2)用户对“已完成的”软件系统不满意的现象经常发生。

软件开发人员常常在对用户需求只有模糊的了解,甚至对所要解决的问题还没有确切认识的情况下,就仓促上阵匆忙着手编写程序。软件开发人员和用户之间的交流往往很不充分,闭门造车必然导致最终产品不符合用户的实际需要。

(3)软件产品的质量常常靠不住。

软件可靠性和质量保证的确切定量概念刚刚出现,软件质量保证技术(审查、复审和测试)还没有坚持不懈地应用到软件开发的全过程中,这些都会导致软件产品的质量问题。

(4)软件常常是不可维护的。

很多程序中的错误很难改正,实际上不可能使这些程序适应新的硬件环境,也不可能根据用户的需求在原有程序中增加新的功能。

(5)软件通常没有适当的文档资料。

软件不仅是程序,还应该有一整套文档资料,这些文档资料是在软件开发过程中产生出来的,而且应该是“最新的”(与代码完全一致)。缺乏文档或者文档不合格必然给软件的开发和维护带来许多严重的困难和问题。

(6)软件成本在计算机系统总成本中所占的比例逐年上升。

随着微电子技术的进步和生产自动化程度的提高,硬件成本逐年下降,然而软件开发需要大量的人力,软件成本则随着通货膨胀以及软件规模和数量的不断扩大而逐年上升。

(7)软件开发速度跟不上计算机发展速度。

软件开发速度跟不上硬件的发展速度,也远远跟不上计算机应用迅速普及的趋势。软件产品“供不应求”的现象使得人类无法充分利用现代计算机硬件提供的巨大潜力。

以上列举的仅仅是软件危机的一些明显的表现,但与软件开发和维护有关的问题远远不止这些。

2. 产生软件危机的原因

1)与软件本身的特点有关

软件不同于硬件,它是计算机系统的逻辑部件而不是物理部件。在编写完程序代码并在计算机上运行之前,软件开发过程的进展情况较难衡量,软件的质量也较难评价。因此,管理和控制软件开发过程相当困难。

2)软件不易于维护

(1)软件维护通常意味着改正或修改原来的设计,客观上使软件较难维护。

(2)软件不同于一般程序,它的规模较大,不易于维护。软件维护费用急剧上升,直接威胁到计算机应用的扩大。

3)缺少规范的理论指导

在软件开发过程中,加剧软件危机的重要原因包括没有统一规范的方法论的指导,文档资料不齐全,忽视了人与人之间的交流;或多或少地采用了错误的生产方法和技术或进步缓慢等。

4)没有软件开发前期的需求分析

忽视了软件开发前期的需求分析,没有完整、准确地认识用户需求,就匆忙着手编写程序,形成了“软件就是程序”的错误观念。

5)忽视测试

忽视测试阶段的工作,提交给用户的软件质量差。

总之,通过以上分析,从技术和管理两个方面入手,引入“软件工程”的概念是必然的。

3. 解决软件危机的途径

解决软件危机的途径主要有如下几个。

(1)技术措施。使用更好的软件开发方法和开发工具。

(2)组织管理措施。软件开发不是某种个体劳动的神秘技巧,而应该是一种组织良好,管理严密,各类人员协同配合、共同完成的工程项目。

现代软件工程技术正是为克服软件危机而提出的一种概念,并在实践中不断探索它的原理、技术和方法。在此过程中,人们研究和借鉴了工程学的某些原理和方法,并形成了一门新的学科——软件工程学,但时至今日人们还没有完全摆脱软件危机。

1.1.3　软件工程

1. 软件工程的产生和发展

软件工程(software engineering)这一概念,主要是针对 20 世纪 60 年代出现的“软件危机”而提出的。它首次出现在 1968 年在前联邦德国召开的北大西洋公约组织(NATO)会议上。自这一概念提出以来,围绕着软件项目开展了有关开发模型、方法及支持工具的研究,其主要思路是把人类长期以来从事各种工程项目所积累的行之有效的原理、概念、技术和方法,特别是人类从事计算机硬件研究和开发的经验教训应用到软件的开发和维护中。

2. 软件工程的定义

软件工程是一门将理论知识应用于实践的工程科学,它借鉴了传统工程的原则和设计方法,以求高效地开发高质量软件。关于软件工程,目前尚无统一的定义,下面给出几个有代表性的定义。

(1)软件工程是指导计算机软件开发和维护的工程学科。它采用工程的概念、原理、技术和方法来开发与维护软件,把经过时间考验并证明正确的管理技术和当前能够得到的最好的技术方法结合起来。

(2)软件工程是一门涉及软件计划,需求分析,设计,编码,测试和维护的原理、方法及工具的研究和应用的学科。

(3)软件工程是应用计算机科学、数学、工程科学及管理科学等原理开发软件的工程。软件工程借鉴传统工程的原则和方法,以提高软件质量、降低成本为目的。其中,计算机科学、数学用于构造模型与算法,工程科学用于制定规范、设计范型(paradigm)、评估成本及确定权衡,管理科学用于管理计划、资源、质量、成本等。

3. 软件工程的基本原理

1968 年,NATO 在前联邦德国加尔密斯召开的有关计算机软件的会议上正式提出了“软件工程”这一术语。目前有 100 多条关于软件工程的准则,其中最出名的是著名软件工程专家 B. W. Boehm 于 1983 年提出的 7 条基本原理。

1)用分阶段的生命周期计划严格管理

相关统计数据表明,不成功的软件项目中有一半左右是由于计划不周造成的。

Boehm 认为,在软件的整个生命周期中应制定并严格执行 6 类计划,即项目概要计划、里程碑计划、项目控制计划、产品控制计划、验证计划和运行维护计划。

2)坚持进行阶段评审

软件的质量保证工作不能等到编码阶段结束之后再进行。这是因为:第一,大部分错误是在编码之前造成的,例如,根据 Boehm 等人的统计,设计错误占软件错误的 63%,而编码错误仅占 37%;第二,错误发现与改正得越晚,改正错误所需付出的代价越高。因此,在软件开发的每个阶段都进行严格的评审,以便尽早发现在软件开发过程中所犯的错误,并及时加以改正,这是一条必须遵循的重要原则。

3)实行严格的产品控制

在软件开发过程中不要随意改变需求,因为改变某项需求往往需要付出较高的代价,但在实践中用户往往会提出需求变更,这时就需要采取科学的产品控制技术。

当改变需求时,为了保持软件各配置成分的一致性,必须实行严格的产品控制,其中主要是实行基准配置管理。所谓基准配置又称为基线配置,它们是经过阶段评审后的软件配置成分(各个阶段产生的文档或程序代码)。基准配置管理也称为变动控制,一切有关修改软件的建议,特别是涉及对基准配置的修改建议,都必须按照严格的规程进行评审,获得批准以后才能实施修改。

4)采用现代程序设计技术

实践表明,采用先进的技术既可以提高软件开发的质量和效率,又可以提高软件维护的效率。20 世纪 60 年代末提出的结构程序设计技术,已经成为公认的先进的程序设计技术;80 年代之前又进一步提出了各种结构化分析和设计技术;90 年代,面向对象分析和设计技术已在许多领域中逐渐取代传统的结构化技术。

5)应该能清楚地审查结果

软件是看不见、摸不着的逻辑产品,其开发过程难以评价和管理。为了提高软件开发过程的可见性,更好地进行管理,应该依据软件开发项目的总目标和完成期限,规定开发小组的责任、产品标准及完成日期,从而使得所得到的结果能够被清楚地审查。

6)开发小组的人员应该少而精

开发小组人员的素质和数量是影响软件产品质量和开发效率的重要因素。此外开发小组人员数目的增加会使相互交流复杂、费用增加。因此,组成开发小组的人员要少而精。

7)承认不断改进软件工程实践的必要性

遵循前 6 条基本原理,就能够按照当代软件工程基本原理实现软件的工程化生产,但不能保证跟上时代前进的步伐。因此,Boehm 指出要积极主动地采纳新的软件技术,且不断总结经验,承认不断改进软件工程实践的必要性。

4. 软件工程的目标、活动及原则

软件工程框架如图 1-2 所示。根据这一框架,可以看出软件工程涉及的要素包括软件工程的目标、活动及原则 3 个方面,具体如下所示。

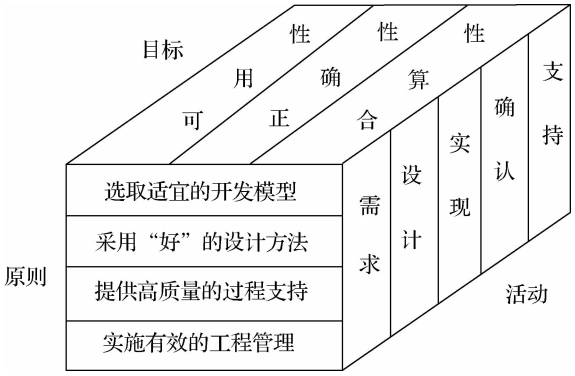


图 1-2　软件工程的要素

(1)软件工程的目标可概括为生产具有正确性、可用性以及开销合算的软件产品。

(2)软件工程的活动是为生产一个最终满足用户需求,围绕软件的需求、设计、实现、确认和支持活动,所进行的达到工程目标的一系列步骤。

(3)软件工程实施过程中的 4 条基本原则如下。

- 选取适宜的开发模型。
- 采用“好”的设计方法。
- 提供高质量的过程支持。
- 实施有效的工程管理。

1.2 软件开发方法

软件工程的最终目标是以较经济的投资生产具有正确性、可用性以及开销合算的高质量的软件产品。软件的开发方法有很多,包括面向过程的开发方法、面向数据结构的开发方法、面向对象的开发方法以及可视化开发方法等。其中,面向数据结构的开发方法可归结为面向过程的开发方法,可视化开发方法也可归结为面向对象的开发方法。这里重点讨论的是面向过程的开发方法和面向对象的开发方法。

1.2.1 面向过程的开发方法

面向过程的开发方法包括面向过程的需求分析、设计、编程、测试、维护和面向过程的管理。面向过程的开发方法又称为结构化方法,具体应用时,包括结构化分析、结构化设计、结构化编程、结构化测试和结构化维护方法。它是 1978 年由 E. Yourdon 和 L. L. Constantine 提出的,也可称为面向功能的软件开发方法或面向数据流的软件开发方法。

Yourdon 方法是 20 世纪 80 年代使用最广泛的软件开发方法,它首先采用结构化分析(SA)方法对软件进行需求分析,然后用结构化设计(SD)方法进行系统总体设计,最后再实现结构化编程(SP)。结构化方法开发步骤明确,SA、SD、SP 相辅相成,而且给出了两类典型的软件结构(变换型和事务型)便于参照,使软件开发的成功率大大提高,从而深受软件开发

人员的青睐。

1.2.2 面向对象的开发方法

面向对象的基本观点可以概括为：客观世界万事万物是由对象组成的，任何客观实体都是对象，复杂对象可以由简单对象组成；具有相同属性和操作的对象可归纳成类，对象只是类的一个实例；类可以派生出子类，子类除了可以继承父类的全部特性外，还可以有自己的特性；对象之间的联系通过消息传递来维系。概括地说，面向对象的程序＝对象＋类＋继承＋消息。

面向对象技术是软件技术的一次革命，面向对象开发方法起源于面向对象编程语言，随着面向对象编程(OOP)向面向对象设计(OOD)和面向对象分析(OOA)发展，最终形成面向对象的软件开发方法 OMT(object modeling technique，对象建模技术)。这是一种自底向上和自顶向下相结合的方法，而且它以对象建模为基础。具体地讲，面向对象的开发方法就是一种运用对象、类、继承、封装、聚合、消息传递、多态等概念来构造系统的软件开发方法。

传统的结构化软件开发方法是从算法的角度建立系统模型，而面向对象开发方法的系统建模则更符合人们的思维习惯。其中，OOA 是指利用面向对象的概念和方法为软件需求建造模型，以使用户需求逐步精确化、一致化、完全化的分析过程。分析的过程也是提取系统需求的过程，主要包括理解、表达和验证。OOA 的任务是通过分析问题域，提取和整理用户需求，建立系统的概念模型，并用相应的符号系统表示出来。

OOD 则是把面向对象分析阶段得到的需求转变成符合成本和质量要求的、抽象的系统实现方案的过程。从 OOA 到 OOD 是一个逐渐扩充模型的过程，即面向对象设计是用面向对象观点建立求解域模型的过程。OOA 主要是模拟问题域和系统任务，而 OOD 则是面向对象分析的扩充，主要增加各种组成部分。

面向对象实现就是用面向对象程序设计语言来实现面向对象设计，因为该类语言支持对象、多态性和继承，因此比较容易实现；而如果使用非面向对象程序设计语言，就需要特别注意和规定保留程序的面向对象模式的程序结构。

与结构化实现技术中先以模块为单位进行过程设计和编码调试相似，面向对象实现技术是先以类为单位进行操作设计和编码调试的；然后实现类与类之间的关联定义，并进行系统测试；最后交予用户使用并根据使用情况进行维护。

1.3 软件生存周期

任何一个软件产品或者软件系统就像人的生存周期(孕育、诞生、成长、衰亡)一样，都要经历软件定义、软件开发、运行维护直至被淘汰这样一个全过程，这个过程的各个阶段可以分别分成若干阶段，每个阶段相对独立又彼此联系。上一阶段的工作是下一阶段工作的依据，下一阶段是上一阶段的进化，而且更接近于问题的解决。

具体来讲，软件生存周期大致可以分为 7 个阶段：问题定义、可行性分析和项目开发计

划、需求分析、设计、编码、测试、运行和维护。

(1)软件定义时期。这个时期的主要任务是定义所要开发的软件，主要解决的是待开发软件“做什么”的问题。也就是要确定软件开发项目的处理对象和可行性、软件与外界的接口、软件的功能和性能、软件界面以及有关的约束和限制，并且制定本项目的开发进度表。这个时期的工作通常又称为系统分析，由系统分析员采用结构化分析方法或面向对象分析方法完成。通常把软件定义时期又进一步划分为问题定义、可行性分析和项目开发计划及需求分析 3 个阶段。

(2)软件开发时期。本时期的主要任务是设计和实现在前一个时期定义的软件，主要解决的是该软件“怎么做”的问题。这个时期具体包括数据结构和软件结构的设计、算法设计、编写程序、测试，最后得到可交付使用的软件。它通常由 4 个阶段组成：概要设计、详细设计、编码和测试。其中，前两个阶段又称为系统设计，常用的设计方法有结构化设计方法和面向对象设计方法，后两个阶段又称为系统实现。

(3)运行和维护时期。本时期的主要任务是保证软件在一个相当长的时期内能够持久地正常运行。本阶段是软件生命周期的最后一个阶段，也是持续时间最长、代价最大的一个阶段。软件工程学的主要目的就是提高软件的可维护性，并降低维护的代价。具体来讲，所谓软件维护就是在软件已经交付使用之后，为了改正错误或满足新的需要而修改软件的过程。

1.4 软件开发模型

在软件工程中，为什么要引入软件开发模型或软件生存周期模型呢？

为了反映软件生存周期中各个阶段的工作是如何组织和衔接的，根据软件生产工程化的需要，生存周期的划分也有所不同，从而形成了不同的软件生存周期模型(life cycle model，LCM)，或称软件开发模型。软件生存周期模型具体规定了把软件生存周期划分成哪些阶段及各个阶段的执行顺序，它是描述软件过程的一种常见的方式，所以有时也称之为软件过程模型。软件开发模型的种类较多，这里重点介绍 5 类常见的软件开发模型，即瀑布模型、快速原型模型、螺旋模型、喷泉模型和构件组装模型。

1.4.1 瀑布模型

瀑布模型(waterfall model)也称为“生命周期模型”或“线性顺序模型”。瀑布模型是将软件开发活动中各项活动规定为依线性顺序连接的若干阶段，并遵循软件生存周期的划分(3 个时期，7 个阶段)，明确规定每个阶段要完成的任务，最终得到软件系统或软件产品。其各个阶段的工作顺序展开如瀑布流水，因而称为瀑布模型，如图 1-3 所示。

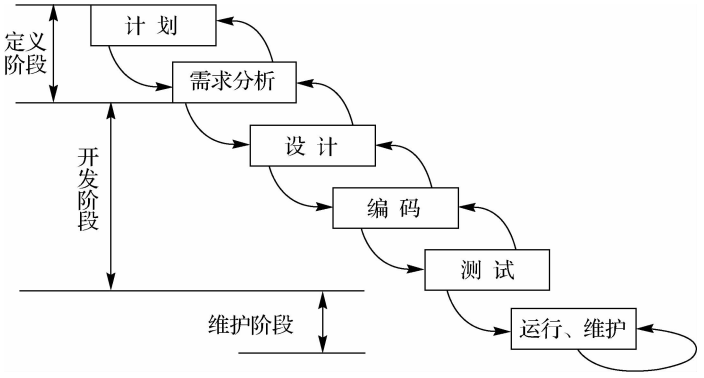


图 1-3　瀑布模型

1)瀑布模型的优点

瀑布模型的优点如下。

(1)符合软件开发的思维过程,并且容易理解和运用。

(2)通过设置里程碑,可以明确每个阶段的任务与目标。

(3)支持结构化软件开发,可为各阶段制订开发计划,进行成本预算。

(4)通过阶段评审,将开发过程纳入正确轨道。

(5)严格的计划性保证软件产品的按时交付。

2)瀑布模型的缺点

瀑布模型的缺点如下。

(1)缺乏灵活性,不能适应用户需求的变化。

(2)开始阶段的小错误会被逐级放大,可能最终导致软件产品报废。

(3)缺乏演化性,返回上一级的开发需要十分高昂的代价。

(4)随着软件规模和复杂性的增加,软件产品成功的机率大幅下降。

3)瀑布模型的适应范围

瀑布模型主要适应于功能和性能明确的小规模软件开发,如学生信息管理系统等。

1.4.2　快速原型模型

1. 基本思想

快速原型模型(rapid prototype model)是相对于生命周期模型而言的另一种系统开发模型。它强调系统设计者与最终用户之间自始至终通力合作,用比较短的时间完成问题空间定义,在获取一组基本的需求定义后,利用高级软件工具的可开发环境,快速地建立一个目标系统的最初版本,并把它交给用户试用;在用户提出修改意见后,再采用迭代法或增量法反复修改、完善产品的功能,形成最终产品/版本。反复进行这个过程,直到得出系统的精确解,即用户满意为止。

2. 原型

原型本是工程设计中的概念,指的是试制品或样品。软件工程中的原型则为系统或软件最终产品的一个试用版本,它指模拟某种产品的原始模型。

3. 快速原型

快速原型是快速建立起来的可以在计算机上运行的程序，它所能完成的功能往往是最终产品所能完成功能的一个子集。

快速原型模型把系统开发大体划分为以下 4 个阶段。

(1)根据用户的需求快速构造一个低成本的用于演示及评价的系统试用版本(原型)。

(2)交付用户运行并听取用户意见或评价。

(3)在用户评价的基础上对原型进行修改或重构。新目标的范围比待修改或补充的原型要大。

(4)直到用户完全满意为止，最后将定型的原型产品转化为最终产品交付给用户。

因此，快速原型模型是增量开发模型，工作顺序呈迭代循环状态，如图 1-4 所示。

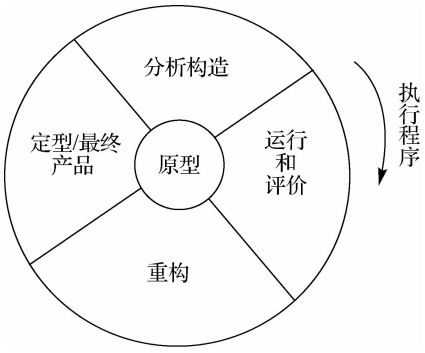


图 1-4　快速原型法的表示

4. 渐增模型

渐增模型(增量模型)是指软件被作为一系列的增量构件来设计、实现、集成和测试，每一个构件是由多种相互作用的模块所形成的提供特定功能的代码片段构成的。

渐增模型的优点：渐增模型在各个阶段并不交付一个可运行的完整产品，而是交付满足客户需求的一个子集的可运行产品。整个产品被分解成若干构件，开发人员逐个构件地交付产品，这样做的好处是软件开发可以较好地适应客户的变化和需求，客户可以不断地看到所开发的软件，从而可以降低开发的风险率。

瀑布模型是一种整体的开发模型，开发的每一阶段必须按线性的顺序来进行，前一阶段的工作没有完成，后一阶段的工作就不能开始。由于需求分析的易变性使得软件开发工作不顺利，同时瀑布模型的每个阶段有不可避免的错误出现，那么延伸到以下的各个阶段错误就会放大。而增量模型是非整体开发的模型，它是逐渐增长和完善的，软件的开发是用增量开发和增量提交的。

5. 快速原型模型的特点

与瀑布模型相比，快速原型模型具有如下特点。

(1)用户参与了软件系统开发的所有阶段，从而使用户的需求可以及时、准确地得到满足，系统的实用性较强。而对于瀑布模型，用户只参与了需求分析阶段，其他阶段只是开发人员“单干”，因此有可能导致最终的系统问题很多，不能投入实际使用。

(2)采用快速原型模型,用户可以及早地接触和使用未来系统的原型,有利于系统后期的使用和维护。而瀑布模型往往需要经过几个月甚至更长的开发时间,用户才能见到最终系统。

(3)使用快速原型模型开发软件,其周期大为缩短,开发费用较少,而瀑布模型的特点是周期长、费用高。

(4)与瀑布模型相比,快速原型模型更适用于解决有不确定因素的问题或用户界面要求较高的中型系统。

1.4.3 螺旋模型

在原型模型的基础上,进行多次原型反复并增加风险评估,就形成了螺旋模型(spiral model),如图 1-5 所示。螺旋模型是在 1988 年由 B. Boehm 正式提出的,它将瀑布模型和快速原型模型结合起来,强调了其他模型所忽视的风险分析,特别适合于复杂的大型软件项目的开发。

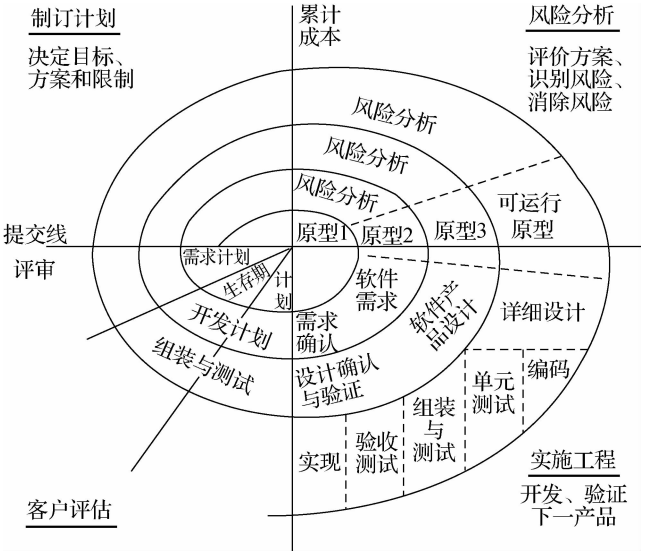


图 1-5 螺旋模型

螺旋模型将软件项目的开发划分为制订计划、风险分析、实施开发和客户评估 4 类活动。沿着螺旋线每转一圈,表示开发出一个更完善的新的软件版本。

图 1-5 显示了螺旋模型的原理,螺旋模型沿着螺线进行若干次迭代,在笛卡尔坐标的 4 个象限上分别表达了如下 4 类活动。

(1)制订计划。确定软件目标,选定实施方案,弄清项目开发的限制条件。

(2)风险分析。分析评估所选方案,考虑如何识别和消除风险。

(3)实施工程。实施软件开发和验证。

(4)客户评估。评价开发工作,提出修正建议,制定下一周期计划。

螺旋模型由风险驱动,强调可选方案和约束条件,从而支持软件的重用,有助于将软件质量作为特殊目标融入产品开发之中。表 1-2 所示为螺旋模型能够解决的一些问题。

表 1-2　螺旋模型能够解决的问题

经常遇到的问题	螺旋模型的解决方案
用户需求不够充分	允许并鼓励用户反馈信息
沟通不明	在项目早期就消除严重的曲解
刚性的体系结构	开发首先关注重要的业务和问题
主观臆断	通过测试和质量保证，做出客观评估
潜在的不一致	在项目早期就发现不一致问题
糟糕的测试和质量保证	从第一次迭代就开始测试
采用瀑布模型法开发	在早期就找出并关注风险

螺旋模型的优点如下。

(1)设计上的灵活性，可以在项目的各个阶段进行变更。

(2)以小的分段来构建大型系统，从而使成本计算变得更容易。

(3)客户始终参与每个阶段的开发，保证了项目不偏离正确方向及项目的可控性。

(4)随着项目的推进，客户始终掌握项目的最新信息，从而能够与管理层有效地交互。

(5)客户认可这种公司内部的开发方式能带来良好的沟通和高质量的产品。

螺旋模型的缺点如下。

很难让用户确信这种演化方法的结果是可以控制的。因为它的建设周期长，而软件技术发展比较快，所以经常出现软件开发完毕之后，与当前的技术水平相比差距较大，无法满足当前用户需求的情况。

螺旋模型适用于对新近开发需求不明确的软件，便于风险控制和需求变更；主要适用于内部的大规模软件的开发，不太适合合同软件的开发。

1.4.4　喷泉模型

随着面向对象系统分析及设计方法的普及，出现了喷泉模型(fountain model)。喷泉模型是由 B. H. Sollers 和 J. M. Edwards 于 1990 年提出的一种新的开发模型。

喷泉模型认为软件开发过程自下而上每个周期的各阶段是相互迭代和无间隙的。迭代是指软件的某个部分常常重复工作多次，相关对象在每次迭代中随之加入渐进的软件成分；无间隙是指在各项活动之间没有明显边界，例如，在分析和设计活动之间没有明显界限等。这正像水喷上去落下来时，既可以落在中间，也可以落在最底部，类似于一个喷泉，如图 1-6 所示。

喷泉模型克服了瀑布模型对软件复用和多项开发活动的集成支持的局限性，由于对象概念的引入，表达分析、设计、实现等活动只用对象类和关系，从而可以较为容易地实现活动的迭代和无间隙，使其开发自然地包括复用。喷泉模型是一种支持面向对象(object oriented，OO)开发方法和生存周期的一种模型。

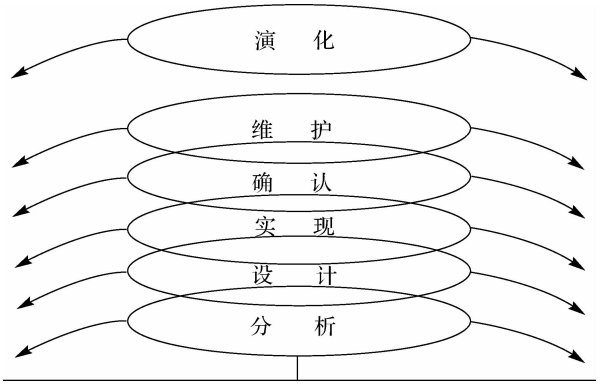


图 1-6　喷泉模型

喷泉模型的优点：喷泉模型的各个阶段没有明显界限，所以开发人员可以同步进行开发，以提高软件项目的开发效率，节省开发时间，适应于面向对象的软件开发过程。

喷泉模型的缺点：由于喷泉模型在各个开发阶段是重叠的，在开发过程中需要大量的开发人员，因此不利于项目的管理。此外，这种模型要求严格管理文档，使得审核的难度加大，尤其是面对可能随时加入各种信息、需求与资料的情况。

1.4.5　构件组装模型

构件组装模型(component composition model)融合了螺旋模型的许多特征，它本质上是演化的支持软件开发的迭代方法。构件组装模型是利用预先包装好的软件构件(有时称为类)来构造应用程序的，如图 1-7 所示。

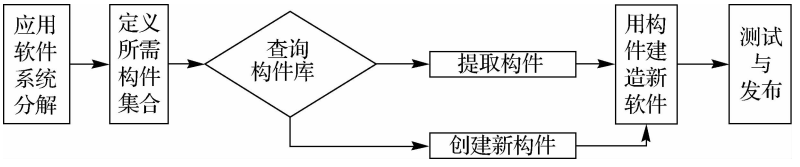


图 1-7　构件组装软件模型

面向对象技术为软件工程基于构件的过程模型提供了技术框架。面向对象范型强调类的创建，类封装了数据和用于操纵该数据的算法。如果经过合适的设计和实现，面向对象的类可以在不同的应用及基于计算机的系统结构中复用。

构件组装模型的优点如下。

(1)构件组装模型导致了软件的复用，提高了软件开发的效率。

(2)构件可由一方定义其规格说明，然后被另一方实现，最后供给第三方使用。

(3)构件组装模型允许多个项目同时开发，降低了费用，提高了可维护性。

(4)可实现分步提交软件产品。

构件组装模型的缺点如下。

(1)可重用性和软件高效性之间的矛盾不易协调。

(2)缺乏通用的组装结构标准，而自定义的组装结构标准引入了较大的风险。

(3)需要精干、有经验的分析和开发人员,一般的开发人员插不上手。

(4)客户的满意度低。

1.5 项目的市场调研

软件工程是一门研究如何用项目化和工程化的手段来开发软件的科学,而项目管理是一门研究如何控制项目开发全过程的科学,它与软件工程既有区别,又相互联系,详细内容读者会在项目 9 中学习到。这里先给出项目的概念及其特点,以便读者对它有一个初步的认识。

1.5.1 项目的概念及特点

什么是项目呢? 美国项目管理协会(PMI)给出的定义是:项目是指在一定的资源约束下,为创造独特的产品或服务而进行的一次性努力。从定义中可以了解到,项目一般要涉及一些人员,以及由这些人员参与的为达成某个目的所采取的一系列活动。

项目的共性是指项目一般都有明确的起止时间、预定的目标,都需要经费和人力资源,都需要项目的全体参与人员为达成预定的目标而共同的工作,即团队协作。此外,项目还具有以下一些特点。

(1)项目具有明确的目标。不同的项目具有不同的预定目标。例如,项目都有特定的交付物,可以是产品,也可以是服务。

(2)项目具有整体性。项目是为实现特定目标而开展的一系列有限任务的有机组合,具有完整的过程。

(3)项目具有一次性。项目是必须能够完成的、临时的、有限的、一次性的、有期限的任务。

(4)项目具有独特性。项目都有一个特定的明确的目标,即明确的独特的产品或服务。

(5)项目具有生命周期。项目是一次性完成的任务,至少要经过项目的启动、计划、实施、收尾 4 个阶段。具体包括项目的前期调研、可行性分析、实际开发、测试、验收与移交等基本环节。

此外,项目还具有以下辅助特点。

(1)项目具有约束性。项目只能在一定的约束条件(如资金、时间、人力和物力资源等)下进行。

(2)项目具有雇主,即委托人。项目具有提供必要资金、环境以实现目标的实体。

(3)项目具有不确定性。项目开发存在影响过程实施的风险性,所以项目的开发过程和结果具有不确定性。

(4)项目结果具有不可逆转性。项目实施结束后,结果就确定了,是不可逆转的。

综上可见,软件项目管理很有必要,它是一门研究如何控制项目开发全过程的科学,同时也是一门艺术。软件项目管理平衡竞争目标,管理风险并克服制约因素,从而最终能够交付同时满足客户和用户双方需要的产品。

1.5.2 学生信息管理系统问题的提出

随着高校规模的扩大和各种业务的扩展，利用人工进行学生信息管理的传统管理模式和教学手段存在很多不足之处，主要表现在以下几方面。

(1)效率低、保密性差，时间一长将产生大量的文件和数据。

(2)信息管理规范性不够，易于出错。

(3)不便于查找、更新和维护等。

例如，对于信息管理工具，有的用 Word 表格管理，有的则用 Excel 表格管理，对于一个字段，如“学号”等，有的用 XH 表示，有的则用 number 表示，即使采用统一的数据库管理，数据维护中往往也需要转换，而且数据的约束和关联不够等。这些情况给学生信息的管理带来了很大困难，严重影响了教育工作者的积极性和工作效率。

随着计算机科学技术和网络技术的日渐成熟，使用先进的信息管理系统代替传统的人工模式来进行信息管理，实现对学生等信息的有序化、科学化和网络化管理已成为各高校信息系统管理的发展新趋势，而高效、准确的高校学生信息管理系统(student information management system，SIMS)是提高高校办学质量、培养一流人才必不可缺的重要手段。

学生信息管理工作是一个系统工程，贯穿于学生在校学习的全过程和各个方面，通常从新生入学报到开始就要录入相关信息，一直到学生毕业离校为止，内容包括学生学籍信息管理、学生收费管理、学生课程管理、学生成绩管理、学生表现管理、学生毕业实训课程管理、学生毕业设计管理、毕业生离校手续办理系统管理、毕业生就业指导管理等各个方面，具有工作量大、分类细、项目多和覆盖面广等特点，本书中的学生信息管理系统的开发只是其中一个子集。

1.5.3 国内外开发现状

实际调查表明，计算机在管理方面的应用开始于 1954 年，当时美国首先用计算机处理工资单。之后，计算机在处理信息管理方面发展迅速。例如，20 世纪 60 年代，美国计算机在管理中应用项目不到 300 项，但到了 1975 年达到了 2 670 项。而现在，美国在财务会计上的工作 90%由计算机来完成；物资管理中 80%～100%的信息处理由计算机完成；在计划管理中则占到了 80%～90%。根据计算机应用方面发展较快的国家统计，计算机用于经济管理的约占 80%；用于科技运算的占 8%；用于生产过程控制的占 12%。我国在全国范围内推广计算机在管理中的应用是在 20 世纪 70 年代末开始的，虽然起步较晚，但近几年纵向发展却较快，特别是微型计算机的出现和网络的普及为信息化处理提供了物美价廉的手段，对推动我国管理信息系统的现代化起了重要的作用。

但是，目前我国计算机产业的发展仍然存在一些问题。从横向发展来看，我国计算机应用水平和企事业信息化程度还比较低，上网企业与上网家庭数量及信息技术在企业与家庭中的应用尚不广泛，与发达国家及某些发展中国家相比都有很大差距。据国家统计中心研究，我国信息化能力不仅远落后于美国、日本等发达国家，也落后于新加坡、韩国、菲律宾、埃及、印度等发展中国家；我国网民占人口的比例仅为 3.5%，还偏低。国内计算机应用发展还

很不平衡，各地区信息化指数高低相差还很悬殊，系统集成、信息服务水平还有待提高，不仅缺乏计算机应用有关标准和规范，而且不统一，急需加强。

随着国内学校规模的不断扩大，学生数量急剧增加，有关学生的各种信息量也在成倍增长。面对庞大的信息量，需要开发学生信息管理系统来统一提高各院校学生管理工作的效率。通过这样的系统，可以做到信息的规范化管理、科学统计和快速的查询，从而减少各院校管理方面的工作量。随着计算机和网络的普及，建立一个客户机/服务器(C/S)或浏览器/服务器(B/S)结构的学生信息管理系统，学生便可以通过网络来选课并且查询自己的有关信息，从而使得学生信息管理工作更加系统化、规范化和自动化，极大地提高了学校管理学生工作的效率。目前，开发 SIMS 的结构形式有上述 C/S、B/S 及两者相结合 3 种结构；开发方法有生命周期法、面向对象法、原形法等；开发方式有独立开发、委托开发、合作开发、购买软件等；开发平台有 Windows XP、Windows NT、Netware 等。

本书中 SIMS 开发的主要目的是让学生、教师或来访者可以方便地进行查询等相关操作，对学生的信息管理能统一化和规范化，功能块的相应操作分别由其对应的子模块来实现。通过网络把整个学校的学生信息汇聚起来，为学校教育主管部门提供全面、及时的学生信息，以方便管理，同时亦可通过校园网扩展各类信息化应用，共享学生信息。学校可通过一个简化的学生信息管理系统，使学生信息管理工作系统化、规范化和自动化，从而达到提高学生信息管理效率的目的。

此外，在信息化社会和知识经济时代，信息化、数字化校园建设是国内外高校建设的热点。在国外，数字化校园建设具有发展早、起点高、投资大和速度快等特点，而国内较多关注数字资源的提供，较少强调高度的系统集成化，更多关注学生活动本身，注重协同科研，而信息管理系统在数字化校园中相对较弱。目前，全国重点高校数字化校园建设研讨会已经召开，各高校也纷纷设立了数字化校园建设项目，数字化校园建设已成为各高校进行信息化建设的新热点。

总之，使用计算机对学生信息进行管理，具有手工管理无可替代的优点。例如，检索迅速、查找方便、易修改、可靠性高；存储量大、数据处理快捷；保密性好、寿命长、成本低、便于打印等。这些优点能极大地提高学生信息管理的效率，也是使学校走向科学化、正规化管理，与世界接轨的重要条件。在有了功能强大的 Internet 之后，还可及时地向学生家长传递学校的教育方针及该生在校的最新成绩，从而有助于家长和学校的沟通。因此，开发这样一套学生信息管理系统软件十分有必要，不仅可以树立良好的学校形象，而且可以提高学校的工作效率。

1.6 认识 Visio 建模工具

常见的软件开发建模工具有 Microsoft Visio 2007、IBM Rational Rose 2003/2007、Play CASE、Power Design、Microsoft Project 等，本书主要介绍前两种常用的计算机辅助软件工程(CASE)工具。

Visio 2007 是功能强大的绘图工具，能够制作精美的图表，还可以在文档中插入业务流

程图、组织结构图、数据库模型图、软件图表、办公室布局图或其他网络图形等，用直观、具有说服力的方式来描述和交流信息。

1999 年，Microsoft 公司并购了 Visio 公司，不久便推出了新版的 Visio，并且让 Visio 成为 Microsoft Office 家族中的一员。使用 Office Visio 2007，可以通过多种图表直观地记录、设计和完全描述业务流程和系统的状态；此外还可将图表链接至基础数据，并将数据链接至形状，可以提供更为完整的画面，从而使图表更智能、更有用。Visio 2007 提供的模板、形状和绘图工具可用于创建有效的业务图表和技术图表，以用来分析业务流程、安排项目日程、形象地表达思维过程及绘制组织结构图等。使用 Office Visio 2007 将图表与数据集成，可以全面了解一个流程或系统。在软件工程的需求分析阶段还可以利用 Visio 绘制系统实体-关系(E-R)图和数据流图等各种图形。

1. 安装 Visio 2007，认识 Visio 环境

Office Visio 2007 有两个独立版本：Office Visio Professional 2007 和 Office Visio Standard 2007。虽然 Office Visio Standard 2007 与 Office Visio Professional 2007 的基本功能相同，但前者包含的功能和模板是后者的子集，因此，通常安装和使用 Office Visio Professional 2007。

(1)运行 Office Visio Professional 2007 的安装程序，按要求输入正确的产品密钥，单击“安装”按钮即可弹出图 1-8 所示的对话框，单击“立即安装”按钮后，弹出图 1-9 所示的“安装进度”对话框，然后按提示操作即可完成安装。

图 1-8　安装方式选择界面

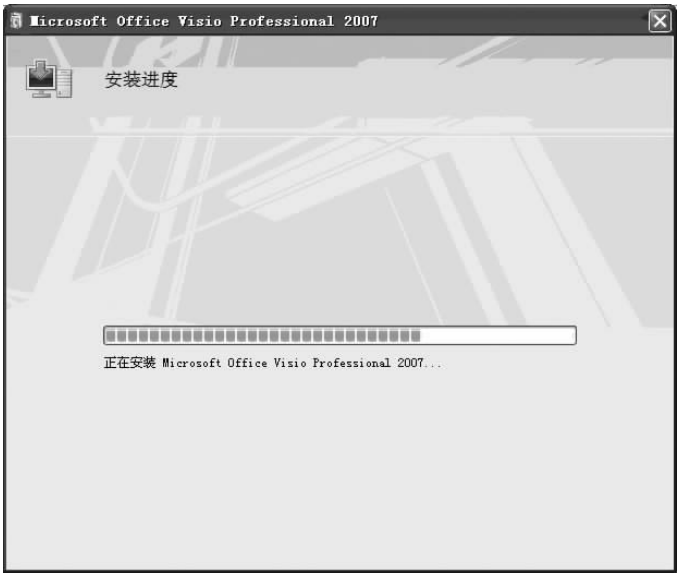


图 1-9　安装进度界面

(2)安装完成后,在“开始”菜单中执行“程序”→Microsoft Office→Microsoft Office Visio 2007 命令或双击桌面上的 Visio 2007 快捷方式图标,即可打开 Visio 2007 的启动界面,如图 1-10 所示。

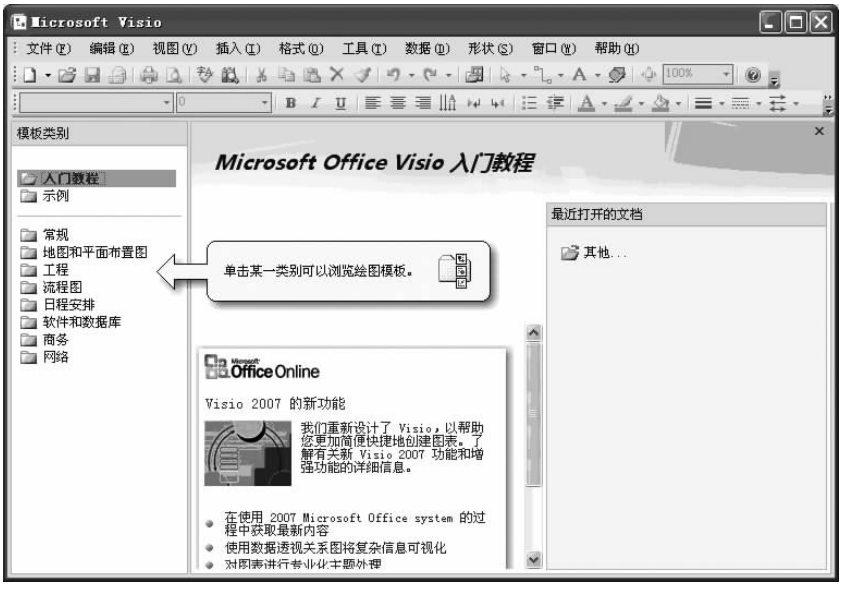


图 1-10　Visio 2007 的启动界面

2. 练习绘制 C/S 结构和 B/S 结构示意图

在 Visio 2007 环境下,通过打开一个模板来创建 Microsoft Office Visio 图表。启动 Visio 2007 后,在图 1-10 所示的工作环境中可以单击某一类别的模板来查看具体模板的缩略图,如图 1-11 所示。

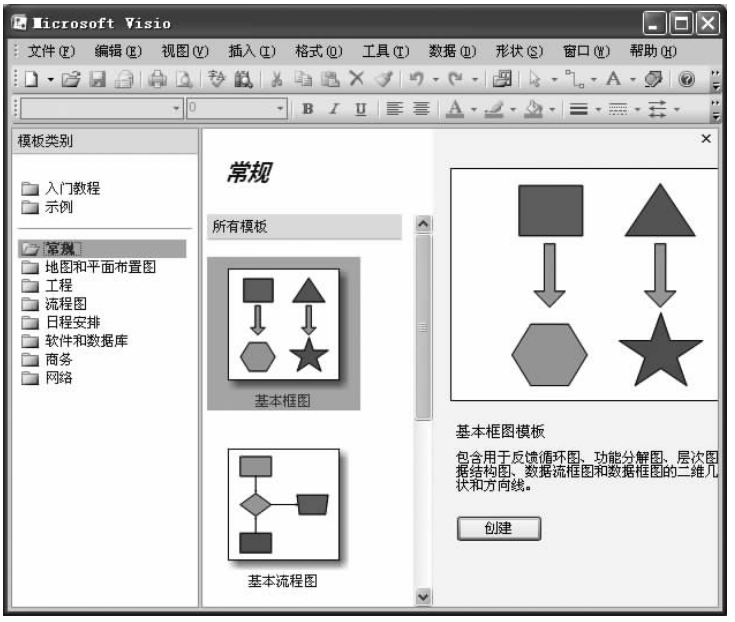


图 1-11　查看具体模板的缩略图

选择某一具体模板后，在绘图页的左侧打开一个或多个模具，模具包含创建图表所需的形状。例如，打开基本框图模板时，可以打开一个绘图页和包含基本流程图形状的模具，如图 1-12 所示。

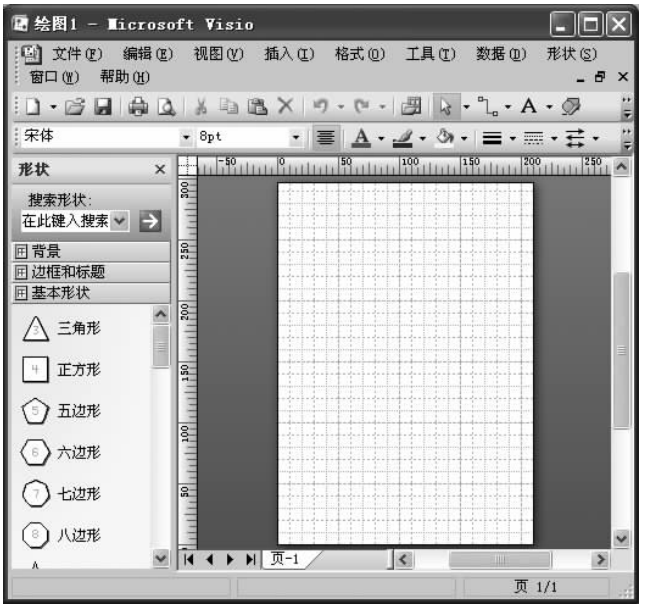


图 1-12　基本流程图绘图界面

提示：执行“文件”→“新建”→“框图”→“基本框图”菜单命令，也可以进入图 1-12 所示的界面。

(1)执行“文件”→“页面设置”菜单命令,弹出“页面设置”对话框,默认打开“打印设置”选项卡。在其中将图纸设置为横向放置,效果如图 1-13 所示。

(2)在模板中绘制两个矩形,两个矩形中间保持一定距离。在其中第一个矩形上双击并输入文字“客户机”,并设置字号为 24,隶书,蓝色;在另一个矩形上也双击并输入文字“服务器”,设置字号为 24,隶书,红色。

(3)单击“连接线工具”按钮,在两个矩形之间画两条连线,然后选择连线,单击“线条端点工具”按钮右边的下三角按钮,再单击“指针工具”按钮取消连线;接着单击菜单栏上的“文本工具”按钮A,在两条线的上方和下方分别单击并输入“请求(Request)”和“响应(Response)”标签内容,字号设为 12,然后在标签外任意位置单击取消文本输入模式。

(4)最后单击“文本工具”按钮A,在图的正下方居中处单击并输入“(a)C-S 结构”,字体设为黑体,加粗。

接下来用同样的方法绘制“(b)B-S 结构图”,最终效果如图 1-13 所示。

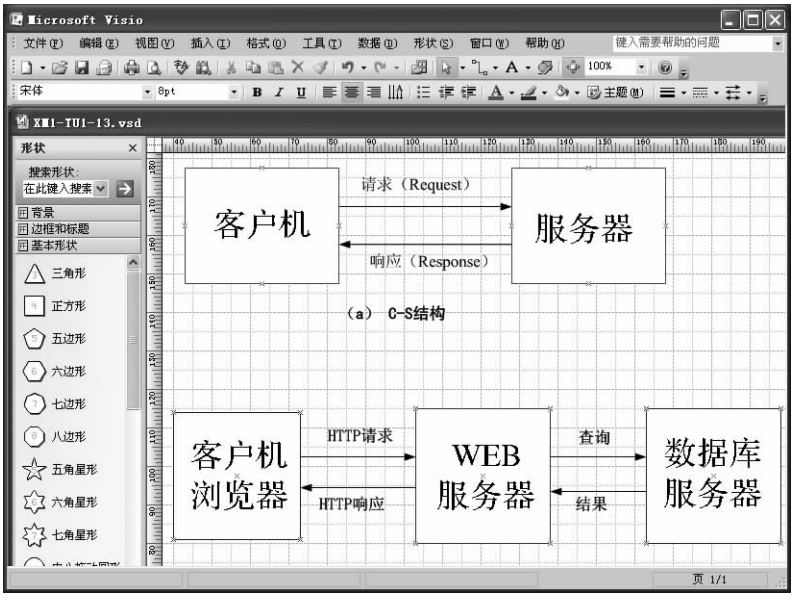


图 1-13　C/S 结构和 B/S 结构示意图

注意: Visio 2007 文件的保存格式主要有以下 3 种。

- 绘图文件(*.vsd)。绘图文件用来存储绘制好的图表,可以包含多个绘图页面及所使用的样板。
- 模具文件(*.vss)。模具文件用来存储各种图件的文件格式。
- 模板文件(*.vst)。模板文件可以将绘图文档与开启的样板一起储存,并事先作好环境设定。

3. 绘制考勤管理系统数据流图

下面以某考勤管理系统数据流图的绘制为例进行介绍,如图 1-14 所示。

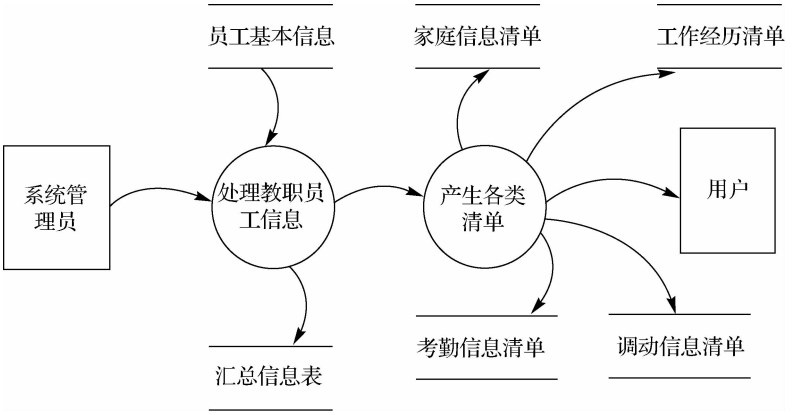


图 1-14　考勤管理系统数据流图

启动 Visio 2007，进入图 1-10 所示的工作环境，在模板类别中选择“流程图”→“数据流图表”模板，如图 1-15 所示。单击其中的“创建”按钮，进入数据流图绘制界面，可以看到窗口界面左侧是绘图模具，里面放置了大量绘制数据流图所需的图件。

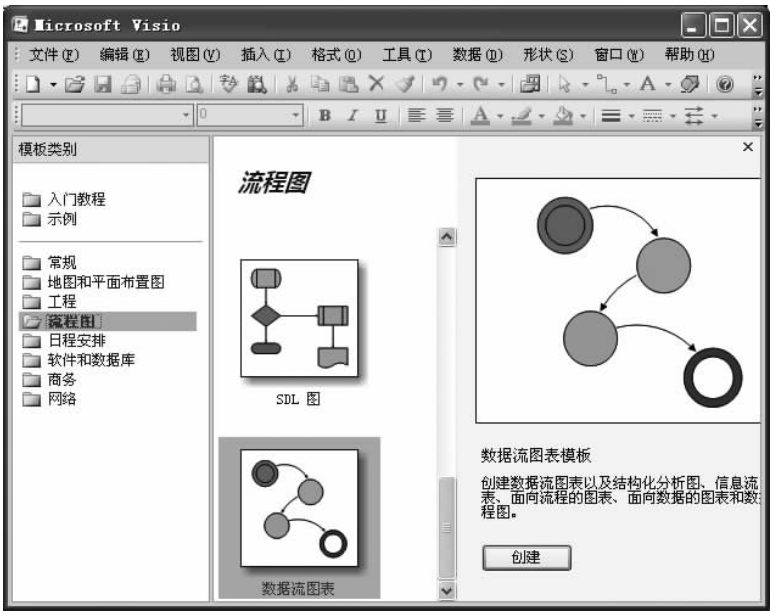


图 1-15　选择数据流图模板

(2)执行“文件”→“页面设置”菜单命令，弹出“打印设置”对话框，在其中将图纸设置为横向放置，并将绘图比例调整到 100%。

(3)依次拖动左侧的“实体 1”、“数据流程”等图件到绘图页上，并分别双击图件输入图 1-16 所示的文字。

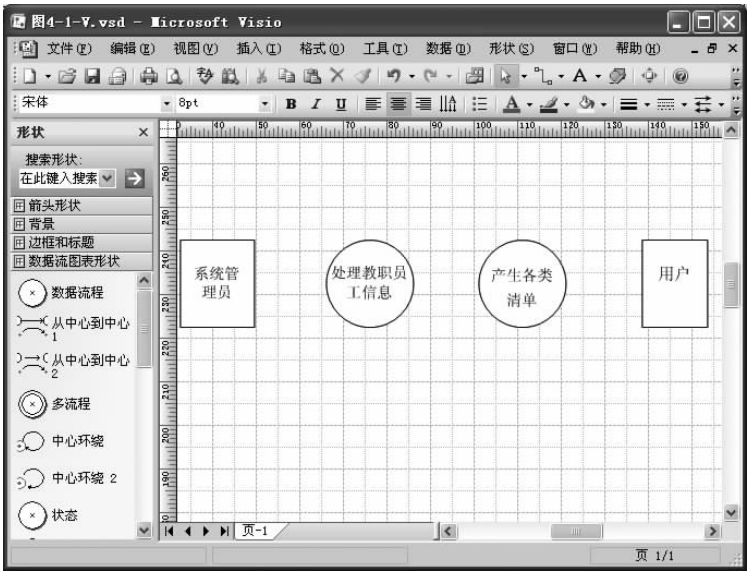


图 1-16　绘制各种图件

(4)可以根据需要调整每个图件的大小,再拖动“从中心到中心 1”、“数据存储”等图件将各实体连接起来,如图 1-17 所示。

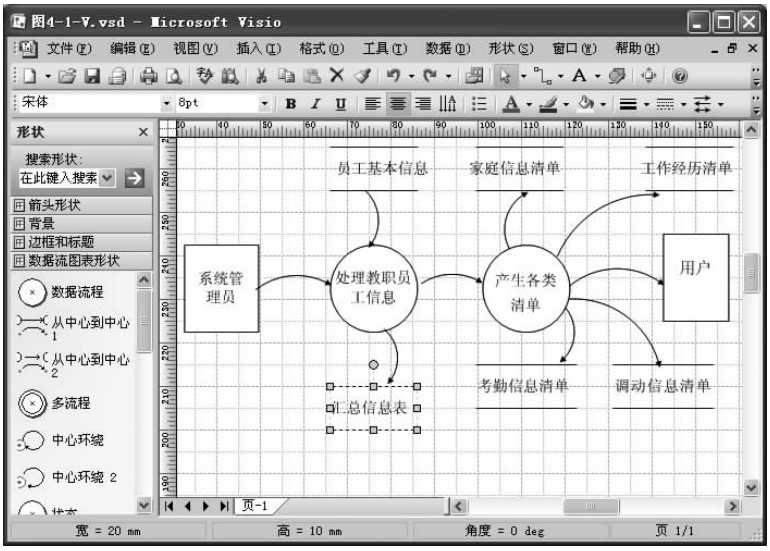


图 1-17　连接各种图件

至此,考勤管理系统数据流图制作完成。如果想要图形更美观和专业,读者还可以给它添加背景页及页眉页脚等,最后将其保存成以.vsd 为扩展名的文件即可。如果读者在学习过程中还有任何疑问,可以执行菜单栏中的“帮助”菜单命令获取帮助信息。

1.7 实训：Microsoft Office Visio 2007 的入门操作

1. 实训的目的和任务

(1)了解 Microsoft Visio 2007 工具软件的特点、安装步骤及工作环境；熟悉 Visio 2007 的基本功能和基本操作。

(2)能绘制一些基本的框图和简单的流程图。

2. 实训环境

(1)Microsoft Visio Professional 2007 软件。

(2)Windows XP 操作系统以及能够安装 Visio 2007 软件的计算机。

3. 实训内容与步骤

(1)Visio 2007 的安装过程。

(2)Visio 2007 的工作环境及应用。

4. 实训练习

(1)利用 Visio 2007 绘制本院校的组织结构图。

(2)利用 Visio 2007 绘制 B/S 系统结构实现流程图，如图 1-18 所示。

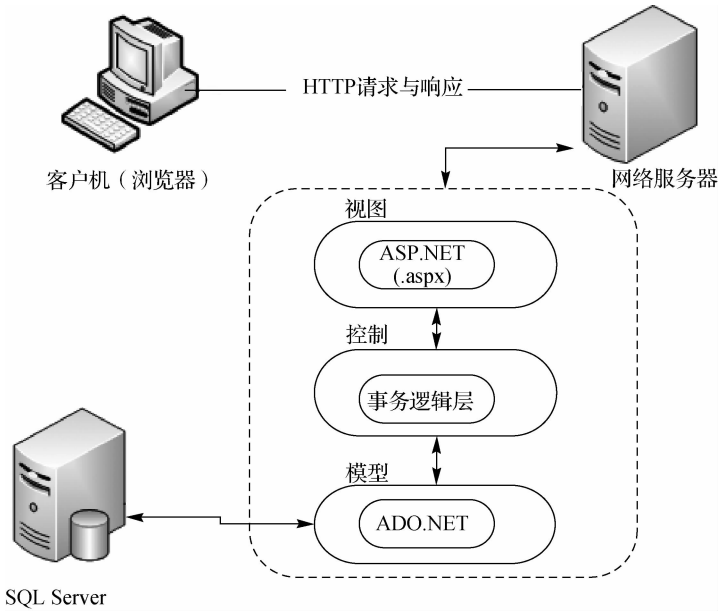


图 1-18　B/S 系统结构实现流程图

1.8 习　　题

一、选择题

1. 硬件与软件的最大区别是(　　)。

A. 软件产品是逻辑产品,硬件产品是物质产品

B. 软件产品是以手工生产方式生产的,硬件产品则是以大工业生产方式生产的

C. 软件产品不存在老化问题,硬件产品存在老化问题

D. 软件产品容易复制,硬件产品很难复制

2. 软件是指(　　)。

A. 按事先设计的功能和性能要求执行的指令系列

B. 使程序能够正确操纵信息的数据结构

C. 与程序开发、维护和使用有关的图文资料

D. 计算机系统中的程序和文档

3. “软件工程的概念是为解决软件危机而提出的”这句话的意思是(　　)。

A. 说明软件工程的概念,即工程的原则和思想、方法可能解决当时软件开发和维护中存在的问题

B. 说明软件工程这门学科的形成是软件发展的需要

C. 强调软件工程成功地解决了软件危机的问题

D. 说明软件危机存在的主要问题是软件开发

4. 软件工程的目标是(　　)。

A. 生产满足用户需要的产品

B. 以合适的成本生产满足用户需要的产品

C. 以合适的成本生产满足用户需要的、可用性好的产品

D. 生产正确的、可用性好的产品

5. 软件开发中大约要付出(　　)的工作量进行测试和排错。

A. 20%　　B. 30%　　C. 40%　　D. 50%

6. 软件生存周期中时间最长的是(　　)阶段。

A. 可行性研究　　B. 概要设计　　C. 测试　　D. 维护

7. 瀑布模型的主要特点是(　　)。

A. 将开发过程严格地划分为一系列有序的活动

B. 将过程分解为阶段

C. 提供了有效的管理模式

D. 缺乏灵活性

8. 软件开发方法是(　　)。

A. 指导软件开发的一系列规则和约定　　B. 软件开发的步骤

C. 软件开发的技术　　D. 软件开发的思想

二、简答题

1. 什么是软件危机？软件危机的主要表现是什么？怎样消除软件危机？
2. 什么是软件？软件和程序的区别是什么？
3. 什么是软件工程？
4. 什么是软件生存周期？软件生存周期为什么要划分阶段？
5. 比较几种软件开发模型的优缺点。
6. 如果要开发高校教材采购系统，你认为采用什么开发模型较为合适？请说明理由。

项目 2

面向对象的建模语言 UML 及工具

随着面向对象软件开发方法的提出和推广，出现了许多建模语言和建模方法，如 OMT 方法、Booch 开发方法、Cord 方法和 Yourdon 方法等。它们形式多样，各具所长，以致采用不同的建模语言描述系统的用户之间很难进行有效的交流，于是统一建模语言(unified modeling language，UML)应运而生，它把 Booch，Rumbaugh 和 Jacobson 等各自独立的面向对象分析(OOA)和面向对象设计(OOD)方法中的特色组合成一个统一的方法，并从其他方法和工程实践中吸收了许多经过实践检验的概念和技术。UML 于 1996 年发布，并于 1997 年 11 月为 OMG(对象管理组织)所采用，现已成为业界公认的面向对象建模语言。

UML 是第三代面向对象的开发方法，为不同领域的用户提供了统一的交流标准，即 UML 建模图。UML 定义良好、易于表达、功能强大且普遍适用，它融入了软件工程领域的新思想、新方法和新技术，其作用域不仅限于支持面向对象的分析与设计，还支持从需求分析开始的软件开发全过程。UML 应用领域很广泛，适用于以面向对象技术来描述的任何类型的系统，而且也适用于系统开发的不同阶段，从需求规格描述直至系统完成后的测试和维护等。商业建模(business modeling)也可用于其他类型的系统。

UML 建模软件是指实现 UML 建模功能的工具软件。最著名的 UML 可视化面向对象建模工具是 IBM Rational Rose，它提供了一组基于 UML 标准的视图，可以实现对软件开发过程的建模。项目 2 主要介绍 UML 及 Rose 2007 的基本图形及其使用。

学习目标

- **理解并掌握**：UML 的内容及特点、静态建模方法和动态建模方法。
- **学会**：Rose 2007 的安装和初步使用。
- **了解**：统一建模过程。关注 Rose 2007 的使用方法。

2.1 UML 概述

面向对象的软件开发方法最早产生于 20 世纪 70 年代中期，面世之后很快受到了计算机软件界的青睐，并成为 90 年代主流的开发方法。它是一种把面向对象的思想应用于软件开发过程中指导开发活动的系统方法，简称 OO 方法，它是建立在对象、类和继承概念基础上的方法。

UML 方法结合了 Booch 方法、OMT 方法和 OOSE 方法的优点，并统一了符号体系。目前，在多数大型软件企业的正规化开发流程中，开发人员普遍使用 UML 进行面向对象模型的建立。作为一名软件开发人员，必须正确理解并学会 UML 的使用。

提示：Visio 2007、UML 及 Rose 建模工具对学习现代软件工程课程起到了“催化剂”的作用。使用 CASE(计算机辅助软件工程)建模工具有助于理解软件工程抽象的原理和方法。

2.1.1 UML 的产生与发展

通常面向对象软件开发方法有很多，各种建模方法有着不同的建模符号体系，而且各自的建模语言也各具所长。其中，3 个最为流行的面向对象方法是：Booch 方法(由 Grady Booch 提出)、OMT 方法(由 James Rumbaugh 提出的对象建模技术)和 OOSE(object-oriented software engineering，面向对象的软件工程)方法(由 Ivar Jacobson 提出)。它们各具优点和不足，Booch 方法注重设计，OMT 方法注重分析，OOSE 方法是一种用例驱动的方法。在 OOSE 方法中，用例模型充当整个分析模型的核心，也是分析阶段、构造阶段和测试阶段的基础，它是一种在 OMT 方法基础上对功能模型进行补充并指导系统开发活动的系统方法。目前，OOSE 方法已经在很大程度上让位给基于 RUP(rational unified process，统一软件开发过程)技术的 UML。

20 世纪 90 年代中期，由提出面向对象方法的 3 位著名专家等人发起，在 Booch 方法、OMT 方法和 OOSE 方法的基础上推出了统一建模语言(UML)。1997 年 11 月，对象管理组织(OMG)确定将 UML 作为基于面向对象技术的标准建模语言。UML 的具体发展历程如下。

(1)1997 年 1 月，UML 1.0 被提交给对象管理组织。

(2)1997 年 11 月，UML 1.1 被 OMG 采纳作为基于面向对象技术的标准建模语言。

(3)1998—2005 年分别发布了 UML 1.2、UML 1.3、UML 1.4、UML 1.5 以及 UML 2.0。

(4)2007 年发布了 UML 2.1.1 和 UML 2.1.2。

(5)2009 年发布了 UML 2.2。

2.1.2 UML 的主要内容及特点

设计 UML 的主要目标是：统一不同的建模语言，统一开发阶段和不同的软件应用领域，并与多种不同的开发过程并存。

1. UML 的主要内容

UML 统一了面向对象建模的基本概念、术语及图形符号，建立了便于交流的通用建模

语言。

作为一种建模语言，UML 的定义包括 UML 语义和 UML 表示法两个部分。UML 语义通过元模型来精确定义，用自然语言描述。UML 表示法定义了图形符号的表示，为开发者和开发工具如何使用这些图形符号和文本语法进行系统建模提供了标准。

UML 用来描述模型的内容主要由 4 部分构成：事物、关系、视图和图。UML 内容结构的具体组成如图 2-1 所示。

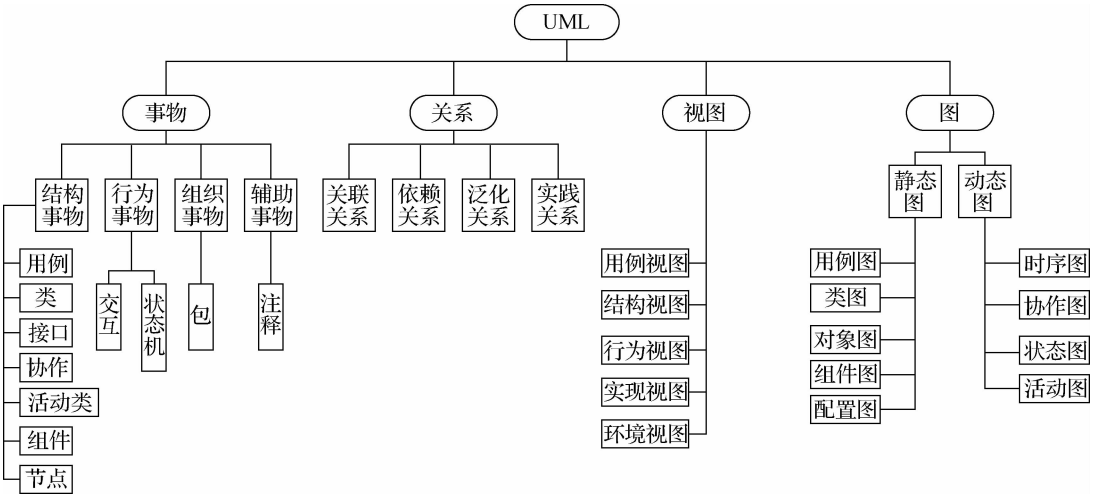


图 2-1　UML 内容结构的具体组成

在 UML 中用 5 种不同的视图来表示一个系统，这些视图从不同的侧面描述系统，每一个视图由一组图形来定义。使用 UML 时，要从不同的角度观察系统。

(1)用例视图。用例视图描述系统的外部特性、系统功能等，可根据它确定重要的设计，即类、包和子系统的设计。

(2)结构视图。结构视图描述系统的静态结构(类图、对象图)。

(3)行为视图。行为视图描述系统的动态行为(交互图、状态图、活动图)。具体描述系统在运行时的并发性，包括任务、线程、进程及其相互作用等。

(4)实现视图。实现视图表示系统的实现特征(源程序、构件、数据文件、可执行程序等)，常用组件图表示。

(5)环境视图。环境视图描述系统的物理配置特征，常用配置图表示。

总之，UML 分析建模着重描述系统的用例模型和结构模型，设计建模着重描述行为模型、实现模型和环境模型。

UML 中图与视图之间是有区别的，图可以由静态图和动态图两大类来定义，或者细分为以下 5 类。

(1)用例图(use case diagram)。用例图从用户角度描述系统功能，并指出各功能的操作者。

(2)静态图(static diagram)。静态图包括类图、对象图和包图。

(3)行为图(behavior diagram)。行为图描述系统的动态模型和组成对象间的交互关系，包括状态图和活动图。

(4)交互图(interactive diagram)。交互图描述对象间的交互关系,包括时序图和协作图。

(5)实现图(implementation diagram)。实现图包括组件图和配置图。

2. UML 的主要特点及用途

UML 的主要特点可归纳如下。

1)统一标准

UML 不仅统一了 Booch、OMT 和 OOSE 等方法中的基本概念,还吸取了面向对象技术领域中其他流派的长处,其中也包括非 OO 方法的影响。它已经成为 OMG 的正式标准。

2)面向对象

UML 支持面向对象技术的主要概念,它提供了一批基本的表示模型元素的图形和方法,能简洁明了地表达面向对象的各种概念和模型元素。

3)可视化建模

UML 是一种图形化语言,利用 UML 的模型图形能清晰地表示系统的逻辑模型或实现模型。UML 还提供了语言的扩展机制,用户可以根据需要增加定义自己的构造型、标记值和约束等。

4)独立于过程,表达能力强

UML 是系统建模语言,不依赖特定的开发过程。UML 强大的表达能力使它可以对各种类型的软件系统建模,包括商业领域的业务过程。

5)容易掌握和使用

UML 概念明确,建模表示法简洁明了,图形结构清晰,容易掌握和使用。

6)与编程语言的关系

支持 UML 的一些 CASE 工具(如 Rose),可以根据 UML 所建立的系统模型自动产生 Java、C++ 等代码框架。

UML 的主要用途可概括为以下两个方面。

1)在不同类型的系统中建模

UML 的目标是以面向对象的方式来描述任何类型的系统。最常用于建立软件系统模型,但它同样可以用于描述非软件领域的系统,如机械系统、分布式系统、企业机构或业务过程,以及处理具有复杂数据的信息系统、具有实时要求的工业系统和工业过程等。

2)在软件开发的各个阶段建模

UML 在需求分析阶段可以用用例图捕获用户的功能需求及不同角色对系统的功能要求。在分析阶段主要关心问题域中类及它们之间的相互关系,类图和对象图描述系统的静态结构,时序图、协作图、状态图和活动图描述系统的动态行为。在设计阶段开始定义软件系统的解决方案细节,如加入处理用户接口、数据库、通信及并行性等问题的类,即把分析阶段的结果加入新的类扩展成为软件系统的技术解决方案细节。实现阶段的任务是把来自设计阶段的类转换成为某种面向对象程序设计语言的代码。在测试阶段的不同测试中,UML 建模图有不同的用途。例如,单元测试可使用类图和类规格说明,集成测试可使用组件图和协作图,系统测试可采用用例图来验证系统的功能,验收测试可由用户进行,也可采用子系统用例图验证不同子系统在分析阶段确定的所有需求是否能实现。

3. UML 的基本图示

UML 提供了统一的图示供分析和设计人员使用。下面介绍 UML 建模的共享元素和关系图示的概念。

1)模型共享元素

模型共享元素包括类、对象、接口、用例、包、组件和节点等，如图 2-2 所示。

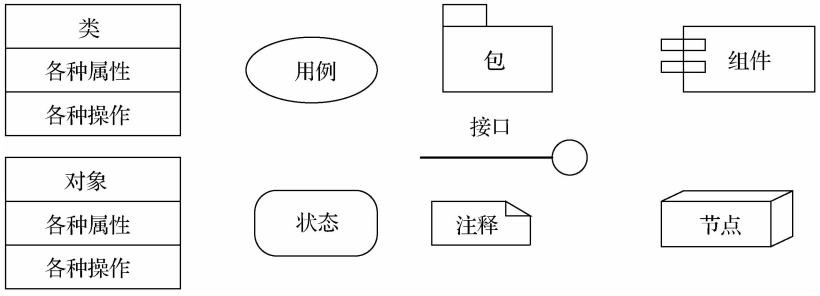


图 2-2　模型共享元素

具体介绍如下。

(1)对象是客观世界的实体，是可以区别的一种存在。

(2)类表示对现实世界中一类具有共同结构特征、行为特征的事物(即对象)的抽象。类是抽象的，是面向对象编程的基础；而对象是具体的，它是类的实例。

(3)接口是类或组件用来为其他类或组件提供特定服务的一组操作的集合。

(4)用例用来描述系统对一个参与者(人或其他系统)所提供的一项服务或功能。它是为达到某个目标所涉及的一系列场景的集合。

(5)组件(构件)是真实驻留在计算机系统中的软件实体单元，是对数据和方法的简单封装。一个组件代表一个系统中可实现的物理部分(并提供了一系列可用的接口)，包括软件代码，如源代码、二进制代码、可执行代码、脚本或者命令文件等。

(6)节点是计算机的资源，包括带有处理器的计算机或其他硬件设备。

2)关系图示

模型共享元素之间的常见关系有关联、依赖、泛化和实现，此外还有聚集和组合，如表 2-1 所示。

表 2-1　模型共享元素之间的常见关系

关　系	功　能	表 示 法
关联	类实例(对象)之间连接的描述	——→
依赖	表示两个模型元素之间的依赖关系	- - - - - - - - - - →
泛化	表示一般与特殊的关系，适用于继承操作	——▷
实现	表示类和接口之间的关系	- - - - - - - - - - ▷
聚集	表示聚集对象由部分对象组成	——◇
组合	是一种特殊的聚集关系	——◆

具体介绍如下。

(1)关联。关联表示模型元素通常要和其他模型元素发生关联。关联具有多种形式、多重性问题,如一对一、一对多关联。在 UML 中关联用一条带箭头的直线符号来表示。

(2)依赖。依赖表示一个模型元素以某种方式依赖于另一种模型元素,如一个类使用了另一个类。在 UML 中依赖用从依赖类到被依赖类的带箭头的虚线符号表示。

(3)泛化。泛化表示一般与特殊的关系,如一个子类继承了其他更一般类的属性和操作。在 UML 中泛化用从子类画一条带空心三角形箭头的连线指向父类的符号来表示。

(4)实现。实现表示类和接口之间的关系。在 UML 中实现关系用一个带空心三角形箭头加虚线的符号表示,箭头指向接口。

(5)聚集。聚集是关联的一种,通常聚集对象由部分对象组成,即整体与部分关联。在 UML 中在整体和部分之间用带空心菱形箭头的连线连接。

(6)组合。组合是一种特殊的聚集关系。在一个组合对象中,部分对象只能作为组合对象的一部分与组合对象同时存在。在 UML 中,整体和部分之间用带实心菱形箭头的连线连接。

2.2 静 态 建 模

UML 的静态建模机制包括 5 种图,即用例图、类图和对象图、包图、组件图以及配置图,使用它们可以建立系统的静态结构,其中,组件图和配置图也可以归为 Rose 其他模型类建模,因此放在最后单独学习。

2.2.1 用例图

用例图(use case diagram)是显示系统中角色和用例关系的图。画好用例图是由软件需求到最终实现的第一步。用例图用于建立系统所要实现的功能,它主要对系统、子系统或类的行为进行建模。用例图表示系统从外部想要实现的行为,而不关心这些行为具体是怎样实现的。

UML 中的用例图描述了一组用例、参与者以及它们之间的关系,因此用例图包括 3 方面内容:用例(use case)、参与者(actor)、关系,而关系主要包括依赖、泛化和关联关系。

参与者即角色,它是系统外部的一个实体(可以是任何事物或人所扮演的角色)。用例是系统的使用过程或要执行的动作序列,用来描述某个参与者使用系统所完成的功能。关联表示角色与用例之间驱动和反馈的关系,也可以表示用例间的包含与扩展关系。在 UML 中,用例图模型如图 2-3 所示,参与者用名字写在下面的人形图标来表示,用例用椭圆来表示,连线表示它们之间的关系。

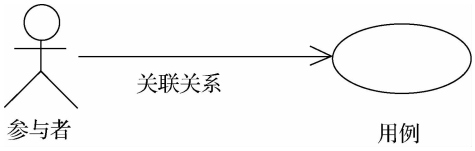


图 2-3　用例模型表示

【例 2-1】　某图书馆借还系统用例图如图 2-4 所示。图中形象地表达了用例图各元素之间的关系，带箭头的实线表示关联关系，带空心箭头的实线表示继承关系，即教师和学生除了继承借阅者的借书和还书功能外，还可以自行到图书馆中查找图书，包括查找文艺书库的书籍和专业书库的书籍；配套光盘用例是借书和还书用例的增量扩展用例。

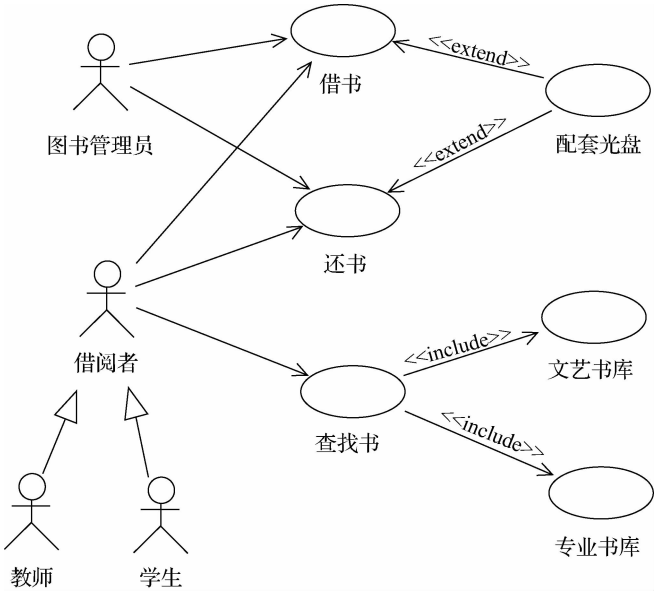


图 2-4　图书馆借还系统用例图

从图 2-4 中可以看出，用例图在对系统行为组织和建模方面是相当重要的。在绘制用例图时，系统分析人员首先要站在系统之外观察和分析系统，识别出系统中的所有角色；接着从分析系统的参与者开始，考虑每个参与者是如何使用系统的，识别和找出系统所提供的功能，即用例都有哪些，这是识别角色和用例的简洁方法。

2.2.2　类图和对象图

人类在面对和解决一个复杂问题时，普遍采用的一个策略就是分类和各个击破，这样可以有效地简化和处理复杂问题。类图技术是面向对象方法的核心，在面向对象建模技术中，也可以使用分类的方法将客观世界的实体映射为对象，并归纳成为一个个的类。类、对象以及它们之间的关联是面向对象技术中最基本的元素，对于一个将要描述的问题域系统来说，其中的类模型和对象模型揭示了问题域系统的结构。

在 UML 中，类和对象的概念前面已经介绍过了，类模型和对象模型则分别由类图和对

象图表示，其中，类图(class diagram)是用来描述类及类与类之间静态关系的一种静态模型。类图是构建其他图形的基础，没有类图就没有时序图、状态图和协作图等，也就无法表示系统其他方面的特性，所以建立类模型时，应尽量与应用领域的概念保持一致，以使模型更符合客观事实，易修改、易理解和易交流。

1. 类的表示

类用图 2-5 所示的符号表示。类图中包括 3 个组成部分，分别是类名、属性和操作。其中，属性和操作均有公有的(public)、私有的(private)和保护的(protected)三种类型，类型的可见性可分别用＋、－、＃来表示。

图 2-5　类的表示

2. 类之间的关系

类之间的关系主要有关联、泛化(继承)、依赖和复合聚合、实现(细化)等，其中关联又可以分为普通关联、角色关联、类关联、限制关联和多重关联等。

【例 2-2】　图 2-6 所示为某学生成绩管理系统中学生类和课程类通过成绩关联类关联的关系，其中学生类和课程类之间是非直接依赖关系。

关联类是用来表示一个类与其他类之间关联关系的类。当一个关联具有自己的属性并需要存储它们时，就需要用关联类建模。关联类用虚线连接在两个类之间的连线上。

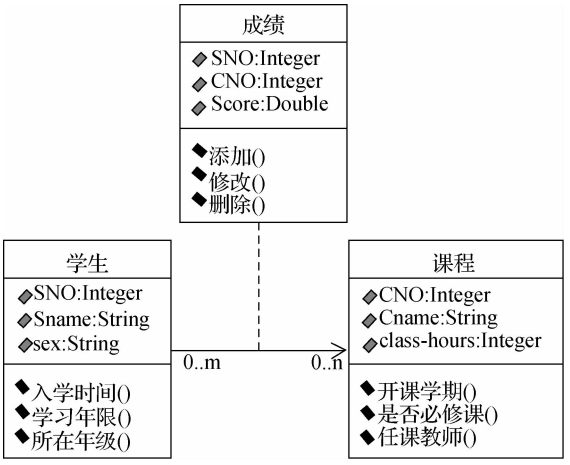


图 2-6　学生成绩管理系统中的关联类

3. 类图

类图从系统构成角度来描述系统，它展示了一组类、接口及它们之间的关系。建立一个

系统的类图，需要从系统的各个用例中识别类、类的属性和操作，并经过分析、抽象后才能得到。一个系统可以有多个类图，单个类图仅表达了系统的一个方面，同时要在高层给出类的主要职责，在低层给出类的属性和操作，类图没有时间概念，可以说是概念数据模型（如 E-R 图）的一种延伸。类图可划分为以下 3 个抽象层次，正确理解类的层次的概念有助于理解和画好类图。

（1）概念层。在需求分析阶段使用概念层类图描述应用领域中的概念。

（2）说明层。在设计阶段使用说明层类图描述软件中类和类的接口部分，而不是描述软件的实现部分。

（3）实现层。在实现阶段使用实现层类图描述软件系统中类的实现。

【例 2-3】 类图应用。图 2-7 所示为某学生成绩管理系统中学生选课实现层的类关系图。其中选课情况类也可称为成绩类。类图中的 0..*、0..m 或 0..n 表示 0 到多个对象，0..1 表示 0 到 1 个对象，m..n 表示 m 到 n 个对象。

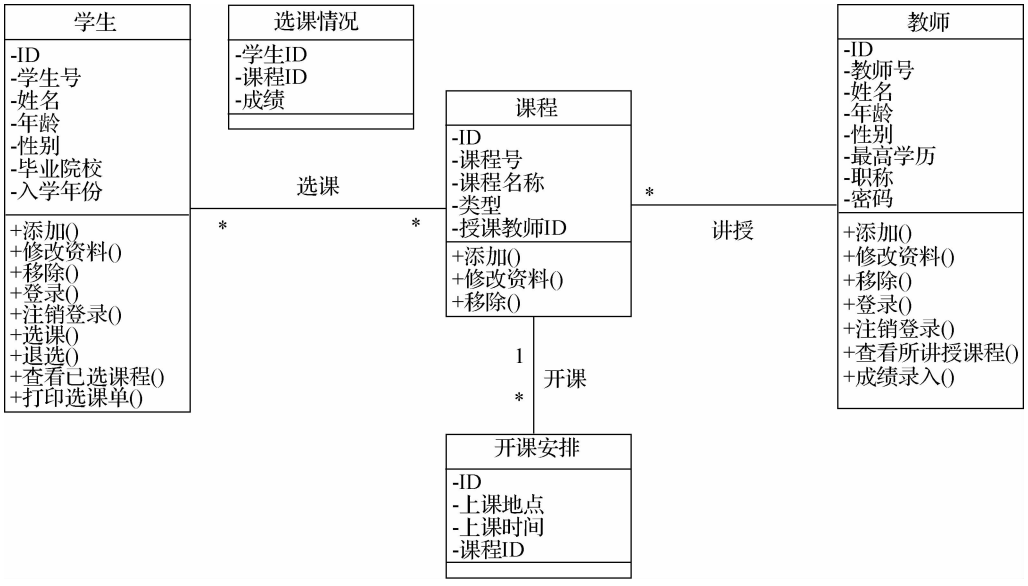


图 2-7　学生成绩管理系统中学生选课类

4. 对象图

对象图（object diagram）展示了一组对象以及它们之间的关系。对象图是类图的实例，对象之间的连接是类之间关联的实例。类图和对象图的不同在于：对象图显示的是类的多个对象实例，而不是实际的类。一个对象图是类图的一个实例，由于对象存在生存周期，因此对象图只能在系统某一时间段内存在，对象图显示某一时刻对象和对象之间的关系。

在 UML 中，对象图和类图一样也采用矩形图示，但是对象名称下方有下划线（类名称下方没有下划线），通常对象名采用“对象名：类名”或仅写“对象名”的格式来表示，如图 2-8 所示。但对象名称也经常被省略，所以常见带有冒号的类名，即省略对象名的对象。

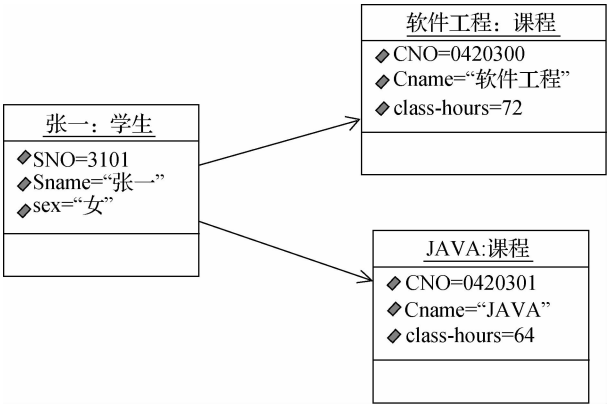


图 2-8　对象图

对象图通常在分析和设计阶段创建，常用于捕获实例和连接、交互的静态部分等。与类图相同，通常也由分析人员、设计人员和代码实现人员进行开发和实现。

另外，使用对象图还可以说明类图中的类或组件、数据结构等的实例静态快照。对象也与协作图相联系，协作图显示处于语境中的对象（类元角色）之间的协作关系。

2.2.3　包图

包图（package diagram）是具有某些共性的类组合在一起构成的图，它由包与包之间的联系构成，是维护和控制系统总体结构的重要建模工具。

在分析和设计一个大型系统时，往往会出现上百个甚至更多的类，这时理解和修改该系统将变得非常困难。如何有效地管理这些类是分析人员首要解决的重要问题，其基本做法是将许多类组合成一个更高层次的单位，形成高内聚、低耦合的类的集合，这种对元素进行分组的通用机制在 UML 中称为包。广义地讲，包的元素包括类、接口、组件、节点、协作、用例等，还可以包括内嵌的其他子包，引入包可以降低系统的复杂性。

构成包的模型元素称为包的内容，通常可以把一个系统划分为不同的主题层或子系统，属于同一个主题层的元素放在一个包中，主题层之间的依赖关系表现为包的依赖关系。

在图形上，包是带有标签的文件夹，与其他建模元素一样，每个包都必须有一个区别于其他包的名字。模型包的名字是一个字符串，它可分为简单名（simple name）和路径名（path name）两种。简单名是指包仅含一个简单的名称，路径名是指以包所位于的外围包的名字作为前缀的包名。

【例 2-4】　图 2-9 所示为某学生信息管理系统的用户接口包图，可以划分为 5 个主题层：用户界面层（放置窗体和表单类）、学生信息管理层（放置系统业务对象类）、数据库层（放置数据库表类）、基础类库层（放置基础类库类）和出错处理层（放置出错信息处理类）。包与包之间的依赖关系用带箭头的虚线表示。

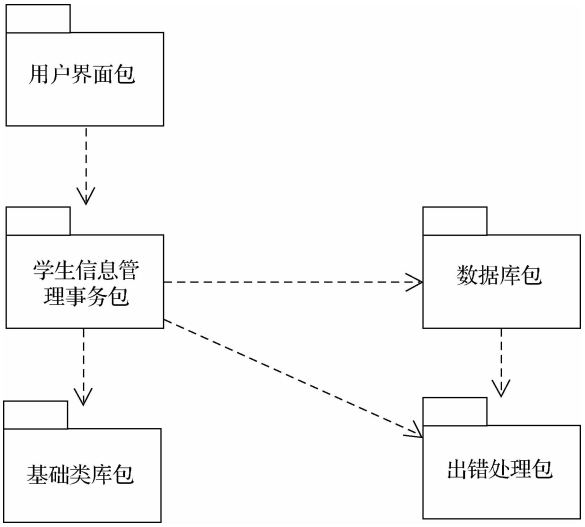


图 2-9　学生信息管理系统用户接口包

包能够引用来自其他包中的模型元素，包之间的访问权限通过输出（输出品）和导入（进口货）来设置，虚线箭头从源包指向目标包。包与包之间除了依赖关系外，还可建立泛化（继承）和精化（实现）关系。

包的概念对测试也特别有用，可以类为单位进行测试，但有时以包为单位进行系统单元测试更为方便、灵活，每个包应包含一个或多个测试类来测试包的行为。

2.2.4　组件图

组件图（component diagram）也称为构件图，由最终组成系统的各种构件（也称为组件）组成。组件图是用来建立系统构件组织实际结构和它们之间依赖关系的模型，组件之间的依赖关系用带箭头的虚线表示。

提示：组件图和配置图一般放在动态建模之后介绍，这里是为了章节系统的完整性，暂时把它们归结为静态建模。

组件（component）是具有一致的、定义良好的界面的可执行软件模块。例如，进程内组件（. dll）、进程外组件（. exe）、C++ 源代码中的头文件（. h）和实现文件（. cpp）、ActiveX、Applet 和可执行程序、物理数据库等。

组件图中通常包含 3 个元素：组件、接口（interface）和依赖关系（dependency）。组件可以通过组件中的类实现接口。组件之间的依赖关系模型如图 2-10 所示，表示客户端组件依赖于提供者组件。

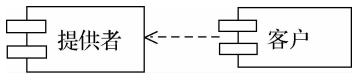


图 2-10　组件之间的依赖关系模型

组件图表示系统的静态实现视图，即用于对系统的实现视图建模，而类图和包图对软件的逻辑设计建模。

2.2.5 配置图

配置图(deployment diagram)也称为部署图,是描述系统中硬件和软件的物理配置情况的图形,显示系统运行阶段的物理硬件(节点)以及节点上的组件、进程和对象的配置情况。其中,组件表示运行阶段的代码单元,进程是一个正在执行的程序,配置情况展示系统运行时刻的系统体系结构。

在 UML 中,配置图中通常包含 3 个元素:节点(node)、组件和连接(关系)。配置图可以显示节点以及它们之间的必要连接,也可以显示这些连接的类型,还可以显示组件和组件之间的依赖关系,但是每个组件必须存在于某些节点上。

节点是系统运行时代表处理或计算资源的物理元素。节点通常拥有一些内存,并具有处理能力。在实际的建模过程中,可以把节点分为以下两种类型。

(1)处理器(processor)。处理器是指能够执行软件组件、具有计算能力的节点,如工作站和各种服务器等,如图 2-11(a)所示。

(2)设备(device)。设备是没有计算能力的节点,通常情况下都是通过其接口为外部提供某种服务。例如,打印机、扫描仪和各种终端等都是设备,如图 2-11(b)所示。

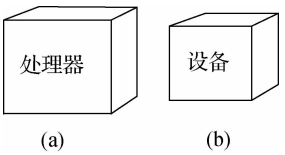


图 2-11 节点图

在 UML 中,节点用一个立方体来表示,每一个节点都必须有一个区别于其他节点的名称。节点的名称是一个字符串,位于节点图标的内部,通常有 3 种,即简单名、路径名和细节名。

在实际应用中,配置图主要用在设计和实现两个阶段。在设计阶段,配置图主要用来描述硬件节点以及节点之间的连接。在实现阶段,已经产生软件组件,因此,可以把组件分配给对应的节点。

【例 2-5】 客户机/服务器系统配置如图 2-12 所示。

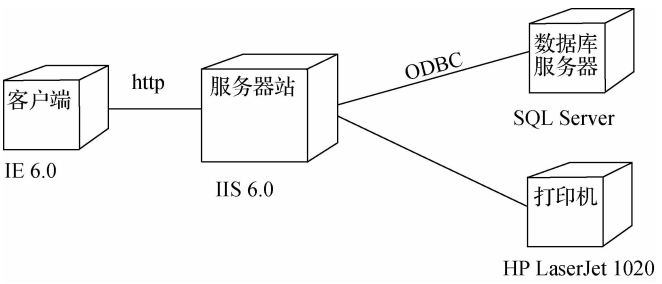


图 2-12 客户机/服务器系统配置

提示：对于配置图，并不是所有系统都要创建它，一个单机系统只需要建立包图和组件图即可，而配置图主要用于网络环境下运行的分布式系统、客户机/服务器系统和嵌入式系统等复杂系统的建模。

2.3 动 态 建 模

在建立好系统静态建模的基础上，接下来分析和设计系统的动态结构并创建相应的动态模型。动态模型描述了系统随时间变化的瞬间行为，在 UML 中，动态模型主要建立系统的交互图（包含时序图和协作图）和行为图（包含状态图和活动图）。

交互图展现了一种交互，它由一组对象及它们之间交互的信息组成，可用于描述一个用例的行为。时序图展示系统中按时间顺序排列的对象之间交互作用的关系。时序图可以类的形式和实例（对象）的形式存在，而协作图主要描述对象之间的交互作用及链接关系。时序图和协作图都是交互图，它们可以相互转换。

行为图主要在分析和设计阶段用于描述对象的行为。状态图适于描述单个对象状态的变化情况，活动图适于描述一个工作过程与多个对象之间的合作。如果希望查看单个对象跨用例的行为，就要使用状态图；如果希望查看多个对象跨用例、跨线程的行为，就要使用活动图。

2.3.1 时序图

时序图（sequence diagram）展现了一组对象和由这组对象收发的消息，用于描述用例中对象之间动态的交互关系，着重体现对象间消息传递的时间顺序关系。时序图中对象的示意图如图 2-13 所示。

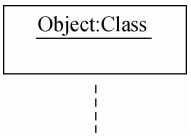


图 2-13　时序图中对象的示意图

其中，对象表述为虚垂线顶端的矩形小框，图中的对象名 Object 和类名 Class 可省写一个；虚垂线是对象的生命线，说明对象的生命；两个对象生命线之间的箭头表示消息，消息出现的次序自上而下，消息箭头可以回到同一条生命线，指明自调用，即对象发给自己的消息。在 UML 中，用符号“×”表示对象的撤销，即生命线的结束。

【例 2-6】　图 2-14 所示为学生毕业管理时序图的一个实例。

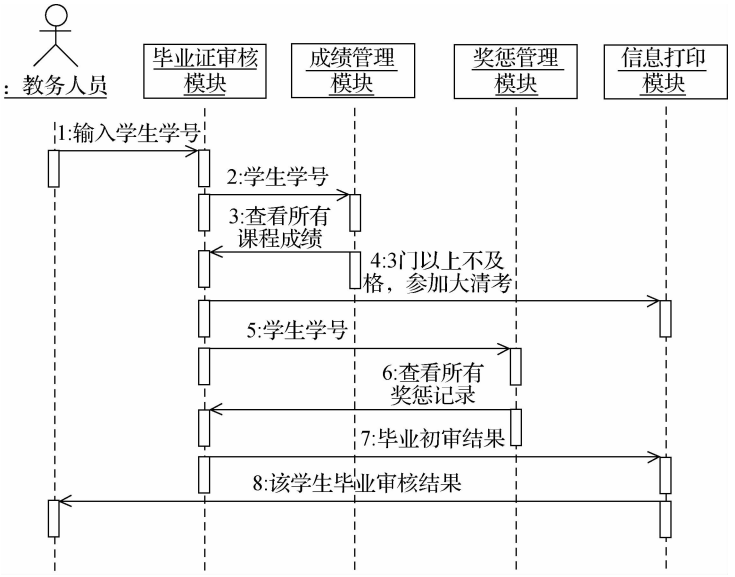


图 2-14　学生毕业管理时序图

2.3.2　协作图

协作图(collaboration diagram)又称为合作图，它展现了一组对象、这组对象之间的链接关系以及这组对象收发的消息。它强调收发消息的对象的组织结构，按组织结构对控制流建模。协作图中的协作不是参与者与系统之间的交互，而是系统内部某一个用例中各个对象之间信息传递的方式，消息上所附编号指明执行顺序。

协作图主要描述相互合作的对象之间的交互作用及链接关系，但时间顺序不明确，要从序列编号获得。虽然协作图和时序图都表示对象之间的交互作用，但时序图展示的是系统中按时间顺序排列的对象之间交互作用的关系，而没有明确表达对象之间的交互关系。时序图和协作图在 UML 中的语义是等价的，它们都是交互图，可以相互转换而不丢失任何信息，操作方法一般是按功能键 F5，然后协调布置好各个对象及它们之间的交互关系，如图 2-15 所示。

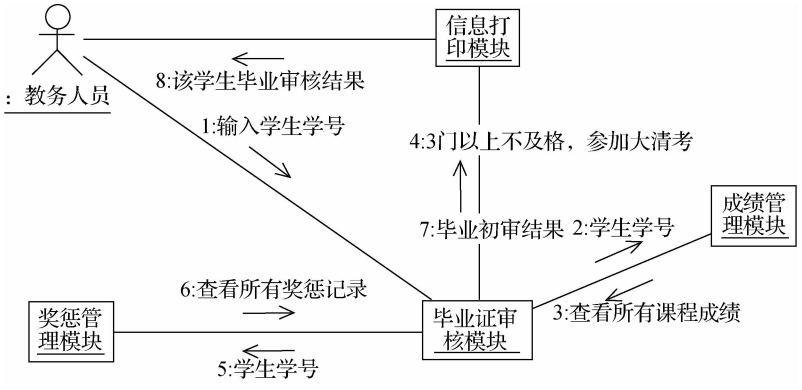


图 2-15　学生毕业管理协作图

2.3.3　状态图

状态图(statechart diagram)和活动图都属于行为图，主要用于分析和设计阶段描述对象的行为。状态图适于描述单个对象状态的变化情况，活动图适于描述一个工作过程以及多个对象之间的合作。

状态图展示了一个特定对象的所有可能状态以及由于各种事件的发生而引起的状态间的转移。状态图用于说明系统的动态视图，它对于接口、类或协作的行为建模尤为重要，可用它来描述用例实例的生存周期。

其中，状态(statechart)是对象生命周期中的一个条件或状况，它是对象执行了一系列活动的结果。当某个事件发生后，对象的状态发生变化，这里称改变状态的事情为“事件”。一个状态图包括一系列状态、事件及状态之间的转移。

在 UML 中，一个状态图可以有一个初始状态和多个最终状态。其中，初始状态用一个小的实心圆表示，最终状态用一个内部实心的两个同心圆表示，中间状态用圆角矩形表示，如图 2-16 所示。若一个状态可以进一步细化为多个子状态，则称其为复合状态。

提示：一个状态图一般只能有一个初始状态，但可以有多个中间状态和最终状态。

【例 2-7】 图 2-17 所示是一个移动用户手机状态变化的状态图。

图 2-16　状态的 3 种表示

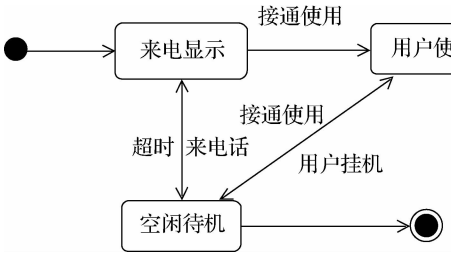


图 2-17　移动手机状态图

2.3.4　活动图

状态图着重描述对象状态的变化以及触发状态变化的事件，活动图则描述系统中各种活动的执行顺序。活动图的用途非常广泛，它不仅可以描述系统的活动、判定和分支等，还可以描述类的方法、用例、对象内部的工作过程，以及系统中多个对象的并发活动过程。

活动图(activity diagram)可以说是一种特殊的状态图，主要描述要进行的活动、执行这些活动的顺序以及工作流。它强调对象间的控制流程，对于系统的功能建模特别重要。

活动(activity)是活动图的核心概念，在 UML 中，活动用圆角矩形表示，活动图的起点用实心圆表示，终点用中心为实心的两个同心圆表示，活动图中还包括判定、水平同步控制条(可表示分叉或汇合)和垂直同步控制条(可表示分叉或汇合)，活动之间用带箭头的直线表示从一个活动到另一个活动的转移，同时可在线上标注活动转移的条件，如图 2-18 所示。图中还画出了泳道，泳道可以将一个活动图中的活动状态进行分组。

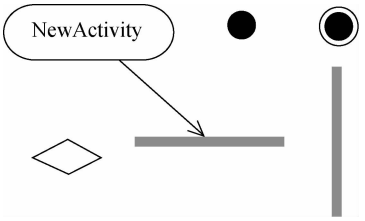


图 2-18　活动图的基本元素

【例 2-8】　图 2-19 所示是某门课程一个期末考场的活动图。

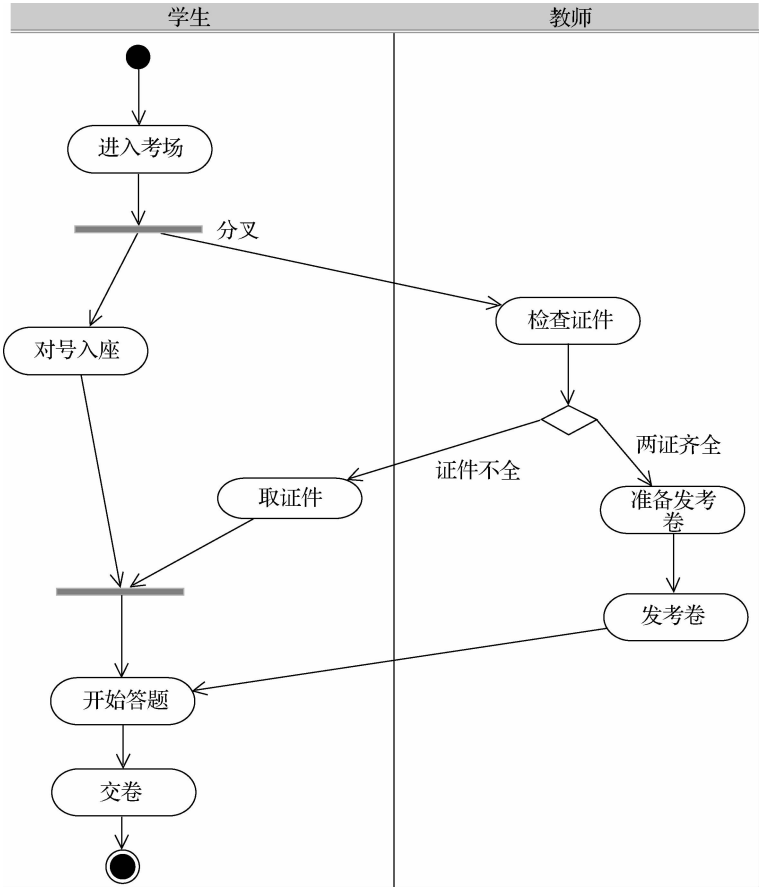


图 2-19　期末考场的活动图

在活动图中，除了可以使用泳道外，还可以使用对象和对象流。在 UML 中，对象是使用矩形符号来表示的，矩形内可以写上对象名或类名，对象和活动之间使用带箭头的虚线表示对象流。

在对一个系统的动态行为建模时，通常有两种使用活动图的方式：一是为系统的工作流建模，二是为系统中对象的操作建模。在 UML 中，可以把活动图作为流程图来使用，用于对系统的操作建模。值得一提的是，在实际进行系统分析时，有些活动很难被归类到某个对象或类中。

2.4 统一建模过程

UML 只是一种建模语言，而不是一种开发方法。统一软件开发过程（rational unified process，RUP）或统一建模过程是一个面向对象且基于网络的程序开发方法，它主要由 Rational Rose 和 UML 开发者 Ivar Jacobson 的对象方法和统一方法理论发展而来，可以为所有层面的程序开发者提供指导方针、模板以及事例支持。同时，RUP 又是文档化的软件工程产品，它吸收了多种开发模型的优点，具有很好的可操作性和实用性，越来越多的组织把它作为软件开发模型框架。

RUP 的软件开发生存周期是一个二维的软件开发模型。横轴通过时间来组织，是过程展开的生存周期特征，体现开发过程的动态结构，用来描述它的术语主要包括周期、阶段、迭代和里程碑；纵轴通过将内容组织为自然的逻辑活动来体现开发过程的静态结构，用来描述它的术语主要包括活动、产物、工作者和工作流。

RUP 是一种基于用例驱动的、以系统架构为中心的迭代与增量开发软件的过程。RUP 中的软件生命周期在时间上被分解为图 2-20 所示的 4 个顺序阶段，分别是初始（或启动）阶段（inception）、细化阶段（elaboration）、构造（或构建）阶段（construction）和移交阶段（transition）。其中每个阶段都可以进一步分解为迭代，一个迭代是一个开发循环，包括 5 个活动的核心工作流，即需求、分析、设计、实现和测试。它产生一个可执行的产品版本，这个版本的产品是最终产品的一个子集。每个阶段结束于一个主要的里程碑（major milestones），每个阶段本质上是两个里程碑之间的时间跨度。在每个阶段的结尾执行一次评估以确定这个阶段的目标是否已经满足要求。如果评估结果令人满意，就允许项目进入下一个阶段。

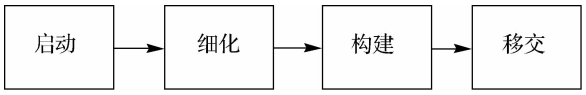


图 2-20　RUP 的统一开发过程

这是一个迭代渐进式的开发过程。在这个过程中，软件不是在工程结束时整个完成，而是一小部分接一小部分地进行开发。在建模阶段包括若干迭代，而每一个迭代都会根据整个工程的部分需求开发出一个经过测试的、集成的且符合质量要求的软件。迭代的结果可能提交给外部的早期用户，也可能在系统内部使用。在移交阶段后进行总体测试、性能测试和用户培训等。

1）初始阶段

初始（或启动）阶段的目标是为系统建立商业案例并确定项目的边界。为了达到此目的，必须识别所有与系统交互的外部实体，在较高层次上定义交互的特性。具体做法如下。

（1）识别系统主要角色并描述高层用例。

（2）划分系统模块为子系统，规划系统体系结构。

(3)进行项目的总体需求和可行性分析,制订出项目的开发计划。

注意:初始阶段结束于第一个重要的里程碑,即生命周期目标(lifecycle objective)里程碑,它用于评价项目基本的生存能力。

2)细化阶段

细化阶段的目标是:分析问题领域,建立健全的体系结构基础;编制项目计划,淘汰项目中具有最高风险的元素。例如,启动该项目后,要进行以下分析。

(1)分析问题领域,识别并确定系统的域角色和用例。

(2)分析用例的处理流程和对象的状态,形成系统的概念模型和体系结构。

(3)分析项目的可能风险,并制订应对策略和构建阶段的计划。

注意:细化阶段结束于第二个重要的里程碑,即生命周期结构(lifecycle architecture)里程碑,它为系统的结构建立了管理基准并使项目小组能够在构建阶段对其进行衡量。此刻,要检验详细的系统目标和范围、选择结构以及确定主要风险的解决方案。

3)构建阶段

在构建阶段,所有剩余的组件和应用程序功能都被开发并集成为产品,所有功能都将被详细测试。具体做法如下。

(1)利用各种 UML 框图来描述系统需求。

(2)根据需求建立系统的静态模型,以构造系统的结构。例如,进行系统用例和类的分析和设计。

(3)描述系统行为,并注意保证代码和模型的一致性。

注意:构建阶段结束于第三个重要的里程碑,即初始功能(initial operational)里程碑,它决定了产品是否可以在测试环境中进行部署。此刻,要确定软件、环境、用户是否可以开始系统的运作。

4)移交阶段

移交阶段即交付阶段,其重点是确保软件对最终用户是可用的。交付阶段可以跨越几次迭代,包括为发布准备的产品测试以及基于用户反馈的少量调整等。具体做法如下。

(1)完成软件产品的开发,并配合用户进行验收测试。

(2)编制用户使用手册,制订培训计划。

注意:移交阶段的终点是第四个重要的里程碑,即产品发布(product release)里程碑。此时,要确定目标是否实现以及是否应该开始另一个开发周期。

核心工作流的 5 个活动要完成的工作如表 2-2 所示。

表 2-2　核心工作流的 5 个活动要完成的工作

活　动	任　务
需求	分析系统要实现的功能,界定系统范围,解决系统“What to do?”的问题
分析	进行系统需求规格说明,主要分析系统或子系统的用例图、类图和包图
设计	用系统架构实现需求,即解决系统“How to do?”的问题
实现	用具体的程序语言实现系统功能
测试	制订测试计划,设计测试用例,实现测试以及执行系统集成测试

2.5 Rose 建模工具的使用

2.5.1 Rose 建模工具介绍

解决面向对象问题的核心是建模，即建立系统的 Rose(rational object-oriented software engineering)模型。软件系统内部的高内聚、低耦合程度以及维护成本是软件设计所关注的问题。Rational Rose 是基于 UML 而产生的，是软件开发过程中不可或缺的一个建模工具，它支持多种语言，如 Ada、CORBA、Visual Basic、Java 等。

Rose 是由美国 Rational 公司推出的面向对象分析与设计的优秀建模工具，利用它可以建立用 UML 描述的软件系统的各种模型，而且可以自动生成和维护，如 C++、Java、VB 和 Oracle 等语言和系统的代码。Rose 包括统一建模语言(UML)、面向对象的软件工程(OOSE)及对象模型技术(OMT)。其中，UML 是由 Rational 公司 3 位世界级面向对象技术专家在早期的面向对象研究和设计方法的基础上进一步扩展而得来的，为可视化建模软件奠定了坚实的理论基础。

1. 安装 Rational Rose

目前，Rational Rose 的最新版本是 2007，下面就来学习 Rational Rose 2007 的安装和使用方法。

提示：如果读者现在使用的是 Rational Rose 2003，那也没关系，两者的基本操作都是一样的。

安装 Rational Rose 需要 Windows 2000 或 Windows XP 及以上版本的 Windows 操作系统。其安装步骤如下。

(1)首先打开配套下载资源中的 Rose 压缩包，再双击执行 IBM Rational Rose 2007 的 Setup. exe 程序，弹出图 2-21 所示的对话框，安装过程开始。单击图 2-21 中的 Install IBM Rational Rose Enterprise Edition 选项，在弹出的对话框中单击“下一步”按钮，进入安装程序欢迎界面。

图 2-21　选择安装产品

(2)依照安装程序向导的提示,直接单击“下一步”按钮,弹出 Deployment Method 对话框,在这里可以选择安装模式。选中 Desktop installation from CD image 单选按钮,如图 2-22 所示,然后单击“下一步”按钮。

图 2-22　选择安装模式

(3)依照安装程序向导的提示,直接单击两次 Next 按钮,进入软件许可证协议界面,单击“接受”按钮,弹出 Destination Folder 对话框,如图 2-23 所示。在其中设置 IBM Rational Rose 2007 的安装路径,默认路径是 C:\Program Files\Rational\,可以单击 Change 按钮来改变其安装路径。

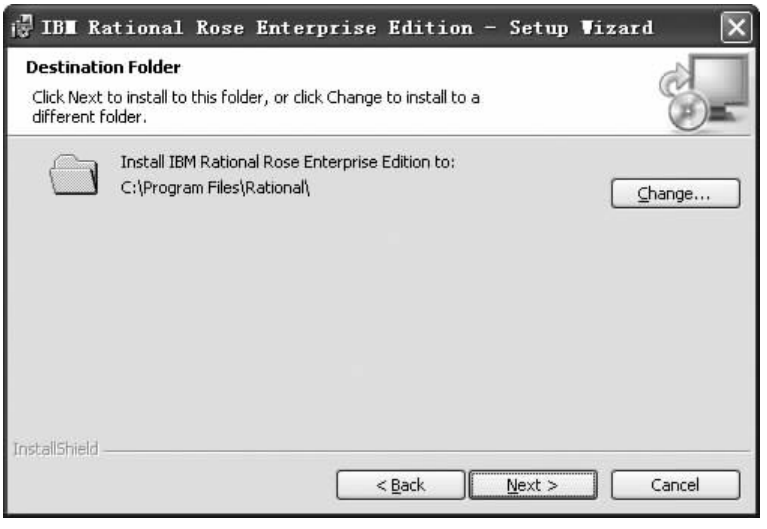


图 2-23　选择安装路径

(4)单击 Next 按钮,弹出 Custom Setup 对话框,在这里可以自定义安装选项,如图 2-24 所示。用户可以根据需要进行选择。

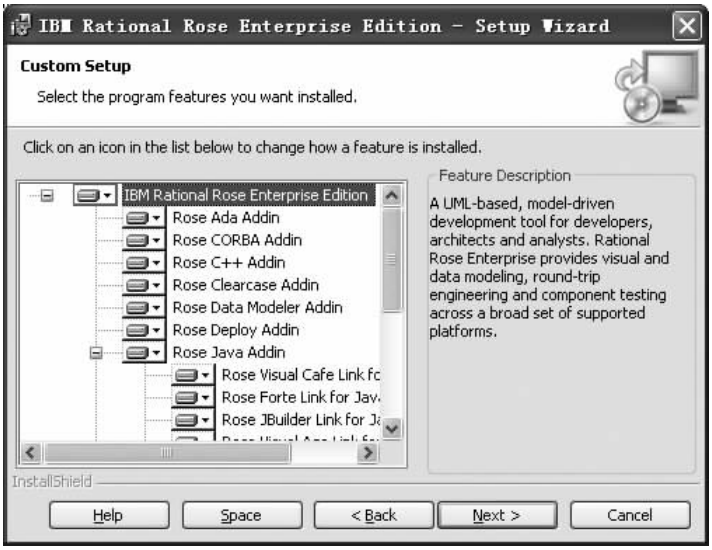


图 2-24　自定义安装选项界面

(5)继续单击 Next 按钮,进入开始安装界面,单击 Install 按钮,程序开始复制文件到安装目录中。

(6)稍等一会儿,当系统安装完成后,会出现一个完成界面,单击 Finish 按钮则会完成系统的全部安装,并自动退出安装程序。

(7)若 Rose 不能正常启动,则可执行“开始”→“程序”→IBM Rational→IBM Rational License Key Administrator 命令,进入 IBM Rational License 配置向导界面,如图 2-25 所示。选中 Import a Rational License File 单选按钮,然后单击“下一步”按钮。

图 2-25　配置向导界面

(8)在弹出的图 2-26 所示的 Import a License File 对话框中单击 Browse 按钮,将文件指向配套下载资源中 Rational Rose 2007-key 压缩文件中的许可证文件 license. upd,并单击 Import

按钮两次，若显示 File imported successfully，则完成配置 License Key 的全部过程。

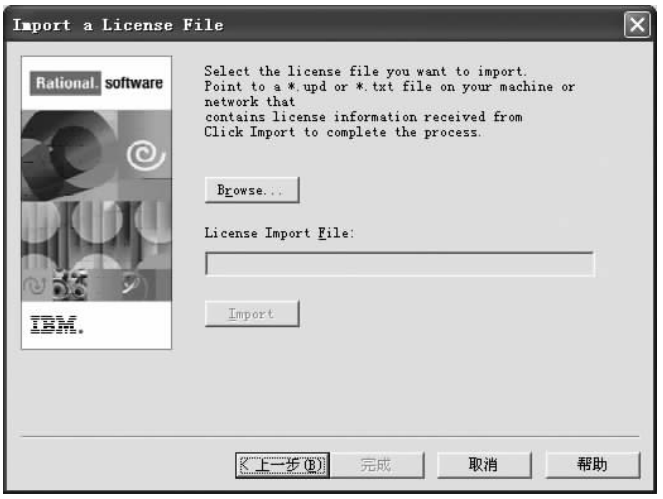


图 2-26　Import a License File 对话框

2. Rational Rose 的使用

完成 Rational Rose 安装后，就可以用 Rose 建立软件模型了。启动 Rational Rose 的方法是：执行“开始”→“程序”→IBM Rational→IBM Rational Rose Enterprise Edition 命令，弹出 IBM Rational Rose 2007 引导界面，之后弹出“新建模型”对话框，在其中可以设置本次启动的初始动作，包括 New(新建模型)、Existing(打开现有模型)和 Recent(最近打开模型) 3 个选项卡。

其中，在 New 选项卡中可以选择新建模型时采用的模板。由于暂时不需要任何模板，只需要新建一个空白模板，因此直接单击 Cancel(取消)按钮，进入 Rational Rose 2007 的主界面，如图 2-27 所示。

图 2-27　Rose 主界面

Rose 创建的模型文件扩展名为. mdl，其模型创建的一般步骤如下。

1)新建模型

执行 File→New 命令,或者直接单击标准工具栏中的 Create New Model or File 按钮,弹出 Create New Model 对话框,在其中选择要用到的框架,单击 OK 按钮。

2)保存模型

执行 File→Save 命令,或者直接单击标准工具栏中的 Save Model,File,or Script 按钮。

3)发布模型

执行 Tools→Web Publisher 命令,弹出图 2-28 所示的对话框,在其中选择要发布到 Web 页面上的内容和 HTML 页面要保存的位置,单击 Publish 按钮,Rose 模型就发布到 Web 页面上了。

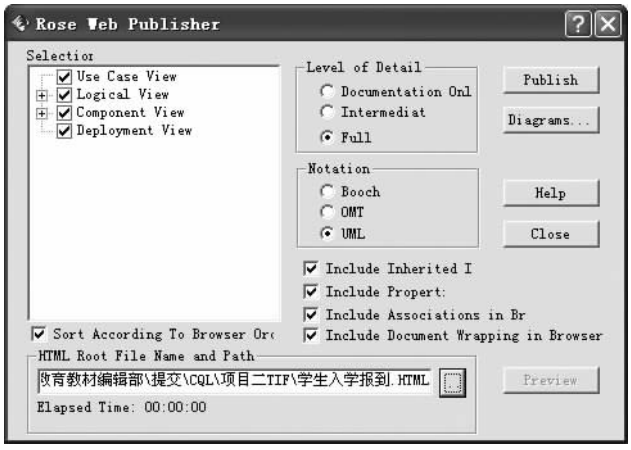


图 2-28　发布模型和保存 html 文件对话框

打开所保存的 html 文件,就可以看到 Rose 模型,如图 2-29 所示。

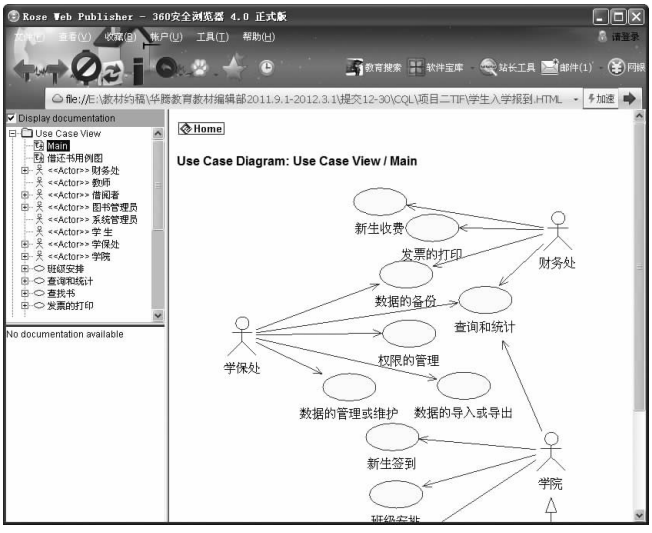


图 2-29　打开所发布的“学生入学报到. html”模型文件

关于 Rose 全局选项,如进行字体和颜色的设置等,可以通过执行 Tools→Options 命令,然后在弹出的对话框中进行设置。但注意应用时要遵循先设置,后生效的原则。

提示：建立模型前一定要先设置好对象的字体和颜色，否则建模后再设置字体和颜色是不会有效的。

有关 Rose 更多的操作应用知识请参照其他书籍，在此不再赘述。

2.5.2 结合项目分析——建模应用实例

下面以“新生入学报到系统”实例中的几个很常用的静态建模图（用例图、类图、包图）和动态建模图（时序图、协作图、状态图和活动图）为例对面向对象的建模语言 UML 及建模工具 Rose 的综合应用进行综合介绍，以使初学者初步学会建模的实际应用。

【例 2-9】 请为一个“新生入学报到系统”创建静态建模图和动态建模图。

1. 创建用例图

用例图是显示系统中角色和用例关系的图，画好用例图是由软件需求到最终实现的第一步。首先应寻找系统参与者，即角色，然后根据角色对系统主要功能的需求描述确定用例。“新生入学报到系统”的用例图如图 2-30 所示，图中形象地表达了用例图各元素之间的关系，带箭头的实线表示关联关系，带空心箭头的实线表示继承关系，即系统管理员除了拥有管理学院计算机系统的权限外，还管理着其他人使用系统的权限。

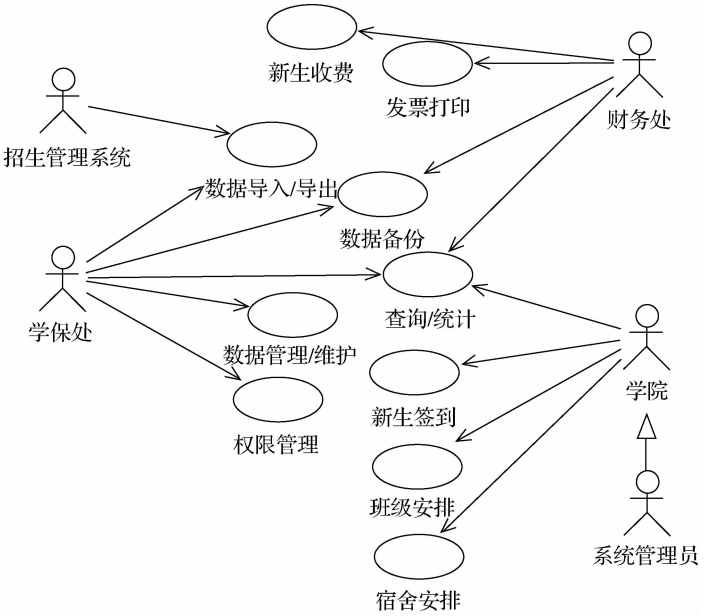


图 2-30 “新生入学报到系统”用例图

2. 创建类图

类图是面向对象方法的核心，它从系统构成角度来描述系统，展示了一组类、接口及它们之间的关系。建立一个系统的类图需要从系统的各个用例中识别出类、类的属性和操作，并经过分析、抽象后才能得到。

通过分析“新生入学报到系统”中的类及其关系绘制该系统的类图，如图 2-31 所示。

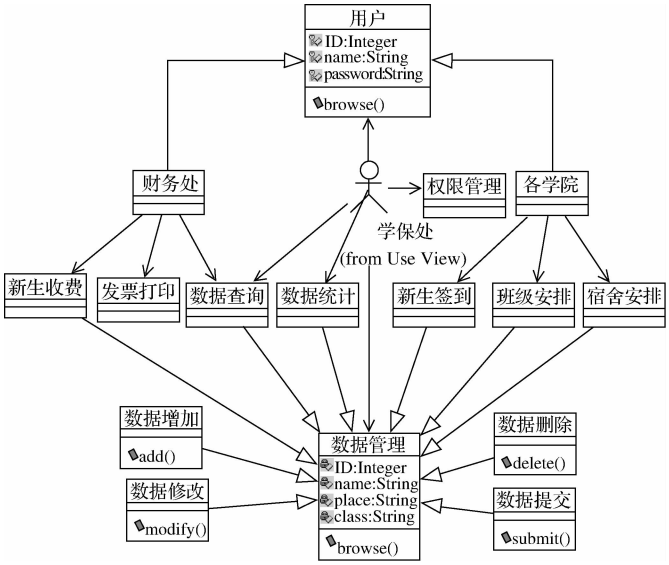


图 2-31　“新生入学报到系统”类图

3. 创建包图

包图是由一些具有共性的类组合在一起构成的图。读者可以参考图 2-9 所示的用户接口包图。

4. 创建时序图和协作图

动态模型描述了系统随时间变化的瞬间行为，可对系统中主要的用例和对象绘制动态建模图。其中，时序图主要表示以时间顺序安排对象之间的交互，时序图中有两个轴，水平轴表示不同的对象，垂直轴表示时间。对象之间的通信通过消息表示，当收到消息时，接受对象即被激活。下面给出新生入学报到系统“新生签到”用例的信息管理时序图和协作图，分别如图 2-32 和图 2-33 所示。

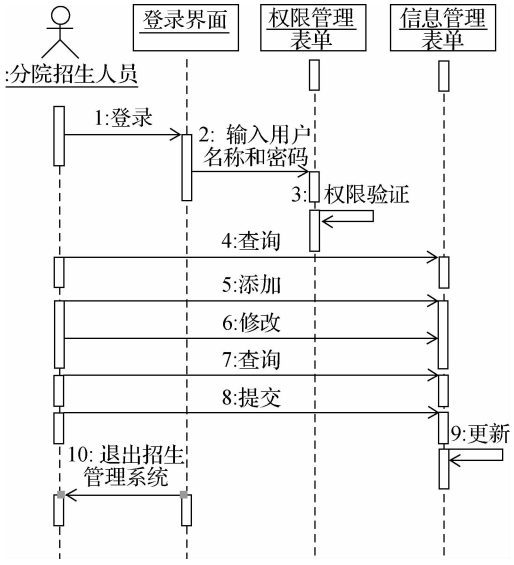


图 2-32　“新生签到”用例时序图

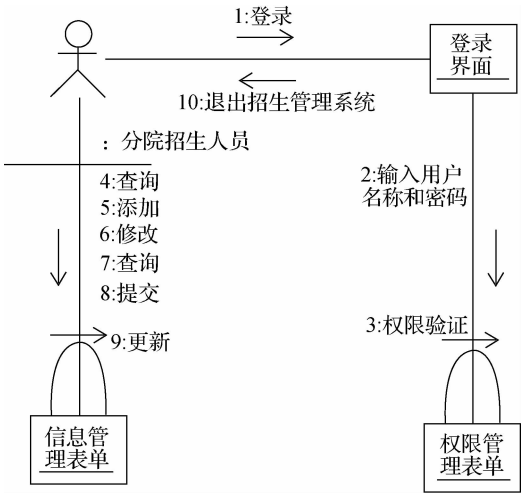


图 2-33　“新生签到”用例协作图

5. 创建状态图和活动图

状态图展示了一个特定对象的所有可能状态以及由于外界事件触发而引起的状态迁移，活动图适于描述一个工作过程以及多个对象之间的合作。

图 2-34 所示为描述新生入学报到时各分院管理人员进行新生录取的状态图，图 2-35 所示为班主任管理学生交费、住宿和入学的活动图。

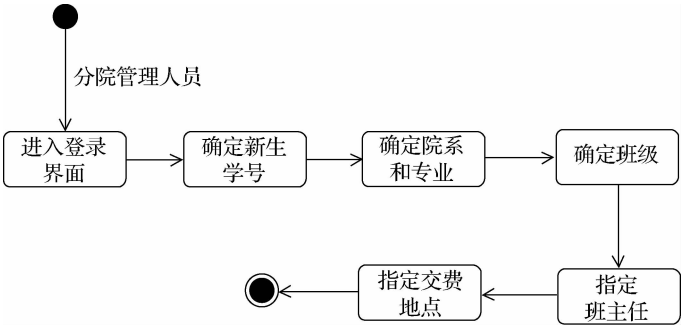


图 2-34　新生入学报到状态图

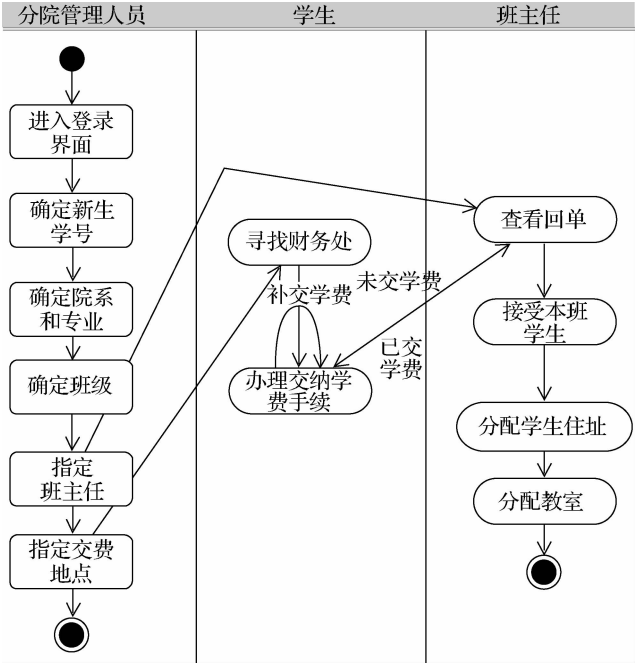


图 2-35　新生入学报到活动图

2.6 实训：Rational Rose 2007 操作入门

学完面向对象建模语言 UML 及建模工具 Rose 以后，下面通过上机训练各种建模图形的创建和操作方法。由于篇幅所限，这里仅给出上机的操作步骤，希望读者在实践中学会建模的技能和要领。

1. 实训的目的和任务

学会利用 Rational Rose 进行面向对象的系统分析、设计及正确建模的方法，并学会正确安装 Rose 及使用 Rose 来完成系统建模的操作。

2. 实训内容与步骤

1）正确安装 Rational Rose 2007

要想真正学会 Rational Rose 2007 的基本操作，正确安装 Rose 是第一步，参看本项目 2.5 节有关的操作步骤，进行安装 Rational Rose 2007 的操作。

2）正确使用 Rose 完成建模操作

（1）用例图建模。分析学生入学报到管理系统（或学生信息管理系统）的需求，确定系统中的参与者和主要用例，画出系统的角色用例视图，并试着描述不同角色的用例功能。

（2）对类图、对象图和包图建模。分析学生入学报到管理系统（或学生信息管理系统），利用 Rational Rose 完成其类图、对象图和包图的建模。

（3）动态模型建模（可放在后续项目中进行）。分析学生入学报到管理系统（或学生信息管理系统），利用 Rational Rose 完成其 4 种动态模型建模。

（4）Rose 其他模型建模（可放在后续项目中进行）。分析学生入学报到管理系统（或学生信息管理系统），利用 Rational Rose 完成组件图和配置图的建模。

2.7 习　　题

一、选择题

1. 面向对象程序设计语言不同于其他语言的最主要特点是（　　）。

A. 模块　　B. 抽象性　　C. 继承性　　D. 共享性

2. 面向对象分析阶段建立的三个模型中，核心是（　　）模型。

A. 功能　　B. 动态　　C. 对象　　D. 分析

3. 对象模型的描述工具是（　　）。

A. 状态图　　B. 数据流图　　C. 对象图　　D. 结构图

4. 软件部件的内部实现与外部可访问性分离是指软件的（　　）。

A. 封装性　　B. 抽象性　　C. 继承性　　D. 共享性

5. 类图反映了系统中对象之间的抽象关系，不包括（　　）。

A. 关联　　B. 聚合　　C. 泛化　　D. 内聚

6. (　　)模型表示了对象之间的相互行为。

A. 功能　　B. 动态　　C. 对象　　D. 分析

7. 描述类中某个对象的行为,反映了状态与事件关系的是(　　)。

A. 状态图　　B. 数据流图　　C. 对象图　　D. 结构图

8. 在面向对象方法中,信息隐蔽是通过对象的(　　)来实现的。

A. 封装性　　B. 分类性　　C. 继承性　　D. 共享性

9. 有关类和对象的描述中,错误的是(　　)。

A. 一个类只能有一个对象

B. 对象是类的具体实例

C. 类是某一类对象的抽象

D. 类和对象的关系是一种数据类型和变量的关系

二、简答题

1. 简述 UML 的主要内容和特点。

2. 类之间的关系有哪些?

3. 什么是对象、属性和服务关系? 分别举实例说明。

4. 什么是状态、事件和行为? 分别举例说明。

5. 用类图描述下列对象的属性、服务及相互关系。

(1)学生:学生的属性有学号、姓名、专业、班级、性别。

班级:班级的属性有系、专业、入学年份、学制。

课程:课程的属性有课程名、任课教师。

教师:教师的属性有姓名、课程、任课班级。

每个学生属于某个班级,每个班级有若干学生。每个学生规定学习若干门课程,每门课程有若干任课教师。每个学生在选修某门课程时,可选某一指定的任课教师。每位教师担任若干门课程的教学工作。

(2)某高职院校由多个院系组成,每个院系管理着多个教研室和多个学生班级,每个教研室由若干名教师组成,每个教师可承担多个教学任务。每个学生班级由许多学生组成,学生分全日制教育高职学生和成人教育高职学生,每名学生可以选修多门不同的课程,一门课程对应一到多个教学任务。

(3)持有驾驶执照的驾驶员可驾驶汽车。每个驾驶员可驾驶多辆汽车,每辆汽车的驾驶员不固定。

6. 分析一个高校的学生成绩管理系统,分别建立该系统的用例图、类图、对象图、时序图和状态图。

项目 3

软件项目的系统分析

软件项目系统分析阶段的任务，主要是对项目进行筹备、规划和分工，和软件企业或公司内部一样，项目规划和分工一定要明确，具体包括确定每个项目开发阶段的准备工作、时间安排以及各个项目开发阶段主要由谁来负责、开发团队成员如何相互协作等。项目 3 主要由两个阶段构成：一是项目的规划及可行性分析；二是项目的需求分析。

软件项目的系统分析分为几个阶段？怎样进行软件项目的可行性分析？可行性分析文档有哪些主要方面的要求？软件项目开发计划有哪些内容？需求分析是怎样一个过程？怎样描绘系统流程图和数据流图，两者之间的区别是什么？怎样描绘软件项目的 E-R 模型？数据字典是什么？如何使用 Visio 2007 绘制 E-R 图及数据流图等模型？这一系列问题都将在下面予以详细介绍。项目 3 中涉及的角色主要有系统分析员兼项目经理(进行系统分析、负责协调分工、监督系统开发工作的完成)、客户、程序员及测试人员等。

学习目标

- **理解并掌握：**可行性分析的任务和步骤，需求分析的步骤、方法，系统流程图的符号，数据流图中的符号，设计数据流图的步骤，数据字典的定义及其用途。
- **使用：**Visio 2007 绘制 E-R 图及数据流图等模型。
- **了解：**数据字典，关注可行性研究报告、项目开发计划书和需求规格说明书的撰写。

3.1 软件项目可行性分析

在计算机软件项目开发过程中，只要资源和时间不被限制，所有项目都是可行的。然而，由于资源缺乏和交付时间的限制，使得基于计算机系统的开发变得越来越困难。因此，尽早对软件项目的可行性进行细致而谨慎的评估是十分必要的。如果在定义阶段及早发现将来可能在开发过程中遇到的问题，并及早做出决定，就可以避免大量人力、财力和时间的浪费。本节将简要介绍有关可行性分析的任务、步骤，以及在撰写可行性分析报告时的要求，同时对可行性分析文档和软件项目开发计划等方面进行概要介绍，通过本节的学习，要求深刻理解可行性分析的重要性。

提示：一个软件项目的开发主要解决以下 3 个方面的大问题。

- Why to do? ——可行性研究。
- What to do? ——需求分析。
- How to do? ——系统设计。

3.1.1 问题的定义

这里所讲的问题是指用户的基本要求，就是确切地定义用户要求解决的问题，即确定问题的性质、工程的目标和规模。

怎样定义问题？问题定义的来源是用户，是提出问题、请求解决的人。

如果问题是以书面形式提出的，那么系统分析员应该认真阅读和分析书面材料；如果问题是以口头形式提出的，那么系统分析员应该认真倾听并仔细记录要点，在适当的时候认真地请用户解释。系统分析员还应该通过对用户的访问调查进一步弄清楚用户为什么提出这样的问题，问题的背景是什么，用户的目标是什么等。

问题定义的目的是要在短时间内对用户的要求有一个比较准确的估计，对要实现的系统规模做到心中有数。但仅有这些还不够，还要弄清用户不打算干什么，在这个系统中哪些内容不用实现等问题。工作的宗旨是弄清要做什么并划清要实现系统的范围边界。

在完成问题定义的过程中，用户一开始可能会给出很多表格，因为他们可能认为只要把表格讲清楚，系统分析员就会将这个系统全部弄清楚了。还有一些人可能会展示一些企业十分详尽的管理示图，如物资流管理图、生产管理图、计划财务管理图等。因为他们可能认为只要系统分析员把这些图看懂了，就会对他们要建立的系统清楚了。

但是，在问题定义阶段千万不要陷入这些表格和图纸中。因为不管是表格还是图纸，其中都包含了大量只有用户才能懂的术语。当然，这并不是说在问题定义阶段这些图纸和表格没有一点作用。对一些关键性的语汇可以请用户讲清楚，这样有利于问题定义的准确性。总之，在问题定义阶段，系统分析员应尽可能站在较高的角度去抽象、概括所要做的事情。

系统分析员对问题有了明确认识之后，应该把自己的认识写成书面报告，并提交给用户和使用部门的负责人审查，以检验系统分析员对所要解决问题的理解是否正确。因为系统分析员对问题的理解会确定今后开发工作的方向。系统分析员对问题理解正确，是确保今后系统开发成功的关键。反之，系统分析员对问题理解不正确，会导致最终开发出来的系统

不能解决实际要求解决的问题。如果一个系统不能解决要求它解决的问题，那么这个系统就一点价值也没有，从而浪费了开发它所用的时间和资源。所以，及时审查问题的定义是非常重要的。理想的做法是系统分析员、用户和使用部门的负责人一起阅读并讨论这份报告，明确含糊不清的地方，改正不正确的地方，通过修改得到一份大家一致认同的文档。

对问题定义的书面报告应该尽可能清楚、简洁，最好写在一页纸内。这份问题定义报告通常应包括工程项目的名称，对问题的概括定义，项目的目标、规模和对可行性研究的具体建议（需要的时间和成本）等。问题定义报告的格式如表 3-1 所示。

表 3-1　问题定义报告

用户单位	某　　校
用户负责人	×××
（系统）分析员单位	×××学院（或×××计算机软件公司）
（系统）分析员	×××
工程项目的名称	高校学生信息管理系统
问题（概括定义）	高校招生就业处和学生管理各部门每学年的招生就业工作及其他各项管理工作繁忙，很多精力都花在了高校学生信息管理的具体事务中（这里主要以学籍管理为例）
项目的目标	研究高校学生信息管理系统开发的可能性
项目的规模	本项目的开发成本 5 万元
对可行性研究的具体建议	建议进行大约一周的可行性研究，研究经费不超过 1 000 元

一旦系统分析员、用户及使用部门的负责人对所要解决的问题取得了完全一致的看法，且在报告书上签了字，问题定义阶段的工作就宣告完成，可行性研究即可开始。

3.1.2　可行性分析的任务和步骤

在问题定义之后，这个问题是否有简单、明显的解决办法呢？因为许多问题事实上不能在预定的系统规模内解决。如果所定义的问题没有可行的解，那么开发这个工程所花费的任何时间、人力和经费都是浪费。所以在软件系统开发之前，还要进行可行性分析。

可行性分析就是系统分析员站在较高的角度去调查现行手工系统及用户提出的项目目标，并且去寻找是否有一种手段能够在现有条件下，实际地达到项目目标，并使用户满意。同时向用户指出该系统实现的意义，以使用户权衡花费这样的代价实现这样的系统是否值得。

1. 可行性分析的任务

可行性分析的任务是从技术上、经济上、使用上、法律上分析应解决的问题是否有可行的解，从而确定该软件是否值得去开发。其目的是用极少的代价在最短时间内确定被开发的软件是否能开发成功，以避免盲目投资带来的巨大浪费。

可行性分析与风险分析是密切相关的。如果项目的风险很大，就会降低产生高质量软件的可行性。可行性分析主要集中在以下 5 个方面。

1)经济可行性分析

经济可行性分析主要进行开发成本的估算及可能取得效益的评估,确定待开发系统是否值得投资开发。

2)技术可行性分析

技术可行性分析主要对待开发的系统进行功能、性能和限制条件的分析,确定在现有的资源条件下,技术风险有多大,系统是否能实现。技术可行性常常是最难决断的方面,因为系统的目标、功能和性能都比较模糊。因此,技术可行性的分析、评估和定义过程并行进行是十分必要的。

一般来说,技术可行性分析要考虑的情况包括如下几个。

(1)开发的风险。确定在给出的限制范围内,能否设计出系统并实现必要的功能和性能。

(2)资源的有效性。确定可用于开发系统元素的人员是否存在问题,是否具备可用于建立系统的其他资源,如硬件、软件等。

(3)技术。确定相关技术的发展是否能支持这个系统。

开发人员在评估技术可行性时,一旦估计错误,将会出现灾难性的后果。

3)操作可行性分析

操作可行性一般是指目标系统的操作在这个组织内部是否行得通。操作可行性分析主要分析用户组织的结构、工作流程、管理模式及规范是否适合目标系统的运行,是否互不相容;现有的人员素质能否胜任对目标系统的操作;如果进行培训,时间是多少? 成本如何?

操作可行性一般包括用户类型(外行型/熟练型/专家型)和操作习惯两个方面。

4)法律可行性分析

法律可行性分析主要确认待开发系统可能会涉及的任务侵犯、妨碍、责任等问题。法律可行性所涉及的范围比较广,包括合同、责任、侵权以及其他一些技术人员常常不了解的陷阱。

5)使用可行性分析

使用可行性是指用户是否容易接受使用方式,如操作方式。对于一个运行方式难以让人习惯的软件,用户是不会满意的。

对于大多数系统(除国防系统、法律委托系统和高技术应用系统等外),一般衡量经济上是否合算,应考虑一个“底线”。经济可行性分析涉及范围较广,包括成本—效益分析、长期的公司经营策略、对其他单位或产品的影响、开发所需的成本和资源,以及潜在的市场前景等。

在进行可行性分析时,通常要先分析目前正在使用的系统,然后根据待开发系统的要求导出新系统的高层逻辑模型。有时可能提出几个可选择的方案,并对每个方案从技术上、经济上、使用上、法律上进行可行性分析,在对各方案进行比较后,选择其中一个作为推荐方案(有时可能要在几个方案之间进行折中),最后对推荐方案给出一个明确的结论,如“可行”、“不可行”或“等某条件成熟后可行”等。

2. 可行性分析的步骤

可行性分析的步骤如图 3-1 所示。

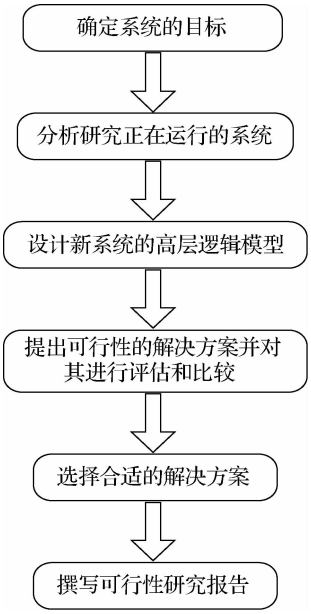


图 3-1　可行性分析的步骤

1)确定系统的目标

系统分析人员要详细阅读各种相关资料,并对用户和市场进行调查,从而确认目标系统要完成的任务。同时,还要明确进行项目开发时的一切限制和约束,以及可以使用的各种资源。

2)分析研究正在运行的系统

对现有系统功能特点的充分了解是成功开发新系统的前提。对现有系统的分析包括阅读和分析各种文档资料、观察系统的运行状况和实地操作系统、收集和分析用户对现有系统的意见等。可以说,现有系统是开发目标系统重要的信息来源。

3)设计新系统的高层逻辑模型

一般来说,新系统应该完成现有系统的功能,并对现有系统中存在的问题进行改善或修复。在分析现存系统的基础上,就可以大体把握新系统的功能和结构,然后从较高层次建立新系统的逻辑模型。

4)提出可行性的解决方案并对其进行评估和比较

基于新系统的高层逻辑模型,系统分析人员可以从技术的角度提出多种解决方案,并从经济、社会和技术等多个方面对各种解决方案进行比较和评估。

5)选择合适的解决方案

在上述分析的基础上,回答该软件产品是否能够解决存在的问题,是否能够带来预期的效果和价值的问题。若该软件开发项目没有必要性和可能性,则应立即停止,并给出详细的理由;如果有开发该软件产品的必要性和可能性,那么应该从上述多个解决方案中选取最合适、最可行的解决方案,列举出选择该方案的原因,并从经济可行性、社会可行性和技术可行性等 3 个方面对该方案进行可行性分析。

6)撰写可行性研究报告

可行性研究报告是可行性分析阶段的输出文档,应该包括的内容有项目背景、管理概要、候选方案、系统描述、经济可行性分析、社会可行性分析、技术可行性分析及可行性分析的结论等。

3. 可行性分析的结论

可行性分析的结论一般有如下 3 种。

(1)可以按计划进行软件项目的开发。

(2)需要解决某些存在的问题(如资金短缺、设备陈旧和开发人员短缺等)或者需要对现有的解决方案进行一些调整或改善后才能进行软件项目的开发。

(3)待开发的软件项目不具有可行性,立即停止该软件项目。

上述可行性分析的步骤只是一个经过长期实践总结出来的框架,在实际的使用过程中,它不是固定的,根据项目的性质、特点以及开发团队对业务领域的熟悉程度会有些变化。

3.1.3 可行性分析文档

可行性分析可以归档为一个单独的报告提供给上级管理部门,也可以包括在“系统规格说明”的附录中。可行性分析报告的形式可以有多种,为了使读者能够具体了解如何编写可行性分析报告技术文档,下面对可行性分析报告的内容要求及写法进行简要说明。

1)引言

引言说明编写本文档的目的,项目的名称、背景,本文档用到的专门术语和参考资料。

2)可行性分析前提

可行性分析前提说明开发项目的功能、性能和基本要求、达到的目标、各种限制条件、可行性分析方法和决定可行性的主要因素。

3)对现有系统的分析

对现有系统的分析说明现有系统的处理流程和数据流程、工作负荷、各项费用的支出、所需各类专业技术人员和数量、所需各种设备、现有系统存在的问题。

4)所建设系统的技术可行性分析

所建设系统的技术可行性分析简要说明所建设系统的处理流程和数据流程,与现有系统比较的优越性,采用所建议系统对用户的影响,对各种设备、现有软件、开发环境和运行环境的影响,对经费支出的影响,对技术可行性的评价。

5)所建设系统的经济可行性分析

所建设系统的经济可行性分析说明所建设系统的各种支出、各种效益、收益/投资比、投资回收周期。

6)社会因素可行性分析

社会因素可行性分析说明法律因素对合同责任、侵犯专利权和侵犯版权等问题的分析,说明用户使用可行性是否满足用户行政管理、工作制度等的要求。

7)其他可选方案

其他可选方案逐一说明其他可选方案,并说明未被推荐的理由。

8)结论意见

结论意见说明项目是否能开发,还需要什么条件才能开发,对项目目标有何变动等。

3.1.4　软件项目开发计划书

软件项目开发计划书是一种管理性文档，其主要内容如下。

(1)项目概述。项目概述包括项目目标、主要功能、系统特点以及关于开发工作的安排。

(2)系统资源。系统资源包括开发和运行该系统所需要的各种资源，如硬件、软件、人员和组织机构等。

(3)费用预算。费用预算说明完成项目的总费用及资金计划。

(4)进度安排。进度安排说明开发项目的周期、开始及完成时间。

(5)交付的产品清单。

软件项目开发计划书一般供软件开发单位使用。

可行性分析是抽象和简化了的系统分析和设计的全过程，它的目标是用最小代价尽快确定问题是否能够解决，以避免盲目投资带来的巨大浪费。可行性分析是从技术上、经济上、使用上、法律上分析应解决的问题是否有可行的解，从而确定该软件是否有可行的解。

3.2　需 求 分 析

软件需求分析是软件生存期的一个重要阶段，是软件开发项目得以成功完成的基础。其最根本的任务是确定为了完成用户需要的软件系统所必须做的事情。

软件需求分析是一个不断发现和决定的过程，在此过程中，软件开发者和软件申请者(用户)同样起着重要的作用。

3.2.1　需求分析的目标和任务

软件需求分析需要实现以下几个目标。

(1)给出软件系统的数据流程图与数据结构，构造一个完整的系统逻辑模型。

(2)提出详细的功能说明，确定设计限定条件，规定性能要求。

(3)密切与用户联系，使用户明确自己的任务，以便实现上述两个目标。

需求分析的基本任务是借助当前系统的逻辑模型导出目标系统的逻辑模型，解决目标系统“做什么”的问题，确定系统必须完成哪些工作，也就是对目标系统提出完整、准确、清晰和具体的要求，并在需求分析阶段结束之前，由系统分析员写出软件需求规格说明书，以书面形式准确地描述软件需求。系统逻辑模型如图 3-2 所示。

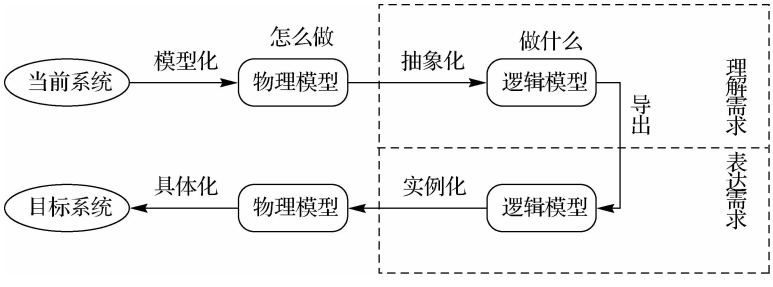


图 3-2　系统逻辑模型

3.2.2 需求分析的步骤

需求分析的步骤如图 3-3 所示。

图 3-3 需求分析的步骤

1. 获取需求,识别问题

开发人员从功能、性能、界面和运行环境等多个方面识别目标系统要解决哪些问题以及满足哪些限制条件,这个过程就是对需求的获取。开发人员通过调查研究来理解当前系统的工作模型和用户对新系统的设想与要求。

此外,在获取需求时,还要明确用户对系统的安全性、可移植性和容错能力等其他方面的要求。例如,需要多长时间对系统进行一次备份,系统对运行的操作系统平台有何要求,发生错误后重启系统允许的最长时间是多少等。

其中,遗漏需求是最难修订的需求错误。

获取需求是需求分析的基础。为了能有效地获取需求,开发人员应该采取科学的需求获取方法。在实践中,获取需求的方法有很多种,例如,问卷调查、访谈、实地操作、建立原型和研究资料等。

问卷调查法是采用调查问卷的形式来进行需求分析的一种方法。通过对用户填写的调查问卷进行汇总、统计和分析,开发人员便可以得到一些有用的信息。采用这种方法时,调查问卷的设计很重要。一般在设计调查问卷时,要合理地控制开放式问题和封闭式问题的比例。

开放式问题的回答不受限制,自由灵活,能够激发用户的思维,使他们能尽可能地阐述自己的真实想法。但是,对开放式问题进行汇总和分析的工作会比较复杂。

封闭式问题的答案是预先设定的,用户从若干答案中进行选择。封闭式问题便于对问卷信息进行归纳与整理,但是会限制用户的思维。

2. 分析需求,建立目标系统的逻辑模型

在获得需求后,开发人员应该对问题进行分析和抽象,并在此基础上从高层建立目标系统的逻辑模型。模型是对事物高层次的抽象,通常由一组符号和组织这些符号的规则组成。常用的模型图有数据流图、E-R 图、用例图和状态转换图等,不同的模型从不同的角度或侧重点描述目标系统。绘制模型图的过程既是开发人员进行逻辑思考的过程,也是开发人员进一步认识目标系统的过程。

3. 将需求文档化

获得需求后要将其描述出来,即将需求文档化。对于大型的软件系统,需求阶段一般涉及以下几个方面。

(1)编写系统需求文档,即软件需求规格说明书,把双方共同的理解与分析结果用规范的方式描述出来。

(2)编写初步用户手册,着重反映所开发软件的用户功能界面和用户使用的具体要求。

用户手册能强制系统分析人员从用户使用的角度考虑软件。

(3)编写确认测试计划。测试计划一般由项目经理制订,它是测试的起始步骤和重要环节,作为今后确认和验收的重要依据。

(4)修改并完善软件开发计划。由于在需求分析阶段对所开发的系统有了进一步的了解,因此能准确估计系统开发成本、进度和资源要求,可以对原开发计划进行适当的修正。

对于简单的软件系统而言,需求阶段只需要输出软件需求文档就可以了。软件需求规格说明书主要描述软件的需求,从开发人员的角度对目标系统的业务模型、功能模型和数据模型等方面内容进行描述。作为后续的软件设计和测试的重要依据,需求阶段的输出文档应该具有清晰性、无二义性和准确性,并且能够全面和确切地描述用户需求。

4. 需求验证

需求验证是对需求分析的成果进行评估和验证的过程。为了确保需求分析的正确性、一致性、完整性和有效性,提高软件开发的效率,并为后续的软件开发做好准备,需求验证的工作是非常必要的。

在需求验证的过程中,可以对需求阶段的输出文档进行多种检查,例如,一致性检查、完整性检查和有效性检查等。同时,需求评审也是在这个阶段进行的。

3.2.3　需求分析的方法

需求分析的方法有很多,这里介绍原型化方法、结构化方法和面向对象分析方法三种。

1. 原型化方法

原型化方法就是尽可能快地建造一个粗糙的系统,该系统实现了目标系统的某些或全部功能,但是这个系统可能在可靠性、界面友好性或其他方面存在缺陷。建造这样一个系统的目的是为了考察待开发系统在某一方面的可行性,如算法的可行性、技术的可行性等,或考察其是否满足用户的需求等。如果为了考察系统是否满足用户的要求,可以用某些软件工具快速地建造一个原型系统,这个系统只是一个界面,然后听取用户的意见,以改进这个原型,以后的目标系统就在此原型系统的基础上开发。

原型主要有三种类型:探索型、实验型和进化型。

(1)探索型的目的是要弄清楚对目标系统的要求,确定所希望的特性,并探讨多种方案的可行性。

(2)实验型在用于大规模开发和实现前,考核方案是否合适、规格说明是否可靠。

(3)进化型的目的不是改进规格说明,而是将系统建造得易于变化,在改进原型的过程中,逐步将原型进化成最终系统。

使用原型化方法的两种不同策略是废弃策略和追加策略。废弃策略是先建造一个功能简单且质量要求不高的模型系统,然后针对这个系统反复进行修改,形成比较好的思想,据此设计出较完整、准确、一致、可靠的最终系统。系统构造完成后,原来的模型系统就被废弃不用。探索型和实验型属于这种策略。追加策略是先构造一个功能简单且质量要求不高的模型系统作为最终系统的核心,然后通过不断地扩充修改,逐步追加新要求,将其发展成为最终系统。进化型属于这种策略。

原型是一个可实地运行的模型,有正式产品的主要特征,但不是全部特征。

软件原型是软件系统的最初版本，是以最少的费用、最短的时间开发出的，以反映最后软件主要特征的系统。原型开发指的是建立一个系统的早期版本的演习(practice)，它不必反映最终产品的所有性能，而只要反映用户感兴趣的一些方面。

原型的作用为：由于系统开发初期很难确定用户需求规格，它是解决用户与开发者之间鸿沟的有效方法，以原型(软件产品的样品)为共同语言，实现用户与开发者的双向沟通。

由于开发一个原型需要花费一定的人力、物力、财力和时间，而且用于确定需求的原型在完成使命后一般就被丢弃，因此，是否使用快速原型法必须考虑软件系统的特点、可用的开发技术和工具等方面的因素。

原型的核心是 DD(data dictionary，数据字典)，它是系统所涉及的各种数据对象的总和。

从数据字典出发可以构建以下图形。

(1)E-R 图(entity-relation diagram，实体-关系图)用于描述数据对象间的关系，它代表软件的数据模型，在实体-关系图中出现的每个数据对象的属性均可用数据对象说明来描述。

(2)DFD(data flow diagram，数据流图)的主要作用是指明系统中数据是如何流动和变换的，以及描述数据流进行变换的功能。在 DFD 中出现的每个功能的描述则写在加工说明(PSPEC)中，它们一起构成功能模型。

2. 结构化方法

结构化方法(structured approach，SA)采用“抽象”和“分解”两个基本手段，用抽象模型的概念，按照软件内部数据传递、变换的关系，由上而下逐层分解，直到找到满足功能需要的所有可实现的软件元素为止。

SA 采用“分解”的方式来理解一个复杂的系统，“分解”需要有描述手段，数据流程图就是作为描述信息流程和分解的手段而引入的。

SA 的基本步骤如下。

(1)自顶向下，功能分解。

(2)画分层 DFD。

(3)由后向前，定义数据和加工。

(4)编写 DD(数据字典)、PSPEC(加工说明)。

(5)根据需要分析复杂数据和动态模型。

(6)画 E-R 图、CFD(控制流图)、CSPEC(控制说明)以及 STD(状态变迁图)。

(7)编写 SRS(软件需求说明)。

3. 面向对象分析方法

1)基本步骤

定义系统的用例；领域分析，建立类—对象模型；建立对象—关系模型；建立对象—行为模型；编写软件需求说明。

2)面向对象分析模型的组成结构

处于 OOA 模型核心的是使用实例(use case)，简称用例。获得软件需求后，软件分析员即可据此创建一组场景(scenario)，每个场景包含一个用例，从这些用例出发，进一步抽取和

定义 OOA 模型的以下三种模型。

(1)类-对象模型。该模型描述系统所涉及的全部类及对象，每个类及对象都通过属性、操作和协作来进一步描述。

(2)对象-关系模型。该模型描述对象之间的静态关系，同时定义系统中所有重要的消息路径，它也可以具体化到对象的属性、操作和协作者。

(3)对象-行为模型。该模型描述系统的动态行为，即对象在特定的状态下如何反映外界的事件。

3.3 系统流程图

在进行可行性分析时需要了解和分析现有系统，并以概括的形式表达对现有系统的认识。进入设计阶段以后应该把设想的新系统的逻辑模型转变成物理模型，因此需要描绘未来的物理系统的概貌。

系统流程图是描绘物理系统的传统工具。所谓物理系统，就是一个具体实现的系统，也就是描述一个单位、组织的信息处理的具体实现的系统。它的基本思想是用图形符号以黑盒子的形式描绘系统里面的每个部件(程序、文件、数据库、表格、人工过程等)。

在可行性分析中，可以通过画系统流程图来了解要开发的项目的大概处理流程、范围和功能等。系统流程图不仅能用于可行性分析，还可用于需求分析。

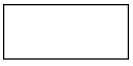
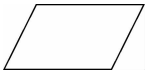
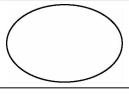
注意：尽管系统流程图的某些符号和程序流程图的符号形式相同，但是它却是物理数据流图而不是程序流程图。

3.3.1 系统流程图的符号

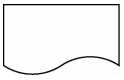
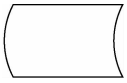
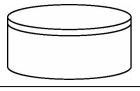
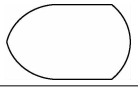
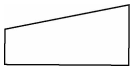
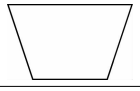
系统流程图可用图形符号来表示系统中的各个元素，如人工处理、数据处理、数据库、文件和设备等。它表达了系统中各个元素之间信息流动的情况。

画系统流程图时，首先要弄清业务处理过程及处理中的各个元素，同时要理解系统流程图各个符号的含义，选择相应的符号来代表系统中的各个元素，所画的系统流程图要反映出系统的处理流程。系统流程图的符号如表 3-2 所示。

表 3-2　系统流程图符号及相关说明

符　号	名　称	说　明
	处理	能改变数据值或数据位置的加工或部件
	输入/输出	表示输入/输出，是一个广义的不指明具体设备的符号
	连接	转出到图的另一部分或从图的另一部分转出来，通常在同一页上
	换页连接	转出到另一页图上或由另一页图转来

续表

符　号	名　称	说　明
	数据流	用来连接其他符号，指明数据流动方向
	文档	通常表示打印输出，也可表示用打印机终端输入数据
	联机存储	表示任何种类的联机存储，包括磁盘、软件和海量存储器件等
	磁盘	磁盘输入/输出，也可以表示存储在磁盘上的文件或数据库
	显示	CRT 终端或类似显示部件，可用于输入或输出，也可既输入又输出
	人工输入	人工输入数据的脱机处理，如填写表格
	人工操作	人工完成的处理，如会计在工资支票上签名
	辅助操作	使用设备进行的联机操作
	通信链路	通过远程通信线路或链路传送数据

3.3.2　一个具体例子

介绍系统流程图最好的办法就是通过一个具体的例子来说明它的用法。下面以某高校考试业务为例规划系统流程。

该系统需要完成如下几件事情。

(1)命题人员依据大纲在试题库中抽取考题形成试卷。

(2)教务部门印制试卷，安排日程及监考人员。

(3)教务部门根据日程安排学生考试，考生完成答卷。

(4)教师批改试卷，成绩交成绩管理子系统处理。

其系统流程图如图 3-4 所示。

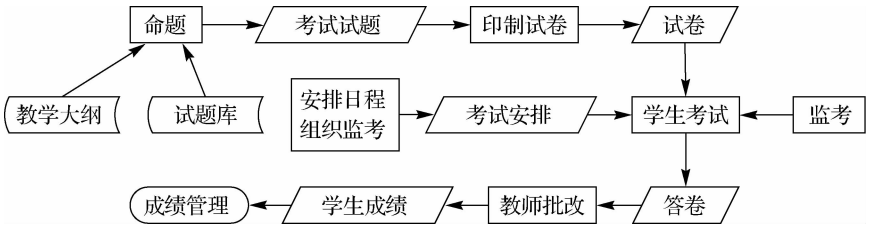


图 3-4　教务子系统系统流程图

3.3.3　分层

面对复杂系统时，一个比较好的方法是分层次地描述该系统。先使用一个高层次的系统流程图描绘总体概貌，画出关键功能；然后再对每个关键功能进行详细描绘，描绘在单独的一页上。这种分层次的描绘方法便于从抽象到具体，一步一步地深入了解复杂系统。

3.4　实体-关系(E-R)图

为了把用户的数据要求清楚、准确地描述出来，系统分析员通常要建立一个概念性的数据模型。概念性数据模型是一种面向问题的数据模型，是按照用户的观点对数据建立的模型。它描述了从用户角度看到的数据，反映了用户的现实环境，而且与软件系统中的实现方法无关。

数据模型中包含 3 种相互关联的信息：数据对象、数据对象的属性及数据对象彼此间相互连接的关系。

3.4.1　数据对象

数据对象是对软件必须理解的复合信息的抽象。所谓复合信息是指具有一系列不同性质或属性的事物，仅有单个值的事物不是数据对象。

数据对象可以是外部实体(如产生或使用信息的任何事物)、事物(如报表)、行为(如打电话)、事件(如响警报)、角色(如教师、学生)、单位(如计算机系)、地点(如仓库)或结构(如文件)等。总之，可以由一组属性来定义的实体都可以被认为是数据对象。

数据对象彼此间是有关联的，如教师“教”课程，学生“学”课程，教或学的关系表示教师和课程或学生和课程之间的一种特定的连接。

数据对象只封装了数据而没有对施加于数据上的操作的引用，这是数据对象与面向对象范型中的类或对象的显著区别。

3.4.2　属性

属性定义了数据对象的性质。必须把一个或多个属性定义为标识符，也就是说，当我们希望找到数据对象的一个实例时，用标识符属性作为关键字。

应该根据对所要解决问题的理解，来确定特定数据对象的一组合适的属性。如为了开发学生学籍管理系统，描述学生的属性应该是学号、姓名、性别、班级、籍贯、家庭住址、联系电话等。但是为了开发学生成绩管理系统，用上述这些属性描述学生就不合适了，其中，籍贯、家庭住址、联系电话等属性应该删去，而描述学生成绩的相关属性则应该添加进来。

3.4.3　联系

客观世界中的事物彼此间往往是有联系的，如教师与课程间存在“教”这种联系，而学生与课程间则存在“学”这种联系。

数据对象彼此之间相互连接的方式称为联系，也称为关系。联系可分为以下 3 种类型。

(1)一对一联系(1∶1)。如一个科室有一个主任,而每个主任只在一个科室任职,则科室与主任的联系是一对一。

(2)一对多联系(1∶n)。如某高校教师与课程间存在一对多的联系“教”,即每位教师可以教多门课程,但是每门课程只能由一位教师来教。

(3)多对多联系(m∶n)。如某高校学生与课程间的联系“学”是多对多的,即一个学生可以学多门课程,而每门课程可以有多个学生来学。

图 3-5 所示为某高校教学管理 E-R 图。

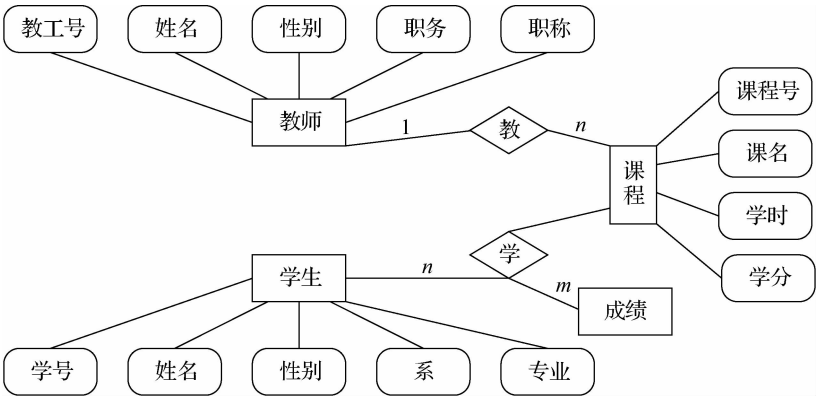


图 3-5　某高校教学管理 E-R 图

联系也可能有属性,如学生“学”某门课程所取得的成绩,既不是学生的属性,也不是课程的属性。由于成绩既依赖于某名特定的学生,又依赖于某门特定的课程,故它是学生与课程之间的联系“学”的属性。

3.4.4　实体-关系图的符号

一般使用实体-关系图来建立数据模型。可以把实体-关系图简称为 E-R 图,相应地可以把用 E-R 图描绘的数据模型称为 E-R 模型。

E-R 图中包含了实体、关系和属性 3 种基本成分,通常用矩形框代表实体,用连接相关实体的菱形表示关系,用椭圆或圆角矩形表示实体的属性,并用直线把实体与其属性连接起来,如图 3-5 所示。

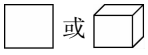
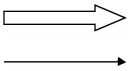
3.5　数 据 流 图

数据流图(data flow diagram,DFD)是用来描述系统逻辑模型的一种图形工具。数据流图从数据传递和加工的角度,以图形的方式描述数据流从输入到输出的移动变换过程。数据流图是描述逻辑模型的图形工具,表示数据在系统内的变化。数据流图可以用来表示一个系统或软件在任何层次上的抽象。较大型软件系统的数据流图可以分成多层(子图、父图),能够表示数据流和功能的细节。

3.5.1　数据流图中的符号

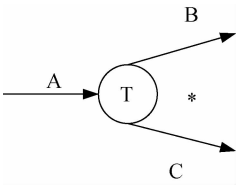
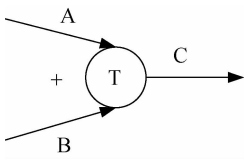
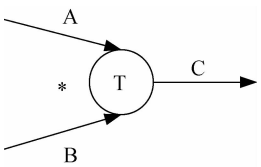
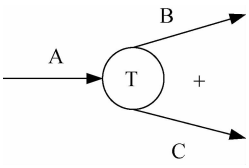
数据流图有 4 种基本图形符号，如表 3-3 所示。

表 3-3　数据流图的 4 种基本图形符号及其说明

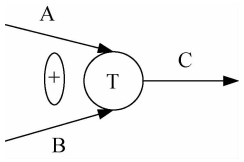
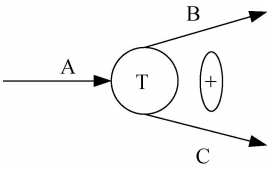
符　号	说　明
	表示外部实体，代表数据源和数据池
	表示加工，代表接收输入，经过变换，继而产生输出的处理过程
	表示数据流，代表数据的流向和路径
	表示数据存储，代表系统加工的数据所存储的地方

另外，还有一些辅助的图例，如表 3-4 所示。

表 3-4　辅助图例及其说明

符　号	说　明
	数据 A 变换成 B 和 C
	数据 A 和 B 同时输入才能变换成数据 C
	数据 A 或 B，或 A 和 B 同时输入变换成 C
	数据 A 变换成 B 或 C，或 B 和 C

续表

符　号	说　明
	只有数据 A 或只有数据 B 输入时变换成 C
	数据 A 变换成 B 或 C，但不能变换成 B 和 C

1. 源点和终点(外部项)

源点和终点是软件系统之外的实体，它可以是人或事物，表示该软件系统数据的外部来源和去处，如顾客、旅行社等。它也可以是另外一个软件系统，它向该软件系统提供数据或接收来自该软件系统发出的数据。旅客管理系统中的旅客就是处理询问系统的源点和终点。

确定了软件系统的外部项，实际上就确定了软件与外界的分界线。因此，要想确定合理的软件系统与外界的分界线，就必须详细分析用户的要求，根据软件系统的目标，确定软件的分界线。可能这个外部项定得不合理，而应该将其作为软件系统的一个组成部分纳入本软件系统；也可能软件系统内部的某一个功能部分不适当，而应从软件内部移出去作为软件系统的外部项对待。

2. 数据流

数据流是由一组数据组成的。在旅客管理系统中数据流“旅客”是由“姓名”、“住址”、“电话号码”等数据组成的。又如，数据流“机票”是由“姓名”、“日期”、“航班号”、“起点”、“终点”等数据组成的。

数据流可以从变换流向变换，也可以从变换流向数据存储或从数据存储流向变换，还可以从源点流向变换，或从变换流向终点。

3.5.2　设计数据流图的步骤

画数据流图的步骤如下。

(1)首先画系统的输入/输出，即先画顶层数据流图。顶层流图只包含一个加工，用以表示被开发的系统，然后考虑该系统有哪些输入数据，这些输入数据从哪里来；有哪些输出数据，输出到哪里去等问题。

(2)画系统内部，即画下层数据流图。一般将层从 0 开始编号，采用自顶向下，由外向内的原则。

需要注意如下几个事项。

(1)命名。不论数据流、数据存储还是加工，合适的命名使人易于理解其含义。

(2)一般不画物质流。数据流反映能用计算机处理的数据,而不是实物,因此对目标系统的数据流图一般不要画物质流。

(3)父图与子图的平衡。子图的输入/输出数据流同父图相应加工的输入/输出数据必须一致,此即父图与子图的平衡。

(4)局部数据存储。当某层数据流图中的数据存储不是图中相应加工的外部接口,而只是本图中某些加工之间的数据接口,则称这些数据存储为局部数据存储。

(5)提高数据流图的易理解性。注意合理分解,要把一个加工分解成几个功能相对独立的子加工,这样可以减少加工之间输入/输出数据流的数目,增加数据流图的可理解性。

3.5.3　案例分析

学校目前所使用的系统是教务处的系统,是对学生信息进行管理的系统,如图 3-6 所示。该系统的主要功能就是输入存储学生信息,对学生信息的基本查询、修改和更新、删除等。性能上,速度不是很快,主要由于后台数据存盘大、数据之间的管理不是很好、数据的交互存在不足等。在应用方面,只有少部分信息对外开放,不能达到多数用户的需求,对学生学籍的管理不能非常系统地进行管理。

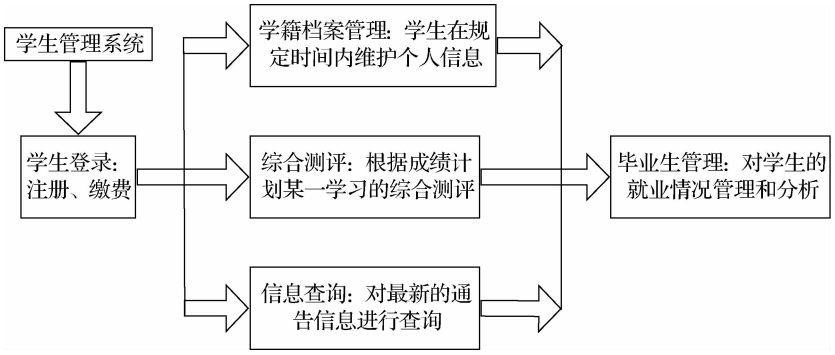


图 3-6　学生管理系统

数据流程图的特点如下。

(1)可以表示任何一个系统(人工的、自动的或混合的)中的数据流程。

(2)每个表示加工的圆圈可能需要进一步分解以求得对问题的全面理解。

(3)着重强调的是数据流程而不是控制流程。

(4)推导数据流程图的简单准则如下。

- 第一层数据流程图应当是基本的系统模型。
- 应当仔细说明原始的输入/输出文件。
- 所有箭头和圆圈均应当加上标注(使用有意义的名字)。
- 必须保持信息的连续性。
- 每次只加工一个圆圈。

3.6 数 据 字 典

数据字典是关于数据信息的集合，也就是对数据流图中包含的所有元素定义的集合。

任何字典最主要的用途都是供人们对不了解的条目的解释进行查阅，数据字典的作用也正是在软件分析和设计的过程中给人们提供关于数据的描述信息。

3.6.1 数据字典的定义

数据字典(data dictionary，DD)是分析模型中出现的所有名字的一个集合，包括有关命名实体的描述。

数据字典有以下两个作用。

(1)它是所有名字信息管理的有效机制。

(2)作为连接软件分析、设计、实现和进化阶段的开发机构的信息存储。

数据字典应该由如下 4 类元素定义组成。

(1)数据流。

(2)数据流分量。

(3)数据存储。

(4)处理。

3.6.2 数据字典的用途

在数据流程图中，对所有图形元素都进行了命名，所有名字的定义集中起来就构成了一本数据字典。

数据字典最重要的用途是作为分析阶段的工具。在数据字典中建立一组严密一致的定义有助于改进分析员和用户之间的通信，因此能消除许多可能产生的误解。对数据的这一系列严密一致的定义也有助于改进不同的开发人员之间或者不同开发小组之间的通信。如果要求所有开发人员都根据公共的数据字典描述数据或设计模块，则能避免许多麻烦的接口问题。

数据字典中包含的每个数据元素的控制信息是很有价值的。因此列出使用一个给定的数据元素的所有程序(或模块)，将很容易估计改变一个数据将会产生的影响，并能对所有受影响的程序或模块进行相应改变。

数据字典是开发数据库的第一步，而且是很有价值的一步。

3.6.3 数据字典的实现

目前，实现数据字典有三种常见的途径：全人工过程、全自动化过程(利用数据字典处理程序)和混合过程(用正文编辑程序、报告生成程序等已有的实用程序辅助人工过程)。无论使用哪种途径，实现的数据字典都应该具有下述特点。

(1)通过名字能方便地查阅数据的定义。

(2)没有冗余。

(3)尽量不重复在规格说明的其他组成部分中已经出现的信息。

(4)容易更新和修改。

(5)能单独处理描述每个数据元素的信息。

(6)定义的书写方法简单方便而且严格。

如果暂时没有自动的数据字典处理程序,建议采用卡片形式书写数据字典,每张卡片上保存描述一个数据元素的信息。这种处理方法容易实现上述要求,尤其是更新和修改起来很方便,能够单独处理每个数据元素的信息。每张卡片上主要应该包含名字、别名、描述、定义、位置等信息。

在开发过程进展到能够知道数据元素的控制信息和使用特点时,就把这些信息记录在卡片的背面。

下面给出订货系统中的几个数据元素的数据字典卡片,以此来具体说明上述数据字典卡片中几项内容的含义。

名字:订货报表
别名:订货信息
描述:每天一次送给采购员的需要订货的零件表
定义:订货报表=零件编号+零件名称+订货数量+目前价格+主要供应者+次要供应者
位置:输出到打印机

3.7 实训:使用 Visio 2007 绘制 E-R 图及数据流图等模型

1. 实训的目的和任务

(1)掌握 Visio 2007 的基本功能操作。

(2)利用 Visio 2007 绘制 E-R 模型。

(3)熟练利用 Visio 2007 绘制数据流图。

2. 实训环境

(1)Windows XP 操作系统、Microsoft Visio 2007。

(2)Windows XP 操作系统及能够安装 Microsoft Visio 2007 的计算机。

3. 实训内容与步骤

(1)Visio 2007 软件工作环境的基本功能操作。

(2)利用 Visio 2007 绘制 E-R 模型。

(3)利用 Visio 2007 绘制数据流图。

4. 实训练习

(1)利用 Visio 2007 绘制教务监考安排系统的 E-R 图,如图 3-12 所示。

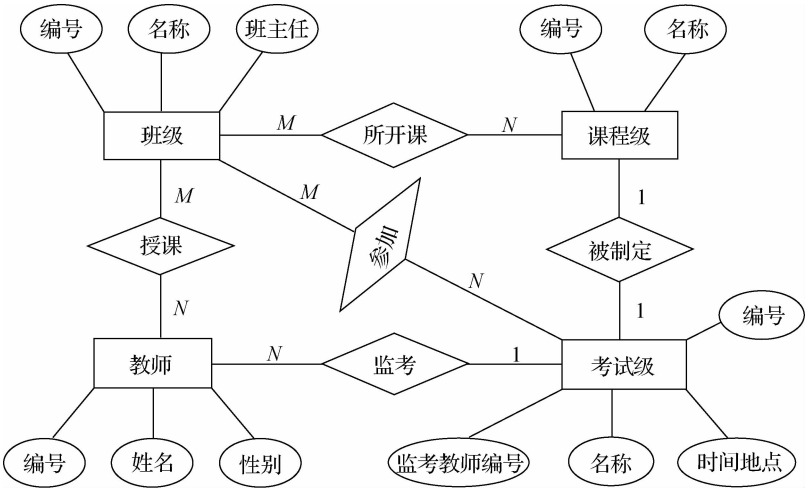


图 3-12　教务监考安排系统的 E-R 图

(2)利用 Visio 2007 绘制订货系统的功能及数据流图,如图 3-13 所示。

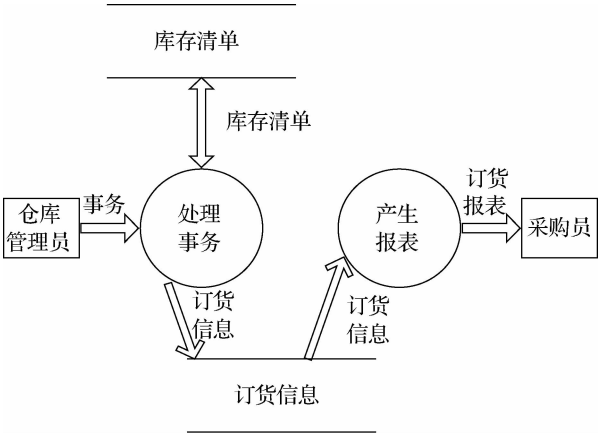


图 3-13　订货系统的功能及数据流图

提示:

(1)Visio 2007 有专用的数据库建模模板,操作为:执行“文件”→“新建”→“软件和数据库”→“数据库”命令。

(2)Visio 2007 的数据库建模模板与 E-R 图有所不同,但更好用,也更先进。

(3)若一定要画 E-R 图,可用“框图”或“基本框图”模板,其中包括所有 E-R 图需要的形状。操作为:执行“文件”→“新建”→“常规”→“框图(或基本框图)”命令。

3.8 项目拓展:高校教材采购系统建模及可行性和需求分析

1. 拓展项目

(1)进行高校教材采购系统的可行性分析和需求分析。

(2)用结构化分析方法完成教材采购系统的分层数据流图。

(3)利用 Rational Rose 绘制教材采购系统的用例图和类图。

(4)完成高校教材采购系统的可行性研究报告、项目开发计划书及软件需求规格说明书。

2. 拓展目的

(1)培养学生运用所学软件项目中进行需求分析的理论知识和技能,灵活分析并解决软件开发实际应用问题的能力。

(2)培养学生面向客户进行调查研究的能力,获得客户对软件功能和性能的需求情况,并运用软件工程的需求分析方法写出相关的文档资料。

(3)通过实训理解结构化分析方法和面向对象分析方法的异同,了解 UML 建模在需求分析阶段的应用。

3. 拓展要求

(1)能够深入所在院校的教材发行科室,了解更多有关学校管理人员对教材采购信息的需求。

(2)实训完成后,能根据可行性分析和需求分析的结果,完成高校教材采购系统的可行性研究报告、项目开发计划书及软件需求规格说明书。

(3)评价本次项目实训报告,即可行性分析报告、项目开发计划书和需求分析报告的优劣。

4. 拓展学时

本项目拓展为 6 学时。

3.9 习　　题

一、选择题

1. 可行性分析中的系统流程图用于描述(　　)。

A. 当前运行系统　　B. 当前逻辑模型

C. 目标系统　　D. 新系统

2. 系统流程图是描述(　　)的工具。

A. 逻辑系统　　B. 程序系统

C. 体系结构　　D. 物理系统

3. 下面不是需求分析具体任务的是(　　)。
A. 确定对系统的综合要求
B. 分系统的数据要求
C. 修正系统开发计划
D. 了解用户的需要
4. 需求分析中开发人员要从用户那里了解(　　)。
A. 软件做什么　　B. 用户使用界面
C. 输入的信息　　D. 软件的规模
5. 需求分析阶段的任务是确定(　　)。
A. 软件开发方法　　B. 软件开发工具
C. 软件开发费用　　D. 软件系统的功能
6. 需求分析阶段最重要的技术文档之一是(　　)。
A. 项目开发计划　　B. 设计说明书
C. 需求规格说明书　　D. 可行性分析报告
7. 需求分析阶段建立原型的目的是(　　)。
A. 确定系统的功能需求和性能需求　　B. 确定系统的运行要求
C. 确定系统是否满足用户需求　　D. 确定系统是否满足开发人员需要
8. 需求分析阶段研究的对象是(　　)。
A. 用户需求　　B. 系统分析员要求
C. 系统要求　　D. 软硬件要求
9. 数据流图是表示软件模型的一种图示方法,画数据流图应遵循的原则是(　　)。
A. 自底向上、分层绘制、逐步求精　　B. 自顶向下、分层绘制、逐步求精
C. 自顶向下、逐步求精　　D. 自底向上、分层绘制
10. 数据流图(DFD)是(　　)方法中用于表示系统逻辑模型的一种图形工具。
A. SA　　B. SD　　C. SP　　D. SC
11. 数据字典是用来定义(　　)中各个成分的具体含义的。
A. 流程图　　B. 功能结构图
C. 系统结构图　　D. 数据流图
12. 需求规格说明书的作用不包括(　　)。
A. 软件验收的依据
B. 用户与开发人员对软件要做什么的共同理解
C. 软件可行性研究的依据
D. 软件设计的依据
13. 软件开发的需求活动,其主要任务是(　　)。
A. 给出软件解决方案　　B. 给出系统模块结构
C. 定义模块算法　　D. 定义需求并建立系统模型
14. 软件需求分析一般要确定的是用户对软件的(　　)。
A. 功能需求　　B. 非功能需求
C. 性能需求　　D. 功能需求和非功能需求

15. 在数据流图中，方框符号表示(　　)。

A. 变换/加工　　B. 外部实体

C. 数据流　　D. 数据存储

16. 需求分析是(　　)。

A. 由开发人员和系统分析人员完成　　B. 由系统分析人员完成

C. 软件生存周期的开始　　D. 软件开发任务的基础性工作

17. 在软件开发过程中常用图作为描述工具。如 DFD 就是面向(　　)分析方法的描述工具。

A. 数据结构　　B. 数据流

C. 对象　　D. 构件

18. 软件开发常使用结构化方法和原型化方法，实施软件开发原型化方法应具备的必要条件是(　　)。

A. 原型系统的积累、需求的准确理解

B. 原型化开发人员、完善的开发工具

C. 软件的支持、原型系统的积累

D. 硬件的支持、原型开发系统的积累

19. 数据字典是对数据定义信息的集合，它所定义的对象都包含于(　　)中。

A. 数据流图　　B. 程序框图

C. 软件结构　　D. 方框图

20. 软件开发的结构化方法中常用数据字典技术，其中，数据加工是其组成内容之一，下述方法中，(　　)是常采用编写加工说明的方法，其中，Ⅰ——结构化语言，Ⅱ——判定树，Ⅲ——判定表。

A. Ⅰ　　B. Ⅱ　　C. Ⅱ、Ⅲ　　D. 全部

二、名词解释

1. 需求分析。
2. 结构化分析方法。
3. 数据流图。
4. 数据字典。

三、简答题

1. 可行性研究的任务是什么？
2. 可行性研究有哪些步骤？
3. 需求分析的任务是什么？
4. 数据流图的作用是什么？它的优缺点是什么？其中的符号表示什么含义？
5. 简述数据字典的用途。