分类号_____ 密级_____

UDC_____

学 位 论 文

UML 的分析与研究及在合同管理信息系统设计中的应用

作 者 姓 名 雷瑛

指导导师姓名 高福祥教授

东北大学计算机系统研究所

申请学位级别 工程硕士 学 科 类 别 工 学

学科专业名称 计算机技术

论文提交日期 2004.10 论文答辩日期 2004.12.21

学位授予日期 _____ 答辩委员会主席 李晶皎

评 阅 人 刘浪涛 王剑

东 北 大 学

2004 年 10 月

UML 的分析与研究及在合同管理信息系统设计中的应用

摘 要

UML 是一种可视化建模语言，它使用图形表示法来表示系统的面向对象的分析与设计。一方面，它是一种对软件密集型系统的各个方面进行可视化、规格说明、构建和文档化的语言；另一方面，使用 UML 有助于用户与软件开发人员之间进行良好的沟通，使可靠的系统设计成为可能。

本文首先对介绍了面向对象的分析与设计过程和 UML 的发展背景，然后对 UML 建模语言的主要特点和表示法进行了分析和讨论，并介绍了 UML 的建模工具 Rational Rose，最后将 UML 语言应用于首钢公司合同管理信息系统的设计与开发过程中。

首钢公司合同管理信息系统是对首钢机动部和各分厂合同管理中的信息存储、预算管理、合同审核、数据维护及数据传送等进行管理的典型系统，不仅实现了对分布在各处的分厂的合同信息进行微机化、统一化的管理，而且还为首钢 ERP 系统的数据利用保留了接口。

该系统主要采用面向对象方法进行分析与设计，用 UML 语言为系统建模。在课题的实施过程中，首先采用用例图对机动部合同系统和分厂合同系统实现的功能进行分析。用例图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定义达成的共识。它不仅描述了待开发系统的功能需求，而且带动了需求分析之后各阶段的开发工作。然后利用交互图、活动图和状态图等对系统的动态行为建模，并在此基础上设计了系统的类图、构件图和部署图等静态行为模型。最后，利用 UML 建模语言对系统的 GUI 进行了设计。

在系统设计开发过程中，坚持实用性和科学性相结合的原则。同时，不断进行迭代开发，通过修改文档和可视化的模型来进一步满足用户的需求。

本系统采用 Microsoft SQL Server 2000 数据库作为后台数据库，选择 JAVA 作为开发工具。系统软件界面友好，操作简便易行，功能实用。

关键词 UML 建模 管理信息系统 静态行为模型 动态行为模型

The Analysis and Study of UML and Its Application in Design of Contract Management Information System

Abstract

UML is a kind of visual modeling language using diagrammatic visual models to express object-oriented analysis and design method for particular software system. On the one hand, it can make each side of software compactness system visible, specifiable, constructible and documental. On the other hand, UML can make for good communication between users and software development personals and make it possible to design reliable system.

At first, this thesis introduces the process of the object-oriented analysis and design (OOA&D) method and the UML development background. Secondly, it discusses UML's primary characteristics, application domain and expression, also introduces Rational Rose-an UML's modeling tool. Finally, the paper analyzes in detail how UML is applied in designing and developing Contract Management Information System of Shougang Corporation.

Shougang Corporation contract management information system is a typical system to manage these kinds of information such as contract' storing, budget management, contract auditing, data maintenance and transmission of Shougang Jidong department and many branch factories. Not only does it realizes computerized and unified management on contract information of many branch factories which spread all over Beijing, but also keep an interface for shared data of ERP system of Shougang Corporation.

We adopt object-oriented design method, and make models of this system with UML. In the process of designing and developing system, use case diagram is used for analyzing the Jidong department and branch factory's functions. Use case diagram is the result of extensive discussion between developers and users, and expresses consensus of developers and users for demand specification definition. It describes function demands of developing system and drives other phase's development following demand analysis phase. And then, we design system's dynamic action model with interaction diagram, activity diagram and state chart diagram, and express system's static action model with class diagram, component diagram and deployment diagram. In the end, we design system's GUI with UML.

In the process of designing and developing system, we adhere to practical and scientific principle. Meanwhile, we develop the system with iterative analysis method and modify files

and perfect visual models in order to meet users' requirements a deep degree.

This system is developed by JAVA and the database Microsoft SQL Server 2000 is also available, which makes user interface more friendly and has many advantages such as convenient operation, strong pertinence and practical function.

Key words UML, Modeling, Management Information System, static action model, dynamic action model

独创性声明

本人声明所呈交的学位论文是在导师的指导下完成的。论文中取 得的研究成果除加以标注和致谢的地方外，不包含其他人已经发表或撰写过的研究成果，也不包括本人为获得其他学位而使用过的材料。与我一同工作的同志对本研究所做的任何贡献均已在论文中作了明确的说明并表示谢意。

学位论文作者签名：雷振

日 期：2024.11

学位论文版权使用授权书

本学位论文作者和指导教师完全了解东北大学有关保留、使用学位论文的规定：即学校有权保留并向国家有关部门或机构送交论文的复印件和磁盘，允许论文被查阅和借阅。本人授权东北大学可以将学位论文的全部或部分内容编入有关数据库进行检索、交流。

(如作者和导师同意网上交流,请在下方签名;否则视为不同意。)

学位论文作者签名： 导师签名：

签字日期： 签字日期：

第一章 引 言

1.1 课题研究的意义

随着计算机技术的飞速发展，使各行各业对用计算机进行管理信息系统的开发需求与日俱增，在改革开放深入发展的今天，企业要适应计划经济向社会主义市场经济体制的转变这场伟大的革命，因而他们对信息的及时性、准确性、数据的共享性深感重要，迫切需要将企业质量信息进行系统的管理，快速实现数据资源的高度共享，为管理、经营的决策者提供更加及时、全面、准确的信息资源。由此，越来越多的企业开始开发自己的管理信息系统，以期使本企业的管理模式提高到一个新的水平。

在我国，经过十几年的管理信息系统软件开发实践，已有许多大型网络管理信息系统，如银行、税务、海关、证券期货业系统相继建立，许多中小企业用计算机代替手工进行业务处理，各种各样的管理软件应运而生，而对于一些较为规范、通用的管理，如以财务软件为代表的管理软件已形成了相当规模的商品化软件市场。在企业其它领域中，由于各企业之间的经营范围不同，加上各企业之间管理上的差异等各种原因，使得较成熟的集成化软件还为数不多，许多项目是针对某个部门的具体要求用手工方法开发的，工作量大，开发周期长，成本高，效率低，便于维护和通用的系统不多。

同时，管理信息系统的建立是一项复杂的系统工程，每个系统都有它自身的复杂性和特殊性。目前，由于开发高质量管理信息系统的能力大大落后计算机硬件日新月异的进展，加上社会对管理信息系统发展和完善需求的增加以及对管理信息系统开发过程中出现的错误认识和行为而导致管理信息系统开发的失败，这些情况已严重妨碍了计算机技术的进步。因此对管理信息系统有关的内容进行深入研究，提高工作效率，提高管理信息系统开发成功率已变得十分重要。

本项目目的就是为了解决现有系统存在的上述问题，对现有的手工管理的模式实行计算机化，使各分厂同公司机动部的联系网络化，使管理电子化。该系统让信息透明化，并使公司的目标一致，以谋求业绩处于最佳状态。这对追求全面附加价值最佳化的首钢集团来说，无疑是提升企业整体竞争力的最佳方案。

1.2 面向对象机制中的模型与建模过程

通过面向对象的方法进行软件开发已经成为主流，这是因为，这种方法对构建各种问题领域、各种规模和复杂程度的系统很有价值。此外，目前的大多数语言、操作系统和工具从某种方式来说都是面向对象的。

面向对象的分析就是依照用户所理解的真实世界中的对象和概念，发现和分析对象

的内部构成和外部关系，建立准确而简洁的软件系统的对象模型。

1.2.1 建模的角色

在现有的软件开发技术下，大多数软件的实现都是复杂的，并且相信在未来，软件开发只会变得更加复杂，因为要求用软件来解决的问题也越来越多，越来越复杂。

在这种面向对象设计中，开发人员创建的代码块称为对象，然后在各种应用程序中使用这些对象。需要修改其中一个对象时，开发人员只要在一个地方进行修改即可。各个公司纷纷采用这种新技术，将其集成到现有应用程序中。利用面向对象方法，把应用程序分成许多相互独立的对象，然后再组合这些对象，建立应用程序。可见，面向对象机制的一个主要优势在于可以一次性地建立组件，然后反复地使用。利用面向对象机制，可以开发出弹性很大和适应结构与功能变化的系统。

解决复杂问题最有效的理论是分层理论，即将问题分为多个问题逐一解决。

我们采用建模来理解复杂事物。我们的模型描述了我们所要建立的，我们正在建立的，和我们已经建立的系统。模型的使用贯穿了整个开发生命周期，而且还是关键性的工作。

模型是对现实世界的简化与抽象，它有助于对复杂问题进行分层，从而更好地解决问题。这就是为什么要对软件进行建模的根本原因。并且，有效的软件模型也有利于分工与专业化生产，从而节省生产成本。

结构是模型的有机结合，是一个系统的灵魂与主线。

一个有效的软件系统就是在系统结构的支撑与带动下解决复杂问题的技术系统。

这些理由与原因应该是每一个软件生产者都应该要领会的，从而应用到实际的软件开发过程中去。

将来，基本的构建软件的元素可能会是模型，而不是某一种高级语言，就象从汇编语言过渡到高级语言一样。高级语言将会由模型自动生成。

在面向对象的分析与设计方法中最引人关注的就是模型，模型在现代软件工程中占有重要的地位。为系统选择和建立模型是当今软件开发中所要做的最重要的工作之一。

体现系统某一方面特征或者从某一视角出发所看到的系统的某一侧面被称为模型。模型是有视角的，从不同的视角可以对同一个系统建立多个模型。因此，必须从多个视角出发建立系统的模型才能获得满足我们需要的对系统的描述。建立系统模型的过程被称为建模，描述模型的工具叫做建模语言。软件开发组织需要一种用模型进行沟通的方法，这种方法不仅用于项目组内部成员之间，还用于与外部的项目相关人员进行沟通。

建模的实质：

建模的一个关键好处就在于，它为项目组的成员提供了一个沟通渠道。如果没有合理的模型，项目组成员只能从各自的视角来理解系统，而一个项目是无法容忍这种个性化的视角的，例如需求的获取。如果确实这样做了，项目就会因为未能"理解"需求而终结。而无论项目规划如何，项目组都会因为未能满足某些人的要求而受到指责。

Booch 指出，建模过程应该完成下列四个目标：

辅助项目组使系统直观化，表现出其原有之风貌或投资者的设想；

辅助说明系统的结构或行为；

提供一个模板，指导系统的构建；

的内部构成和外部关系，建立准确而简洁的软件系统的对象模型。

1.2.1 建模的角色

在现有的软件开发技术下，大多数软件的实现都是复杂的，并且相信在未来，软件开发只会变得更加复杂，因为要求用软件来解决的问题也越来越多，越来越复杂。

在这种面向对象设计中，开发人员创建的代码块称为对象，然后在各种应用程序中使用这些对象。需要修改其中一个对象时，开发人员只要在一个地方进行修改即可。各个公司纷纷采用这种新技术，将其集成到现有应用程序中。利用面向对象方法，把应用程序分成许多相互独立的对象，然后再组合这些对象，建立应用程序。可见，面向对象机制的一个主要优势在于可以一次性地建立组件，然后反复地使用。利用面向对象机制，可以开发出弹性很大和适应结构与功能变化的系统。

解决复杂问题最有效的理论是分层理论，即将问题分为多个问题逐一解决。

我们采用建模来理解复杂事物。我们的模型描述了我们所要建立的，我们正在建立的，和我们已经建立的系统。模型的使用贯穿了整个开发生命周期，而且还是关键性的工作。

模型是对现实世界的简化与抽象，它有助于对复杂问题进行分层，从而更好地解决问题。这就是为什么要对软件进行建模的根本原因。并且，有效的软件模型也有利于分工与专业化生产，从而节省生产成本。

结构是模型的有机结合，是一个系统的灵魂与主线。

一个有效的软件系统就是在系统结构的支撑与带动下解决复杂问题的技术系统。

这些理由与原因应该是每一个软件生产者都应该要领会的，从而应用到实际的软件开发过程中去。

将来，基本的构建软件的元素可能会是模型，而不是某一种高级语言，就象从汇编语言过渡到高级语言一样。高级语言将会由模型自动生成。

在面向对象的分析与设计方法中最引人关注的就是模型，模型在现代软件工程中占有重要的地位。为系统选择和建立模型是当今软件开发中所要做的最重要的工作之一。

体现系统某一方面特征或者从某一视角出发所看到的系统的某一侧面被称为模型。模型是有视角的，从不同的视角可以对同一个系统建立多个模型。因此，必须从多个视角出发建立系统的模型才能获得满足我们需要的对系统的描述。建立系统模型的过程被称为建模，描述模型的工具叫做建模语言。软件开发组织需要一种用模型进行沟通的方法，这种方法不仅用于项目组内部成员之间，还用于与外部的项目相关人员进行沟通。

建模的实质：

建模的一个关键好处就在于，它为项目组的成员提供了一个沟通渠道。如果没有合理的模型，项目组成员只能从各自的视角来理解系统，而一个项目是无法容忍这种个性化的视角的，例如需求的获取。如果确实这样做了，项目就会因为未能"理解"需求而终结。而无论项目规划如何，项目组都会因为未能满足某些人的要求而受到指责。

Booch 指出，建模过程应该完成下列四个目标：

辅助项目组使系统直观化，表现出其原有之风貌或投资者的设想；

辅助说明系统的结构或行为；

提供一个模板，指导系统的构建；

提供一个模板，指导系统的构建；

将项目开发组的决策付诸于文档。

所有这些目标都反映出一个共同的主题：保持良好的沟通机制。没有良好的沟通，项目就会失败。UML 满足了这些目标，并不限于此。

1.2.2 建模过程

统一建模语言 UML 等仅仅是一种建模语言。它不是系统设计方法，而是一种系统建模方法。UML 的创始者们提出了一种面向对性的软件开发过程，称为 Rational 统一（Rational Unified Process,RUP），如图 1.1 所示。

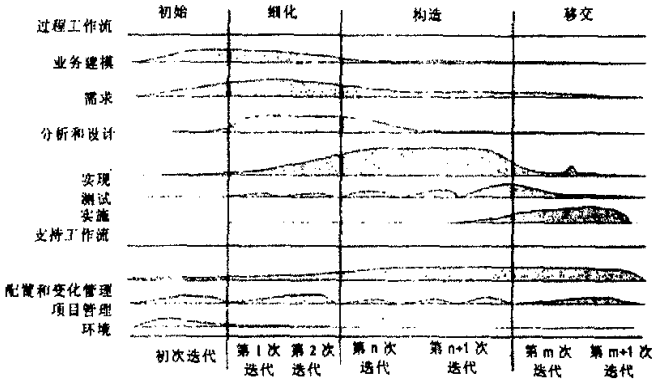

图 1.1 RUP 的软件开发生命周期

Fig.1.1 Software developing cycle of RUP

软件开发过程是将产品或者系统从概念形成为成品所遵循的一系列阶段：

(1) 初始阶段（Inception）

是进行最初分析的阶段，确定要开发的系统，包括其内容及业务。在本阶段，应该确定出业务需求建立用例图模型。

(2) 细化阶段（Elaboration）

是进行设计的阶段，实施详细的设计，确定系统的功能。在细化阶段，应把初始阶段发现的用例发展成为子系统和与之相关的业务对象的设计。从这一阶段，会反复进行以便将模型变为一个用更为细化的模型图表示的软件设计，这些模型将最终建立类和类成员的模型。

(3) 构造阶段（Construction）

从系统设计进行实际的软件产品构建的阶段。在构造阶段会提出修改设计的请求，对分析提出一些问题。

(4) 移交阶段（Transition）

将系统交付给用户。

应当明确的是，建模的任务并不是自动生成代码，也不是通过逆向工程生成文档。模型和建模的真正价值在于它通过一个特殊的视点来看一个系统的简化形式的能力，通过这个观点，系统变得更加易于理解。如果模型跟建模的事物一样复杂，那么建模几乎没有意义。

1.3 用 UML 建模合同管理信息系统

开发工程合同管理系统，着重实现企业需要用计算机进行工程合同管理的需求，以减少人工复杂、烦琐的管理工作，充分利用计算机高速信息传递和数据共享这一现代化的管理工具，提高管理的水平。为严格的质量控制，精确快速的数据统计分析，为领导决策和降低工程成本提供可靠的手段和保证。

合同管理涉及到签订合同和执行合同。业主在签定施工合同时，应根据工程特点、规模、工期、施工难易程度来确定合同形式。施工工期短(一年左右时间)的简单项目适宜固定总价合同(也称闭口合同)。施工工期长，复杂项目适宜单价合同(也称开口合同)。比较常见的是中标价加变更签认。无论采用哪种形式的施工合同，合同中都应明确工程价款是否调整、调整依据、调整方式、结算方式以及有关合同价款争议的解决方法，以减少投资控制的纠纷及隐患。在执行合同过程中，业主应注意按合同办事，按程序办事。该监理工程师管的事情，全权委托，决不插手。切忌自以为是，杜绝随意变更。工程确实需要变更之处，一定有文字记录，涉及到工程质量标准及其它实质性变更，一定经设计院同意。任何变更，都应及时计量、及时签认。由变更引起的价格调整，应及时确认调整价格，避免事后纠缠不清。事实证明，在施工过程中及时做好中间结算，可大大减少竣工结算的工作量，收到事半功倍的效果。

合同管理系统是根据首钢内部工程合同管理的基本原理和实施过程，体贴客户的实际需求和工作习惯，采用大量先进管理思想和技术开发而成的，是办公自动化于一体的应用型软件。

合同管理系统本质上是一套应用于首钢内部的工程合同管理与工作流控制软件，并可扩充为首钢内部的工程合同的全面管理信息系统。其作用如下：

① 帮助企业快速查找某个合同，迅速获得合同内容。

② 在合同实施前后都能认真而有效的贯彻执行，对合同进行有效的控制，充分发挥合同管理系统应发挥的效益与作用。

③ 避免重复结算而带来的巨大损失。

④ 提供一种质量管理与工作流程管理的强有力手段，提高管理水平。

鉴于 UML 的强大功能和适合于建模面向对象系统的优势，我们在开发合同管理信息系统时就是基于 UML 建模的。

该系统是针对首钢总公司内部工程合同管理开发的一个系统。首钢许多的工厂，比如：二炼钢、计算机公司等的合同数据要上报首钢总公司机动部进行审批，汇总。随着市场供求需求的不断增加，其规模也将不断扩大。为了充分利用计算机高速信息传递和数据共享这一现代化的管理工具，提高管理的工作效率；同时实现合同管理科学、高效的运营和管理，开发一套既方便操作、又合理健壮特别是具备敏捷化的合同管理信息系统显得及时而又必要。以此来提高劳动效率及管理水平，使企业全方位受益。

由于数据量大，工作繁杂，手工操作极易出错。况且首钢公司像二炼钢这样的厂有 50 多个厂归首钢机动部管理，工程合同数据都要上报机动部。机动部也要求利用该软件（此软件与下边各厂不尽相同），能高效地进行合同管理。同时，所用数据是首钢总公司要应用的 ERP 的基础数据，要求能与 ERP 做无缝连接（只需要接口模块），与 ERP 建设不会冲突，能为 ERP 提供基础数据。

在该信息系统的设计和开发过程中，如何科学、合理地进行开发是我们首先思考的问题。目前在信息系统分析与建模领域主要的方法有：IDEF 方法、OMT 方法和 UML 方法。UML 融合了三种主要的面向对象技术 Booch、OMT 和 OOSE 中的精华，UML 目前已成为软件工业界事实上的标准。UML 是一种可视化的建模语言，它能让系统构造者用标准的、易于理解的方式建立起能够表达出他们想象力的系统蓝图，并且提供了便于不同的人之间有效地共享和交流设计结果的机制。它能支持用不同实现技术进行的软件开发全过程。

合同管理信息系统主要采用面向对象方法进行开发设计，用 UML 语言为系统建模，将系统分解为一些功能模块，以便客户、系统分析员、数据库设计者和程序开发者更好地理解和管理。在系统设计中，坚持实用性和科学性相结合的原则。同时，不断进行迭代开发，通过修改文档和可视化的模型来进一步满足用户的需求。

实践证明，用 UML 进行管理信息系统开发具有很多优势。一方面，UML 综合了以前流行于面向对象开发领域的 Booch、OMT 和 OOSE 等方法的优点，为开发者和开发工具建立了统一的、易于理解的图形和符号的标准。另一方面，它的强大的功能使其能够贯穿于面向对象分析、面向对象设计、面向对象实现的整个过程之中，其所附带的各种类型的视图能够从不同应用层次、不同角度为系统从分析、设计直到实现提供支持。UML 使可靠的系统设计成为可能。

该系统的技术特点如下：

(1) 提供了合同管理信息系统的计算机应用平台，实现了公司机动部对分布在各处的分厂的管理控制，保证了整个信息系统的协同工作。

(2) 系统功能齐全，提供了合同信息管理、闭口合同管理、预算信息管理、字典管理、系统设置管理、施工单位管理、合同信息查询、预算信息查询、及各级各类查询、打印、数据备份、数据分割、数据合并等诸多功能。

(3) 系统安全可靠，对不同管理层次、不同业务分工的用户进行系统功能授权，依据系统密级不同进行不同权限的操作，从而保证了系统的安全性和可靠性。

(4) 系统具有良好的可扩展性，该系统的网络设计保证了增加其他分厂时，系统可以灵活方便地进行扩展。

(5) 应用 UML 对系统建模，保证了对象模块的快速开发，使模块和代码具有可重用性，缩短了开发周期。

(6) 通过建模 GUI，开发出用户易于理解和满意的操作界面，成为该系统深受欢迎的一个主要方面。

总之，基于 UML 建模实现该管理信息系统的开发，不仅保证了理解用户需求，而且保证了开发过程的科学性、合理性，因而对其它合同系统的开发具有一定的借鉴价值。

第二章 统一建模语言（UML）的分析

2.1 UML 的发展背景

统一的建模语言 UML（Unified Modeling Language） 是 Grady Booch、James Rumbaugh 和 Ivar Jacobson 智慧的结晶。该方法结合了 Booch, OMT, 和 Jacobson 方法的优点，统一了符号体系，并从其它的方法和工程实践中吸收了许多经过实际检验的概念和技术。

2.1.1 Booch 方法

Booch 方法的过程包括以下步骤：
在给定的抽象层次上识别类和对象
识别这些对象和类的语义
识别这些类和对象之间的关系
实现类和对象
这四种活动不仅仅是一个简单的步骤序列，而是对系统的逻辑和物理视图不断细化的迭代和渐增的开发过程。

在类和对象的实现阶段要考虑如何用选定的编程语言实现，如何将类和对象组织成模块。

在面向对象的设计方法中，Booch 强调基于类和对象的系统逻辑视图与基于模块和进程的系统物理视图之间的区别。他还区别了系统的静态和动态模型。然而，他的方法偏向于系统的静态描述，对动态描述支持较少。

Booch 建议在设计的初期可以用符号体系的一个子集，随后不断添加细节。对每一个符号体系还有一个文本的形式，由每一个主要结构的描述模板组成。符号体系由大量的图符定义，但是，其语法和语义并没有严格地定义。

2.1.2 OMT 方法

Rumbaugh 的 OMT 方法从三个视角描述系统，相应地提供了三种模型，对象模型，动态模型和功能模型。

该方法将开发过程分为四个阶段：

(1) 分析

基于问题和用户需求的描述，建立现实世界的模型。分析阶段的产物有：
问题描述
对象模型＝对象图＋数据词典
动态模型＝状态图＋全局事件流图
功能模型＝数据流图＋约束

⑵ 系统设计

结合问题域的知识和目标系统的体系结构（求解域），将目标系统分解为子系统。

⑶ 对象设计

基于分析模型和求解域中的体系结构等添加的实现细节，完成系统设计。主要产物包括：

细化的对象模型

细化的动态模型

细化的功能模型

⑷ 实现

将设计转换为特定的编程语言或硬件，同时保持可追踪性、灵活性和可扩展性。

2.1.3 Coad/Yourdon 方法

Coad/Yourdon 方法严格区分了面向对象分析 OOA 和面向对象设计 OOD。该方法利用五个层次和活动定义和记录系统行为，输入和输出。这五个层次的活动包括：

发现类及对象。描述如何发现类及对象。从应用领域开始识别类及对象，形成整个应用的基础，然后，据此分析系统的责任。

识别结构。该阶段分为两个步骤。第一，识别一般——特殊结构，该结构捕获了识别出的类的层次结构；第二，识别整体——部分结构，该结构用来表示一个对象如何成为另一个对象的一部分，以及多个对象如何组装成更大的对象。

定义主题。主题由一组类及对象组成，用于将类及对象模型划分为更大的单位，便于理解。

定义属性。其中包括定义类的实例（对象）之间的实例连接。

定义服务。其中包括定义对象之间的消息连接。

在面向对象分析阶段，经过五个层次的活动后的结果是一个分成五个层次的问题域模型，包括主题、类及对象、结构、属性和服务五个层次，由类及对象图表示。五个层次活动的顺序并不重要。

2.1.4 Jacobson 方法

Jacobson 方法与上述三种方法有所不同，它涉及到整个软件生命周期，包括需求分析、设计、实现和测试等四个阶段。需求分析和设计密切相关。需求分析阶段的活动包括定义潜在的角色（角色指使用系统的人和与系统互相作用的软、硬件环境），识别问题域中的对象和关系，基于需求规范说明和角色的需要发现 use case，详细描述 use case。设计阶段包括两个主要活动，从需求分析模型中发现设计对象，以及针对实现环境调整设计模型。第一个活动包括从 use case 的描述发现设计对象，并描述对象的属性、行为和关联。在这里还要把 use case 的行为分派给对象。

在需求分析阶段的识别领域对象和关系的活动中，开发人员识别类、属性和关系。关系包括继承、熟悉（关联）、组成（聚集）和通信关联。定义 use case 的活动和识别设计对象的活动，两个活动共同完成行为的描述。Jacobson 方法还将对象区分为语义对象（领域对象）、界面对象（如用户界面对象）和控制对象（处理界面对象和领域对象之间的控制）。

在该方法中的一个关键概念就是 use case。use case 是指行为相关的事务（transaction）序列，该序列将由用户在与系统对话中执行。因此，每一个 use case 就是一个使用系统的方式，当用户给定一个输入，就执行一个 use case 的实例并引发执行属于该 use case 的一个事务。

2.1.5 UML

UML 代表统一建模语言(Unified Modeling Language)，它是一种标准的软件建模语言，是一种用于对软件系统的模型绘制可视化的标准蓝图或者以图表的方式对所加工的产品进行可视化描述的工具。换句话说，它是"软件分析与设计中的标准语言"。

面向对象技术是软件工程领域中的重要技术，它不仅是一种程序设计方法，更重要的是，它是一种对真实世界的抽象思维方式。尽管 UML 可以应用于任何开发方法中对任何系统建立模型，但它特别适合采用面向对象的思维方式对软件建模。现代软件项目的规模越来越大，越来越复杂，建立简明准确的模型是把握系统的关键。

在 UML 统一建模语言出现以前，没有一种占统治地位的建模语言。各种语言各有特色，用户必须选择几种类似的建模语言，以完成复杂的体系结构描述。大部分建模语言都有一些主要的、共同的概念，而在描述和表达方面却又有所不同，缺乏一种强大的具有扩展能力的建模语言，给使用者带来许多麻烦，不利于软件的推广和重用。由 Rational 公司的专家 Grady Booch、James Rumbaugh 和 Ivar Jacobson 联合完成的统一建模语言，是当前主流的面向对象的软件开发建模语言，也是最有前途的建模语言。UML 的发展如图 2.1 所示。

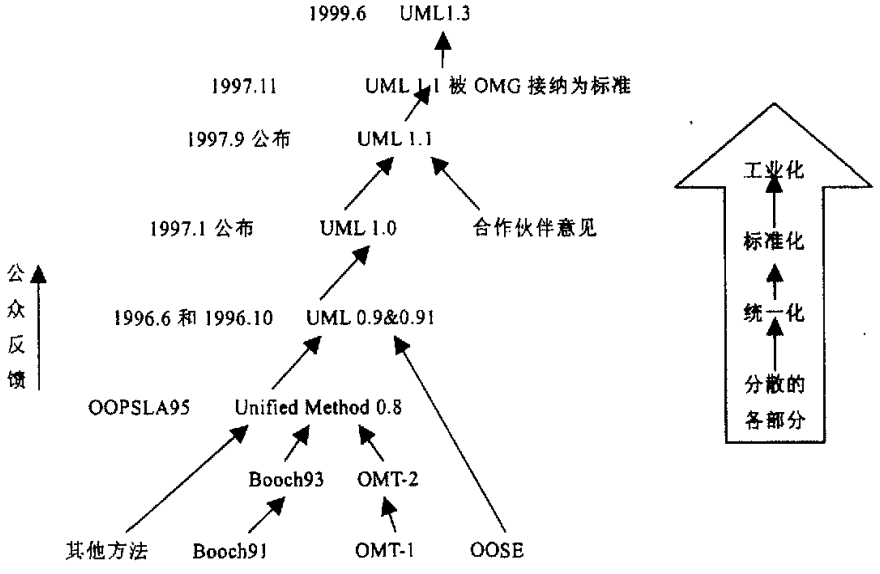

图 2.1 UML 的发展

Fig.2.1 Diagram of UML's development process

总之，UML 是一种定义良好、易于表达、功能强大且普遍适用的建模语言。它溶入了软件工程领域的新思想、新方法和新技术。

2.2 UML 的主要特点

2.2.1 建模方法的特点比较

面向对象技术从七十年代末逐渐兴起，到现在已占据了软件开发领域的主导地位。其中，建模被视为面向对象分析和设计的核心，也是分析和设计过程中最基本和最关键的活动之一。面向对象的分析和设计应当从建模开始，建模语言则一直是面向对象技术研究的重点。

九十年代中期，出现了一批新的建模方法，其中最引人注目的有 Booch1993、OMT、OOSE 和统一建模语言 UML。

Booch 是面向对象方法的最早倡导者之一，他在 1986 年描述了面向对象软件开发的基础问题，提出了面向对象软件工程的概念。1991 年，他在"Object-Oriented Design"一书中将先前面向 Ada 的工作扩展到整个面向对象领域，他认为建立模型对于复杂系统的构造是非常重要的。Booch 提出了面向对象开发的 4 个模型：用于描述逻辑结构的逻辑模型，用于描述物理结构的物理模型，用于描述静态语义的静态模型和用于描述动态语义的动态模型。

他对继承和类的阐述特别值得借鉴。他还开发了表示模型各个细节的图注方法。例如，图注中把对象画成云图，表示对象几乎无所不包。Booch 方法还用各种箭头表示对象之间的关系类型。Booch 方法对面对象系统的设计提供了详细的指导，但对分析阶段的描述则相对较简单。

OMT(对象建模技术)方法来自 James Rumbaugh 博士，该方法采用了面向对象的概念并引入各种独立于语言的表示符号。该方法用对象模型、动态模型、功能模型共同完成对整个系统的建模，所定义的概念和符号可用于软件开发的分析、设计和实现的全过程，软件开发人员不必在开发过程的不同阶段进行概念和符号的转换。OMT 使用比 Booch 更简单的图形表示系统。OMT 适用于分析和描述以数据为中心的信息系统。

Jacobson 于 1994 年提出了 OOSE(Object-Oriented Software Engineering，面向对象软件工程)方法，其最大特点是面向用例，并在用例的描述中引入了外部角色的概念。用例的概念是精确描述需求的重要武器，但用例贯穿于整个开发过程，包括对系统的测试和验证。目前在学术界和工业界已普遍接受了用例这一概念，并认为是面向对象技术走向第二代的标志。OOSE 比较适合于支持商业过程和需求分析。

20 世纪 90 年代，3 个最流行的面向对象方法是：OMT 方法、Booch 方法和 OOSE 方法，每个方法都有自己的价值和重点。分析是 OMT 方法的强项，设计是 OMT 方法的弱项；设计是 Booch 方法的强项，分析是 Booch 方法的弱项；Jacobson 擅长行为分析，而在其他方面比较弱。

各种建模语言，实际上各有千秋，但仍存在着某些差别，这极大地妨碍了用户之间的交流。因此有必要在精心比较不同建模语言优缺点及总结面向对象应用技术实践的基

础上，根据应用需求，取其精华，努力统一建模语言。人们普遍认为，软件工程领域在最近的几年内取得了前所未有的进展，其成果超过软件工程领域过去十几年来的成就总和。其中最重要的成果之一就是统一建模语言 UML 的出现。

UML 是由著名的面向对象技术专家 Grady Booch、James Rumbaugh 和 Ivar Jacobson 发起，在 Booch 表示法、OMT 方法和 OOSE 方法的基础上，广泛征求意见，集众家之长，反复修改而完成的。在美国，目前已有几千家公司表示支持采用 UML 作为建模语言。UML 不仅融合了 Booch 表示法、OMT 方法和 OOSE 方法中的基本概念，而且还对现有方法的应用范围进行扩展，这是 UML 受到企业界和用户普遍欢迎的一个主要原因。

2.2.2 标准建模语言 UML 的主要特点

统一建模语言 UML 的主要特点有：

(1) 统一标准

UML 融合了当今一些流行的面向对象开发方法的主要概念和技术，成为一种面向对象的标准化的统一的建模语言，结束了以往各种方法的建模的不一致和差别。

(2) 面向对象

UML 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。UML 考虑了各种方法的图形表示，删掉了大量易引起混乱的、多余的和极少使用的符号，也添加了一些新符号。

(3) 可视化、表示能力强大

UML 是一种图形化的语言，系统的逻辑模型或实现模型都能用 UML 的模型图形清晰地表示。另外，UML 不只是一堆图形符号，在每一个 UML 的图形表示符号背后，都有良好定义的语义。UML 还可以处理与软件的说明和文档有关的问题，包括需求说明、体系结构、设计、源代码、项目计划、测试、原型、发布等。

UML 的强大表示能力使它可以用于各种复杂类型的软件系统的建模。

(4) 独立于过程

UML 是系统建模语言，独立于开发过程。虽然 UML 与 Rational 统一过程配合使用，将发挥强大的作用，但是 UML 也可以在其他面向对象的开发过程中使用，甚至在常规的软件生命周期法中使用。

使用 UML 进行软件系统的分析和设计，能够加速软件开发的进程，提高代码的质量，支持变动的业务需求。UML 适用于各种大小规模的软件系统项目，能促进软件复用，方便地集成已有的系统软件资源。

2.2.3 标准建模语言 UML 的应用领域

UML 独立于任何一种开发过程，这意味着用户在使用 UML 进行建模时，可以选用任何适合的过程。过程的选用与软件开发过程的不同因素有关。UML 的重点是标准建模语言，不是标准过程。UML 的开发者提出的开发过程是用例驱动、以体系结构为中心、迭代递增的过程，这可以说是一个元过程。

UML 的目标是以面向对象图形的方式来描述任何类型的系统，具有很宽的应用范

围。它可以用于描述机械系统、一个企业的机构或企业过程的信息系统、具有实时要求的工业系统或工业过程、嵌入式实时系统、分布式系统、系统软件及商业系统等。

此外，UML 适用于系统开发过程中从需求规格描述到系统完成后测试的不同阶段。在需求分析阶段，可以用用例来捕获用户需求。通过用例建模，描述对系统感兴趣的外部角色及其对系统（用例）的功能要求。分析阶段主要关心问题域中的主要概念（如抽象、类和对象等）和机制，需要识别这些类以及它们相互间的关系，并用 UML 类图来描述。为实现用例，类之间需要协作，这可以用 UML 动态模型来描述。在分析阶段，只对问题域的对象（现实世界的概念）建模，而不考虑定义软件系统中技术细节的类（如处理用户接口、数据库、通讯和并行性等问题的类）。这些技术细节将在设计阶段引入，因此设计阶段为构造阶段提供更详细的规格说明。

编程（构造）是一个独立的阶段，其任务是用面向对象编程语言将来自设计阶段的类转换成实际的代码。在用 UML 建立分析和设计模型时，应尽量避免考虑把模型转换成某种特定的编程语言。因为在早期阶段，模型仅仅是理解和分析系统结构的工具，过早考虑编码问题十分不利于建立简单正确的模型。

UML 模型还可作为测试阶段的依据。系统通常需要经过单元测试、集成测试、系统测试和验收测试。不同的测试小组使用不同的 UML 图作为测试依据：单元测试使用类图和类规格说明；集成测试使用部件图和合作图；系统测试使用用例图来验证系统的行为；验收测试由用户进行，以验证系统测试的结果是否满足在分析阶段确定的需求。

总之，标准建模语言 UML 适用于以面向对象技术来描述任何类型的系统，而且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。

2.2 基于 UML 设计的基本目标

在建立一个信息管理和事务处理的模型及文档系统时，可以利用面向对象的分析和设计方法建立基于 UML 的模型。为了构造成功的系统，一个健全的模型是很重要的，它将总的系统计划和整个开发组联系在一起。像其他任何语言一样，UML 有它自己的组成和原则，可用来构造不同模型模式。

基于 UML 设计的基本目标是：

- 向用户提供了一种随时可用的可视化模型语言以便他们能够开发和交换有意义的模型。
- 为扩展核心概念提供了扩展和规范机制。
- 独立于特定的编程语言和开发过程。
- 为理解模型语言提供了正式基础。
- 鼓励面向对象工具市场的发展。
- 支持较高层次的开发概念，如合作、框架、模式和组件。
- 集成最好的实践经验。

基于以上所有理由，UML 是设计面向对象系统的首选语言。

UML 并不只是实体关系图的替代物，而是包含了许多组成部分，它们结合在一起提供了一个完整的面向对象的开发环境。

2.3 UML 模型图语义及应用

UML 包括了一些可以相互组合图表的图形元素。UML 是一种语言，它具备组合这些元素的法规。

UML 提供了五个观点，可用来描述软件系统的架构，每个观点都有其特殊的定义，并提供一种以上的模型图来配合分析师做系统的分析与设计，并称之为 4+1 观点，如图 2.2 所示。

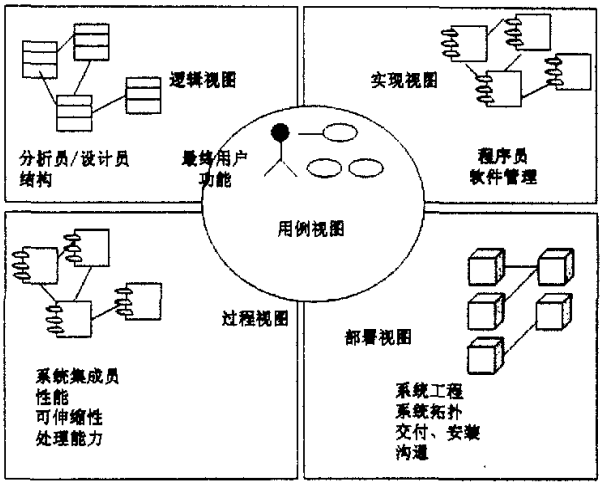

图 2.2　UML 体系结构视图

Fig.2.2 View model of UML system

(1) 逻辑视图是系统体系结构视图，提供问题的解决方案。

(2) 实现视图包含实现模型的概括，并且以模块形式组成包和层。

(3) 过程视图描述了相关任务（进程和线程），以及它们之间的交互和架构，设计对象和对象类对任务的分配。

(4) 部署视图描述典型操作平台上的物理接点，以及任务对物理节点的分配。

而最后一个观点为 Use Case 观点，上述的 4 个观点都是依据此观点，贯通而产生的，因此 4+1 的观点形成了密不可分的关系。

统一建模语言 UML 的模型可以分为下列五类（共 9 种模型）。

2.3.1 用例图(Use Case Diagrams)

(1) 用例图的语义

用例是从用户的观点对系统需求进行描述。对于系统开发人员来说，用例是一个有价值的工具，即它是用来从用户的观察角度搜集系统需求的靠得住的一项技术。

用例是一个被称作参与者的实体所发起的场景的集合。用例的执行必须对发起该用例的参与者或者其他参与者产生影响。

用例可以被重用。包含用例(Include)是将一个用例中的步骤作为另一个用例的步骤序列的一部分，使用包含用例可以避免重复描述同样的事件流。扩展用例(Extend)是通过对现有的用例增加新的步骤来创建新的用例。

与用户会谈是导出用例的最好方式。

(2) 用例图的表示法

用例图是包括参与者、用例及参与者与用例之间的关联、用例间关系以及参与者的泛化组成的模型图。

用例图的表示法很直观。用例用一个椭圆形表示，直立人形图标表示参与者。参与者的名字放在参与者图标的下方，用例的名字可以放在椭圆形里面也可以放在椭圆形下面。关联线连接参与者和用例，并且表示参与者与用例之间有通信关系。关联线是实线，和类之间的关联线类似。

一个简单的用例图的样例如图 2.3 所示。

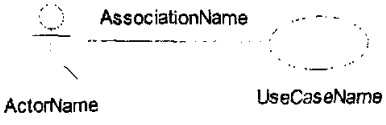

图 2.3　用例图样例

Fig.2.3 An example of use case diagram

(3) 用例图在开发中的作用

为系统定义的用例是整个开发过程的基础。这些用例提供贯穿系统的统一线程，并定义系统的行为。用例在每个核心过程工作流中都发挥作用：

- 用例模型是需求工作流的结果。在此早期过程中，需要通过用例对从用户的观点来看系统应该做什么这一内容进行模型化。
- 在分析和设计阶段，在设计模型中实现用例。应该创建用例实现，描述如何在设计模型中从交互对象的角度执行用例。此模型从设计对象的角度描述所实现的系统的不同部分，并描述各部分应该如何交互来执行用例。
- 在实现阶段，设计模型包括在实现规格说明中。因为用例是设计模型的基础，所以，需要根据设计类来实现用例。
- 在测试阶段，用例构成了确定测试用例和测试过程的基础。通过执行每个用例来验证系统。
- 用例还有其他作用：
  - 可作为规划迭代开发的基础。
  - 可作为用户手册中描述的内容的基础。
  - 定义单元排序。例如，客户得到的系统可能是通过某种组合的用例配置的系统。

2.3.2 静态图（Static Diagrams）

静态图包括类图和对象图。

2.3.2.1 类图（Class Diagrams）

类图描述系统中类的静态结构。类图不但定义了系统的类，表示了类之间的联系（如关联、依赖、聚合等），还描述了类的内部结构（类的属性和操作）。类图描述的是一种静态关系，在系统的整个生命周期中都是有效的。

(1) 类图的语义

类图是 UML 和许多面向对象方法的关键，随着使用 UML 进行数据库设计的出现，类图变得更加重要。现在，类图成为共用的中心图，应用程序和数据库开发者把它作为他们设计的基础。

Class(类)：类是对具有相似结构、行为和关系的一组对象的描述。UML 对类提供了 3 种图形表示符。第一种是细节抑制方式，只在一个方框中给出类名；第二种是分析级细节方式，在上、中、下三栏分别给出类名、属性名、操作名；第三种是实现级细节方式，给出了更多的细节。

类名：定义了类符号的名字栏的书写规范。

属性：规定了属性的写法，以及 3 种可见性符号："+"表示 public(公共)，"#"表示 protected(保护)，"-"表示 private(私有)。

操作：规定了操作的写法，采用与属性相同的 3 种可见性符号。

(2) 类图的表示法

通常，为了表示一个完整的静态视图，需要几个类图。每个独立的类图需要说明基础模型中的划分，即某些逻辑划分，如包是构成该图的自然边界。

一个类图的样例如图 2.4 所示。

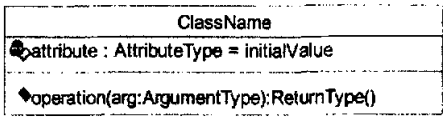

图 2.4　类图样例

Fig.2.4 An example of class diagram

在上面的类图中，属性前面的图标表示该属性是私有属性。类操作前面的图标表示该操作是公共操作。

2.3.2.2 对象图(Object Diagrams)

对象图是类图的实例，使用与类图类似的标识。他们的不同点在于对象图显示类的多个对象实例，而不是实际的类。一个对象图是类图的一个实例。由于对象存在生命周期，因此对象图只能在系统某一时间段存在。

(1) 对象图的语义

对象(object)是一个类的实例，是具有具体属性值和行为的一个具体事物。

对象图显示某些时刻对象与对象之间的关系，比如对象是类的实体，那么对象就是将类图中的类换成该类的实体——对象，此时的这个图就是对象图。对象图和协作图相关，协作图显示处于语境中的对象模型。

(2) 对象图的表示法

由于对象是类的实体，因而，在类图中将属性和操作具体化就形成了对象图。对象

图本身并不显示系统的演化过程。

对象图的图标也是一个矩形，和类的图标一样，但是对象名下面要带下划线。具体实例的名字位于冒号的左边，而该实例所属的类名位于冒号的右边。

一个对象图的样例如图 2.5 所示。

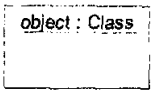

图 2.5 对象图样例

Fig.2.5 An example of object diagram

2.3.3 行为图（Behavior Diagrams）

行为图描述了系统的动态模型和系统对象间的交互关系。行为图包括状态图和活动图。两者都显示参与使用案例流程的对象和对象之间发送的消息。前者按时间排序，后者按对象本身来组织。下面就来介绍这两种交互图。

2.3.3.1 状态图(Statechart Diagrams)

状态图描述了类的对象所有可能的状态以及事件发生时状态的跃迁条件。状态图通常是对类图的补充，在实际应用中并不需要为所有的类图画状态图，仅为那些有多个状态且起行为受外界环境的影响并发生改变的类图画状态图。

(1) 状态图的语义

一种表征系统变化的方法是对象改变了自己的状态以响应事件和时间的流逝。UML 的状态图能够展示这种变化。它描述了一个对象所处的可能状态以及状态之间的转移，并给出了状态变化的起点和终点。

状态图与类图、对象图和用例图有着本质的不同。后三种图能够对一个系统或者至少是一组类、对象或用例建立模型。而状态图只是对单个对象建立模型。

UML 必须包括状态图，因为它能帮助分析员、设计员和开发人员理解系统中各个对象的行为。开发人员尤其需要知道对象是如何体现各自的行为的，因为他们要用软件实施这些行为。只实施对象的静态特征是不够的；开发人员必须要让对象能够做一些事情。

(2) 状态图的表示法

状态图的焦点是一个对象的状态变化。

状态用一个圆角矩形表示，状态转移用带箭头的实线表示，它指向目标状态。

状态图中要写明状态名，并且可以包括状态变量和活动的列表。转移可能作为对触发事件的响应而发生的，并且需要一个活动。转移也可能因为状态中的活动的完成而引起；这种方式发生的转移叫做无触发器转移。最后，转移还可能起因于一个特定条件(监视条件)的满足而引起。

有时候状态可以包含子状态。子状态可能是顺序的(一个接着一个地发生)或者是并发的(同时发生)。包含子状态的状态被称为组成状态。状态图中有时候还包括历史状态。历史状态是说明一个组成状态在对象转移出该组成状态之后还能够记住的子状态。历史状态可能是浅的也可能是深的。这个术语和嵌套的子状态有关。浅的历史状态只记忆了最顶层的子状态。而深的历史状态能够记忆所有层次的子状态。

当一个对象向另一个对象发送消息时，就触发了第二个对象的状态图中的某个转移，

这个消息就被称为信号。使用扩展的构造型<<Signal>>的类图标，可以建立信号的继承层次。

状态图的样例如图 2.6 所示。

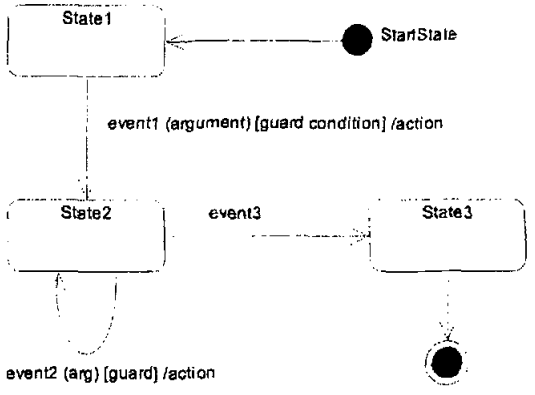

图 2.6 状态图样例

Fig.2.6 An example of statechart diagram

2.3.3.2 **活动图**(Activity Diagrams)

活动图描述了满足用例要求所要进行的活动以及活动间的约束关系，活动图有利于识别并发活动。

(1) 活动图的语义

活动图被设计用于简化描述一个过程或者操作的工作步骤。它是状态图的一种扩展形式。状态图显示出一个对象的状态并用状态之间的箭头连线来表示活动。而活动图则突出了活动。

UML 的活动图和旧的流图很类似。它显示出活动、判定点和分支。用于表达一个对象的操作和一个业务过程。

活动图表示一个程序或工作流。活动图通常出现在设计的前期，即在所有实现决定前出现，特别是在对象被指定执行所有的活动前，其状态代表活动的执行，就像一个计算机或真实世界不间断的操作，而转换由状态内活动的完成来触发(若有约束条件，可能有几个可能不同的出口)。

(2) 活动图的表示法

几个概念的说明：

判定：一个活动序列几乎总是要到达某一点，在这一点处要做出判定。一组条件引发一条执行路径，另一组条件则引发另一条执行路径，并且这两条执行路径是互斥的。

动态并发性：具有动态并发性的活动状态表示并发执行多个独立的计算。活动与一个参量表集合同时被调用。集合中的每一个成员都是活动的并行调用的参量表。调用是相互独立的，当所有的调用完成时，活动结束并触发它的完成转换。

对象流：有时，查看一下操作和作为它的参量值或结果的对象之间的关系有好处的。一个操作的输入和输出可以表示成一个对象流状态。它是一个状态的构造型，表示在计算过程中特定点的给定类的对象流状态。

泳道：活动图中可以增加角色的可视化维数。要对角色可视化，应该将图分割成多个平行的段，这些段被称为泳道(swimlane)。每个泳道的顶部可以显示出角色名，每个角色负责的活动放在各个角色的泳道中。一个泳道到另一个泳道之间可以发生转移。

活动图是对状态图的扩展。状态图突出显示的是状态，状态之间的转移箭头代表的是活动。而活动图突出显示的是活动。每个活动的图标被表示为圆角矩形，比状态图标更扁更接近椭圆。活动图的起始点和终止点图符和状态图一样。

当一个活动路径分成两个或多个路径时，可以用一个与路径垂直的粗实心线来代表路径的分支，两个并发路径的合并可以用相同方式表达。

还可以在活动图中出现其他图的图符，并可绘制混合图。

一个简单的活动图的样例如图 2.7 所示。

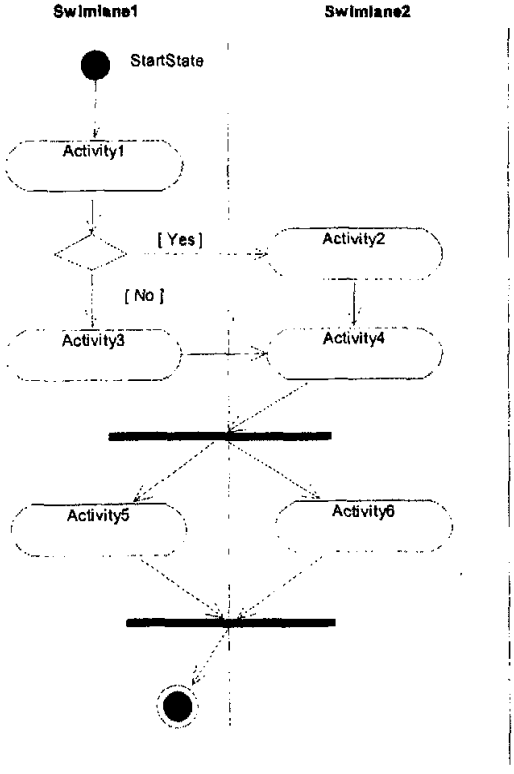

图 2.7 活动图样例

Fig.2.7 An example of activity diagram

2.3.4 交互图（Interactive Diagrams）

交互图描述了对象间的交互关系。交互图包括顺序图和协作图。

2.3.4.1 顺序图(Sequence Diagram)

顺序图描述了对象之间的动态合作关系，它强调对象之间消息发送的时间顺序，同时显示对象之间的交互。

(1) 顺序图的语义

在一个运行的系统中，对象之间要发生交互，并且这些交互要经历一定的时间。UML顺序图表达的正是这种基于时间的动态交互。

UML 顺序图在对象交互的表示中加入了时间维。

UML 顺序图不仅仅用于系统分析，还可用来说明一个组织之中的各种交互关系。

(2) 顺序图的表示法

顺序图有两个方向，就是我们所说的两维，垂直方向代表时间，水平方向代表参与交互的对象。在顺序图中，对象位于图的顶部，从上到下表示时间的流逝。每个对象都有一个垂直向下的对象生命线。对象生命线上的窄矩形条代表激活——该对象某个操作的执行。可以沿着对象的生命线表示出对象的状态。

消息(简单的、同步的或异步的)用连接对象生命线之间的带箭头连线代表。消息在垂直方向上的位置表示了该消息在交互序列中发生的时间。越靠近图顶部的消息发生的越早，越靠近底部的消息发生的越晚。

在一些系统中，一个对象的操作可以调用该对象自身的操作。这被称之为自身调用或递归。自身调用的表示是从一个激活框中引出消息线又重新回到这个激活框，并在该激活框中附加上一个小的矩形框。

在顺序图中，对象也会消亡。对象的消亡用该对象生命线底部的"X"来表示。

- 确定交互作用的上下文。上下文可以是系统、子系统、操作、类、用例或协作的一个脚本。
- 确定哪些对象参与了交互作用，将这些对象从左到右放在顺序图中，将重要的对象放在左边。
- 确定每个对象的生命线。对于那些在交互作用过程中创建和破坏的对象，要用合适的消息原型显式地标出对象的产生和破坏。
- 从发起交互作用的消息开始，将后来的消息从上到下地放在生命线之间。
- 如果需要规定时间或空间约束，可以为消息附加适当的时间或空间约束。如果想更正式地描述这个控制流，可以为每个消息添加前置条件和后置条件。

单个顺序图只能描述一个控制流，通常可以有多个交互作用图，一些交互作用图描述主要过程，其他的描述备选过程或例外过程。一个简单的顺序图的样例如图 2.8 所示。

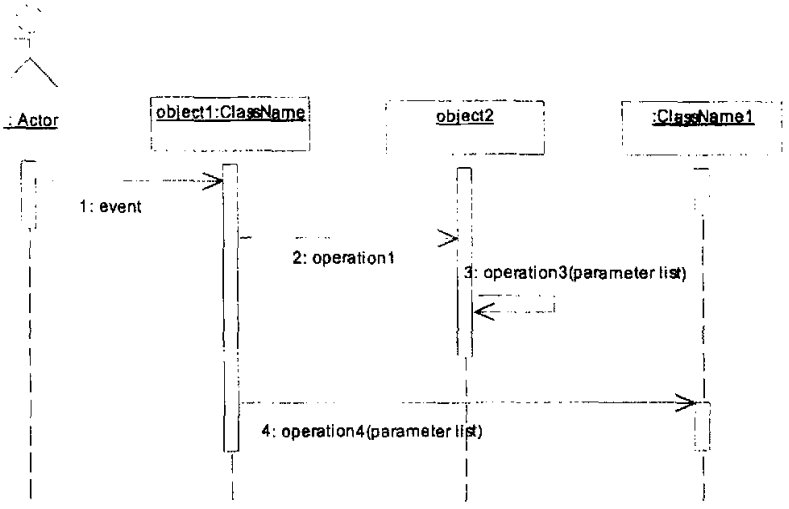

图 2.8 顺序图样例

Fig.2.8　An example of sequence diagram

2.3.4.2 协作图(Collaboration Diagram)

协作图描述了对象之间的协作关系，协作图与顺序图相似，描述了对象间的动态协作关系。除显示信息交换外，协作图还显示对象以及对象之间的关系。

(1) 协作图的语义

使对象和链共同工作以实现某种目的而进行的安排称为协作。在协作中实现的行为的消息序列称为交互。

协作由静态部分和动态部分组成。静态部分描述在协作实例中对象和链可能承担的角色；动态部分包含一个或多个动态交互，表示在执行计算过程中不同时间里协作中的消息流。在协作中流动的消息流可以选用状态机来进行描述，状态机中规定合法的行为顺序。而状态机中的事件代表协作中各角色间的消息交换。

协作由角色组成，同时角色也仅仅在协作中才有意义，在协作之外无意义。

在程序的实施过程中，我们经常考虑的问题就是对象的运行期(Runtime)问题，其始终就是对象在运行期的协作(这个协作我们称为运行期绑定)。

与顺序图一样，协作图也展示对象之间的交互关系。它绘制出对象和对象之间的消息连接。顺序图和协作图很相似。实际上两者是语义等价的。也就是说这两种图表达的是同一种信息。并且可以将顺序图转换为等价的协作图，反之亦然。

顺序图强调的是交互的时间顺序。协作图强调的是交互的语境和参与交互的对象的整体组织。还可以从另一个角度来看两种图的定义：顺序图按照时间顺序部图，而协作图按照空间组织部图。

(2) 协作图的表示法

协作图不仅展示了对象和对象之间的关联，还展示了对象之间的消息传递。通常在

协作图中省略掉关联的多重性，因为表示出多重性会使图变得混乱。

关联线旁的箭头表示对象之间传递的消息，箭头指向消息接收对象。消息名称和消息序号附在箭头线附近。消息序号代表消息发送的时间顺序，消息名和序号之间用冒号隔开。消息的一般含义是触发接收消息的对象执行它的一个操作。消息名后面一般有一个双括号，说明它代表的是操作。在双括号内可以指明操作需要的参数。

条件的表示与在顺序图中相同——将条件表达式用方括号括起来加在图中。

消息之间有从属关系。协作图中的消息序号命名方案与技术文章中的标题和子标题的命名类似——使用圆点来说明嵌套的层次。

一个简单的协作图的样例如图 2.9 所示。

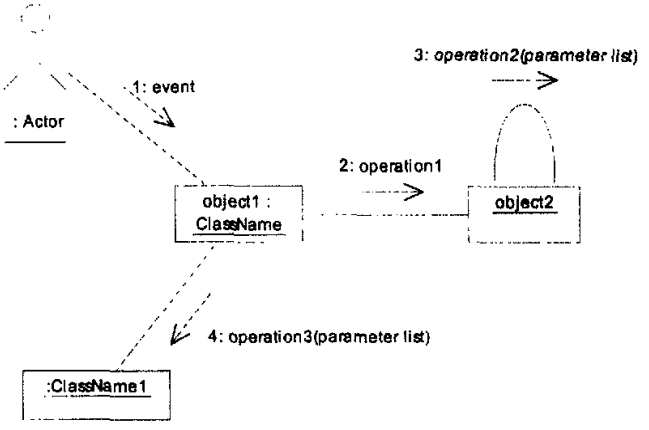

图 2.9 协作图样例

Fig.2.9 An example of collaboration diagram

2.3.5 实现图（Implementation Diagrams）

实现图包括组件图和部署图。

2.3.5.1 组件图(Component Diagrams)

组件图描述代码组件的物理结构及各组件之间的依赖关系。一个组件可能是源代码组件、二进制代码组件或可执行组件。组件图包含逻辑类或实现类的有关信息。有助于分析和理解组件之间的相互影响程度。

(1) 组件图的语义

软件组件是软件系统的一个物理单元。什么样的软件实体才能成为组件呢？数据表、数据文件、可执行文件、动态链接库和文档等均可。

类和组件之间的一个重要关系是：一个组件可以是多个类的实施。

组件的一个重要特征是它具有潜在的重用性。在当今高节奏的商业竞技场中，所建造的系统发挥功能越快，在竞争中获得的利益就越多。如果在开发一个系统中所构造的组件能够在开发另一个系统中被重用，那么就有利于获得这种竞争利益。在建立组件模型的工作上花费一些努力将有助于重用。

组件图表达的是现实世界中的实体——软件组件。它包括以下三种组件：

- 部署组件(deployment component)：它形成了可执行系统的基础。例如动态链接库(Dynamic Link Library, DLL)、二进制可执行体(executable)、Active X 控件以及 Java Beans 等。
- 工作产品组件(work product component)：它是部署组件的来源，例如数据文件和程序源代码。
- 可执行代码组件(execution component)：是可运行系统产生的运行结果。

软件模块可以用一个组件来表示。有些组件存在于编译时，有些存在于链接时，有些存在于执行时，有些在多种场合存在。一个编译时使用的组件只在编译时有意义。

组件图只有描述符形式，没有实例形式。要表示组件实例，应使用部署图。

(2) 组件图的表示法

组件图中包括组件、接口和关系。组件图的主图标是一个左侧附有两个小矩形的大矩形框，组件的名字位于组件图标的中央，组件名是一个文本串。

一个组件图的样例如图 2.10 所示。

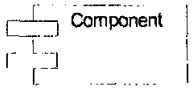

图 2.10　组件图样例

Fig.2.10 An example of component diagram

组件图表示了组件类元，以及其中定义的类(或其他的类元)与组件间的关系。组件类元还可以嵌套在其他组件类元之中，从而表示定义关系。

组件中定义的类在组件中表示(在组件内画出类图)。

可以用包含组件类元和节点类元的图来表示编译依赖关系。该关系用带箭头的虚线表示，箭头从用户组件指向它所依赖的服务组件。

若从一个组件指向另一个组件上的接口应采用虚线表示。

2.3.5.2 部署图(Deployment Diagrams)

部署图定义系统中软硬件的物理体系结构。部署图不但描述了实际的计算机和设备（用节点表示）以及它们之间的连接关系，还描述了连接的类型及组件之间的依赖性。在节点内部，可放置可执行组件和对象以显示节点跟可执行软件单元的对应关系。

(1) 部署图的语义

部署图是表示运行时过程节点、组件实例及其对象的配置的视图。运行时不存在的组件不出现在部署图中，而是在组件图中表示。

部署图含有用通信链相连的节点实例。节点实例包括运行时的实例，如组件实例和对象组件实例。该模型可以表示实例及其接口之间的依赖关系，还可以表示节点或者其他设备之间的实体移动。

(2) 部署图的表示法

部署图是节点符号与表示通讯关联的路径构成的网状图，节点符号可以包含组件实例，用于说明组件存在或运行于该节点上。组件符号可以包含对象，说明对象是组件的组成部分。组件之间用虚线箭头相连，说明一个组件使用了另一个组件的服务。必要时可以用构造型说明依赖关系。

部署图类似于对象图，通常用于表示系统中的各个节点的实例。

一个简单的部署图样例如图 2.11 所示。

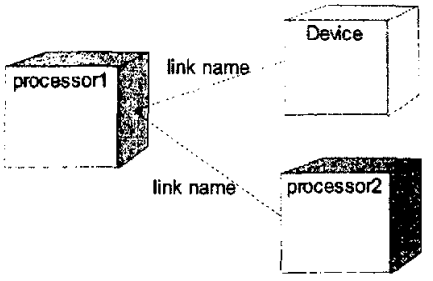

图 2.11 部署图样例

Fig.2.11 An example of deployment diagram

2.3.6 UML 模型图的作用

从应用的角度看，当采用面向对象技术设计系统时，

第一步是要描述需求；

第二步是根据需求建立系统的静态模型，以构造系统的结构；

第三步是描述系统的行为。

其中，在第一步和第二步中所建立的模型都是静态的，包括用例图、类图、对象图、组件图和部署图等五种模型，是统一建模语言 UML 的静态建模机制。而第三步中所建立的模型或者可以执行，或者表示执行时的顺序状态或交互关系。它包括状态图、顺序图、活动图和协作图四种模型图，是统一建模语言 UML 的动态建模机制。因此，统一建模语言 UML 的主要内容也可以归纳为静态建模机制和动态建模机制两大类。

正如前面所分析的，各种 UML 图可以让人们从多个视点考察一个系统，而所有这些框图可以从不同方面描绘一个系统。事实上，并非每个 UML 模型都必须包含所有的图，大多数 UML 模型只包含所有图的子集。

对系统建立模型为什么需要这么多种图？

首先，使用多重视图描述软件体系结构，可以使最终用户、开发人员、系统工程师、项目总监等独立地对待所关注的问题以及功能上和非功能上的需求。而使用单张视图来捕捉所有的系统体系结构要点，有一定的局限性，容易在单张视图中表达超过限度的内容。

其次，前面介绍的九种模型图，各自的侧重点不同，因此用于不同的目的。类图描述了类、接口、协作以及它们之间的关系，是最常用的图。对象图从实例角度描述对象及对象之间的关系。组件图说明代码本身的结构，部署图说明系统运行时刻的结构。用例图描述了用例、参与者以及它们之间的关系，用例图在系统需求分析阶段尤其重要。状态图描述跨越多个用例的单个对象的行为，不适合描述多个对象间的交互作用。因此，常将状态图与其他图示，如顺序图、协作图和活动图组合使用。顺序图和协作图适合描述单个用例中几个对象的行为，其中顺序图突出对象间交互作用的时间顺序，而协作图则能更清楚地表示出对象之间的连接关系。当行为较为简单时，顺序图和协作图是最好的选择。如果想显示跨越多用例或多线程的复杂行为，可考虑使用活动图。

最后，一个科学合理的系统设计需要考虑到诸如数据库设计、指令编写、界面操作、网络布局等等所有这些视点。每一种 UML 图都为我们提供了一种组成特殊视图的方式。采用多视点的目的是为了满足每一类与这个系统有利益关系的风险承担人的需要。

2.4 UML 与 Rational Rose 建模工具

2.4.1 Rational Rose 简介

Rational Rose 是分析和设计面向对象软件系统的强大的可视化工具，可以用来先建模系统再编写代码，从而一开始就保证系统结构合理。利用模型可以更方便地捕获设计缺陷，从而以较低的成本修正这些缺陷。

Rational Rose 支持业务模型，帮助了解系统的业务，有助于系统分析，可以先设计使用案例和 Use Case 框图，显示系统的功能。也可以用 Interaction 框图显示对象之间如何配合，提供所需功能。Class 框图可以显示系统中的对象及其相互关系。Component 框图可以演示类如何映射到实现组件。最后，Deployment 框图可以显示系统的网络设计。

Rose 模型是系统的图形，包括所有 UML 框图、参与者、用例、对象、类、组件和部署节点。它详细描述系统的内容和工作方法，开发人员可以用模型作为所建系统的蓝图。Rose 模型包含许多不同框图，使项目小组(客户、设计人员、项目经理、测试人员，等等)可以从不同角度看这个系统。同时，Rose 是整个项目组使用的工具，是每个小组成员分析设计信息的仓库。

除此之外，Rational Rose 还可以帮助开发人员产生框架代码，适用于市面上的多种语言，包括 C++、Ada、CORBA、Java、COM objects、Visual Basic 和 XML。Rational Rose 还可以逆向转出工程代码，根据现有系统产生模型。根据现有系统产生模型的好处很多。模型发生改变时，Rational Rose 可以修改代码，做出相应改变。代码发生改变时，Rational Rose 可以自动将这个改变加进模型中。这些特性保证模型与代码同步，避免遇到过时的模型。

利用 RoseScript 可以扩展 Rose，这是 Rose 附带的编程语言。利用 RoseScript 可以编写代码、自动改变模型、创建报表及完成 Rose 模型的其他任务。

目前 Rose 有三种版本：

- Rose Modeler，可以对系统创建模型，但不支持代码产生和逆向转出工程代码。
- Rose Professional，可以用一种语言生成代码。
- Rose Enterprise，可以用 C++、Ada、CORBA、Java、COM、Oracle8i、Visual Basic 和 XML 结构生成代码。模型的组件可以用不同语言生成。

该论文中涉及的 UML 图例，均由 Rational Rose Enterprise Edition2003 生成。

2.4.2 Rose 模型的四个视图

Rose 模型的四个视图是：Use Case 视图、Logical 视图、Component 视图和 Deployment 视图。每个视图针对不同对象，具有不同用途。Rose 模型的四个视图如图 2.12。

图 2.12 Rose 的四种视图

Fig.2.12 Four views of Rose

下面对这四种视图分别加以介绍：

(1) Use Case 视图

Use Case 视图包括系统中的所有角色、使用案例和 Use Case 框图，还可能包括一些顺序图或协作图。Use Case 视图是系统中与实现无关的视图，它关注系统功能的高层形状，而不关注系统的具体实现方法。

随着项目的进行，小组的所有成员可以通过 Use Case 视图了解正在建立的系统用例文档，并通过用例描述事件流程。利用这个信息，质量保证人员可以开始编写测试脚本，技术作者可以开始编写用户文档，分析人员和客户可以从中确认捕获了所有需求，开发人员可以看到系统创建哪些高级组件，系统逻辑如何。

一旦客户同意了角色和用例，就确定了系统范围。然后可以在 Logical 视图中继续开发，关注系统如何实施用例中提出的功能。

(2) Logical 视图

Logical 视图提供系统的详细图形，描述组件间如何关联。此外，Logical 视图还包括需要的特定类、类图和状态图。利用这些细节元素，开发人员可以构造系统的详细设计。

Logical 视图关注的焦点是系统的逻辑结构。在这个视图中，要标识系统组件，检查系统的信息和功能，检查组件之间的关系。重复使用是主要目的。通过认真指定类的信息和行为、组合类，以及检查类和包之间的关系，就可以确定重复使用的类和包。完成多个项目后，就可以将新类和包加进重复使用库中。今后的项目可以组装现有的类和包，而不必一切从头开始。

几乎小组中每个人都会用到 Logical 视图中的信息，但主要用户是开发人员和建筑

师。开发人员关心创建什么类，每个类包含的信息和功能以及类之间的关系。建筑师更关心系统的总体结构。建筑师要负责保证系统结构稳定，并考虑重复使用，系统能灵活地适应需求变化。分析人员利用类和类图信息确定代码会实现哪些业务需求。质量保证人员通过类、包和类图看看系统中的组块有哪些，那些需要测试。通过使用状态图显示特定类的功能。项目管理员通过类和框图确定系统结构是否合理，并估计系统的复杂程度。

一旦标识类并画出框图后，就可以转入 Component 视图，了解物理结构。

(3) Component 视图

Component 视图包括包含模型代码库、可执行文件、运行库和其他组件的信息。组件是代码的实际模块。在 Rose 中，组件和 Component 框图在 Component 视图中显示。

Component 视图的主要用户是负责控制代码和编译部署应用程序的人。有些组件是代码库，有些是运行组件，如可执行文件或动态链接库文件。开发人员也用 Component 视图显示已经生成的代码库和每个代码库中包含的类。

(4) Deployment 视图

Deployment 视图包含处理器、设备、进程和处理器与设备之间的连接。它关注系统的实际部署，可能与系统的逻辑结构有所不同。即系统可能用三层逻辑结构，但部署可能是两层的。界面放在一台机器上，而业务和数据库逻辑放在另一台机器上。Deployment 视图还要处理其他问题，如容错、网络带宽、故障恢复和响应时间。

第三章 合同管理信息系统的总体设计

3.1 合同管理信息系统的需求分析

系统分析与系统设计在开发一个应用系统中的作用是一个不容忽视的环节。一个应用系统的开发成功与否，系统分析与系统设计起着决定性的作用。开发一个应用系统，首先要进行系统分析。为了将一个人工管理系统转换成一个相应的计算机管理系统，系统的开发者应当详细调查将要为其开发的计算机管理系统的那个部门的人工管理系统是如何运行的。也就是要了解该部门的业务职能，每天所要处理的数据，文字和报表等，听取该部门的领导和业务人员对人工管理系统的描述和对要实现的计算机系统的想法与要求，明确要开发的应用系统应实现的目标和应具备的功能.再此基础上开始为实现每一个目标与功能作出总体设计和详细设计，包括所应使用的技术与方法。另一方面，所开发的应用系统都是人一机系统，即使用者通过计算机进行交互来获取他所需要的信息，这就需要有一个良好的人一机界面，使操作者能方便地使用，这些都是在系统设计时要仔细考虑的。系统的总体设计，详细设计及人-机交互界面的设计都是与系统程序设计息息相关的。如果在程序中还去不断地修改系统的目标和功能，或不断调整各种屏幕界面等,不仅会浪费大量的时间，还会影响所开发出来的系统的质量，因此，为开发出优秀的合同管理系统，必须对其现行的合同系统进行了详细的调查分析，才能作出系统分析与系统设计。

3.1.1 选题背景

在当今各领域的 MIS 开发中，对用户的需求进行分析已变得非常重要，这往往是 MIS 开发成功与否的关键。对用户需求的分析应该具有全面性、深入性和发展性。全面性是指考察由 MIS 管理的信息是否存在纰漏，必须保证各静态、动态信息的安全；深入性是指对信息的内容、结构、含义、变换、生存周期的分析和认识；发展性是指对信息未来发展变化的预测，因为信息在某个系统、机构内的变化往往存在着自身特有的发展规律，需求分析应该预见这种规律，否则就会缩短 MIS 的使用寿命。

为了保证需求分析的完备性，就必须保证需求分析的时间，还要有资深用户的参加，尽量不漏掉重要的细节。最有效的方式应该是先让用户讲述，问清其中的疑问，再与用户进行深入的讨论，同时还要写出需求分析说明书。只有经过这样反复的过程，才能最终获得合理的符合实际需要的系统需求。

在用 UML 进行用户需求分析阶段，可以使用用例来(Use Case)捕获用户需求。通过用例建模，描述对系统感兴趣的外部角色及其对系统(用例)的功能要求。分析阶段主要关心问题域中的主要概念(如抽象、类和对象等)和机制，需要识别这些类以及它们相互间的关系，并用 UML 类图来描述。为实现用例，类之间需要协作，这可以用 UML 动态

模型来描述。在分析阶段，只对问题域的对象(现实世界的概念)建模，而不考虑定义软件系统中技术细节的类(如处理用户接口、数据库、通讯和并行性等问题的类)。这些技术细节将在设计阶段引入。

3.1.2 合同管理信息系统的需求分析

本系统是针对首钢总公司内部工程合同管理开发的一个系统。首钢许多的工厂，比如：二炼钢、计算机公司等的合同数据要上报首钢总公司机动部进行审批，汇总。由于数据量大，工作繁杂，手工操作极易出错。况且首钢公司像二炼钢这样的厂有 50 多个厂归首钢机动部管理，工程合同数据都要上报机动部。机动部也要求利用该软件（此软件与下边各厂不尽相同），能高效地进行合同管理。同时，所用数据是首钢总公司要应用的 ERP 的基础数据，要求能与 ERP 做无缝连接（只需要接口模块），与 ERP 建设不会冲突，能为 ERP 提供基础数据。

调研阶段，就必须充分考虑该系统的通用性，和可移植性。以二炼钢需求作为平台进行分析，然后调研其他相关各厂。

合同管理信息系统是基于 Windows 平台开发的管理软件，该管理软件它可以帮企业进行工程项目合同管理，提高工作效率及管理水平。为领导决策提供依据。

开发合同管理信息系统软件，着重实现合同基本信息入机，根据合同信息进行随机查询、统计并打印等一系列工作，用计算机实现合同管理的需求，以减少人工重复、复杂、烦琐的工作，充分利用计算机高速信息传递和数据共享这一现代化的管理工具，提高管理的工作效率。为严格进行合同管理，打印出一套完整、标准、准确的数据表报，精确快速的数据统计分析，提供可靠的手段和保证。提高劳动效率及管理水平，使企业全方位受益。

本系统设计原则，是以结构简单，存取方便为目标。系统设计要求方便用户使用，可通过 “人 —— 机”对话的交互方式， 完成所想做的工作。

本合同管理信息系统由分厂合同管理和机动部合同管理组成。在对系统进行开发时，由于分厂和机动部实现的业务存在一定差异，因而分别对其需求进行分析。

(1) 机动部需求分析

- 工程合同信息由机动部统一管理，包括上报合同信息的入库，修改、检索、删除、浏览过程；
- 对总库中的数据进行备份保存，防止数据被破坏。如被破坏用备份数据进行恢复。
- 对总库中的数据进行分割保存，把要的数据从总库中分离出来，进行存档，数据库中还有这些数据。如把上一年的数据从总库中分离出来，进行存档。
- 对总库中分割出的数据与总库中数据进行合并。如把上一年的数据从总库中分离出来数据合并到库中，可对数据进行统计、分析，今年与上一年的数据进行对比等数据处理。
- 在总库中闭口合同资金修改时要进行提醒，以便做出相应处理；
- 分厂合同信息发生变更时，总库应该及时得到通知并进行更改调整；
- 机动部应具备首钢所有合同信息查询统计的功能，包括各分厂合同个数、分项的合同个数、及分项的合同个数占合同个数的百分比、合同分项金额、合同总金额、合

同分项金额占合同总金额的百分比等；

- 机动部应进行合同质量管理，同时将所用数据提供给首钢总公司 ERP 作为基础数据。

(2) 分厂需求分析

- 实现所有合同信息的入机管理，包括记录各种序号、合同编号、委托编号、施工单位、合同类型等；
- 分厂合同信息由管理系统实时记录，并可进行及时的查询和统计；
- 分厂系统应实现及时通过网络或其他通信方式向机动部上传合同信息等；
- 分厂的合同个数、分项的合同个数、及分项的合同个数占合同个数的百分比、合同分项金额、合同总金额、合同分项金额占合同总金额的百分比，能及时上传至机动部；
- 对分厂库中的数据能进行备份保存，防止数据被破坏。如被破坏用备份数据进行恢复。
- 对分厂库中的数据进行分割保存，把要的数据从总库中分离出来，进行存档，数据库中还有这些数据。如把上一年的数据从总库中分离出来，进行存档。
- 用户界面友好，操作灵活、简便，运行速度快；

合同管理信息系统应是基于 Windows 平台开发的管理软件，该管理软件它可以帮企业进行工程项目合同管理，提高工作效率及管理水平。为领导决策提供依据。

开发合同管理信息系统软件，着重实现合同基本信息入机，根据合同信息进行随机查询、统计并打印等一系列工作，用计算机实现合同管理的需求，以减少人工重复、复杂、烦琐的工作，充分利用计算机高速信息传递和数据共享这一现代化的管理工具，提高管理的工作效率。为严格进行合同管理，打印出一套完整、标准、准确的数据表报，精确快速的数据统计分析，提供可靠的手段和保证。提高劳动效率及管理水平，使企业全方位受益。

本系统设计原则，应是以结构简单，存取方便为目标。系统设计要求方便用户使用，可通过“人 —— 机”对话的交互方式， 完成所想做的工作。

3.2 系统体系结构的选择

3.2.1 几种主要客户机/服务器结构的实现方式和特点比较

客户机/服务器系统有三个主要部件：数据库服务器、客户应用程序和网络。服务器负责有效地管理系统的资源，其任务集中于：数据库安全性要求；数据库访问并发性控制；数据库前端的客户应用程序的全局数据完整性规则；数据库的备份与恢复。客户端应用程序的主要任务是：提供用户与数据库交互的界面；向数据库服务器提交用户请求并接收来自数据库服务器的信息；利用客户应用程序对存在于客户端的数据执行应用要求。网络的主要作用是完成数据库服务器和客户应用程序之间的数据传输与数据交换。

(1) 两层 Client/Server 体系结构

目前大多数 MIS 系统都是使用两层体系结构来实现的。这种体系结构将应用程序分

成两部分：客户端应用程序和数据库服务器。在这种模式中，客户机上要安装专门的应用程序来操作后台数据库服务器中的数据，显示和交互、计算和接收处理数据的工作由客户端应用程序完成；数据的处理和维护工作由数据库服务器完成；而业务工作由客户端应用程序和数据库服务器共同承担。

相对于主机带终端的集中式结构，两层 Client/Server 体系结构中作为客户机的 PC 机一般性能比终端高，容易实现较丰富的图形显示和交互。同时，相对于文件服务器的 PC 局域网结构，两层 Client/Server 结构由于在网络上传送的是数据操作的请求和数据操作的结果，网络传输量小，并且由于是数据库服务器负责数据操作，数据的安全性、完整性维护和开放性都较好。相对于其它 Client/Server 结构，两层结构使用时间较长，技术成熟，开发人员经验丰富，同时可供利用的开发工具和资源也较丰富。

两层 Client/Server 结构的主要缺点是没有将业务处理单独独立出来，而是分布在各客户端和数据库服务器上，这就给各客户端上的软件的升级和维护带来不便。

(2) 三层 Client/Server 结构

三层 Client/Server 结构是在两层结构基础上的扩展，它将业务处理的工作从数据库服务器和客户端独立出来，由新增加的应用服务器来完成，客户端只完成显示和交互的工作，数据库服务器只完成数据的处理和维护工作。

与两层 Client/Server 结构相比，由于业务处理集中在应用服务器上，大大地减轻了维护升级工作的复杂性，同时也简化了客户端的工作，解决了"胖客户机"的问题。但是，由于三层 C/S 结构在开发工具和资源方面，不如两层结构丰富。所以在选择体系结构时，应根据系统的开发周期、规模和开发人员等实际情况，对具体问题进行具体分析。

(3) Browser/Server 体系结构

随着 Internet 的广泛应用，又出现了一种新的体系结构，即 Browser/Serve 结构．该结构本质上也是客户机/服务器，又称为"瘦客户机"模式，是三层 Client/Server 结构在 Web 上应用的特例。

Browser/Server 体系结构下的客户机只需要安装浏览器软件，如 Windows98 内嵌的 IE 即可，无需开发前端应用程序，它负责实现显示和交互。中间层的 Web 应用服务器，如 Microsoft 公司的 IIS 等是连接前端客户机和后台数据库服务器的桥梁，它的任务是接受用户的请求，执行相应的扩展应用程序与数据库进行连接，通过 SQL 等方式向数据库服务器提出数据处理申请，而后将数据库服务器的数据处理结果提交给 Web 服务器，再由 Web 服务器传送回客户端。因此对中间层数据库服务器的要求较高。后台数据库服务器负责接受 Web 服务器对数据库操纵的请求，实现对数据库查询、修改、更新等功能。

该结构安全性较差，数据传输量一般较大。

3.2.2 选择两层 C/S 体系结构构建合同管理信息系统

通过对上述几种体系结构的比较和分析，结合合同管理信息系统的实际，开发该合同管理信息系统时，对机动部合同管理信息系统和分厂管理信息系统均采用两层 C/S 体系结构。

采用两层 C/S 体系结构，有以下突出特点：

(1) 由于该体系结构使用时间较长，技术较为成熟，因而不仅可以缩短开发周期，还可以保证系统的安全性，见效快、性价比和稳定性相对较高。

(2) C/S 体系结构的显著优点是速度较快，功能完备，非常适合于子系统规模不是很大的系统采用。

(3) 机动部与分厂之间通过网络及时传输合同信息，数据量一般不大。

(4) 该体系结构的可维护性好，在经营方式和管理方式发生变化后，可以在保护原有资源的条件下，便于系统功能的重组与扩充，使开发出的系统可以灵活地适应需求的变化。

(5) 可将该系统配置成分布式数据库管理系统，实现信息处理的分布性。

3.2.3 系统的设计原则

管理信息系统的建立是一项复杂的系统工程，每个系统都有它自身的复杂性和特殊性。该系统的设计原则是：

(1) 实用性和可靠性原则

如何使信息系统中各种类型的信息更好、更方便地被用户利用和操作，是开发该信息系统的首要原则。本系统在开发过程中通过选用成熟的产品和进行定制开发不仅提高了系统的可靠性，同时也使客户端软件操作简便，查询统计快速、准确、具有可靠性和简明易用性。

可靠性包括硬件可靠性和软件可靠性。硬件可靠性主要体现在选用设备性能方面的稳定；软件可靠性则指应用软件系统平台稳定，应用软件功能可靠、无故障及具有可用性等。

(2) 可扩充性和易维护性原则

由于该系统选用先进的、性能稳定的后台数据库和功能完善的前端开发工具进行软件的开发，增加了系统的可扩充性及其复用能力，也减小了维护的技术难度，同时利用网络传输信息，计算机审批，提高管理的信息化程度。

(3) 先进性和安全性原则

本系统在开发时选用 Microsoft 和 Sun 公司的产品，其主要原因在于这些公司具有良好的发展前景，其产品较为成熟，并有着最强的对国际标准和国际流行标准的控制能力，这就保证了系统具有一定的先进性和安全性。

系统的软件安全性还表现在对各级用户分配不同的权限，以身份认证来辨别用户；系统的硬件安全性包括采用备份服务器和硬盘镜像技术等。

(4) 易管理和复用性原则

该系统采用面向对象的方法和模块化的思想，将一个系统分解为更小、更容易管理的独立的小模块，这就使得系统易于管理、易于修改以及其模块可重复使用等。

(5) 兼容性原则

该系统中所用合同相关数据兼容首钢总公司 ERP 的基础数据，能与 ERP 做无缝连接（只需要接口模块），与 ERP 建设不会冲突，能为 ERP 提供基础数据。

3.3 数据库及开发工具的选择

本系统采用 Microsoft SQL Server 2000 数据库作为后台数据库，选择 JAVA 作为开发工具。

3.3.1 数据库系统平台

早期面向单用户桌面的数据库（如 Microsoft Access 与 Microsoft FoxPro）属于文件/服务器数据库，数据存放在文件中，数据的各个用户直接从文件中存取所需数据，意外改变或崩溃破坏关键数据的可能性较大；现在面向部门、大环境的数据库（如 Oracle, Sybase 与 Microsoft SQL Server）为客户机/服务器数据库，在客户机/服务器数据库中，用户应用程序（客户机）存取数据通过一个主程序（服务器）控制，一方面减少意外改变或崩溃破坏关键数据的可能性，另外客户机/服务器数据库还能提供保护数据的特性。

Microsoft SQL Server 2000 为组织提供了企业级数据管理平台，具有以下特点：

(1) 与 Windows NT 系统有机集成，多线程体系结构设计，提高了系统对并发用户的响应速度。

(2) 丰富的编程接口工具：SQL Server 提供了 Transact-SQL、DB-Library for C 和 DB-Library for Visual Basic、嵌入式 SQL 等开发工具，此外还支持 ODBC 和 OLE DB 规范，可以使用 ODBC、OLE DB 接口函数访问 SQL Server 数据库。

(3) 隐含的并发控制能力：MS SQL Server 2000 利用动态锁定功能使得并发用户可安全而高效地访问数据。

(4) 数据可靠性，支持数据复制：提供方便而灵活的备份和恢复方法以及复制功能。数据库的两个或两个以上的副本可以保持同步，改变其中一份时，其他的副本也随之改变。数据复制模型如图 3.1 所表示。

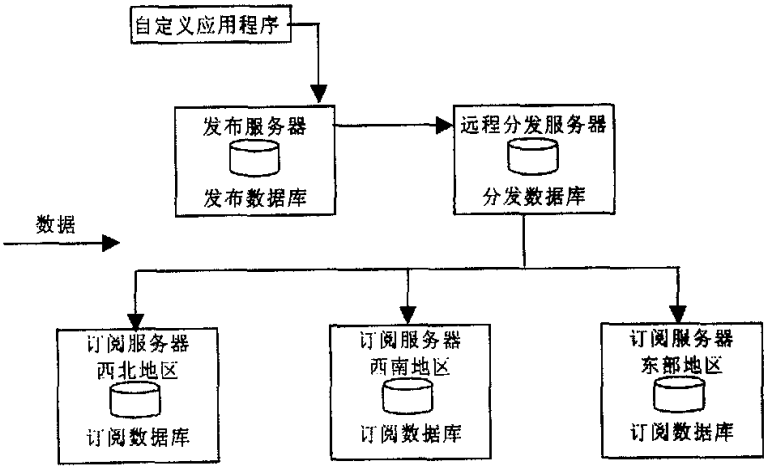

图 3.1 数据复制模型

Fig.3.1 Model of data copy

(5) 易于安装、部署和使用，管理方便：MS SQL Server 2000 易用性强，操作界面简单，利用微软提供的管理工具（SQL Enterprise Manager），可以方便管理多个服务器。

(6) 数据仓库：SQL Server 为建立数据仓库提供了几种工具。使用"DTS 设计器"，可以定义从各种数据源建立数据仓库的步骤、工作流和数据转换。

根据合同管理信息系统的建设及服务特点，从业务运行角度出发，为了方便各子系

统的数据调用和处理，在首钢机动部建立数据的集中存储和管理；从服务角度出发，为了方便各用户单位的数据（合同）的查询和调用，需提供客户端服务。为此，采用小型的分布式数据库系统、建立客户机／服务器（C／S）数据库。我们选用了 Microsoft 公司的 MS SQL Server 2000 作为数据库系统平台。

3.3.2 Java 的特性及与 UML 的映射关系

Java 是一种跨平台，适合于分布式计算环境的面向对象编程语言。具体来说，它具有如下特性：简单性、面向对象、分布式、解释型、可靠、安全、平台无关、可移植、高性能、多线程、动态性等。下面我们将重点介绍 Java 语言的面向对象、平台无关、分布式、多线程、可靠和安全等特性。

(1) 面向对象

面向对象其实是现实世界模型的自然延伸。现实世界中任何实体都可以看作是对象。所有面向对象编程语言都支持三个概念：封装、多态性和继承，Java 也不例外。Java 语言的封装性较强，因为 Java 无全程变量，无主函数，在 Java 中绝大部分成员是对象，只有简单的数字类型、字符类型和布尔类型除外。而对于这些类型，Java 也提供了相应的对象类型以便与其他对象交互操作。

(2) 平台无关性

Java 是平台无关的语言是指用 Java 写的应用程序不用修改就可在不同的软硬件平台上运行。Java 主要靠 Java 虚拟机（JVM）在目标码级实现平台无关性。另外，Java 采用的是基于 IEEE 标准的数据类型。通过 JVM 保证数据类型的一致性，也确保了 Java 的平台无关性。

Java 的平台无关性具有深远意义。首先，它使得编程人员所梦寐以求的事情（开发一次软件在任意平台上运行）变成事实，这将大大加快和促进软件产品的开发。

(3) 分布式

分布式包括数据分布和操作分布。数据分布是指数据可以分散在网络的不同主机上，操作分布是指把一个计算分散在不同主机上处理。

Java 支持 WWW 客户机/服务器计算模式，因此，它支持这两种分布性。Java 提供了一整套网络类库，开发人员可以利用类库进行网络程序设计，方便得实现 Java 的分布式特性。

(4) 可靠性和安全性

Java 最初设计目的是应用于电子类消费产品，因此要求较高的可靠性。Java 虽然源于 C++，但它消除了许多 C++不可靠因素，可以防止许多编程错误。首先，Java 是强类型的语言，要求显式的方法声明，这保证了编译器可以发现方法调用错误，保证程序更加可靠；其次，Java 不支持指针，这杜绝了内存的非法访问；第三，Java 的自动单元收集防止了内存丢失等动态内存分配导致的问题；第四，Java 解释器运行时实施检查，可以发现数组和字符串访问的越界，最后，Java 提供了异常处理机制，程序员可以把一组错误代码放在一个地方，这样可以简化错误处理任务便于恢复。

由于 Java 主要用于网络应用程序开发，因此对安全性有较高的要求。Java 通过自己的安全机制防止了病毒程序的产生和下载程序对本地系统的威胁破坏。

(5) 多线程

线程是操作系统的一种新概念，它又被称作轻量进程，是比传统进程更小的可并发执行的单位。C 和 C++采用单线程体系结构，而 Java 却提供了多线程支持,提高程序执行效率。

Java 在两方面支持多线程。一方面，Java 环境本身就是多线程的。若干个系统线程运行负责必要的无用单元回收，系统维护等系统级操作；另一方面，Java 语言内置多线程控制，可以大大简化多线程应用程序开发。提供类 Thread，由它负责启动运行，终止线程，并可检查线程状态。必须注意的是，Java 的多线程支持在一定程度上受运行时支持平台的限制。例如，如果操作系统本身不支持多线程，Java 的多线程特性可能就表现不出来。

在为元素建模时，所有的 UML 图表都是为了创建更好的 Java 应用程序。但是，某些 UML 图表会与实际产生的 Java 代码有更密切的对应关系。表 3.1 提供了 UML 图到 Java 的映射。

表 3.1 将 UML 图映射到 Java

Table 3.1 Mapping from UML to Java

UML 图	特定的元素	Java 中的对应物
包图	实例	Java 包
用例图	实例	以路径形式提供的用户界面元素，最终会变成序列图
类图	操作	操作/方法
	属性	成员变量和相关的存取器操作
	关联	成员变量和相关的存取器操作
顺序图	实例	控制器类中用来协调流程的操作
	消息目标	目标类中的操作
协作图	实例	控制器类中用来协调流程的操作
	消息目标	目标类中的操作
状态图	动作/活动	处于生命周期中的类的操作
	事件	处于生命周期中的类或另一个协作类的操作
	状态变量	处于生命周期中的类的属性
活动图	动作状态	方法代码，实现一个复杂操作或对向一个用例路径发送的消息进行协调
组件图	组件	通常是一个.java 和/或一个.class 文件
部署图	节点	物理的，可部署的安装程序集，可部署到客户端和/或服务器端

对于核心业务和商业应用，有三种 UML 图对所交付的 Java 程序影响最大：用例图、类图和顺序图（或协作图）。其他的图根据项目的特征和需求而定。UML 和 Java 配合很好。通过利用可视化建模工具来支持正向和逆向（从模型创建代码和从代码创建模型）。

3.4 系统实现的功能简介

模块化，简单地说就是把系统划分为若干个模块，每个模块完成一个特定的功能，然后将这些模块汇集起来组成一个整体(即系统)，用以完成指定功能的一种方法。

采用模块化设计原理可以使整个系统设计简易，结构清晰，可读性、可维护性增强，提高系统的可行性，同时也有助于信息系统开发的组织和管理。

3.4.1 课题的主要研究内容

根据工程合同的特点和内容，主要了分成以下几个模块：字典管理模块、基本数据管理模块、统计查询模块、打印表报模块、数据维护管理模块。系统的设计既要考虑系统的独立性，又要考虑系统的可扩充性、可移植性和今后开发的兼容性要求。系统的运行环境是 Win NT 或 Novell 网络环境和 Win95、Win98 中文版环境。

该系统特点如下：

◆先进的管理理论和企业管理实践的高度融合

该软件是根据首钢内部工程合同的基本特点，工作流控制等先进模型，加入大量统计分析方法，贴近于用户的实际需求，为各个厂提供解决方案。根据的数据调研和大量的用户反馈，我们在软件的设计中既充分考虑了合同的规范性标准，又为企业的实施过程，例如合同的综合查询和合同的综合统计的编写，提供了相当的灵活性，能充分适应各种复杂多变的状况。

◆严格的合同控制管理

该软件允许用户系统管理员自行定义每个使用者对每个工作过程，每个记录文件乃至每个数据的操作权限，操作员在每一个使用过程中系统都会自动检验操作权限，实现多级安全控制，极大的提高了用户系统和操作安全和数据安全性，防止未经许可的操作和应用户使用不当而造成的损失。

◆简便,实用性强的操作界面

该软件具有详细的在线帮助信息，用户可以随时得到程序使用的指导，用户界面，数据输入/输出界面和功能键风格统一，使用方便灵活，用户可迅速掌握使用和操作。

◆灵活的查询和报表功能

该软件提供了灵活的查询和报表功能，用户可自行设计，定义表报格式，可自由设计工作记录的内容，修改权限，数据来源等等，并可自由运用从单项条件查询到多条件组合模糊查询等多种方式，从多角度方便的查询所需了解的信息，为日常工作和决策辅助提供了极大的便利。

◆强大的图形分析功能

该软件提供了合同管理常用的多种统计图形，可直接运用于各类数据统计分析，更增设了数据的分析预警功能，可直观的向管理者提供合同进行情况的图示资料，并及时就不正常情况向管理者发出警报，使合同的进行状况始终处于受控状态。

◆先进的客户机/服务器结构

该软件采用先进的客户机/服务器(Client/Server)结构,能充分利用客户的现有微机资源，最大限度的减少投资，提高运行效率。

◆充分考虑企业发展

该软件考虑到企业现在或将来可能使用的其他管理软件，如 ERP 或财务系统等，为系统预留了与外部各类数据库的接口，可方便的向外部数据库中输入或提取数据，从而保护了客户在现有系统上的投资，该软件也可方便的扩充为企业的全面管理系统软件。

3.4.2 机动部合同管理实现的功能

首钢机动部合同管理信息系统实现的职能可划分为六大功能模块，即字典管理模块、基本数据管理模块、合同信息统计、基本信息查询模块、打印表报模块、数据维护

管理模块。机动部合同管理信息系统实现的功能如图 3.2 所示。

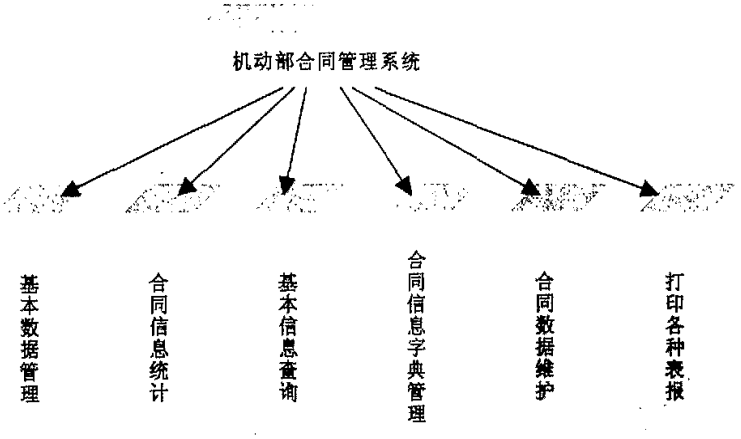

图 3.2 合同管理信息系统实现的功能

Fig.3.2 Functions of contract management information system

(1) 基本数据管理模块

本模块实现的功能如图 3.3 所表示。

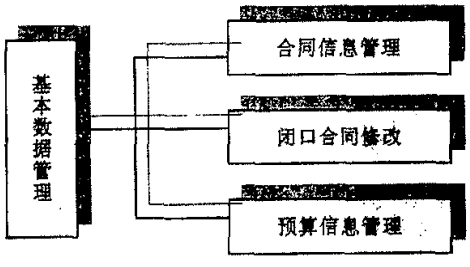

图 3.3 基本数据管理模块的功能

Fig.3.3 Functions of basic data management module

① 合同信息管理：对合同基本信息进行管理。要求合同员输入的表有施工单位表、资金来源表、修理类别表及合同类型表；要求输出的表有合同数据表。

- 合同编号。确定合同年号、序号，其中根据系统日期自动生成年号，用户也可修改，根据年系统自动生成序号，序号从 1 开始，自动加 1，年发生变化时序号从 1 开始；并进行委托编号、合同编号，都不允许重复；
- 入库记录。各相关人员做详细的入库记录。包括：合同内容、施工单位等；施工单位内容应是字典内容。
- 合同的资金管理。合同管理员根据入库记录中的若干项进行填写。核实合同金额、实际金额、资金来源。其中，资金来源有：修理费、生产费、其他费用。实际金

额：程序用合同金额替代实际金额，即实际金额=合同金额。

- 合同的类型管理。确定合同的修理类别、开工时间、竣工时间、合同类型；其中修理类型包括：中修、小修、炉化、强化、炉役、抢修等。合同类型包括：事先、事后、开口、附条件等。
- 合同维护管理。明确分厂各合同送机动部时间、自机动部取回时间、合同终审是否合格。
- 权限分配。不同工作人员如核检员、验收员和保管员具有不同的操作权限，各自以自己独有的密码登录系统，出现问题时，各负其责。

② 闭口合同情况管理：修改合同闭口。

- 浏览查看合同：对输入的合同数据进行查看，发现错误及时修改，也可以按照合同序号定位合同，找到某合同数据，并显示到屏幕上。
- 修改功能：对合同类型为事先和开口的合同进行修改，只能修改合同金额，输入是否闭口，闭口金额、闭口时间并进行存盘。其中，闭口类型：闭口或成就。闭口时间的输入格式为：年月日。闭口金额：当输入闭口金额时，要求同时把表中的实际金额修改，用闭口金额替换实际金额。
- 还原功能。要求能恢复对当前的操作的前一个操作。
- 打印功能。对闭口合同按照合同年号的范围进行预览、打印报表。并要求将打印的数据转到 Microsoft Excel，可以对数据进行编辑和修改及打印。
- 简单查找合同：按照合同序号或委托编号其中之一进行查找，找到所要的合同，可对其进行修改、删除等功能的操作。

③ 预算管理：完成预算基本信息的输入、修改、删除、打印、浏览、查找等功能。

- 预算信息编号。确定委托编号、预算编号，其中，预算编号要求自行输入，不允许重复。委托编号应该与合同的委托编号相同。查找合同数据，是否有委托编号的合同数据，如没有，系统发出报警和提示。
- 预算内容入库。各相关人员做详细的入库记录。包括：预算内容、施工单位等；施工单位内容应是数据字典内容。
- 预算资金管理。合同管理员根据入库记录中的若干项进行填写。核实施工单位申报预算金额、二炼钢审定预算金额、预算初审人、终审预算金额等。其中，二炼钢审定预算金额中可用户输入数字或汉字或选择，选择仅包括两项：执行中标价格、计划向量。
- 核减预算金额管理。确保预算金额的准确性。核减信息包括：二炼钢核减预算金额、计财部核减预算金额等。二炼钢核减预算金额：如果二炼钢审定预算金额为执行中标价格、计划向量时，二炼钢核减预算金额应该等于二炼钢审定预算金额；否则，二炼钢核减预算金额为施工单位申报预算金额减掉二炼钢审定预算金额。计财部核减预算金额计算方法：计财部核减预算金额=施工单位申报预算金额—终审预算金额。
- 预算时间管理。对合理预算信息，信息齐全的标定时间信息，包括：送计财部时间、自计财部取回时间。

⑵ 基本信息查询

① 机动部合同查询

基本功能：根据送机动部时间，查找出所要的合同数据，并行打印，也可以把查找出所要的合同数据转到 Excel 中，对数据进行排版、编辑、打印等操作。

② 合同信息综合查询

基本功能：

根据数据库的字段名及任意条件进行查询，按用户的需要任意组合排序并显示、打印、任意组合字段的查询结果，也可以把查询的数据转到 Excel 中。可把经常用查询的条件保存起来，为以后查询用（可以调出）。

要求能实现对年号、序号、合同编号、施工单位、合同金额、资金来源、修理类别、开工时间、竣工时间、合同类型、闭口类型、闭口时间、闭口金额、送机动部时间、自机动部取回时间、终审是否合格等各个字段的单一或多重操作。

要求实现：

- 具有排序功能，数据表可按选项值进行排序（升序），被排序项可以任意组合，即可以组合排序，查询的结果将按选择的排序字段排序。
- 实现通用查询功能，用户可以随意选择要查询的项目，“操作符”的内容是固定的，包括“=”、“<”、“>”、“<=”、“>=”。“操作值”选项的内容是可变的它与所选择查询的字段有关，它把被选择的字段的数据库中所有的值（去掉重复的）装入数组以方便供选择；同时允许用户输入数据。
- 保存查询条件，用户一次输入的查询条件，希望能保存下来，供以后重复使用。此查询条件将被显示在“查询条件”中，此时，可直接按此条件查询，也可利用“（”、“）”、“与”、“或”功能键与其他查询条件组合成为复合的条件，即拼写成表达式。如果表达式出错，系统会提示出错功能，要求用户重新拼写。选择好查寻条件后，可以显示符合条件的记录，也可按打印符合条件的记录，同时可以把查询的数据转到 Excel 中。

⑶ 数据统计及输出

① 合同信息综合统计

功能描述：

根据数据库的主要字段名及任意条件对数字型的字段进行统计，并显示或打印统计结果。根据统计条件计算出的合同个数、分项的合同个数、及分项的合同个数占合同个数的百分比、合同分项金额、合同总金额、合同分项金额占合同总金额的百分比，并显示或打印统计结果。

② 报计财部预算查询

基本功能：根据送计财部时间，查找出所要的预算数据，并行打印，也可以把查找出所要的预算数据转到 Excel 中，对数据进行排版、编辑、打印等操作。

③ 预算信息综合查询

基本功能：

根据数据库的字段名及任意条件进行查询，按用户的需要任意组合排序并显示、打印、任意组合字段的查询结果，也可以把查询的数据转到 Excel 中。可把经常用查询的条件保存起来，为以后查询用（可以调出）。（功能与合同综合查询基本相同。）

④ 预算信息综合统计

功能描述:

根据数据库的主要字段名及任意条件对数字型的字段进行统计,计算出根据条件的预算个数、分项的预算个数、及分项的预算个数占预算个数的百分比.,并显示或打印统计结果。

⑤ 首钢检修预算收文台帐

基本功能：根据送审时间，输入送审单位、送审人，查找出所要的预算数据，并进行打印，也可以把查找出所要的预算数据转到 Excel 中，对数据进行排版、编辑、打印等操作。

(1) 数据字典管理

本模块实现的功能如图 3.4 所表示。

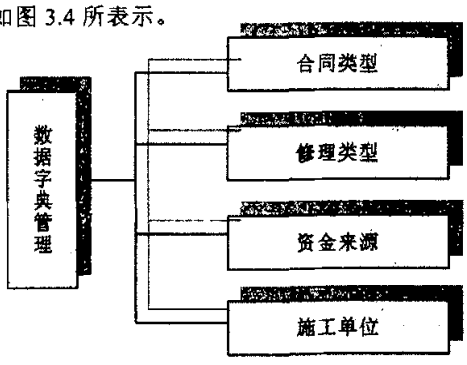

图 3.4 数据字典管理模块的功能

Fig.3.4 Functions of data dictionary management module

本模块对施工单位进行编码，对编码、单位名称进行输入，为合同、预算输入程序中使用。这样在合同、预算输入程序中可以直接使用的下拉框内容进行输入，为了以后用户改变下拉框值时，只改变字典数据，不用改变程序，规范软件开发的过程，缩减软件开发的周期。

① 合同类型:

合同类型的设定。预先设置类型为：事先、事后、开口、附条件；合同管理员可根据需要插入、增加、删除某种类型。

② 修理类型:

管理员可根据修理类合同的具体类型进行设定。预先设定类型为：中修、小修、例修、强化、炉役、炉化、抢修、月修。合同管理员可根据需要插入、增加、删除某种类型。

③ 资金来源:

由于首钢内部资金管理的特点，确定合同资金的具体来源。预先设定来源为：修理费、生产费、其他费用。合同管理员可根据实际资金情况插入、增加、删除某种类型。

④ 施工单位管理：

由于首钢内部单位众多，并且一些单位下面还有子单位，要求能合理高效地管理这些施工单位，以便查询、统计、修改、变更、分析等：

施工单位编码：编码原则，编码第 1-2 位，为大单位编码，编码第 3-4 位，为此大单位下的小单位编码。例如：

电子公司大编码：01

电子公司编码：　0101

电子公司电讯：　0102　　01：表示电子公司；02：表示电讯子公司

电子公司计算机：0103　　01：表示电子公司；03：表示计算机公司

在本模块中要求实现功能：

- 管理员能查找某单位编码，并显示到屏幕上。可根据需要再进行其它的操作，如修改、删除等操作。
- 管理员对对单位编码、单位名称数据进行录入并进行存盘。
- 修改数据。对输错的单位编码、单位名称数据进行修改并进行存盘。
- 还原功能。能自动恢复对当前的操作的前一个操作。
- 浏览功能。对输入的单位编码、单位名称数据进行查看，发现错误及时修改。
- 打印功能。对输入的单位编码、单位名称按照单位编码的顺序进行预览、打印。

⑸ 合同数据维护模块

对合同数据、预算数据进行备份保存，防止数据被破坏。如被破坏用备份数据进行恢复。

- 对库中的数据进行备份保存，防止数据被破坏。如被破坏用备份数据进行恢复。
- 对库中的数据进行分割保存，把要的数据从总库中分离出来，进行存档，数据库中还有这些数据。如把上一年的数据从总库中分离出来，进行存档。
- 对总库中分割出的数据与总库中数据进行合并。如把上一年的数据从总库中分离出来数据合并到库中，可对数据进行统计、分析，今年与上一年的数据进行对比等数据处理。
- 对库中的数据进行删除。为了提高处理速度，对一些前几年的不用的数据分割保存后，从库中删除。如把上一年的数据从总库中分割保存后进行删除。

⑹ 打印表报模块

包括上报机动部合同查询打印、上报计财部预算查询打印、首钢检修预算收文台帐打印等功能：

① 合同信息打印

功能描述：

根据数据库的主要字段名及任意条件对数字型的字段进行统计，并显示或打印统计结果。根据统计条件计算出的合同个数、分项的合同个数、及分项的合同个数占合同个数的百分比、合同分项金额、合同总金额、合同分项金额占合同总金额的百分比，并显示或打印统计结果。

② 报计财部预算打印

基本功能：根据送计财部时间，查找出所要的预算数据，并行打印，也可以把查找出所要的预算数据转到 Excel 中，对数据进行排版、编辑、打印等操作。还可根据数据库

的主要字段名及任意条件对数字型的字段进行统计，计算出根据条件的预算个数、分项的预算个数、及分项的预算个数占预算个数的百分比、，并显示或打印统计结果。
以后查询用（可以调出）。（功能与合同综合查询基本相同。）

③ 首钢检修预算收文台帐

基本功能：根据送审时间，输入送审单位、送审人，查找出所要的预算数据，并进行打印，也可以把查找出所要的预算数据转到 Excel 中，对数据进行排版、编辑、打印等操作。

3.4.3 分厂实现的功能

根据首钢检修模式开发的合同管理软件，紧跟首钢检修管理与首钢管理结合非常紧密，要求能应用于二炼钢厂，机动部，有很高的通用性，基本上可移植到各厂，同时也可增加个性化需求。

根据首钢合同管理分厂的实际业务情况，其管理信息系统应该与机动部合同管理系统相似。主要包括 5 个模块：系统管理模块、基本信息管理模块、字典管理模块、查询统计模块、数据维护模块、打印表报模块。

★系统管理模块

进行工作人员的角色设定与权限分配，工作人员的添加与删除，工作人员登录密码的修改等。

★基本信息管理模块

本模块是其他模块的基础。对合同数据进行输入，修改、查找、删除、浏览等。对委托编号进行唯一性检查，减少了人工可能重复登记的错误。可根据序号、合同编号、委托编号中任何之一进行查找某一条合同信息，提高了查找的速度及准确性（与人工查找帐本相比），可以打印所需的合同数据，可把合同数据转入到 EXCEL 中。

★字典管理模块

本模块对施工单位、合同类型、修理类别等进行输入，在合同数据输入程序中使用，提高了输入的准确性和一致性。

★查询统计模块

本模块对合同信息进行综合查询、综合统计打印。

根据合同信息中的任意一个名称及任意条件组合进行查询，按用户的需要任意组合排序并显示、打印、任意组合字段的查询结果，也可以把查询的数据转到 Excel 中。可把经常要用查询的条件保存起来，为以后查询用。

根据合同信息的主要名称及任意条件对数字型的字段进统计，并显示或打印统计结果。根据统计条件计算出的合同个数、分项的合同个数、及分项的合同个数占合同个数的百分比、合同分项金额、合同总金额、合同分项金额占合同总金额的百分比，并显示或打印统计结果。

★数据维护模块

. 对库中的数据进行备份保存，防止数据被破坏。如被破坏用备份数据进行恢复。

. 对库中的数据进行分割保存，把要的数据从总库中分离出来，进行存档，数据库中还有这些数据。如把上一年的数据从总库中分离出来，进行存档。

. 对总库中分割出的数据与总库中数据进行合并。如把上一年的数据从总库中分离

出来数据合并到库中，可对数据进行统计、分析，今年与上一年的数据进行对比等数据处理。

. 对库中的数据进行删除。为了提高处理速度，对一些前几年的不用的数据分割保存后，从库中删除。如把上一年的数据从总库中分割保存后进行删除。

★打印表报模块

. 根据用户的要求打印固定的一些表格

. 用户输入一些数据如时间等，程序对数据进行处理并打印表报，也可以把表报数据转到 Excel 中，用户根据自己的需要对数据进行排版、编辑、打印等操作。

. 根据数据库的字段名及任意条件对数字型的字段进统计，并显示或打印统计结果。并显示或打印统计结果。也可以把统计出结果数据转到 Excel 中，用户根据自己的需要对数据进行排版、编辑、打印等操作。

. 可把数据上报公司机动部、计财部拷贝到软盘。

由于机动部和分厂的网络环境及实现的功能不同，因而用 UML 进行建模时也存在一定的差异。从下一章开始，将统一建模语言应用于机动部和分厂管理信息系统的建模过程中。

第四章 UML 在建模合同管理信息系统中的应用

4.1 UML 进行 OOA &D 工作的步骤

4.1.1 用 UML 进行可视化建模

模型是现实的简化，模型提供了系统的设计图。模型可以包含详细的规划，也可以包含概括性的规划，这种规划高度概括了正在考虑的系统。好的模型包括那些具有高度抽象性的元素。每个系统都可以使用不同的模型，从不同的方面来描述，因此，每个模型从语义上来说都是系统的封闭抽象。模型可以是结构性的，强调系统的组织；也可以是行为性的，情调系统的动态。

收集系统需求时，把用户的业务需求映射成开发小组能理解的要求，最终可利用这些需求产生代码。通过将需求映射为代码，可以保证代码满足这些需求，代码也能方便地回溯成需求，这个过程称为建模。建模过程的结果就是可以跟踪从业务需求、到要求、到模型、到代码的过程及其相反的过程，而不会在这个过程中迷路。

可视化建模将模型中的信息用标准图形元素直观地显示。标准对实现可视化建模的通信功能至关重要。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的人员之间的通信。利用非可视信息如文本，也能进行通信，但人类毕竟是视觉动物，通过图形比通过文字更容易理解事务的结构。利用系统的可视化建模，可以在几个层次上显示系统如何工作。我们可以建模用户与系统间的交互，可以建模系统对象间的交互，甚至可以建模系统之间的交互。

建立模型后，可以向所有相关部门演示这个模型，让人们对模型中的重要信息一目了然。用户可以通过模型直观地看到用户与系统间的交互；分析人员可以看到模型对象间的交互；开发人员可以看到要开发的对象和每个对象的任务；测试人员可以看到对象间的交互并根据这些交互准备测试案例；项目管理人员可以看到整个系统及各部分的交互；而信息总管可以查看高层模型，看看公司的各个系统如何相互交互。总之，可视化建模提供了向各有关部门显示系统计划的强大工具。

可视化建模的一个重要问题是用哪些图形标注方法表示系统的各个方面。这个标注方法应能向各有关方面传达意图，否则模型就用处不大。

在面向对象开发中，存在很多种可视化建模图注方法，在前面介绍的最常用的几种图注方法中，统一建模语言 UML 是最成熟、最常用、最受欢迎的建模语言。在世界范围内，至少近十几年内，UML 将长期统领建模领域。

统一了的建模语言 UML 是一种定义良好、富于表达、功能强大且普遍适用的建模语言。它融入了软件工程领域的新思想、新方法和新技术。它不但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的全过程。

4.1.2 用 UML 建模的原则

建模不仅仅用于大系统。即使是规模很小的软件，也可以从建模中受益。不过，系统越大、越复杂，建模的重要性就必定更加明显，如果复杂的系统不建模，人们就无法全面理解这样的系统。人类理解复杂性的能力是有限的。通过建模，一次只侧重于问题的一个方面，从而简化了问题的研究。

建模要实现四个重要目的：

- 模型有助于按原样或根据需要使系统可视化。
- 通过模型可以详细说明系统的结构或行为。
- 模型可以提供一个指导我们构建系统的模板。
- 模型可以记录已经做出的决策。

建模的原则：

(1) 选择创建什么模型对于如何处理问题以及如何形成解决方案有很深远的影响。

应该谨慎地选择模型。如果模型合适，即使对最棘手的开发问题，模型也能很好地进行描述，给人以其他模型可能无法提供的启示；而不适当的模型会起误导作用，将侧重点转移到不相关的问题上。

(2) 每一种模型可以在不同的精度级别上表示。

在任何情况下，最佳类型的模型是这样的：它允许根据查看模型的人以及他们为什么需要查看模型来选择模型的详细程度。分析员或最终用户侧重于了解问题是什么，而开发人员则注重如何解决问题。但这两类人都希望以不同的详细级别在不同的时候实现系统的可视化。

(3) 最佳模型与现实情况是紧密相连的。

在软件中，结构化分析技术的致命弱点是，分析模型与系统的设计模型之间基本没有什么关联。随着时间的推移，这个问题将导致所建立的系统与所设想的系统大相径庭。而在面向对象的系统中，可以将系统中所有几乎相互独立的视图联系起来，成为一个语义整体。

(4) 一个模型不足以全面地反映实际情况。每个具有一定规模的系统都是通过一组几乎独立的模型来表示的。

其中，用例视图（揭示系统需求）、设计视图（确定问题空间和解决方案空间的专用词汇）、过程视图（对系统的过程和线程的分布进行建模）、实现视图（侧重于系统的物理实现）和部署视图（侧重于系统工程问题）。其中每个视图都可能具有结构和行为两个方面。这些视图和在一起代表了软件的设计图。

4.1.3 用 UML 建模面向对象系统的过程

UML 的目标是以面向对象图的方式来描述任何类型的系统，具有很宽的应用领域。从应用的角度看，当采用面向对象技术设计系统时，首先是描述需求；其次根据需求建立系统的静态模型，以构造系统的结构；第三步是描述系统的行为。其中在第一步与第二步中所建立的模型都是静态的，包括类图(包含包)、对象图、组件图和部署图等四个图形，是统一建模语言 UML 的静态建模机制。其中第三步中所建立的模型或者可以执行，或者表示执行时的时序状态或交互关系。它包括用例图、状态图、活动图、顺序图

和协作图等五个图形，是统一建模语言 UML 的动态建模机制。

UML 适用于系统开发过程中从需求规格描述到系统完成后测试的不同阶段。用 UML 进行系统开发的过程如下：

需求分析阶段：使用用例图来捕获用户需求。通过用例建模，描述对系统整体结构起作用的外部角色及其对系统(用例)的功能要求。对每个功能模块通过一组顺序图和协作图来描绘对象之间的交互，将状态变化包括在内。在该阶段，只对问题域的对象(现实世界的概念)建模，而不考虑定义软件系统中技术细节的类(如处理用户接口、数据库、通讯和并行性等问题的类)。

设计阶段：对分析阶段得到的模型进行反复筛选提炼。包括开发和细化对象图、修改活动图、开发组件图、制定部署计划、设计和开发用户界面原型及测试用例等。

编程阶段：用面向对象编程语言将来自设计阶段的类转换成实际的代码，并根据编码过程中出现的问题对建立的模型作相应的调整。在用 UML 建立分析和设计模型时，应尽量避免考虑把模型转换成某种特定的编程语言。因为在早期阶段，模型仅仅是理解和分析系统结构的工具，过早考虑编码问题十分不利于建立简单正确的模型。

此外，UML 模型还可作为测试阶段的依据。系统通常需要经过单元测试、集成测试、系统测试和验收测试。不同的测试小组使用不同的 UML 图作为测试依据：单元测试使用类图和类规格说明；集成测试使用部件图和合作图；系统测试使用用例图来验证系统的行为；验收测试由用户进行，以验证系统测试的结果是否满足在分析阶段确定的需求。

总之，统一建模语言 UML 适用于以面向对象技术来描述的任何类型的系统，而且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。

4.2 合同管理信息系统用例图

根据上一节用 UML 进行可视化建模的原则和过程，在使用 UML 进行系统开发的需求分析阶段，我们首先设计了系统的用例图。这些用例图的设计实现是与用户不断探讨研究的结果。然后，对每个用例建立顺序图、协作图和活动图等用来描绘系统的动态行为的模型。

在设计阶段，对前一阶段的模型进行筛选提炼后，又设计了系统的静态行为模型，本章以类图为例，在下一章中还有组件图和部署图。为了便于管理，我们将类图按功能进行了组合，形成了系统包图。

4.2.1 系统用例图

用例图用于需求分析阶段，它的建立是通过与用户反复讨论的结果，表明了我们与用户对系统需求达成的共识。

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，从外部参与者的角度来理解系统；另外，它带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，而且用于验证和检测所开发的系统是否满足用户需求，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，用例图的主要元素是用例和参与者。

由于合同管理系统主要提供业务人员使用，因此公司的业务员为一个系统行为者。

另外，在执行使用用例之时，会使用到后端整合数据库的内容，因此使用的整合数据库(ERP 数据库)，是另一个系统行为者。

通过较长时间的调研及不断与用户交流，在需求分析阶段，根据机动部系统实现的功能，我们对机动部设计了用例图。机动部合同管理信息系统使用用例模型图如图 4.1 所示。

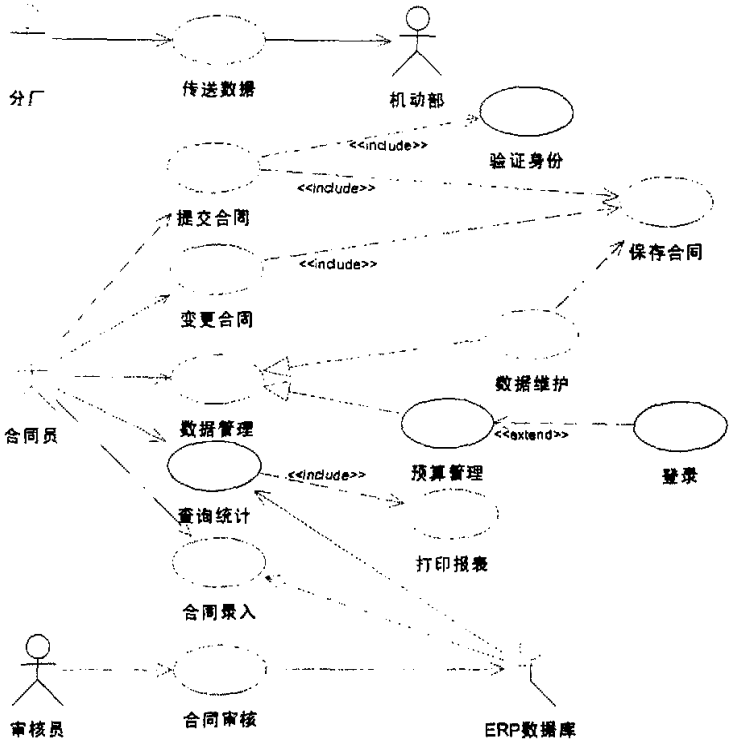

图 4.1 系统部分用例图

Fig. 4.1 Initial use case diagram

4.2.2 用例图中的参与者分析

- 业务员

结合公司或单位需要，在首钢内部或外部调研、引进适合的合同。

- 合同员

具有最高权限的管理员。进行系统维护，可以添加、修改和删除用户；管理统计各合同信息；同时对系统运行过程中产生的各类报表进行管理，包括送给领导审阅的报表和送到财务科作为财务核算依据的报表。对总库中的数据进行备份保存，防止数据被破

坏。如被破坏用备份数据进行恢复。

- 机动部

负责定时从各分厂接收合同的相关、预算信息等。工程合同信息由机动部统一管理，包括上报合同信息的入库，修改、检索、删除、浏览过程；

- 分厂

是受益参与者，传送合同详细信息及相关预算信息等。对总库中的数据进行分割保存，把要的数据从总库中分离出来，进行存档，数据库中还有这些数据。如把上一年的数据从总库中分离出来，进行存档。

- 审核员

负责对合同及预算信息进行检验，包括合同的类型情况及预算费用等。对于不合格合同，需填写不合格原因，这类合同不准入库；并负责管理合同档案，包括管理合同报表和合同档案等。

- ERP 数据库

机动部应进行合同质量管理，同时将所用数据提供给首钢总公司 ERP 作为基础数据。

以上对管理信息系统的参与者进行了描述，通过对需求的进一步分析，得出了如图所示的若干用例。在对用例进行描述之前，首先对用例间存在的关系进行说明。

在用例之间存在着类属(Generalization)、包含(Include)和扩充(Extend)关系。应用这些关系是为了抽取出公共行为和变种。

- 类属关系

用例间的类属关系如同类间的类属关系。也就是说，子用例继承父用例的行为和含义，子用例可以添加新行为或覆盖父用例的行为。

用例间的类属关系的表示方式与类间类属关系的表示方式相同，都用带空心箭头的实线表示，箭头由子用例指向父用例。

- Include 关系(包含关系)

多个用例可能具有一些相同的功能，共享的功能通常被放在一个单独的用例中，在这个新用例和其他需要使用其功能的用例之间创建 Include 关系。

用例间的包含关系表示在基用例的指定位置，基用例显式地包含另一个用例的行为。被包含的用例是不能独立存在的，只是包含它的更大的用例的一部分。

使用 Include 关系可以避免重复描述同样的事件流，因为公共的行为被放入一个专门的用例中，这个专门的用例是被基用例包含的。

在 UML 中，Include 关系可以用原型为<< Include>>的依赖关系表示。

- Extend 关系(扩充关系)

扩充关系用来说明可选的、只在特定条件下运行的行为，具有扩充关系的用例基于参与者的选择，可以运行几个不同的流。

用例间的扩充关系表示基用例在指定的扩充点隐式地含有另一个用例的行为。基用例可以独立存在，但在特定条件下，它的行为会被另一个用例的行为扩充。基用例只在被称为扩充点的特定点被扩充。可以认为，扩充用例将行为推进基用例。

扩充关系被用来描述特定的用例部分，该用例部分被用户视为可选的系统行为。这样，就将可选行为与义务行为区分开来。扩充关系还被用来为只在给定条件下执行的独立的子流建模。扩充关系用原型为<<extend>>的依赖关系表示，并可在基用例中列出基用例的扩充点。

包含关系(抽取公共行为)和扩充关系(识别变种)对于创建简单、易于理解的系统用例集是非常重要的。

4.2.3 用例具体描述

根据以上内容，找出了主要的使用用例。接下来配合企业流程执行顺序图功能需求的分析，将针对此使用部分用例，做具体的内容描述，如下列表 4.1-4.8 所示。

表 4.1 用例（1）:系统登录

Table 4.1 Use Case: Login System

用例名：	系统登录
执行者：	合同员
使用用例目的：	使用者登录系统
使用用例描述：	当合同员提出进入合同管理系统要求后，为了确保系统的安全性，业务员需要输入帐号、密码、权限，进行登录查核的工作。系统会与后端数据库中帐号、密码、权限进行比对，密码查核通过，才可进入系统首页。

表 4.2 用例（2）：查询

Table 4.2 Use Case: Require

用例名：	查询
执行者：	合同员、ERP 数据库
使用用例目的：	查询产品信息、客户信息、合同执行情况
使用用例描述：	进入合同管理系统后，为了方便业务员了解数据库中，有哪些的产品，业务员可进行产品查询请求，了解后端数据库中产品的名称、规格、价格，提供给业务员来进行产品的选择；客户查询，了解客户的信息，以便更好的为客户服务；合同查询，了解合同的执行情况。

表 4.3 用例（3）：合同审核

Table 4.3 Use Case: Contract Confirmation

用例名：	合同审核
行为者：	审核员、ERP 数据库
使用用例目的：	审查合同是否符合标准，并给与批示。
使用用例描述：	接到待审合同后，首先审核该客户的资金情况，此次金额等信息后，做出判断；给出意见。

表 4.4 用例（4）：打印表报

Table 4.4 Use Case: Print Table

使用用例：	打印表报
行为者：	合同员、ERP 数据库
使用用例目的：	打印各部门所需要的数据表
使用用例描述：	合同员根据用户的要求打印固定的一些表格、也可以把表报数据或统计结果转到 Excel 中，用户根据自己的需要对数据进行排版、编辑、打印等操作。

表 4.5 用例（5）：预算管理

Table 4.5 Use Case: Manage Budget

使用用例：	预算管理
行为者：	合同员、ERP 数据库
使用用例目的：	管理各厂合同的预算信息
使用用例描述：	合同员核实施工单位申报预算金额、二炼钢审定预算金额、预算初审人、终审预算金额等；另外负责核减预算金额管理，确保预算金额的准确性，同时对预算时间进行管理。对合理预算信息，信息齐全的标定时间信息，包括：送计财部时间、自计财部取回时间。

表 4.6 用例（6）：数据维护

Table 4.6 Use Case: Data maintenance

使用用例：	数据维护
行为者：	合同员、ERP 数据库
使用用例目的：	对合同数据、预算数据进行备份保存，防止数据被破坏。如被破坏用备份数据进行恢复。
使用用例描述：	对库中的数据进行分割保存，将数据从总库中分离出来，进行存档，对总库中分割出的数据与总库中数据进行合并，还可对库中的数据进行删除。

表 4.7 用例（7）：变更合同

Table 4.7 Use Case: Alter Contract

使用用例：	变更合同
行为者：	合同员、ERP 数据库
使用用例目的：	修改合同闭口
使用用例描述：	对合同类型为事先和开口的合同进行修改，只能修改合同金额，输入是否闭口，闭口金额、闭口时间并进行存盘。其中，闭口类型：闭口或成就。闭口金额：当输入闭口金额时，要求同时把表中的实际金额修改，用闭口金额替换实际金额。

表 4.8 用例（8）：传送数据

Table 4.8 Use Case: Transfer Data

使用用例：	传送数据
行为者：	分厂、机动部
使用用例目的：	将分厂合同数据传送给机动部
使用用例描述：	将合同数据按照数据库设定的格式，通过网络等通信方式上报给首钢总公司机动部进行审批，汇总，并入总库。

4.2.4 系统的服务设计

在这一过程当中，主要的意义是，针对每个使用用例，找出用以解决合同员需求的相关系统的服务。而利用具体的使用用例描述，可找出这些服务。在系统登录使用用例中，有核查账号、权限等的服务，在查询的使用用例中，查询合同相关情况等的服务，而在合同审核使用用例中，有查询合同资金、查询合同施工单位其它合同执行情况及审批合同等的企业服务。如表 4.9－4.16 所示。

表 4.9　Use Case：系统登录

Table 4.9　Use Case: Login System

Use Case Step	Use Case Step Detail
使用者帐号确认	登录系统请求
	要求输入帐号、权限
	比对使用者帐号
	进入系统首页

表 4.10　*Use Case*：查询统计

Table 4.10 Use Case: Require and Count

Use Case Step	Use Case Step Detail
合同信息综合查询	合同查询请求
	查询结果合同内容、施工单位、闭口情况
	列出结果合同内容、施工单位、闭口情况
合同信息综合统计	选择统计项目
	选择统计条件
	列出结果：某种合同类型的合同份数、合同总金额、及占据的比例份额
预算信息综合查询	预算查询请求
	查询本合同的资金来源情况、初审预算金额、终审预算金额、核减预算金额、合同执行情况等
预算信息综合统计	选择统计项目
	统计预算份数、施工单位预算申报金额、核减金额等

表 4.11 Use Case：合同审核

Table4.11 Use Case: Contract Confirmation

Use Case Step	Use Case Step Detail
查询待审批的合同	接到待审合同后，首先审核该客户的资金情况
	判断金额、合同类型、闭口是否合理
	给出审批结论，做标记和数字签名。合格通过，否则核减合同金额
	更新数据库

表 4.12　Use Case：打印表报

Table 4.12　Use Case: Print Table

Use Case Step	Use Case Step Detail
合同信息打印	打印请求输入
	显示符合条件合同信息
	打印输出或转出 Excel 文档
报计财部预算打印	输入送计财部时间
	显示预算数据：预算个数、分项的预算个数、及分项的预算个数占预算个数的百分比等
	打印输出或转出 Excel 文档
首钢检修预算收文台帐	选择或输入送审日期、送审单位和送审人
	结果预览、打印或转出 Excel 文档

表 4.13　Use Case：预算管理

Table 4.13　Use Case: Manage Budget

Use Case Step	Use Case Step Detail
信息编号	合同员输入预算编号
	搜索该预算编号是否被使用过，报警后提示重新编号
	搜索到预算编号号未使用过，编号成功
预算资金管理	填写预算资金
	核实施工单位申报预算金额、二炼钢审定预算金额、预算初审人、终审预算金额等
核减预算金额管理	判断核减预算金额
	如果二炼钢审定预算金额为执行中标价格、计划向量时，二炼钢核减预算金额应该等于二炼钢审定预算金额；计财部核减预算金额计算方法：计财部核减预算金额=施工单位申报预算金额—终审预算金额。
预算时间管理	判断预算信息是否合理，信息齐全
	若符合标准，则标定送计财部时间、自计财部取回时间。否则，给出信息提示。

表 4.14　Use Case：数据维护

Table 4.14　Use Case: Data maintenance

Use Case Step	Use Case Step Detail
备份保存	选择备份库
	确定备份时间
	确定备份目的地文件夹
分割保存	选择需要分割数据
	选择数据的年份
	分离数据
数据合并	选择已经分割出来的数据
	与总库数据汇总、比对
数据删除	选择需要删除的数据
	确定该数据已经进行分割处理，若是则删除，否则，给出提示信息

表 4.15　Use Case：变更合同

Table 4.15　Use Case: Alter Contract

Use Case Step	Use Case Step Detail
合同浏览	输入查询请求或定位合同序号
	查询合同信息
	修改错误信息
修改金额	判断合同类型
	对合同类型为事先和开口的合同进行修改，只能修改合同金额，输入是否闭口，闭口金额、闭口时间并进行存盘。
还原合同	输入合同号或委托编号
	判断是否恢复到当前的操作的前一个操作；若是，进行还原。

表 4.16　Use Case：传送数据

Table 4.16　Use Case: Transfer Data

Use Case Step	Use Case Step Detail
数据传输	利用 INTERNET 网传输
	用电话线、Modem 传输 （用 Windows 98 超级连接的功能或特定的发送软件）。
	用拷贝软盘

4.3 用例的事件流描述及对应的活动图

用例可以用事件流来描述，用例的事件流是对完成用例行为所需的事件的描述。事件流描述了系统应该做什么，而不是描述系统应该怎样做，也就是说，事件流描述是用域语言描述的，而不是用实现语言描述的。

通常，事件流文档的建立主要在细化阶段进行。开始，只是对执行用例的常规流(即用例提供了什么功能)所需的步骤的简单描述。随着分析的进行，通过添入更多的详细信息，步骤不断细化。最后，将例外流添加到用例的事件流描述中。

在 UML 中，活动图是为系统的动态行为建模的 5 个图之一。活动图主要是一个流图，描述了从活动到活动的流。活动是在状态机中进行的一个非原子的执行，它由一系列的动作组成。

动作是由可执行的不可分的计算组成，这些计算可以引起系统的状态发生变化或者返回一个值，例如调用另一个操作、发送一个信号、创建或破坏一个对象、或者是纯粹的计算等都是动作。

交互作用图强调从对象到对象的控制流，活动图则强调从活动到活动的控制流。活动图可以用来描述对象在控制流的不同点从一个状态转移到另一个时的对象流。交互作用图着眼于传递消息的对象，活动图则着眼于在对象间传递的操作。

活动图是根据对象状态的变化来确定动作与动作的结果。在活动图中，一个活动结束后将自动进入下一个活动；而在状态图中，状态的跃迁可能需要事件的触发。

活动图是一种特殊的状态机，在该状态机中，大部分的状态都是活动状态，大部分跃迁都是由源状态活动的完成来触发的。由于活动图是一种状态机，状态机的所有特性都适用于活动图，即活动图可以含有简单状态、组合状态、分支、分叉和联结。

通常可以将活动图用于两种情况：

(1) 为工作流建模

工作流通常被用于可视化、规范、构建和文档化系统的商业过程，在这种情况下，为对象流建模尤其重要。

在为工作流建模时：

- 确定工作流的中心。对于比较复杂的系统，用一个活动图描述所有令人感兴趣的工作流是不可能的。
- 选择与工作流有关的商业对象。为每个重要的商业对象创建一个泳道。
- 识别工作流初始状态的前置条件和工作流最终状态的后置条件。这对于确定工作流的边界是重要的。
- 从工作流的初始状态开始，规定随时间发生的活动和动作，将它们作为活动状

态或动作状态放在活动图中。

- 对于复杂的动作或多次出现的动作集合，可以将它们折叠为活动状态，再提供一个单独的活动图来展开活动状态。
- 用跃迁连接这些活动状态和动作状态，从工作流中的顺序流开始，然后考虑分支，再考虑分叉和联结。
- 如果在工作流中，有重要的对象被涉及，将对象放在图中，必要时说明对象属性值和状态的变化。

(2) 为操作建模

这种情况下，活动图被用作流程图。

在为操作建模时：

- 收集与操作有关的抽象，包括操作的参数、返回类型、操作所在的类的属性等。
- 识别工作流初始状态的前置条件和工作流最终状态的后置条件，还要识别出在操作执行过程中必须持有的操作所在类的不变量。
- 从工作流的初始状态开始，规定随时间发生的活动和动作，将它们作为活动状态或动作状态放在活动图中。
- 必要时使用分支、分叉和联结。

合同管理系统的部分用例事件流及对应的活动图描述如下：

① 登录（Log in）

本用例描述了用户如何登录到系统中。

前置条件 (Pre-Conditions)

没有。

后置条件 (Post-Conditions)

如果用例成功，参与者可以启动系统并使用系统所提供的功能。反之，系统的状态不变。

扩充点（Extension Points)

没有。

事件流

基流 (Basic Flow)

当用户希望登录到系统时，用例启动。

系统提示用户输入用户名和密码。

用户输入用户名和密码。

系统验证输入的用户名和密码，若正确（E-1），则用户登录到系统中。

替代流（Alternative Flows)

E-1: 如果用户输入无效的用户名和/或密码，系统显示错误信息。用户可以选择返回基流起始点，重新输入正确的用户名和/或密码；或者取消登录，用例结束。

该用例可以用如图 4.2 所示的活动图描述，首先系统提示用户输入用户名和密码，然后合同员输入上述信息后提交，系统验证用户名和密码是否正确，如若正确，则启动系统，否则，显示错误提示信息，并提示用户重新输入用户名和密码。

该活动图共有两个泳道，其中的对象分别是合同员和系统。

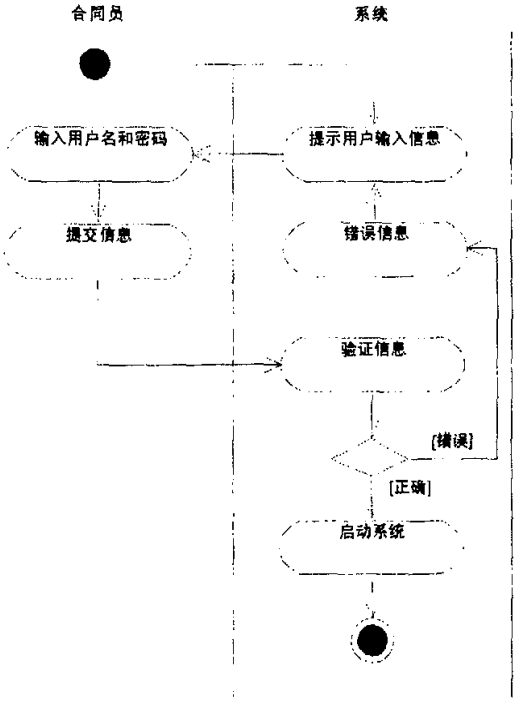

图 4.2 "登录"活动图

Fig. 4.2 Activity diagram of login

② 合同录入（Contract Input ）

前置条件 (Pre-Conditions)

在此用例开始前，合同员必须登录到系统中。

后置条件 (Post-Conditions)

如果此用例成功，系统添加、修改、查找或删除、提交合同信息。否则，系统的状态没有变化。

扩充点 (Extension Points)

没有。

事件流

基流 (Basic Flow)

当合同员要进行合同信息管理时，用例启动。

合同员对合同进行验收。

合同员根据合同信息填写委托编号、合同编号、合同内容、资金来源等信息。

如果合同信息齐全、编码正确，执行分支流 S-1：信息入库。

如果合同信息不全、编码错误，执行分支流 S-2：拒绝入库。

分支流 (Subflows)

S-1：信息齐全

系统提供合同的序号

检索委托编号（E-1）

合同编号（E-2）

确定施工单位。

选择合同资金来源。

选择修理类别。

选择合同类别。

存储信息。

S-2：信息不全

系统提示含有生产厂家的单位代码，单位名称和生产批号的域，填写完毕后，进行保存。此时弹出对话框，提示合同信息不全，拒绝入库。用例结束。

替代流（Alternative Flows)

E-1：委托编号已经存在，系统显示错误信息。用户可以选择返回基流的起始点，重新输入正确的委托编号；或者取消该次提交，用例结束。

E-2：合同编号已经存在，系统显示错误信息。用户可以选择返回基流的起始点，重新输入正确的合同编号；或者取消该次提交，用例结束。

该用例可以用如图4.3所示的活动图描述。

首先，系统弹出合同输入对话框，合同员根据真实情况输入合同信息。然后，合同员将此信息提交。系统对输入信息进行判断，如果合同编号已经存在，说明该输入合同的合同编号错误，则系统显示出错信息，并可返回输入点进行更改；否则系统继续判断委托编号是否存在，如果存在，则系统显示出错信息，返回输入点进行更改。接着，输入其他合同信息，确定施工单位、选择合同资金来源、选择修理类别、选择合同类别。一旦其他某项错误，系统均给合同员显示错误信息，同时该用例结束。若合同信息均正确，则该合同正确加载到系统中，在合同员泳道显示合同的正确信息，此刻合同员可修改合同其他信息，核实无误后，合同员可向系统提交保存请求，若系统保存有故障，系统向合同员发出存储错误信息，该用例终止，若正确保存，系统用例正常结束。

在此活动图中，出现了分叉和联结。分叉表示将单一的控制流分成两个或多个并发的控制流。此图中的分叉有一个输入跃迁和三个输出跃迁，每个输出代表一个独立的控制流。在分叉下面，与每个输出路径相关的活动是并行进行的。联结代表了两个或多个并发控制流的同步。此图中，联结有三个输入跃迁和一个输出跃迁。在联结以上，与各路经有关的活动是并行的。在联结处，并发的流同步，即每个流都要等到所有的输入流到达同步条，然后同步条将多个输入控制流合并，输出一个控制流，进而执行后面的活动。

分叉和联结应该是平衡的，也就是说离开分叉的控制流的数目应该与进入相应联结的控制流数目相等。

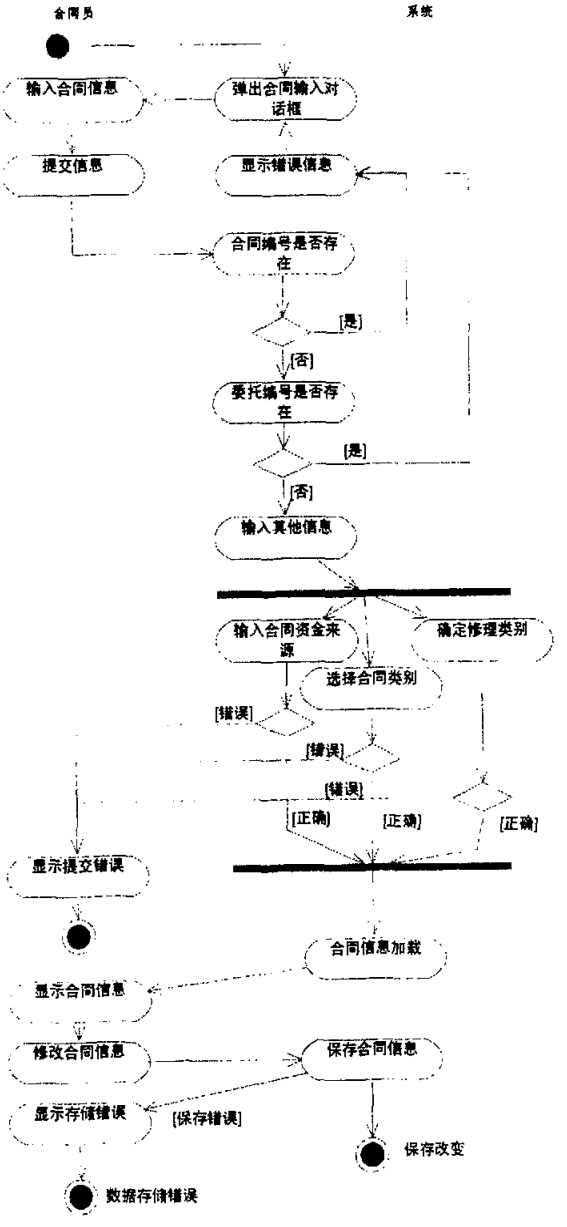

图 4.3 "合同录入"活动图

Fig. 4.3 Activity diagram of contract input

③ 数据维护（Data maintenance）

前置条件 (Pre-Conditions)

在此用例开始前，合同员必须登录到系统中。

后置条件 (Post-Conditions)

如果此用例成功，系统备份、删除、分割或合并数据。否则，系统的状态无变化。

扩充点 （Extension Points)

没有。

事件流

基流 (Basic Flow)

当合同员想维护数据时，用例启动。

系统要求合同员选择想要执行的活动(备份数据、分割保存数据、数据合并、数据删除)。

如果所选的活动是"备份数据"，则执行分支流 S-1：将数据备份。

如果所选的活动是"分割数据"，则执行分支流 S-2：将数据分割。

如果所选的活动是"数据合并"，则执行分支流 S-3：将数据合并。

如果所选的活动是"数据删除"，则执行分支流 S-4：将数据删除。

分支流 (Subflows)

S-1：将数据备份

选择备份库

确定备份时间

确定备份目的地文件夹

将数据存储到系统中（E-1）

S-2 ：将数据分割

选择需要分割数据

选择数据的年份

从总库中将数据分离出来

选择分离数据存储位置

将数据存储到系统中（E-1）

S-3 ：将数据合并

查找外部数据所在位置

选择已经分割出来的数据 （E-2）

合并数据到总库

与总库数据汇总、比对

更新系统中合同库的信息

S-4：将数据删除

提供所要删除的合同的年份信息

查询所要删除的内容（E-3）

确认要删除数据已经进行分割保存 （E-4）

删除物理数据

从系统中删除所有物理信息，并更新相关信息

替代流 (Alternative Flows)

E-1: 若文件名已经存在，系统显示提示信息，用例终止。

E-2: 若需要合并的数据没有找到，系统将显示错误信息。用户可以选择返回基流的起始点，重新选择已经分割出来的需要合并的数据；或者取消本次操作，用例结束。

E-3: 若查询不到该内容，系统显示提示信息，用例终止。

E-4: 若该数据未进行分割保存，系统显示提示信息，用户可以选择新的删除数据对象；或者取消本次操作，用例终止。

"备份数据"的活动图如图 4.4 所示。"分割数据"的活动图如图 4.5 所示。"数据合并"的活动图如图 4.6 所示。"数据删除"的活动图如图 4.7 所示。

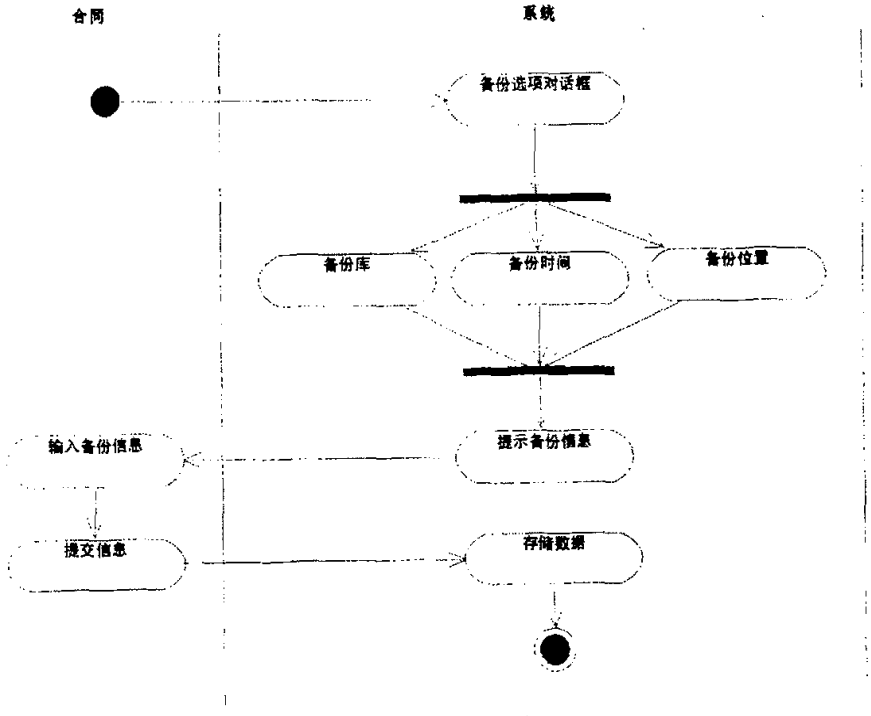

图 4.4 "备份数据"活动图

Fig. 4.4 Activity diagram of backup contract data

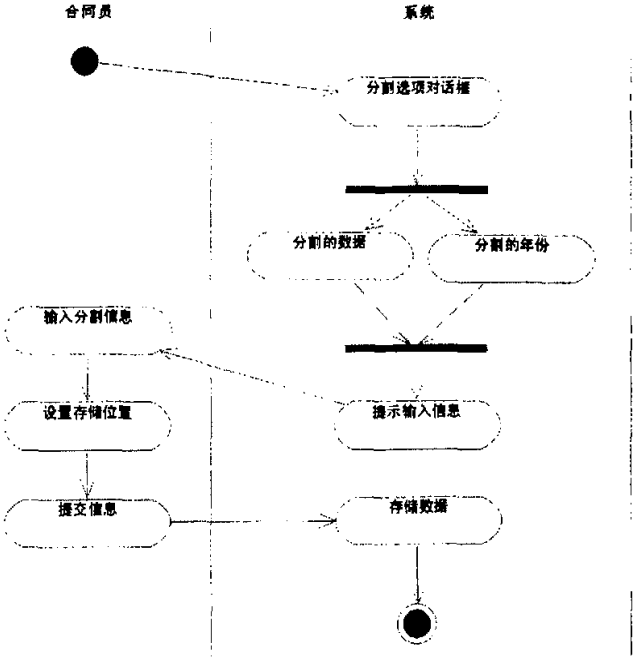

图 4.5　"分割数据"活动图

Fig. 4.5 Activity diagram of data partition

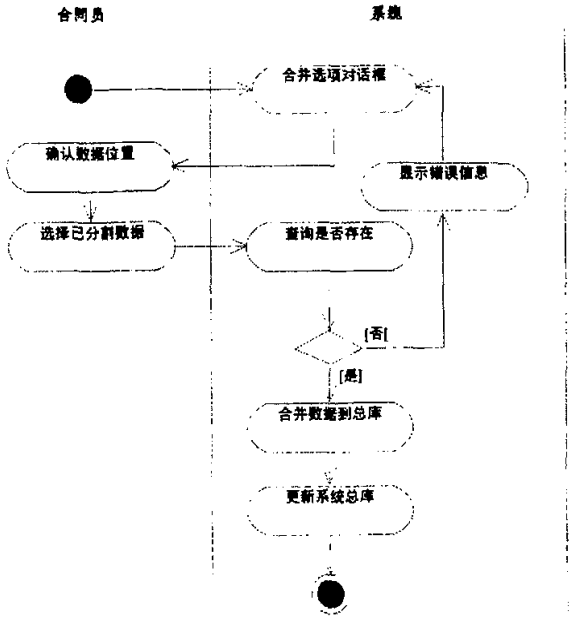

图 4.6　"数据合并"活动图

Fig. 4.6 Activity diagram of data uniting

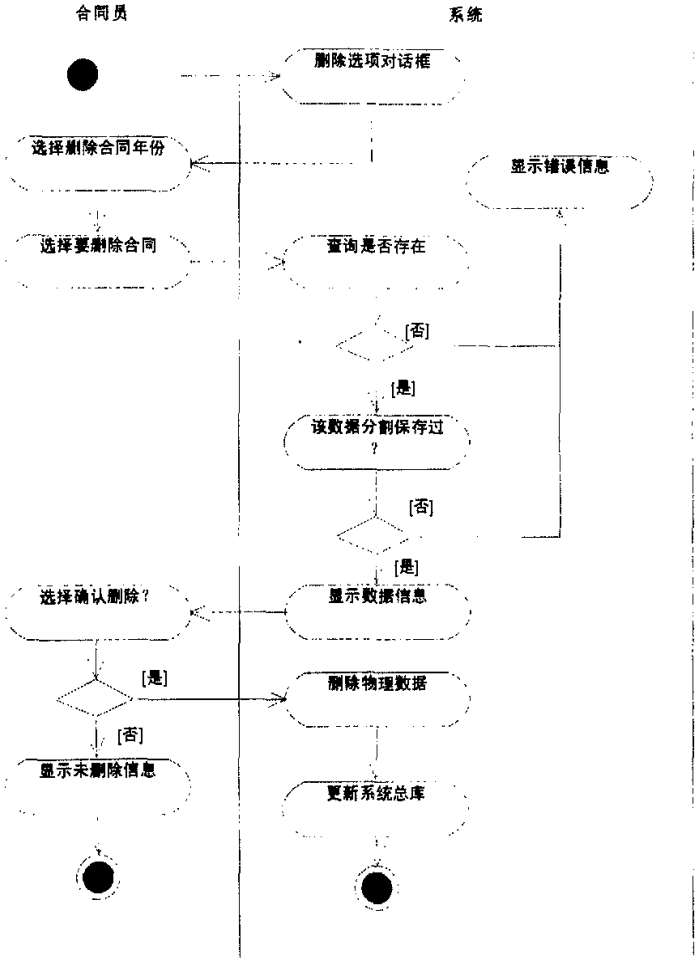

图 4.7 "数据删除"活动图

Fig. 4.7 Activity diagram of data to be deleted

4.4 合同管理信息系统中用例的交互图

顺序图和协作图都被称为交互作用图(Interaction Diagrams)，这两个图被用于为系统的动态行为建模。交互作用图描述了交互作用，由对象、对象间的关系组成，并包含对象间传递的消息。

顺序图和协作图以不同的方式表达了类似的信息，顺序图描述了消息的时间顺序，适合于描述实时系统和复杂的脚本；协作图描述了对象间的关系。顺序图是强调消息的时间顺序的交互作用图，图形上，顺序图是一个表，对象沿着 X 轴排列，消息按照时间

递增沿着 Y 轴排序。协作图是强调发送和接收消息的对象的组织结构的交互作用图。顺序图和协作图在语义上是相当的，可以彼此转换而不损失信息。

对应于系统中的每一个用例，都有一个顺序图和协作图，从不同的角度来描述用例中的各个对象之间的相互关系和先后动作的时间顺序。分析系统的用例也是发现对象的过程。下面说明描述本系统用例场景的顺序图。

4.4.1 合同提交入库用例的顺序图

在合同管理信息系统开发过程中，我们对一些主要用例(Use Case)，建立了顺序图(Sequence Diagram)，来分析消息交互的时间顺序，找出其主要情节，便于客户理解。从顺序图中可以找出系统的有关对象，及对象之间的相互关系。

下面举例说明顺序图的意义。

如图 4.8 所示，描述了合同管理信息系统基本数据功能模块中合同提交入库用例的顺序图。在该顺序图中，存在两个轴：水平轴表示不同的对象，垂直轴表示时间。其中的对象用一个带有垂直虚线的矩形框表示，图中的箭头表示消息传递的方向。此图中包含七个对象，分别是合同员、登录窗口、主菜单、审核窗口、入库窗口、合同档案和 ERP 数据库等。

“合同提交入库”用例的参与者是合同员，合同员类向边界类登录窗口对象发出启动系统的消息，登录窗口对象返回用来填写登录信息的对话框。合同员按对话框要求填写用户名和用户密码后，提交。注册窗口对象根据保管员输入信息进行用户名和密码确认。若正确，则发送消息给主菜单对象。 主菜单对象收到消息后，打开主菜单。合同员在主菜单中发送查询信息给审核窗口，审核窗口收到来自主菜单的消息后，在审核窗口返回审核信息。此时，合同员即可对审核员填写的审核单进行审核，若审核结论是同意入库，则合同员对审核单进行确认，并发送填写入库单信息给入库窗口。入库窗口对入库信息进行更新后，向入库合同档案 GUI 对象发送保存信息，进行入库单保存。最后，合同员将同意入库的合同送入 ERP 数据库。

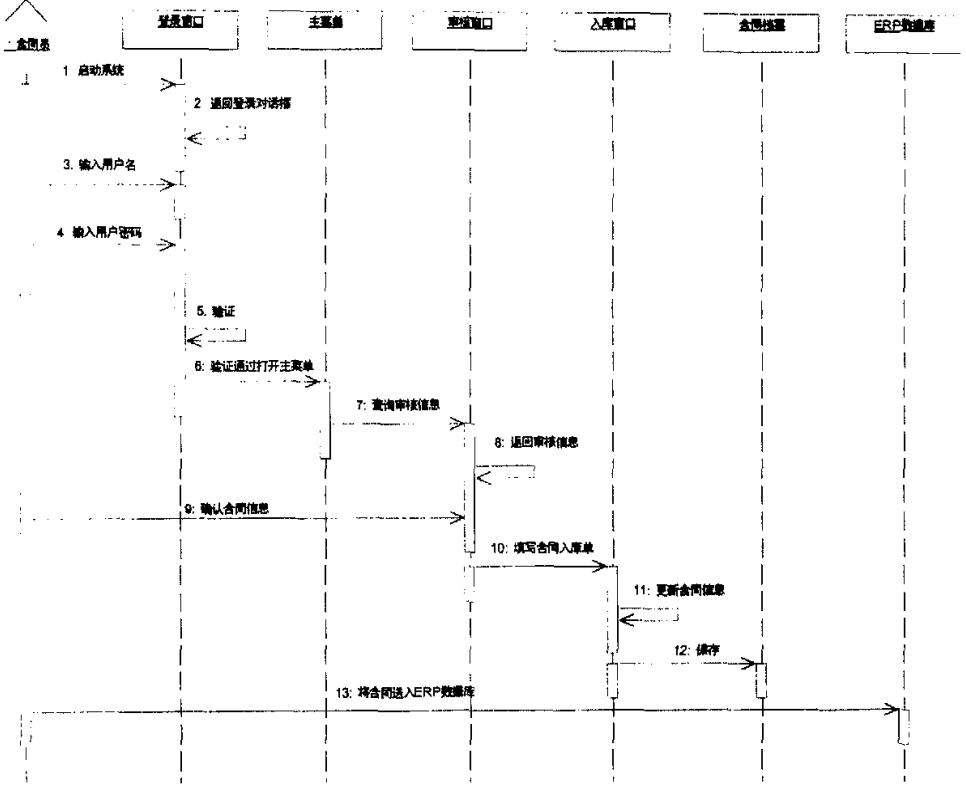

图 4.8 合同提交入库顺序图

Fig.4.8 The sequence diagram of submitting contract to DB

由合同提交入库顺序图可见，顺序图强调消息的时间顺序。不必显式地标出消息的序列号，因为顺序图中消息从顶到底的物理排序已经表明了消息的顺序。顺序图具有两个特点，区别与协作图：

(1) 有对象生命线

对象生命线是垂直的虚线，代表对象存在一段时间。出现在交互作用图中的大部分对象，在整个交互作用期间存在。对象也可以在交互作用的过程中创建。这些对象的生命线从接收创建该对象的消息开始。对象也可以在交互作用的过程中被破坏。

(2) 有控制中心

控制中心是细长的矩形，它表示了对象直接、或通过子过程执行一个动作的时间段。矩形的顶端和动作的开始对齐，矩形的底部和动作的完成对齐（可以用返回消息来标记）。

4.4.2 合同提交入库用例的协作图

协作图强调了参与交互作用的对象的组织。在形成协作图时，首先将参与交互作用的对象放在图中，然后连接这些对象，并用对象发送和接收的消息来装饰这些连接。协作图没有时间维，所以消息和并发线程的时间顺序必须由序列号表示。协作图描述了两个方面：对交互作用的对象的静态结构的描述，包括相关的对象的关系、属性和操作；为完成工作在对象间交换的消息的时间顺序的描述。

虽然顺序图和协作图都用来描述对象间的交互作用，但侧重点不一样：顺序图着重体现交互的时间顺序，协作图则着重体现交互作用的对象间的静态连接关系。

为此，可以对一个情景同时创建协作图和顺序图。彼此转换不损失信息。将合同提交入库实例的顺序图转换之后得到的协作图如图 4.9 所示。

此图中，可视化地表示出对象之间操作的相互调用关系。

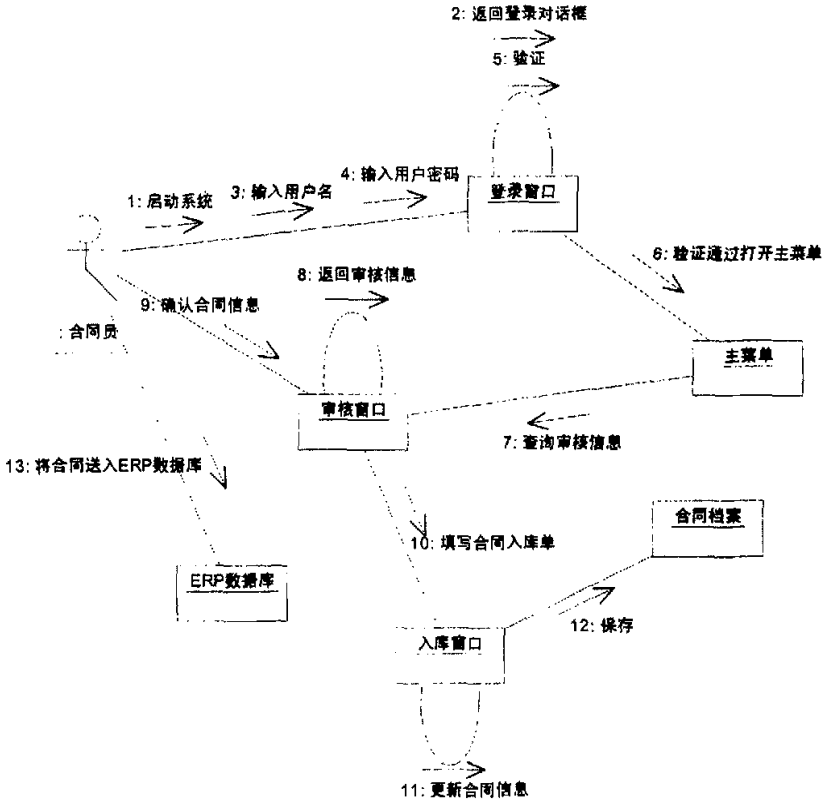

图 4.9 合同提交入库协作图

Fig.4.9 The collaboration diagram of submitting contract to the DB

协作图中的连接用于表示对象间的各种关系，消息的箭头指明消息的流动方向，消息串说明要发送的消息、消息的参数、消息的返回植以及消息的序列号等消息，由合同

提交入库协作图可见协作图有两个特点，区别于顺序图：

(1) 有路径

为了表示一个对象怎样与另一个对象连接，可以用在连接的远端添加一个路径原型。

(2) 有序列号

为了表示消息的时间顺序，可以给消息加一个数字前缀。第一个消息的序列号"1"，第二个消息的序列号为"2"，依次类推。为了表示嵌套，可以用杜威小数编号进行嵌套表示，嵌套可以为任意深度。在同一连接上，可以有多个消息，但每个消息都有一个独一无二的序列号。

4.5 建立机动部合同管理信息系统的静态结构模型

进一步分析系统需求，发现类以及类之间的关系，确定他们的静态结构和动态行为，是面向对象分析的基本任务。系统的静态结构模型主要用类图和对象图描述。

在对系统的用例图、交互图和活动图等进行细化基础上，我们设计了机动部合同管理信息系统的静态结构模型，以下主要以类图为例。为了便于管理，又按功能将类图划分为包，形成了系统的包图。

4.5.1 总店管理信息系统的类图

类图是面向对象系统建模最常用的图，类图描述了类集、接口集、协作以及它们之间的关系。类图是定义其他图的基础，在类图的基础上，状态图、协作图等进一步描述了系统其他方面的特性。

类图描述了系统的静态设计视图，该视图主要支持系统的功能需求——系统应该提供给用户的服务。在为系统的静态设计视图建模时，类图可以用来描述以下 3 种建模。

(1) 为系统的词汇表建模

模拟系统的词汇表涉及到确定哪些抽象是系统的一部分，哪些抽象不在系统的边界内。可以用类图规定这些抽象和它们的责任。

(2) 为简单的协作建模

类图可以用来为构成系统设计视图的部分元素和关系建模，因为这个原因，每个类图可以一次聚焦于一个协作。为协作建模时：

- 确定要为之建模的机制。机制代表了需要被模拟的部分系统的功能和行为，这些功能和行为是由类、接口等元素交互作用产生的。
- 对于每个机制，确定参与这个协作的类、接口和其他的协作，确定这些元素间的关系。
- 根据协作的脚本，发现遗漏的模型部分，以及简单的语义错误。
- 确定对象的属性和操作。

(3) 为逻辑的数据库模式建模

数据库的模式可以看作是数据库概念设计的蓝图。在许多问题领域，都需要在关系数据库或面向对象数据库中存储持久性信息。可以用类图来为数据库模式建模。

为数据库模式建模时：

- 确定模型中的一些类，这些类的状态的存在超过了程序的生命周期。

- 创建一个类图，在这个类图中含有这些类，并将这些类标记为持久类。
- 扩充这些类的结构信息，例如属性以及类的阶元等。
- 如果必要，创建中间抽象以简化数据库的逻辑结构。
- 考虑类的行为，扩充对于数据访问和数据完整性很重要的操作。
- 如果可能，用工具将逻辑设计转变为物理设计。

从机动部合同管理信息系统中寻找类，我们主要采用了两种办法：

从用例图和用例的事件流描述中，查找事件流中涉及的名词，从这些名词中可归纳出类。

检查顺序图和协作图中的对象，通过对象的共性即可寻找到类。

下面，对机动部合同管理信息系统中的部分类作一介绍。

在系统需求中可识别出系统中存在的对象，抽象出实体类：合同类、业务员类、合同员类、审核员类、预算单类、闭口合同单类、入库合同单类、机动部合同档案类、分厂合同档案类、分厂合同库类、机动部合同库类、ERP 数据库类等。

为了便于管理，除了合同类外，我们将以上其他类归纳成 4 个抽象类，即工作人员类(Employee Class)、单据类(Receipt Class)、档案类(File Class)和部门类(Department Class)。这些抽象类用作其它类的超类。

(1) 类 Employee

类 Employee 描述了工作人员的信息。工作人员的信息包括 ID 号、姓名、密码、权限等。类 Employee 的所有对象都是持久的（Persistent）。

私有属性（Private Attributes）：

id：String　工作人员的统一编号。

name：String 工作人员的姓名。

password：String 工作人员的密码。

privilege：String　工作人员的职责权限。

公共操作（Public Operations）：

register（id:String，name:String，password:String）

以工作人员的 ID 号、姓名、密码为参数进行身份验证，登录系统。

select（）:Integer

返回工作人员下一步操作的序号。

(2) 类 Receipt

类 Receipt 描述了工作人员对合同的操作信息。

私有属性（Private Attributes）：

id：String　单据的 ID 编号。

time：Date 单据的产生时间。

type：String　单据的类型：预算单、闭口合同单、入库合同单等。

operator：String 该单据的实现者是谁？是由哪些工作人员产生的单据。

conclusion：String 该单据的完成情况如何？

flag：String 允许操作的工作人员进行标志修改。

公共操作（Public Operations）：

add（id：String，time:Date, type:String, operator:String,conclusion:String, flag:String）

增加一个单据 Receipt。

update（）
更新数据库文件中的持久对象（类 Receipt 或其子类的对象）
该类的定义如下：

```
import java.util.*;
public class Receipt
{
    private String id;
    private Date time;
    private String type;
    private String operator;
    private String conclusion;
    private String flag;

    public Receipt()
    {
    }

    public update()
    {
        return true;
    }

    public add(String rid , Date rtime, String rtype,String roperator,String
rconclusion,String rflag)
    {
        id=rid;
        time=rtime;
        type=rtype;
        operation=roperation;
        conclusion=rconclusion;
        flag=rflag;
    }

    public static void main(String[] args)
    {
        Receipt receipt=new Receipt();
        System.out.println("instantiated Receipt");
        System.out.println(receipt.getClass().getName());
        System.out.println(receipt.id);
        System.out.println(receipt.operator);
        System.out.println(receipt.conclusion);
```

```
        try{
            Thread.currentThread().sleep(5*1000);}
            catch(InterruptedException e){}
            }

    }
```

(3) 类 File

私有属性（Private Attributes）：

id：String　档案的 ID 编号。

operator：String 该档案的实现者是谁？是由哪些工作人员产生的档案。

date：Date 档案的产生时间。

class：String 档案的类别。

公共操作（Public Operations）：

select（id:String, operator:String, date:Date）: File

选择符合条件的档案，返回指定 id 号的 File 对象。

maintain（id:String, operator:String）:File

档案维护更新。

(4)类 Department

私有属性（Private Attributes）：

name：String　部门名称。

manager：String 部门的负责人。

telNumber：String 部门的联系电话。

工作人员类包括业务员类（BusinessMan Class）、合同员类（ContractMan Class）、审核员类(Assessor Class)；

单据类包括预算单类（BudgetReceipt）、闭口合同单类（CorkContractReceipt）、入库合同单类（InWarehouseReceipt)。

档案类包括机动部合同档案类（JDBContractFile）、分厂合同档案类（BranchContractFile）；

部门类包括分厂合同库类（BranchFactory）、机动部合同库类（JDB）、ERP 数据库类（ERP)。

抽象类及其子类之间的关系如图 4.10 所示。

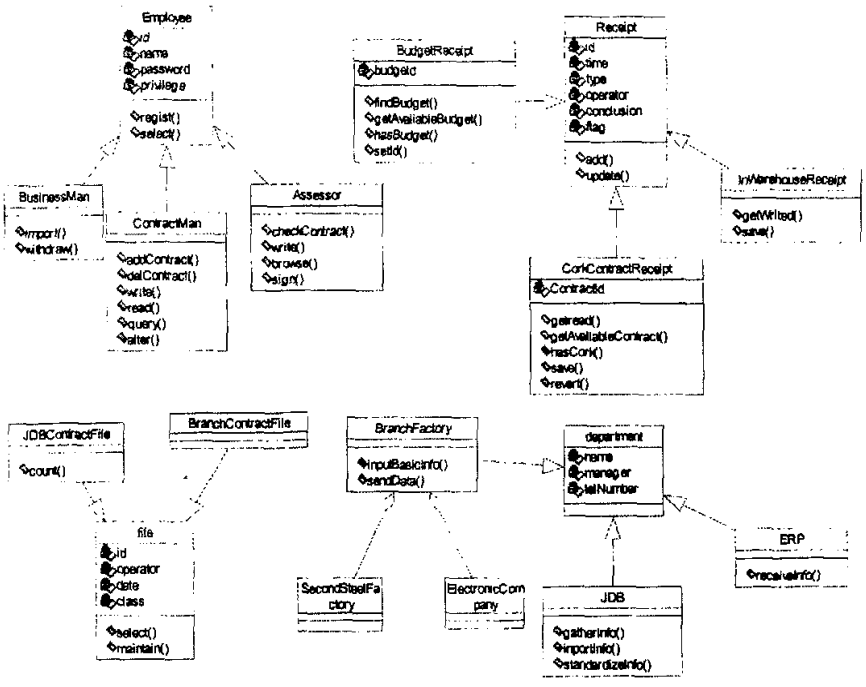

图 4.10 抽象类及其子类之间的关系

Fig.4.10 Abstract classes and their sub classes

识别出各抽象类与子类的关系后，还要识别出类间的关系，然后就可以建立类图了。如图 4.11，描述了实体类之间的关系。

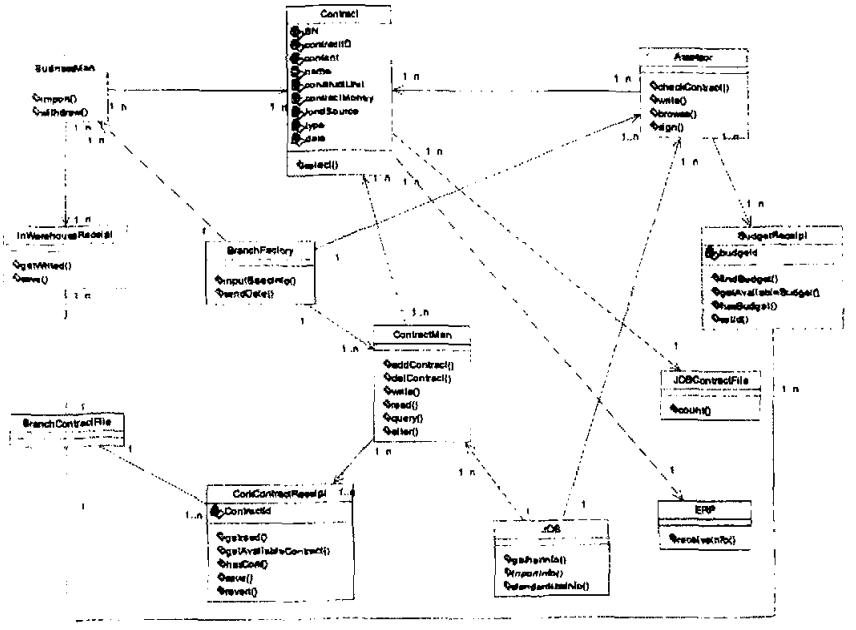

图 4.11 类图

Fig.4.11 The class diagram

在上述类图中，存在着一些关联关系，部分关联说明如下：

类 BusinessMan、类 ContractMan、类 Assessor 都是永久类，它们都是 Employee 的子类。它们与类 Contract 之间存在着"多对多"的关联关系，每个对象至少有一个 Contract 对象，每个 Contract 对象至少对应于一个 Employee 子类的对象。比如：一个或多个合同员录入一份或多份采购合同。其中，类 BusinessMan 与类 InWarehouseReceipt 之间存在"多对多"的关联关系；类 ContractMan 与类 CorkContractReceipt 之间存在"多对多"的关联关系；类 Assessor 与类 BudgeReceipt 之间存在"多对多"的关联关系。类 InWarehouseReceipt、类 CorkContractReceipt、类 BudgeReceipt 是类 Receipt 的子类。它们与类 BranchContractFile 均存在"多对一"的关联关系；类 Contract 与类 JDBContractFile 存在"多对一"的关联关系，一份或多份合同与一个机动部合同档案文件关联；类 Contract 与类 ERP 存在"多对一"的关联关系，一份或多份合同与一个 ERP 库文件关联。

在该类图中，同一个类可参与多种关联，以类 ContractMan 进行说明。类 ContractMan 参与的关联包括：一个或多个 ContractMan 在一个 BranchFactory 或 JDB 中；一个或多个 ContractMan 生成一份或多份 CorkContractReceipt；填写一份或多份 Contract。

另外，该类图中还存在一些双向关联关系。比如：一份或多份入库合同单（InWarehouseReceipt）与一份分厂合同档案（BranchContractFile）相关联。

4.5.2 机动部合同管理信息系统的包图

包将具有一些共性的类组合在一起。

包装类时有几种常用方法。一种方法是按版型组合类，另一种方法是按功能组合类，第三种方法是使用以上两种方法的组合。即包可以通过嵌套进一步组织类。

为了系统层次清晰、便于管理，我们将机动部合同管理系统的类按功能进行了组合，分成 3 个包：GUI 包，Application 包和 JDBDB 包。

包 GUI 由界面类组成，包 Application 由实体类组成，包 JDBDB 由与数据库有关的类组成。包 GUI 依赖于 Application 包和 JDBDB 包，Application 包依赖于包 JDBDB。其关系如图 4.12 所示。

Application 包是一个嵌套包，其中又分成 3 个包：入库类包，出库类包和查询类包。查询类包依赖于入库类包、出库类包。

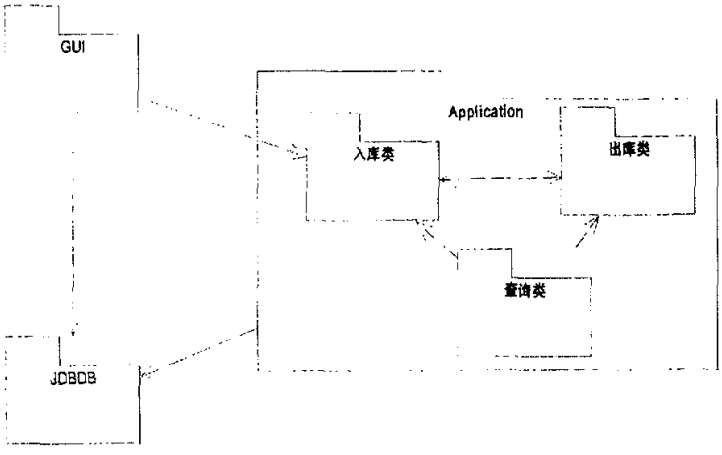

图 4.12 机动部合同管理信息系统包图

Fig.4.12 The package diagram of the Contract Management Information system

4.6 应用 UML 建模分厂合同管理信息系统

目前，许多单位以合同管理为中心控制投资。上述功能只是机动部合同管理系统软件的功能，对于机动部来说，需要将二炼钢等分厂的数据能够加载到机动部合同数据库中。否则，机动部对于其它厂数据只能自己输入，这样必然增加了数据出错的概率，给机动部增加了许多的工作量。因此为了提高工作效率，减少工作强度，减少重复录入，首钢总公司各厂检修合同数据，根据各厂的要求利用合同管理信息系统进行计算机管理，录入、提交、审核合同数据，对合同数据进行随机查询、统计、制作各种表报。同时实现数据共享，把各厂数据传输到公司机动部。

对于分厂而言，实现的数据量比机动部合同管理信息系统少得多。分厂实现的大部

分功能与机动部管理系统相同，不同之处主要表现在合同引入功能和传送数据功能模块上。

合同引入功能模块主要是各分厂业务员根据本单位业务发展情况，根据工程特点、规模、工期、施工难易程度来确定合同形式，向相关部门或单位确定合同意向。

无论采用哪种形式的施工合同，合同中都应明确工程价款是否调整、调整依据、调整方式、结算方式以及有关合同价款争议的解决方法，以减少投资控制的纠纷及隐患。在执行合同过程中，各分厂应注意按合同办事，按程序办事。该监理工程师管的事情，全权委托，决不插手。切忌自以为是，杜绝随意变更。工程确实需要变更之处，一定有文字记录，涉及到工程质量标准及其它实质性变更，一定经设计院同意。任何变更，都应及时计量、及时签认。由变更引起的价格调整，应及时确认调整价格，避免事后纠缠不清。事实证明，在施工过程中及时做好中间结算，可大大减少竣工结算的工作量，收到事半功倍的效果。

另外，分厂的主要部分在于传送数据模块的实现：向机动部合同管理系统上传合同数据信息。

4.6.1 分厂合同管理信息系统用例图

通过对分厂实现的功能进行分析，得出如图4.13所示的分厂合同管理信息系统用例图。

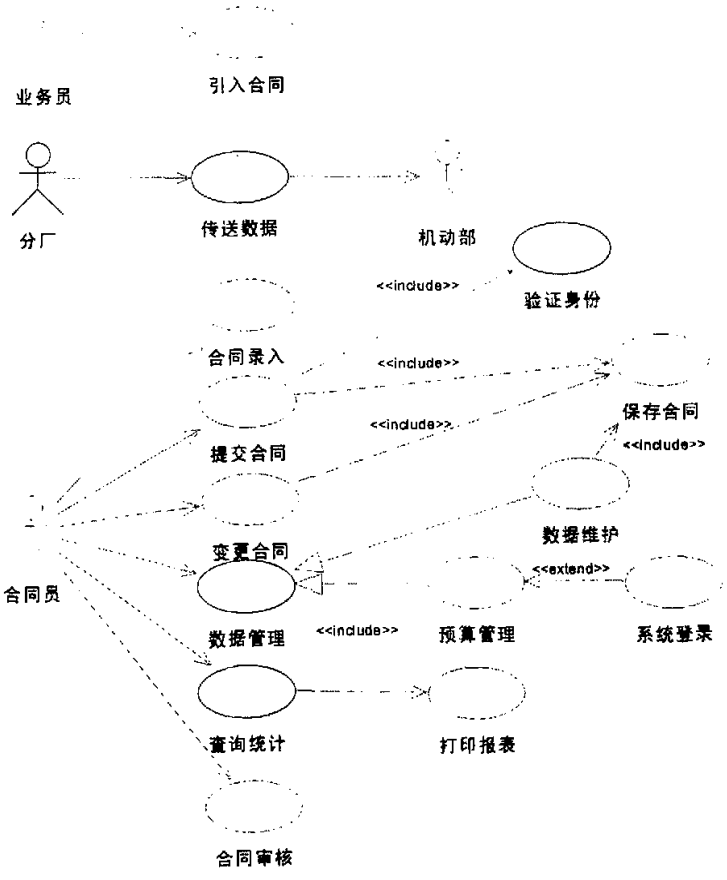

图 4.13 分厂系统用例图

Fig.4.13 The use case diagram of the branch factory

对该用例图中的参与者分析说明如下：

- 业务员

结合本公司或单位需要，在首钢内部或外部调研、引进适合的合同。

- 合同员

具有最高权限的管理员。进行本分厂系统维护，可以添加、修改和删除合同；管理统计各分厂合同信息；同时对系统运行过程中产生的各类报表进行管理，包括送给领导审阅的报表和送到财务科作为财务核算依据的报表。对本系统的合同数据进行备份保存，防止数据被破坏。如被破坏用备份数据进行恢复。

- 机动部

负责定时从各分厂接收合同的相关、预算信息等。工程合同信息由机动部统一管理，

包括上报合同信息的入库，修改、检索、删除、浏览过程；

- 分厂

是受益参与者，传送合同详细信息及相关预算信息等。

以上对分厂合同管理信息系统的参与者进行了描述，通过对需求的进一步分析，得出了如图所示的若干用例。对这些用例描述如下：

- 引入合同

本用例根据工程特点、规模、工期、施工难易程度来确定合同形式，向相关部门或单位确定合同意向。

- 传送数据

本用例提供了分厂阶段性地向机动部合同管理系统传送合同信息的功能。

- 合同录入

本用例提供将审查过合格的合同作上标记，使之成为正式合同的功能。

- 提交合同

本用例提供把合同的相关的信息输入入分厂数据库，即合同号、合同类型、金额资金来源、修理类别及施工单位、开工时间等，等待审核人员审批。

- 变更合同

本用例提供合同的变更，包括添加、删除、修改合同等功能。

- 数据管理

本用例提供数据的维护和合同预算管理、生成数据字典等功能。

- 查询统计

本用例提供对分厂合同信息进行综合查询、综合统计及打印的功能，计算出根据条件的预算个数、分项的预算个数、及分项的预算个数占预算个数的百分比，并显示或以标准格式打印统计结果。

- 合同审核

本用例提供审查合同是否符合标准，并给与批示的功能。

4.6.2 分厂部分用例交互图

4.6.2.1 分厂“合同录入”用例顺序图

在分厂系统开发过程中，我们对一些主要用例，也使用了顺序图(Sequence Diagram)，来分析对象之间交互的时间顺序。

合同录入用例是分厂系统的一个主要用例，因为这一用例的实现可以使合同员能及时掌握本单位工程合同的相关信息，以便及时为各部门提供统计、审核信息。

下面就来介绍分厂合同录入用例的顺序图，如图4.14所示。

该顺序图中包含五个对象，分别是合同员、登录窗口、主菜单、合同录入对话框、分厂合同档案等。

“合同录入”用例的参与者是合同员，合同员类向边界类登录窗口对象发出启动系统的消息，登录窗口对象验证输入信息。如果合同员输入了正确的用户名和密码，则验证通过，系统打开主菜单。合同员在主菜单中选择打开合同信息管理菜单项后，即可在该合同录入对话框中输入该合同的相关信息：委托编号、合同编号、合同金额、资金来源、

开工时间、竣工时间等。一旦确认输入信息没有错误，对填写内容进行确认，最后，保存该合同信息，更新分厂合同档案信息。

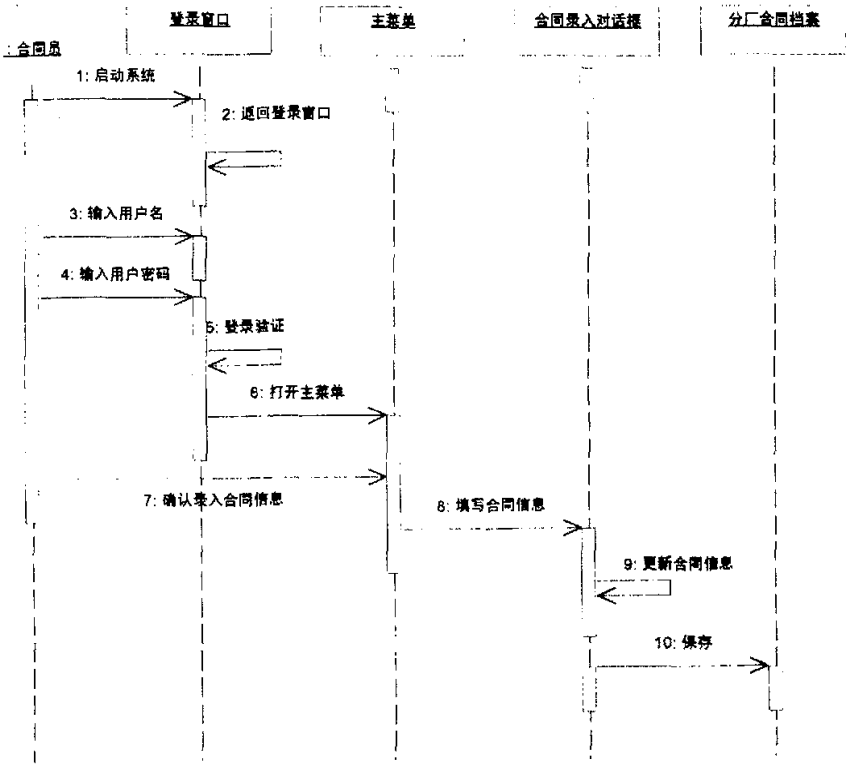

图 4.14 分厂合同录入顺序图

Fig.4.14 The sequence diagram of the branch factory's contract inputing

4.6.2.2 分厂"合同录入"用例协作图

由于协作图与顺序图是语义等价的，因此只给出分厂"合同录入"用例的协作图，对其含义在此不再赘述。

分厂"合同录入"用例的协作图见图 4.15 所示。

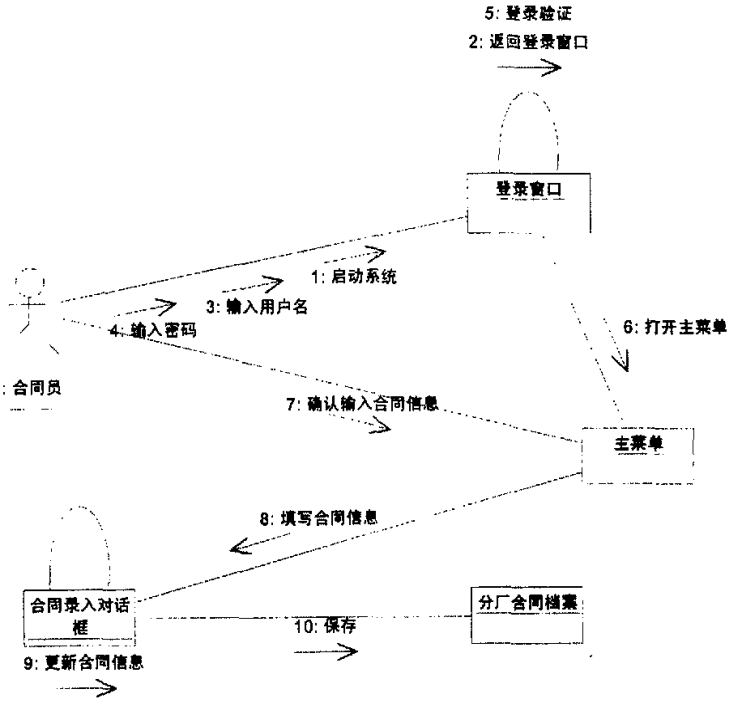

图 4.15 分厂合同录入协作图

Fig.4.15 The collaboration diagram of the branch factory's contract inputing

4.6.3 活动图、顺序图的传送数据工作流建模

由于机动部和分厂被设计成不同的管理系统，因而各分厂与机动部之间并非实时交互。为了保持机动部合同信息和分厂的合同信息的一致，分厂必须定时向机动部传送合同的更新信息，同时机动部还需掌握分厂合同的完成情况，进行全公司合同信息的汇总、统计、查询等，必要时还需要将审核、修改后的信息发送给各分厂。要达到这一目的，双方就必须进行必要的通信，实现资源共享，即各分厂阶段性地将合同信息发送给首钢机动部。

可采用的数据传送方式为：

(1) 利用 Internet 传送。

(2) 利用电话线、Modem 传输

（用 Windows 98 超级连接的功能或特定的发送软件）

合同数据传送示意图如图 4.16 所示。

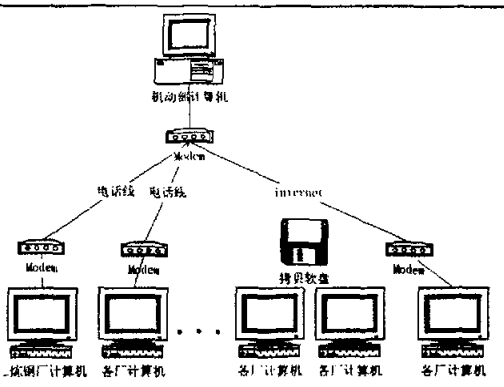

图 4.16 合同数据传送

Fig.4.16 The diagram of contract data transfering

各分厂通过各种渠道把合同数据加载到机动部的数据库中，机动部可对各厂传送的数据进行一些加工修改。同时，机动部对数据进行管理，对各厂数据进行处理，查询、统计、制作各种表报等。

分厂数据传送模块就是针对这一实际而设计的。为了提高传输性能，在传送时，只对数据库中发生变化的部分表进行更新操作。下面利用活动图和顺序图来描述数据传送模块功能。

4.6.3.1 活动图的工作流建模

活动图对用例尤其有用，可以提供明显的开始和结束状态。在用来建模用例的工作流时，活动图可以显示用例内部和用例之间的路径。

活动图可以向开发人员说明需要满足什么条件用例才会有效，以及用例完成后系统保留的条件或者状态。

分厂数据传送模块的活动图，如图 4.17 所示。

该活动图共有五个泳道，其中的对象分别是合同员、系统主菜单、传送文件队列、传送窗口和机动部。

本活动图的初始活动是合同员注册登录，打开系统主菜单。在主菜单中，查询需要传送的合同信息记录。系统接收到查询信息后，返回查询结果。合同员将查询到的传送记录保存到传送文件队列中。然后合同员进行远程连接操作，经过电话线、宽带连接到机动部合同管理系统。与机动部建立连接后，进入传送窗口，并启动传送程序。然后从传送文件队列中取出一个欲传送的文件，准备进行传送操作。此时，如果合同员不想进行传送操作或选择有误的话，可直接退出操作，数据传送活动结束。否则，进入活动图的分支中：分厂进行传送文件操作，同时机动部进行接收文件操作。选择的文件传送结束后，系统提示是否继续传送下一文件记录。如果选择"是"，则系统返回到传送文件队列处，重复前述操作，否则，退出操作，该传送活动结束。

分厂数据传送活动图清晰地显示了对象执行传送活动时所经历的步骤和判定点，易于客户的理解。

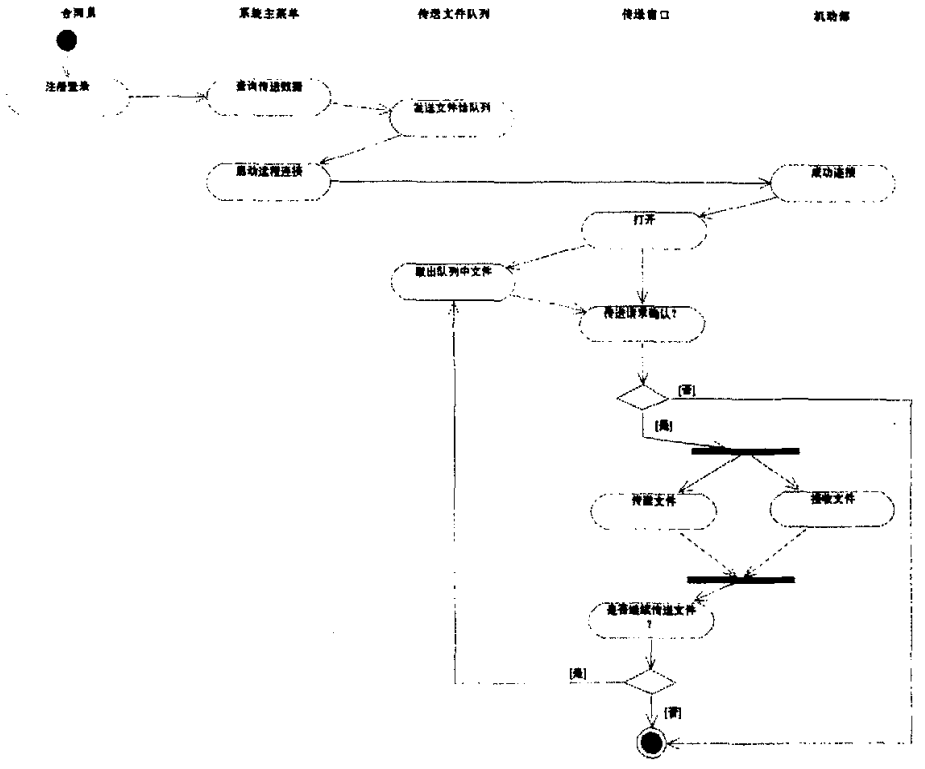

图 4.17 分厂数据传送模块活动图

Fig.4.17 Tthe activity diagram of the branch factory's data transferring module

4.6.3.2 顺序图的工作流建模

协作图强调参与交互作用的对象的组织，描述了两个方面：对交互作用的对象的静态结构的描述，包括对象的关系、属性和操作；为完成工作在对象间交换的消息的时间顺序的描述。

顺序图是两种类型的交互图之一。顺序图用来建模以时间顺序安排的对象的交互，并且把用例行为分配给类。顺序图贯穿 Unified Process 描述的软件开发生命周期的始终。顺序图与活动图具有类似的作用，都是用来实现用例。实际上，顺序图和活动图一样可以用来提供某个用例指定的泛化功能所缺乏的解释。顺序图可以用来演示某个用例最终产生的所有的路径，这一点和活动图类似。当需要按时间顺序为控制流建模时，一般需要使用顺序图，强调消息的传递，这对于可视化用例脚本上下文的动态行为是非常有用的。分厂数据传送模块的顺序图，如图 4.18 所示。

该活动图包含八个对象，他们分别是合同员、登录窗口、系统主菜单、查询 GUI、传送文件队列、远程连接窗口、数据传送窗口和机动部。

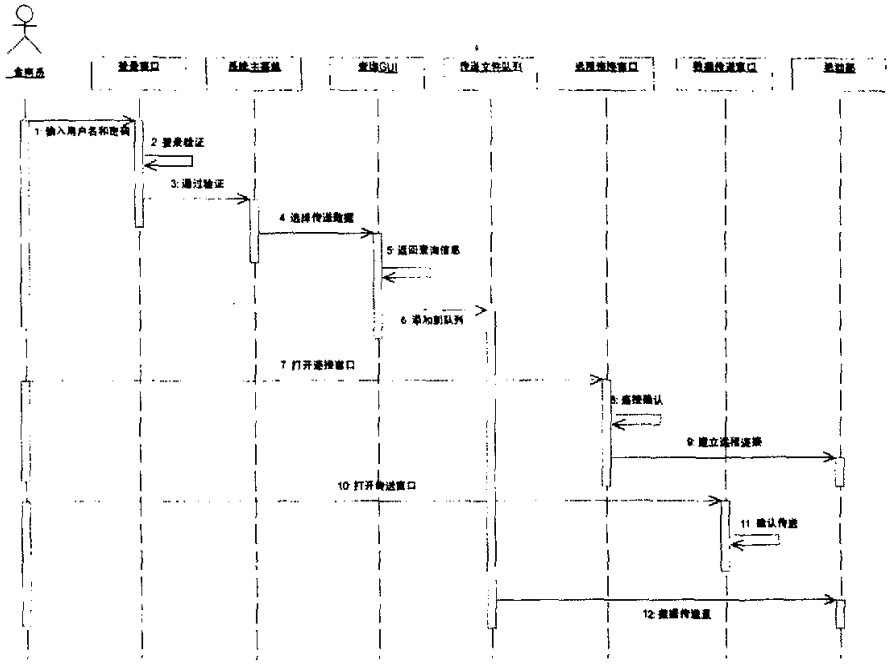

图 4.18 分厂数据传送模块顺序图

Fig.4.18 The sequence diagram of the branch factory's data transferring module

4.6.4 分厂合同管理信息系统包图

由于各分厂信息管理过程与机动部相似，也可将分厂管理系统中的类组合成三个包，即 GUI 包，Application 包，DB 包。其包图与机动部合同管理信息系统包图相同，如图 4.19 所示的管理包图。

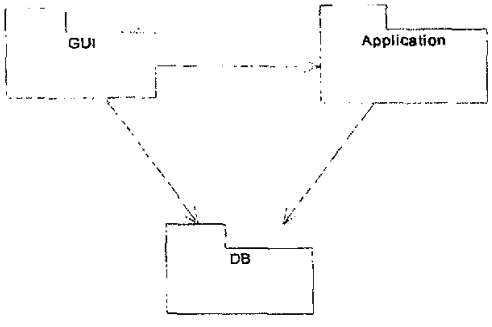

图 4.19 分厂合同管理信息系统包图

Fig.4.19 The package diagram of the contract management information system of branch factory

第五章 合同管理信息系统外观设计与部署

图形用户界面(GUI)决定了系统可使用性的好坏，而部署就是要实现预先计划好的系统物理体系结构。通过描绘系统的部署图，可以向用户显示合同网络中所有节点、节点间的连接和每个节点上运行的进程。在这一部分，用 Rational Rose 工具生成用于 GUI 设计的类图、状态图和系统的部署图。

5.1 GUI 设计的一般原则

用户界面设计不仅要讲究艺术性和科学性，还要考虑到系统用户的直观感觉。GUI 设计的一般原则如下：

- 理解用户要做什么。典型的用户界面设计都要进行任务分析(task analysis)来理解用户任务的性质。
- 让用户在与系统的交互中有掌握控制权的感觉。无论何时用户发起的交互都应该可以被取消。
- 要提供多种方式来完成每个与界面相关的动作(例如关闭一个窗口或者文件)并且能够友好地容忍用户操作中的错误。
- 由于受习惯影响，我们的眼睛通常对屏幕的左上角最敏感。可以将最重要的信息放在屏幕左上角。
- 充分利用空间关系。屏幕上的图形组件之间的距离不要太远，必要的时候可以用一个框将它们包围起来。
- 重视可读性和可理解性。系统应使用主动语气与用户交流想法。
- 即使能够在屏幕上添加很多种颜色，也要限制颜色的数量。颜色的使用要慎重。太多的色彩会分散用户对手头任务的注意力。另外，一个好的做法是让用户选择和修改颜色。
- 和使用颜色的限制一样，字体也不能滥用。要避免使用斜体或者过于华丽的字体。
- 组件和数据域应当左对齐——按照左边界将这些组件对齐。这样在用户浏览屏幕时可以减少眼球的移动范围。同时，也要尽量保持界面组件(例如按钮和列表框)的尺寸相同。
- 当用户阅读和处理信息并但按钮时，按钮应该放在信息框的右边并排成一列，或者放在信息框的有下放的一行中。这样的布局符合我们从左到右的阅读习惯。

并不是只有上述 10 个设计原则，但是这些设计原则是 GUI 设计的基本思想。GUI 设计的难题是在一种不复杂的、容易理解的和直观的可视化环境中向用户传达正确信息。

5.2 用 UML 设计合同管理信息系统的 GUI

为了开发出既设计合理又能充分满足用户需求的合同图形界面，我们不仅要描绘系统的静态机制，也要描绘系统的动态机制，这样才能便于与用户的交流。因此，我们利用可以展示系统静态机制的类图来描绘系统界面中的实体以及实体之间的关联；同时我们又利用可以展示系统动态机制的状态图来描绘一段时间内菜单或窗口中的对象所处的状态和状态变化。

下面就来介绍在描绘用户界面切换流程中使用的状态图和类图。

5.2.1 用于机动部合同管理信息系统 GUI 设计的状态图

以机动部合同管理信息系统登录窗口设计为例，介绍用于 GUI 设计的状态图。

为保证系统的安全，进入机动部管理信息系统之前，首先需要用户注册，即在主窗口中输入相应的用户名和密码进行身份验证。利用 UML 的状态图对此过程建模，如图 5.1 所示。

以下对用户注册过程的状态图说明如下：工作人员，如合同员、审核员等，单击安装在每台客户机上的应用程序 htgl.exe 的图标，即可进入验证用户名和密码的登录状态。

系统处于登录状态的同时，其内部也在经历着状态的变化。即登录状态中包含子状态(substate)。子状态以两种形式出现：顺序子状态(sequential substate)和并发子状态(concurrent substate)。这里，可以得到以下的状态序列：系统等待用户输入用户名和密码状态；系统登记用户输入信息(其中密码用秘文******字样显示)状态；系统对用户输入的信息进行验证状态（is validated)。这里的几个子状态显然是按顺序发生的状态，因而是顺序子状态。在验证用户输入信息状态时，如果用户名或密码输入错误，则系统进行提示，用户可重新输入信息，进入注册的自跃迁状态。状态图中能够表达出这种思想。UML 提供了表达这一历史状态的符号来表示当对象转移出该组成状态后，该组成状态能够记住它的活动子状态。如果用户注册 register()时出错不超过 3 次，则进入历史状态。用户可重新输入用户名和密码。

如果用户输入的信息连续出错三次，则系统认为有不安全因素存在，经确认后，拒绝该用户进入系统，而转入状态图的结束状态，即跳出应用程序。此时，登录状态结束。

如果用户输入信息正确，则系统转入打开系统主菜单状态（menu open）中。具有不同权限的用户，可打开不同的主菜单界面。随后，系统就转入执行业务工作状态中（business execute)。该状态由系统等待用户选择操作状态(用户可单击菜单项进行操作选择)（choosing operation)、系统执行操作状态（executing operation）和系统保存用户操作状态（saving the operation result）这三个顺序子状态组成。用户每完成一项操作，系统就进入等待用户再次选择操作状态。当用户完成所需的全部操作后，选择关闭或退出系统按钮，即可进入关闭注菜单窗口状态(close menu)。此时，系统执行退出主菜单操作，进入终止状态。

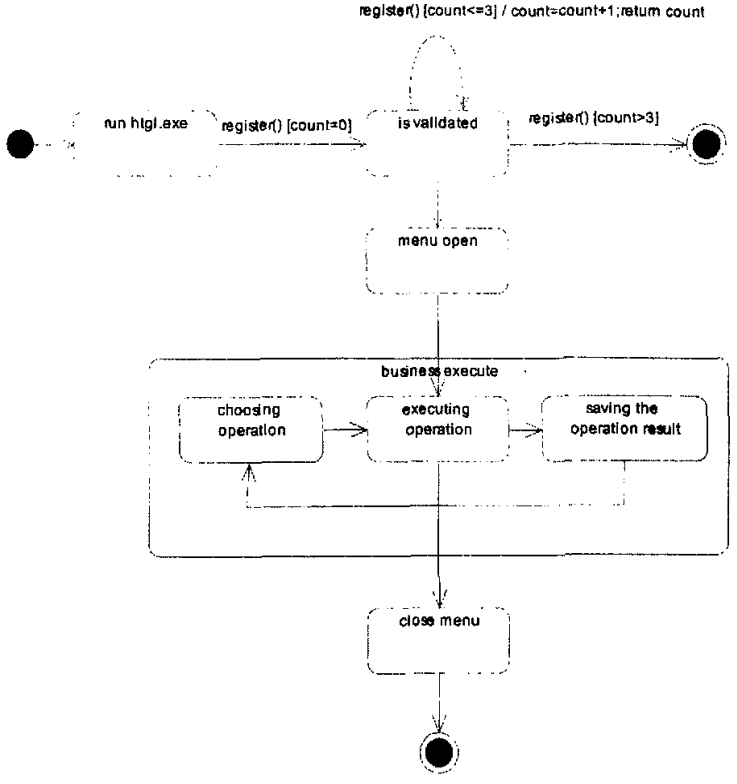

图 5.1　机动部合同管理系统 GUI 设计的状态图

Fig.5.1 The statechart diagram of the registered on GUI

5.2.2 定义用户界面类

用户与系统需要进行交互，一个用户友好的系统通常都采用直观的图形可视化界面，因此需要定义系统的用户界面类。通过对系统的不断分析和细化，可识别出下述界面类、类的操作和属性。下面说明主要的用户界面类：

(1) 类 MainWindow

MainWindow 是系统的主界面，系统的主界面具有菜单和菜单项，当选择不同的菜单项时，用户可以执行不同的操作。当程序退出时，主界面窗口关闭。

公共操作（Public Operations）：

createWindow() 创建管理系统的图形用户界面主窗口。

ContractInfo() 当选择"合同信息"菜单项时，该操作被调用。

DictionaryManage() 当选择"字典管理"菜单项时，该操作被调用。

QueryInfo() 当选择"查询"菜单项时，该操作被调用。

CountInfo()当选择"统计"菜单项时，该操作被调用。

DataMaintenance()　当选择"数据维护"菜单项时，该操作被调用。

Report()　当选择"打印报表"菜单项时，该操作被调用。

(2) 类 ContractMenu

界面类 ContractMenu 是进行操作"合同信息管理"、"闭口合同信息管理"、"预算信息管理"时所需要的菜单。

(3) 类 DictionaryMenu：字典管理菜单项。

(4) 类 QueryMenu：合同查询菜单项。

(5) 类 CountMenu：合同统计菜单项。

(6) 类 DataMaintMenu：数据维护菜单项。

(7) 类 LoginWindow

界面类 LoginWindow 是用来输入用户名和密码的对话框。

公共操作（Public Operations）：

createDialog():void　创建用来输入用户名和密码的对话框。

validate(id:String, name:String, password:String):Boolean 验证用户名和密码是否正确。

submit():void　当对话框被提交时，该方法被调用。

(8) 类 InWarehouseMenu：入库菜单项

(9) 类 BudgetMenu：预算信息菜单项

(10) 类 CorkContractMenu：开口合同菜单项

(11) 类 ContractStorageMenu：合同存储菜单项

(12) 类 DataMaintMenu：数据维护管理菜单项

(13) 类 JDBFileMenu：机动部文件菜单项

(14) 类 ContractFormMenu：合同报表管理菜单项

(15) 类 DictionaryMenu：字典管理菜单项

(16) 类 CheckDialog：信息核对对话框

(17) 类 MessageWindow

界面类 MessageWindow 是用来显示提示信息的窗口。

公共操作（Public Operations）：

creatWindow（msg：String）

创建窗口，显示提示信息。

(18) 类 FindReceiptDialog：查找单据对话框

界面类 FindReceipDialog 是用来根据单据（Receipt）的 id 信息查找的对话框。当主菜单中的菜单项"修改单据"或"审核单据"等操作被选择时，该对话框弹出，工作人员输入 Receipt 单据的 id 信息，单击按钮"OK"，系统查找数据库中具有指定 id 号的 Receipt 信息。

公共操作（Public Operations）：

createDialog() 创建用来填写 Receipt 单据 id 信息的对话框。

findId（）当对话框被提交时，该操作被调用。

(19) 类 FillReceiptDialog：填写单据对话框

(20) 类 InWarehouseDialog：入库单据对话框

界面类 InWarehouseDialog 是用来填写或显示合同单据信息的对话框，如图 5.2 所

示。

图 5.2 入库单据对话框

Fig.5.2 The dialogue box of filling receipt

当按下主窗口 InWarehouseMenu 中的按钮"入库单据"操作时，对话框弹出，单击提供的操作按钮，进行对应的操作。如单击"增加"按钮，工作人员填写单据信息（序号、委托编号、合同编号、合同内容、施工单位、合同金额、资金来源、开工时间、竣工时间、合同类型等），系统创建单据并将之存储在系统中。

公共操作（Public Operations）：

newRDialog():void　创建用于填写单据信息的窗口。

newRDialog(receipt:Receipt)：void 创建用于显示单据信息的窗口。

newReceipt（）：void　按钮"增加"被按下时，该方法被调用。

delReceipt（）：void　按钮"删除"被按下时，该方法被调用。

modReceipt（）：void　按钮"修改"被按下时，该方法被调用。

firstReceipt（）：void　按钮"第一个"被按下时，该方法被调用。

previousR（）：void　按钮"上一个"被按下时，该方法被调用。

nextR（）：void　按钮"下一个"被按下时，该方法被调用。

lastR（）：void　按钮"最后一个"被按下时，该方法被调用。

browR（）：void　按钮"浏览"被按下时，该方法被调用。

(21) 类 ReturnDialog：返回对话框。

因为一个特定的屏幕是由许多组件组成，说明类之间的组成关系的类图也适用于对屏幕界面建立模型。如图 5.3 所示的类图，描述了用于主菜单 GUI 设计的类之间的关系。主要关系说明如下：类 MainWindow 是系统的主界面，系统的主界面具有菜单和菜单项。当选择不同的菜单项时，用户可以执行不同的操作。当程序退出时，主界面窗口关闭。类 LoginWindow 是用户注册窗口界面。类 MainWindow 与类 LoginWindow 之间是关联关系。而类 InWarehouseMenu、BudgetMenu、CorkContractMenu、ContractStorageMenu、DataMaintMenu、JDBFileMenu、ContractFormMenu 和 DictionaryMenu、类 QueryMenu、类 CountMenu 分别表示入库菜单项、预算信息菜单项、开口合同菜单项、合同存储菜单项、数据维护管理菜单项、机动部文件菜单项、合同报表管理菜单项、字典管理菜单项、查询菜单项、统计菜单项。这些菜单项都是 MainWindow 的一部分，它们与类 MainWindow 具有一致的生命周期，因此它们与类 MainWindow 之间是聚合关系。

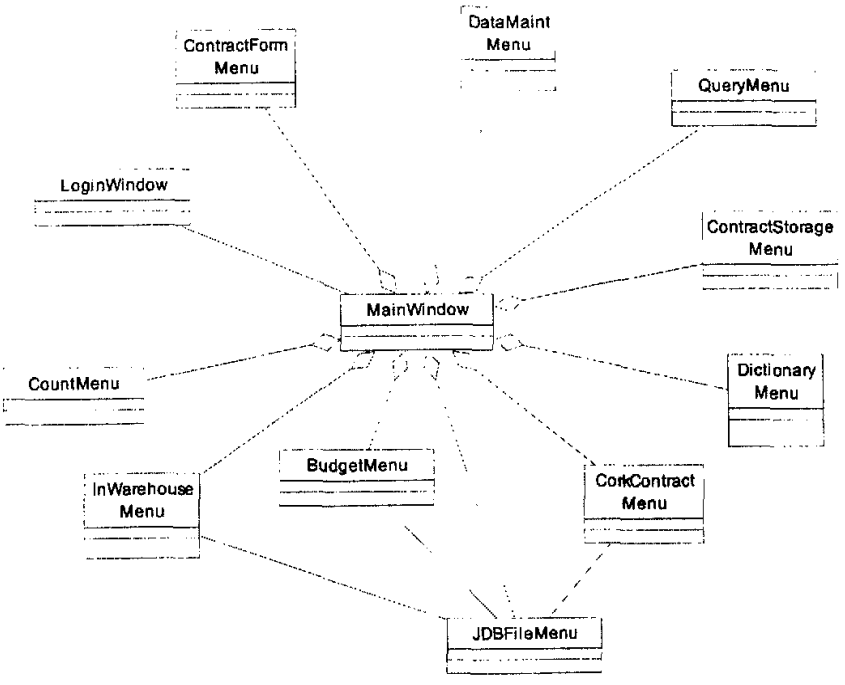

图 5.3 用户界面类的类图

Fig.5.3 The class diagram of main menu

另外，类 CheckDialog、MessageWindow、FindReceiptDialog、FillReciptDialog、InWarehouseDialog 和 ReturnDialog 都是菜单项 InWarehouseMenu 的一部分，因此它们与类 InWarehouseMenu 之间也是聚合关系。如图 5.4 所表示。

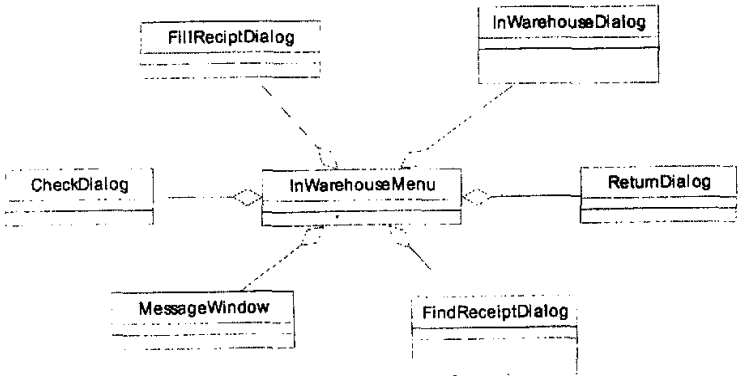

图 5.4　GUI 类图

Fig.5.4 The class diagram of main menu's GUI

同理，图中其它类之间也存在着聚合关系，在此不再赘述。

使用以上表示系统 GUI 的类图进行建模后，首先交给用户审阅，然后与用户不断探讨和修改 GUI 布局及设计，直到用户满意为止。只有这样才能真正设计出合理的、友好的、满足用户需求的图形用户界面。

5.3 用 Deployment 图建模系统的网络部署

在多数系统中，硬件是一个重要方面。现在的计算领域，一个系统可能要包括多种的操作平台，并且要跨越很长的物理距离。一个坚实的系统硬件部署图对系统设计来说是必需的。UML 提供了一组图符，用于创建一幅图来描述最终系统的硬件设置。

5.3.1 确定网络类型

UML 的部署图(Deployment)用于描述系统的物理部署，如网络布局和组件在网络上的位置的问题。该类图中包含处理器、设备、进程和处理器与设备之间的连接。每个系统只有一个物理部署图。

在描绘系统的部署图之前，首先要根据系统的实际，确定所需的网络节点和网络类型。其次，根据节点之间连接情况，描绘出系统的部署图。

5.3.2 构建本系统局域网

节点是运行时存在的物理单元，它代表了具有内存以及处理能力的各种计算资源。节点是为系统物理方面建模的重要的模型元素。通常，节点被用于为运行系统的硬件拓扑结构建模。节点代表了运行组建的处理器或设备。

节点有两种类型：处理器(Processor) 和设备(Device)。处理器是可以执行程序的硬件组件，设备是没有计算能力的硬件组件。设备通常都具有某种形式的与外部世界的接口。

在我们开发的合同管理信息系统中，机动部合同管理信息系统与分厂合同管理信息系统实现的功能不是完全相同，决定了机动部与分厂的网络结构也各不相同。以下对合同管理信息系统的网络类型、网络需求和远程网的连接等进行分析和比较。

(1) 合同网络设计分析

合同管理信息系统由机动部和分布在北京市各区的其他若干个分厂组成。由于该系统采用双数据库设计，即机动部数据库和分厂数据库各自独立操作的方式，因而决定了机动部和分厂的网络结构和规模也不相同。机动部管理信息系统规模较大，可由各部门等构成一个协调工作的局域网。分厂管理信息一般规模较小，有的分厂只需一台同时具备服务器和客户端功能的 PC 机即可。

(2) 基于以太网构建本系统的局域网

经过对局域网传播方式、拓扑结构和性能的比较，结合合同管理系统的实际，本系统局域网采用以下的网络方案：

- 局域网采用星型快速以太网结构，传输速率 100Mbps，采用高速智能交换式集线器进行连接。该结构的性价比是最适合该公司的实际的。

- 为了保证系统的安全，机动部可采用一台主服务器与一台备份服务器的二机容错模式对网络进行管理，并在主服务器上建立双硬盘镜像。
- 采用大型关系数据库产品作为后台数据库，以客户机/服务器模式对网络数据库进行管理。

以上方案采用成熟的网络技术和数据库产品，所用产品均遵循国际公认的正式标准，扩展性很好，可与不同的系统交换信息，另外可靠性、安全性等方面也有足够保障，除用于业务管理外，还可实现企业内部的电子邮件传送和提供其它网络服务。

(3) 远程连接

计算机网络中所使用的远程通信技术主要有：公用电话交换网(Public Switched Telephone Network，PSTN)、数字数据网(DDN)、综合业务数字网 ISDN(Integrated Service Digital Network)、非对称数字用户线 ADSL(Asymmetric Digital Subscriber Line)、帧中继 FR(Frame Relay)、X.25 等等。

合同管理系统不需要时时进行网络传输，并且网络传输的数据量不大。结合这一实际，我们选用了公用电话网 PSTN 或 Internet 作为机动部与分厂之间进行数据传送的物理信道。

机动部与分厂之间实现数据上传和下载的过程如下：

当机动部和分厂之间需要进行数据交换时，分厂通过 PSTN 以拨号方式，向机动部发起连接请求。机动部接受请求后，即可在机动部和分厂之间建立物理连接信道。接通后，分厂首先向机动部上传合同信息和审核信息，然后机动部向分厂下载数据库更新信息等。这种访问方式只要求计算机通过 Modem 连到电话线上，对机动部和分厂的要求都比较简单，也可利用 Internet 接入。当连接建立之后，通信双方存在一条实际的物理信道，可以进行点对点的通信，能保证固定的传输带宽。当双方通信结束后，只需断开连接即可。

这种方式对该系统特别适用。因为该系统中的各分厂一般相距较远，因而无法保证在同一局域网内。机动部和分厂之间不必实时连接，只当分厂与机动部之间需要传合同相关信息时，才通过 PSTN 建立点对点的连接。这种设计方式不仅十分方便，而且花费较少。

(4) 系统的运行环境及设备分布

该系统分为机动部局域网和各分厂局域网(其中，规模较小的分厂只有一台 PC 机，并不构成局域网)。机动部局域网需要如下的网络设备：一台主服务器、一台备份服务器、若干台二级交换机和若干智能 Hub 及 Modem，若干台客户机。对于规模较大的分厂局域网，所需网络设备可与机动部基本相同。对于只有一台 PC 机的分店，可选择性能较高的 PC 机，在该 PC 机上安装为分厂开发的软件。

5.3.3 描绘系统的 Deployment 图

基于以上分析，合同网络系统由机动部局域网和若干分厂局域网组成，其部署情况，如图 5.5 所示。

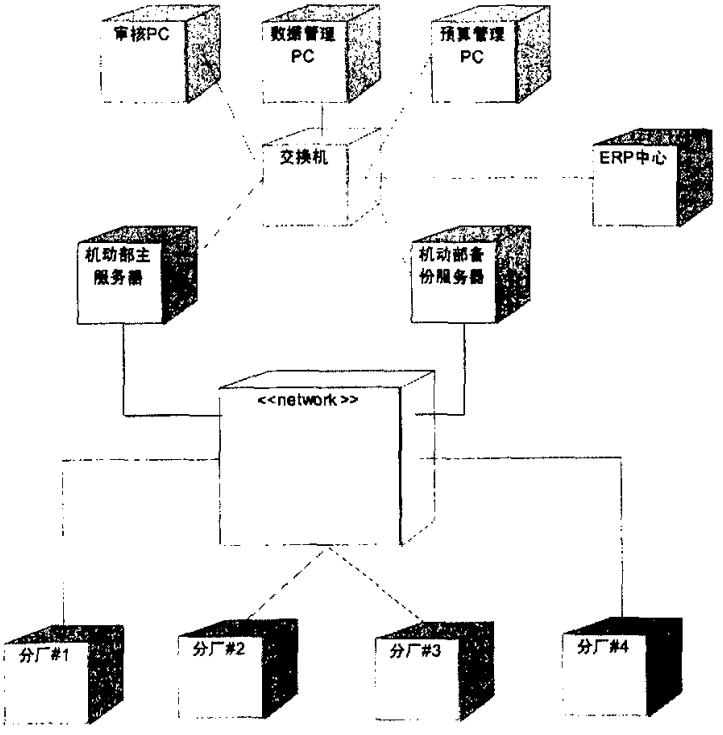

图 5.5 系统部署图

Fig.5.5 the deployment diagram of the system

第六章 合同管理系统中的 UML 模型的前向工程与逆向工程

UML 不是可视化的编程语言，但它的模型可以直接对应到各种各样的编程语言，这个过程就称为前向工程（Forward Engineering），从代码实现生成 UML 模型的过程被称为逆向工程（Reverse Engineering）。目前许多 CASE 工具都支持前向工程又支持逆向工程。Rational Rose 同样支持 UML 模型与 Java 等源代码之间的相互转换。

6.1 Java 的代码生成

在 Rational Rose 中，从 UML 模型生成 Java 代码的步骤如下：

(1) 将 Java 类分配给模型中的 Java 组件。

Rational Rose 用组件来描述 Java 文件，所以，为了成功地产生 Java 代码，需要将模型中的类指定给模型组件视中的 Java 组件。将多个类指定给一个组件，在生成代码时就可以生成一个.java 文件，这个文件包含指定给组件的多个类。

将 Java 类指定给模型中的 Java 组件有两种方法，一种是采用拖放的方法，另一种方法是采用组件的规范定义。

(2) 检查语法。

这个步骤是可选的。可以在生成代码前选择检查模型组件的语法，但实际上，语法检查会在生成代码时自动进行。语法检查是基于 Java 语言的语义。

(3) 检查类路径。

Java Project Specification 中的类路径制表页可以用来为模型定制特点的 Java 类路径（Classpath）。当进行前向工程从模型生成代码或进行逆向工程从代码生成模型时，Rational Rose 需要使用这个类路径信息解析文件位置和类库引用。

当需要动态地添加一些目录以扩充类路径时，可以使用 Java Project Speicification 中 Class Path 制表页的 Directoried 部分。在 Directories 部分添加的任何设置都和当前模型一起保存。

(4) 设置影响代码生成的项目属性（可选）。

选择 Tools：Java：Project Specification 菜单项，在 Project Speicification 窗口中选择 Detail 制表页或 Style 制表页，设置影响代码生成的项目属性。

(5) 备份源代码。

在为已存在的.java 文件生成代码时，Rational Rose 会为当前的源文件产生扩展名为".~jav"的备份。但是，如果要进行多次前向和逆向工程，一定要注意备份，因为如果对模型多次产生代码，Rational Rose 创建的".~jav"备份文件就不再是最初的源代码。

(6) 从模型生成 Java 源代码。

在 Rational Rose 中，利用菜单项 Tools:Java:Generate Java，选择组件图中的一个或

多个包和组件，既可以从组件图生成 Java 源代码,也可以从类图生成 Java 源代码。

(7) 查看并编辑所产生的源代码。

在生成代码后，通常希望浏览产生的代码。用户可以在编辑器中修改产生的源代码，为了使模型与源代码保持一致，需要更新模型。更新模型，需要使用逆向工程，将.java 文件的变化反馈到模型中。

6.2 逆向工程

逆向工程是通过分析 Java 源代码创建或更新模型的过程。在逆向工程过程中，Rational Rose 分析.java 文件和.class 文件，将文件中的类和对象添加到模型中。

在 Rational Rose 中，从 Java 代码生成 UML 模型的逆向工程的步骤如下：

(1) 如果是更新一个已经存在的模型，打开模型。

(2) 选择 Tools:Java:Reverse Engineer Java 菜单项。

(3) 从目录结构，选择包含要进行逆向工程的文件的目录。

(4) 设置 Filter 域，显示需要进行逆向工程的 Java 文件的类型（.java 或.class 文件）。

(5) 采用下述方式之一将所选类型的 Java 文件添加到所选文件列表中。

- 选择一个或多个文件。
- 选择所有的文件。

(6) 确认需要进行逆向工程的文件。

(7) 单击按钮 Reverse，从所选的 Java 文件生成模型或更新已有的模型。如果逆向过程中有错误发生，会有错误对话框弹出。

(8) 检查发生的错误列表。

逆向工程产生的类或组件可以通过 Rational Rose 的浏览器浏览，Rational Rose 不能自动创建基于逆向工程产生的类的类图或组件图，将类或组件添加到图中，有两种办法：

- 将类或组件从浏览器中拖放到新的或已存在的图中。
- 使用 Query:Add Classes 或 Query：Add Components 菜单项。

通过这些方法，就能产生类图或组件图。

第七章 结束语

本文对统一建模语言(UML)的主要特点、现状与未来及相关知识等进行了研究，并阐述了使用统一建模语言进行合同信息管理信息系统建模的具体实现过程。

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。在 UML 中汇入了面向对象领域中很多人的思想。这些思想是开发者们依据最优秀的 OO 方法和丰富的计算机科学实践经验综合提炼而成的。

在此课题的开发过程中，首先分析了合同信息管理信息系统的需求，确定了系统的体系结构，并选择了适合本系统的大型关系数据库产品 SQL Server 2000 作为后台数据库，同时选择 JAVA 作为前端开发工具。其次，分析了机动部合同管理和分厂合同管理信息系统实现的功能。最后，应用 UML 分别对机动部和分厂管理信息系统进行建模。

在课题实施过程中，应用 UML 进行系统建模，使我受益匪浅。以下是我在建模过程中的体会：

(1) 需求分析阶段，建立了用于描述系统实现功能的用例图。用例图的建立是我们与用户反复讨论的结果，它描述了所开发系统的功能需求。用例图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是否满足系统需求。也就是说，建立系统用例图后，在系统开发后期，如发现所建系统用例图没有对应的程序代码时，则可判断出该用例图的功能尚未实现，这样就能提示开发人员更改模型或实现该用例的功能。

(2) 利用顺序图和协作图为系统的动态行为建模时，虽然顺序图用于描述消息的时间顺序，协作图用于描述对象间的关系，但由于二者是语义等价的图形，因此，我们可利用 Rational Rose 建模工具方便地实现二者的同步转换。这就使得用户既可以通过前者按时间顺序查看信息流，又可以通过后者查看对象间的关系和对象间传递的消息，非常直观、清晰。

(3) 系统在开发过程中，利用 JAVA 实现系统编程。应用 UML 建模语言及 Rational Rose 建模工具，实现前向工程与逆向工程，便于软件和对象的一致与重用。信息技术项目的一个挑战就是保持对象模型与代码的一致。

综上所述，将 UML 应用于管理信息系统的开发中，不仅是可行的，而且使我们收获颇丰。统一建模语言 UML 代表了面向对象软件开发技术的发展方向，正引导着当今软件工业界的潮流，具有巨大的市场前景和重大的经济价值。

参考文献

1. Mark Priestley, Practical Object-Oriented Design with UML, McGraw-Hill Companies,Inc., 2002

2. [美]Paul R.Reed, Jr.著 郭旭译，JAVA 与 UML 协同应用开发，北京：清华大学出版社，2003 年 9 月

3. [美]刘润东，UML 对象设计与编程，北京希望电子出版社，2001 年 5 月

4. 张龙祥，UML 与系统分析设计，北京：人民邮电出版社，2001 年 8 月

5. [美]Jim Conallen 著 陈起 英宇译，用 UML 构建 Web 应用（第二版），2003 年 11 月

6. [美]Jason T.Roff 著 张瑜 杨继萍等译，UML 基础教程，北京：清华大学出版社，2003 年 10 月

7. 冀振燕 编著. UML 系统分析设计与应用案例[M]，北京：人民邮电出版社，2003 年 6 月

8. Douglas Scherer, William Gaynor, Jr., Arlene Valentinsen. Xerxes Cursetjee 著. 京京工作室译. Oracle8i 数据库开发技术与技巧[M]，北京：机械工业出版社，2002.6

9. Booch G, Object-Oriented Analysis and Design With Applications[M]. RedWood City, California: Benjamin/Cummings Publishing Company, 1994

10. Jacobson I, Object-Oriented Software Engineering[M], A Use Case Driven Approach, NewYork: Addison-Wesley Publishing Company, 1992

11. Eric J. Naiburg. Robert A.Maksimchuk 著，陈立军，郭旭 译，UML 数据库设计与应用[M]，北京：人民邮电出版社，2002 年 11 月

12. Joseph Schmuller 著，李虎. 王美英. 万里威等译，UML 基础、案例与应用[M]，北京：人民邮电出版社，2002 年 10 月，126-129

13. Wendy Boggs, Michael Boggs 著，邱仲潘等译，UML 与 Rational Rose 2002 从入门到精通[M]，北京：电子工业出版社，2002 年 7 月

14. 邵维中. 梅宏. 统一建模语言 UML 述评[J]，计算机研究与发展，1999 年 4 月，第 36 卷第 4 期，385-394

15. 王云等. 标准建模语言 UML 简介[J]，计算机应用研究，1999 年，第 12 期，44-49

16. 萨师煊. 王珊 著.数据库系统概论[M]，北京：高等教育出版社，2000 年 2 月第三版

17. 王瑞金. 段会川. Martin Gogolla. 统一建模语言 UML 及其建模实例[J]，计算机应用研究，2002 年，第 8 期，80-84

18. 郁磊. 窦延平. 何厚存. 统一建模语言在小型信息系统设计中的应用[J]，计算机工程，2002 年 7 月，Vol.28，272-275

19. 黄贤英. UML 建模过程及在需求分析中的应用[J]，计算机工程，2001 年 11 月，Vol.27，184-186

20. 王建军. UML 建模实例分析[J]，微计算机信息，2002 年，Vol.18，66-68

21. 张肖朝. 王愚. UML 与 MVC 设计模式在社区信息系统中的应用与实现[J], 计算机应用，2003 年 6 月，第 23 卷增刊

22. He Ke-qing. Extended UML with Role Modeling[J], Wuhan University Journal of Natural Sciences, 2001,vol 6, 175-182

23. 统一建模语言 UML 概述，http://hongying.8u8.com/lis/rjg1_6.htm

24. 软件模型设计基础，http://www.51CMM.COM.com

25. 统一建模语言 UML 的静态建模机制，http://www.51CMM.COM.com

26. 标准建模语言 UML 概述，http://programminglife.8u8.com/rjgc/rjgc_uml.htm

27. UML 概述，http://modalhouse.myrice.com/index.htm

28. 标准建模语言 UML 概述，http://www.umlchina.com

致 谢

在该论文的整个工作过程中，我得到了许多人无私的帮助和支持。

首先特别感谢我的导师高福祥教授。我的论文从选题、设计直至今天的提交，无一不是高教授悉心指导和帮助的结果。在我的人生旅途中，能成为高教授的学生，并得到高教授无微不至的关怀和帮助，是我今生最大的财富！在整个论文撰写期间，高教授悉心传授给我们许多新知识、新思想和新技术。他宽广的胸襟、无私的品德以及严谨的学风，是我一生学习的榜样！在此，我要再次对我的恩师表示崇高的敬意和诚挚的感谢!!

我还要由衷地感谢我的学友何美丽，她乐观的态度以及高超的水平，不断激励着我。几年里，我们一起学习、互相帮助，共同进步。

感谢首钢高新公司高工夏秀玲，感谢她为我提供了参加实际项目开发的机会，从她身上我学到许多现场工作的经验和专业的工作态度。

非常荣幸能成为东北大学的一员，感谢东北大学的培育之恩！感谢所有曾关心、帮助过我的人们！