西安电子科技大学软件学院

**软件工程硕士学位论文**

写作模板

应用软件开发类

西安电子科技大学软件学院

鲍 亮（修订）

2015年04月

**00000000**

**10701**

**学号**

**代号**

**2015年5月**

**提交论文日期**

**工程领域**

**作者姓名**

**based on Node**

**The Design and implementation of Pos trading system**

**ine**

**line**

**基于Node的Pos机交易系统的设计与实现**

**题（中、英文）目**

`

**软件工程**

**杨涛**

**曾浩 高工**

**李青山 教授**

**学校指导教师姓名职称**

**企业指导教师姓名职称**

**编号**

**UDC**

**公开**

**TP000**



**密级**

**分类号**

**西安电子科技大学**

**学位论文创新性声明**

秉承学校严谨的学分和优良的科学道德，本人声明所呈交的论文是我个人在导师指导下进行的研究工作及取得的研究成果。尽我所知，除了文中特别加以标注和致谢中所罗列的内容以外，论文中不包含其他人已经发表或撰写过的研究成果；也不包含为获得西安电子科技大学或其它教育机构的学位或证书而使用过的材料。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做了明确的说明并表示了谢意。

申请学位论文与资料若有不实之处，本人承担一切的法律责任。

本人签名： 杨涛 日期 2015/5/18

**西安电子科技大学**

**关于论文使用授权的说明**

本人完全了解西安电子科技大学有关保留和使用学位论文的规定，即：研究生在校攻读学位期间论文工作的知识产权单位属西安电子科技大学。学校有权保留送交论文的复印件，允许查阅和借阅论文；学校可以公布论文的全部或部分内容，可以允许采用影印、缩印或其它复制手段保存论文。同时本人保证，毕业后结合学位论文研究课题再攥写的文章一律署名单位为西安电子科技大学。

（保密的论文在解密后遵守此规定）

本学位论文属于保密，在 年解密后适用本授权书。

本人签名： 日期

导师签名： 日期

摘 要

建立一个通用电子商务交易平台是加速电子商务领域应用开发与集成的重要途径。POS机交易系统是电子商务交易平台集成框架的关键组成部分，为电子商务工作提供电子交易功能，并提高电子商务交易系统的运行性能。目前已有的POS机交易系统在通用性和扩展性方面还不能满足平台的需要。

本文在分析了系统功能需求的基础上，设计并实现了一个高性能的基于Node的POS机交易系统。通过用例和逻辑等视图分析了系统的设计架构，采用Express为系统框架，使系统具备更好的可拓展性。系统框架采用Ejs模板，使系统具备明确，维护性良好的HTML代码结构。系统采用MongoDB为后台数据库，使系统具备良好的数据存取性能。最后，部署了一个POS机交易系统应用环境。实验结果表明，该POS机交易系统的可行性和有效性达到了设计要求。

关键字：交易系统 Node Express Embedded JavaScrip MongoDB

**Abstract**

Establish a common e-commerce trading platform is an important way to accelerate the development and integration of e-commerce. POS trading system is a key part of the integrated frame of the e-commerce trading platform, providing electronic trading function for e-commerce, and improve the operating performance of e-commerce trading system. The existing POS machine trading trading system can not meet the needs of the platform in general and scalability.

This paper is on the basis of analyzing the functional requirement of the system, design and implement a high performance POS machine trading system based on Node. By analyzing the architecture of the system with use case and logic, the system use the Express as the system frame, which make a better scalability for the system. And the system take the Ejs as the view template engine, which make the system clear, maintain good HTML code structure. And the system uses MongoDB as the background database, which make the system has good data access performance. Finally, the application environment of a POS transaction system is deployed. The experimental results show that the feasibility and effectiveness of the POS system based on Node can meet the design requirements.

**Keyword: Trading system, Nodejs, Express, Embedded JavaScrip, MongoDB**

目 录

[第一章 绪论 1](#_Toc339447714)

[1.1 选题背景及意义 1](#_Toc339447715)

[1.2 国内外现状分析 1](#_Toc339447716)

[1.3 论文工作内容 1](#_Toc339447717)

[1.4 论文组织结构 1](#_Toc339447718)

[第二章 相关技术概述 3](#_Toc339447719)

[2.1 ×××理论 3](#_Toc339447720)

[2.2 ×××技术 3](#_Toc339447721)

[2.3 ×××技术 3](#_Toc339447722)

[2.4 本章小结 3](#_Toc339447723)

[第三章 ××业务需求分析 5](#_Toc339447724)

[3.1 ×××业务陈述 5](#_Toc339447725)

[3.2 ×××需求建模 5](#_Toc339447726)

[3.3 ×××数据建模/×××数据分析 6](#_Toc339447727)

[3.4 ×××过程建模 7](#_Toc339447728)

[3.5 本章小结 7](#_Toc339447729)

[第四章 ××系统设计与实现 9](#_Toc339447730)

[4.1 ×××系统应用架构 9](#_Toc339447731)

[4.2 ×××系统功能设计 10](#_Toc339447732)

[4.2.1 ×××软件组件 10](#_Toc339447733)

[4.2.2 ×××硬件选型 10](#_Toc339447734)

[4.3 ×××系统数据库设计/×××数据结构设计 10](#_Toc339447735)

[4.4 ×××系统功能实现 11](#_Toc339447736)

[4.4.1 ×××组件实现 11](#_Toc339447737)

[4.4.2 ×××系统集成 13](#_Toc339447738)

[4.5 本章小结 13](#_Toc339447739)

[第五章 ××系统测试及分析 15](#_Toc339447740)

[5.1 系统运行环境 15](#_Toc339447741)

[5.2 测试用例及过程 15](#_Toc339447742)

[5.3 测试结果分析 16](#_Toc339447743)

[5.4 本章小结 17](#_Toc339447744)

[第六章 结束语 19](#_Toc339447745)

[6.1 论文工作总结 19](#_Toc339447746)

[6.2 后续工作展望 19](#_Toc339447747)

[致 谢 21](#_Toc339447748)

[参考文献 23](#_Toc339447749)

[附录一：UML图使用指南 25](#_Toc339447750)

[附1.1 UML2.0图示类型 25](#_Toc339447751)

[附1.2 UML2.0图使用方法 26](#_Toc339447752)

[附1.3 UML2.0图绘制方法 27](#_Toc339447753)

[附1.3.1 活动图 27](#_Toc339447754)

[附1.3.2 用例图 28](#_Toc339447755)

[附1.3.3 类/对象图 30](#_Toc339447756)

[附1.3.4 顺序图 32](#_Toc339447757)

[附1.3.5 通信图 33](#_Toc339447758)

[附1.3.6 状态图 34](#_Toc339447759)

[附1.3.7 组件图 35](#_Toc339447760)

[附1.3.8 部署图 36](#_Toc339447761)

第一章 绪论

学位论文主要是向读者介绍工作的背景、现状、工作内容和过程。绪论部分通过介绍为什么做？别人做了什么？我要做什么？准备如何做？四个方面引出论文的主体内容。

1.1 选题背景及意义

POS(point of Sales)交易系统最初定义为即时点销售管理系统或前台系统。该系统有收款机，条形码扫描仪，数据显示器，票据打印机，电子秤，磁场阅读器，封闭式钱箱，信息记录与传送设备等硬件设备和相应的软件构成，是通过收银员在营业场所内的收银机上的实际操作，将商品卖给顾客，实现即时销售管理的过程。

POS系统通常是一个前，后台相关联的系统，它将企业前台电子收银机有后台计算机通过网络连接起来，收银员将销售终端采集到的商品号件数，交易时刻等信息数据通过特定的方式传送给后台，并用后台的计算机处理完整的商品销售数据，将结果存储到服务器数据库中，便于企业管理者掌握当时，当日当月等各种商品销售信息，为决策提供依据。POS系统还可以处理非直接销售业务内容，如前台商品退货，优惠销售以及盘点等，从而构成一整套以销售业务为主，其他业务处理为辅的前台系统。

然而随着互联网IT技术的发展以及电子商务的兴起与壮大，POS机交易系统已经从传统的实体商场，商店的简单收款机，逐渐发展成为电子商务系统交易环节的不可或缺的重要组成部分。

而基于智能手机和平板电脑的移动POS，使POS硬件市场发生巨大的变化。以打补丁方式逐步升级的老一代POS正在被新一代开放，全功能POS平台所替代。而随着零售商与消费者各自需求的提升，现在的POS机已经不仅仅是商家为消费者提供服务的销售点或者服务点，它已经拓展延伸到消费者手中的搜索点，社交点以及购物点。基于互联网技术的电子商务为商家与消费者开辟了一个现实世界意外的虚拟零售世界，移动技术则讲着两个世界整合在一起，为商家与消费者建立了一个1+1>2的多维零售世界。

进来随着云计算等科技的发展，POS机也往云POS方向发展。而Nodejs与MongoDB对于构建云应用有着语言与性能的优势。

实现基于Node的以MongoDB为后台数据库的POS机交易系统，将会为电子商务，移动支付以及云POS交易系统带来良好的稳定性，更加便捷良好的可拓展性， 以及更加快速处理庞大复杂的交易数据的性能，为社会带来巨大收益。

本文以Nodejs为平台，Express为系统框架，EJS为系统框架视图模板引擎，并且以MongoDB为后台数据库，实现了基于Node的POS机交易系统。

1.2 国内外现状分析

1.2.1 国外发展现状：

POS机最先发展于国外，第一代机采用总线方式联网，具有简单收款机功能，客观上不需要强大的计算机信息管理；随着功能需求的增加，出现了第二，第三代POS。前面的一二三代POS机，由大厂商把控，售价高昂，一般比较大型的商场才负担得起。随着互联网技术的发展，出现了第四代POS机。

而随着移动技术的发展，出现了移动POS机。欧美许多大型零售企业2013年已经开始大量部署移动POS，这些移动POS包括基于智能手机，平板电脑以及各种移动装置的具有传统POS机功能的POS、店员助理、导购机以及各种门店与消费者互动的接触点。大型零售商虽然大量部署移动POS，但目前还是采用固定POS与移动POS混合使用的系统。而小型零售点与餐饮店等则趋向于逐步全部采用移动POS。

而在移动POS软件市场，零售企业成了市场的主宰者，自行开发的，包括委托第三家开发采用自家品牌的专用POS移动程序成了这个市场的主流。

BYOD式CPOS受到商家与消费者青睐开始进入实用阶段  
　　在零售业IT应用中，最近出现了一种称为自带设备（BYOD）的使用移动设备的工作模式。简单得说，自带设备方式就是将用户自己的手机或平板电脑装上企业免费提供的移动软件，变成一台具有某些特殊功能的移动终端。前面提到过的个人购物助手、Walmart的Scan&Go，都是属于这一类。如果用户自己的移动设备装上某家公司提供的POS软件，那么这个设备就可以在该公司作为POS机使用。这类按BYOD方式工作的POS称为CPOS（Customer POS），即客户POS。  
　　这种新生的CPOS对零售商与顾客都有一定的吸引力。零售商可以省一笔购买移动POS的费用，顾客可用自己手上的设备自助完成某些原来需要店员协助的POS操作，获得更惬意的购物体验。然而，在具体实现中也会遇到种种困难，虽然业内人士对这类CPOS普遍看好，同时也可以听到异议的声音。对于任何一种新兴技术，这种现象是十分正常的。

1.2.2 国内发展现状：

与国外相比，国内POS机起步较晚，早期的高端POS机几乎都是进口于外国厂商， 价格昂贵，因此只有大型商场才负担得起，而小型卖场都使用国产的具有简单记账功能的收款机。上世纪九十年代末，随着PC硬件的迅速发展，出现了组装POS机。而随着国产POS品牌机的出现，国产POS机在价格方面的优势以及在性能于技术上与进口POS机相差不大，因此国产POS机已经逐渐占据国内中低端的市场，并向高端发起了攻势。

近年来随着阿里巴巴，京东等电子商务网站的发展壮大以及移动技术的蓬勃发展，基于智能手机与平板电脑的移动POS，已经使POS硬件市场发生了巨大的变化。而全新的开放的，全功能的POS平台将逐渐代替老旧的POS机。移动POS发展动向移动技术正在改变零售业和餐饮业的面貌，现状的POS机已经不仅仅是商家为消费者提供服务的销售点或者服务点，它已经拓展延伸到消费者手中的搜索点，社交点，已经购物点。移动POS的发展使得商品交易的主控从商家转移到消费者手中，让消费者更加主动，更有发言权。基于互联网技术的电子商务为商家与消费者开辟了一个现实世界以外的虚拟零售世界。移动技术则将这两个世界整合在一起,为商家与消费者建立了一个1+1﹥2的多维零售世界。

进来随着云计算等科技的发展，POS机交易系统也往云POS方向发展，2011~2012年是云POS从概念到使用的发展时期。这段时期内不少公司都在市场推出云POS产品，一些小型零售商与餐饮店也开始采用云POS，来扩充或者接替传统POS。虽然云POS还处于刚刚发展的初期，云POS产品水平与应用水平都还不高，但随着技术条件的实现，以及日益完善的发展生态环境的建立，未来云POS将会展现出良好的发展前景！

1.3 论文工作内容

本文主要研究基于Node的POS机交易系统的设计与实现，在整个过程中主要完成以下工作：

1. 基于Node的POS机交易系统的整体设计。本系统采用Nodejs为开发平台语言，采用Ejs为视图模版引擎，MongoDB为后台数据库，采用Express为系统框架，该框架有较高的可拓展性。
2. 根据需求分析，设计并实现系统的注册功能。
3. 根据需求分析，设计并实现登录功能。
4. 根据需求分析，设计并实现退出功能。
5. 根据需求分析，设计并实现添加商品功能。
6. 根据需求分析，设计并实现搜索商品功能。
7. 根据需求分析，设计并实现商品页功能。
8. 根据需求分析，设计并实现添加商品到购物车功能。
9. 根据需求分析，设计并实现在购物车增减，删除商品模块。
10. 根据需求分析，设计并实现用户添加收货人地址，姓名，联系方式功能。
11. 根据需求分析，设计并实现提交订单功能。
12. 根据需求分析，设计并实现查看订单功能。
13. 部署一个分布式仿真环境，验证XX系统的有效性。

1.4 论文组织结构

论文共分为六章，各章主要内容如下：

第一章：绪论。提出论文选题背景及意义、国内外研究现状分析、论文主要的工作内容和组织结构。

第二章：相关技术概述。列举并简要描述了项目开发过程中涉及到的相关技术，其中包括Nodejs技术、Express框架技术、EJS视图模板引擎技术、MongoDB数据库技术等。

第三章：基于Node的POS机交易系统的业务需求分析。在介绍系统业务数据和过程的基础上，完成系统应用需求建模、数据建模和过程建模。

第四章：基于Node的POS机交易系统的设计与实现。设计系统的应用架构，按照功能分解结构细化系统功能，设计软件组件，设计系统数据库。按照组件设计结构进行开发，并完成系统集成。

第五章：基于Node的POS机交易系统测试及分析。阐述系统运行和部署环境，采用测试用例介绍系统测试过程，针对测试结果进行分析给出结论。

第六章：结束语。总结了本文的主要工作，指出工作的不足及进一步的改进方向。

1.5 本章小节

本章提出了论文的选题背景以及意义、国内外研究现状分析、论文主要的工作内容和组织结构。

第二章 相关技术概述

2.1 Nodejs技术

Node.js于2009年由开发者Ryan Danl在github上发布初始版本的node.js包，后来获得云计算服务商Joyent资助，2011年在微软支持下发布了windows版本，目前已经支持Linux，Mac，windows三大平台。

Node.js是一个基于Chrome JavaScript运行时建立的平台，用于方便地搭建响应速度快，易于拓展的网络应用。Node.js使用事件驱动，非阻塞I/O模型而得以轻量和高效，非常适合在分布式设备上运行的数据密集型的实时应用。

Nodejs有以下优点：（1）RESTful API；（2）单线程：Node.js可以在不新增额外线程的情况下，依然可以对任务进行并行处理 —— Node.js是单线程的。它通过事件轮询（event loop）来实现并行操作，对此，我们应该要充分利用这一点 —— 尽可能的避免阻塞操作，取而代之，多使用非阻塞操作；（3）非阻塞I/O；（4）V8虚拟机；（5）事件驱动。

Node是一个JavaScript运行环境（runtime）。实际上它是对Google V8 引擎进行了封装。Node对一些特殊用例进行了优化，提供了替代的API，使得V8在非浏览器环境下运行得更好。

V8引擎本身使用了一些最新的编译技术。这使得JavaScript这类脚本语言编写出来的代码运行速度获得了极大提升却节省了开发成本。对性能的苛求是Node的一个关键因素。JavaScript是一个事件驱动语言，Node使用了这个优点，编写出可拓展性高的服务器。Node采用了一个成为“事件循环（event loop）”的架构，使得编写可拓展性高的服务器变得既容易又安全。提高服务器性能的技巧是多种多样的。Node选择了一种既能提高性能，又能减低开发复杂度的架构。这是一个非常重要的特性。并发编程通常是很复杂而且地雷。Node绕过了这些，但仍然提供很好的性能。

Node采用一系列“非阻塞”库来支持事件循环的方式。本质上就是为文件系统、数据库之类的资源提供接口。向文件系统发送一个请求时，无需等待硬盘（[寻址](http://baike.baidu.com/view/1303626.htm)并检索文件）的结果，当硬盘准备好的时候非阻塞接口会通知Node。该模型以可扩展的方式简化了对慢资源的访问， 直观，易懂，高效。尤其是对于熟悉[onmouseover](http://baike.baidu.com/view/3587213.htm)、onclick等[DOM](http://baike.baidu.com/subview/14806/8904138.htm" \t "_blank)事件的用户，更有一种似曾相识的感觉。

虽然让JavaScript运行于服务器端不是Node的独特之处，但却是其一强大功能。[浏览器](http://baike.baidu.com/view/7718.htm" \t "_blank)环境限制了我们选择编程语言的自由。任何服务器与日益复杂的[浏览器](http://baike.baidu.com/view/7718.htm)客户端应用程序间共享代码的愿望只能通过JavaScript来实现。虽然还存在其他一些支持JavaScript在服务器端 运行的平台，但因为上述特性，Node发展迅猛，成为事实上的平台。

Node.js使用Module模块去划分不同的功能，以简化应用的开发。Modules模块有点象C语言中的类库。每一个Node.js的类库都包含了十分丰富的各类函数，比如http模块就包含了和http功能相关的很多函数，可以帮助开发者很容易地对比如http,tcp/udp等进行操作，还可以很容易的创建http和tcp/udp的服务器。

要在程序中使用模块是十分方便的，只需要如下：

在这里，引入了http类库，并且对http类库的引用存放在http变量中了。这个时候，node.js会在我们应用中搜索是否存在node\_modules的目录，并且搜索这个目录中是否存在http的模块。如果node.js找不到这个目录，则会到全局模块缓存中去寻找，用户可以通过相对或者绝对路径，指定模块的位置，比如：

var myModule = require('./myModule.js');

模块中包含了很多功能代码片断，在模块中的代码大部分都是私有的，意思是在模块中定义的函数方法和变量，都只能在同一个模块中被调用。当然，可以将某些方法和变量暴露到模块外，这个时候可以使用exports对象去实现。

在Node启动的很短时间内，社区就已经贡献了大量的扩展库（模块）。其中很多是连接数据库或是其他软件的驱动，但还有很多是凭他们的实力制作出来的非常有用的软件。

虽然Node项目从2009年才开始发展，还非常年轻，但很少看到对一个项目如此狂热的社区。不管是新手，还是专家，大家都围绕着项目，使用并贡献自己的能力，致力于打造一个探索、支持、分享、听取建议的乐土。

Node.js发展迅猛，目前在不断完善中，版本已经发布至0.12.3.

Node.js集成开发环境：具备书写JavaScript的IDE均可，普通的记事本也可以进行开发。

2.2 Express技术

Express是一个简洁而灵活的Node.js Web应用框架，提供一系列强大特性帮助开发者创建各种Web和移动设备应用。它是目前最稳定，使用最广泛，而且是Node.js官方推荐的唯一一个Web开发框架。

Express有丰富的HTTP快捷方法和任意排列组合的connect中间件，让开发者创建健壮，友好的API变得既快速又简单。

Express并不对Node.js已有的特性进行二次抽象，而是在其基础上拓展Web应用所需的基本功能。

2.3 Ejs技术

Ejs是Express框架的一种模板引擎，它将页面模板和要显示的数据集合起来从而生成HTML。在MVC架构中，模板引擎包含在服务器端。控制器得到用户的请求后，从模型获取数据，调用模板引擎。模板引擎以数据和页面模板为输入，生成HTML页面，然后返回给控制器，由控制器交回客户端。

Ejs只有三种标签：

（1）<% code %>：其中的code是被执行的JavaScript代码；

（2）<% =code %>：显示替换过的HTML特殊字符的内容；

（3）<% -code %>：显示原始HTML的内容。

<%= code %>和<%- code %>的区别在于，当变量code为普通字符串时，两者没有区别；而当code比如为<h1>hello</h1>这种字符串时，<%= code %>会原样输出<h1>hello</h1>，而<%- code %>则会显示H1大的hello字符串。

下面是ejs的官方示例：

**The Data:**

Supplies: [‘mop’, ’broom’, ‘duster’ ]

**The Template:**

<ul>

<% for(var i=0; i<supplies.length; i++) {%>

<li><%= supplies[i] %></li>

<% } %>

</ul>

**The Result:**

<ul>

<li>mop</li>

<li>broom</li>

<li>duster</li>

</ul>

我们可以用上述三种标签实现页面模板系统能实现的任何内容。

2.4 MongoDB技术

MongoDB 是一个基于分布式文件存储的 NoSQL（非关系型数据库）的一种，由 C++ 语言编写，旨在为 WEB 应用提供可扩展的高性能数据存储解决方案。MongoDB 支持的数据结构非常松散，是类似 json 的 bjson 格式，因此可以存储比较复杂的数据类型。MongoDB 最大的特点是他支持的查询语言非常强大，其语法有点类似于面向对象的查询语言，几乎可以实现类似关系数据库单表查询的绝大部分功能，而且还支持对数据建立索引。

MongoDB 没有关系型数据库中行和表的概念，不过有类似的文档和集合的概念。文档是 MongoDB 最基本的单位，所谓文档就是一个关联数组式的对象，每个文档都会以唯一的 \_id 标识，文档的属性为 key/value 的键值对形式，一个值可以是一个数，字符串，日期，数组，甚至可以嵌套另一个文档，因此可以存储比较复杂的数据类型。集合是许多文档的总和，一个数据库可以有多个集合，一个集合可以有多个文档。

MongoDB轻巧，灵活，最大的优势在于所有的数据持久操作都无需开发人员手动编写SQL语句，直接调用方法就可以轻松的实现CRUD操作。

2.5 本章小结

本章概述了项目的相关技术，其中包括 Node.js技术，Express技术，EJS技术，MongoDB技术。

第三章 基于Node的POS机交易系统需求分析

系统分析是研究系统能做什么，是软件系统设计的前提，在系统开发中决定着项目的成败。系统分析的过程是对系统所需要的功能、质量和约束进行准确描述的过程，即需求分析。需求分析通常包括业务过程陈述及问题分析、需求建模、数据建模和过程建模等。

3.1 系统总体分析

本部分主要对系统进行总体分析，主要做两件事情，确定系统的参与者和系统的边界。

1. 系统的参与者是指那些与系统交互的人、时间、其他系统等角色，可以采用文字的形式对系统参与者的名称和主要任务进行描述。

系统的参与者：注册用户，未注册用户

注册用户任务：登录系统，搜索商品，浏览商品，提交订单，查询订单

未注册用户任务：浏览商品，搜索商品，注册请求

2. 系统的边界采用**上**下文数据流图（Context data flow diagram）来表示，上下文数据流图是用来记录系统的范围的过程模型，也称为环境模型。

记录系统边界的方法：

（1）将系统看作“黑盒子”（圆角矩形），放在中间。

（2）询问最终用户系统需要响应什么业务事务，这些业务事务就是系统的净输入（箭头），而来源则为外部代理（矩形）。

（3）询问最终用户系统必须产生什么响应，这些响应就是净输出（箭头），而目的地也将成为外部代理（矩形）。

（4）确定外部数据存储（外部代理的一种，矩形）。

（5）绘制上下文数据流图。

图3.1本系统的上下文数据流图如下：



图3.1系统上下文数据流图

3.2 系统业务陈述

本部分主要描述待系统的主要业务过程及分析可能存在的问题。采用**活动图**（Activity Diagram）+文字形式进行描述。下面是一个活动图的样例：



图3.2系统总体活动图

描述如下：用户进行注册，登录，然后也可以浏览商品，搜索商品，点击商品之后进入商品页面，在商品页面有“加入购物车”按钮，登录的用户点击之后商品加入购物车，在导航条点击购物车链接之后进入用户购物车页面，在购物车页面，用户可以对在购物车的商品进行数量增减或者删除商品，购物车下方的收货人地址，用户点击“添加收货地址”后，填写收货人地址，收货姓名，收货人电话，然后点击“提交”按钮后，生成新的收获人信息，用户选收货人信息后，点击“提交订单”按钮，系统生成订单，用户在导航条点击“我的订单”按钮后，进入用户订单页面，页面按时间由近到远列表显示用户的订单信息，登录的用户随时都可以退出，未注册的用户随时都可以注册，未登录的用户随时都可以登录，未登录的用户可以浏览商品，搜索商品，但不能进入购物车页面和订单页面。

3.3 基于Node的POS机交易系统需求建模

对业务过程的内容进行分析，抽取出业务功能、质量属性和约束。业务功能可采用**用例图**（Use case Diagram）+文字形式进行描述，质量属性和约束可采用文字进行描述。用例图构建过程：

1. 按照角色识别并定义系统参与者；
2. 按照功能识别并定义系统用例；
3. 分析参与者、用例的复用性；
4. 将用例分配到参与者；
5. 采用用例表格描述用例图。

本部分首先给出系统整体的功能可以采用总体用例图的方式进行呈现

|  |
| --- |
| 图3.3系统总体用例图 |

以下是详细用例描述：

表3.1 用户添加商品用例详细描述

|  |  |
| --- | --- |
| 用例ID | 103 |
| 用例名称 | 添加商品到购物车 |
| 用例简介 | 用户往购物车中添加库存量大于0的商品，商品将在购物车中呈现。 |
| 参与者 | 用户 |
| 前置条件 | 商品数量不为0 |
| 基本事件流 | 1. 用户登录 2. 搜索商品 3. 浏览商品 4. 选择商品 5. 添加商品到购物车 |
| 其他事件流 | 在购物车中可更改商品数量，移除商品等 |
| 异常事件流 | 1． 库存量不足，则重新选择其他商品  2． 若没有用户搜索的商品，则显示没有相关商品 |
| 后置条件 | 购物车更新 |
| 扩展点 | 暂无。 |

表3.2 用户添加选择收货信息用例详细描述

|  |  |
| --- | --- |
| 用例ID | 105 |
| 用例名称 | 添加选择收货人信息 |
| 用例简介 | 若还没有收货信息，则添加；若已经有，则选择收货信息 |
| 参与者 | 用户 |
| 前置条件 | 用户已经登录 |
| 基本事件流 | 1. 点击添加地址  2. 填写收货人地址，姓名，电话等  3. 提交地址  4. 生成新收货信息 |
| 其他事件流 | 若存在多个收货地址，则只能选择一个 |
| 异常事件流 | 若没有选择地址，则不能提交订单 |
| 后置条件 | 收货人信息栏更新 |
| 扩展点 | 暂无。 |

表3.3 用户查询订单用例详细描述

|  |  |
| --- | --- |
| 用例ID | 107 |
| 用例名称 | 用户查询订单 |
| 用例简介 | 用户查询已经生成的订单 |
| 参与者 | 用户 |
| 前置条件 | 用户已经登录，订单已经生成 |
| 基本事件流 | 1. 用户登录  2. 进入订单列表页面  3. 查询订单（按订单生成的时间倒序列出） |
| 其他事件流 | 用户登录，订单生成 |
| 异常事件流 | 1. 新用户，没有历史订单  2. 没有要查询的订单 |
| 后置条件 | 显示订单信息列表 |
| 扩展点 | 暂无。 |

3.4 基于Node的POS机交易系统数据建模

对业务过程进一步细化，按照数据建模过程分析数据实体及关系。通常可采用**实体联系图**（Entity Relation Diagram）+文字进行描述。实体联系图构建过程：

1. 构造上下文数据模型：包括实体（包括实体属性）和联系，确定系统范围；
2. 用文字描述实体联系图：用户属于用户库，N对1的关系，用户有编号，姓名，地址，电话等属性，用户有库编号属性：商品属于商品库，N对1的关系，商品有商品编号，商品名称，商品价格等属性，商品库有库编号属性，用户购买商品，N对N的关系，有购买时间和数量等属性。

购买

用户库

属于

用户

1

N

商品库

属于

商品

1

N

图3.4基于Node的POS机交易系统的实体联系图

3.5 本章小结

本章概述了系统总体分析，给出了上下文数据流图；概述了系统业务分析，给出了系统活动图并进行了描述；概述了系统需求建模，给出了系统总体用例图以及各个用例详细描述；概述了系统数据建模，给出了系统实体联系图并进行了描述。

第四章 基于Node的POS机交易系统设计与实现

系统设计是在系统分析的基础上研究系统如何实现需求中所描述功能。系统设计的过程包括系统应用架构设计、功能模块设计、数据库设计、界面和接口设计等。系统实现的过程是按照模块功能分别采用类图、序列图、接口、算法、程序流程图等方式描述实现方法和过程。

4.1 基于Node的POS机交易系统应用架构

系统应用架构设计用来描述系统的骨架和轮廓。集中式系统可以采用**层次模型图**描述；分布式系统可采用**网络结构图**描述。

系统为集中式系统，用户注册，登录，添加收货人信息，删除收货人信息，提交订单，添加商品到购物车，在购物车中增减，删除商品等操作向MongoDB数据库发送数据；

而用户登录后，以及用户搜索商品，查看购物车，选择收货人信息，查看订单等操作需要从MongoDB中请求数据。

|  |
| --- |
| 图4.1 基于Node的POS机交易系统的层次模型图 |

4.2 基于Node的POS机交易系统功能设计

基于Node的机交易系统包含：用户模块，商品模块，购物车模块，收货信息模块，订单模块，其中：

1. 用户模块包含：用户注册，用户登录，用户退出；

（2）商品模块包含：浏览商品，搜索商品，商品单页；

（3）购物车模块包含：商品加入购物车，增减商品，删除商品；

（4）收货信息模块包含：添加收货人信息，删除收货人信息，选择收货人信息；

（5）订单模块包含：提交订单，查看订单。

用户模块接口为USER，商品模块接口为COMMODITY，购物车模块接口为CART，收货信息模块接口为ADDRESS，订单模块接口为ORDER。

系统功能模块分解图如下：

|  |
| --- |
| 图4.2 基于Node的POS机交易系统功能模块分解图 |

4.3 基于Node的POS机交易系统数据库设计

数据库设计是在概念数据建模的基础上根据特定数据库系统构建系统逻辑数据模型。通常可采用**逻辑数据模型**+文字进行描述。物理数据模型构建过程：

1. 每个实体创建一张表；
2. 每个属性创建一个字段；
3. 为主键创建索引；
4. 为关系指定外键；
5. 确定每个属性的数据类型、字段大小、空或非空、域及默认值；
6. 用文字描述逻辑数据模型。

系统数据库逻辑数据模型图如下：

|  |
| --- |
| 图4.3 系统逻辑数据模型图 |

逻辑数据模型图说明：

(1)用户对象有用户ID，用户名，用户密码，用户Email四个属性，其中用户ID为其主键；

(2)购物车对象有购物车ID，商品名称，商品价格，商品数量，用户ID五个属性，其中购物车ID为其主键，用户ID为其外键；

(3)商品对象有商品ID，商品名称，商品价格，商品商家，商品图片等属性，商品ID为其主键；

(4)收货人信息对象有信息ID，收货人地址，收货人姓名，收货人电话等属性，信息ID为其主键；

(5)订单对象有订单ID，用户对象，收货人信息对象，购物车对象等属性，信息ID为其主键。

系统根据需求分析以及逻辑数据模型图，设计了五个数据库，分别是用户表，商品表，收货人信息表，购物车表，订单表：

1. 用户表如下：

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Name** | **Comment** | **Default** | **DataType** | **P** | **F** | **Null** |
| userID | 用户ID |  | INTEGER(30) | √ |  |  |
| userName | 用户名 |  | VARCHAR(16) |  |  |  |
| userPassword | 用户密码 |  | VARCHAR (6) |  |  |  |
| Tel | 电话 |  | VARCHAR (20) |  |  |  |
| Address | 地址 |  | VARCHAR (100) |  |  |  |
| Email | 邮箱 |  | VARCHAR (20) |  |  | √ |

1. 商品表如下：

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Name** | **Comment** | **Default** | **DataType** | **P** | **F** | **Null** |
| c.\_id | 商品ID |  | INTEGER(30) | √ |  |  |
| cname | 商品名称 |  | VARCHAR(16) |  |  |  |
| cimage | 商品图片 |  |  |  |  |  |
| cprice | 商品价格 |  | INTEGER |  |  |  |
| name | 商品商家名称 |  | INTEGER |  | √ |  |

1. 收货人信息表：

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Name** | **Comment** | **Default** | **DataType** | **P** | **F** | **Null** |
| address.\_id | 信息ID |  | INTEGER(30) | √ |  |  |
| IP | 收货人地址 |  | VARCHAR(16) |  |  |  |
| who | 收货人姓名 |  | VARCHAR (6) |  |  |  |
| phone | 收货人电话 |  | VARCHAR (20) |  |  |  |

1. 购物车信息表：

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Name** | **Comment** | **Default** | **DataType** | **P** | **F** | **Null** |
| cart.\_id | 购物车ID |  | VARCHAR(30) | √ |  |  |
| cname | 商品名称 |  | VARCHAR(16) |  |  |  |
| cprice | 商品价格 |  | FLOAT (6) |  |  |  |
| camount | 商品数量 |  | INTEGER (20) |  |  |  |
| user | 用户对象 |  | USER |  |  | √ |

1. 订单表如下：

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Name** | **Comment** | **Default** | **DataType** | **P** | **F** | **Null** |
| order.\_id | 订单号 |  | INTEGER(30) | √ |  |  |
| user | 用户对象 |  | USER |  | √ |  |
| address | 收货人信息对象 |  | ADDRESS |  |  | √ |
| cart | 购物车对象 |  | Cart |  |  |  |

4.4 基于Node的POS机交易系统功能实现

### 4.4.1 系统用户模块功能实现

用户点击系统导航条的注册或者登录链接后，系统返回注册或者登录的ejs页面，用户填写注册或者登录用的用户个人信息并点击“提交”按钮，用户提交后，ejs页面将数据发送到MongoDB数据库，如果用户是注册，则数据库查询有没有该用户，如果有，则注册失败，提示用户重新注册；如果没有，则数据库生成一个新的用户对象实例，并保存到数据库，然后返回登录后的ejs主页，此时用户注册成功；如果用户是登录，则数据库查询有没有给用户，如果有，则返回登录后的ejs主页，如果数据库没有该用户，则系统提示没有该用户，并转到注册ejs页面。

序列图如图4.4。

|  |
| --- |
| 图4.4 用户注册、登录序列图 |

### 4.4.2 系统商品模块功能实现

用户登录后，系统返回主页，主页列表显示商品。

用户在搜索框输入要搜索的商品的关键字并点击“搜索”按钮后，搜索框ejs页面将关键字发送到MongoDB数据库进行查询，如果数据库有与关键字相关的商品，则返回商品，系统转到搜索结果页面；如果没有，则返回空白页。

用户点击某个商品后，进入商品单页ejs页面，显示单个商品的信息，商品下方有“加入购物车”按钮，用户点击“加入购物车”按钮后，ejs将请求传送到MongoDB数据库，数据库生成并保存新的购物车对象实例，然后购物车对象实例在购物车页面按时间由近到远显示，此时用户成功将商品添加到购物车。

序列图如图4.5。

|  |
| --- |
| 图4.5 搜索、浏览商品，添加商品到购物车序列图 |

### 4.4.3 系统购物车模块功能实现

登录用户点击系统导航条的“购物车”链接后，购物车ejs页面按时间由近到远依次显示用户已经添加到购物车而且未提交到订单的商品的数据，用户可以对商品进行数量的增减或者删除商品，当增减或者删除商品后，购物车ejs页面将数据传送到数据库，数据库更新页面购物车对象信息，并将结果返回到购物车ejs页面。

序列图如图4.6。

|  |
| --- |
| 图4.6 购物车，收货人，订单序列图 |

### 4.4.4 系统收货人信息模块功能实现

当登陆的用户在购物车ejs页面下方点击“新添地址”按钮后，系统弹出三个文本框，提示用户分别填写收货人地址，收货人姓名，收货人电话，当填写完毕后，点击“提交”按钮，ejs页面将数据发送到MongoDB数据库，数据库生成并保存一个新的收货人信息对象实例，然后对象实例显示在收货人信息栏。

当用户点击已有收货人信息的“删除”按钮后，ejs页面将操作请求发送到MongoDB数据库，数据库更新收货人信息对象数据并返回数据给ejs页面。

序列图如图4.6。

### 4.4.5 系统订单模块功能实现

当登录用户在购物车ejs页面已经确定购物车商品的信息，并且已经选择收货人信息后，点击页面“提交订单”按钮，ejs页面将数据发送到MongoDB，MongoDB生成并保存一个新的订单对象实例，订单对象实例在订单ejs页面按时间由近到远列表显示，购物车ejs页面将已经提交订单的商品信息清除，MongoDB数据库信息更新。

用户点击系统导航条“我的订单”链接后，系统转到订单ejs页面，页面按时间由近到远列表显示已经生成的订单的信息；如果用户还没有提交过订单，则页面提示“订单为空”。

序列图如图4.6。

4.5 本章小结

第五章 ××系统测试及分析

系统测试是在仿真或真实运行环境中，采用测试用例测试系统需求分析中所定义的功能和质量属性。系统测试的内容包括测试环境、测试用例、测试过程和测试结果分析等。

5.1 系统运行环境

系统测试平台需要真实或者模拟的运行环境支持。集中式系统通过软硬件平台描述；分布式系统在软硬件平台的基础上，还需要描述其物理分布和网络环境。通常可采用部署图+文字描述网络环境。

|  |
| --- |
| 图5.1 ×××系统部署图 |

5.2 测试用例及过程

**测试用例**是测试的大纲，通常可通过需求分析用例经过转化到得到。测试过程按照测试活动的先后顺序进行描述，通常分步骤采用文字进行描述。（切忌描述测试的概念和方法）

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| （1）小波包提取有效信号测试过程  这两部分在频域中距离较远，有利于使用小波包对信号进行分解和重构。本文采用Daubechies小波系列的DB10小波进行3层小波包变换仿真分析，用代表原始信号，代表小波包分解的第1层低频系数，代表小波包分解的第1层高频系数，其他以此类推，由于保持了多分辨分析中的正交分解特性，每一个节点分解后的两个频带互不交叠，输出两个频带的带宽减半，假设在原始信号中，最低频率成分为1，最高频率成分为0，则第3层中每个节点代表的频带范围就是0.125[18]。需要注意的是，对于第3层按照自然数增加顺序排列的，其小波包分解的频率分布不是严格按照自然数的递增排列的。为了保持主要频率随i的增加单调递增的性质，需要对小波包分解后的系数重新排序，以来表示。表3.3是用db10小波3层分解后按频率重新排序的情况。  表5.1各频率成分所代表的频率范围   |  |  |  |  | | --- | --- | --- | --- | | 信号 | 频率范围 | 信号 | 频率范围 | |  | 0~0.125 |  | 0.500~0.625 | |  | 0.125~0.250 |  | 0.625~0.750 | |  | 0.250~0.375 |  | 0.750~0.875 | |  | 0.375~0.500 |  | 0.875~1.000 |   小波包分解结构树中第3层各结点所代表的频率范围为2.5kHz。结点(3,1)，(3,2)，(3,3)，(3,4)所代表的频率范围为2.5kHz~12.5kHz，与振动信号频率成分的频率范围相一致。因此，对结点(3,1)，(3,2)，(3,3)，(3,4)重构原信号可去除低频和高频干扰，更加真实的反映振动过程。如图3.13所示：  图5.2小波包分解滤波后的振动波形 |

5.3 测试结果分析

功能测试的结果能满足功能需求即可，通常可以用系统运行时截图进行描述；对于系统质量属性的测试结果需要进行分析，通常采用**柱状图**、**折线图**或**数据表**的形式描述测试结果。对于质量属性测试的结果进行分析、讨论，并给出能够满足系统质量要求的结论。（切忌只给出运行时截图，数据和图表必须进行分析并给出结论）

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 本文进行100组对比测试，得到98组有效数据，其中有2次测试出现明显错误，在剔除这两错误数据后得到如表4.5所示的分合闸起始运动时间实测数据与从动触头位移信号（行程信号）中得到分合闸起始运动时间之间平均值误差对比的测试结果：  表4.5大量实测数据与行程信号对比分析   |  |  |  |  | | --- | --- | --- | --- | | 98次测试 | 实测数据 | 行程信号 | 平均误差 | | 合闸起始运动时间/ms | 23.54 | 24.14 | 2.49% | | 分闸起始运动时间/ms | 19.63 | 20.08 | 2.24% |   分析以上数据可以得出如下结论：在100次测试的有效数据中，所有测试数据的相对误差均不大于3%，这在在线监测状态下是完全可以接受的。而在这100次测试期间发生2次明显的数据错误，检查监测系统并分析对应的振动信号发现，产生错误的原因是由于多次试验后振动传感器和永磁铁之间连接的松动造成的，实际使用时注意检查固定情况能有效减少这种错误的产生。 |

5.4 本章小结

第六章 结束语

结束语是对论文工作的总结。在给出论文工作结论的基础上，分条目介绍论文主要完成的工作，并对系统进行客观评价和分析，提出后续改进的建议。

6.1 论文工作总结

按照总分式介绍论文所完成的工作。先给出论文工作的结论，然后再按照工作内容分条给出所完成的具体工作。（切忌和摘要重复）

6.2 后续工作展望

针对系统中存在的问题，分析后续工作内容和改进方案。（切忌存在问题过多，一般2个为宜）

致 谢

对在项目实施、论文工作和学习过程中给予指导和帮助的人表示感谢。一般感谢的对象包括导师、同学朋友、家人等，对被感谢者不要直书其名，而应该冠以敬称，如某某教授、某某博士等学术头衔。（切忌天马行空，姓名和单位切勿写错，语言尽量认真、诚恳和严谨）

参考文献

【期刊论文】

[序号] 作者1, 作者2, 作者3, 等. 论文名称[J]. 期刊名称, 年,卷(期):页码-页码.

[1] 宋胜利, 鲍亮, 陈平. 多层文本分类性能评价方法[J]. 系统工程与电子技术, 2010, 32(5): 216-222.

【会议论文】

[序号] 作者1, 作者2. 论文名称[C]. 会议名称. 出版地:出版社.年份. 页码-页码.

[2] Shengli Song, Xiaofei Qiao, Ping Chen. Hierarchical Text Classification Incremental Learning[C]. Proceedings of the 16th International Conference on Neural Information Processing. Heidelberg: Springer-Verlag. 2009. 247-258.

【学位论文】

[序号] 作者. 论文名称[D]. 所在地:学校, 年份.

[3] 宋胜利. 文本语义表示及多层分类关键技术研究[D]. 西安:西安电子科技大学, 2011.

【普通图书】

[序号] 作者. 书籍名称[M]. 版本. 出版地:出版社, 年份. 页码-页码.

[4] 陈平, 禇华. 软件设计师教程[M]. 第二版. 北京:清华大学出版社, 2006. 1-24.

【电子文献】

[序号] 单位. 名称[/OL]. 年份. 网址.

[5] 中国互联网络信息中心. 2012年中国网民搜索行为研究报告[[R/OL]. 2012. <http://www.cnnic.cn/gywm/xwzx/rdxw/2012nrd/201208/t20120806_32995.htm>.

附录一：UML图使用指南

UML是一种面向对象的建模语言，它是运用统一的、标准化的标记和定义实现对软件系统进行面向对象的描述和建模。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。它融入了软件工程领域的新思想、新方法和新技术。它的作用域不限于支持面向对象的分析与设计，还支持从需求分析开始的软件开发的全过程。

附1.1 UML2.0图示类型

UML2.0所包含的图示：

* 类图 (Class Diagram)
* 组件图(Component diagram)
* 复合结构图(Composite structure diagram)
* 部署图(Deployment diagram)
* 对象图(Object diagram)
* 包图(Package diagram)
* 行为图(Behavior diagrams）
* 活动图(Activity diagram)
* 状态图(State diagram)
* 用例图(Use Case Diagram)
* 交互图(Interaction diagrams）
* 通信图(Communication diagram)
* 交互概述图(Interaction overview diagram) (UML 2.0)
* 顺序图(Sequence diagram)
* 时间图(UML Timing Diagram) (UML 2.0)

在学位论文写作中，常见的有9类图示，其作用包括：

* 用例图：用例图是从用户角度描述系统功能， 是用户所能观察到的系统功能的模型图，用例是系统中的一个功能单元。
* 类图：类图描述系统中类的静态结构。不仅定义系统中的类，表示类之间的联系如关联、依赖、聚合等，也包括类的内部结构(类的属性和操作)。
* 对象图：对象图是类图的实例，几乎使用与类图完全相同的标识。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。
* 顺序图：顺序图显示对象之间的动态合作关系，它强调对象之间消息发送的顺序，同时显示对象之间的交互，顺序图的一个用途是用来表示用例中的行为顺序。当执行一个用例行为时，顺序图中的每条消息对应了一个类操作或引起状态转换的触发事件。
* 通信图：通信图描述对象间的交互关系，显示对象间的动态合作关系。除显示信息交换外，通信图还显示对象以及它们之间的关系。
* 状态图：状态图是一个类对象所可能经历的所有历程的模型图。状态图由对象的各个状态和连接这些状态的转换组成。
* 活动图：活动图是状态图的一个变体，用来描述执行算法的工作流程中涉及的活动。活动图描述了一组顺序的或并发的活动。
* 组件图：组件图为系统的构件建模型—组件即构造应用的软件单元—还包括各构件之间的依赖关系，以便通过这些依赖关系来估计对系统组件的修改给系统可能带来的影响。
* 部署图：部署视图描述位于节点实例上的运行构件实例的安排。节点是一组运行资源，如计算机、设备或存储器。这个视图允许评估分配结果和资源分配。

附1.2 UML2.0图使用方法

UML是一种面向对象分析与设计的重要工具，常见的图示类型与面向对象分析与设计的对应关系如下表所示：

附表1.1UML图示与面向对象分析与设计对应关系

|  |  |
| --- | --- |
| **OOSAD各阶段** | **图示类型** |
| **需求陈述** | 活动图、系统用例图、业务用例图 |
| **静态分析** | 类图、对象图 |
| **动态分析** | 顺序图、通信图、状态图 |
| **物理设计** | 组件图、部署图 |

附1.3 UML2.0图绘制方法

### 附1.3.1 活动图

* 基本作用

活动图是用来描述一系列顺序动作、结果及其它们之间关系的图，主要用来表示系统控制流程和业务处理流程，它重点关注业务过程中的动作和结果。在软件开发中，活动图主要用来和用户交流，以辅助需求采集。有时候，可以把活动图看成一种特殊的状态图。活动可以绑定到任何建模元素，用来反映该元素的行为，该元素提供该活动的语境，活动绑定的元素有;用例、类、接口、组件、协作、操作，但在早期建模时不需要明确是哪个元素。活动图在使用时要注意目标用户，力求简单。活动图元素包括活动、转移、分支、分叉、合并、汇合、泳道、对象流。活动图的典型应用包括：用例中控制流和用例间的控制流，操作和算法的细节以及业务流程。

* 建模过程

（1）在采集的原始需求中选择重点流程。

（2） 首先要确定要设计的活动图是针对业务流程还是用例。

（3） 其次要设计活动过程的起点和终点。

（4） 确定活动图所有执行对象。

（5） 确定活动节点，并根据执行对象进行活动分组。

（a） 如果对用例建活动图，则把角色所发出的每一个动作变为活动节点。

（b） 如果对业务流程建活动图，则把每一个流程步骤(或片段)变为活动节点。

（6） 确定活动节点之间转移。

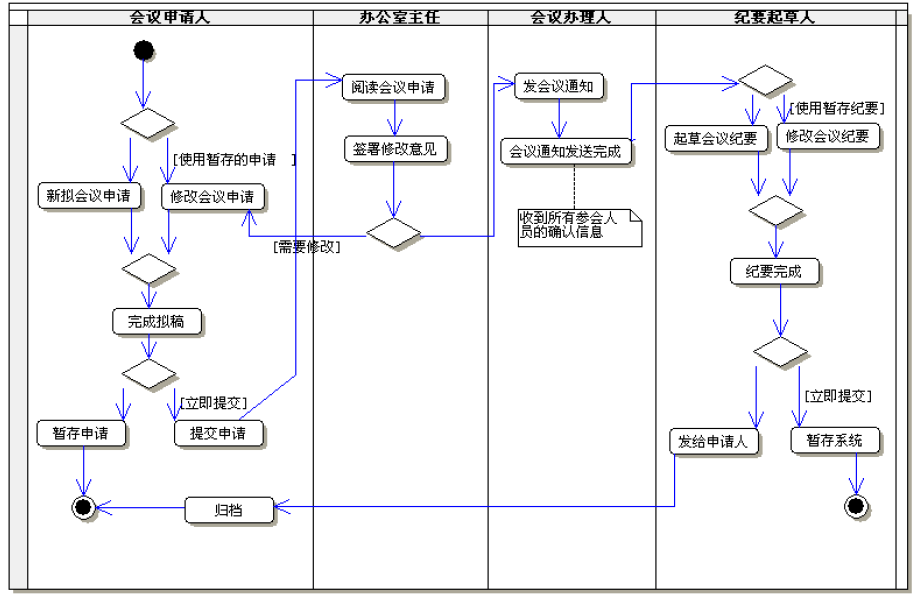
（7） 处理在活动节点之间的分支和合并。

（8） 处理在活动节点之间的分叉和汇合

（9） 用UML 建模工具进行活动图建模。

（10） 编写必要的补充文档。

* 绘图示例



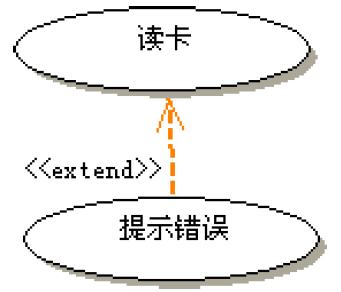
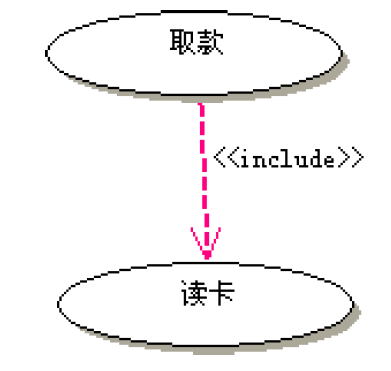
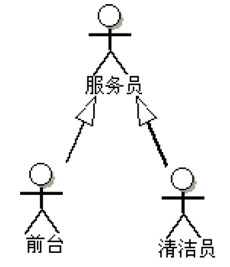
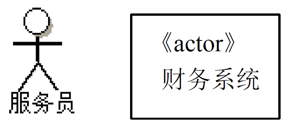
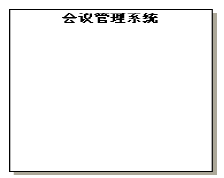
### 附1.3.2 用例图

* 基本作用

用例模型主要用来描述系统和系统外部环境的关系，直接影响着其他模型。用例是一组系统使用场景的集合，每个场景又是由一些事件序列构成，发起这个事件的用户就是系统使用的参与者。用例图是系统的高层描述，角色和用例，在实现阶段则变成了对象和接口这样的底层描述。用例可以帮助每一用户知道他们未来怎样使用系统。对开发者来讲，在不关注细节的情况下，可以快速搜集系统需求，形成总体样式。

* 表示规范

用例模型元素包括：系统（或主题）、角色（或活动者）、用例以及各种关系。



* 用例描述

用例名称： 起草会议申请。

参与者： 会议申请人。

前置条件： 会议申请人有条件通过网络访问系统并已成功地登录系统。

后置条件： 系统保存一份新的会议申请。

基本事件流： 1.用户通过网络登录后成功访问系统。

2.用户选择会议管理后，再选择浏览会议信息。

3.浏览结束后用户选择查看暂存会议申请。

4.在确认无合适的会议申请后，用户选择起草会议申请。

5.用户输入会议申请的相关信息。

6.会议申请经过校验后提交办公室主任。

可选事件流： 1.用户发现有可用的暂存申请可以修改，系统进入修改会议用例界面。

2.新起草的会议申请被暂存。

异常事件流： 1.会议室已被预定，给出错误信息提示。

2.会议信息校验失败，给出错误信息提示。

* 建模过程

（1）首先要确定将要设计的系统和它的边界。

（2）其次要确定系统外的活动者。

（3）从活动者（用户）和系统对话的角度继续寻找一下两方面的特征：

（a）寻找活动者怎样使用系统。

（b）系统向活动者提供什么样的功能。

（4）把离用户最近（接口）的用例作为顶级用例。

（5）对复杂的用例做进一步分解，并确定低级用例以及用例间的关系。

（6）对每一用例做进一步细化。

（7）寻找每一个用例发生的前提条件和发生后对系统产生的结果。

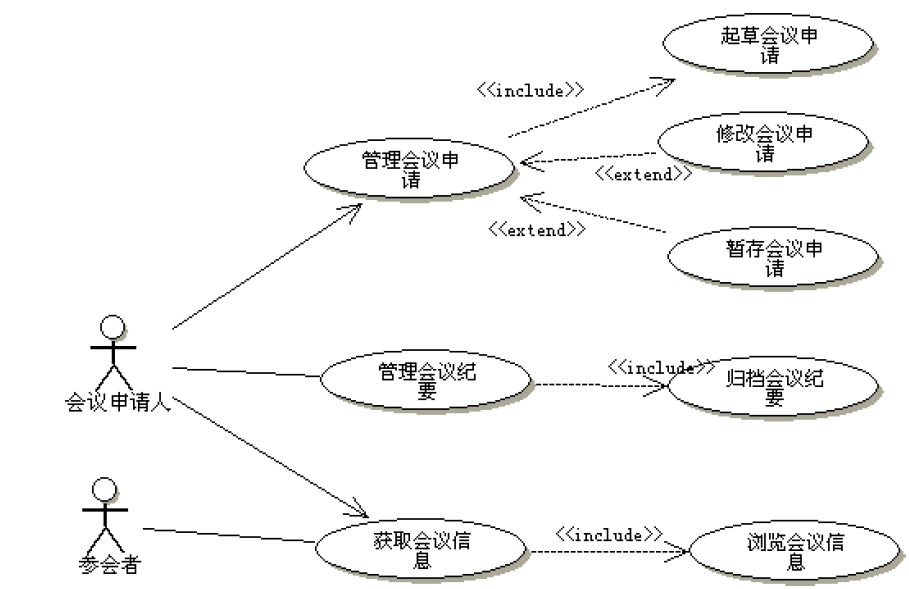
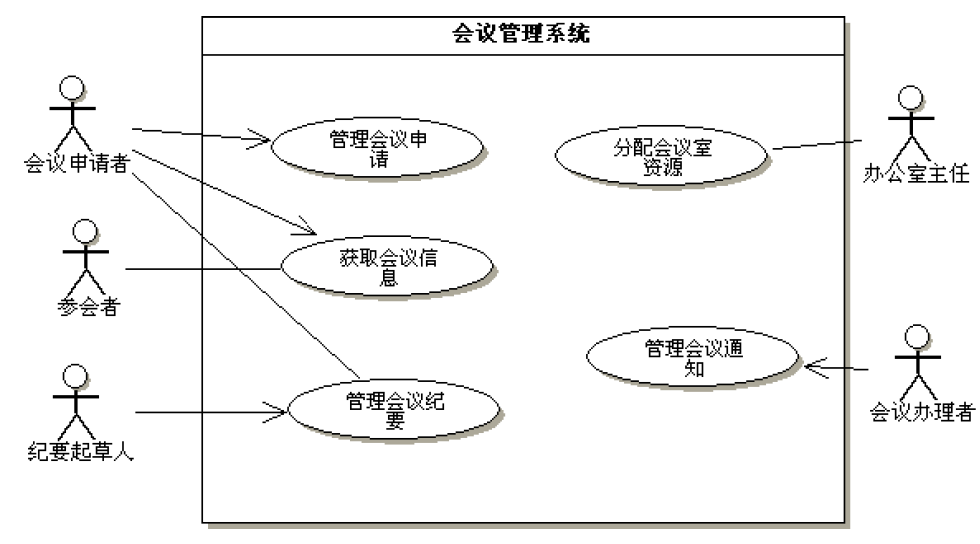
（8）寻找每一个用例在正常条件下的执行过程。

（9）寻找每一个用例在非正常条件下的执行过程。

（10）用UML 建模工具画出用例模型图。

（11）编写用例模型图的补充说明文档。

* 绘图示例（系统用例和业务用例）



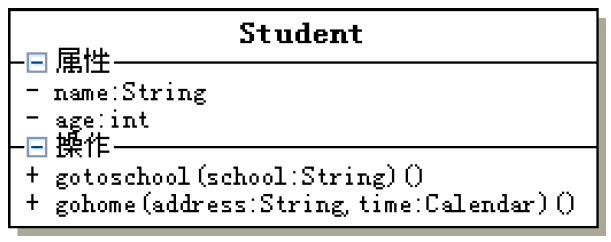
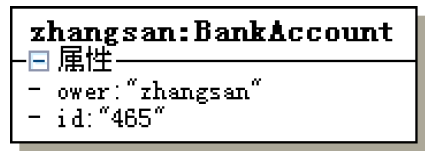
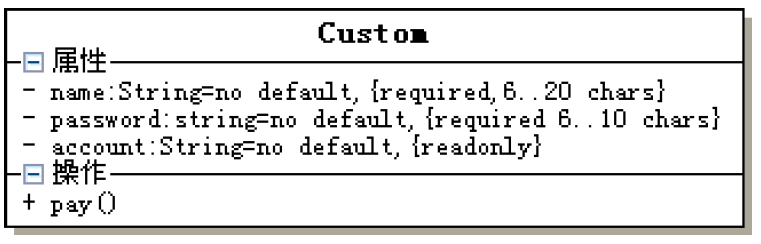
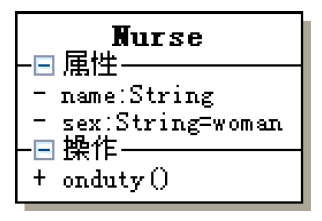
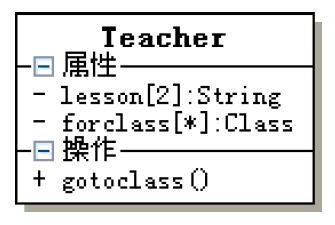
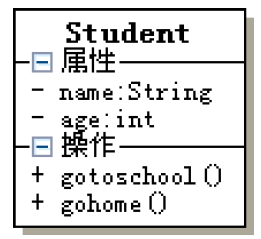
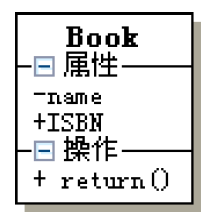
### 附1.3.3 类/对象图

* 基本作用

面向对象的思想来自人们对客观世界的认识，希望把对客观世界的认识，映射到计算机世界。对象是真实世界中某一具体事务，可以是有形的，也可以是无形的。如果把真实世界的每一个对象都搬进计算机，其开发和管理的复杂程度及效率可想而知，因此，面向对象思想提出了一种类的概念。即把满足一组规则的所有对象的集合称为类。这组规则就是类属性和操作。类是一种模板，它的实例化就是对象。

* 表示规范

类图元素包括：类、可见性、操作名称和数据类型、多重性、默认值、特定字符串、对象图、参数列表、返回值、。



* 建模过程

（1） 寻找出需求中的名词(候选对象)。

（2） 合并含义相同的名词，排除范围以外的名词，并寻找隐含的名词。

（3） 去掉只能作为类属性的名词。

（4） 剩下的名词就是要找的分析类（候选类)。

（5） 根据常识、问题域、系统责任确定该类有那些属性。

（6） 补充该类动态属性，如状态、对象间联系（如聚合、关联）等属性

（7） 从需求中的动词、功能或系统责任中寻找类的操作（候选操作）。

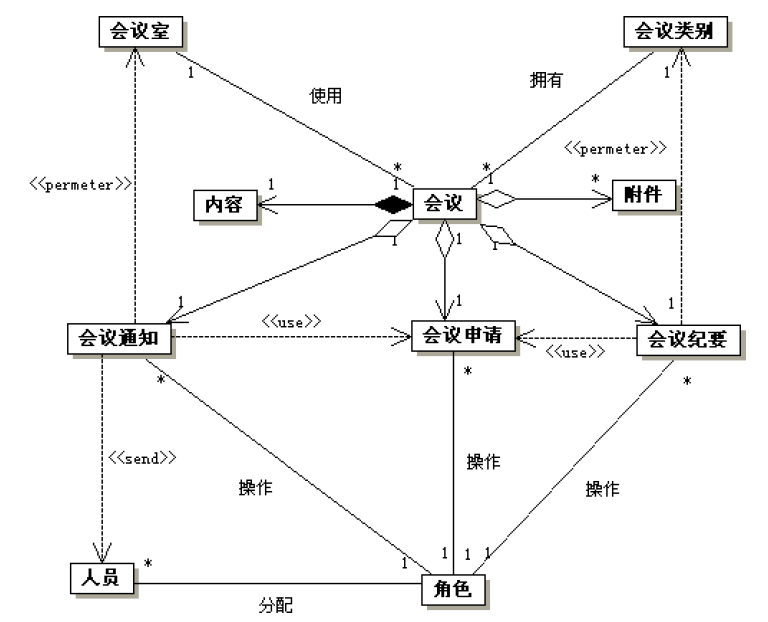
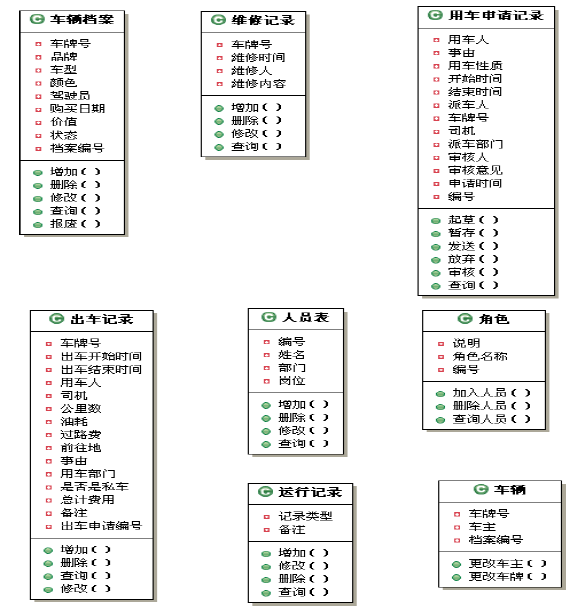
（8） 从状态转换，流程跟踪、系统管理等方面补充类的操作。

（9） 对所寻找的操作进行合并、筛选。

（10） 对所寻找的操作在类间进行合理分配（职责分配）。形成每个类候选操作。

（11） 补充每个类的的分析文档，为类的进一步设计打下基础。

* 绘图示例



### 附1.3.4 顺序图

* 基本作用

显示一组对象为了实现某种功能，而彼此发送和接收的一串消息，这组对象可能是类、接口、构件和节点的具体的或原型化的实例。它是一种以时间为序的表示方法。消息的接收方是通过接口（操作）来接收消息的，消息中的数据是具体的，而不是象类中数据是定义的规则，通过顺序图，可以发现新的操作。顺序图强调消息的事件时间顺序，首先把参与交互的对象或角色放在图上方沿水平轴方向排列，对象的排列顺序从左到右依次为边界对象、控制对象、实体对象。对象的发送和接收消息沿垂直轴方向按时间顺序从上到下放置。

* 建模过程

（1） 完成用例图的分析。

（2）对每个用例，识别出参与基本事件流的对象(包括接口、子系统、角色等)。

（3）识别出这些对象是主动对象还是被动对象。

（4）识别出这些对象发出的消息是同步消息还是异步消息。

（5）从主动对象开始向接收对象发消息。

（6）接收对象再调用自己的服务为主动对象返回结果。

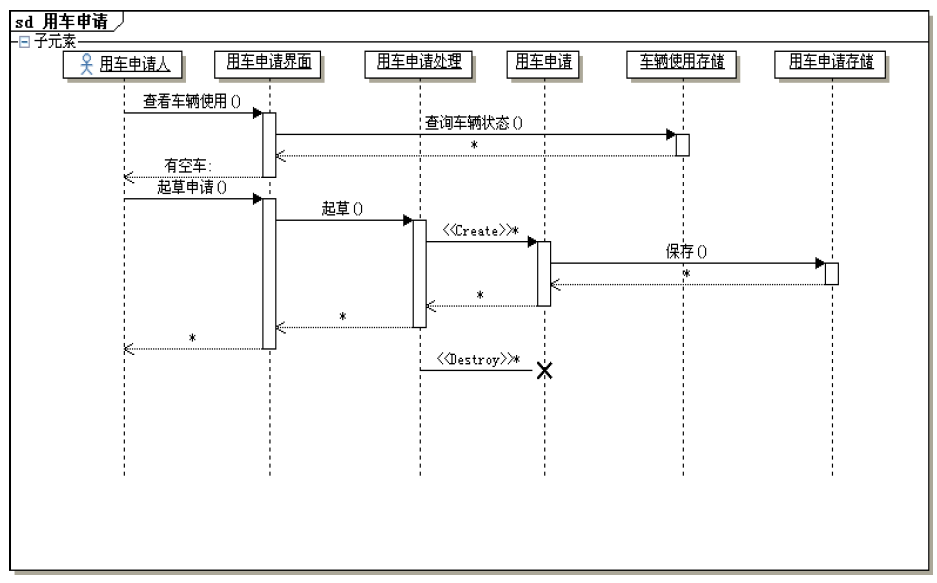
（7）如果接收对象需要再调用其他对象的服务，需要向其他对象再发消息。

（8）如此反复，最后返回给主动对象有意义的结果。

（9）用UML 建模工具绘出顺序图。

（10）给顺序图补充必要的说明文档。

* 绘图示例



### 附1.3.5 通信图

* 基本作用

在UML2.0 中，通信图实际上是以前版本的协作图，使用通信图重点是把消息和对象之间的链直观的布局展示出来，它从空间角度反映对象之间的组织关系。通信图侧重对象之间的交互、对象的结构，有助于验证类间的关联。它同样可以表示消息的类型，如同步消息、异步消息、返回消息、丢失消息、发现消息以及对象的创建消息，但其表示方法和顺序图截然不同。

* 建模过程

（1）完成用例图的分析。

（2）对每个用例，识别出参与基本事件流的对象(包括接口、子系统、角色等)。

（3）识别出对象间的连接关系。

（4）识别出这些对象发出的消息顺序。

（5）从主动对象开始向接收对象发消息。

（6）接收对象再调用自己的服务为主动对象返回结果。

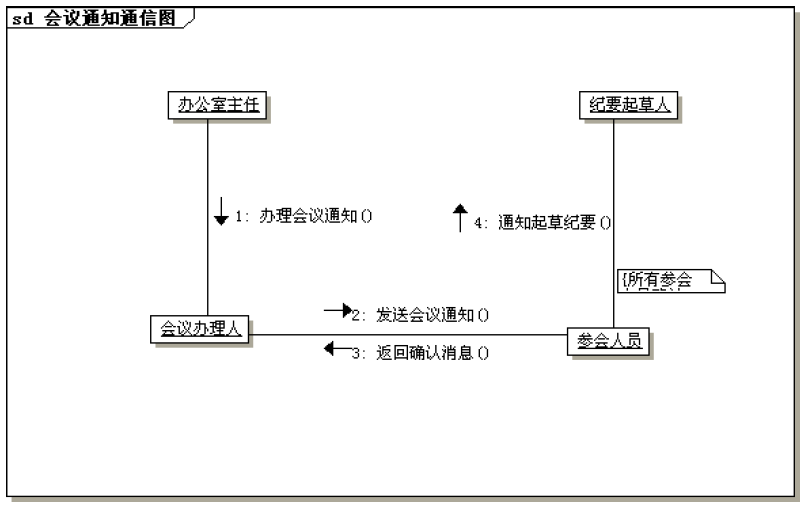
（7）如果接收对象需要再调用其他对象的服务，需要向其他对象再发消息。

（8）如此反复，最后返回给主动对象有意义的结果。

（9）用UML 建模工具绘出通信图。

（10）给通信图补充必要的说明文档。

* 绘图示例。



### 附1.3.6 状态图

* 基本作用

状态图描述交互对对象内部的影响，交互图中的消息在这里变成外部事件对对象发出的命令，对象对这些命令的响应导致对象的状态发生变化。因此，从这个意义上说，状态图是顺序图的进一步细化，并且是对核心对象（选择核心对象的依据是看是否在多个交互图中有多个消息指向该对象）的细化。

* 建模过程

（1）首先要确定进行系统控制的对象，可以从前面分析的顺序图中寻找。

（2）确定对象的起始状态和结束状态。

（3）在对象的整个生命周期寻找有意义的控制状态。

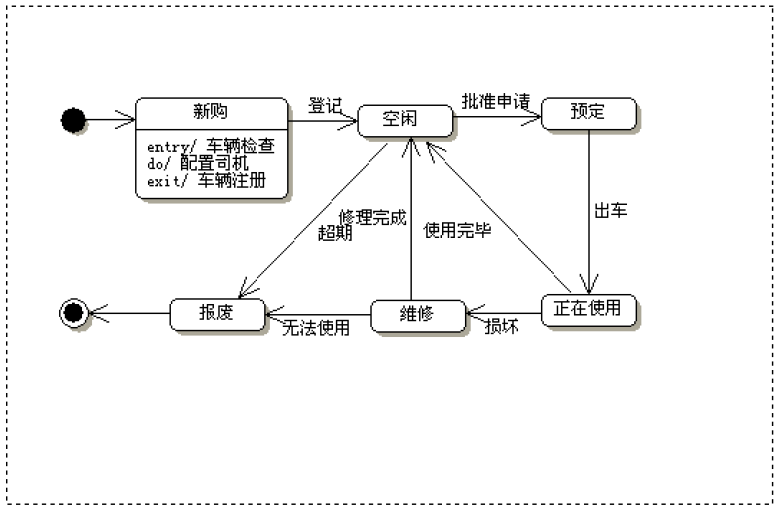
（4）寻找状态之间的转换。

（5）补充引起转换的事件。

（6）UML 建模工具画状态图。

（7）补充必要的文档。

* 绘图示例。

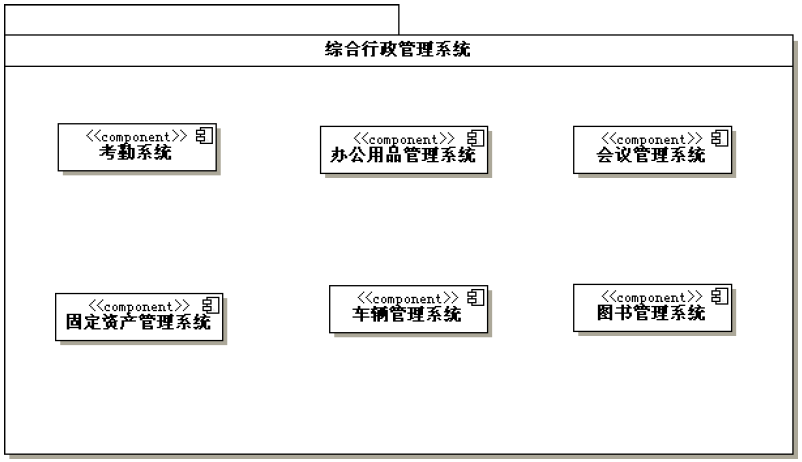


### 附1.3.7 组件图

* 基本作用

组件又叫构件，代表从类到应用、子系统和系统的任何事物，组件是对物理实现类型的定义，是一个抽象的物理软件需求。每个组件定义了一个或多个和其他构件通信的接口，组件暴露了所包含类的一个或多个接口，而不是全部接口（其余用于内部通信）。一方面组件依靠自身的类指定其行为。另一方面，组件依靠多个实现它的制品。组件是一种容器，没有自己的特征，一般用组件表示一个业务过程。

* 绘图示例。



### 附1.3.8 部署图

* 基本作用

部署图是展示系统实现环境的静态视图，对一个系统的管理可以从多个方面描述，每一个描述构成了一个部署视图，对整个系统的完整描述必须由多个部署视图来完成。部署图是由节点和节点之间的关联组成，节点代表处理资源的类型或执行工作的任何事物，其上的制品可以被部署和执行。节点具有属性和操作，也可参与关联，节点的属性和操作一般用注释来表示。

* 绘图示例。

