Thinking in UML

第11章 系统分析

报告人：马银萍

本章主要内容

- 11.1、确定系统用况
- 11.2、分析业务规则
- 11.3、用况实现
- 11.4、软件架构和框架
- 11.5、分析模型
- 11.6、组件模型
- 11.7、部署模型

11.1 确定系统用况

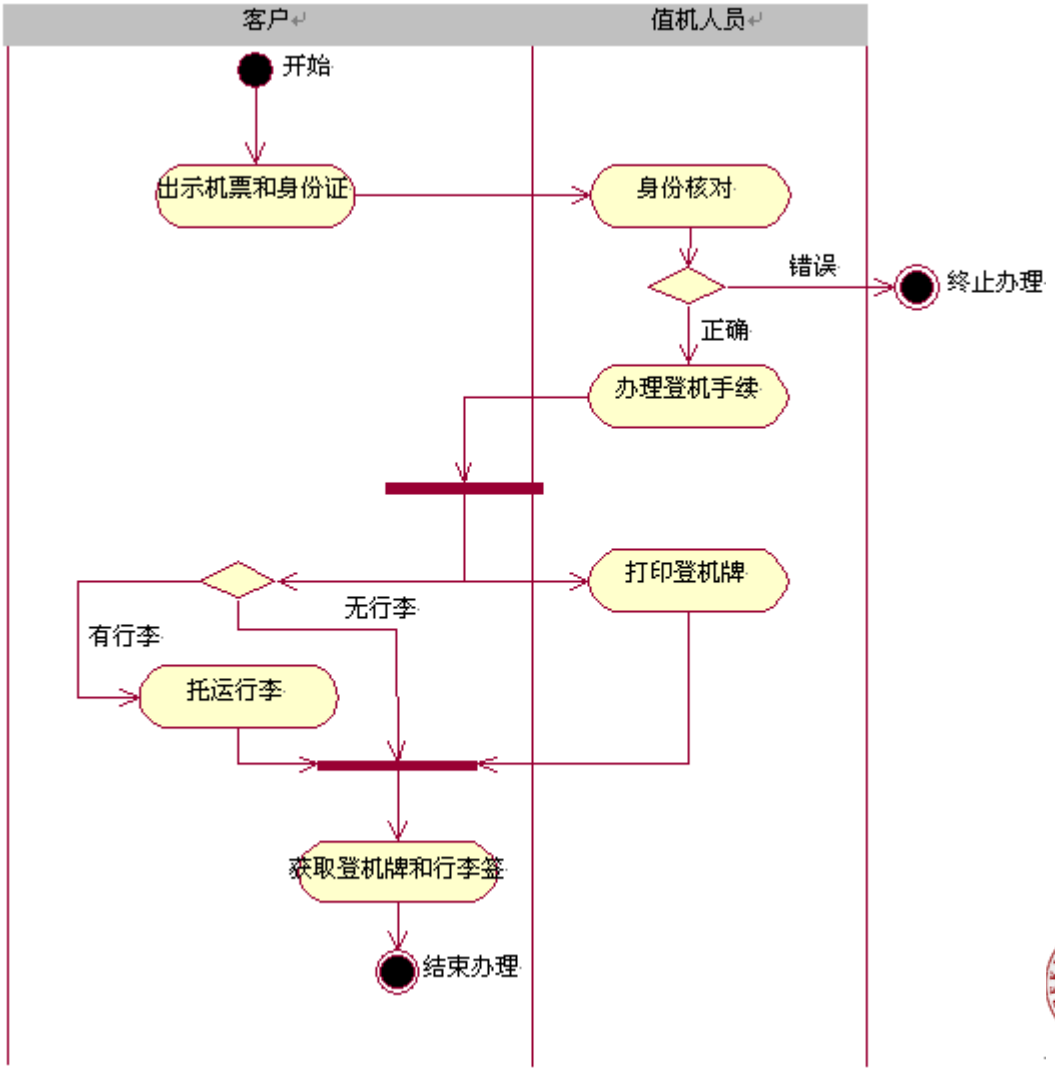
- 目的：确定系统范围

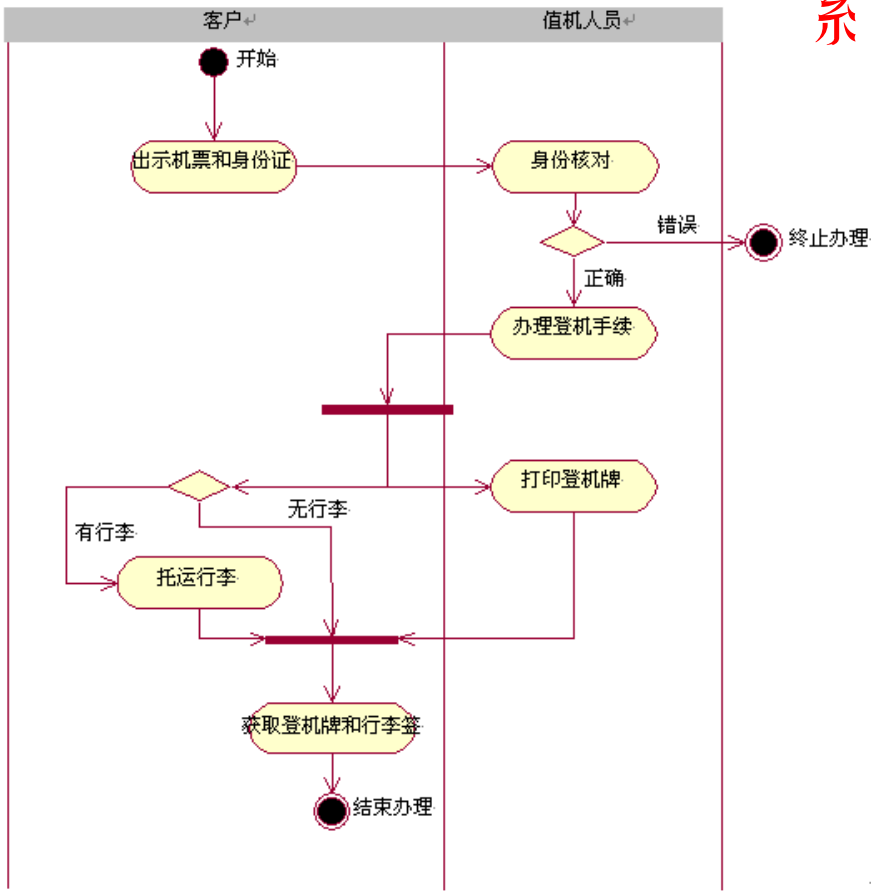
- 从业务用况到系统用况，更适当的说法是抽象关系，或者说是映射关系。我们可以说从业务用况当中抽象出系统用况，也可以说是把业务用况映射到系统用况。关键的问题是怎么抽象与怎么映射。

11.1 确定系统用况

- 分析概念模型：从业务用况场景当中找到概念用况
- **系统用况**：分析业务用况场景，从业务用况场景当中抽出那些可以在计算机当中实现的单元来。业务用况场景通常被描述成某某做什么，然后某某又做什么……，某某做什么就是系统用况的来源

下图是一个办理登机手续业务用况场景。
备选的系统用况：客户出示机票和身份证、值机人员核对身份证、值机人员办理登机手续、值机人员打印登机牌





系统用况：可以用计算机实现

判断是否为系统用况：

1）客户出示机票和身份证：

--非系统用况

2）值机人员核对身份证：

--传统方式，肉眼核实，非系统用况

--电子客票，身份证验证，系统用况范围

3）值机人员办理登机手续：

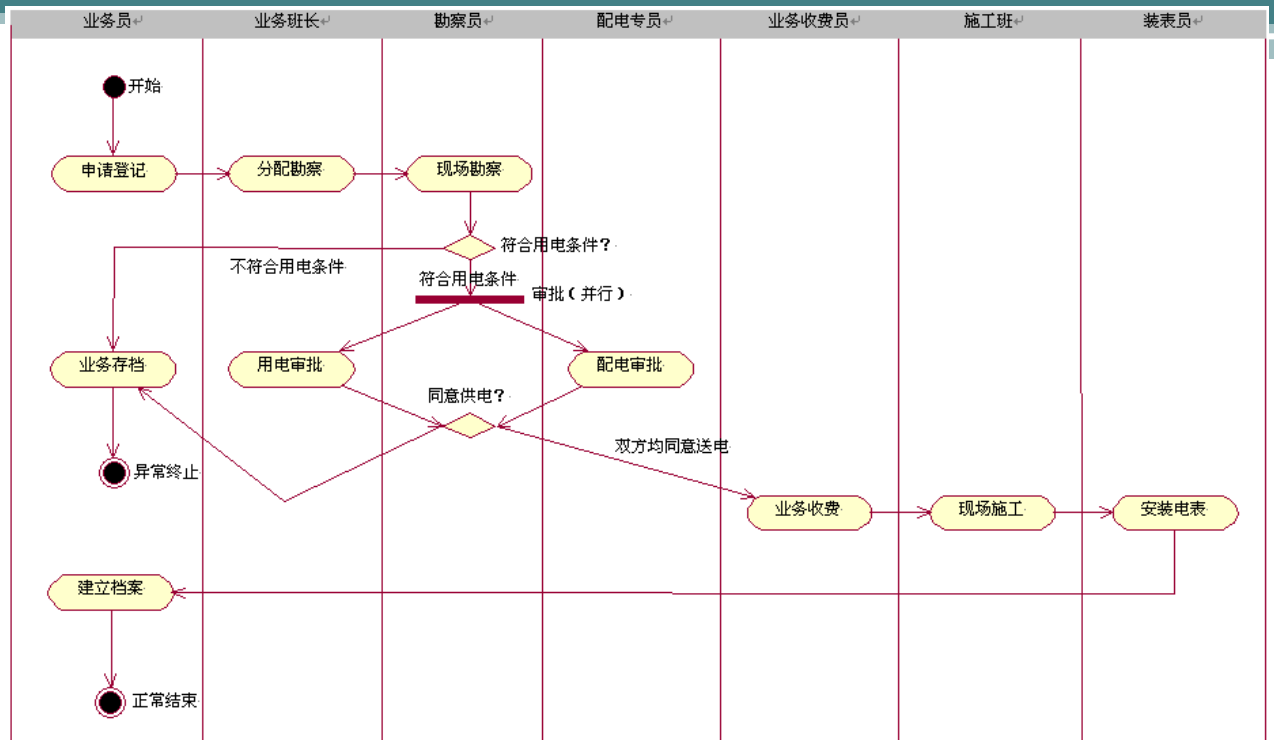
--系统用况

4）值机人员打印登机牌

---取消值机人员打印登机牌这个备选用况的独立用况资格，将它作为值机人员办理登机手续的一个包含用况

判断备选系统用况是否被纳入系统的基本方法：

- **映射**：最直接的办法
- **抽象**：找出备选用况中计算机真正做的事情
- **合并**：当业务场景中备选用况不具备独立性时，它必然是其他某个事件的组成部分。
- **拆分**：用况场景中的备选用况粒度很大，在这个备选用况当中包含几件事情，就需要进行拆分
- **演绎**：业务用况场景中找不到备选用况，或者备选用况看上去不适合用计算机来实现。但我们能够预见到某个可能的系统用况潜伏在这个场景当中，我们就需要使用演绎法将它找出来。



举例：低压用户申请永久用电业务用况场景

针对低压用户申请永久用电业务场景中的活动逐个分析：

1.申请登记　　---直接映射成系统用况

2.分配勘察　　---抽象出一个查询勘察员的系统用况

3.现场勘察　　---拆分成打印勘察单和录入勘察单两个系统用况

4.是否符合用电条件　---交互类规则，不是系统用况

5.业务存档　　---抽象出终止业务流程系统用况

6.用电审批　　---直接映射成系统用况

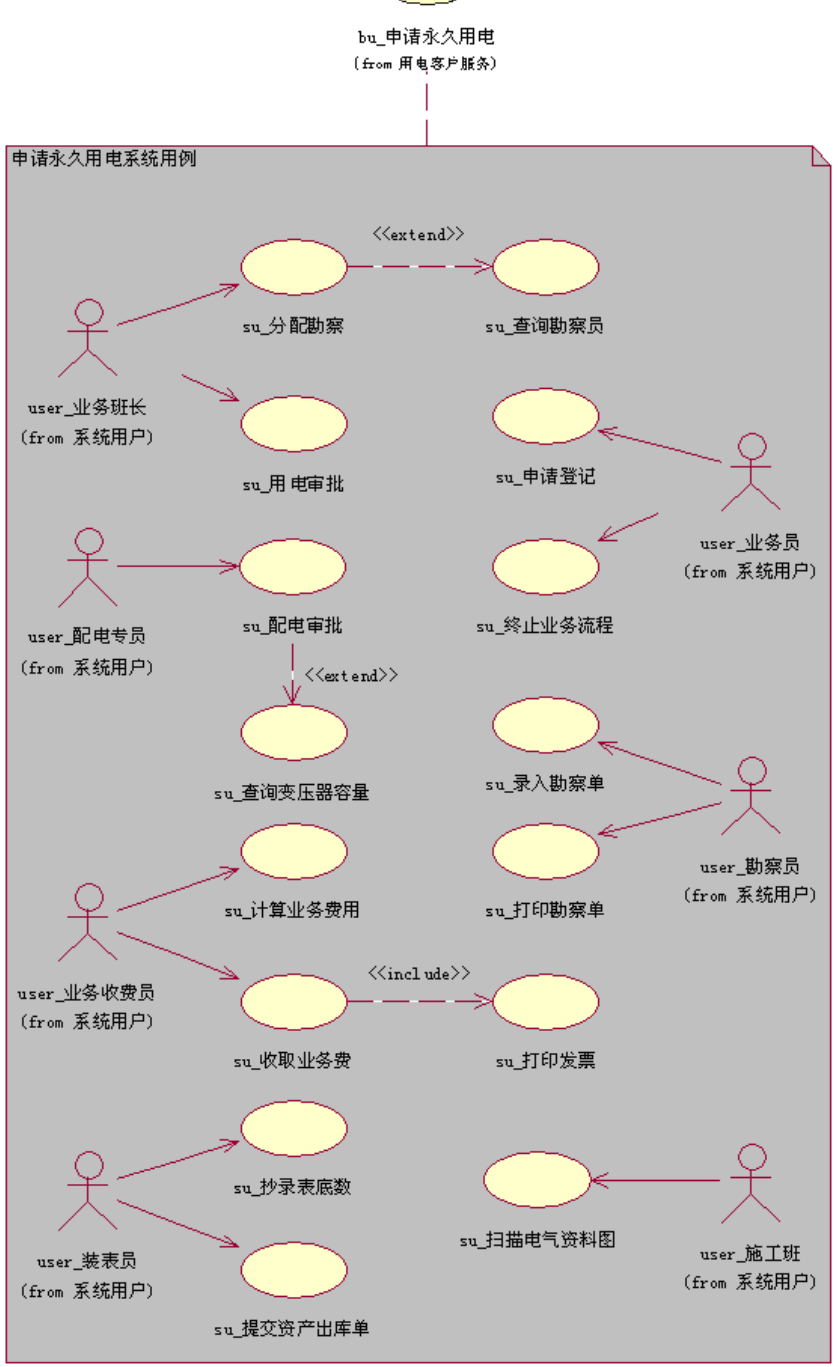
7.配电审批　　----从配电审批系统用况拆分出查询变压器容量系统用况

8.业务收费　　---拆分出计算业务费和收取业务费两个系统用况

9.现场施工　　---抽象出扫描电气资料图系统用况

10.安装电表　---抽象出抄录表底数系统用况，增加提交资产出库单系统用况。

- 举例：低压用户申请永久用电业务用况场景
- 针对低压用户申请永久用电业务场景中的活动逐
- 1.申请登记　　---直接映射成系统用况
- 2.分配勘察　　---抽象出一个查询勘察员的系
- 3.现场勘察　　---拆分成打印勘察单和录入勘
- 4.是否符合用电条件　---交互类规则，不是系统
- 5.业务存档　　---抽象出终止业务流程系统
- 6.用电审批　　---直接映射成系统用况
- 7.配电审批　　----从配电审批系统用况拆分出
- 8.业务收费　　---拆分出计算业务费和收取业
- 9.现场施工　　---抽象出扫描电气资料图系统
- 10.安装电表　---抽象出抄录表底数系统用况，增
- 右图：申请永久用电系统用况图



如何描述系统用况

- 描述系统用况的方法和描述业务用况方法如出一辙，描述系统用况的过程，也就是系统建模的过程。
- 之前描述的是原来的业务是什么样子的。工作人员应该怎样完成业务，而现在的描述应该变成计算机怎样做。工作人员怎样操作计算机。

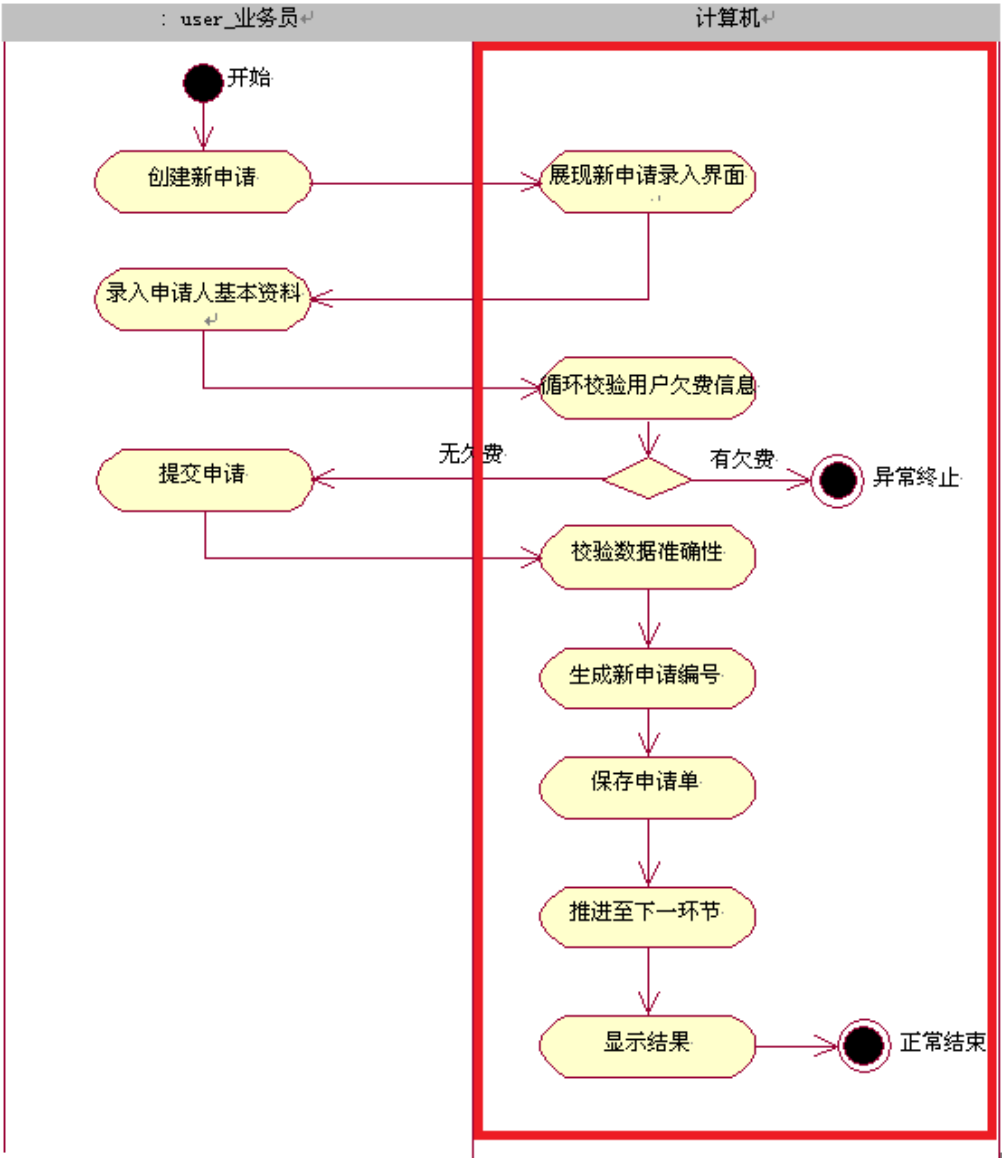


图11-4 申请登记用况场景实例

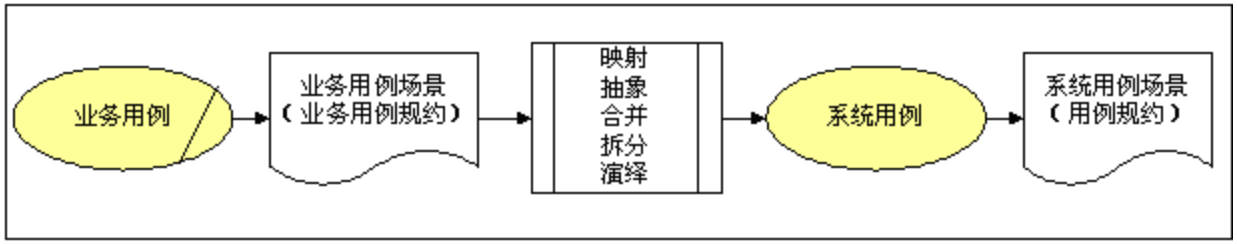
- **选择活动图原因:**
- **活动图:** 适合解释职责类的场景。
- 活动图相当于纲领，不容易因为要处理信息过多而丢失内容
- 系统用况仍属于要向客户提交的文档范围，活动图客户容易看明白并提供反馈。
- 测试！活动图绘制的用况场景，基本就是一个黑盒测试的现成的测试用况

表 11-1　用例规约示例

用例名称	su_申请登记
用例描述	业务员创建新的申请单，录入用电客户申请资料，创建申请流程
执行者	业务员（代理用电客户操作）
前置条件	业务员成功登录系统
后置条件	1．创建新的申请单并生成唯一的申请编号 2．创建新的永久用电申请流程实例 3．推进至分配勘察流程环节 4．提交后的申请单不得再修改
主事件流描述	1．业务员选择创建申请单，计算机展示申请单录入界面，执行 2；业务员选择继续编辑保存过的申请单，执行 3 2．业务员录入用户名称，计算机自动查询该用户在历史上有无欠费记录，应用业务规则 a。若有欠费记录，执行异常过程 2.1.1；无欠费记录执行主过程 3 3．业务员录入其他资料，选择提交，执行主过程 4；选择保存，执行分支过程 3.1.1；选择放弃，执行分支过程 3.2.1 4．计算机校验数据准确性，应用业务规则 b，若有不符合的数据，执行分支过程 4.1.1，否则执行主过程 5 5．计算机生成唯一申请编号 6．计算机保存申请单 7．计算机将申请过程推进至下一环节 8．计算机向业务员展示申请单最终结果，用例结束
分支事件流描述	3.1.1 计算机保存目前录入的信息，生成临时编号 3.2.1 计算机不保存任何数据，用例结束 4.1.1 计算机提示错误数据详细情况，提示业务员，返回 3
异常事件流描述	2.1.1 该用户名历史上有欠费记录，计算机显示欠费情况 2.1.2.1 业务员确认该欠费情况属实，用例终止 2.1.2.2 业务员确认情况有误，返回 3
业务规则	a．根据用户名从欠费历史中查询该户名有无欠费记录，若有记录，由人工判断该用户名欠费是否属实，若属实应停止申请（这是一条交互性业务规则，若该业务规则已经在业务规则文档中记录，此处可直接引用规则编号而无须文字解释） b．用户名、身份证号、地址、用电类别……必填（这是一条内禀规则，应像现在这样写在用例规约文档里）
涉及的实体	Be_申请单 Be_现场勘察单 Be_业务收费清单 Be_电表安装工作单 Be_用户档案 Be_收费账号 Be_结算账号

- 用况规约描述系统实现需求的所有细节

- 讨论一：从业务需求到系统需求
- 系统需求能够追溯到业务需求，系统实现能追溯到系统需求。



- 需求可追溯的关键是映射、抽象、合并、拆分和演绎过程能否被记录下来。
- 讨论二：业务用况和系统用况的粒度
- 建议：系统建模阶段，用况的粒度以一个用况能够描述操作者与计算机的一次完整交互为宜。
- 以本书的供电企业管理信息系统用况来说，业务用况是以永久用电申请业务为粒度；概念用况以核心业务当中的关键步骤为粒度；系统用况基本上就是一次完整的人机交互过程为粒度。

11.2 分析业务规则

- 业务规则是程序逻辑的一部分
- 业务规则引擎

---将业务规则从程序逻辑中剥离出来，通过业务规则管理工具将其纳入业务规则库。应用程序处理过程中需要用到业务规则时通过业务规则引擎解释业务规则并返回所需的结果。

- 业务规则通常以决策表、决策树、规则预言和脚本的形式来维护。

11.2 分析业务规则

- 业务规则分为全局规则、交互规则和内禀规则。
- 全局规则交由架构师处理，交互规则交由设计师处理，内禀规则交由程序员处理。
- 分析业务规则的目的是从业务规则中发现那些将对系统构成重大影响的部分，将其转化为系统需求。

分析全局规则

- 全局规则是值那些对于系统大部分业务或者系统设计都起到约束作用的规则。

- 举例：用电管理企业管理系统中，对于业务规则：所有的办理业务产生的相关文件都要存档，原始手续文件也要存档，以保证整个业务办理过程的痕迹以供事后查证。
- 将这条全局规则交由架构师处理，由架构师在软件架构的层次上解决历史数据版本管理问题。

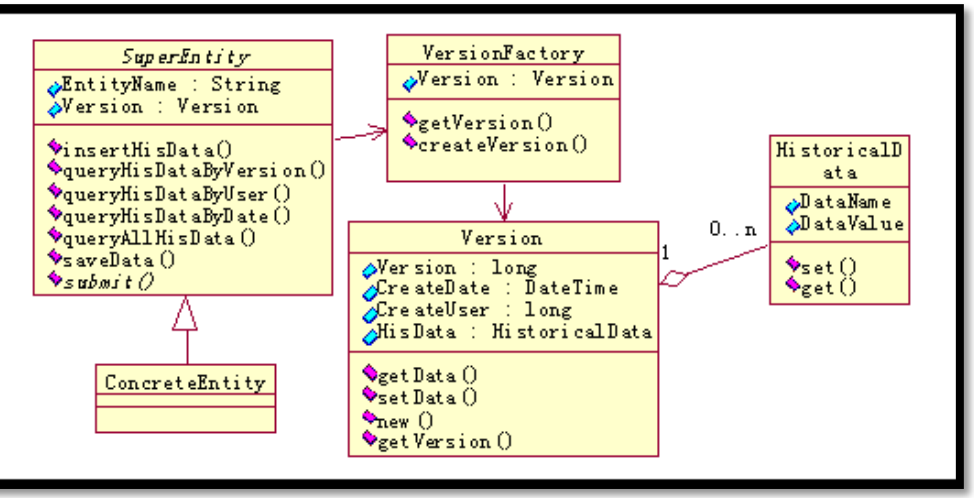


图11-6　历史数据管理框架示例

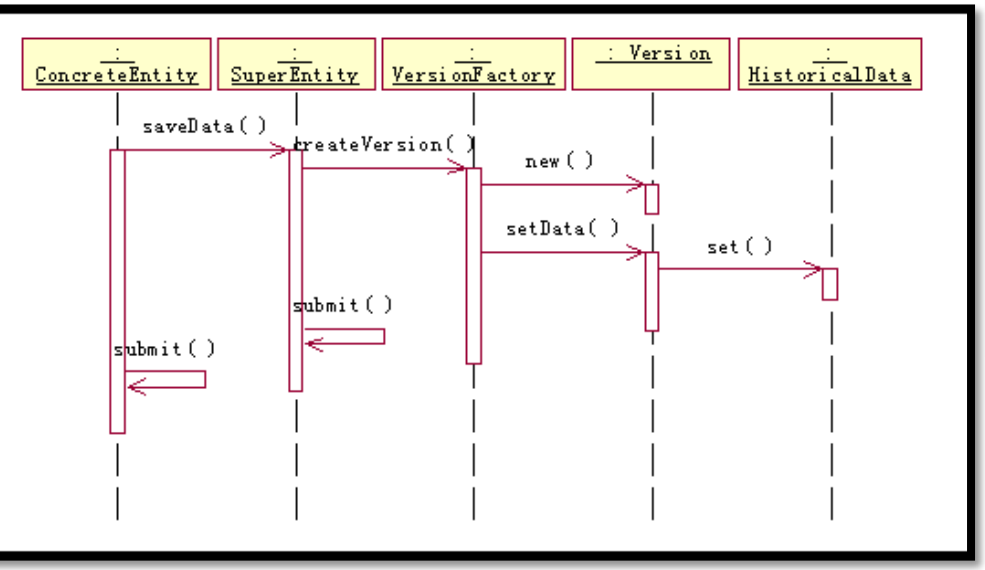


图11-7　创建历史数据过程

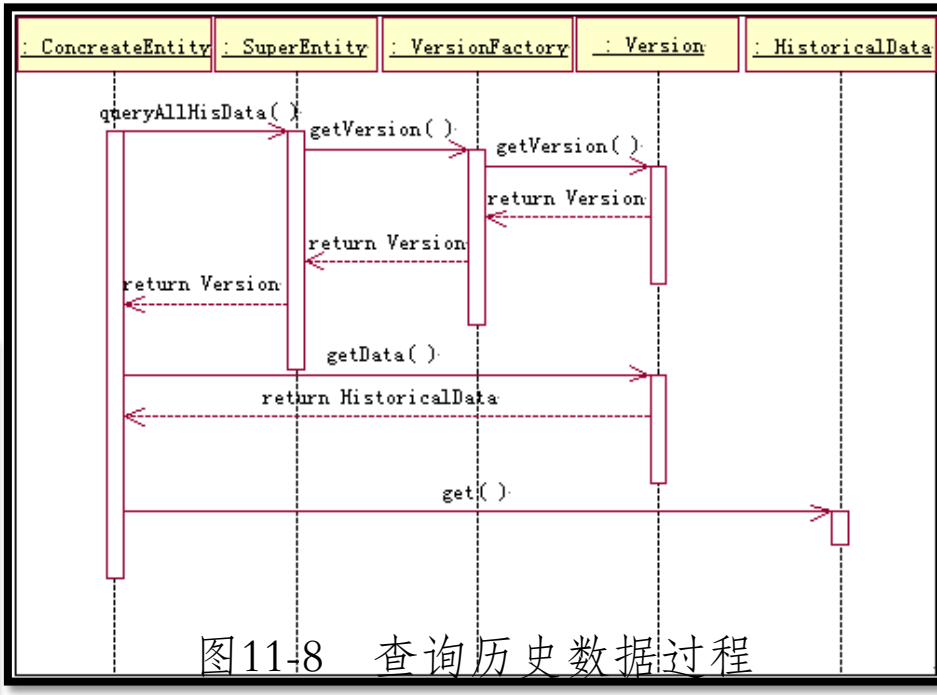


图11-8　查询历史数据过程

北京大学 PEKING UNIVERSITY

分析交互规则

- 产生于用况场景中。
- 用况场景由活动图、交互图等来描述，不论是活动、状态还是业务对象，他们都在活动转移、状态变迁和对象交互时必然会有一些限制性的条件。这些条件就是交互规则。

分析交互规则

- 低压用电申请业务用况场景中两条业务规则：
  - 是否符合用况条件？符合则继续办理流程，否则终止流程办理。
    - 人工判断，不需过多处理，只需留下输入最终结论的地方
  - 用电审批和配电审批是否都同意供电？是则继续办理，否则终止流程。
    - 需要计算机判断，业务班长和配电专员分别签署自己的意见，计算机需要进行逻辑判断，然后返回结果来决定流程走向。这种情况下考虑计算机来实现业务规则。

分析交互规则

- 申请登记用况场景中业务规则：
  - 根据用户名从欠费历史中查询该户名有无欠费，若有记录，有人工判断该用户名欠费是否属实。若属实应停止申请。
    - 根据输入是否有欠费记录，有记录，还需加上人工判断，申请登记的业务会被打断。编写申请登记记录的程序员还需编写欠费查询的程序。
    - 这条业务规则跨越了用况，而且跨越了部门，欠费信息来自营业财务部门营业会计负责的统计欠费明细业务用况。
    - 如果在申请登记的程序逻辑中加入欠费查询的逻辑，就表示这两个用况之间产生了依赖关系。换句话说，申请登记程序是否能正常运行，依赖于统计欠费明细用况是否能正常返回结果。

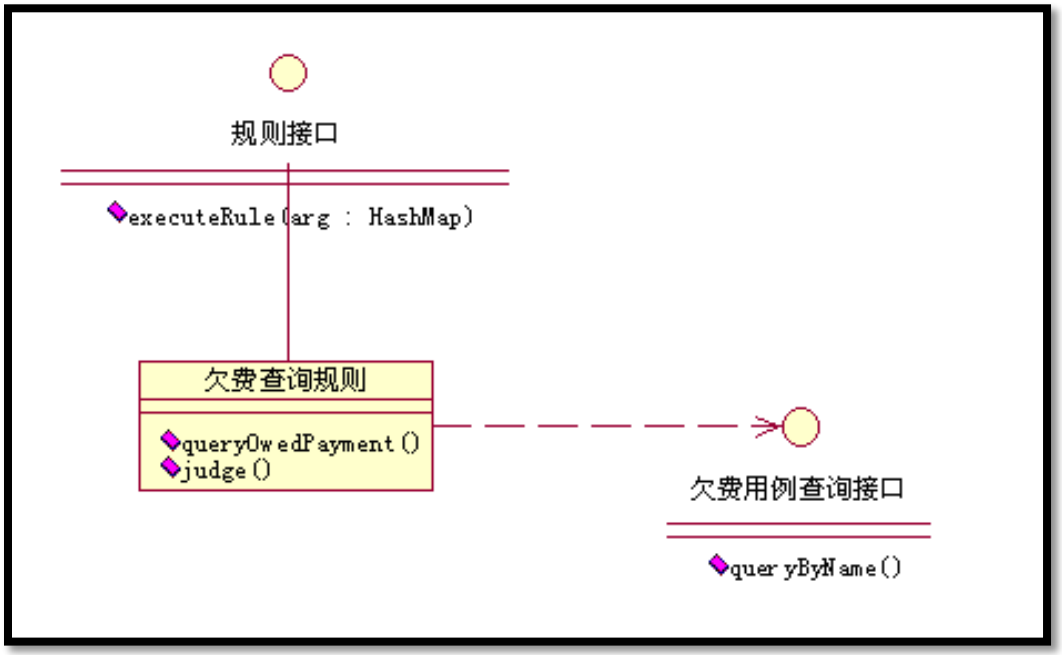


图11-9　欠费业务规则类示例

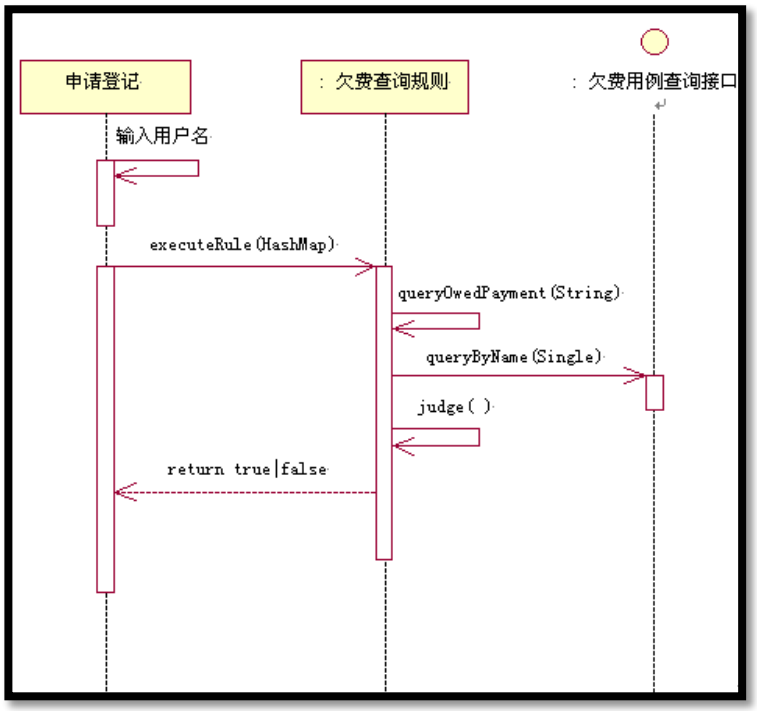


图11-10　欠费业务规则实现示例

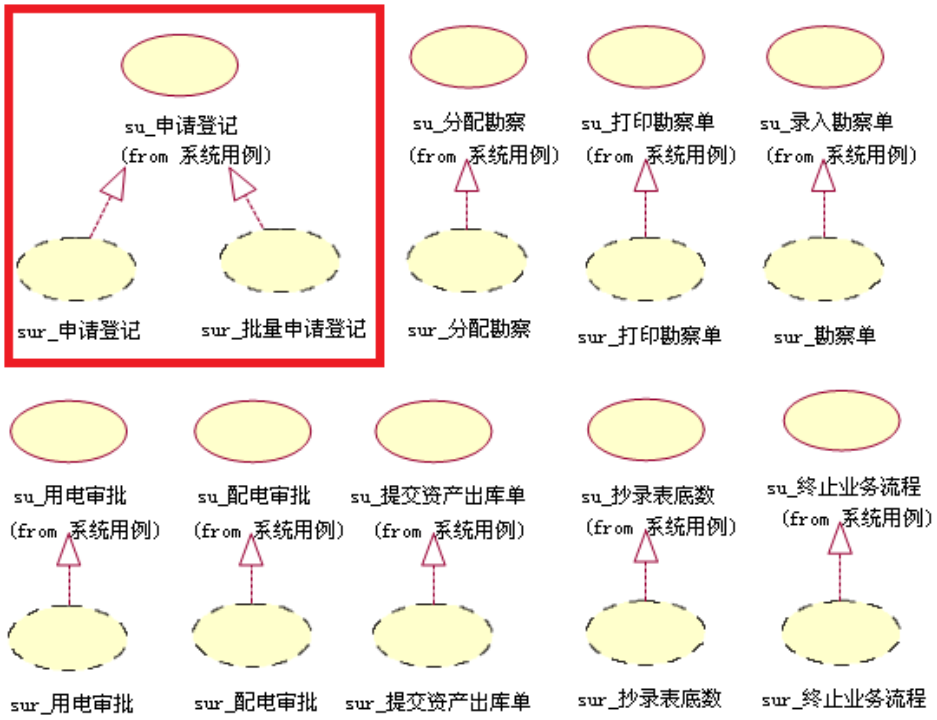
分析内禀规则

- 业务本身具备的，并且不因为外部的交互而变化的规则。
- 例：填写申请单时，用户名、身份证号、地址、用电类别等为必填项。这样的业务规则与其他用况无关，也不会因为跟不同的对象交互而变化。

11.3 用况实现

- 用况的实现方式，实现系统需求，将设想变成现实。
- 跨越系统需求到设计模型之间的桥梁
- 三类重要对象：边界类对象，控制类对象和实体类对象
- 为用况实现建模，经过三个步骤
  - 第一步：在用况场景中发现和定义实体对象
  - 第二步：用控制对象来操作和处理实体对象中的数据
  - 第三步：用边界对象来构建接收外部指令的界面

例：su_申请登记用况



低压电申请用电系统系统用况

图11-13 用况实现到系统用况关系图

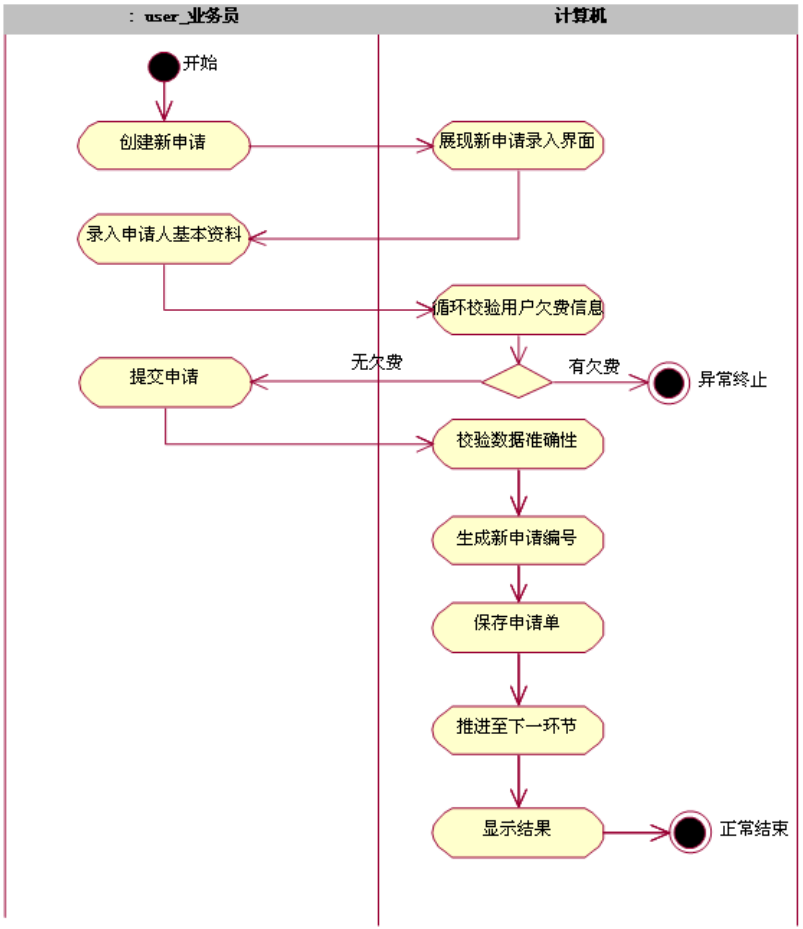
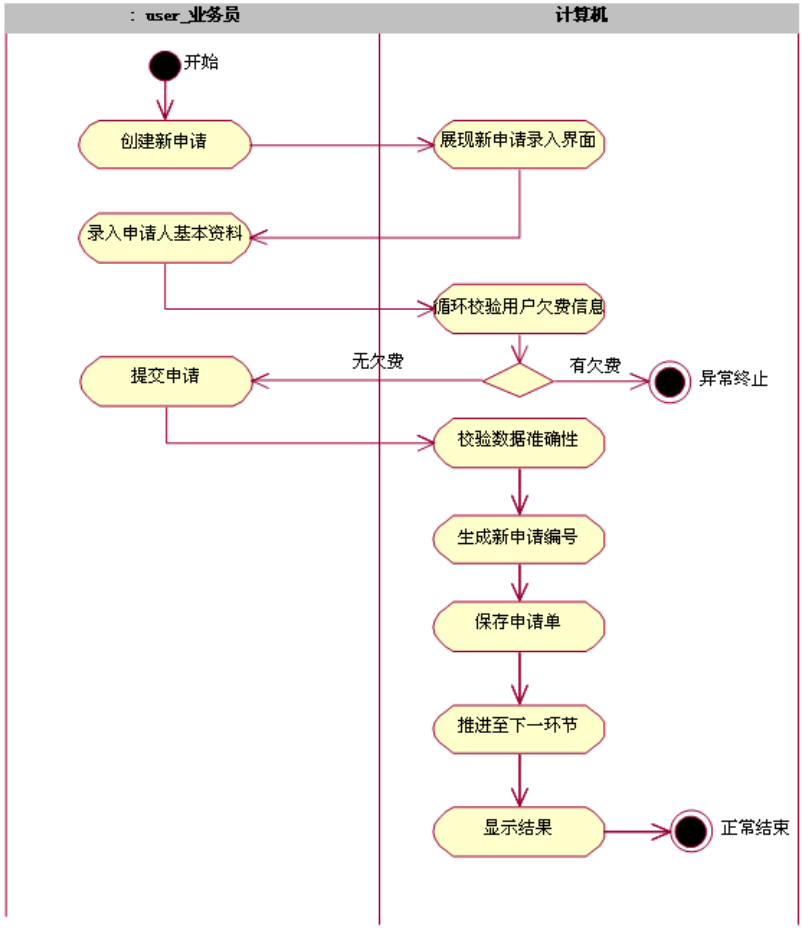


图11-14 申请登记用况场景实现

例：su_申请登记用况



- 创建新申请 ---边界类对象接收指令
- 展示新申请录入界
  ---控制类对象处理，结果反映到边界对象
- 录入申请人基本资料 ----人工活动
- 校验用户欠费信息 ----业务规则
- 提交申请 ---边界类对象接受指令
- 校验数据准确性 ---控制类对象处理
- 生成新申请编号 ---控制类对象处理
- 保存申请单 ---控制类对象处理
- 推进至下一环节 ---控制类对象处理
- 显示结果
  ---控制类对象处理，结果反映到边界对象

图11-14 申请登记用况场景实现

例：su_申请登记用况

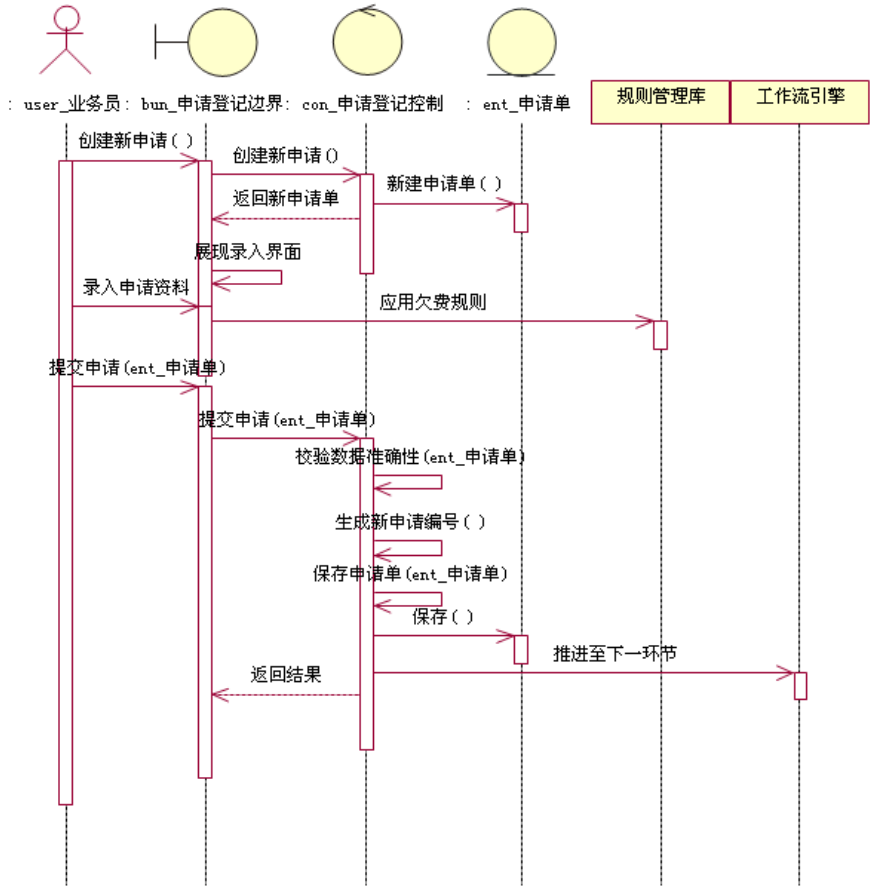


图11-15 sur_申请登记用况实现

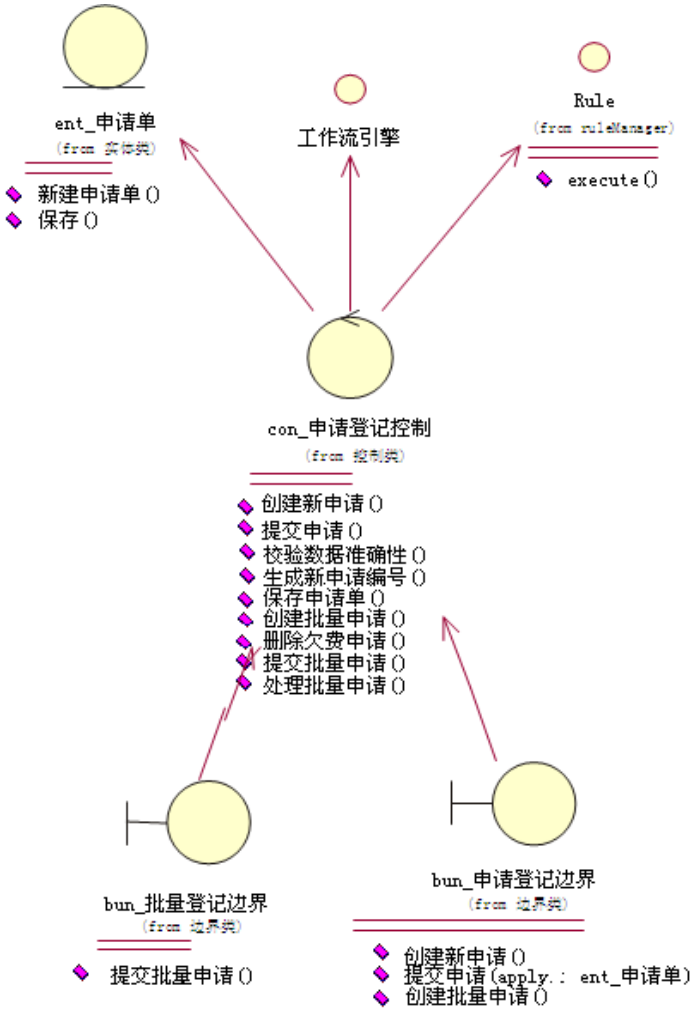


图11-18 申请登记分析类图

- 讨论一：分析类是沟通需求和设计的桥梁
- 分析类实现了用况场景，而用况场景描述了需求
- 使得从需求到设计的推导有迹可循，可以验证

- 讨论二：为什么用分析类而不是设计类来实现用况场景
- 分析类是从业务需求向系统设计转化过程中最为主要的元素，他们是在高层次抽象出系统实现业务需求的原型，分析类的抽象层次高于设计实现，高于语言实现，也高于实现方式。
- 分析类抽象层次高，可以不用理会实现时的复杂设计要求，分析类比设计类验证需求的工作量以及可能的变化要少很多。
- 分析模型的高抽象层次易于人们理解系统

11.4 软件架构和框架

- 软件架构
  - 软件架构是一种思想，一个系统蓝图，对软件结构组成的规划和职责设定。
  - 软件架构的意义就是将这些可逻辑划分的部分独立出来，用约定的接口和协议将它们有机结合在一起，形成职责清晰、结构清楚的软件结构
- 软件框架
  - 软件架构的一种实现，是一个半成品。通常针对一个软件结构当中某一个特定的问题提供解决方案和辅助工具
- 例如，MVC是一种设计思想，是软件架构，而Struts，JSF，WEBWork实现了这一架构，是软件框架

软件架构

- 通用的软件架构，最重要的是规范和协议
- 特定的软件产品，一个软件架构包括软件层次、每一个层次的职责、层次之间的接口、传输协议和标准以及每一层次上所采用的软件框架。

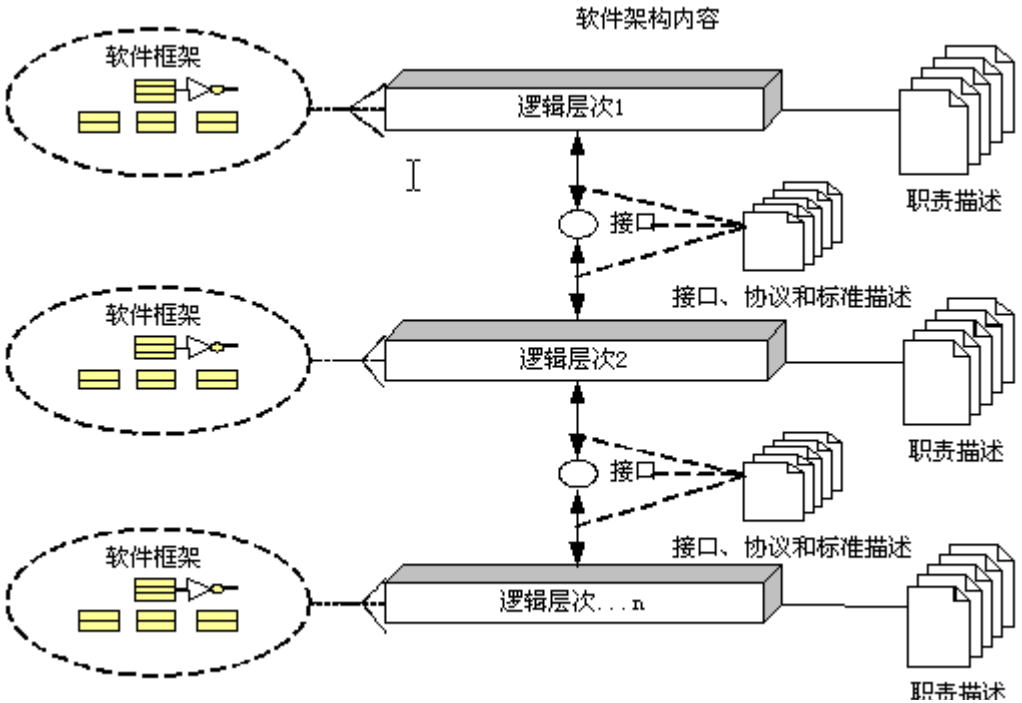


图11-22 软件架构的内容

软件架构

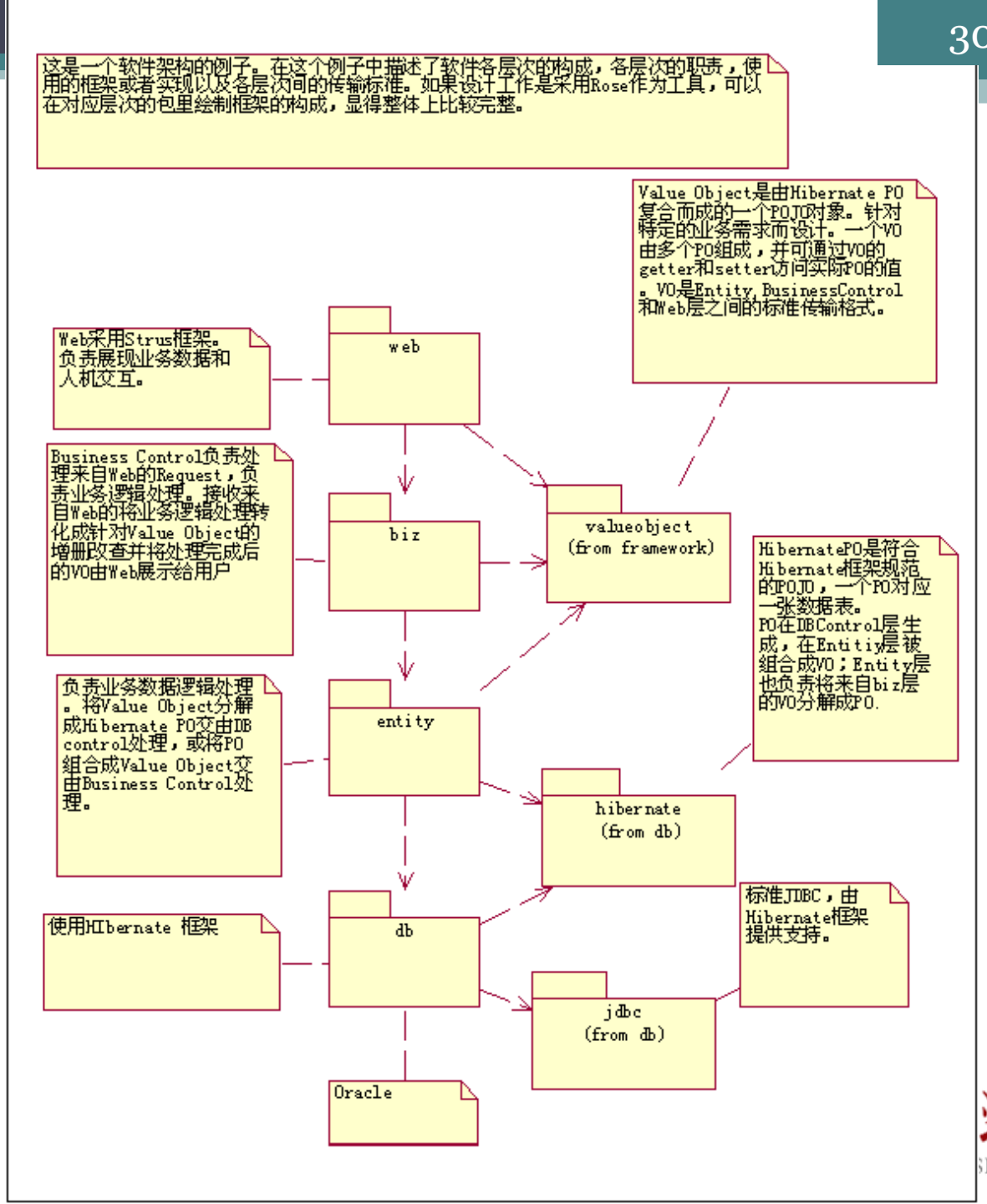


图11-23 用包图描述软件架构

软件架构

这是Entity层的一个框架例子。只是为了示例，并不完善，读者不必深究。这个例子的部分思路来自SDO规范。对这个框架有兴趣的读者可以直接去研究SDO规范，这里只简单的解释一下。
1. PO是一个虚拟超类，它的子类是可以直接被Hibernate操作的标准POJO对象。一个PO对应一张数据表。id是唯一主键，name是PO的逻辑名称，易于被应用程序访问。Verb对应SDO规范，指示对PO的操作是什么。PO给出了默认的增删改查实现，子类继承即可。
2. Relationship维护组成VO的多个PO之间的对应关系。Hibernate可维护PO之间的一对多，多对一，多对多等关系，但这些关系是指关系型数据库的表之间的关系。Relationship管理的是非数据库的，业务逻辑要求的关系。
3. VO是一个虚拟超类，是Business层和Web层使用的数据标准格式。VO提供了默认的实现，可以让程序通过一个统一的接口访问聚合在VO中的PO的实际值。当对数据有修改时，VO根据Relationship决定该采取的下一步动作。
4. EntityControl是Business层访问Entity层的接口类。EntityControl负责将PO组合成VO或将VO分解成PO。Business层通过访问EntityControl来存取VO，同时Entity负责将分解出的PO传给DBControl

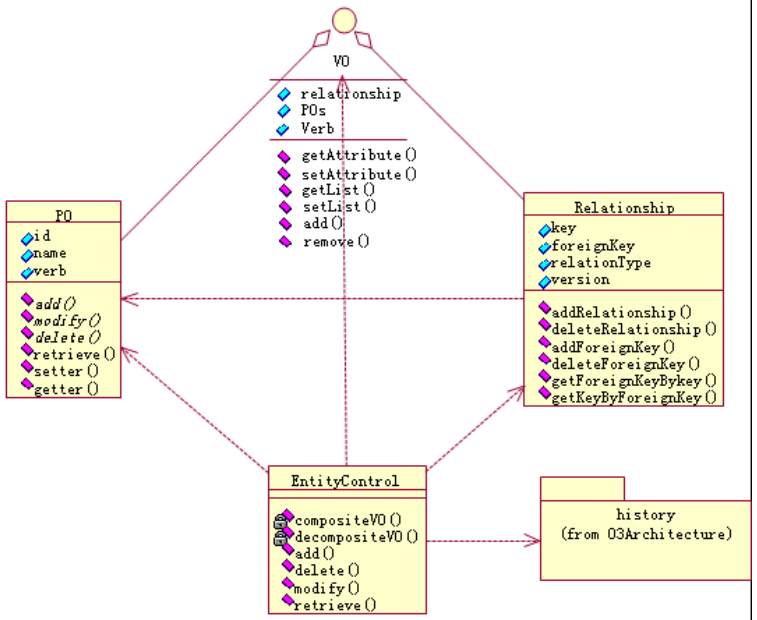


图11-24 框架实现示意图

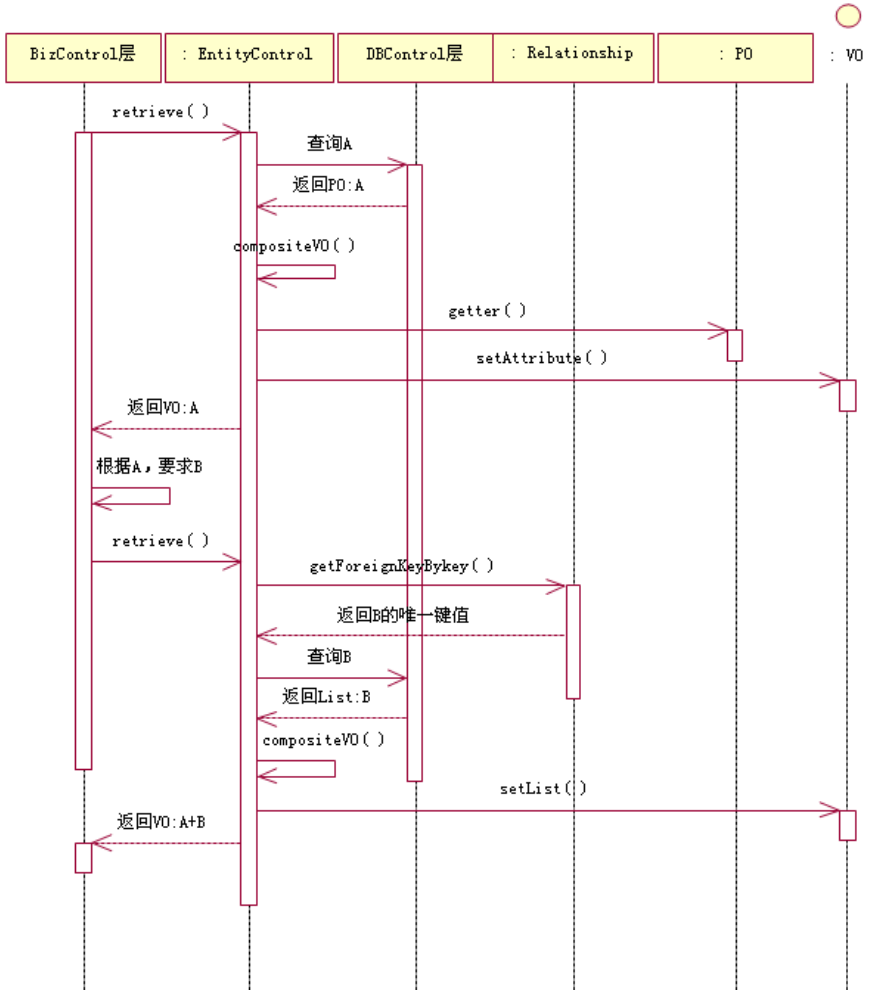
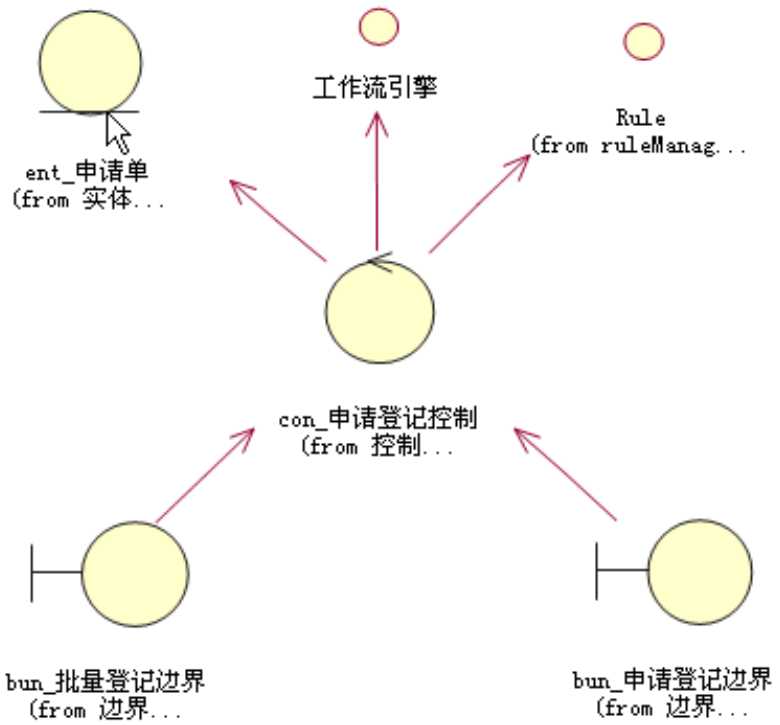


图11-25 查询数据架构实现示意图

11.5 分析模型

- 在建立领域模型时，采用分析模型来获得针对某一问题领域的系统视角理解；在建立概念模型时，采用分析模型来获得针对核心业务的系统视角理解；在建立用况实现模型时，采用分析模型获得针对系统需求的系统视角理解。
- 使用分析模型获得系统需求在软件架构上的系统视角理解
- 建立分析模型的过程，是采用分析类，一步步将系统需求这个蓝图在软件架构和框架构成的骨架当中注入血肉的过程。

例：申请登记用况



- bun_批量登记边界和bun_申请登记边界 ---WEB层
- con_申请登记控制 ---BusinessControl层
- 工作流引擎 ---BusinessControl层
- Rule接口 ---BusinessControl层
- ent_申请单

---以不同形式位于WEB、BusinessControl、Entity层

图11-26 申请登记分析类图

例：申请登记用况

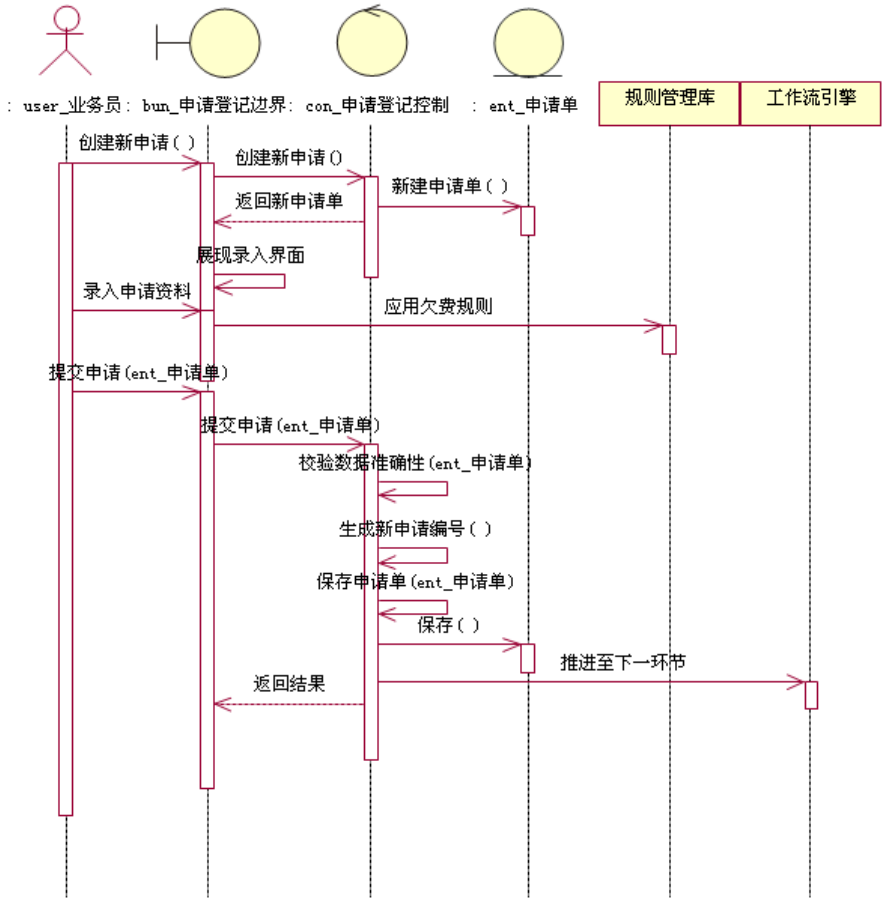


图11-15 sur_申请登记用况实现

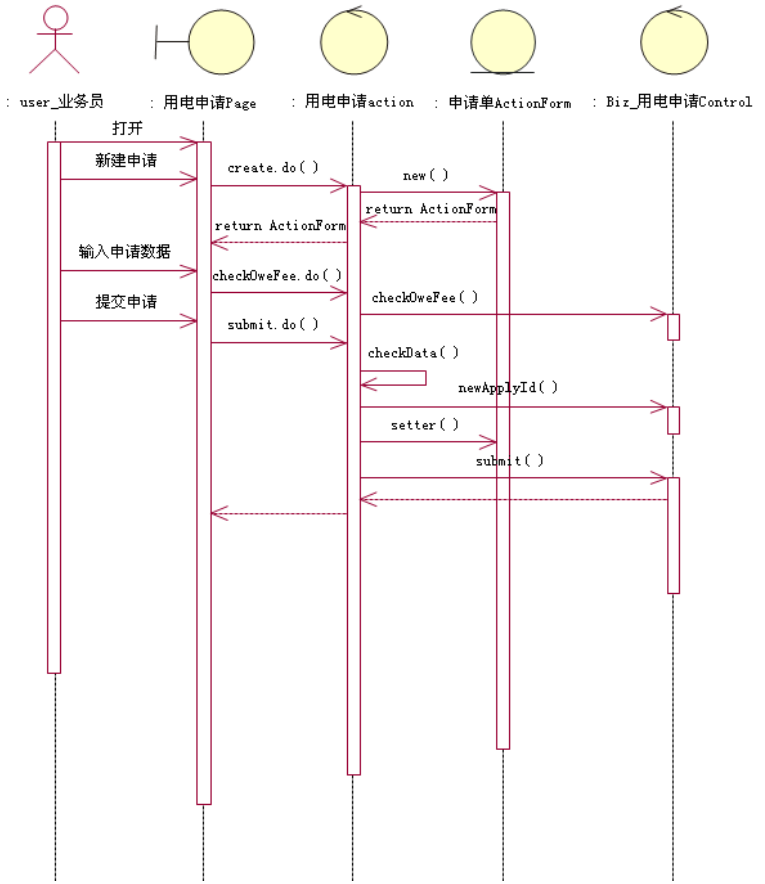


图11-27 申请登记WEB层分析模型实现

例：申请登记用况

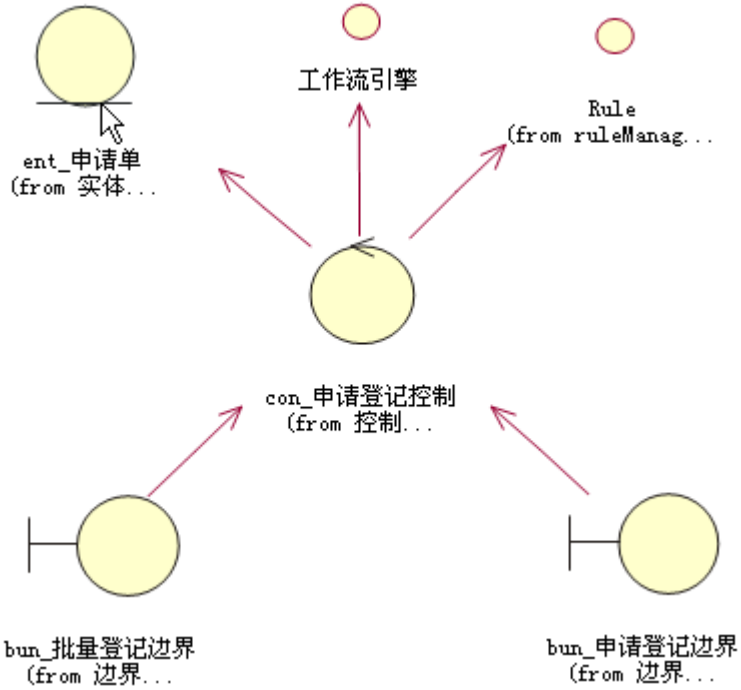


图11-26 申请登记分析类图

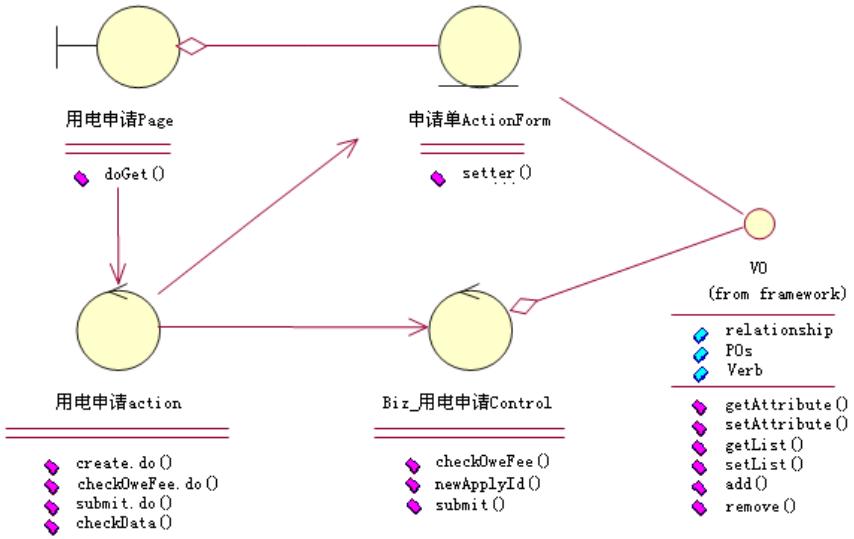
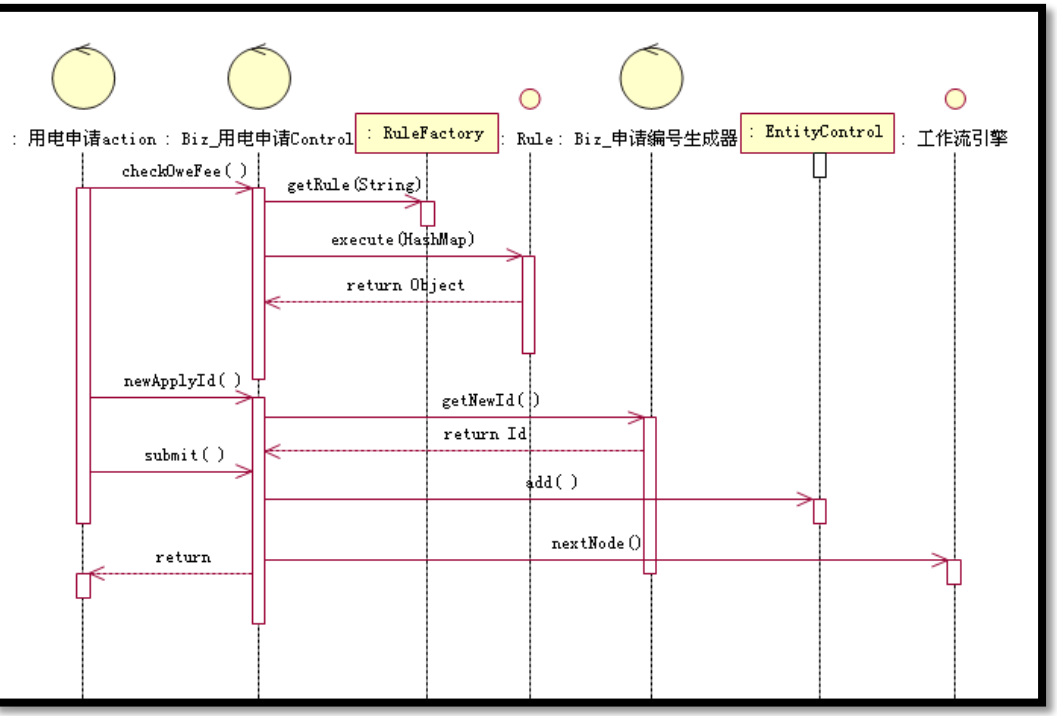


图11-28 申请登记WEB层分析类图

例：申请登记用况



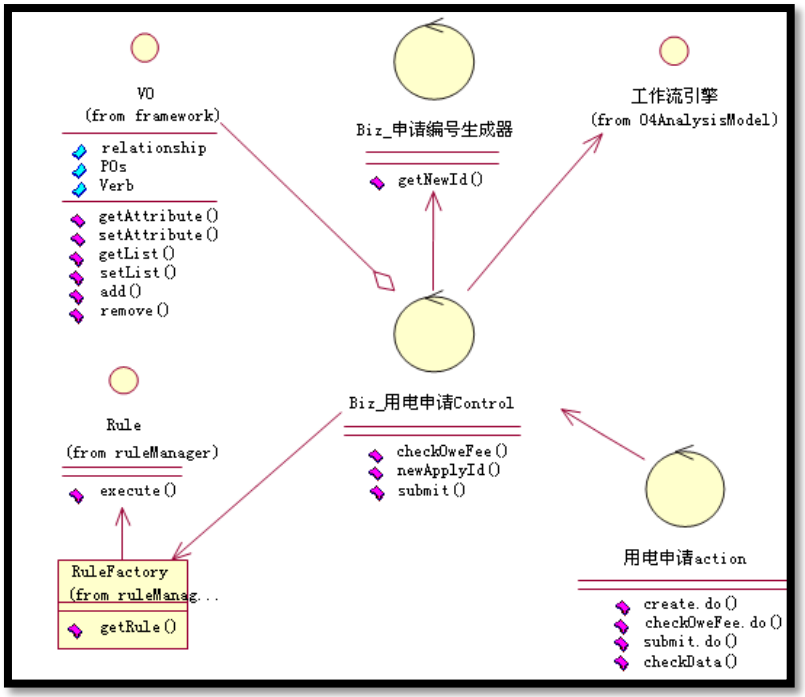


图11-29 申请登记BusinessControl层实现

图11-30 申请登记BusinessControl层分析类图

例：申请登记用况

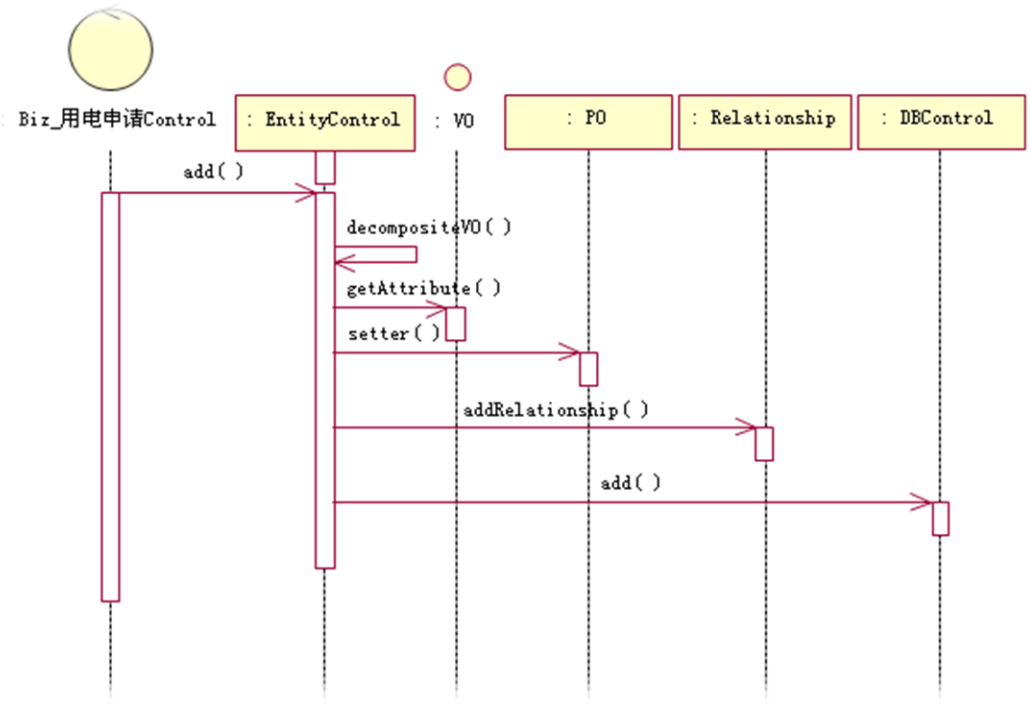


图11-31 申请登记Entity层实现

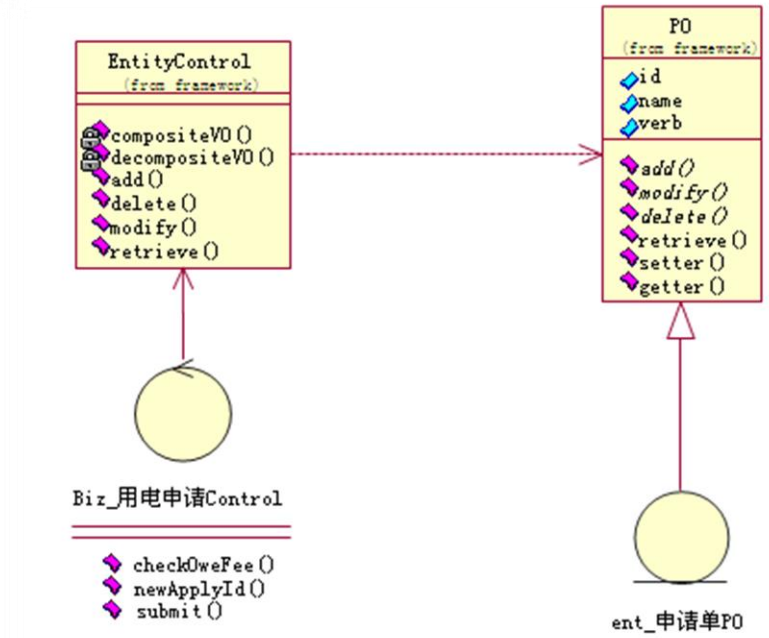


图11-32 申请登记Entity层分析类图

例：申请登记用况

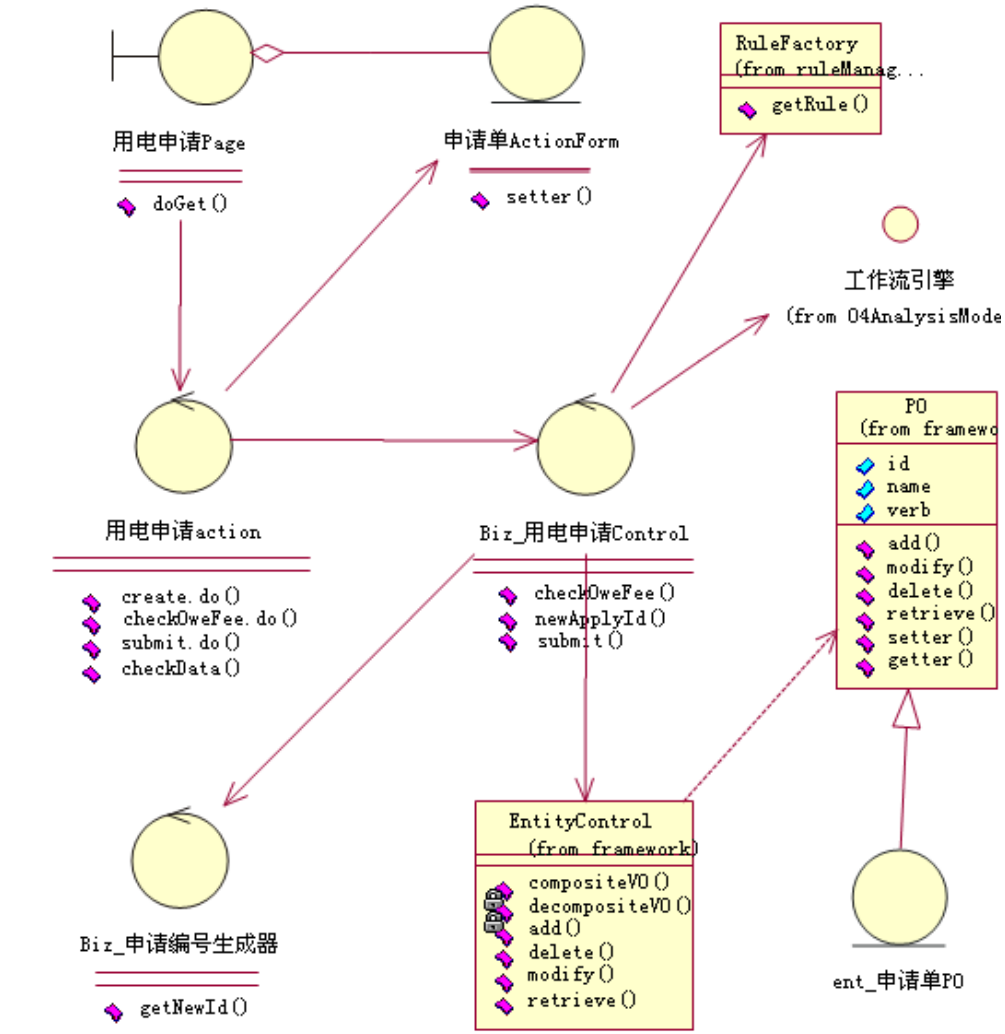


图11-33 申请登记用况最终分析模型

- 推导过程：
- 通过用况确定系统需求
- 通过用况实现，得到系统需求的计算机视觉理解
- 规定软件架构，去认定软件层次
- 在每一个层次上决定适用的软件框架
- 分析用况实现在每个软件层次上如何动作的
- 根据每个软件层次上适用的软件框架并使用分析类来实现用况
- 综合各个软件层次得到分析类，形成分析模型
- 实现系统需求最基本的类和方法

11.6 组件模型

- 组件是用来容纳分析类或设计类的
- 目的是将一些类组织在一起完成一组特定的功能
- 建立组件模型需要考虑的问题：
  - 这些组件将成为可复用的单位
  - 这些组件都完成了一个或一组特定的功能
  - 这些组件将成为可独立部署的单位
  - 每个组件都要遵循架构规范

例：申请登记用况

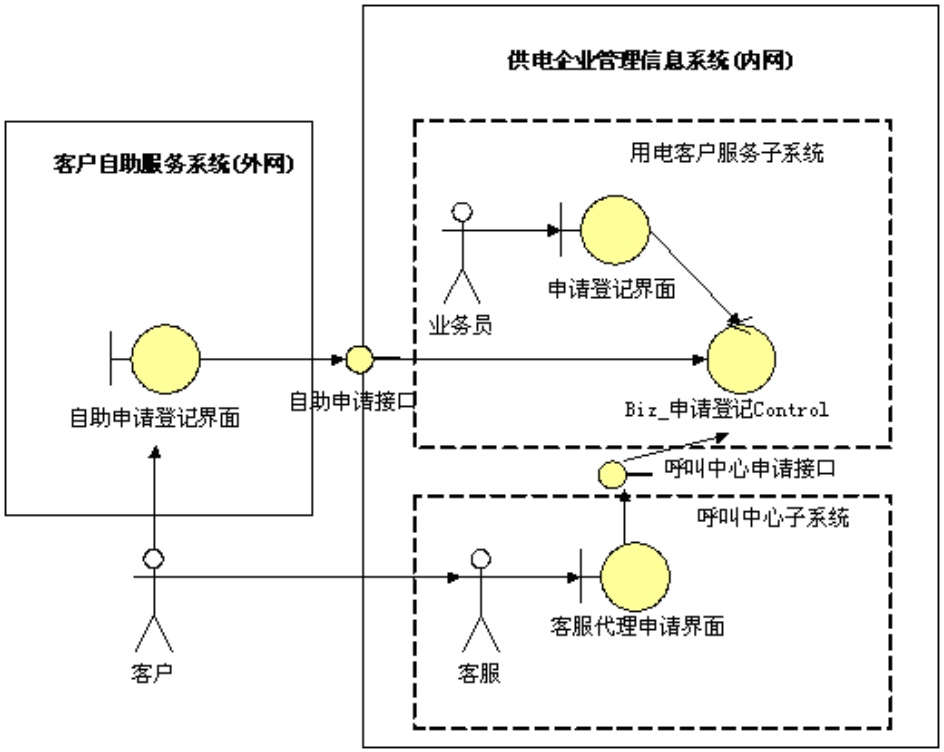


图11-34 申请登记业务运行环境

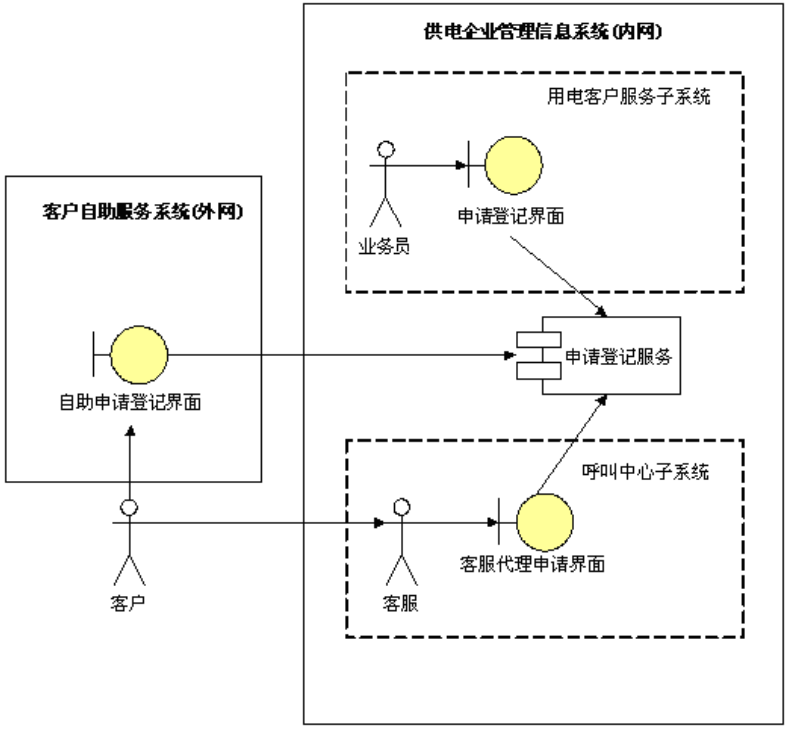


图11-37 申请登记服务组件运行环境

11.7 部署模型

- 又称实施模型
- 作用：定义构成应用程序的各个部分在物理结构上的安装和部署位置。
- 其中，物理结构包括客户机、服务器、网络节点、移动设备等所有可能的程序逻辑处理设备和文件存放设备。

例：供电企业管理系统

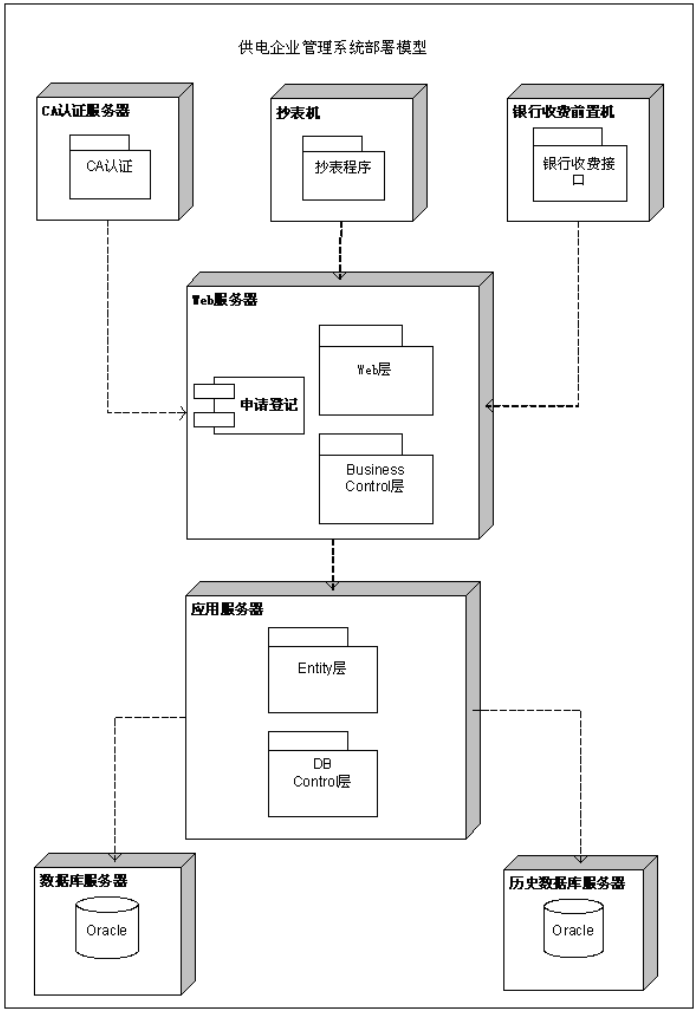


图11-48 供电企业管理系统部署模型

- 基于Web的应用程序，采用Oracle数据库
  ------Web服务器，数据库服务器
- BusinessControl层、Entity层、DBControl层。
- 银行代收电费，需要银行收费接口，这个接口的安全要求、通信协议与其他业务不同。考虑增加一台收费前置机部署银行收费接口
- 呼叫中心系统申请登记，需要数字证书认证才能操作系统，因此添加CA认证服务器
- 历史数据备份、管理和查询，增加历史数据库服务器。
- 为抄表机部署硬件，便于抄表示数导入导出到管理系统。

本章小结

- 11.1、确定系统用况
- 11.2、分析业务规则
- 11.3、用况实现
- 11.4、软件架构和框架
- 11.5、分析模型
- 11.6、组件模型
- 11.7、部署模型

谢 谢！