分类号：　　单位代码：10014

密　级：　　学　　号：98082

中国农业大学

UML 的研究及其软件需求分析实现

Research on UML and Implement of Software Requirement Analysis with UML

研　究　生：　李丽

指　导　教　师：　邱丽娟教授

副 指 导 教 师：　许永诚　　周南

学位门类、级别：　工学硕士

专　业　名　称：　农业电气化与自动化

研　究　方　向：　计算机网络应用

答　辩　日　期：

二 OO 一年三月

摘　要

UML(Unified Modeling Language)是 Rational Software 公司和她的合作伙伴共同制定的用于描述、可视化和构架软件系统以及商业建模的语言，它涵盖了面向对象的分析、设计和实现，融合了早期面向对象建模方法和各种建模语言的优点，为面向对象系统的开发、软件自动化工具与环境提供了丰富的、严谨的、扩充性强的表达方式。UML 代表了在大型、复杂系统的建模领域得到认可的“ 优秀的软件工程方法 ”。

UML 的应用是以系统的开发流程为背景，但 UML 只是标准的建模语言，而不是一个标准的开发流程。不同的组织、不同的应用领域需要不同的开发过程。软件系统的开发流程中，需求分析是其中决定性的一步，本文的主旨就在于探讨 UML 如何应用于软件需求分析阶段并在此基础上建立了一套模型。

最后，本文以 “ 基于 Web 的远程教学系统应用平台 ” 的开发为例，展示了所建模型的具体运用。

关键词：统一建模语言、需求分析、Rose、UML

Abstract

The Unified Modeling Language (UML), which is designed by Rational Software Corporation and its partners , provides system architects working on object analysis and design with one consistent language for specifying, visualizing, constructing, and documenting the artifacts of software systems. UML is the convergence of best practices in the object-technology industry. And it is a rich, precise, extensible modeling language for object-oriented system development and software developing automation environments. UML is the representation of excellent software engineering methodology which is approbated in large-scale and complex modeling field.

Although system development flow is the background of UML applciation, UML is only a standard modeling language and not a standard development flow. Different

organizations and different application field has different development process. During the course of software developing, requirement analysis is a decisive step. The lemma of this article is to discuss UML application on requirement analysis and construct a model based on it.

Finally, this thesis uses *DEVELOPMENT OF REMOTE EDUCATION SYSTEM OF CHINA AGRICULTURAL UNIVERSITY* as a example to explain how to use this model.

Key words: UML、Rose、Requirement Analysis

目　录

引　言

1．软件系统开发方法的发展

在近代技术发展的历史上，工程学科的进步一直是产业发展的巨大动力。传统的工程学科走过的道路已为人们所熟知。但人们将工程的概念引入计算机软件领域是从六七十年代才开始的。“ 软件工程 ”的概念要求人们采用“ 工程 ”的方法来开发、维护和管理软件，这对软件产业的形成和发展起着决定性的推动作用，与此同一时期出现的结构化分析和设计在一定程度上缓解了“ 软件危机 ”。

但随着人们对软件提出的要求越来越高以及计算机的更新换代，传统的软件分析和设计方法已经无法胜任快速高效地开发当今复杂并具有人性化的软件系统。80 年代起逐步形成的面向对象的概念、方法和技术，使得人们在软件的分析、设计和实现上有了全新的认识。由于面向对象技术的模块性、封装性、继承性、多态性和动态性能满足软件工程要求的局部化、易维护、可重用、易扩充以及当今多媒体和分布式计算的诸多要求，一时成为计算机各领域争相采用的新技术焦点。整个八十年代到九十年代中期，面向对象成为业界有口皆碑的技术。

80 年代，出现了许多面向对象分析设计的方法和工具，它们有各自的优缺点，使得人们难以选择哪种方法和工具来开发他们的系统，由于它们一些概念的细微差别使得技术人员交流很困难。人们迫切需要一种面向对象分析设计的方法和工具，以便能够充分享受面向对象技术带来的便利。统一建模语言 UML（Unified Modeling Language）正是在这要求下孕育而生，结束了面向对象方法标准不统一的局面。UML 是建立在已有的三大面向对象方法学的基础上，抽象出它们的模型语言，并吸取了其它面向对象开发方法和近 30 年软件工程的成果。在面向对象分析和设计文档表达上，UML 引入了图符，体现了“ 可视化 ”这一特点，因而 UML 也称为可视化建模语言。UML 为面向对象系统的分析、设计、软件自动化工具

与环境提供了丰富、无二义性、扩充性强的表达方式，广泛地应用于系统的分析、设计和维护，成为面向对象建模的标准建模语言。

2．软件需求分析方法

与此同时，软件工程方法学也经历了结构化方法到面向对象方法的飞跃。

软件的需求分析的主要目的是，通过与用户广泛的交流得出所要完成的目标系统必须具备哪些功能，应该为用户完成些什么工作，即确定“目标系统必须做什么？”。需求分析相当于从用户到软件工程人员之间架设了一道桥梁，软件工程人员通过需求分析得到用户的需求，成为软件编制所实现的目标。需求分析的好坏直接关系到软件的成功与否，是软件生命周期中的关键一环。

一般来说，用户对计算机技术了解并不多，计算机工程人员又对用户的问题不很了解，这就阻碍了用户与计算机工程人员之间的交流，使计算机工程人员不能很好地理解问题域，用户又对目标系统存在好多不清楚的地方。传统的数据流分析法，功能分析法等对这个问题并不能有效地解决。面向对象的方法的出现，正好为此问题提供了一个较好的解决方案。因为人类自然地趋向于用“对象”的观点或方法来认识问题，分析问题以及解决问题，用基于"对象"的概念模型来建立问题域模型自然成为系统分析员与用户交流的有效工具。

用面向对象的方法进行需求分析，其根本要点在于，利用"对象"的概念模型建立一个针对于问题域的模型，用户和软件工程师通过该模型进行交流。通过在这么一个基于“对象”的问题域模型的基础上形成需求规格说明书。

采用图形描述需求是人们由来已久的愿望，可视化建模就是基于这样的思想，它将模型中的信息用标准图形元素直观地显示。可视化建模的主要目的就是用户、开发人员、分析人员、测试人员、管理人员和其他涉及项目人员之间的通信。利用非可视信息（文本）也能进行通信，但毕竟百闻不如一见，通过图形比通过文字更容易理解事情的结构。标准对实现可视化建模的通信功能至关重要，也就是用哪种图形标注方法表示系统的各个方面。常用的图注方法有 Booch、对象建模技术（OMT）和统一建模语言（UML），其中 UML 是大多数公司采用的标准。

Rational Rose 是 Rational 公司开发的一个适用于大型系统开发的面向对象的可视化分析、设计建模工具。在 Rose 工具中，可以通过四种视图来描述系统的模型：用例视图（use case view）、类和对象视图（logical view）、构件视图（component view）和配置视图（deployment view），通过这些视图可以对系统需求、处理过程、对象、构件、系统结构等进行可视化建

模。该软件支持统一建模语言（UML）、Booch 及 OMT。

3．本文的研究目标

UML 这套语言规范在面向对象分析、设计及实现上是极其优秀的可视化建模语言，但 UML 只是标准的建模语言，而不是一个标准的开发流程。不同的组织、不同的应用领域需要不同的开发过程。软件系统的开发流程中，需求分析是其中决定性的一步，本文的研究目标就在于如何将 UML 应用于软件需求分析阶段，确立了需求分析的步骤并在此基础上建立了一套模型。

4．本文的组织

本文在引言部分介绍了软件系统开发方法和软件需求分析方法的发展情况，并针对 UML 只是规范不是标准的开发流程这一背景，提出本文的研究目标：UML 软件需求分析实现。第一章提出软件系统的概念并指出当前软件系统开发的难题，然后介绍了 UML 和图式语言在开发软件系统的地位及其作用。第二章概述 UML 语言、UML 建模环境 Rose 并指出它们的不足。第三章简要地介绍了软件需求工程的发展历程及其方法和工具。第四章建立了一套利用 Rose 工具所支持的 UML 建模进行软件需求分析的模型。第五章介绍了所建模型在《基于 Web 的远程教学系统应用平台》的开发中的实际应用。第六章给出本论文的主要结论和展望。

第一章、背景知识

1．1 软件系统

在日常生活中，我们会遇到很多系统，那么什么是系统呢？系统是由一些部件有机地组合在一起，共同实现某一特定的功能或任务。在现实世界中，有两类不同的系统：自然系统和人造系统。自然系统可以大到宇宙系统、银河系统，小到分子、原子系统，它们几乎无处不在。人造系统是人们为了某一目的而构造的系统，电冰箱、洗衣机等都是人造系统，它们的出现都是为了改善人们的生活。

信息系统也是一种特殊的人造系统，可以以不同的形式不同的大小出现，仅仅受限于人类的想象力。信息系统由**数据**、**人**、**规程**、**软件**和**硬件**五大部分组成。**人**在系统中有时起提供输入数据作用，有时作相应的控制作用，有时提供系统一些反馈信息，总之人以某种方式介入到系统中；**规程**是规定人们如何去影响系统；**数据**是人们提供系统处理的资料；**软件**是信息系统的灵魂部分、主体部分，软件的好坏是直接影响到该系统的存在价值；**硬件**是软件运行的平台，是软件的物质基础。我们通常所说的软件系统就是上面所提到的软件部分，是信息系统的子系统。由于软件系统是信息系统的主要部分，有时将信息系统与软件系统不加区分地使用。

在大大小小的企事业单位，政府机构、军队都会有许多信息系统，而且他们还有改善当前信息系统和开发新信息系统的要求。例如，图书馆的借书、书目查询系统、军队的导弹制导系统、民航的机票预定系统等。随着计算机硬件的发展，这些单位部门对信息系统的需求量越来越大，质量要求越来越高。提高信息系统的生产率，提高信息系统的质量是当前信息产业的迫切要求。软件系统是信息系统的灵魂部分，构造信息系统关键是如何构造优秀的软件系统。一种优秀的软件建模方法和开发环境对于提高软件的生产率和提高软件的质量往往起到至关重要的作用。

1．2 开发软件系统的挑战

开发软件系统一般涉及三个步骤：分析、设计和实现（如图 1 所示）。分析是问题抽象 (做什么)，设计是问题求解 (怎么做)，实现是问题的解 (结果)。在每一个阶段上，又可以分为不同的子阶段：如分析阶段又分为计划、可行性分析、需求的分析和需求的确认。不管传统的结构化方法还是现代的面向对象方法都要涉及这三个阶段，在这三个阶段主要产生三种不

同的系统开发文档，即**需求说明书**、**设计规范**和**源程序**。这些文档主要用于系统分

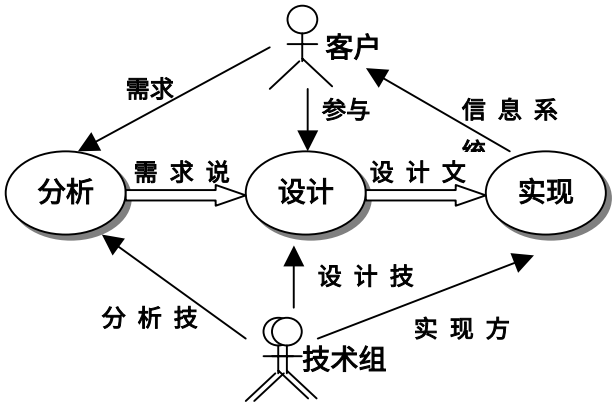


图 1-1、开发软件系统模型

析员与客户、系统分析员与设计人员之间的交流，这些文档应尽量做到无二义性，且具有良好的可理解性，对于需求说明书还必须具备通俗易懂的特点，使得业外人士也可以方便地理解。另外，技术人员如果拥有一套较好的方法和工具，对于复杂系统的分析、设计、实现和维护是极其重要的。因而自从“软件危机”出现，科学家就一直不停地探索，寻找优秀的软件系统分析设计方法，最初的结构化分析和设计方法、后来出现的面向对象分析和设计方法，都是为了有效地解决软件系统的复杂性。

1．3 UML 和图式语言的位置、作用

由上可知，具备有效地解决软件系统复杂性的分析和设计方法与可理解性和无二义性的标准描述语言对于提高软件的生产率、质量、可靠性和可维护性是具有举足轻重的作用，UML 正应这一要求孕育而生。

在 UML 出现以前，有许多面向对象建模方法和工具，但是开发人员发现没有一种建模工具能够完全满足要求，它们有各自的优缺点。主要的建模工具有 Booch3，OMT-2，OOSE，这些方法有各自的特长。简单地说：OOSE 的 Use-Case 是支持商业工程和需求分析的极好方法；OMT-2 则对描述分析和大量数据处理的信息系统有特别的表达能力；Booch3 则在项目的设计和构造阶段有极强的表达能力。由于这些工具没有一种处于领导地位，开发人员很难从这些细微差别的建模工具中选择一种来做分析和设计。正因为这些细微的差别，使得开发人员不愿采用面向对象方法来建模。在这种形势下，1996 年，OMG（对象管理集团）发起统一面向对象分析和设计方法的请求，在许多计算机科学家的共同努力下，促使 UML 的

出现。UML 的出现，标志着面向对象方法真正地走向成熟。

UML 从静态模型和动态模型两个方面来描述系统模型。静态模型，也称结构模型，主要强调一个系统中的对象结构，包括它们的类、接口、属性和关系。动态模型，也称行为模型，强调系统的对象行为，包括它们的方法、相互作用、协作和状态变化。在描述系统模型时尽量采用建模图符来刻画，具有良好的可理解性。在 UML 规范说明中，给出每个图符的详细语义，尽可能地消除软件文档存在的二义性，成为面向对象分析和设计的工业标准。怎样用这一标准来指导软件的分析、设计和实现，将是我们工作的重点。

在面向对象分析阶段所撰写的文档、所分析的需求将作为软件工程的重要文档。传统的软件开发方式是因为软件需求不充分、软件开发无计划性、软件开发过程无规范性、软件产品无评策手段等问题的日益尖锐而逐步暴露出自身的问题进而诱发“软件危机”。许多计算机和软件科学家尝试把其他工程领域中行之有效的工程学知识运用到软件开发工作中来，经过不断实践和总结，证明按工程化的原则和方法组织软件开发是有效的，也是摆脱软件危机的一个主要出路。随着软件项目的复杂化、大型化趋势，对软件文档提出了更高的要求：不但表达清晰，而且可理解性要强。图式语言以其比较直观、简洁的特点，非常适合编写软件文档。UML 元模型为 UML 的所有元素在语法和语义上提供了简单、一致、通用的定义性说明，使开发者能在语义上取得一致，消除了因人而异的最佳表达方法所造成的影响；UML 符号的表示法，为开发者或开发工具使用这些图形符号和文本语法为系统建模提供了标准。在面向对象分析和设计阶段如果采用 UML 来实现系统，将会极大提高软件文档的可理解性和可维护性。Rational 公司将 UML 融入它的 CASE 工具 ROSE 中，试图用 UML 语言支持软件开发的大部分过程，是软件分析和设计的较好的面向对象的 CASE 工具。

第二章、UML 及其 Rose2000 的简介

Rational Software 公司依据 UML 规范开发了可视化建模环境 Rose2000，它不但全面支持 UML 建模语言，而且提供了第三方二次开发的扩展接口 REI（Rose Extensibity Interface）。

2．1 UML 可视化建模语言

统一建模语言是通用的可视化建模语言，用于可视化描述和构造软件系统。UML 既简单又功能强大，在提供了大多数面向对象核心概念的同时，还提供了扩展方案，使得面向对象建模专家可以使用 UML 方便地定义大多数领域中的复杂系统。统一建模语言规范由两组相关的部分组成：

UML 语义（UML Semantics）：定义 UML 对象建模概念的抽象文法和语义的元模型。元模型为 UML 的所有元素在语法和语义上提供了简单、一致、通用的定义性说明，使开发者能在语义上取得一致，消除了因人而异的最佳表达方法所造成的影响。此外 UML 还支持对元模型的扩展定义。

UML 图符（UML Notation）：定义可视化描述 UML 语义的图符集。UML 符号的表示法，为开发者或开发工具使用这些图形符号和文本语法为系统建模提供了标准。这些图形符号和文字所表达的是应用级的模型，在语义上它是 UML 元模型的实例。

在说明 UML 语义时，用元模型来说明建模概念的抽象文法和语义，主要是对结构和行为的对象模型的语义做详细说明。静态模型（Static Models）主要强调在一个系统中的对象结构，包括它们的类、界面、属性和关系。行为模型（Behavioral Models）主要强调系统中的对象行为，包括它们的方法、相互作用、协作和状态变化，并对 UML 图符中出现的建模符号给出完整的语义说明。下面先介绍版型，然后从类的静态图、动态图以及系统的实现图三方面来介绍 UML。

2．1．1 版型（Stereotype）

版型（图 2-1 所示）是 UML 提供的一种扩展该语言的机制。通过继承 UML 中现成的建模元素，得到与其具有相同

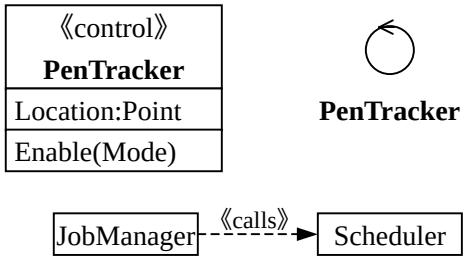


图 2-1、版型化元素

内部结构但有不同用途的建模元素。有时版型化元素（Stereotyped element）会在基类的基础上加上一些限制。对于版型化元素，没有额外的图符表示，只是在基类的名字上加上带有尖括号“《》”的关键字，例如《control》、《 calls》等。基于可视化的目的，版型化元素也可以用图标（ICON）表示，例如 PenTracker。

2．1．2 静态图

静态图包括类图和对象图，类图主要描述模型的静态结构，包括描述类（它的变形：模板类、实例化类）和类型以及它们的内部结构（属性和操作）及其它们之间的关系（关联、一般化）。对象图是类图的实例图，包含对象和数据值，描述系统在某一时刻状态。由于类图中含有实例化的元素，因而可以把对象图看作类图的一个特例。静态图最主要的图符元素是类图符和关系图符。

2．1．2．1 类

类是一组具有相似结构、行为和关系的对象的描述。UML 提供类图符用来描述类的属性、操作以及使用它的不同形式。类在类图符中说明并且可以被其他图符使用。UML 的类图符用矩形来表示，如图 2-2 所示，第一层是类名（黑体字）或其他性质（如版型）；第二、第三层是可选项，分别描述类的属性和操作，抽象类的操作采用斜体字。其中属性或操作前面的“ + ”表示 public 属性或操作；“ # ”表示 protected 属性或操作；“ - ”表示 private 属性或操作。带有下划线的属性表示类范围（calss-scope）的属性，与 C++中的静态变量具有相同的语义，也就是说该类的所有实例都共享该属性；其他属性是实例范围（instance-scope）的属性，每个实例拥有其各自所对应的属性。带有下划线的操作表示类范围（calss-scope）的操作。

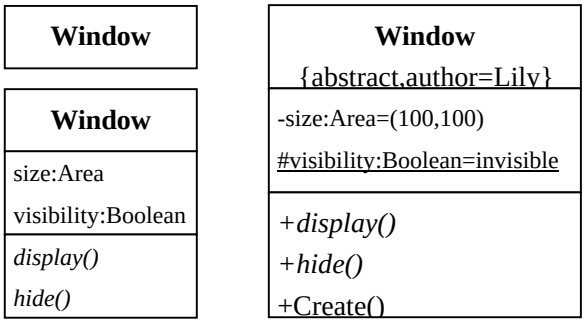


图 2-2 类图符

2．1．2．2 类型、实现类、接口

这几个建模元素从建模角度来看，与类图符具有类似的内部结构，因而 UML 并没有提供它们独立的建模图符，而是通过版型扩展类图符来提供的。类型（type）是对具有外部操作规格的对象描述，类型没有描述对象的实现方法，只提供外部行为规格。类型可能包括属

性和操作，属性定义类型的抽象状态，也可以定义类型操作的效果。实现类（implementation class）用来定义面向对象语言（C++、smalltalk）中对象的物理数据结构和过程。接口（Interface）是用来描述类、部件或其它实体的外部可见的行为。

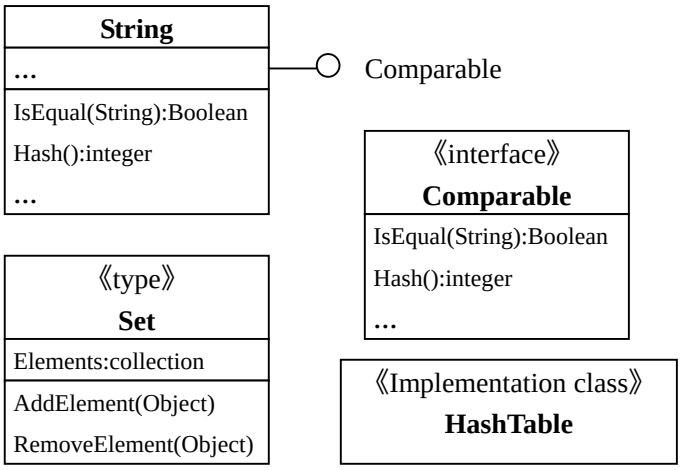


图 2-3、类型、实现类、接口

图 2-3 是这几个建模元素的图符，它们通过版型形式从类图符扩展而来，其中如果某个类、部件或实体继承某一个接口并提供了接口，可以用带有接口名的小圆圈附加到这些实体上来显示接口。如图中的 String 类继承接口 Comparable 并显示了该接口。

2．1．2．3 模板

模板（Template）也称参数类（Parameterized class），用来描述带有一个或几个未绑定形式参数的类。模板不是类，而是一组类的抽象，只有将参数绑定到具体的值才能产生类，其形式参数可以是属性类型、数据类型、甚至操作。图 2-4 中带虚框的矩形图符为模板，虚框为形式参数表，是由逗号隔开、采用 name:type 格式的参数表，name 是指模板内的参数标志符，type 是参数的类型。

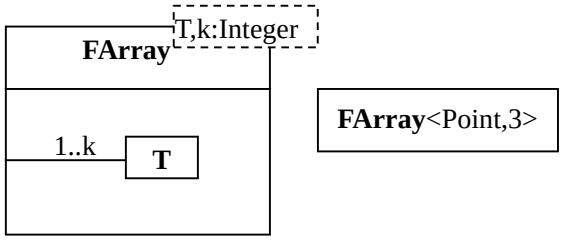


图 2-4、模板

2．1．2．4 对象

对象（Object）是类的实例，它的图符和类图符很相似。上层显示对象名和类，格式为“ Objectname：Classname ”。并在类名和对象名下面加一下划线。如果省略对象名，则代表类内满足要求的任意对象。第二层描述对象的属性以及值的列表，格式为“ Attribute：type=value ”，类型可以省略，值采用数值方式，如图 2-5 所示。

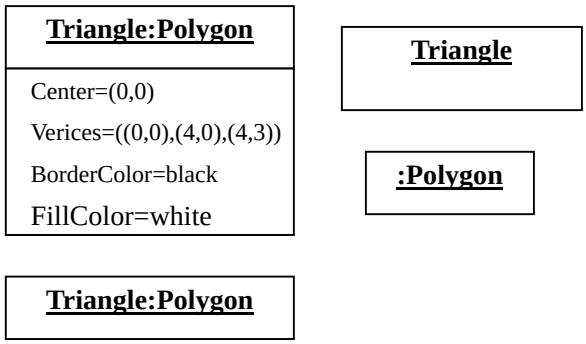


图 2-5、对象

2．1．2．5 关系

软件系统是多个不同的组件相互协作，共同完成某一特定的任务。所以，UML 定义了几种关系，描述类之间的关系（relationship），如图 2-6 所示。

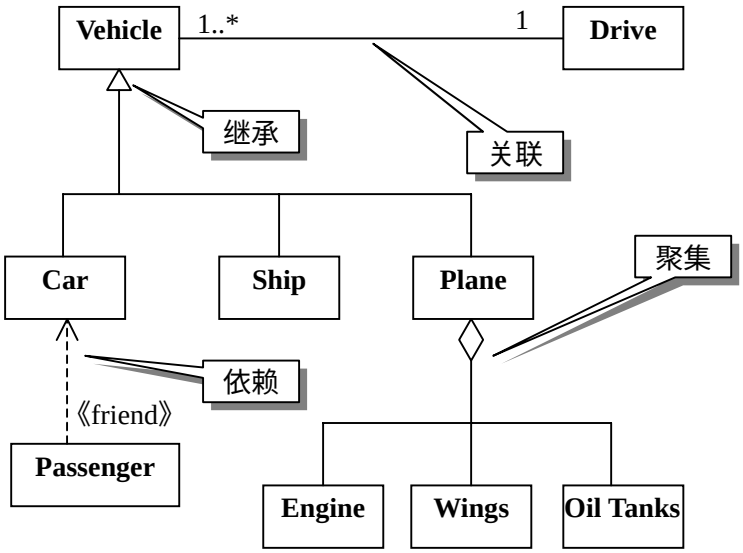


图 2-6、Transport 类图

● **关 联** association 定义了类之间的一种语义联系，是对类实例间连接的抽象。关联用带修饰符的直线表示，用带菱形的直线表示聚集。与类相连的关联终点称为关联角色，每个关联可以有两个或多个角色。关联的多数信息，如关联的阶(multiplicity)、角色名(rolename)、限定符(qualifier)、遍例(navigability)等均附加在关联角色上，阶说明参与关联实例的数量范围，如 0、1、0..1、0..n 等；角色名指明类在该关联中扮演什么角色；限定符对参与关联的类实例进行划分；遍例性指明关联是单向的还是双向的。如果没有特殊标记，一般是双向关联。

●**聚集** aggregation 是一种特殊形式的关联，用来说明对象间的整体/部分关系。聚集关系分为引用（by reference）和传值(by value)，分别用空心菱形和实心菱形表示。

●**继承** Inheritance 描述一个类共享其它类的结构和行为。

●**依赖** Dependency 表明某个元素的变化对其他元素产生的影响。

2．1．3 动态图

传统的面向对象需求分析方法只给出了系统的对象模型，没有描述系统的功能需求，而系统的功能是用户最关心的问题，为此，UML 采用（Use Case）图描述用户所关心的系统功能，用交互图和状态图从不同侧面描述各种功能的具体实现。交互图和状态图可以相互翻译，它们在描述行为采用的方式不同，区别主要表现在两个方面：一方面体现在状态图描述了它所反映的类型或协同实例的潜在行为 ;交互图仅描述在某个环境下某个行为的一条简单路径；另一方面体现在状态图通常描述单个类型的行为，而交互图涉及多个类型的行为。

2．1．3．1 Use-Case 图

Use-Case 图（图 2-7 所示）由一组被系统边界包围的 use case、系统边界外的 actor、actor 与 use case 间的通讯组成，主要在分析阶段刻画系统的功能需求和环境约束。actor 是指使用系统的用户或者与其交互的其他系统，它的引入有利于划清系统的边界。use case 是一组事务的集合，系统在 actor 触发下执行该组事务以完成某项特定的任务。use case 和 actor 之间还存在各种关系，在 UML 中用类图中的关系图元来表达这些关系，如：Communicate、Extends、Uses 等关系。

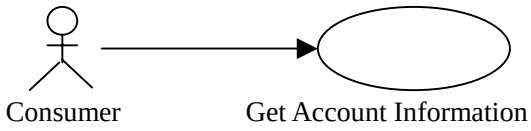


图 2-7、Use Case 图

Use-Case 图实例化后产生的实例——场景(Scenario)用来描述 use case 的动作序列的执行，可以用不同的场景刻画同一个 use case 实例。多个 use case 的多个场景建立系统的行为模型，为了刻画了这些场景，UML 使用交互图来对场景进行建模。

2．1．3．2 交互图

类图和 Use-Case 图刻画了系统的结构特性和功能需求，却无法刻画实时需求，而交互图却能应用于时序需求。交互图分为顺序图和协作图，它们的功能类似，但侧重点不同。顺序图以时间的顺序来描述对象之间的交互 ;与顺序图不同的是，协作图刻画了对象之间交互的上下文，以及与之有关的对象关系。

●**顺序图**

顺序图以时间顺序显示对象在其生命周期内的交互活动。它只显示参与的对象，而不刻画对象间的关系或对象的属性。如图 2-8 所示，顺序图采用两维坐标：垂直轴表示时间，水平轴表示不同的对象。水平有向线代表对象间交互的消息，并用消息名标记，或者用顺序号显示交互顺序。而且可以在交互附近附加各种补充信息，如：定时标记、动作等。图左边还可以增加脚本加以补充说明。

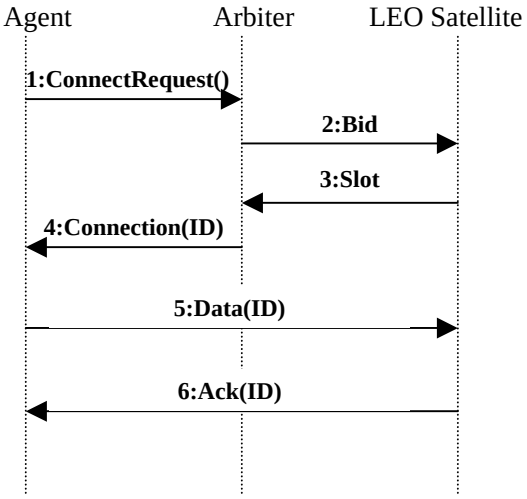


图 2-8、顺序图

●协作图

如图 2-9 所示，协作图主要描述对象间的关系，用顺序号决定消息的顺序，协作图强调系统内部结构。协作图是描述类型间的交互活动的建模元素，由两部分组成，对象静态结构的描述，如：对象的关系、属性、操作、又称为上下文 context；执行行为的描述；这两个方面构成了完整的行为规格。

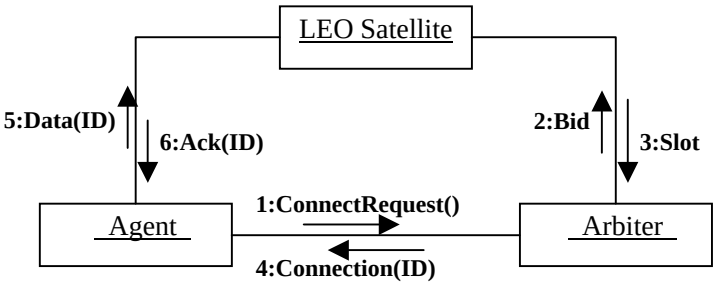


图 2-9、协作图

2．1．4 状态图

UML 提供状态图来刻画系统的动态特性。如图 2-10 所示，状态图是规定对象在其生命周期内对事件的响应所经历的状态序列，以及对象的响应和动作。状态图包括**状态**和**状态转换**，描述系统状态在外界事件作用下的变迁，**状态**可以是单个状态，也可以是复合状态；**状态转换**是关系的一个子类型，指明从一个状态到另一个状态的通路。**状态转换**是由事件激发的，通过动作来完成。UML 状态图在传统状态机的基础上，作了进一步的扩展，其描述能力相当的强。具体表现在：

●转换卫士（Guard on transition）：状态转换必须满足的条件；

●转换转播(Propagated transition)：一个事件激活某个状态转换的同时产生另外一个事件；

●进入状态的动作(Action on Entry state)：进入某个状态时，引发的动作；

●退出状态的动作(Action on Exit state)：退出状态时，引发的动作；

●嵌套状态：UML 支持复合状态，允许状态的分解；

●并发。

另外，UML 还提供一种特殊的状态图，称为活动图。在状态图中，状态可以是简单状态、伪状态、复合状态、动作状态。在活动图中所有的状态都是动作状态。动作状态只由操作的完成而引发，与外部事件无关。

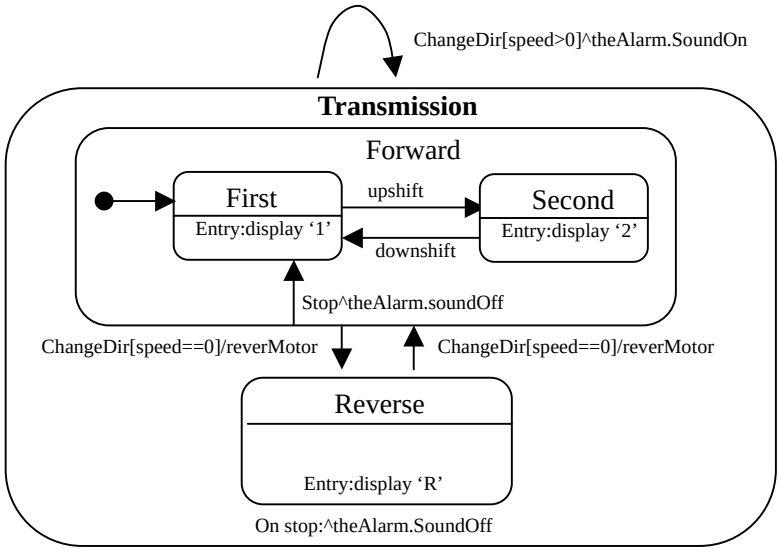


图 2-10、状态图（机器人）

2．1．5 实现图

实现图包括组件图（图 2-11）和配置图（图 2-12），展示系统的源代码的结构和运行时刻的实现结构。

2．1．5．1 组件图（Componet Diagram）

组件通常为代码块、二进制代码块和可执行部件等，是对建模元素物理实现的抽象。组件图是指用依赖关系链接起来的组件集合，可以描述与特定语言相关的编译时刻的依赖关系。

2．1．5．2 配置图（Deployment Diagram）

配置图主要表现运行时刻各个处理元素的配置以及位于其上的组件、进程(主动对象的特例)、对象。配置图是由通信链接起来的节点图，节点代表具有独立的内存和处理能力的计算资源。每个节点可以包含组件实例，也可以包含若干个对象。

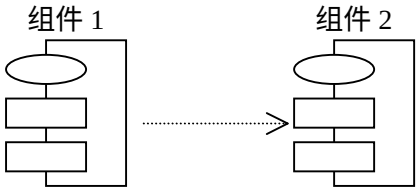


图 2-11、组件图

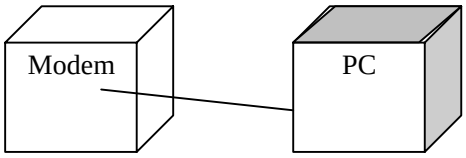


图 2-12、配置图

2．2 Rose2000 的简介

Rational 公司的 Rose 2000 是 UML 建模环境，提供了 UML 的所有建模元素的可视化编辑环境、基于组件的开发以及对软件开发全过程的支持。除了提供 UML 的类图、交互图、状态图、组件图和配置图支持外，还与工业标准的软件开发环境 Microsoft Visual Studio 6.0 等开发工具融合在一起，从而全面支持面向对象分析、设计和编码。在面向对象分析时，主要使用 Use-Case 图来刻画软件系统的需求和功能说明；在设计阶段利用类图来描述系统的静态结构，用交互图和状态图来描述系统的动态行为，同时还使用组件图、配置图来描述系统的源代码结构、运行配置；在分析设计完成之后，从模型中提取程序代码框架，利用第三方程序设计环境来实现该软件系统。另外通过从源代码中提取接口、类声明信息来获取源码中的模型信息，即对逆向软件工程的支持。目前，Rational Rose 已经被广泛地应用于通信、金融、企业信息系统分析和设计。例如，美国信用卡公司 Capital One 公司使用它提取原来信用卡信息系统的建模信息；阿尔卡特、贝尔等公司利用 Rational Rose 分析和设计通信协议等。

2．3 UML 及其 Rational Rose2000 的特点

2．3．1 UML 及其 Rational Rose2000 的优点

支持可视化：主要使用图符来描述软件系统分析和设计产生的文档，具有极强的表达能力和可理解性，有利于开发人员与用户之间、开发人员之间的交流。

可扩展性：允许开发人员根据自己开发的需要，使用版型对语言进行扩充，以适合特定领域系统的开发需要。

支持形式化描述：使用 OCL（Object Constraint Language）描述模型语义，减少理解的二义性。

组件建模：Rose 支持二进制组件（COM、Active）建模，可以很方便地取出组件的接口信息，有利于提取组件中的建模信息，另外提供 Drag&Drop，可以方便地将含有组件的文件加入 Rose 来获取组件的接口信息。

多语言支持：Rose2000 与语言无关，具体支持 C++、Java、Visual Basic、Ada、PowerBuilder 、Smalltalk 和 Oracle8 等多种语言。

开发组的支持：支持整个开发组成员协同完成一个系统的分析、设计。

扩展的支持：通过提供定制 Rose 菜单、Rose 脚本语言、Rose 自动化和 Add-In 管理等方法，以便用户扩充 Rose，定制自己的可视化建模环境。

2．3．2 UML 及其 Rose2000 的不足

第一：UML 被称作建模语言，而不是一种方法。因为从原则上讲，大部分方法是由一种建模语言和一种过程共同组成的。其中建模语言是一种(以图形方式为主的)表示符号，用来表达人们的设计；过程则是对进行这种设计应采取哪些步骤所提出的建议。UML 不包含任何过程指导。就是说，它并不讲述如何运用面向对象的概念与原则去进行系统建模，而只是定义了用于建模的各种元素，以及由这些元素所构成的各种图的构成规则。在文学领域，一本词典加一本语法手册，并不能构成一种文学理论或创作方法，也不能使大部分学习者学会如何进行文学创作。同样，在软件领域，仅有一种建模语言不能算作一种建模方法，也不能为工程技术人员提供一套可遵循的开发准则。

第二：语言的统一历来是困难的。在人类社会中多种多样的自然语言孕育了多姿多彩的人类文化，也为不同国家、地区和民族之间的人际交流带来了不便。可是没有哪种语言能够取代其它所有的语言而在全世界成为一种统一的语言。在计算机领域，用于编程、需求定义、数据库、图形、人工智能等不同用途和不同分支领域的语言也都是多种多样的，技术人员和用户早已感受到语言的多样化所带来的不便。但是迄今生存下来的每一种语言，都有其存在的理由。没有哪个语言能够取代其它语言。系统建模语言与编程语言相比情况有什么根本不同？使编程语言保持多样化局面的各种因素对于建模语言是否都不存在？编程语言未曾具备的统一条件建模语言是否已经具备？对这些问题，至少到目前还没有令人信服的解答。尽管 UML 已作为一种标准建模语言被 OMG 采纳，但并不意味着它必然能取代其它所有的 OO 建模语言。标准从来不是“天无二日”的帝王宝座。

第三： UML 自身作为一种建模语言，能否被广泛采用，甚至使许多采用不同 OO 方法

的用户转向 UML，关键问题在于采用这种建模语言所带来的好处与付出的代价的对比。从功能(表达能力)来看，UML 无疑是很强的，它从参加联合的各种 OO 方法中吸取了许多扩充的概念和图形表示法。但恰恰因为这一点，UML 的复杂性也远远超出了以往各种 OO 方法，以至它所使用的建模元素达到一百多个，开发一个应用系统要画的图达 9 种之多。这意味着，无论是学习它还是在工程中使用它，都要付出比以往高得多的代价。如果说这么多建模元素和图对于清晰、准确地表达大部分应用系统的建模问题是必不可少的，那么为此付出较高的代价是值得的。但是实际情况并非如此，否则就无法解释，为何以往许多开发者运用相对简单的 OOA/OOD 方法也能成功地开发许多应用系统。

UML 没有指明它要求建立的 9 种图有哪些是可以缺省的，以及在什么情况下，对哪类系统可以缺省。在建模时全面地建立这些图将使工程文档的种类和数量远远超过以往任何一种 OOA/OOD 方法，甚至达数倍之多。实际上 UML 的各种图对建模问题的表达能力有很大的冗余性。

第三章、软件需求工程概述

3．1 引言

需求工程是随着计算机的发展而发展的。在计算机发展的初期，软件规模不大，软件开发所关注的是代码编写，需求分析很少受到重视。后来软件开发引入了生命周期的概念，需求分析成为其第一阶段。随着软件系统规模的扩大，需求分析与定义在整个软件开发与维护过程中越来越重要，直接关系到软件的成功与否。人们逐渐认识到需求分析活动不再仅限于软件开发的最初阶段，它贯穿于系统开发的整个生命周期。80 年代中期，形成了软件工程的子领域—需求工程（requirement engineering，RE）。进入 90 年代以来，需求工程成为研究的热点之一。

3．2 软件需求基本内容

3．2．1 需求的定义

IEEE 为需求作如下定义：

（1） 用户解决问题或达到系统目标所需要的条件；

（2） 为满足一个协约、标准、规格或其他正式制定的文档，系统或系统构件所需要满足和具有的条件或能力；

（3） 对上述条件的文档化的描述。

3．2．2 需求的层次

软件需求包括三个不同的层次——业务需求、用户需求和功能需求——也包括非功能需求。不同层次是从不同角度与不同程度反映着细节问题。业务需求（business requirement）反映了组织机构或客户对系统、产品高层次的目标要求，它们在项目视图与范围文档中予以说明。用户需求（user requirement） 文档描述了用户使用产品必须要完成的任务，这在使用案例（use case）文档或方案脚本（scenario）说明中予以说明。功能需求（functional requirement）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。所谓特性（feature）是指逻辑上相关的功能需求的集合，给用户提供处理能力并满足业务需求。软件需求各组成部分之间的关系如图 3-1 所示。

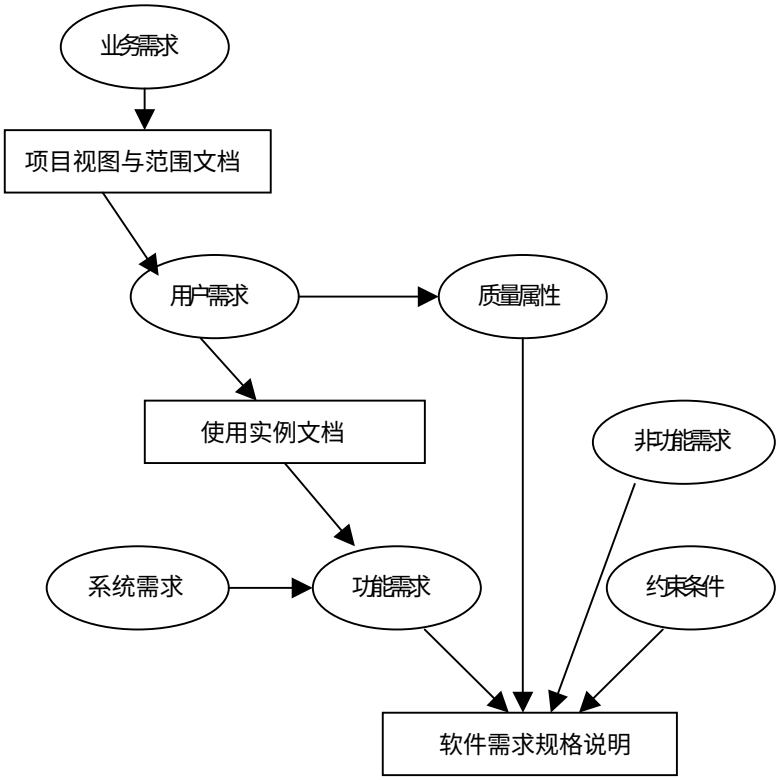


图 3-1 软件需求各组成部分之间的关系

3．2．3 需求工程的概念

需求工程是指应用已证实有效的技术、方法进行需求分析，确定客户需求，帮助分析人员理解问题并定义目标系统的所有外部特征的一门学科 ;它通过合适的工具和记号系统描述待开发系统及其行为特征和相关约束，形成需求文档，并对用户不断变化的需求演进给予支持。需求工程可分为系统需求工程（如果是针对由软硬件共同组成的整个系统）和软件需求工程（如果仅是专门针对纯软件部分）。软件需求工程是一门分析并记录软件需求的学科，它把系统需求分解成一些主要的子系统和任务，把这些子系统或任务分配给软件，并通过一系列重复的分析、设计、比较研究、原型开发过程把这些系统需求转换成软件的需求描述和一些性能参数。本论文以下如无特别说明，需求工程主要是指软件需求工程。

3．2．4 需求工程的阶段

需求工程是一个不断反复的需求定义、文档记录、需求演进的过程，并最终在验证的基础上冻结需求。80 年代，HerbKrasner 定义了需求工程的五阶段生命周期：需求定义和分析；需求决策；形成需求规格；需求实现与验证；需求演进管理。近来，MatthiasJarke 和 KlausPohl 提出了三阶段周期的说法：获取、表示和验证。综合这几种观点，可以把需求工程的活动划分为以下 5 个独立的阶段：

（1） 需求获取：通过与用户的交流，对现有系统的观察及对任务进行分析，从而开发、捕获和修订用户的需求；

（2） 需求建模：为最终用户所看到的系统建立一个概念模型，作为对需求的抽象描述，并尽可能多的捕获现实世界的语义；

（3） 形成需求规格：生成需求模型构件的精确的形式化的描述，作为用户和开发者之间的一个协约；

（4） 需求验证：以需求规格说明为输入，通过符号执行、模拟或快速原型等途径，分析需求规格的正确性和可行性；

（5） 需求管理：支持系统的需求演进，如需求变化和可跟踪性问题。

3．2．5 需求规格

规格就是一个预期的或已存在的计算机系统的表示。它可以作为开发者和用户之间协议的基础，来产生预期的系统。规格定义系统所有必须具备的特性，同时留下很多特性不做限制。通常，我们要求规格比组成特定系统的实际的软件和硬件更简洁、更全面、更易于修改。

需求工程的主要结果是软件需求规格（software requirement specification，SRS），SRS 是对外部行为和系统环境（软件、硬件、通信端口和人）接口的简洁完整的描述性文档。项目管理者用它来对项目进行计划和管理，在许多情况下，它也被作为是用户的使用手册或帮助用户理解系统的文档。它广泛地适用于对各类应用领域中的客户问题进行理解与描述，实现用户、分析员和设计人员之间的通信，为软件设计提供基础，并支持系统的需求验证和演进。

3．2．5．1 需求规格基本内容

SRS 的基本内容包括行为需求和非行为需求。行为需求定义系统需要“做什么”，描述

系统输入输出的映射及其关联信息。完整地刻画系统功能，是整个软件需求的核心。非行为需求定义系统的属性，描述和行为无关的目标系统特性。包括系统的性能、有效性、可靠性、安全性、易维护性、可见性和顺应性，好的 SRS 应具有如下特点：正确性、无二义性、完整性、一致性、可验证性、可修改性、可跟踪性、易理解性以及有好的注解等。

3．2．5．2 需求规格基本描述方法

现有的需求规格描述方法有 3 种：形式化方法、非形式化方法和半形式化方法。

形式化方法是具有严格数学基础的描述系统特征的方法。形式化方法具有准确、无二义性的特点，有助于验证有效性和完整性。

非形式化方法使用未作任何限制的自然语言，易于理解和使用。但它具有固有的二义性，且难以保证其正确性、可维护性，难以用计算机系统提供自动化的支持。

半形式化方法介于上述两者之间，在宏观上对语言和语义有较精确的描述，而在某些局部方面则允许使用非形式化的自然语言。

3．2．5．3 需求规格的技术支持

需求工程研究的核心是关于需求规格描述方法和技术的研究，它致力于寻求以下几点支持：1、需求规格的表示、获取机制；2、需求规格文档制作及品质保证机制；3、需求规格的演示验证机制。需求规格技术支持工具主要有：

有限状态机(FSM)、决策树和决策表、伪代码程序设计语言(PDL)、状态图(Statecharts)、需求工程验证系统(REVS)、(实时)结构化析(SA/RT)方法、Petri 网、需求描述语言(SDL)以及需求语言处理器(RLP)等。

3．3 需求工程活动

需求工程的活动可分为 3 个层次：方法学——即整体的、全面的、指导性的方法；方法——具体的、详细的实现途径和策略；工具——指一步步形成的手工或自动的辅助过程。下面分别来讨论这 3 个层次的应用及发展。

3．3．1 需求工程方法学

需求工程方法学包括大家所熟悉的软件工程的生命周期模型，如瀑布型、渐增型、快速原型及螺线型。另外还有 Lano 所提出的操作概念规格，在需求产生前由开发者写成，既能满足开发者希望需求规格精确的要求，又能满足用户希望其易于理解的要求。Howes 还提出用用户手册的方法来解决用户和开发者之间的通信问题。Sutcliffe，Maiden 等人提出从领域

知识的角度出发在需求工程的环境中定义通用的领域语义模型和组合模型。Alford 提出的“任务分割”的概念，大大降低了需求分析的问题复杂度。Chou 和 Eckert 的文章讨论了面向对象的需求工程方法学的概念和模型。Drake 提出用于确定系统需求边界的限定过程。Gotel 在他的文章中对需求跟踪性问题进行了阐述。Rosca，Krogsite，Jaffe，Zave，Robinson，Basili，Yu，Mathalone 等人的文章也分别从不同方面对需求工程方法学进行了论述。

3．3．2 需求工程方法

需求分析方法可大致分为 4 类，如图 3-2 所示。

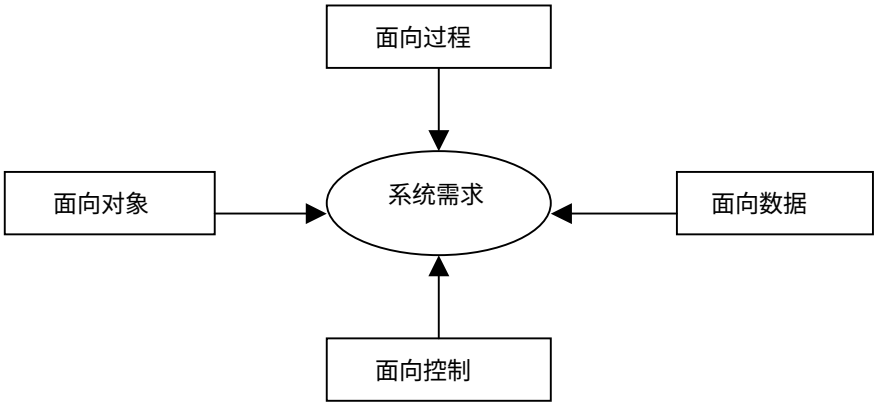


图 3-1　需求分析方法分类图

面向过程的分析方法的主要研究系统输入输出转化的方式，对数据本身及控制方面并不很重视。传统的结构分析方法 SA （structure analysis），SADT（structure analysis and design technique）就属于这一范畴；另外还有可执行/操作模型如 PAISley 和 Descartes；以及形式方法如 VDM（Vienna design method）和 Z。

面向数据的方法强调以数据结构的形式描述和分析系统状态。JSD 和关系实体（ER）模型都是面向数据的。

面向控制的方法强调同步、死锁、互斥、并发以及进程激活和挂起。数据流图就是典型的面向控制的方法。SADT 是以面向控制的方法为辅的。

面向对象 OO 的方法把分析建立在系统对象以及对象间交互的基础之上，使得我们能以 3 个最基本的方法框架—对象及其属性、分类结构和集合结构来定义和沟通需求。面向对象的问题分析模型从 3 个侧面进行描述，即对象模型（对象的静态结构）、动态模型（对象相

互作用的顺序）和功能模型（数据变换及功能依存关系）。面向对象的方法正在成为需求分析中的一个热点，并展现出良好的应用前景。面向对象的方法产生了一些流派，如 Yourdan 和 Coad 的 OOA 方法、Booch 的方法、Jacobson 的 OOSE、Rumbaugh 的 OMT 方法等。

3．3．3 需求工程工具

随着需求工程方法研究的不断成熟，计算机技术的迅速发展，需求工程工具的发展也产生了巨大飞跃。它们辅助需求的捕获、管理及需求文档的生成过程，并对需求工程的自动化提供支持。

3．3．3．1 基于操作方法的需求工程工具

典型的操作方法的工具主要有 GIST、PAISLey、STATEMATE、JSD。此类工具的共同特点即它们最终结果是严格的形式化需求规格，对快速原型提供支持，需求能得到及时的验证和反馈。可执行规格和原型技术无疑为软件工程提供了很好的实现途径。上述几种方法又各有侧重，如 PAISLey 和 STATEMATE 主要适于嵌入式实时系统，GIST 在人工智能和数据库系统的应用比较广泛。

3．3．3．2 基于知识的需求工程工具

典型的基于知识的工具主要有 RA（Requirement Apprentice）、TMMRP （Technology to Manage Multiple Requirement PersPective）、QRACC（Quality Attribute Risk and Conflict Consultant）、PROMIS（Prototyping MIS）。上述几种工具的共同点即它们均把人工智能技术应用于需求工程领域，具有一个知识库和一个推理机制，在此基础上进行需求分析、检测等活动。人工智能中知识表示和知识获取、定向推理等方法对于领域建模、问题理解和需求获取的研究是有重要意义的。这也是目前需求工程很有特色并且很有前景的一个研究方向。

3．3．3．3 面向对象的需求工程工具

此类工具的代表目前主要是指支持 UML 语言的工具，关于 UML 语言已经在前述章节做了介绍。归结来说，用面向对象的思想进行需求分析，其根本要点在于，利用对象的概念模型建立一个针对于问题域的模型，用户和软件工程师通过该模型进行交流，在此模型基础上形成需求规格说明书。OO 建模的好处还在于它不用在分析和设计之间划一道鸿沟，设计只需要在分析的基础上进一步根据实现系统的限制不断进行各种成分的扩充和细化。而且具有模型稳定性、可重用等等优点，这将大大降低软件维护和升级的成本。

UML 可以说是代表了面向对象方法的软件开发技术的发展方向，具有巨大的市场前景，

也具有重大的经济价值。因此在课题《基于 Web 的远程教学系统应用平台》的需求分析中，我们选用 Rational 公司的可视化建模工具 ROSE，应用 UML 的思想和方法进行建模。面向对象的需求工具还有 Bucci 等人的 TROL，Tkach 等人开发的 VMT，以及北大的 JB 工程等。

3．3．3．4 需求跟踪工具

典型的需求跟踪工具有 ARTS（The Automated Requirement Traceability System）。ARTS 是 LMSC 公司为软件工程师开发的需求跟踪性系统。它以“需求树”的形式链接用户定义的需求，并以这种分层结构提供对可跟踪性的支持。其首要解决的问题是选择一个数据库管理系统并加入可跟踪性构件。ART 是一个数据库管理系统，在这个系统中需求是作为数据库中的记录被定义的，这些记录是由包含和需求对象有关的信息的域和属性组成。数据库系统能对需求进行选择、存储和操作。ART 的开发方法是建立在快速原型和渐增模型二者结合的基础之上的。

需求管理，尤其是需求的可跟踪性问题一直是开发大型系统的主要问题之一。需求跟踪性问题是当前需求工程的研究核心问题之一，它把需求、设计和执行相互联系起来。能有效地验证需求规格，检测错误，管理需求演进，是保证系统成功的关键点之一。当前的需求跟踪工具还有 Remap 系统，RETH 和 Radix 工具等。

3．3．3．5 其它

还有一些有代表性的工具，如最先为结构化分析提供计算机辅助手段的 PSL/PSA；支持 SREM 方法的工具 REVS 和 RSL；能有效描述需求接口和任务间相互关系的 N2 图；突破传统的 VonNeumann 思想束缚的过程无关的第四代语言 4GL；Golidin 等开发的协助需求引出的工具 AbstFinder；可视化需求分析语言 VRDL 等等。

尽管工具和技术近 20 年内有了质的飞跃，它们还是存在局限性的。目前的需求工具之间及系统间数据的传输是非常困难的，这些工具并不支持从任意级抽象的层次观察需求信息。大多工具都是为单用户设计的，对于目前较为普遍大型的协同开发环境的支持很少。

目前需求工程的方法和工具的主要不足之处在于它们与实际应用领域尚有很大差距，未能充分重视影响需求、开发及演进的目标系统开发环境。需求工具和方法的开发方向应致力于缩短与实际应用领域的差距。

3．4 需求工程目前的一些问题及其讨论

需求工程的研究到今天已相对成熟了，能作为一个独立的领域出现，有大量优秀的方法和工具对它提供支持。但需求工程的发展不仅只受技术因素的影响，还受一些重要的社会因

素的影响，因而不可避免的存在着一些问题。

（1）用户和开发者的协同

需求工程应该是一种协同、契约型作业，用户和开发部门一同达到的一个精确、无二义的协议声明。软件发展近十年来的发展趋势——系统小型化，软件生命周期缩短、通用构件及软件体系结构的重用——使大多软件开发者忽视了这一要求。

（2）对市场驱动的软件的支持

现在开发的大量软件的需求不是从用户角度出发，而是基于市场驱动的。需求工程活动通常是在对具体领域问题的观察得到基本解决方案形成后才进行的，其内容需涉及产品规划和市场分析。传统的需求工程对这些问题的支持很少。

（3）需求优先级分配问题

竞争使软件商要限制软件产品某些功能来加速开发进程以缩短投入市场的时间。某些非关键性需求的修改会简化目标系统的设计与实现，开发者应区分需求的优先级，在理想目标系统和需求实现的系统功能之间作适当取舍。目前，在需求的优先级分配及需求集的选取机制两方面，RE 进展缓慢。

（4）需求不完备性问题的处理

80 年代软件开发模型发生很大转变，原因之一是人们认识到一开始就对需求和实现做出所有正确的决定几乎是不可能的。问题的关键在于为需求完整性确定合适的边界条件，决定能接受的不完备性的种类和程度，留出一些需求有待开发阶段完善。

（5）对设计产品的重用

开发者需能较快地描述问题及求解限制，因此需要一些评估选择策略和需求技术，使需求工程能捕获和操纵设计级的产品（如通用构件）。但目前具体支持很少。需求跟踪技术正迅速增涨的研究兴趣及活动，或许会对此提供一些支持。

（6）需求分析方法和工具的支持

考虑到确定需求和确立系统的开发环境的广泛性，目前能对整个需求分析进程进行全面描述的方法和大型工具是很少的。开发者应能把需求分析过程划分成若干子问题，合理利用现有的通用工具，为具体的子需求提供自动化的支持。

3．5 小结

从需求工程的过程各阶段的作用来看，日后研究的重点将放在需求分析、建模和可跟踪性问题的研究。当前软件开发中的一些热点技术，如面向对象技术、自动化工具、构件技术，

以及传统的形式化技术、领域建模技术的发展，仍将继续为需求工程的研究提供有力的支持。需求工程研究现状中另一明显的不足是理论解决方案通常是在对实际问题简化的基础上得到的，理论研究和实践脱节。要获取需求突破，改善需求工程的开发效率和质量，很重要一点是探索有效的解决途径，缩小理论和应用之间的正在增长的差距，使开发出的系统和模型切实满足应用领域的需要。我们期待着需求工程领域的研究在与丰富的计算机科学实践经验结合的过程中，提炼出更多成熟、完善的方法和工具，从而推动整个需求工程的进程。

第四章、软件需求分析模型

由前述章节可以看出，需求工程发展到今天，仍然没能很好地解决用户和开发人员之间的协同和共识以及需求变动的问题。而 UML 主要使用图符来描述软件系统分析和设计产生的文档，具有极强的表达能力和可理解性，这对开发人员与用户之间、开发人员之间的交流非常有利，但 UML 并没有给出统一的软件开发过程。探索如何应用 UML 指导软件开发，是一件非常有意义的工作。

“基于 Web 的远程教学系统”是本校校园网建设的一个项目，本项目是为了充分利用校园网资源，补益我校传统教学方式，在先期实现网络化教学最终实现远程教育的思想理念指导下开始建设的。在本项目立项阶段与建设之初，开发小组进行了充分的调查研究，借鉴国内外远程教学的较成功的解决方案，最终结合我校的实际情况和基础条件，确定了系统需求和技术路线。

教学系统大而庞杂，各个环节错综相连，而远程教学几乎等同于把整个教学系统迁移到网络，这就使得分析和设计工作非常繁重。而需求分析是软件开发中最关键的一步，这就要求对系统的需求分析一方面要尽可能详细，不要漏掉重大问题，以免后期结构性的变动；另一方面分析，设计和开发过程不仅要满足现时需求， 还要能适应需求的变化，即具备一定的应变能力。也就是常说的易读，易修改， 易扩充。经过对比和摸索，本开发小组选择了 UML 和 Rose 作为分析和设计的规范和工具。

在分析阶段，本人采用了将软件需求分析过程划分为若干个子过程的方式，由高层功能开始逐步细化，综合分析得到系统的功能框架与需求规格说明。归结起来，可以展示为图 4-1 所示模型。事实证明，采用此模型指导需求分析，不仅改善了需求的开发效率和质量，还为软件的设计和实现奠定了良好的基础。

在设计和实现阶段，我们也尝试应用 UML 和 Rose，有一定收效，这种尝试需要继续、需要改进，目前还不能从理论上进行归结。UML 虽然没有给出过程，但它规定所有的过程都要用例驱动、体系结构中心和过程迭代，项目开发中，我们始终遵循了这一原则。

此项目的展开吸收了面向对象的软件工程理论的优越性而没有生硬地套用面向对象软件开发方法的固定模式，无论是在分析、设计还是在实现阶段，我们都坚持具体项目具体分析。本项目开发小组的每个成员都有一个共识，那就是每个项目都有它解决问题的针对性，

因而都有它的特殊性，我们争取在所针对的这个领域所做的开发有它的普适性。我们遵循了软件工程的规范和 UML 规范，同时遵循了我国所颁布的软件开发规范，使得本项目具有良好的可传递性。

下面结合本项目的开发对需求分析模型中的各步进行详细介绍。

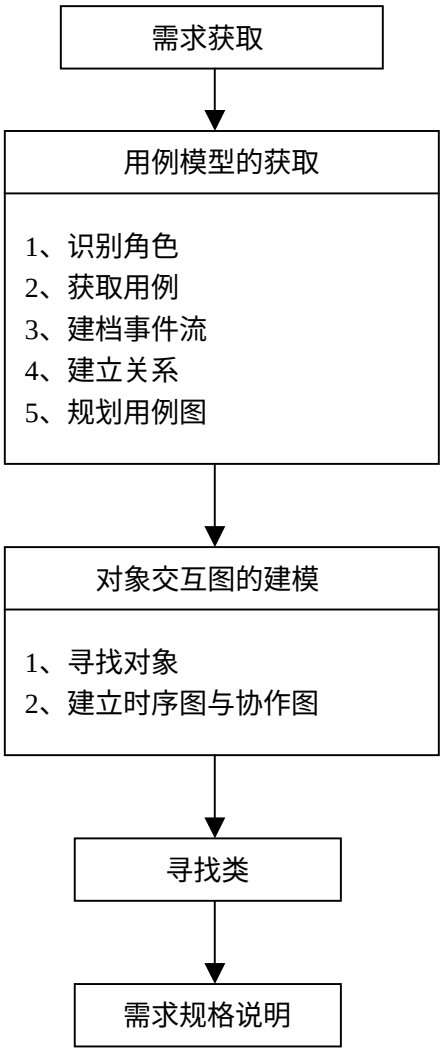


图 4-1 软件需求分析模型

4．1 需求获取

这一步是指对系统需求的识别过程。第三章讨论了需求的三个层次：业务，用户和功能。在项目中它们在不同的时间来自不同的来源，也有着不同的目标和对象，并需以不同的方式编写成文档。业务需求（或产品视图和范围）不应包括用户需求（或用例），而所有的功能需求都应该源于用户需求。同时也需要获取非功能需求，如质量属性。在本项目中采用了以

下需求获取方式：

（1）将用户群分类并归纳各自特点。

为避免出现疏忽某一用户群需求的情况，要将可能使用系统的客户分成不同组别。他们可能在使用频率、使用特性、优先等级或熟练程度等方面都有所差异。详细描述出它们的个性特点及任务状况，将有助于系统设计。本项目中我们将用户分为学生、教师和管理人员，并对他们需要依赖本系统完成的学、教、管理功能进行了归纳。

（2）让用户代表确定用例。

从用户代表处收集他们使用软件完成所需任务的描述——用例，讨论用户与系统间的交互方式和对话要求。在用例基础上可得到功能需求。本项目中我们在收集时将学生和教师的描述编写成文档。

（3）确定质量属性和其它非功能需求。

在功能需求之外还需要再考虑一下非功能的质量特点，这些特点包括性能、有效性、可靠性、使用性等。比如我们考虑教学系统用户量较大的特点，除了注意硬件系统的可扩展性和性能设计，也非常注意软件系统的安全性和可靠性设计；考虑到系统运行于 CERNET 乃至国际互联网的环境，软件开发遵循开放系统的原则，做到与平台无关，同时遵循公共的国际标准，以方便今后的升级维护；考虑到大批量数据的吞吐，采用大型商业数据库系统，使整个系统管理规范化，数据的完整性、安全性得到保障。

（4）通过考查当前系统的运转来进一步完善需求。

通过考查当前系统，有利于获取用户没有意识到或者描述不清楚的需求，也有利于认识用户所提出的当前系统所不能提供的需求。我们通过总结自身接受课堂教育的体验及教师当前的办公、授课与教学管理机制的考查，进一步完善了系统需求。

4．2 用例模型的获取

用例模型描述的是外部执行者(Actor)所理解的系统功能。用例模型用于需求分析阶段，它的建立是系统开发者和用户反复讨论的结果，表明了开发者和用户对需求规格达成的共识。首先，它描述了待开发系统的功能需求；其次，它将系统看作黑盒，从外部执行者的角度来理解系统；第三，它驱动了需求分析之后各阶段的开发工作，不仅在开发过程中保证了系统所有功能的实现，而且被用于验证和检测所开发的系统，从而影响到开发工作的各个阶段和 UML 的各个模型。在 UML 中，一个用例模型由若干个用例图描述，用例图主要元素是用例和执行者。

4．2．1 识别角色

获取用例首先要找出系统的角色。角色是与所建系统交互的人与物。角色描述系统范围外的一切。

角色有三大类：系统用户、与所建系统交互的其他系统和时间。

第一种角色是实际的人，即用户，是最常用的角色，几乎每个系统都有。命名这些角色时，按作用命名而不是按位置命名。一个人可能有许多作用。利用作用名而不是位置名就可以得到更稳定的角色图形。位置名随时改变，而作用和责任从一个位置移到另一个位置。利用作用命名，就不必在每次增加新位置或改变位置时更新模型。

第二种角色是另一个系统。例如，假设银行有个信息系统，用于维护每个客户的信用帐号信息。ATM 系统应与这个信用系统交互。这时，信用系统就成为角色。

第三种常用角色是时间。时间作为角色时，经过一定时间触发系统中的某个事件。例如，ATM 系统可能每天午夜运行一些协调处理。由于时间不在我们的控制之内，因此是个角色。

角色中有一种是没有实例的，称为抽象角色。例如，你可能有几种角色：计时工、合同工和临时工。所有这些都属于第四种角色——员工。但公司中没有一个人是单纯的员工，每个人都是计时工、合同工和临时工中的一种。员工角色只是显示计时工、合同工和临时工的一些共性。员工角色没有实例，是个抽象角色。

可以通过用户回答一些问题的答案来识别角色。以下问题可供参考：

· 谁使用系统的主要功能(主要使用者)。

· 谁需要系统支持他们的日常工作。

· 谁来维护、管理使系统正常工作(辅助使用者)。

· 系统需要操纵哪些硬件。

· 系统需要与哪些其它系统交互，包含其它计算机系统和其它应用程序。

· 对系统产生的结果感兴趣的人或事物。

通过回答以上问题，可以识别出远程教学系统的角色有三个：学生、教师、管理员。

4．2．2 获取用例

用例（Use Case）是系统提供的功能块。换句话说，用例演示了人们如何使用系统。

通过用例观察系统，能够将系统实现与系统目标分开，有助于了解最重要的部分——满

足用户要求和期望，而不会沉浸于实现细节。通过用例，客户可以看到系统提供的功能，先确定系统范围再深入开展项目工作。

用例采用与传统方法不同的方法。将项目分解成用例是个面向对象过程而不是面向实现的系统，因此不同于传统的功能分解法。尽管功能分解法关注如何分解成系统能处理的小块，但用例首先关注用户对系统的需求。

如何寻找用例呢？一个好办法是检查客户提供的文档。通常，高级范围和观点文档有助于确定用例。还可以考虑项目的保管人，询问每个保管人要什么系统功能。对每个保管人，问下列问题：

· 这个保管人要用这个系统干什么？

· 保管人是否需要维护任何信息（生成、读取、更新、删除）？

· 保管人是否要把外部事件告诉系统？

· 系统是否要把某些改变告诉保管人？

还有一些针对整个系统的问题：

· 系统需要何种输入输出?输入从何处来?输出到何处?

· 当前运行系统(也许是一些手工操作而不是计算机系统)的主要问题?

需要注意，最后两个问题并不是指没有角色也可以有用例，只是获取用例时尚不知道角色是什么。

用例中也有一种不是由角色直接启动的，而是提供一些其他功能，让其他用例使用，称为抽象用例。抽象用例是参与使用或扩展关系的用例。

用例是独立于实现的用户需求高级视图。以下单独检查这个定义的每一部分：

第一，用例独立于实现。定义用例时，假设正在建立用户手册系统。用例可能用 Java、C++、Visual Basic 建立。用例关注系统的作用而不是如何实现这个作用。

第二，用例是系统的高级视图。如果系统有 3000 个用例，则失去了简单性的好处。用例的集合应让用户易于了解高层的整个系统，有这么多用例，客户就会在一页页文档间犹豫不决。确定用例的过程是对获取的用例进行提炼和归纳的过程。一般用户的用例个数为 20 到 50。可以用使用与扩展等关系将用例进行必要的分解，也可以将用例组合成组，以便于组织。

第三，用例关注使用系统的用户。每个用例应表示用户与系统间的完整事物，为用户提供一定价值。用例应按业务术语命名，而不是按技术术语命名，应让用户一目了然。用例通常用动词或短语命名，描述用户看到的最终结果。用户不关心你要

面对多个其他系统、要完成哪些具体步骤或者需要几行代码，只关心是否满足了他的要求。这里，关注用户对系统的需求，而不是达到结果所需的步骤。

这样，有了用例的最后清单时，怎么知道是否完整呢？可以回答以下问题：

· 每个功能要求是否至少在一个用例中？如果要求不在用例中，则不会实现。

· 是否考虑了每个保管者如何使用系统？

· 每个保管人向系统提供什么信息？

· 每个保管人从系统接收什么信息？

· 是否考虑了维护问题？要有人启动和关闭系统。

· 是否标识了系统要交互的所有外部系统？

· 每个外部系统从系统接收什么信息、向系统发送什么信息？

在“基于 Web 的远程教学系统”中，分别对三个角色获取用例，如下所述。

学生：学习、交流、维护个人信息。

学习包括：课件学习、实时课堂、在线考试、在线作业。

课件学习包括：浏览课件、下载课件、定制课件。

实时课堂包括：提问、共享白板、听课。

在线考试包括：选择试卷、答卷、查分。

在线作业包括：给出答案、查询答案、提交作业、查看成绩。

交流包括：收发邮件、参加 BBS、在线聊天、浏览新闻。

收发邮件包括：准备原稿、收邮件、发邮件、删除邮件、管理地址簿。

参加 BBS 包括：发帖、回帖、按主题搜索。

在线聊天包括：读信息、发信息、查看在线用户。

浏览新闻包括：浏览主题、浏览内容、按主题搜索。

维护个人信息：注册、修改个人信息。

教师：办公、交流、维护个人信息。

办公包括：实时授课、管理课件、管理试题库、管理作业、管理学生。

实时授课包括：课堂管理、播放课件、同学生交流、板书。

管理课件包括：上传课件、修改课件、删除课件。

管理试题库包括：添加试题、删除试题、组卷、试卷改组。

管理作业包括：发布作业、删除作业、查阅学生答案。

管理学生包括：管理班级、查阅学生信息、评价学生、添加学生记录。

交流包括：收发邮件、参加 BBS、在线聊天、浏览新闻、发布公告。

发布公告包括：发布公告、删除公告。

其余的交流用例如同学生。

维护个人信息：修改个人信息。

管理员：管理、交流。

管理：管理学习、管理用户、管理交流。

管理学习包括：添加课件、删除课件、课件分类、试题库分类。

管理用户包括：维护学生信息、维护教师信息。

管理交流包括：管理邮件系统、管理聊天室系统、管理新闻公告系统、管理论坛。

交流：如同学生

以上为分别对是三个角色获取的用例。另外，考虑到系统的易管理性与安全性的要求，我们要求这三个角色在使用本系统时首先通过身份认证，也就是说，用户首先要经过注册成为合法用户。所以，又获取了三个系统用例，即学生身份验证、教师身份验证和管理员身份验证。归结起来，就是本项目在需求阶段获取的全部用例。

4．2．3 建档事件流

用例开始描述系统的作用。要实际建立系统，则需要更具体的细节。这些细节写在事件流文档中。事件流的目的是建档用例中的逻辑流程。这个文档详细描述系统用户的工作和系统本身的工作。

尽管事件流很详细，但仍然是独立于实现方法的。可以在编写事件流时假设有一个自动化系统，但还不应考虑系统到底用 C++ 、PowerBuilder 还是 Java 建立。这里的目的是描述系统干什么，而不是怎么干。事件流通常包括：

· 简要说明

· 前提条件

· 主事件流

· 其他事件流

· 事后条件

简要说明

每个用例应有一个相关的说明，描述该用例的作用。说明要简单扼要，但应包括执行用

例的不同类型用户和用户通过这个用例要达到的最终结果。随着项目的进行，这些用例定义有助于整个小组记住项目中每个案例的意义和作用，还有助于建立目的明确的用例，减少小组成员的混淆。

前提条件

用例的前提条件列出开始使用案例之前必须满足的条件。例如，前提条件可能是另一个用例已经执行或用户具有运行当前用例的访问权限。并不是所有的用例都有前提条件。

User Case 框图并不显示用例执行的顺序。但前提条件可以建档这方面的信息。例如，一个用例的前提条件可能是另一个用例已经执行。

主事件流和其他事件流

用例的具体细节在主事件流和其他事件流中描述。事件流描述执行用例功能的具体步骤。事件流关注系统干什么，而不是怎么干，是从用户角度写成的。主事件流和其他事件流包括：

· 用例如何开始

· 用例的各种路径

· 用例的正常（主）流程

· 用例主事件流的变形（其他事件流）

· 错误流

· 用例如何结束

建档事件流时，可以用标号清单、段落文本、强调符清单或事件流程图。事件流应与收集的要求一致。确定文档流程时，记住文档的接收者。客户通过审查这个文档相信其准确反映客户的期望。分析人员通过文档保证与要求一致。项目管理员检查文档，更好地了解要建的项目，并可调整项目结算。编写流程时，一定要避免详细讨论怎么干。

事后条件

事后条件是用例执行完毕之后必须为真的条件。例如，可以在用例完成之后设置一个标志。这种信息就是事后条件。和前提条件一样，事后条件可以增加用例顺序方面的信息。例如，如果一个用例运行之后必须运行另一个用例，则可以在事后条件中说明这一点。并不是每个用例都有事后条件。

在“基于 Web 的远程教学系统”中，我们建档了每个用例的事件流，由于用例较多，下面仅以“定制课件”用例的事件流做说明。

定制课件

简要说明

本用例让学生对课程按照自己的需要和兴趣进行定制，这样每次进行课程学习时将会看到自己希望的页面，避免庞杂信息造成的时间浪费。

前提条件

本用例的前提是学生的身份验证用例必须已经执行。

主事件流

1、学生选择“个性化定制”选项。

2、系统显示所有的课件类别。

3、学生在页面上选择需要的课件类别。

4、学生保存此选择。

5、系统显示更新后的个性化页面并将改变保存到数据库。

6、用例结束。

其他事件流 A1：学生没有做出任何选择

1、系统页面显示所有课件类别。

2、用例结束。

其他事件流 A2：学生放弃选择

1、系统取消学生所做的任何选择。

2、用例结束。

错误流 E1：保存定制时登录超时

1、系统告诉学生超时信息，并提供给学生重新登录的页面。

2、用例结束。

4．2．4 建立关系

UML 对用例和角色支持几种类型的关系，包括：通信关系、使用关系、扩展关系和角色一般化关系。

通信关系

通信关系（communicates relationship）描述角色与用例之间的关系。在 UML 中，通信关系是使用箭头表示的。

每个用例都应由角色启动，除下面介绍的使用关系和扩展关系中的用例。

使用关系

使用关系（uses relationship）使一个用例可以利用另一个用例提供的功能。使用关系通常用于造型一些两个或多个用例的共同的可复用功能。本项目中，学生身份验证、教师身份验证和管理员身份验证与其它用例构成使用关系。

UML 中将使用关系显示为箭头和<<users>>字样。

扩展关系

扩展关系（extends relationship）允许一个用例（可选）扩展另一个用例提供的功能。它与使用关系相似，这两个关系都是把共同功能分离到另一个用例中。

Uml 中，扩展关系表示为箭头加<<extends>>字样。

角色一般化关系

角色一般化关系（actor generalization）表示几个角色有一些共性。

这种关系并非总是必需的。一般来说，只有在关系认为一种角色与另一种角色的表现不同时才需要。这些框图的作用仍然是通信。如果角色一般化使小组得到有用的信息，则进行角色一般化，否则就没必要进行角色一般化。

4．2．5 规划用例图

4．2．5．1 设置规范

用例规范：

Rose 提供每个用例的详细规范。这些规范可以帮助建档用例的特定属性，如用例名称、优先级和版型。

（1）命名用例

模型中的每个用例应有唯一名称。用例应从客户角度命名，以帮助确定项目范围。用例名还应独立于实现方法。要避免使用与特定实现方法相联系的短语，如 Internet。用例通常用动词和短语命名。本项目中，我们尽量用短语命名用例。

（2）指定用例版型

在 UML 中，版型（stereotypes）用于分类模型元素。例如，如果有两种用例，A 类和 B 类，可以生成两个新用例版型 A 和 B。版型不常用于用例，更常用于其他模型元素，如类和关系。但如果喜欢，也可以增加用例版型。

（3）指定用例优先级

定义用例时，可能要指定优先级。利用优先级，随着项目的进展，可以知道处理用例的顺序。如本项目中，学生的课件浏览用例优先级就比较高。

角色规范：

Rose 中可以指定角色名、版型、基数和其他细节。

（1）命名角色

每个角色应有唯一名称。

（2）指定角色版型

除了 Actor 外，角色没有其他可用的版型。但可以定义自己的角色版型并在自己的 Rose 模型中使用。

（3）设置角色基数

可以在 Rose 中指定某个角色要有多少实例。例如，可能要知道许多人起客户角色的作用，而只有一个人起管理者的作用。这可以用基数字段表示。如，“ 学生 ”角色有多个实例。

Rose 提供几个基数选项：

Cardinality（基数）	**含义**
n（缺省）	多
0..0	0
0..1	0 或 1
0..n	0 或多
1	1
1..n	1 或多

4．2．5．2 增加说明

增加用例说明。比如说，“ 定制课件 ”用例，可以说明为 “ 提供给学生选择课件分类，定制个性化页面 ”。

增加角色说明。比如说，“ 学生 ”角色，可以说明为 “ 已经在学校注册并准备学习课程的人 ”。

4．2．5．3 文件和 URL 连接

可以将文件连接用例。例如，文档中可能包含用例的事件流。可能还有其他文档，介绍用例情形、用例中要求的规范和其他描述用例的文件。也可以将 URL 连接用例。如将 “ 定制课件 ”事件流文档连接 “ 定制课件 ”用例。

可以将文件和 URL 连接角色。连接角色的文件或 URL 应为只适用于该角色的文档。例

如，可能有个工作流框图，演示角色的责任和工作流程。如果角色是另一系统，则可能要连接一些与这个外部系统相关的文档或规范。

4．2．5．4 规划用例图

Use Case 框图显示系统中的用例与角色及其相互关系。用例是系统提供的高级功能块，角色是与所建系统交互的对象。

Use Case 框图的一大优势是通信。客户可以从这个 Use Case 框图取得大量信息。通过这个用例，客户可以知道谁要反对系统。通过查阅用例与角色，他们知道项目的具体范围。这样就有助于寻找缺少的功能。

通常，一个系统要生成几个 Use Case 框图，高层框图在 Rational Rose 中称为主框图，显示用例包（组）。其他框图显示一组用例与角色。也可能要生成一个所有用例与角色的组。生成多少 Use Case 框图和取什么名称完全由分析者决定。框图中应有足够的有用信息，又不至于太拥挤。在本项目中，用例较多，如果在一个主框图中全部显示，复杂与凌乱可想而知。因此引入了用例包，系统的主框图显示了四个用例包：学习用例包、办公用例包、交流用例包、管理用例包。这四个用例包涵盖了获取的全部用例。

Use Case 框图有个特定目的——建档角色（系统范围内的一切）、用例（系统范围内的一切）及其关系。生成 Use Case 框图要记住：

· 不要建模角色与角色的通信。按定义，角色在项目范围之外，因此，角色之间的通信不在所建范围之内。可以用工作流框图检查角色的通信。

· 不要在两个用例之间直接画箭头（除了后面介绍的使用与扩展关系）。框图显示可用的用例，但不显示用例执行的顺序，这可以用活动框图。

· 每个用例都应由角色启动。即应当有个从角色指向用例的箭头，除了后面介绍的使用与扩展关系。

· 可以把整个数据库看成整个 Use Case 框图下面的层。可以用一个用例在数据库中输入信息，然后在另一个用例中访问数据库中的信息。不必在用例之间用箭头显示信息流程。

生成框图时，可以给用例和角色加上图注说明。例如，可能要表明哪个角色与哪个用例交互，为什么一个用例参与使用或扩展关系，为什么一个角色继承另一角色。可以加进的图注有两种：说明和文本框。一般来说，说明适用于框图上的一个项目，而文本框在框图中放上框图名或其他一般信息。

4．3 对象交互图的建模

这个阶段是指建模系统对象之间的交互（Interaction）。交互框图分两种：Sequence（时序图）和 Collaboration（协作图）框图。两者都显示参与用例流程的对象和对象之间发送的信息。时序图按时间排序，协作图按对象本身组织。

4．3．1 寻找对象

研究事件流中的名词是寻找对象的好办法，另一个办法是查阅情境文档。情境是事件流的特定实例。比如“定制课件”用例的事件流中，学生要做出选择就必须先看到所有选项，那么一个窗口对象就是必须的，另外，选择之后保存数据，则必须有一个数据对象。

研究情境时，有些名词是角色，有些是对象，有些是对象属性，可以看看它是否有任何行为。如果是单纯的信息，则可能是属性，如果还有一些行为，则可能是对象。

并非所有对象都在事件流中。例如，表单可能不出现在事件流中，但必须放在框图中，角色才能输入或浏览信息。其他不在事件流中的对象是控制对象。这些对象控制用例中流程的顺序。

4．3．2 建模时序图与协作图

时序图按时间排序，用于通过情境检查逻辑流程。

协作图用于了解改变的影响，从协作图中很容易看出哪个对象与哪个对象通信。如果要改变对象，就可以方便地看到受影响的其他对象。

时序图和协作图显示同一信息，但组织方式不同。时序图可以显示控制焦点，协作图可以显示数据流。在时序图中，时间是作为一个显式的因素出现的，这样的时序图在构造实时系统时特别有用。而在协作图中，没有显式的时间因素，但是对象之间的关联是一目了然的，这对我们在一组相互关联的对象的语境中考察它们之间的消息传递是很有帮助的。时序图和协作图是对同一事物的不同角度的考察。

交互框图包含事件流中的许多细节，但这里的信息表示成对开发人员更好用的方式。这些框图关注实现用例功能时要生成的对象。时序图和协作图可以显示对象、类或两者都显示。通过交互框图，设计人员和开发人员可以确定需要开发的类，类之间的关系和每个类的操作或责任。交互框图是设计工作的基石。

交互框图包括：

对象：交互框图可以使用对象名、类名或两者。

消息：通过消息，一个对象或类可以请求另一对象或类完成特定功能。

生成交互框图时，是在向对象指定责任。将消息加进交互框图时，是在向接收消息的对象指定责任。一定要向适当对象指定适当责任。在大多数应用程序中，屏幕和窗体不应进行业务处理。它们只能让用户输入和浏览信息。通过分开前端与业务逻辑，生成的结构可以减少改变对效果的影响。如果业务逻辑需要改变，不应影响界面。如果改变一个或两个屏幕的格式，也不应影响业务逻辑。

4．4 寻找类

可以检查时序图和协作图中的对象。通过对象的共性即可寻找类。时序图和协作图中的每个对象都要映射相应的类。

从用例的事件流中，研究一些名词也可以知道某些类。

寻找类时要考虑三种不同的版型：Entity、Boundary 和 Control（项目、边界和控制）。它们并不是都能从事件流和交互图中找到。

4．5 需求规格说明

经过以上各环节，对用例的分析及细化，对象和类的的识别，综合起来，就可以对系统的动态行为有一个较全面深刻的理解。

在此基础上，用精确的形式化的描述形成需求规格说明，以和用户达成共识。

第五章、 需求分析模型在“ 基于 WEB 的远程教学系统 ”的应用

5．1 项目背景

“ 基于 Web 的远程教学系统 ”从立项至今已经近一年的时间，其中调查与需求分析占用了 40%左右的时间。在此历程中，深刻体会到，无论是用户还是开发人员，对他的认识对象的认识，都有一个由浅入深， 由表及里， 由简单到复杂， 由局部到全面， 由不准确到准确的不断渐进，螺旋上升的过程。所以需求是不可能一次性获取永不更改的。我们力求完全、准确但也经历了不少的反复，同时我们注重使开发出的系统具有弹性，可以适应不断的变化。

在开发的每一个阶段，我们都建档了必要的文字，有些时候进行了建模的工作。这样使得所有的分析和设计不再仅仅停留在个人的思想中，而是把这种思想表达出来，这也有利于本开发小组成员之间的交流和共识，同时使得本项目的后续开发工作和新的小组成员加入之后的工作有踞可依。

5．2 模型的应用

按照第四章所述软件需求分析步骤，本小组在充分调查研究需求的基础上，选用了 Rose2000 作为 UML 建模工具。

鉴于此系统较为庞杂，我们在用例中引入了包，这里包的概念是指用例组。这样便于在模型中组织项目，进行嵌套管理。

本项目建模了三个角色、四个用例组、一个主用例图、三个用例。以下将详细说明。

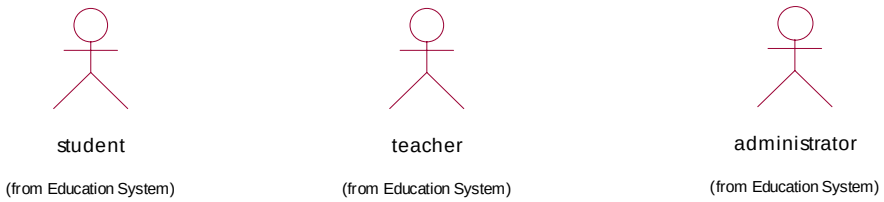
5．2．1 三个角色

分别是：教师、学生和管理员。

（1）教师：是指利用本系统进行教学活动的授课老师或者教育专家。

（2）学生：是指利用本系统进行学习活动的人，可能是在校生，也可能是社会中希望从本系统获取知识信息的任何人。

（3）管理员：是指有权力维护本系统的安全、有序、无故障运营的管理者。



5．2．2 主用例图

本系统的主用例图，包含四个用例组，分别是：

（1）办公：这部分是指教师工作区；

（2）学习：学生所进行的活动；

（3）交流：指系统用户之间的交流；

（4）管理：系统管理员的工作；

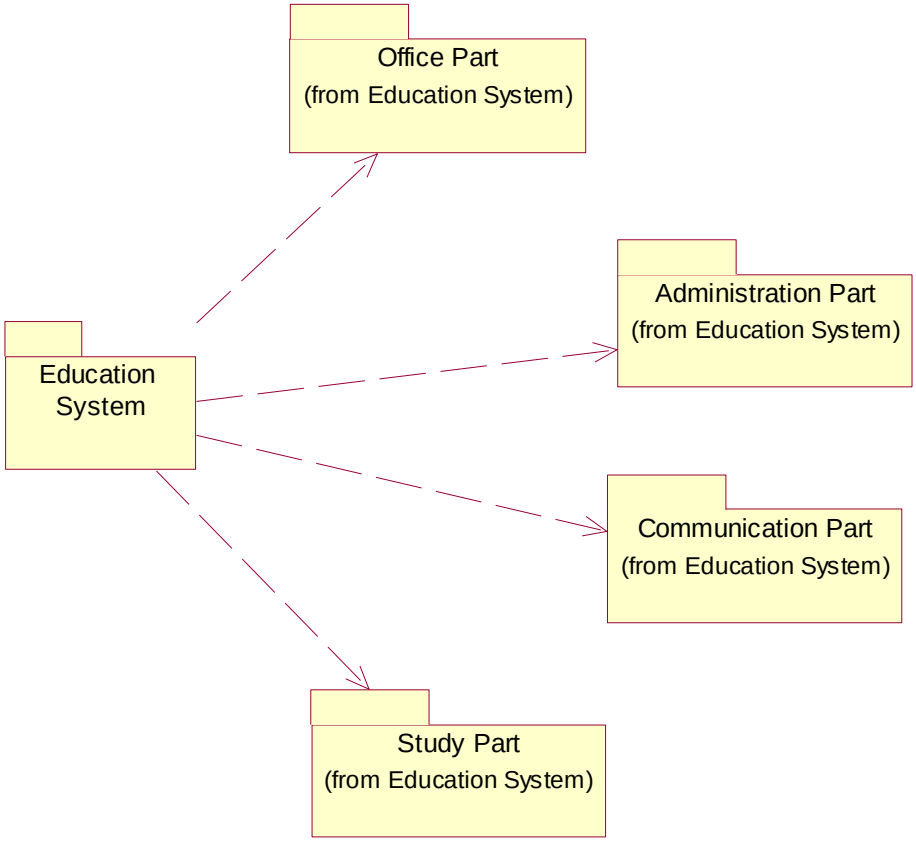
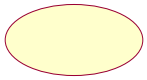


图 5-1 远程教育系统主用例图

5．2．3 三个用例

（1）验证管理员身份：管理员需要身份验证，才能实施管理权力；



（2）验证教师身份：教师需要通过身份验证，才能实施教学活动；

（3）验证学生身份：学生需要通过身份验证，才能参加学习活动；

此三个用例被别的用例所使用，跟别的用例构成使用关系，用于实现一些需要通过身份验证才能实现的功能。

5．2．4 四个用例组

这四个用例组已经体现在系统的主用例图中，在 Rose 的用例视图中，是作为包的形式出现的。下面分开介绍：

（1） 学习用例包：含一个用例图，四个用例包；

一个用例图：为学习用例包的主用例图，如图 5-2 所示。

四个用例包：

1、学习课件：含一个用例图，三个用例，如图 5-3 所示。

2、实时课堂：含一个用例图，三个用例，如图 5-4 所示。

3、在线作业：含一个用例图，四个用例，如图 5-5 所示。

4、在线考试：含一个用例图，三个用例，如图 5-6 所示。

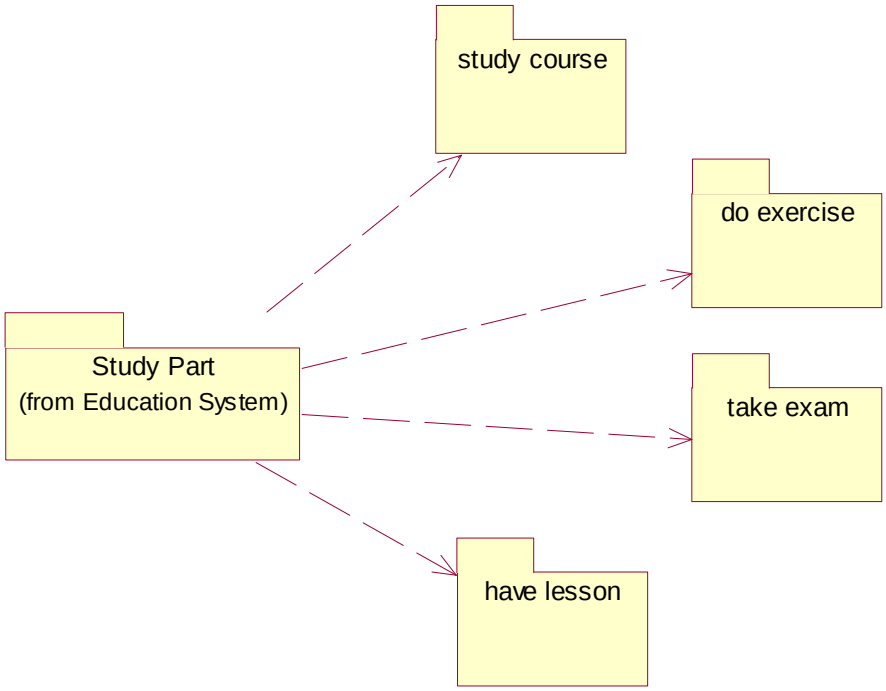


图 5-2 学习用例包主用例图

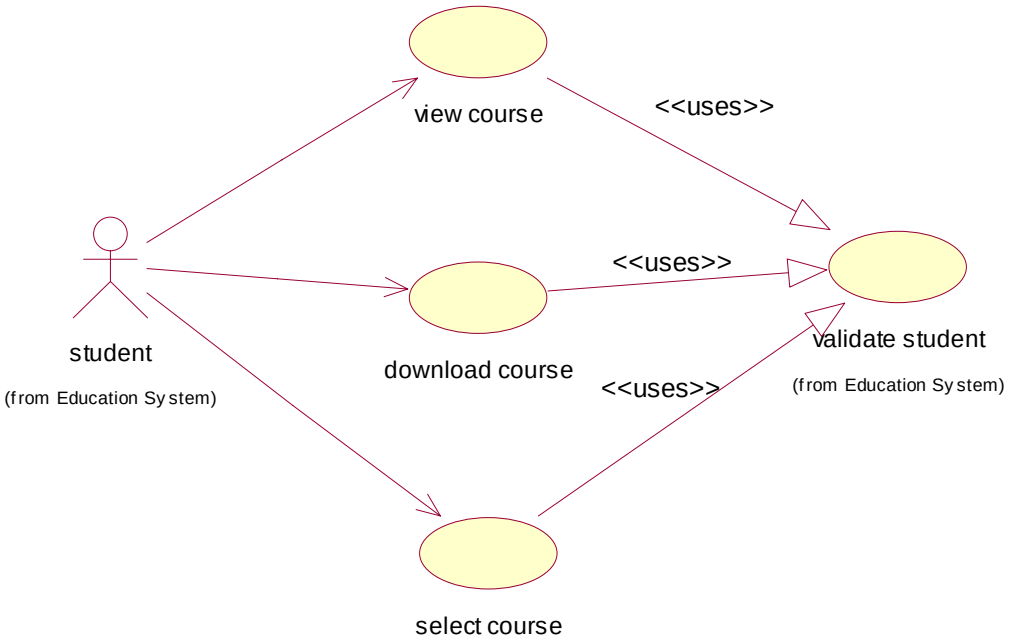


图 5-3 学习课件用例图

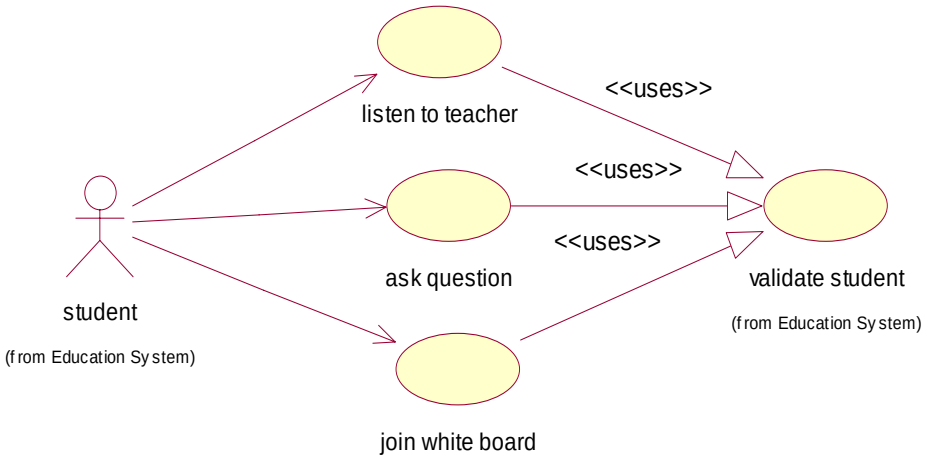


图 5-4 实时课堂用例图

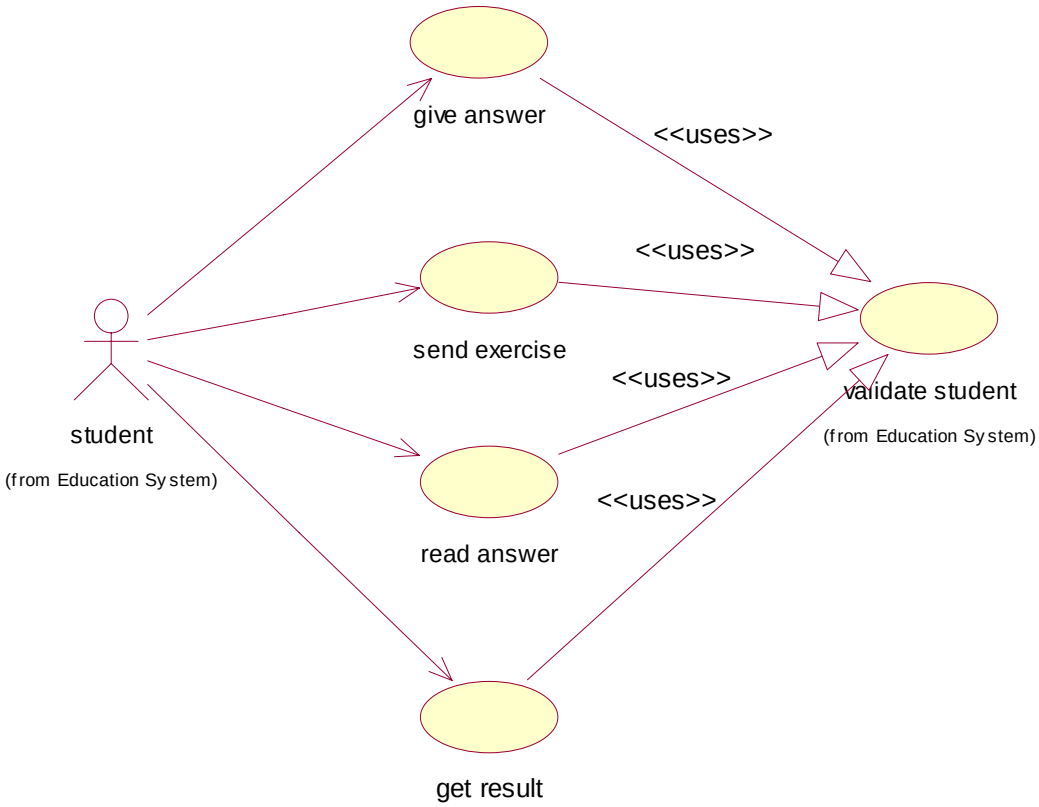


图 5-5 在线作业用例图

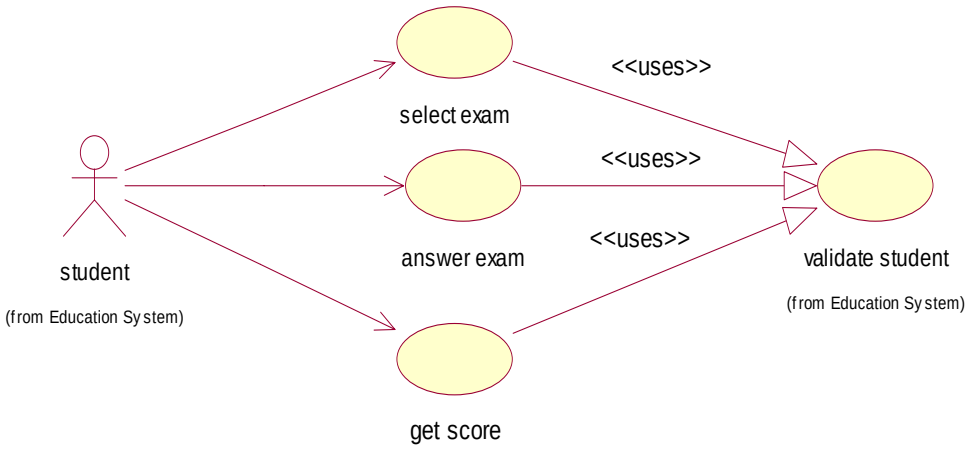


图 5-6 在线考试用例图

(2) **办公用例包：**含一个用例图，六个用例包；

一个用例图：办公区的主用例图，如图 5-7 所示。

六个用例包：

1、 **管理课件用例包：**含一个用例图，三个用例，如图 5-8 所示。

2、 **实时教学用例包：**含一个用例图，四个用例，如图 5-9 所示。

3、 **管理学生用例包：**含一个用例图，四个用例，如图 5-10 所示。

4、 **管理作业用例包：**含一个用例图，三个用例，如图 5-11 所示。

5、 **管理试题用例包：**含一个用例图，四个用例，如图 5-12 所示。

6、 **发布公告用例包：**含一个用例图，二个用例，如图 5-13 所示。

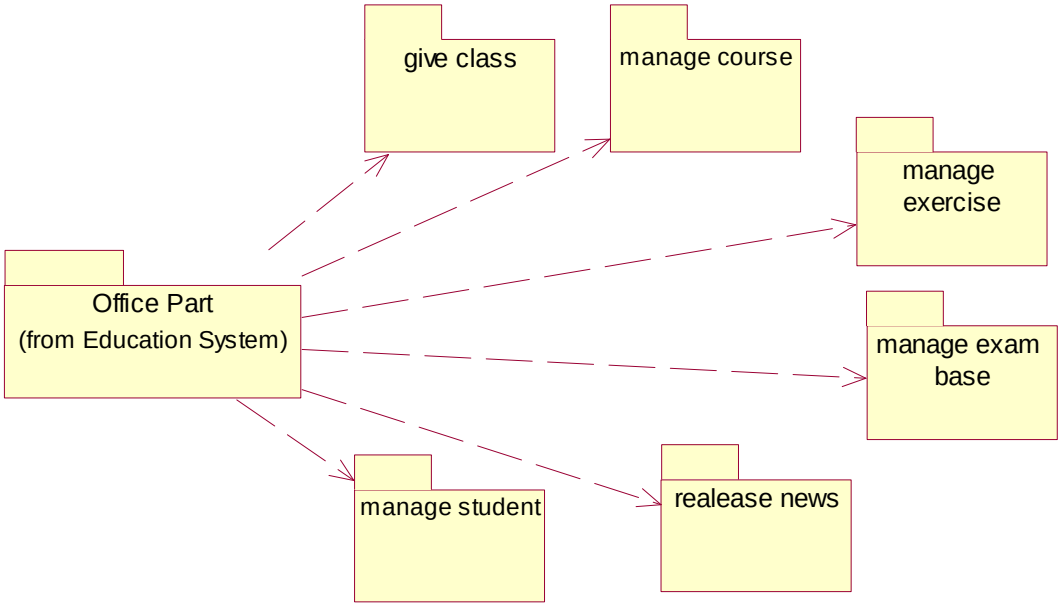


图 5-7 办公区主用例图

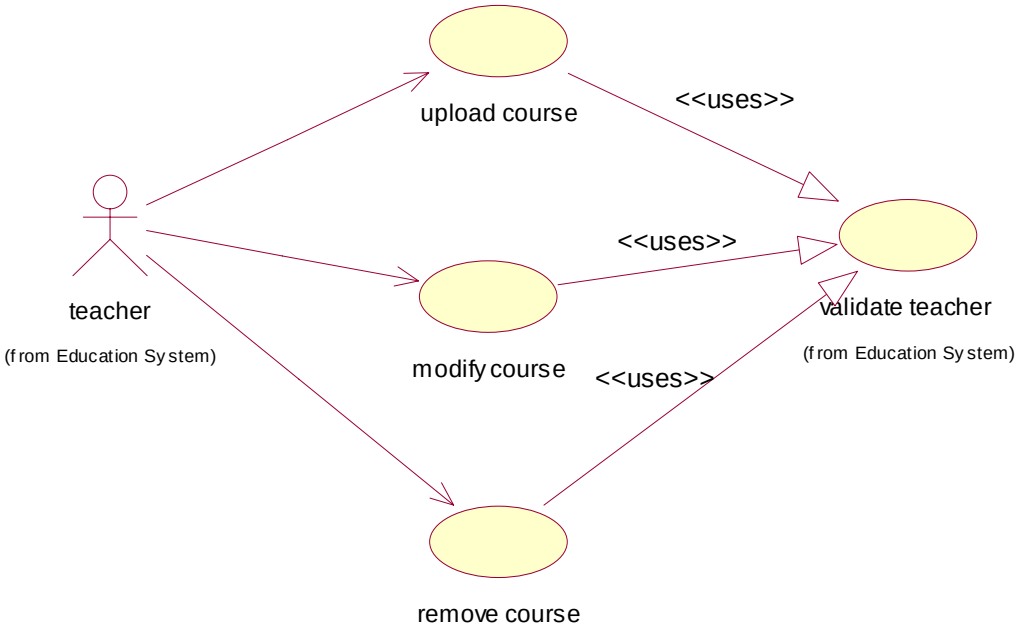


图 5-8 管理课件用例图

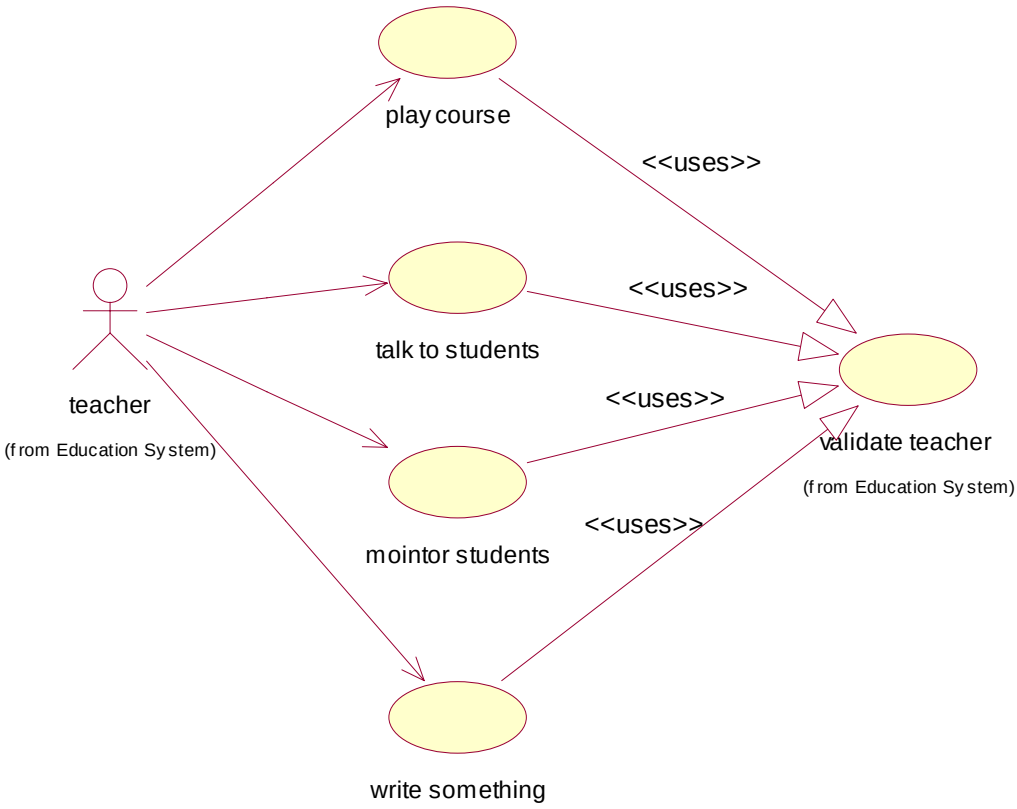


图 5-9 实时教学用例图

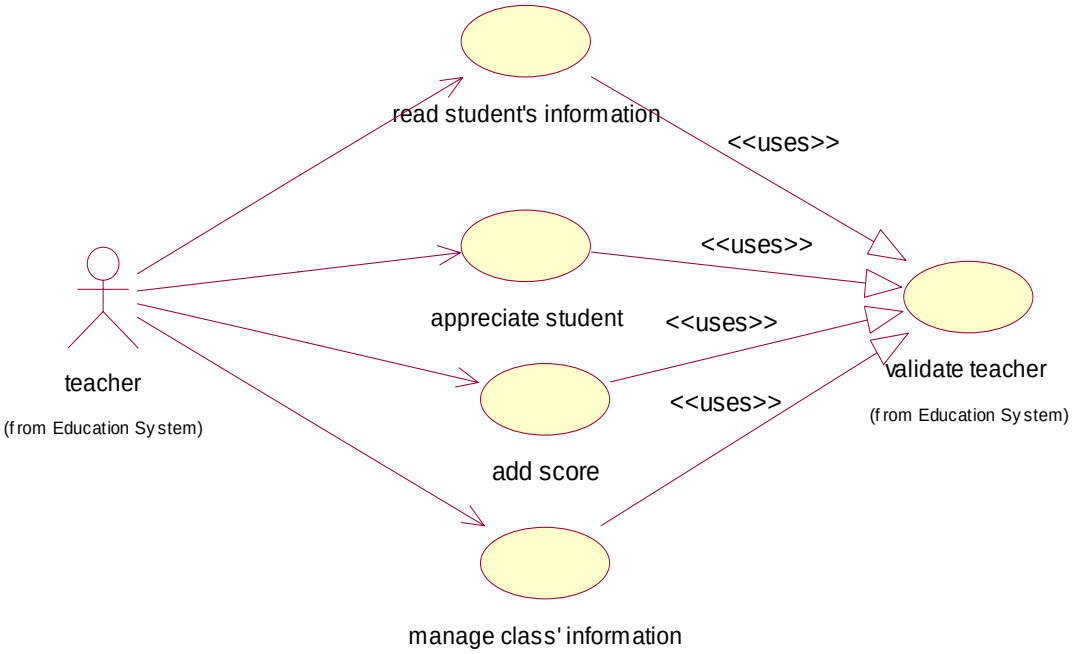


图 5-10 管理学生用例图

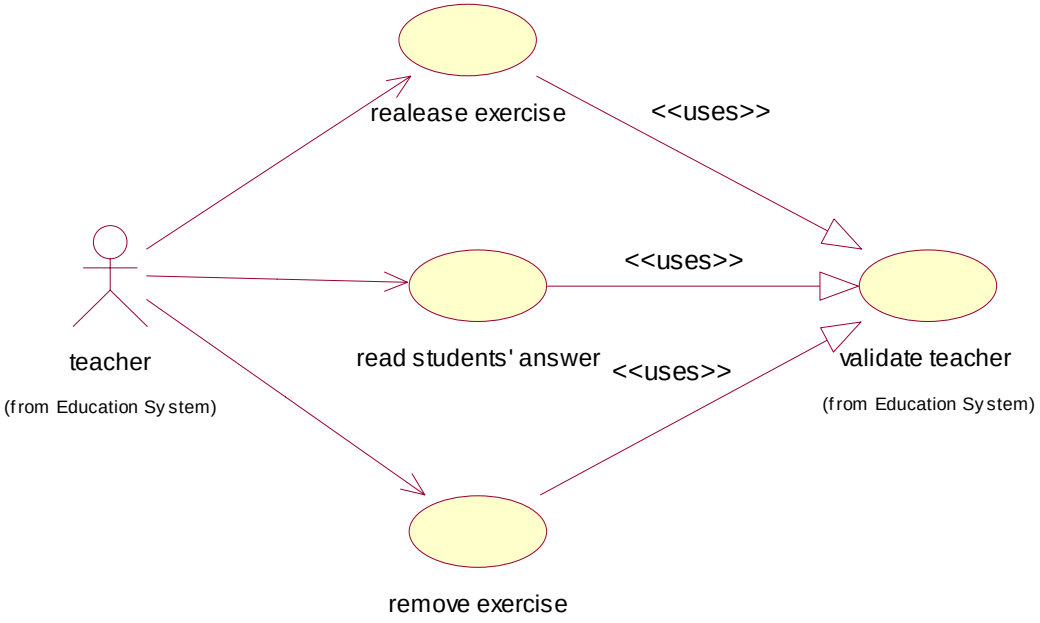


图 5-11 管理作业用例图

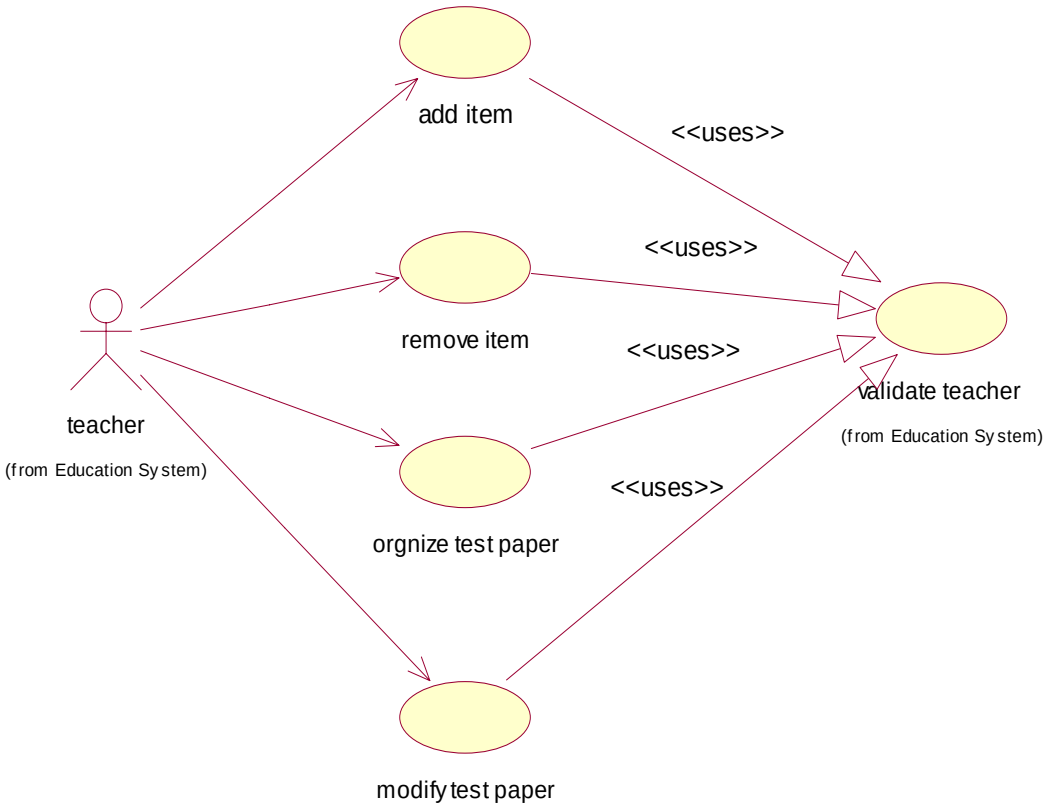


图 5-12 管理试题用例图

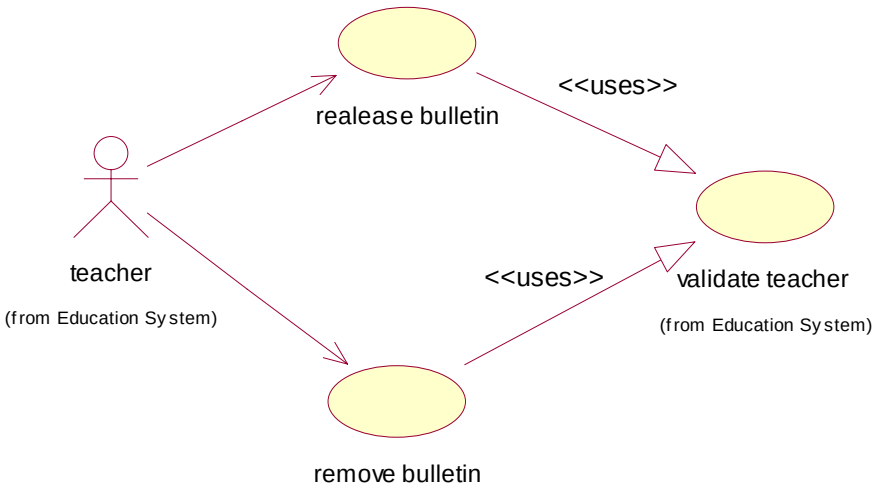


图 5-13 发布公告用例图

（3） **交流用例包：**含一个用例图，五个用例包；

一个用例图：交流区的主用例图，如图 5-14 所示。

五个用例包：

1、聊天室用例包：含一个用例图，二个用例，如图 5-15 所示。

2、论坛用例包：含一个用例图，三个用例，如图 5-16 所示。

3、邮件用例包：含一个用例图，五个用例，如图 5-17 所示。

4、新闻用例包：含一个用例图，三个用例，如图 5-18 所示。

5、个人信息用例包：含一个用例图，二个用例，如图 5-19 所示。

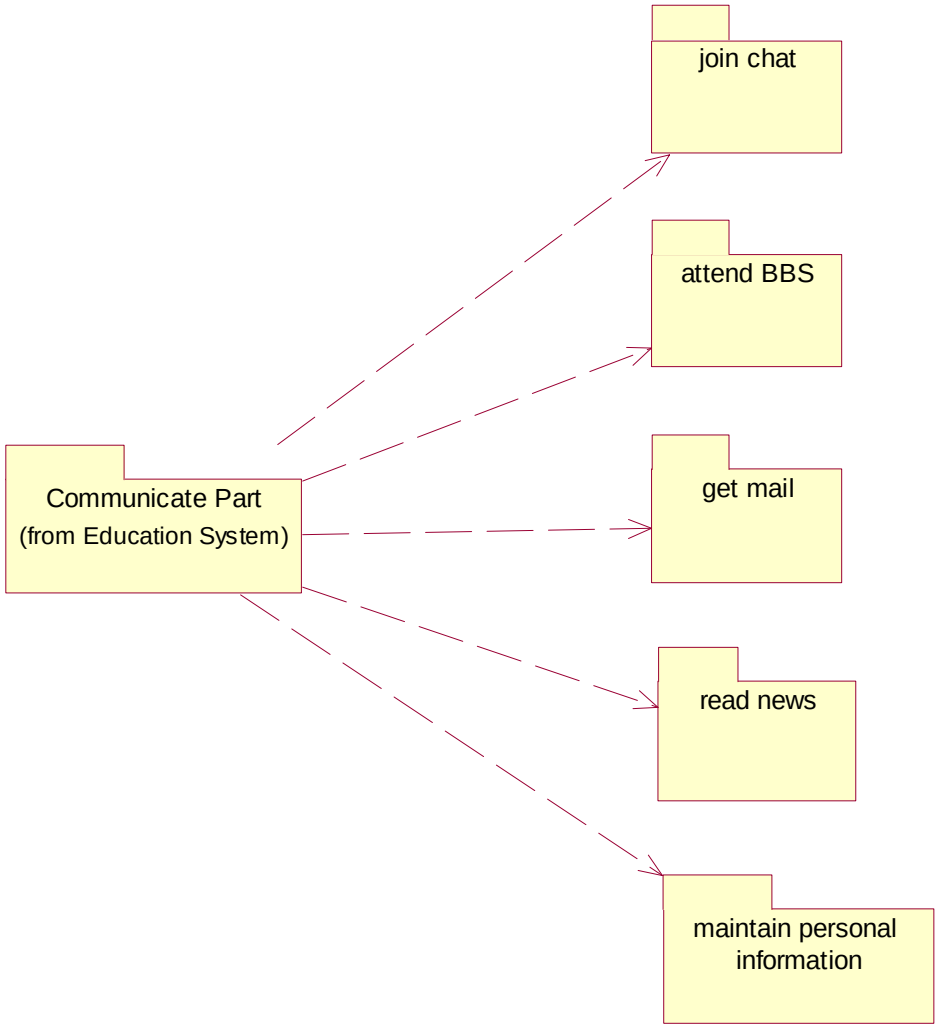


图 5-14 交流区主用例图

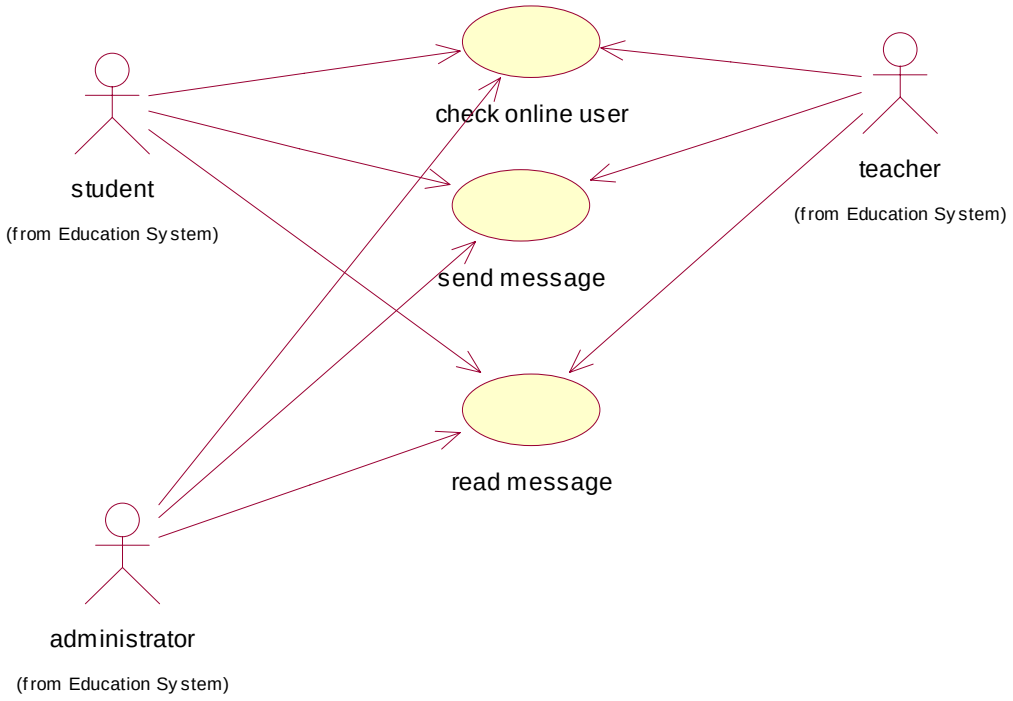


图 5-15 聊天室用例图

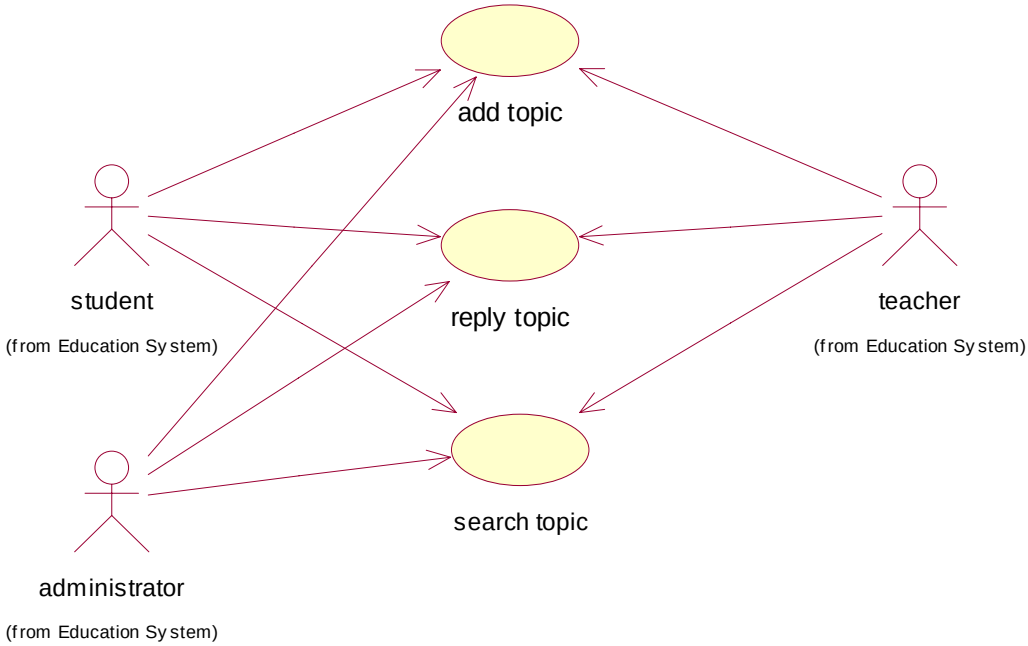


图 5-16 论坛用例图

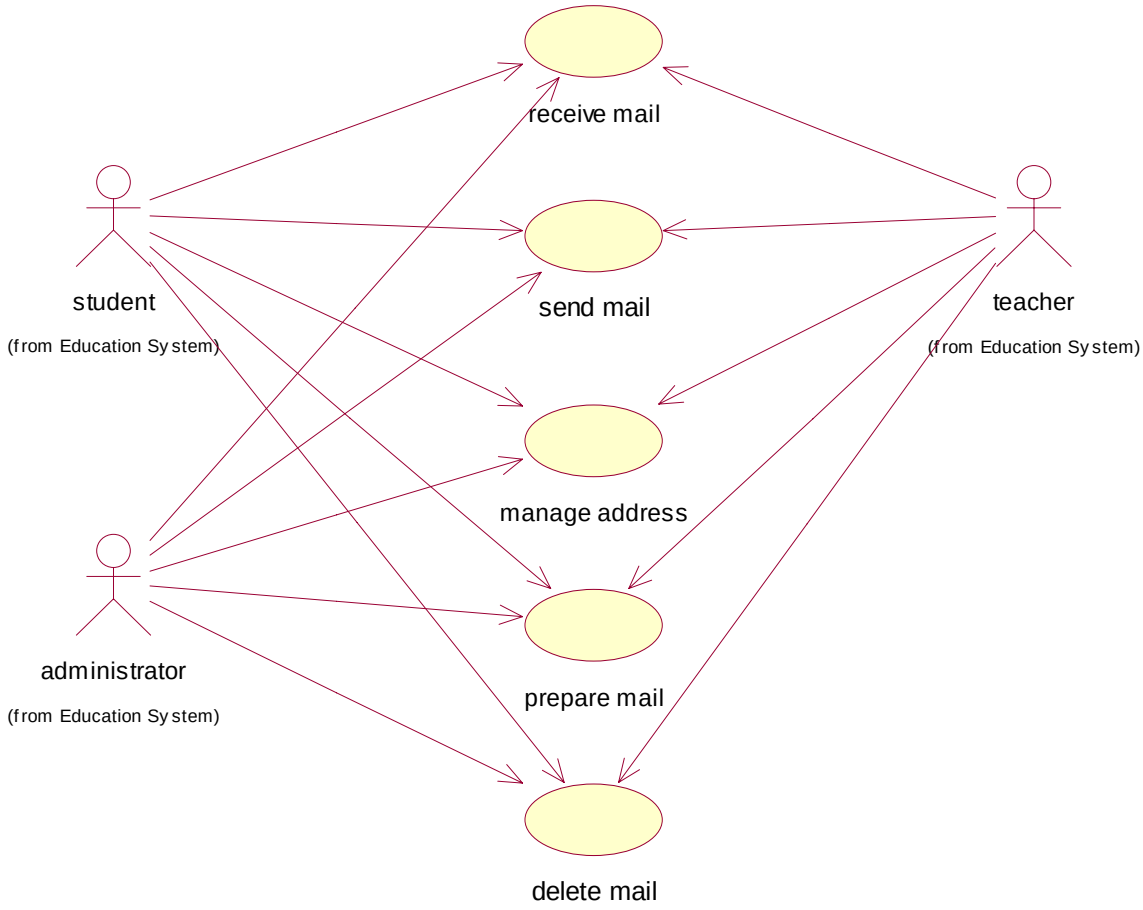


图 5-17 邮件用例图

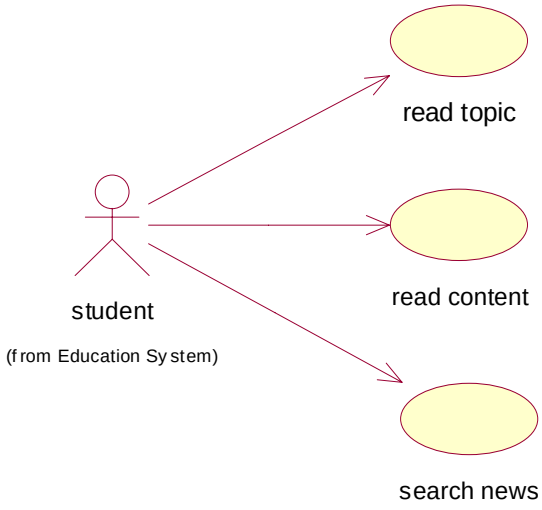


图 5-18 新闻用例图

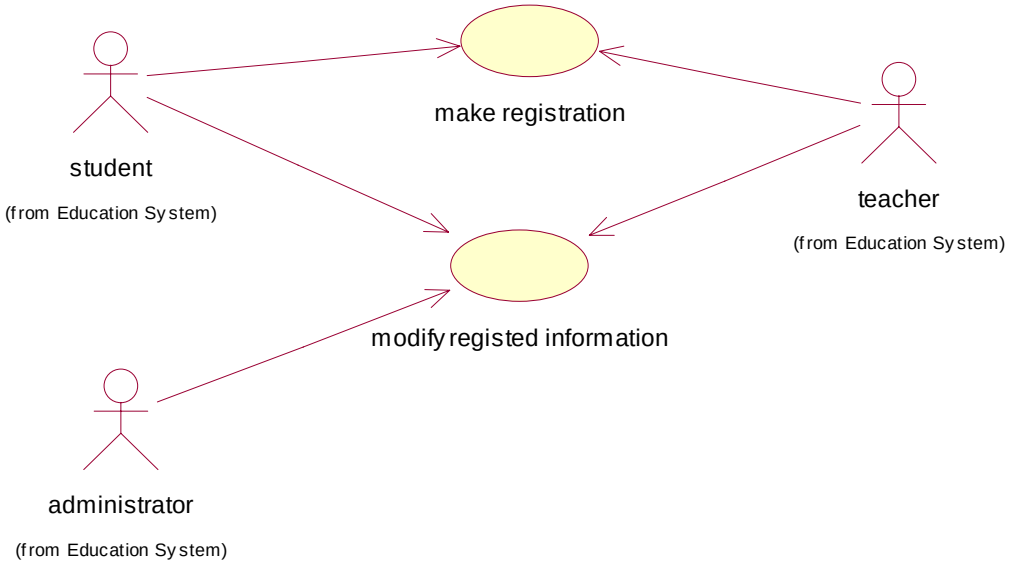


图 5-19 个人信息用例图

（4） 管理用例包：含一个用例图，三个用例包。

一个用例图：管理区主用例图，如图 5-20 所示。

三个用例包：

1、 管理用户用例包：含一个用例图，二个用例，如图 5-21 所示。

2、 管理交流用例包：含一个用例图，四个用例，如图 5-22 所示。

3、 管理学习用例包：含一个用例图，四个用例，如图 5-23 所示。

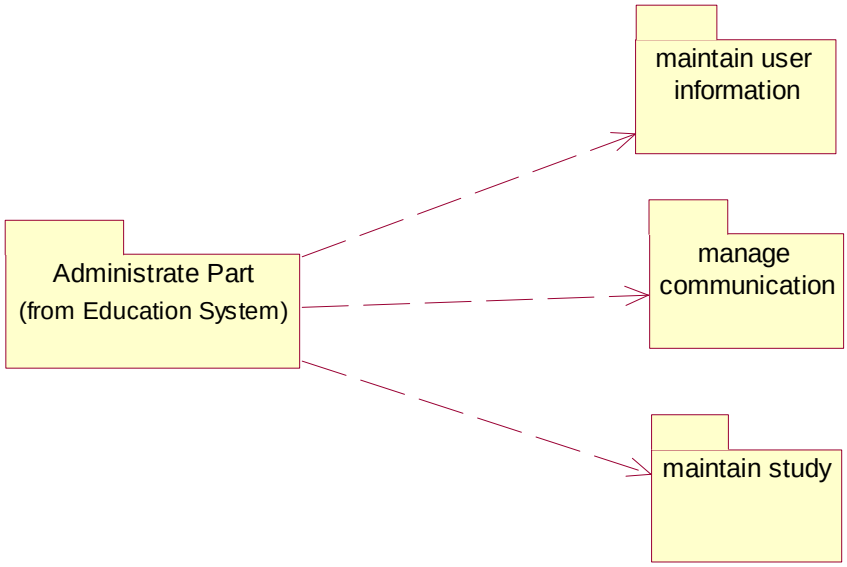


图 5-20 管理区主用例图

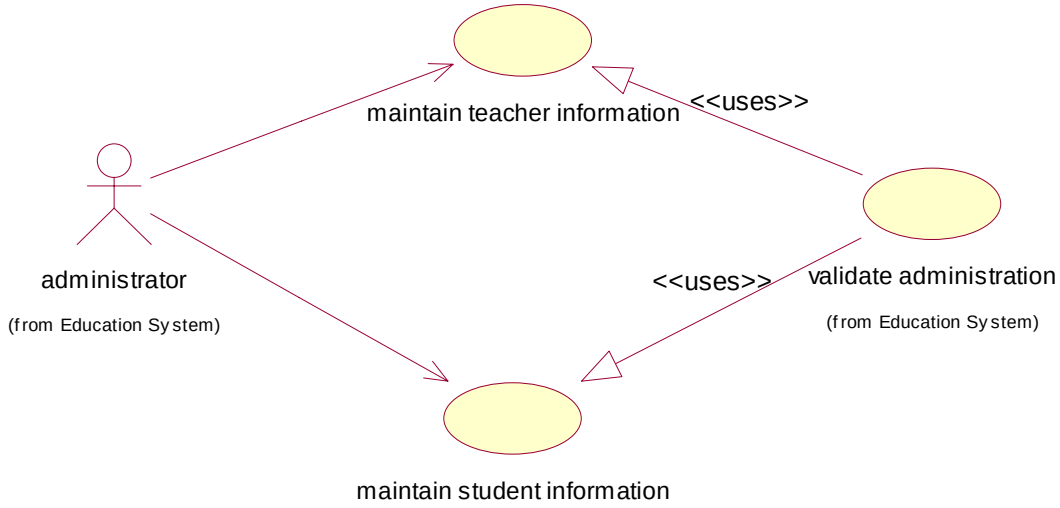


图 5-21 管理用户用例图

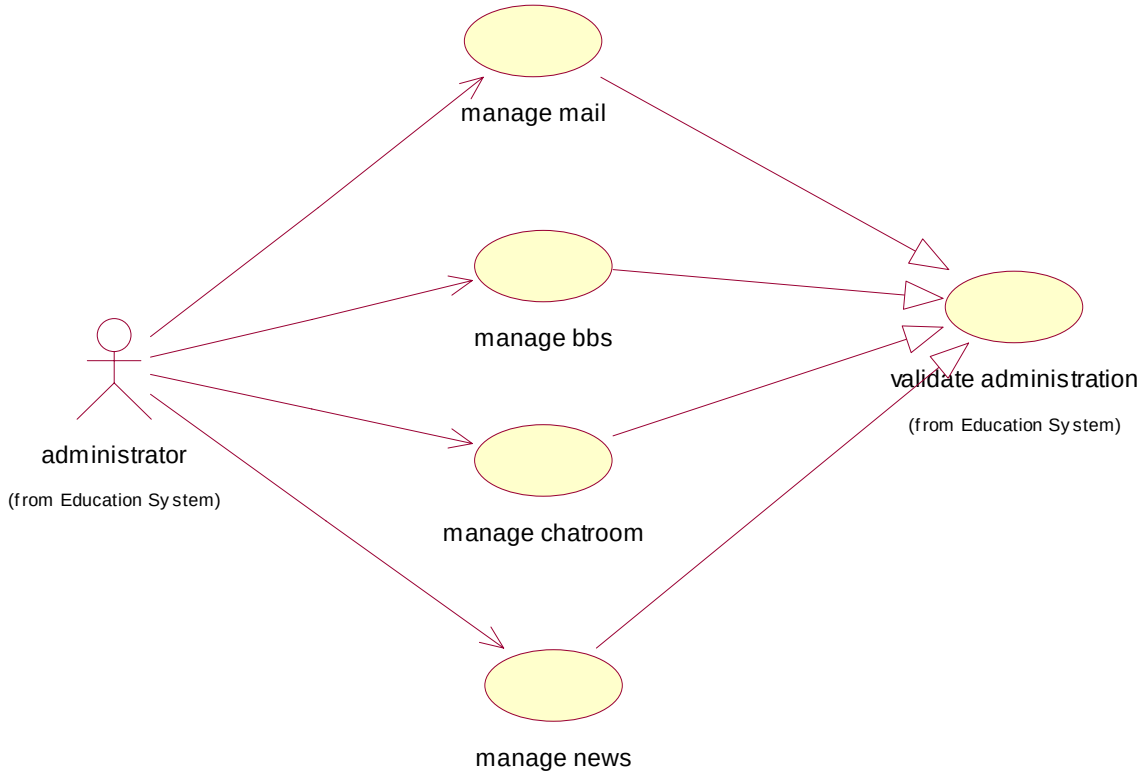


图 5-22 管理邮件用例图

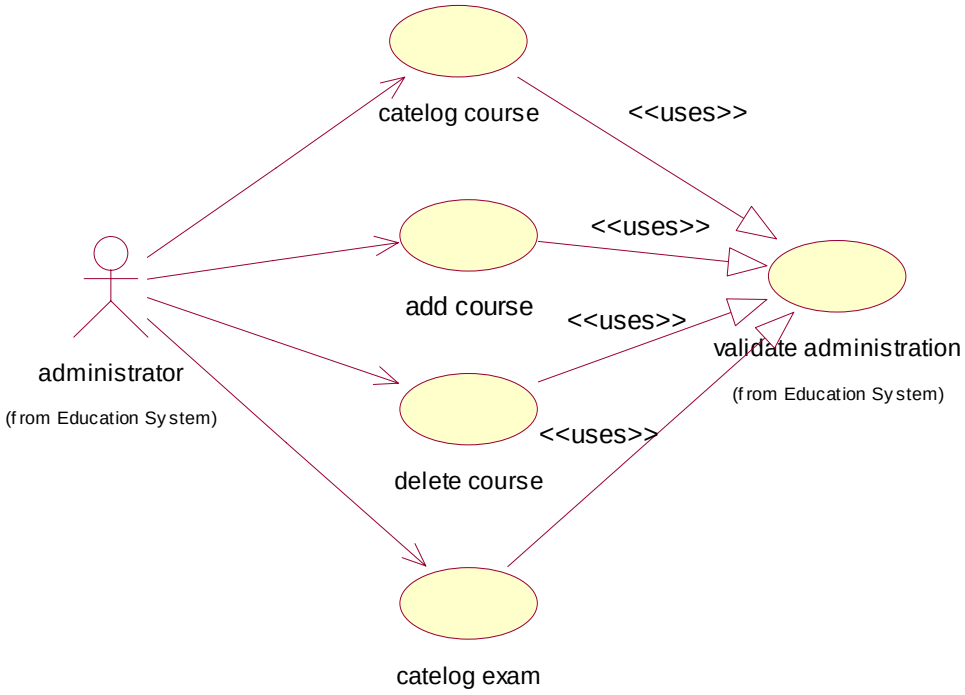


图 5-23 管理学习用例图

5．2．5 时序图和协作图

在对以上所列用例图进行细化分析的基础上，建档了一些用例的事件流，又建模了每个用例的顺序图或者协作图。篇幅限制，仅举例说明。

下面以“ 学生注册用例 ”为例来展示时序图和协作图的建立。

（1）“ 学生注册用例 ”时序图

通常可以采用两步法生成时序图。

第一步关注客户关心的高级信息。信息还不映射操作，对象还不映射类。这些框图只是让分析人员、客户和让对业务感兴趣的其他人了解系统的逻辑流程。

本例时序框图的第一步如图 5-24 所示。

第二步，客户同意第一步框图的流程后，小组加进更多细节。这时的框图对客户可能作用不大，但对开发人员、测试人员和项目组的其他成员更有用。

首先，可能要向框图中增加一些对象。通常，每个交互框图有一个控制对象，负责控制情景中的程序。一个用例的所有交互框图可能共享同一控制对象，因此一个控制对象可以处

理该用例的所有程序信息。如图 5-25 中 Student Manager 就是一个控制对象。控制对象并不进行任何业务处理，只是象其他对象发消息。控制对象负责协调其他对象的效果和委托责任。为此，控制对象有时也称为管理者对象。使用控制对象的好处是能够将业务逻辑与程序逻辑分开。如果程序需要改变，只影响控制对象。

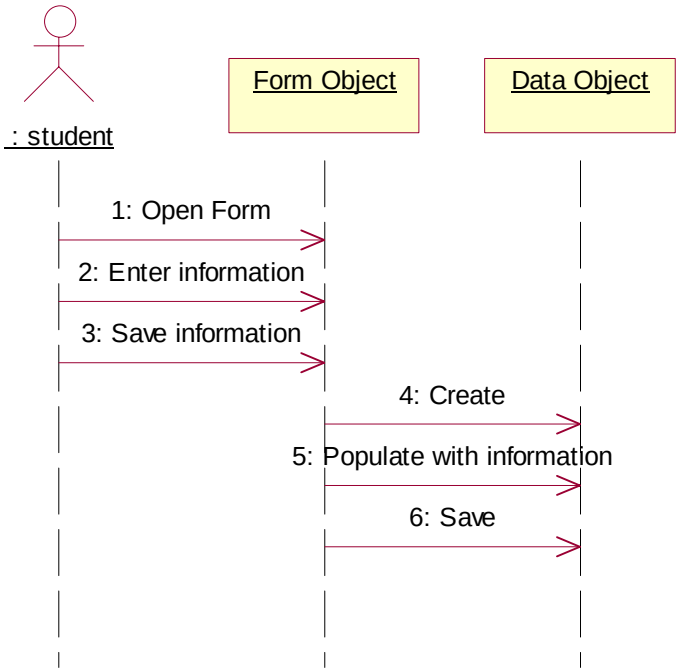


图 5-24　Sequence 框图的第一步

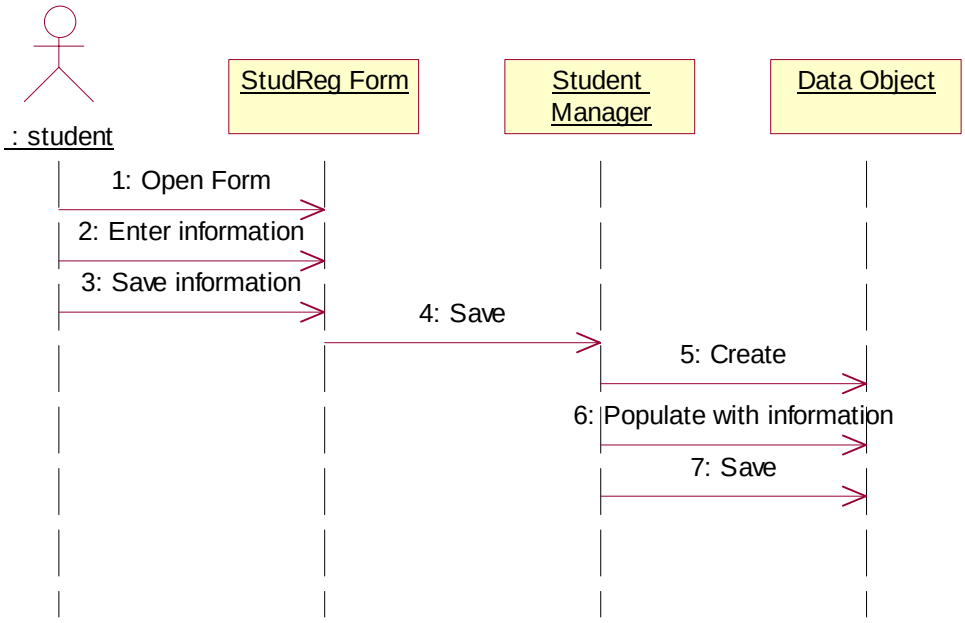


图 5-25　带控制对象的 Sequence 框图

时序图中可能还要增加其他对象来处理安全、错误和数据库连接等问题。许多对象是一般性的，建立一次即可在多个应用程序中复用。例如，本例中采用了事务处理器来处理数据库连接问题。

将信息存入数据库和从数据库中取出信息时，通常有两个选项。假设要将新学生 Tommy 保存到数据库中。Tommy 对象可能知道数据库，这时可以把自己加进数据库中。Tommy 对象也可能与数据库逻辑完全分开，这时要通过另一个对象将 Tommy 保存到数据库中。图 5-26 中，假设 Tommy 知道数据库。在这种情形中，应用程序逻辑与数据库逻辑是不分开的。Tommy 对象负责应用程序逻辑与数据库逻辑，前者可能包括 Tommy 的注册与注销，后者包括将 Tommy 存到数据库中和从数据库中取出。这样如果数据库逻辑改变，则会影响应用程序逻辑，因为许多对象都包含一些数据库逻辑。但这种方法更容易建模和实现。

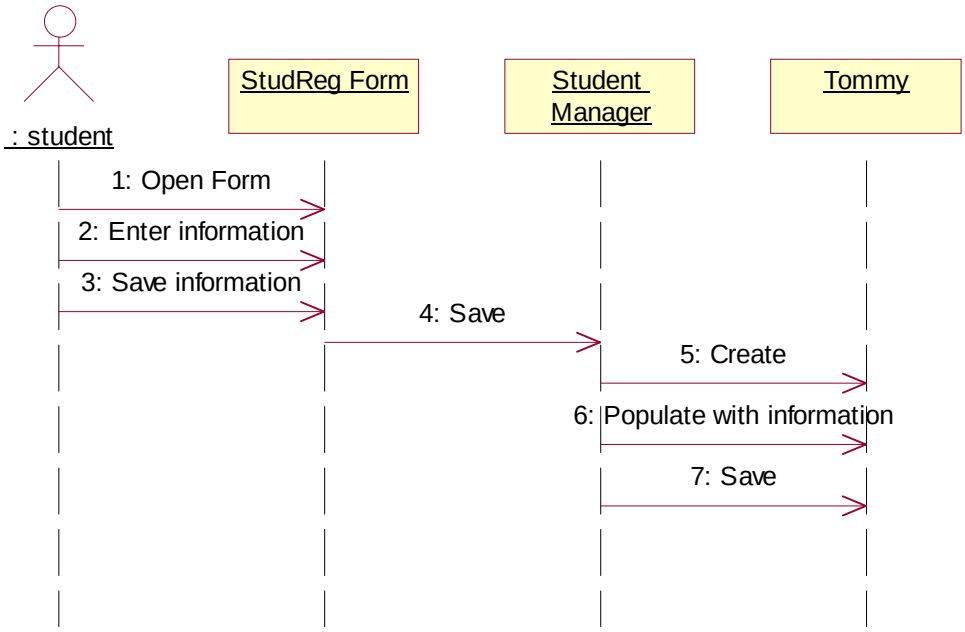


图 5-26 应用程序逻辑与数据库逻辑集成

另一种选项是应用程序逻辑与数据库逻辑分开。这种情况下，需要生成另一对象来处理数据库逻辑。这个对象可以称为事务管理器。Tommy 对象仍然保持业务逻辑，知道如何注册和注销 Tommy。而事务管理器对象知道如何从数据库中取出 Tommy 或将其存入数据库中。这种情形如图 5-27 所示。这种方法的好处是更容易在另一个具有不同数据库或根本没有数据库的应用程序中复用 Tommy 对象，还可以减少需求改变时的影响。数据库改变并不影响应用程序逻辑，应用程序改变也不影响数据库逻辑。缺点是造型和实现更费时间。

在本项目中处理数据库问题时采用了后一种方法，即应用程序逻辑与数据库逻辑分开。在各个交互框图中保持了一致的处理方法。

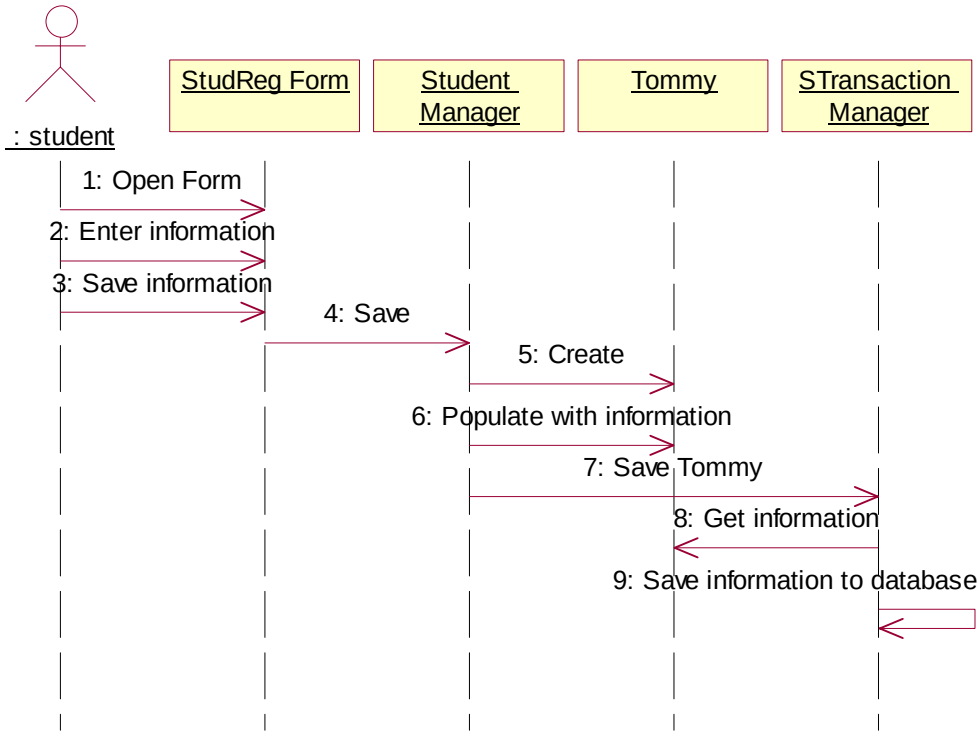


图 5-27 数据库逻辑与应用程序逻辑分开

(2)“学生注册用例”协作图

Rose 支持时序图自动转换为协作图，也支持直接创建，协作图的创建同样可以采用以上所述的两步法。图 5-28 为 Rose 自动生成的协作图。

5．2．6 对象和类

由上述步骤可以看出，创建时序图和协作图时，已经能够识别一些对象。交互框图的作用就是显示对象如何配合，实现用例的功能。其中协作图显示对象间的关系和对象间的消息。从协作图中，系统设计人员可以看出哪个对象是瓶颈，或发现哪些对象需要直接相互通信。到此，需求分析可以告一段落了。接下去，交互图中的每个对象映射成类，每个消息映射类的操作，则在完成交互框图时，Rose 已经生成系统所需的一些类，为生成类框图和实际开发类打下基础。这里也可以很明显地看出，将 UML 引入软件系统的另一个优点，就是分析阶段和设计阶段没有很清晰的界限，很自然的过渡更符合人的思维。

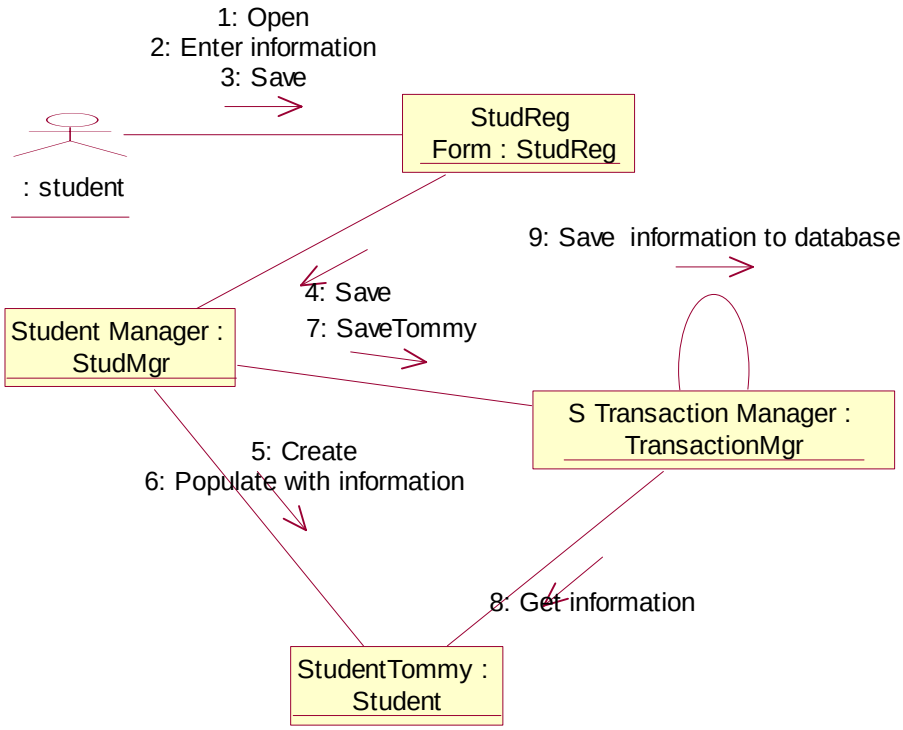


图 5-28 协作图

5．2．7 需求规格说明

经过以上各步的设计与分析，从时序图中可以浏览用例进行中的功能，从协作图中可以浏览对象间的关系和对象间的消息。综合考查每个用例图以及对象交互图和用例的事件流等说明文档，整个系统架构就很清晰了。对系统需求的描述体现在规格说明中，因此我们建档了项目的需求规格说明书用于指导开发。基于 Web 的远程教学系统可以划分为以下功能模块：用户注册、用户登录、远程学习、讨论区、在线考试、在线答疑、聊天室、邮件、新闻公告、教师工作区、管理员工作区、在线作业、个人信息、帮助、离站。整个系统的体系结构如图 5-29 所示。

5．2．8 项目开发简介

本系统在设计软件架构时采用了微软所提出的三层架构的方案，即客户端/服务器/服务器，中间层服务器选型定为 IBM HTTP SERVER 与 IBM WEBSPHERE APPLICATION SERVER，后端数据库服务器选型 IBM DB2 6.1 通用版本，客户端只需普通的浏览器即可。

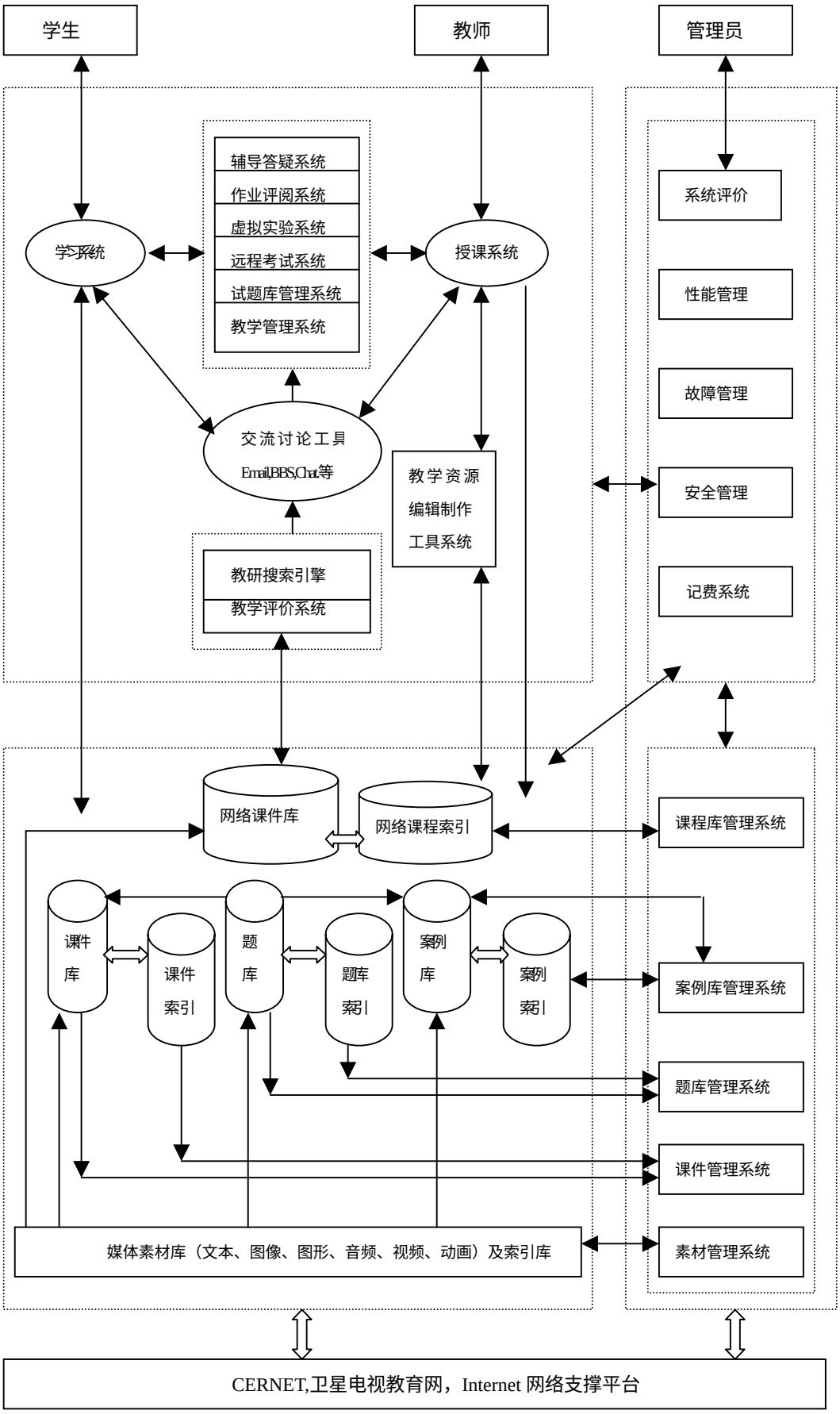


图 5-29 远程教学系统体系结构图

本系统选用 Java 作为主开发语言。目前本系统已经可以完成基本的学习、交流和管理功能，以下为其中几个功能模块的简介。

如图 5-30 为本系统的登录页面，其中提供了注册和身份验证的功能。

如图 5-31 为本系统的主要功能，课件学习模块的页面，其中提供了用户页面个性化定制的功能。

如图 5-32 为本系统的新闻公告系统。

如图 5-33 为本系统提供的非实时交流系统，邮件收发功能的页面。

如图 5-34 为本系统的实时交流系统，聊天室系统的主页面。

图 5-30 远程教学系统登录页面

图 5-31 课件学习页面

图 5-32 邮件系统

图 5-33 新闻公告系统

图 5-34 聊天室系统

第六章、结论与展望

6．1 结论

本文主要的研究成果是将统一建模语言引入软件项目的分析阶段，并在实践经验的基础上建立了一套软件需求分析模型，此模型在“基于 Web 的远程教学系统”的建设中证明行之有效。在研究与应用中，主要的结论有以下几点：

（1）、在面向对象的软件开发方法中，客户关心的是软件系统的功能，而开发工作是围绕目标软件系统的对象模型来进行的。本需求分析模型采用用例（Use Case）技术，就实现了对象模型和功能模型的有机结合。这样既保证了客户和开发人员对系统需求在思想认识上的统一，又为后续的的软件设计和开发工作奠定了良好的基础。

（2）、本模型的建立是为了解决大型、复杂软件系统的需求分析过程，它有利于构建正确的系统。但它对小型软件系统的开发并不适用，一方面是由于 UML 语言本身的复杂性使得它较难掌握，建模花费时间长，另一方面是由于小型系统的需求一般不是很复杂，利用传统的软件分析和设计方法就可以解决问题。

（3）、在本模型的使用过程中可以看出，用例图是需求分析的核心，牵涉全局工作的成败。所以，在软件需求分析阶段，确定角色和捕获用例是非常重要的。绝大多数用例可以在系统需求分析阶段确定，但随着系统的进展，可能会发现更多的用例，甚至会发现前面定义的用例存在不够确切或错误的地方，需要重新修改。因此，在整个系统开发过程中，都应当时刻关注用例。

（4）、要维持系统分析和系统设计之间合理的分离。在分析阶段，用例主要解决“做什么”的问题，因此不要指定任何与需求相关的系统的内部结构，也就是说不要涉及实现细节；而在设计阶段，主要解决“如何去做”的问题，这时才需要指定系统的内部结构。

6．2 UML 技术在软件开发其它阶段的应用

在对 UML 的研究和项目的开发中，我们还试验性地将建模技术用于软件开发的不同阶段，总结起来可以归纳为以下步骤：

（1）设计阶段

设计阶段的任务是通过综合考虑所有的技术限制，以扩展和细化分析阶段的模型。设计的目的是指明一种易转化成代码的工作方案，是对分析工作的细化，即进一步细化分析阶段

所提取的类（包括其操作和属性），并且增加新类以处理诸如数据库、用户接口、通信、设备等技术领域的问题。

设计阶段可以分为两个部分：

第一部分是结构设计，它是高层设计，其任务是定义包（子系统），包括包间的依赖性和主要通信机制。我们希望得到尽可能简单和清晰的结构，各部分之间的依赖尽可能的少，并尽可能的减少双向的依赖关系。

第二部分是详细设计，细化包的内容，使编程人员得到所有类的一个足够清晰的描述。同时使用 UML 中的动态模型，描述特定情况下这些类的实例之间的行为。

（2）实现阶段

构造或实现阶段是对类进行编程的过程。可以选择某种面向对象编程语言（如 Java）作为实现系统的软件环境。Java 很容易实现从逻辑视图到代码部件的映射，因为类到 Java 代码文件之间是一一映射关系。

在实现阶段中，可以选取下列图的说明来辅助编程：

· 类的规格说明：每个类的规格说明详细显示了必要的属性和操作。

· 类图：显示类的静态结构和类之间的关系。

· 状态图：显示类的对象可能的状态、所需处理的转移以及触发这些转移的操作。

· 包含某个类的对象的动态图（时序图、协作图、活动图）：显示该类的某个方法的实现或别的对象是如何使用该类的对象的。

· 用例图和规格说明：显示系统需求和结果。

编码期间也可能会发现设计模型的缺陷。这时需要开发者修改设计模型。修改设计模型时一定要保持设计模型与编码的一致性，以便将来易于维护。

（3）测试和配置

完成系统编码后，需要对系统进行测试，它通常包括：单元测试、集成测试、系统测试和验收测试。在单元测试中使用类图和类的规格说明，对单独的类或一组类进行测试；在集成测试中，使用组件图和合作图，对各组件的合作情况进行测试；在系统测试中，使用用例图，以检验所开发的系统是否满足用例图所描述的需求。

系统的配置是实际的交付系统，包括文档和组成模型等。对“基于 Web 的远程教学系统”来说，是一个典型的客户/服务器/服务器系统。可以用配置图显示系统的物理结构。

6．3 展望

由于时间和精力的原因，对 UML 的认识还有欠缺，软件需求分析模型有待完善，软件分析、设计和测试阶段的 UML 应用也有待形成理论并在实践中检验。鉴于 UML 本身还存在一个不断完善和发展的过程，接下去本人要做的工作可以归纳为：

（1）、此模型虽然在“基于 Web 的远程教学系统”中得到了成功应用，但尚需在更多软件项目的应用中提高它的适用性；

（2）、将 UML 应用于软件开发的整个过程，争取建立一整套面向对象的软件工程方法学，在理论上有所突破；

（2）、UML 在面向对象的分析、设计上是极其优秀的可视化建模语言，但 UML 并没有在面向对象程序代码实现上提出较好的可视化方案。以 Rose 为基础，加入可视化程序代码实现模块，提供一个扩展的建模环境，将是一个研究方向。

致　谢

本论文是在导师邱丽娟研究员的悉心指导下完成的，从论文选题、开发实现到论文写作、初稿修改、直至最后定稿，都倾注了邱老师大量的心血与精力。在攻读硕士学位期间，邱老师在学习、生活各方面都给予我无微不至的关怀与指导。邱老师严谨的学者风范、精益求精的治学精神、深厚的理论修养、平易近人的态度都令我收益匪浅，为我终身之楷模。

我在网络中心渡过了两年愉快的时光，得到两位辅导老师——许永诚老师和周南老师——全方位的教导，无论是基础课程的教授还是课题项目的开发，都可以体会得到他们的认真负责和高瞻远瞩。许永诚老师和周南老师在计算机领域的不同分支有着非凡的造诣，学高，身正，为我典范。同时要感谢网络中心的每一位老师，他们是：张晓泉老师、李娟老师、乔军老师、劳凤丹老师、杨莉莉老师、邱小斌老师和虞萍老师，他们中的每一个人都为我的学习和研究提供了很大帮助。在这两年中，我们中心研究生机房的所有成员团结友爱，互助向上，形成了良好的学术氛围，这个集体有着大家庭的和睦温暖。

在此，还一定要提及我的家人，最近的两年半时间里，他们承担了全部的家务和经济负担，尽全力为我提供了一个轻松的就读环境。

值此论文完成之际，谨向多年以来关心、爱护和支持我的人们致以诚挚的谢意！

参考文献

[1]、 UML Notation Guide version 1.1，Rational Software Corporation

[2]、 UML Semantics version 1.1，Rational Software Corporation

[3]、 UML Summary，Rational Software Corporation

[4]、 Applying Requirements Management with Use Cases，Rational Software Corporation

[5]、 Traceability Strategies for Managing Requirements with Use Cases，Rational Software Corporation

[6]、 Modeling Web Application Architectures with UML，Rational Software Corporation

[7]、 Developing Large-Scale Systems with the Rational Unified Process，Rational Software Corporation

[8]、 Unified Modeling Language for Real-Time Systems Design，Rational Software Corporation

[9]、 The Unified Modeling Language Reference Manual，Rational Software Corporation

[10]、 Functional Requirement and Uses Cases，By Ruth Malan，Hewlett-Packard

[11]、 Software Process，modeling and Supporting Enviroment，周智英、钱岭，清华大学

[12]、 COM Versus CORBA：A Decision Framework，QUOIN Inc.

[13]、 标准建模语言 UML 及其支持环境，北京航空航天大学软件工程研究所

[14]、 Rose2000e 教程，济南伟仕光电技术公司

[15]、 UML with Rational Rose 从入门到精通，Wendy Boggs 等著，邱仲潘等译

[16]、 实用软件工程，郑人杰、殷人昆、陶永雷，清华大学出版社

[17]、 UML 可视化程序扩展的研究及其实现，周建武，华东师范大学

[18]、 软件需求，kwiegers

[19]、 软件需求工程——方法及工具评述，卢梅、李明树，中国科学院软件研究所

[20]、 统一建模语言 UML 述评，邵维忠、梅宏，北京大学

[21]、 面向对象方法综述，陈小群

[22]、 面向对象的软件开发，原作：Linda M.Northrop，译：waterbird

[23]、 构件技术“应用”先行，仲萃豪

[24]、 建立面向管理的过程——CMM 2 级概述，宁家骏，国家信息中心数据库部，蒋善宁，北京华力拓软件工业有限公司

[25]、 计算机软件需求说明编制指南 GB9385-88，国家标准局

[26]、 计算机软件开发规范 GB8566-88，国家标准局

[27]、 Thinking in Java，Bruce eckel，Mind View，inc.

[28]、 Java 服务器程序设计，宋辉，江峰，清华大学出版社

[29]、 Java 语言 API 类库，清华大学出版社

[30]、 Java 2 从入门到精通，John Zukowski，电子工业出版社

[31]、 DB2 通用数据库实用指南，IBM

[32]、 DB2 通用数据库自学教程，清华大学出版社

[33]、 IBM VisualAge for Java，IBM

[34]、 Servlet and JSP Programming，IBM

[35]、 Using XML and XSL with IBM WebSphere 3.0，IBM

[36]、 Programmin Documentation，IBM

[37]、 Design and Implement Servlets，JSPs and EJBs for IBM WebSphere Application Server，

IBM
[38]、 WebSphere 快速入门，瞿裕忠 张剑锋 王丛刚 陈峥，东南大学计算机科学系
[39]、 Developing Server-Side Java Web Applications，Arthur Ryman，IBM
[40]、 现代远程教育工程教育资源建设技术规范（征求意见稿），北京师范大学