第4章

系统分析

4.1 项目导引

经过了几周的需求确定，项目组进入到了设计的环节。项目组组长老李组织大家开了个阶段性会议，总结了上个阶段的工作。“我们对招聘系统通过构建用例模型，深入了解了用户的系统目标和主要解决的问题。”老李说：“为了保证系统有一定次序地进行开发，根据优先级制定的三个原则（高业务价值、具有架构意义、高风险）确定了系统开发的增量，为我们后面顺利地进行系统的设计奠定了一个基础。但是，用例模型中的描述是从用户使用系统的角度来描述的，也就是从系统外部来看待系统的一系列响应，而我们最终需要的是系统内部代码的组织形式。大家想想看，我们还缺少什么东西呢？”

小李接着话题说：“我们现在需要完成的系统打算采用的是 Java 这种面向对象的语言。我们在构建系统的时候，应该依据面向对象的思想来组织我们的代码。”

“如果采用面向对象的语言，类应该是代码构建的核心，类的交互完成了系统的目标和服务。可是，类是从哪里来呢？不应该是拍脑袋想出来的吧，而且类的交互又是依据什么完成的呢？”小张有些疑惑地问。老李微笑着点头表示同意。

小张的问题引发了项目小组其他成员的热烈的讨论。组长在白板上画出了目前项目组所具有的资源和期望达到的目标，如图 4-1 所示。

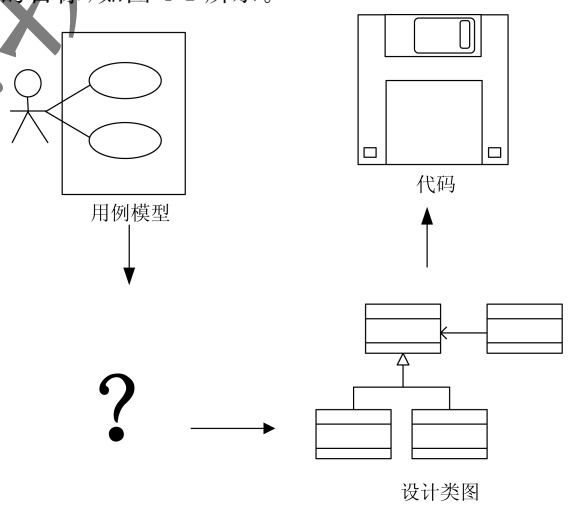

图 4-1　如何通过用例模型与代码建立起联系

4.2　项目分析

类的产生并不是我们凭空想象出来的。在面向对象思想中，系统中应该使用现实世界的隐喻来帮助我们构建系统，以达到系统代码易于理解的目的。那么，如何将现实世界人们对事物的认知转换成抽象的，系统内部类的表现形式呢？构建领域模型是我们首次以对象的视角来看待现实世界中对象之间关系的第一步。

通过领域模型我们了解了对象之间的关系，而系统需要通过对象之间的交互，消息的传递才能够相互协调起来完成相应的系统功能，此时需要使用动态模型对对象之间的交互进行描述。在 UML 中我们可以通过顺序图来描述对象之间的交互，但是为了能够更顺利地了解顺序图中需要明确的系统内部类，都有哪些职责需要进行分配，我们需要使用健壮性分析来帮助我们达到这个目的。

总结起来，分析阶段的目标是能够以面向对象的视角描述我们要构建的系统环境，同时能够初步了解系统内部的组成形式及运作方式。在此阶段，我们将会得到静态类的初步表示，系统内部动态模型的初步构建，为后续的详细设计做好准备，请参见图 4-2。

本阶段的任务归纳起来总共有 3 项：

(1)构建领域模型。抽象现实世界中的对象之间的关系，并将这种隐喻映射到系统内部的类之间的关系上。

(2)完成健壮性分析。帮助软件开发人员从系统外部的响应转换成系统内部各对象之间的交互与协作，挖掘和完善系统内部类的组成。

(3)进行交互建模：根据健壮性分析的结果，通过构建顺序图为系统对象分配职责，构建系统内部类的初步设计。

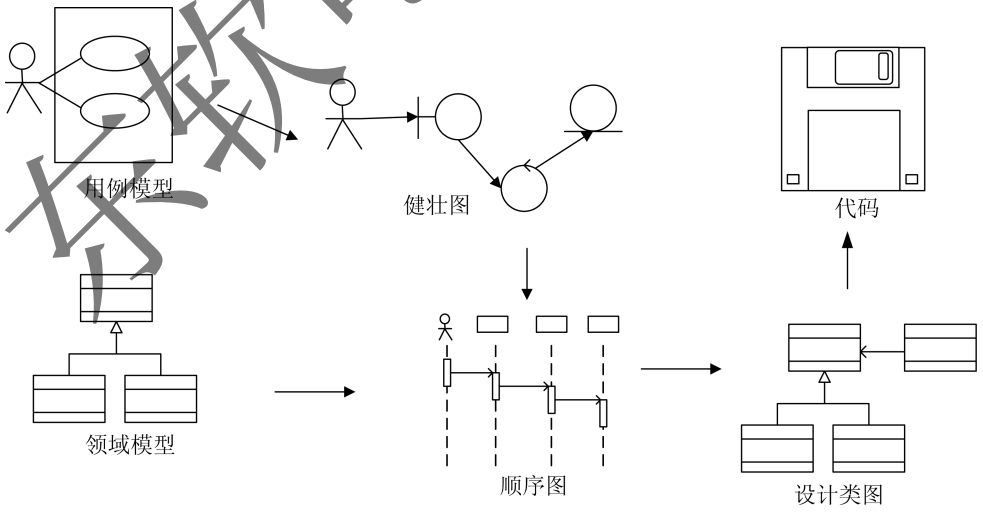

图 4-2　各阶段之间的演变关系

4.3 领域模型

从面向对象基本概念的描述中我们知道，对象及对象之间的关系提取是基于对真实世界的认知。它模拟了人们在真实世界中对领域或业务的理解和认识，阐述了其中的重要概念，它以对象的形式进行描述，从而构建出静态模型的基础。它描述了领域中的重要概念，并且为设计系统内部对象提供灵感。领域模型是面向对象分析中最重要的和经典的模型。它将用例模型中的描述和业务规则作为构建领域模型的输入。而该模型的创建又会对词汇表、系统需要提供的服务、软件的设计模型中的对象产生影响。

4.3.1 什么是领域模型

领域模型是一个商业建模范畴的概念，它和软件开发没有任何关系。领域模型是对行业背景下的领域内概念或现实世界中对象的可视化表示，它强调了概念和这些概念之间的关系。

即便一个企业不开发软件，它也会具备自己的业务模型，所有的同行业企业的业务模型必定有着非常大的共性和内在的规律性。由这个行业内的各个企业的业务模型再抽象出来整个行业的业务模型，这个模型即“领域模型”。领域模型可以用 UML 中的类图进行表示，图 4-3 展示了银行领域的部分领域模型。

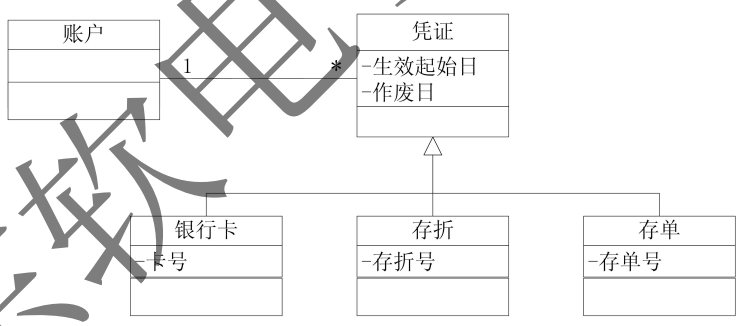

图 4-3 银行领域模型的凭证相关部分

我们通过这幅 UML 类图中的抽象能够了解到银行领域中和凭证相关的部分领域知识。任何一个银行中储户的“账户”可能有会有一种或多种“凭证”相关，例如我们手中可能会同时持有建行的存折和银行卡，或是可以通过存单的形式向账户中存钱；无论是哪种凭证都必须有生效起始日和作废日；然而，不同种类的凭证其凭证号确实不尽相同的，例如银行卡的编号规则就有别于存折的编号规则。

这个模型虽小，却涵盖了银行一些实际的业务情况。从这个例子可以看出，领域模型是对现实世界中实际问题领域的抽象表示，它专注于分析问题领域本身，发掘重要的业务领域概念，并建立业务领域概念之间的关系。

领域模型是面向对象分析中最重要的和经典的模型，它阐述了领域中的重要概念。领域模型可以作为设计某些软件对象的灵感来源，可以作为我们对系统深入理解的一种渠道。我们构建领域模型的范围限定于当前所要完成项目范围内的用例场景，它能够被不断地改进，用以展示相关的重要概念。相关的用例概念和专家的观点将作为创建领域模型的输入。反过来，该模型又会影响系统提供服务的特性、词汇表和设计模型，尤其是设计模型中领域层的软件对象。

那么，领域模型中所描述的概念最后是否需要我们将其转化成数据库中需要持久化保存的对象呢，也就是后面我们将要提到的数据模型。此外，领域模型和数据模型是一回事吗？

首先明确一点的是，领域模型不是数据模型。在领域模型中，并不会排除需求中没有明确要求记录其相关信息的类(这是对关系数据库进行数据建模的常见标准，但与领域模型无关)，也不会排除没有属性的概念类。也就是说，没有属性的概念类是合法的，或者在领域内充当单纯行为角色而不是信息角色的概念类也是有效的。数据模型的实体对象虽然也取材于真实世界，但是它通过对数据模型的定义，来表示存储于某处的持久性数据。

4.3.2　如何构建领域模型

领域模型是面向对象可视化模型中(UML 模型)静态部分的基础。建立领域模型时，首先确定真实世界中的抽象，即系统中将涉及到的主要概念性对象。设计面向对象的软件时，应根据这些实际的问题空间对象设计软件的结构。这里的意思是同软件需求相比，真实世界发生变化的频率更低。这些问题域抽象的模型是整个对象建模工作(尤其是静态模型部分)的基础。

以当前迭代(当前的项目范围)中的需求为界，创建领域模型必须要经过下面四个步骤：

(1)寻找(识别)类。

(2)筛选类。

(3)确定关系。

(4)识别类的属性。

1. 类的识别

领域对象类的最佳来源很可能是用户的高级问题陈述、低级需求、用例描述和客户业务所处背景下的专业知识。要发现领域对象类，就要从这些来源中挖掘尽可能多的相关陈述，然后根据一定的策略寻找概念类。目前常用的方式是确定名词短语，还有一些其他的方式，请参见知识扩展部分的内容。

确定名词短语的方法是一种简单易行的方式，它主要是在对领域的文本性描述中识别名词和名词短语，将其作为候选的概念类或属性。这是一种简单而有效的语言分析技术。非正式形式的用例是挖掘名词短语的一个重要来源。

根据表 4-1 所示的项目用例描述，大家可以先找出其间的名词或名词短语。

表 4-1 非正式形式的“申请职位”用例

用例 UC05:申请职位

基本流程:

1. 应聘者在{应聘者主页面}点击[申请链接]或者{职位详细信息页面}。

2. 系统显示{申请职位页面}。

3. 应聘者为申请的职位填写履历信息,提交申请请求。

4. 系统确定应聘者填写的履历信息的有效性。

5. 系统确定该应聘者没有申请过当前职位。

6. 系统保存应聘信息。

7. 系统将应聘者的状态从还未申请状态修改为还未面试状态。

8. 系统重新显示{应聘者主页面}。

分支流程:

4.1 如果应聘者曾经应聘过该职位,但还没有被取消资格,即没有设定为取消状态,那么系统将会拒绝该应聘者的请求,并给出提示“您已经申请了该职位,请耐心等待!”

*注意:您在申请了一个职位之后必须等待该职位的处理结束后才能够应聘其他职位。这意味着你在同一时间内只能申请一个职位。

4.2 如果应聘者曾经应聘过该职位,并且没有通过面试,那么系统将会拒绝该应聘者的请求,并给出提示“您目前还不适合当前职位,请选择其他职位!”

通过提取出这段用例描述中的名词或名词短语,形成初步类的候选对象,得到如表 4-2 所示的名词。

这种方法的弱点是自然语言的不确定性。不同名词短语可能表示同一概念类或属性,此外可能存在歧义。因此,建议与知识扩展部分介绍的“概念类分类列表”一起配合使用。

2. 应用筛选原则

在表 4-2 中得到的一些名词短语是候选的概念类,有一些可能只是概念类的一些属性。在完成名词的初步列表之后,我们应用一些简单的筛选规则进行精简,将类的范围进行调整。

表 4-2 得到的初步概念类的候选对象

类的候选对象	
应聘者	应聘者状态
职位	还未申请状态
系统	还未面试状态
应聘信息	履历
申请请求	职位详细信息

下面是常用的几种筛选原则:

(1)冗余。

表示相同事物的两个名词就是冗余。

例如,“履历”和“应聘信息”实际上都指的是针对特定职位需要应聘者填写的信息,因此选择简洁的“履历”作为概念类;“职位”和“职位详细信息”也都是指的职位相关的信息,选择“职位”作为概念类。

(2)不相关。

名词与当前研究的问题域没有关系。它也可能是有效类,但不在当前项目的范围之内。

例如,“员工考绩标准”是个名词,但订单处理系统不会测量或跟踪员工的工作实绩;电话和传真不是系统所关注的内容。“系统”是我们常见的名词,但是系统指的是实际的计算机软件,并不是我们当前研究的业务范畴,因此,在领域模型中不应出现“系统”这个名词。

(3)属性。

实际上描述了另一个类结构特征的名词是属性。属性往往是一些简单的文字性描述或是数字,这些内容必须依附于某个对象才能够体现出其自身的价值。例如,“书名”、“作者”、“译者”描述的是“图书”类的一个组成部分。书名作为属性描述了特定图书的一个特征,如果没有这种依附关系,书名自身是没有任何意义的。

然而类和属性的识别,还与具体的应用领域相关。例如,“邮政编码”一般是“地址”类的一个属性,但对于邮政服务,邮政编码就是一个类,因为它同时包含了属性(地理位置、统计、费率结构和运送信息)和行为(投送路线和日程)。

在图 4-4 列出的候选类列表中,应聘者的状态实际上是对“还未申请状态”、“还未面试状态”的统称,是描述应聘者当前应聘所处阶段的一种描述,因此应聘者的状态,简称应聘状态应作为应聘者的属性。

(4)操作。

描述某个类职责的名词自身不是一个类,而是一个操作。例如,“申请请求”。但需要注意的是,如果在操作过程中需要使用到一些信息的记录,那么这种操作可能需要通过这种信息记录,以关联类的形式保留下来。例如,在借阅图书的时候,图书管理员需要通过借阅记录来保存读者的借阅时间等信息。那么,借阅这个动作就通过借阅记录的形式保留下来。

(5)角色或状态。

描述一个特定实体的状态或其分类的名词多半不是一个类。例如,“还未申请状态”是一个应聘者在一定时间下的特定应聘状态。

(6)时间。

描述特定时间频率的名词,通常表示了领域必须支持的一个动态元素。例如,“每星期打印一次发票”中的“星期”就不是候选类。

(7)实现结构。

描述硬件元素或算法的名词最好是删除或指派为某个类的操作。例如,“打印机”和“傅立叶算法”。注意,作为一个类选择最终名字的时候,一定要使用明确而简洁的名词,且单数优于复数。

初始的样例项目候选列表应用筛选原则的过程如表 4-3 所示。

表 4-3　　应用筛选原则筛选候选类的过程

概念类	属性	冗余	不相关	操作	角色或状态
应聘者	应聘状态	应聘信息	系统	申请请求	还未申请状态
职位		职位详细信息			还未面试状态
履历					

3. 关系

获得了概念类列表之后，我们需要建立类之间的关系，这些关系能够帮助我们理解领域中的业务常识。我们说领域模型是静态模型，但也能表示出工作的活动片段，这就是关系所起到的作用。关联就是类(或类的实例)之间的关系，表示有意义和值得关注的连接。

关联表示的这种值得关注的关系是需要持续一段时间的，它不是数据流、数据库外键联系、实例变量或软件方案中的对象连接的操作语句。关联声明的是针对现实领域从纯概念角度看实际存在的有意义的关系。如果存在需要保持一段时间的关系，就将这种语义表示为关联。

建立关联可以从用例中找到显式的关联，需要注意的是应该避免加入大量的关联。构建关联的目的是能够从模型中简单明了地了解概念之间的关系，而如果在领域模型中加入太多的关联，这样会使图形显得很混乱，影响了对图的理解，背离了构建模型的初衷。所以应该慎重地添加关联线，只记录那些需要记住的关联。

我们在添加关联时，需要尽量避免出现下述的情形：

(1)立即给关联制定多重度，确保每个关联都有明确的多重度。

(2)不对用例和时序图进行研究，就将操作分配给类。

(3)在确保已满足用户需求之前，就开始对类结构进行优化以提高重用性。

(4)对于每个“……部分(part-of)”关联，就因使用聚集还是组合而争论不休。

(5)未对问题空间进行建模之前，就假定一种具体的建模策略。

(6)在领域类和关系型数据库表之间建立一对一的映射。

(7)过早地执行“模式化”，根据与用户问题毫无关系的模式创建解决方案。

针对招聘管理系统中的描述，我们关注的是应聘者申请职位的过程。在这个过程中，应聘者需要提交履历以完成申请职位的过程。因此，通过图 4-4 展示出当前用例所关注的对象之间的关系。

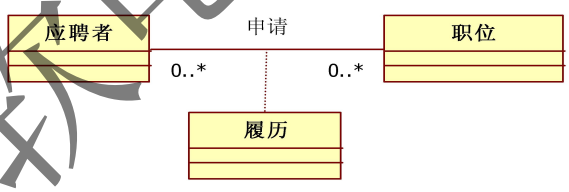

图 4-4　申请职位用例的初步领域模型

4. 识别属性

前面我们已经通过几种途径获得了初步的领域模型，确定了基本的对象范围和它们之间的关系。但是领域模型的内容还不够充实，对于对象所具有的属性特征我们还没有加以关注，下面我们就来看一看属性的识别过程。

(1)当需求中建议或暗示需要记住信息时，引入属性。

(2)获取属性的渠道。

查看用例文档，寻找事件流中的名词；

查看需求文档，发现系统要搜集的信息；

在需求确认的过程中所构建的界面原型中出现的界面信息；

若已经定义了数据库结构，则数据库表中的字段就是属性。

(3)选择属性时应考虑的因素。

只有系统感兴趣的特征才包含在类的属性中,分析系统建模的目的,也会影响属性的选取。

(4)每条属性都能够回溯到用户的需求,不要盲目添加不必要的属性,造成系统混乱。

(5)类的属性要适当。若某个类的属性太多,则可考虑分解成更小的类;若某个类的属性太少,可考虑将类进行合并。

根据表 4-3 应用筛选原则筛选出候选类,同时参考图 4-5 界面原型中的界面信息,将分析出来类的属性分别添加到对应的类中,形成如图 4-6 所示添加了属性的部分示例项目的领域模型。

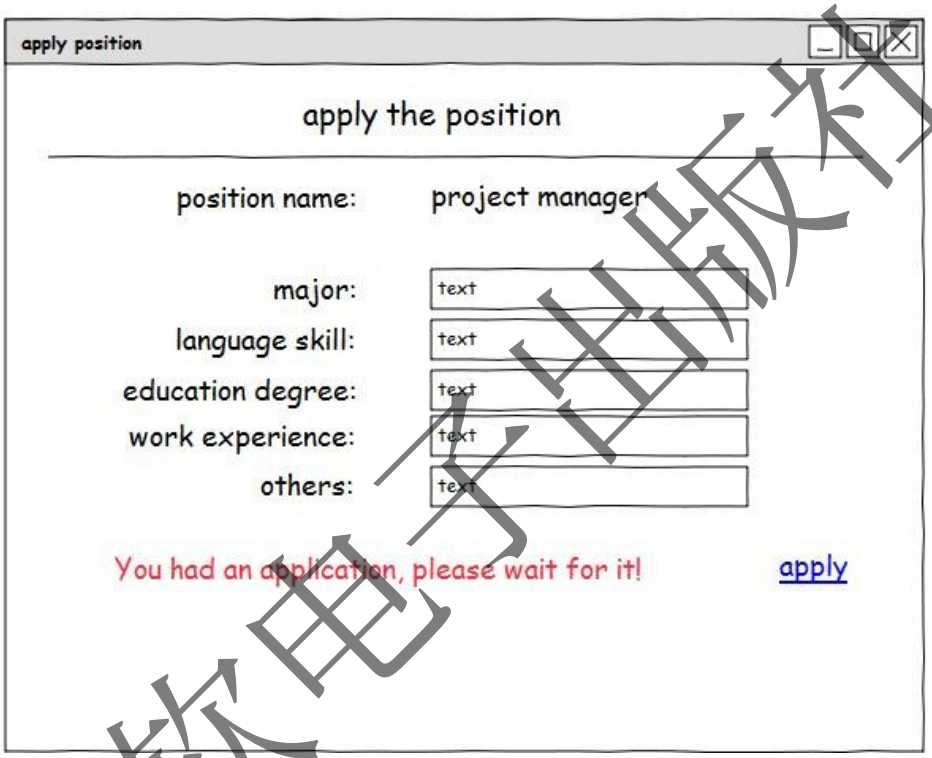

图 4-5　界面原型

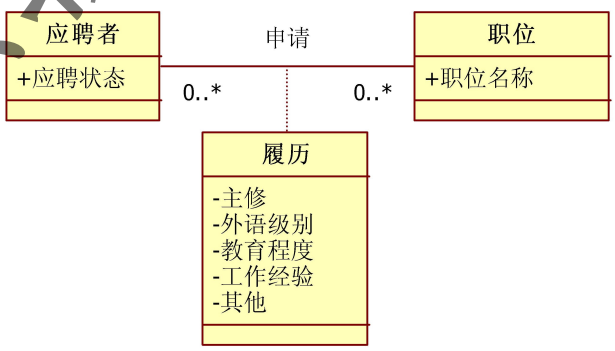

图 4-6　添加了属性的申请职位用例的领域模型

4.3.3 何时构建领域模型

领域模型的构建可以在进行需求确定过程的后期介入，特别是在进行基本用例模型的构建，了解大部分业务之后。此时通过构建领域模型，一方面帮助我们摸清业务领域中业务对象之间的关系，明确一些规则，另一方面也有助于对用例描述中所使用到的业务术语进行统一和整理。

领域模型描述真实世界的类(对象)以及它们之间的相互关系，展示了系统的静态模型和一部分的工作活动。在分析过程中，类模型的优先级要高于状态和交互模型，这是因为静态结构容易更好地定义，而且会较少地依赖应用程序的细节，当系统的解决方案发生变化时，会更加稳定。领域模型的信息来源于问题陈述、其他的相关系统制品(用例模型)、专家对应用领域的了解以及对真实世界的总体认识。应确定是否已经考虑了所有的可用信息，而没有只依赖于单个的信息来源。

领域模型描述的形式也可以采用纯文本方式(业务术语表)表示。但是采用可视化语言来描述更容易理解这些术语，特别是它们之间的关系。因为人们的思维更擅长理解形象的元素和线条的连接，因此，领域模型我们采用 UML 中类图进行表示。

4.4 健壮性分析

健壮性分析可以看成是从用例模型中分析出基本的(对象)类，并描述系统内部各部分是如何响应用户的请求的，从而开始了对系统内部实现结构的初步设计。健壮图可以看成是描述对象之间的协作图，关注各种构造定型的类之间是如何协同工作的。在健壮图的基础之上，我们就可以很顺利地完成顺序图的绘制，从而进入到了系统设计的更深入的阶段。参见图 4-7。

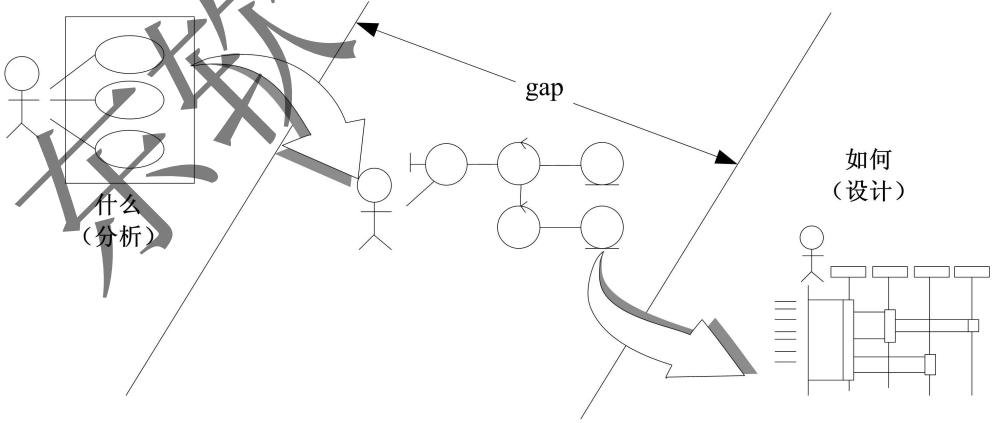

图 4-7 健壮性分析是跨越“分析”到“设计”的桥梁

通过健壮性分析，我们可以改进用例文本和静态模型：

1. 确保用例文本的正确性

通过对用例文本的分析，并将其转化为健壮图的过程中，可能会发现用例文本描述过程中的一些不合理的地方。

例如，只描述了用户的动作，却忽略了系统的相应，那么在绘图过程中就会看到只有用户对边界对象进行操作，而没有控制对象和实体对象的参与，从而发现用例文本的缺陷。

使用例文本的特性从纯粹的用户手册角度变为对象模型在上下文中的使用描述。

2. 方便绘制时序图

由于在健壮图绘制过程中已经将处理过程中所使用的对象基本分析出来了，因此，在绘制时序图时将是对健壮图的一种更细致的描述。同时，通过健壮图的检查机制，确保了时序图绘制的顺利进行。

3. 帮助发现对象

在领域建模期间肯定会遗漏一些对象，在进行健壮性分析的过程中会帮助我们捋清思路，甚至发现对象命名矛盾或冲突的情况。

4. 进入初步设计阶段

由于在健壮性分析的过程中涉及到了系统内部的类(构造定型)，因此开始了对系统内部结构的初步设计。它可以帮助我们改进用例文本和静态模型(领域模型)。

4.4.1　健壮图的表示法

在健壮性分析过程中，系统中的类将被划分成三种构造定型之一：实体类(对象)、边界类(对象)或控制类(对象)，如图 4-8 所示。

图 4-8　健壮性分析中使用的图形

(1)边界对象。

参与者使用边界对象来同系统交互。通常，边界对象用作屏蔽和媒介，隔离了如何取得应用程序提供的服务的大部分交互细节，位于系统与外界的交界处，包括所有的窗体、报表、系统硬件接口、与其他系统的接口。识别边界类的简单途径就是注意系统中的参与者，每个参与者都需要与系统建立接口。

(2)实体对象。

实体对象(类)是通常是来自领域模型的对象。它们用来保存持久性的应用程序实体的有关信息，提供用于驱动应用程序中大多数交互所需的服务。实体对象通常提供一些非常具体的服务：

①存储和检索实体属性。

②创建和删除实体。

③提供随着实体的改变而有可能改变的行为。

(3)控制对象。

控制对象(类)对应用领域中的活动进行协调，即将边界对象和实体对象关联起来。每一个用例中通常有一个控制类，它控制用例中的事件顺序。

通常，控制类可以扮演以下几种角色：

①与事物相关的行为。

②特定于一个或少量用例(或用例中的路径)的一个控制序列。

③将实体对象与边界对象分离的服务。

4.4.2 健壮图的使用规则

绘制健壮图时，仔细检查用例文本，每次检查一个句子，并绘制参与者、边界对象、实体对象和控制器以及图中不同元素之间的关系。我们应该能在一个图中指出基本流程和所有的分支流程。在使用过程中需要遵循下述几条规则，参见图 4-9 所示。

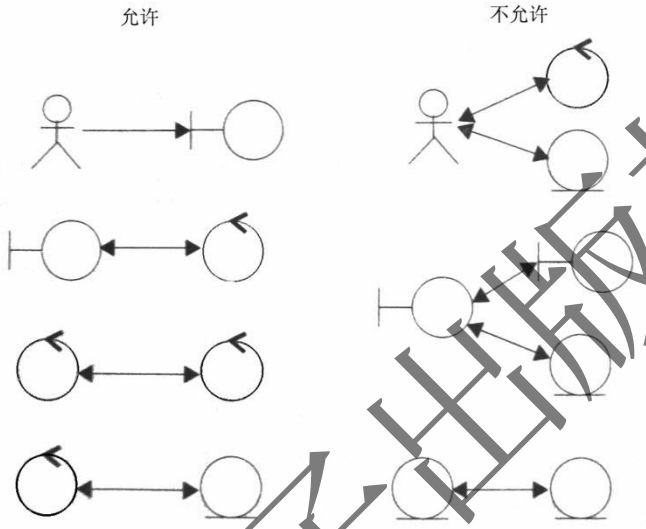

图 4-9 健壮图的使用规则

(1)参与者只同边界对象交互。

(2)边界对象只能同控制器和参与者交互。

(3)实体对象只能同控制器交互。

(4)控制器可同边界对象、实体对象以及其他控制器交互，但不能同参与者进行交互。

需要注意的是，边界对象和实体对象都是名词，而控制器是动词。名词和名词之间不能互动，但动词可同名词或动词进行交互。确保每一个用例，对应一个健壮图。

下面以登录用例为例，介绍一下如何进行健壮性分析，用例描述参见表 4-4。

表 4-4 登录用例的非正式形式

用例 UC02:登录 基本流程： 1.用户在{登录页面}输入用户名密码并提交。 2.系统检验该用户为系统有效用户。 3.系统记录入口事件日志。 4.系统认定用户的身份是应聘者，系统显示{应聘主页面}。 分支流程： 1.1 如果用户点击{登录页面}上的[注册链接]，则系统转入注册用例。 2.1 如果用户名正确但密码不正确，系统再次显示{登录页面} 。 2.2 如果用户名不存在，系统将显示 {注册页面}。 4.1 系统认定用户身份是招聘人员，系统显示{招聘主页面}。 4.2 系统认定用户身份是系统管理员，系统显示{系统管理员主页面}。 4.3 系统认定用户身份是 HR，系统显示{人力资源主页面}。

第一步，需要针对用例描述中的每一句话进行分析，确保其中描述的内容出现在健壮图中。

针对于“登录”用例基本路径 1 的内容来说，用户发起动作，用户作为一个参与者需要进行描述。用户必须通过边界对象与系统进行交互，这就是“登录页面”，通过动作“点击”确认信息的提交。注意，这里的“点击”虽然是动词，但它只是用户的一个动作，而不是一个系统处理，因此不能将“点击”看作是一个控制对象。参见图 4-10 绘制健壮图步骤一。

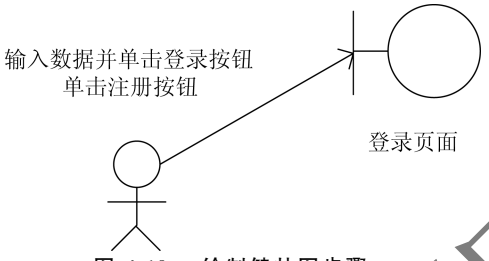

图 4-10　绘制健壮图步骤一

第二步，基本路径 2 开始描述系统对用户发起请求的响应。信息通过边界对象“登录页面”向系统内部进行传递，系统需要对传入的信息进行检验，以保证登录用户的合法性。检验是一个动作，因此由控制对象“检验用户合法性”来表示。在进行检验时需要由两方面提供数据，一方面是用户通过界面传递进来的待验证数据，另一方面是系统内部已经存在的合法数据。这时需要查看领域模型中是否有对象能够表示用户的注册信息。针对于前面获得的局部领域模型中“应聘者”对象描述了应聘者的基本信息。如果是系统的合法用户就必须注册有用户名和密码，因此在原有的领域模型中需要在类对象应聘者中增加用户名和密码的属性描述，因此形成修改后的静态模型，参见图 4-11。在健壮性分析中我们使用系统用户信息作为实体对象记录和保存用户信息。根据这些信息绘制图 4-12。

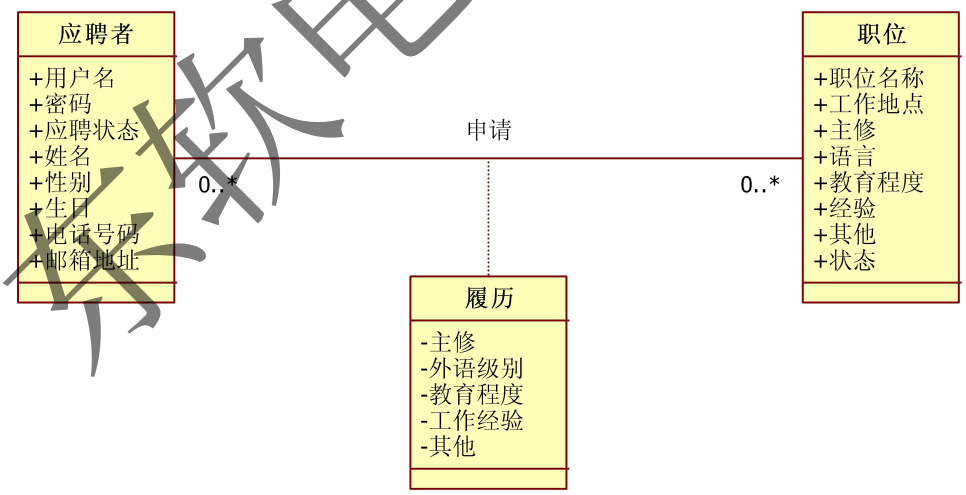

图 4-11　调整过的静态类图

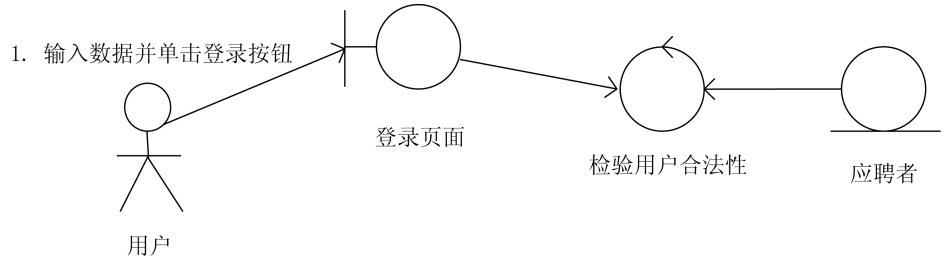

图 4-12　绘制健壮图步骤二

基本路径 3 中系统在验证完用户合法性之后需要将系统入口的事件记录在事件日志中。首先需要有个控制对象来记录动作“记录入口事件”，同时，还需要一个实体对象来对事件日志进行保存，绘制后的结果参见健壮图 4-13。

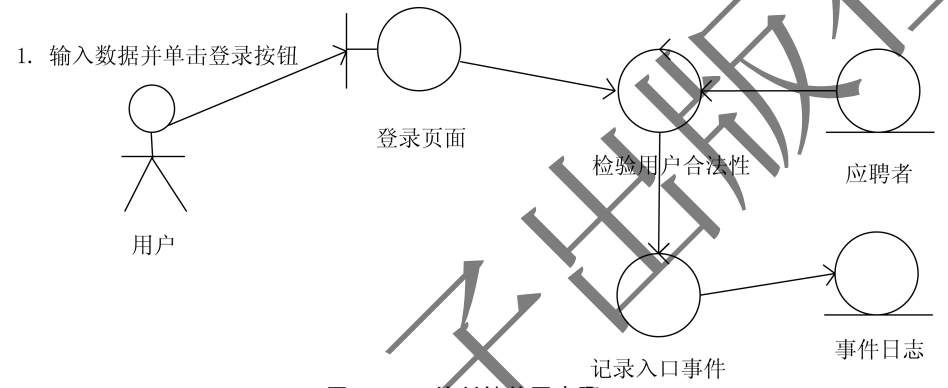

图 4-13　绘制健壮图步骤三

紧接着用例中基本路径 4 描述了系统根据验证结果，将信息反馈给用户。系统要求控制对象“显示”逻辑根据登录用户的不同身份显示相应的边界对象“欢迎界面”。到此，根据基本路径进行的健壮性分析将一个参与者，两个边界对象，三个控制对象和两个实体对象分析出来。参见图 4-14 绘制健壮图步骤四。

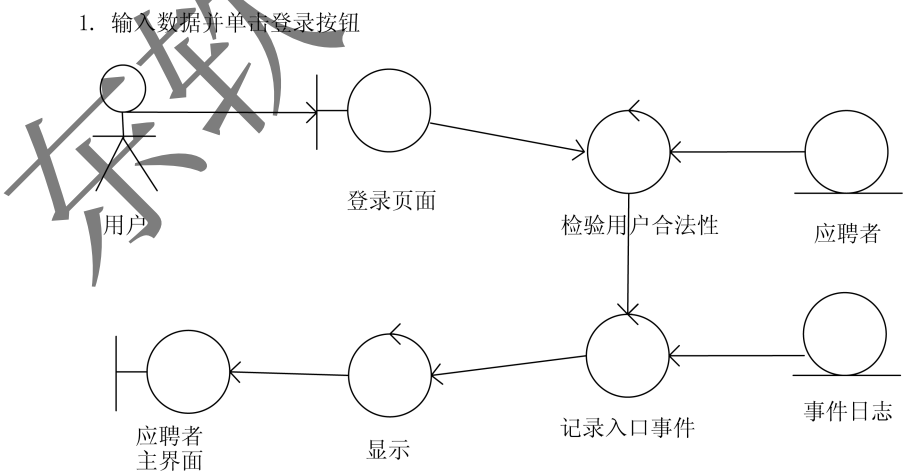

图 4-14　绘制健壮图步骤四

为了保证系统分析的全面性，分支流程中的内容也需要根据上述方法进行健壮性分析并添加到健壮图中。同时，如果发现用例的描述不符合分析的结果，应该及时修改用例中的描述。

例如，在添加 2.2 分支流程时，发现有可能用户在录入界面信息时就存在输入错误，像用户名或密码输入为空的情况，系统也应该进行相应的处理，那么就需要修改完用例描述后，在健壮图中体现出来。修改后的用例描述参见表 4-5。

表 4-5　　登录用例的非正式形式

用例 UC02：登录
基本流程：
1. 用户在{登录页面}输入用户名密码并提交。
2. 系统检验用户输入信息的有效性。
3. 系统检验该用户为系统有效用户。
4. 系统记录入口事件日志。
5. 系统认定用户的身份是应聘者，系统显示{应聘主页面}。
分支流程：
1.1 如果用户点击{登录页面}上的[注册链接]，则系统转入注册用例。
2.1 如果用户名或密码为空，则系统在{登录页面}上显示该信息。
3.1 如果用户名正确但密码不正确，系统再次显示{登录页面}。
3.2 如果用户名不存在，系统将显示 {注册页面}。
5.1 系统认定用户身份是招聘人员，系统显示{招聘主页面}。
5.2 系统认定用户身份是系统管理员，系统显示{系统管理员主页面}。
5.3 系统认定用户身份是 HR，系统显示{人力资源主页面}。

最后，我们得到如图 4-15 所示的一幅完整的健壮图。

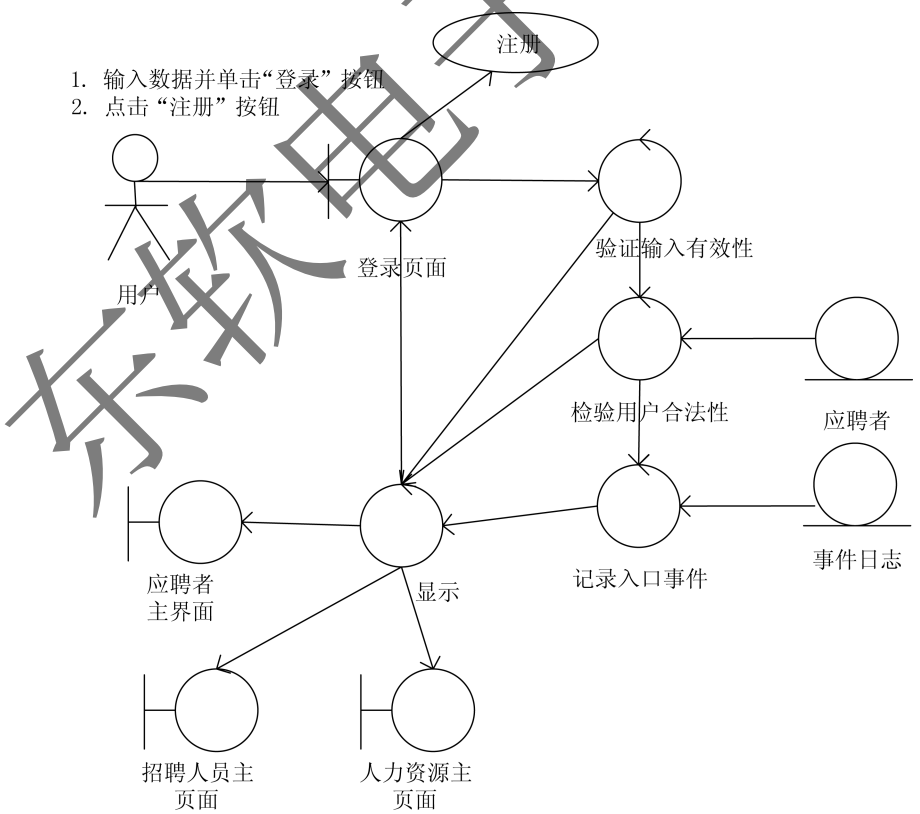

图 4-15　“登录”用例的完整健壮图

在进行健壮性分析的过程中可能发现用例文本存在模糊的地方，需要重新编写用例文本，并明确地引用边界对象和实体对象。大多数人的第一稿用例文本都会存在遗漏的地方。除了使用健壮性分析的结果来改进用例文本外，还应不断地改进静态模型。应将新发现的对象和属性加入到静态类图中。

4.5 顺序图的转换

4.5.1 将健壮性分析与顺序图对应

健壮性分析旨在发现对象，而顺序图则主要关注行为的分配——将确定的软件函数分配给发现的一组对象。顺序图的构建是与用例一一对应的。它是为了识别设计类或子系统，其实例需要去执行用例的事件流。通过顺序图把用例的行为分布到有交互作用的设计对象或所参与的子系统。同时，顺序图定义对设计对象或子系统及其接口的操作需求，为用例捕获实现性需求。注意，除非在两个类之间定义了关联，否则这两个类的对象之间是无法发送消息的。如果一个用例路径需要在两个对象之间通信，而对应的两个类之间不存在相应的关联，那么类图就是不正确的。

延续上面健壮性分析的例子，将健壮图转换为初步的顺序图。

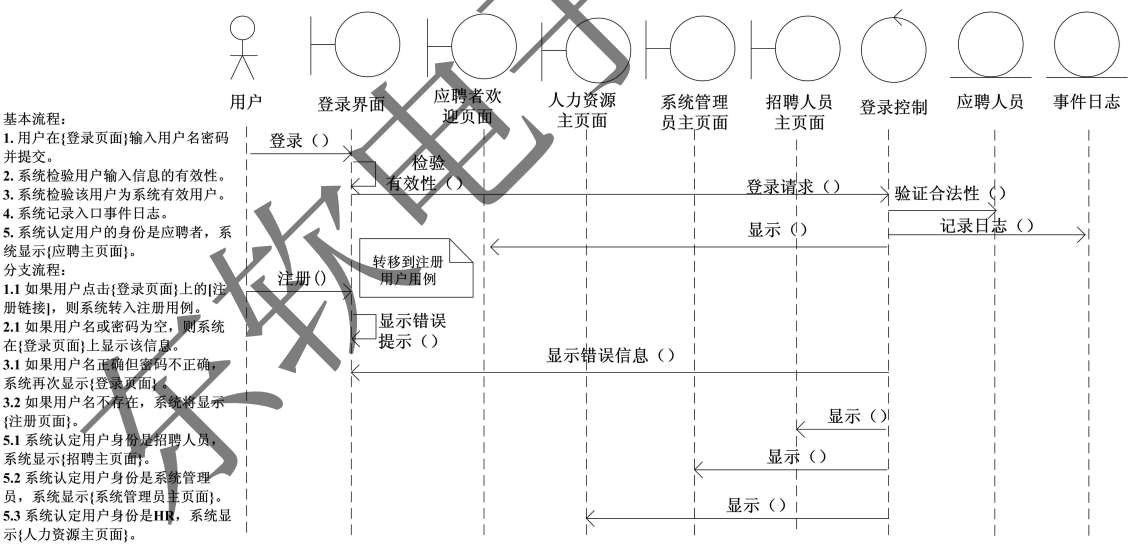

图 4-16 根据用例场景绘制的顺序图

在图 4-16 中，我们会注意到顺序图的左边出现了对应的用例文本。这样做会可以使我们总能看到要求的系统行为，并不断提示我们需要做什么。我们可以将分析出来的方法增加到对应的静态类图中。

在分析阶段的顺序图只是对初步分析出来的系统对象进行行为的分配，将顺对象的基本职责，还没有体现体系结构设计结果和设计类的内容，指导编程人员进行代码编写的详细顺序图将在对象设计之后完成。图 4-17 所示为运用了执行体系结构后的顺序图。在此图的基础之上，程序的执行框架就较为清晰了。从这里也可以看出来，面向对象分析设计方法在应用了 UML 之后，

设计的内容是逐步细化的一个过程。并不是所有的用例都需要在详细设计环节绘制对应的顺序图，为了能够体现出体系结构设计的结果，需要选取具有代表性的用例进行详细地实现描述。

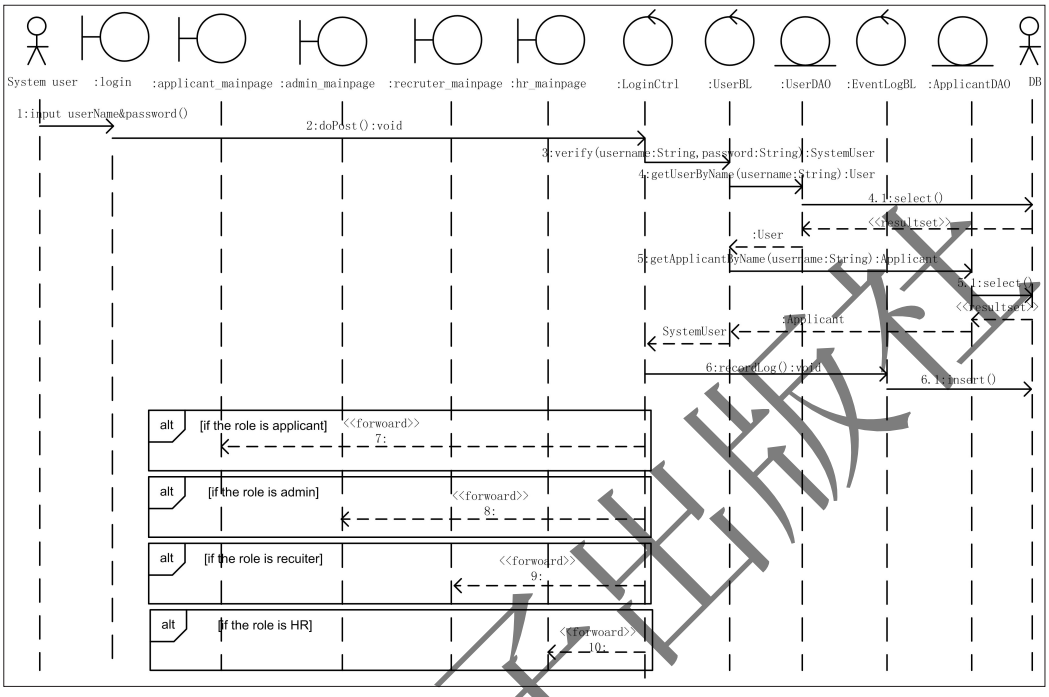

图 4-17　应用了执行体系框架的顺序图

4.5.2　为静态类图增加方法

我们获得了顺序图，对系统的动作进行了分配，那么对应的类就应该担负起自己的职责，这就是类所具有方法了。

我们知道在面向对象系统中，目标任务的完成是依赖于对象之间的消息发送。消息传递给某个对象就意味着接受消息的对象应该提供相应的响应。以图 4-18 中的例子来说，对象 A 向对象 B 发送消息 getB1()，对象 B 接收到消息后完成相应的任务，也就是 B 本身的职责，同时在完成任务的过程中向自己发送消息 calculate()，由于消息的接收者是对象 B 本身，因此，这个职责也是由对象 B 完成。在执行完 getB1()任务后向对象 A 返回处理结果。此时对象 A 又向对象 B 发送了 saveB2()的请求，对象 B 接受请求并完成相应的任务。按照前面的分析，对象 A、B 对应的静态类图展示了它们的类的职责及关系，参见图 4-19 所示。

在静态类图中我们会发现，getB1()和 saveB2()两个方法是公开方法，而 calculate()方法是私有的。这是因为对象 B 在调用 calculate()时，只需要自己知道 calculate 的存在，不需要向外界公布，因此设置成私有类型以保证信息的隐藏性。

那么，针对于招聘管理系统就可以整理出初步的静态类图了，如图 4-20 所示。

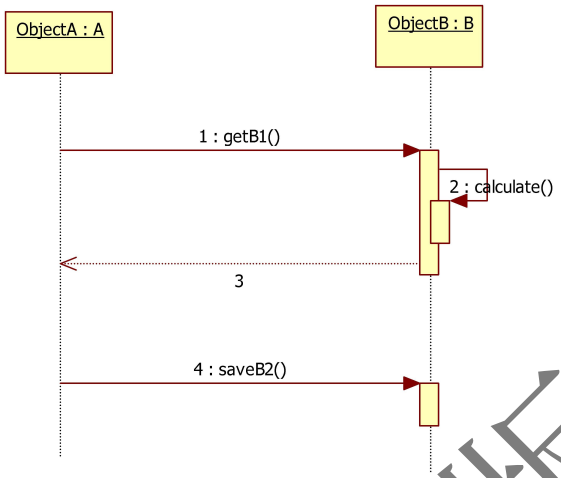

图 4-18　对象 A 与对象 B 之间的消息传递

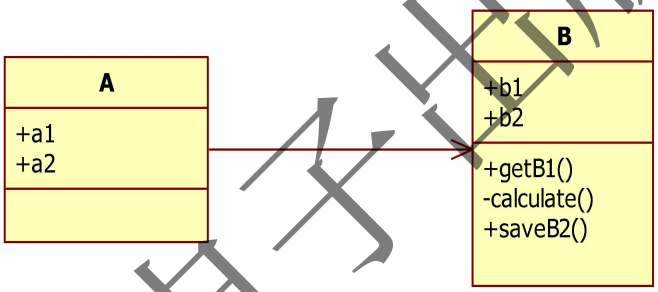

图 4-19　对象 A、B 对应的静态类图

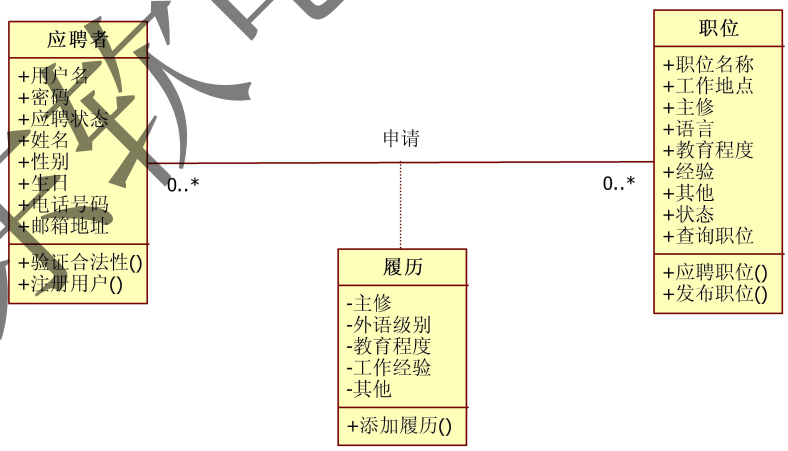

图 4-20　初步静态类图

4.6　状态的标识

状态是对象属性值的抽象。对象的属性值按照影响对象显著行为的性质将其归并到一个状态中去。状态指明了对象对输入事件的响应。状态图是对一个类的生命周期进行建模，它追踪了一个对象从诞生到消亡的全过程。当正在建模的类呈现出值得关注的和复杂的动态行为时，状态图才是有价值的。

状态图反映了状态与事件的关系。当接收一事件时，下一状态就取决于当前状态和所接收的事件，由该事件引起的状态变化称为转换。状态图中用结点表示状态，结点用圆角矩形表示；圆角矩形内有状态名，用箭头连线表示状态的转换，上面标记事件名，箭头方向表示转换的方向。

一般情况下状态建模采用下述步骤：

(1)识别状态。

(2)选择使用该类的任意用例的主路径。

(3)该路径的上下文环境加入到状态图中。

(4)选择同一个用例中的另一条路径或一个不同的用例，直至无法继续获取信息为止。

许多事件包括动作和活动，最终都会导致对类操作进行建模，许多操作都是私有的。向其他对象发送的消息都会导致在目标类中定义一个操作，任何识别出来的状态变量都会成为所建模类的成员变量。

状态图的绘制并不是必须的步骤，在发现系统中有一个对象的状态频繁发生变化，并且系统的行为会随之发生变化时，就需要记录这个对象的状态了。在招聘系统中，应聘者将会随着应聘过程的不断变化而处在应聘的不同阶段。在这期间，系统的行为将会随着应聘者状态的变化而有所不同。例如，当应聘者刚完成注册还未提出职位申请时，他处在未应聘状态，在这种情况下，系统是不会向他发出面试请求的，也不会出现上岗通知。这些是符合现实世界中的基本业务规律的。因此，应聘者是需要我们记录的状态变迁的对象，这有利于我们对系统使用的业务规则作进一步的了解。参见图 4-21 的应聘者的状态变迁图。

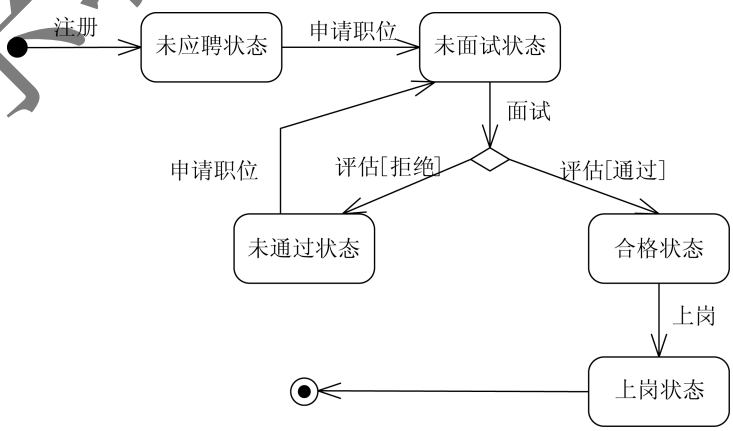

图 4-21　职位状态变迁图

4.7 案例分析

项目小组在听取了技术顾问老丁的讲解之后，明确了这个阶段的任务目标的策略，也清楚了各自在小组中的角色和位置，项目组长为这个阶段布置了小组成员各自的任务。

小王在需求确定阶段一直负责用例模型的构建，因此在分析阶段仍然对用例负责，继续完善和细化用例模型。

由于小王对需求较为熟悉，在分析阶段需要协助小李构建领域模型。在构建领域模型过程中，如果发现用例描述中术语存在不一致的地方，则需要对用例描述进行修改，保证术语的使用与领域模型中描述的一致性。在此期间，如果发现有某个对象的状态经常发生变化，则以状态图的形式记录下来。

小董负责进行健壮性分析，他在小李构建领域模型任务启动之后开始进行健壮性分析。在分析过程中需要使用到小李和小王已经构建好的领域模型和用例模型，如果在此过程中发现有不一致或不完整的内容，则需要相关负责的人员进行及时修正。

健壮性分析为顺利构建动态模型中的顺序图打下了良好的基础，因此，在接下来构建动态模型的过程中，老李将用例进行分工，让小组成员分别来绘制相应的顺序图。

项目组长老李强调了在系统分析阶段主要的目标是能够将从用户角度了解的系统转换为系统内部的表示，在此阶段可以不考虑具体的实现技术手段。因此，这个阶段的重点应集中在从需求到分析的转换复核上。这里需要对构建的每个场景的健壮图和用例文本进行复核，确保它们的一致性、完整性、正确性。同时确保领域模型与健壮图的一致，即领域模型中包含了健壮图中的所有实体对象。

4.7.1 构建领域模型和状态模型

在项目进行的过程中并不能够保证领域模型一次性构建完成，往往是根据用例分析的不断增加而不断补充完善的过程。表 4-6 给出了《招聘管理系统》案例的补充需求，我们先看看根据新补充进来的内容是否会存在新的领域概念，如果存在的话，如何补充到已有的领域模型中去。

表 4-6　　**发布职位用例**

用例名称	【006】发布职位	
概述	招聘人员在门户网站上发布一个职位	
参与者	招聘人员	
基本流程	Actor	System
	1. 招聘人员点击〈招聘人员主页面〉上的[发布职位链接]。 3. 招聘人员输入职位信息并发布提交请求。 8. 招聘人员在〈发布职位页面〉点击[返回链接]。	2. 系统显示为招聘人员录入职位信息的〈发布职位页面〉。 4. 系统保存职位信息。 5. 系统保存发布的日期。 6. 系统保存招聘人员与职位之间的关系。 7. 系统重新显示〈发布职位页面〉，并提示“职位发布成功！”。 9. 系统显示〈招聘人员主界面〉
分支流程	no	

根据前面所讲述的领域模型的构建过程，我们首先需要挑选类的候选对象，如表 4-7 所示。

表 4-7　　**类的候选对象**

招聘人员	发布日期
职位	关系
系统	页面
职位信息	
发布请求	

在挑选类的候选对象时我们会发现，多次的经验会帮助我们更为接近地挑选到领域概念。

在这些候选对象中，我们可以根据前面的经验首先将不相关的系统、页面两个候选对象拿出来；发布日期是对职位发布的一种描述，因此是发布的一种修饰，属于属性；虽然发布请求是动作，但是存在了对这种动作的特征描述——日期，因此，将发布请求归为领域类对象；职位与职位信息实际上描述的是同一件事情，因此选取简单的职位作为领域类对象。关系描述的是在应用过程中的一种约束，强调的是哪个招聘人员发布的哪个特定的职位，因此最终会转化为对类属性的约束。

表 4-8 描述了应用筛选原则进行筛选的结果。

表 4-8　　应用筛选原则筛选候选类的过程

概念类	属性	冗余	不相关
招聘人员	发布日期	职位信息	系统
职位	关系		页面
发布请求			

根据筛选后的结果将属性增加进去之后绘制的局部领域模型，如图 4-22 所示。

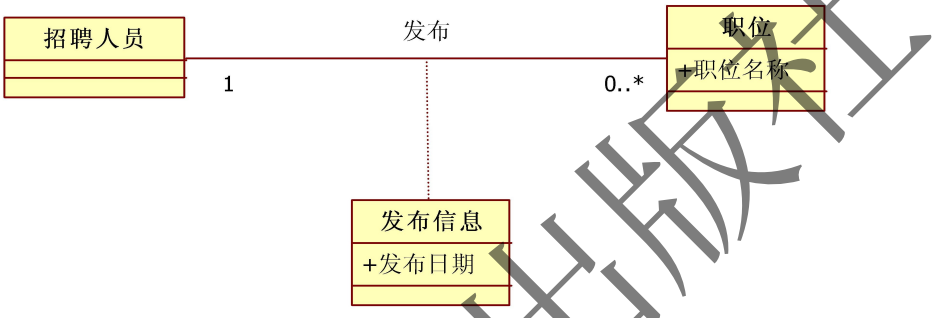

图 4-22　部分领域模型

用户的界面原型有助于我们补充类的属性，图 4-23 和图 4-24 显示了发布职位所需的界面原型。

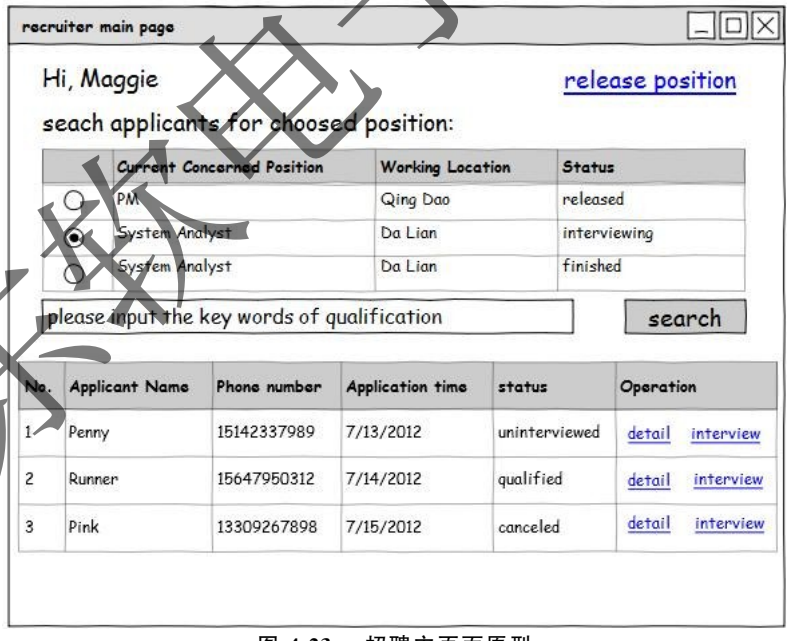

图 4-23　招聘主页面原型

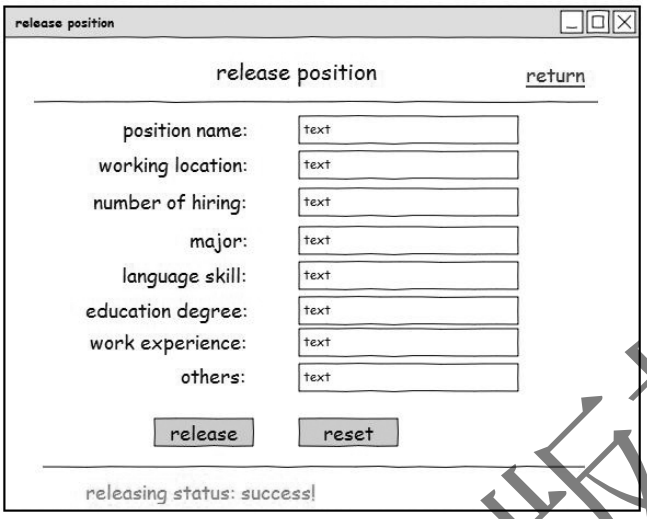

图 4-24　发布职位页面原型

将申请职位用例与发布职位用例所获得的领域模型进行合并之后获得的综合领域模型如图 4-25 所示。

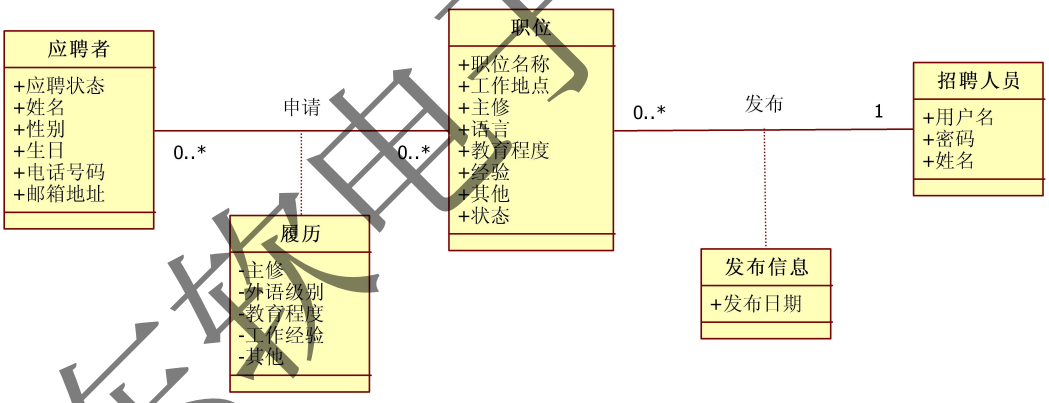

图 4-25　补充后的静态类图

领域模型帮助我们捋清领域内概念对象及它们之间的关系，为后续对系统的设计提供了统一的可视化术语表。

在构建领域模型的过程中，我们会发现职位从发布之后到最终确定人选，也有一个变化的过程。通过阅读如表 4-9 所示的查找职位用例，可以发现这个内容。

表 4-9 查找职位用例

用例名称	【003】查找职位	
场景	在门户网站上查找职位。	
触发事件	应聘者想找找到特定的职位。	
概要描述	应聘者输入查询条件后，他们可以获得职位列表极其详细信息。	
参与者	应聘者	
基本流程	Actor	System
	1. 应聘者在〈应聘主页面〉上输入职位的关键字，并提交请求。 3. 应聘者点击[详细信息链接]。	2. 系统在〈应聘主页面〉上显示状态为 ＜发布状态＞ 和＜面试状态＞的职位列表（状态为 ＜结束状态＞的职位将不被显示）。 4. 系统执行“获取职位详细信息”用例。
分支流程	2.1 应聘者点击[申请职位链接]，系统将执行”申请职位”用例。	

通过阅读与职位发布相关的用例我们可以获得职位的总体信息。

当招聘人员在门户网站上提交了职位信息后，职位就处在发布状态；

当招聘人员评估了第一位应聘人员后，该职位就转换成面试状态；

如果面试合格的人员人数还不到该职位所需要的人数时，则需要继续面试；直到面试人数符合职位要求人数为止，该职位状态转换成结束状态；此时，该职位将不再显示在应聘者的应聘页面上了。图 4-26 给出了职位状态变化的状态变迁图。

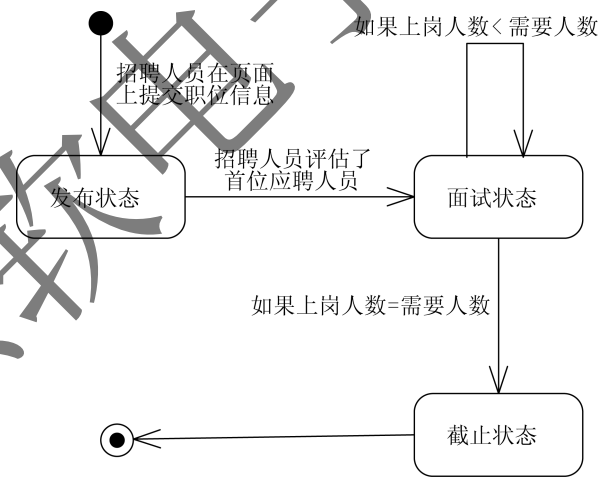

图 4-26 职位状态变迁图

4.7.2 健壮性分析

健壮性分析需要在具有基本流程和分支流程概述型用例描述的基础上来进行。因此，在完成了相应的用例模型之后，我们就可以进入到健壮性分析的环节。

针对于用例 5 申请职位，我们发现在确定是否申请过该职位的环节需要查找领域模型中对该信息的描述。领域模型中一个应聘者在应聘某个职位时会填写相应的履历，因此要确定该应聘者是否已经应聘过该职位，查找一下他是否已经提交过该职位的履历表就可以了。而对于该

应聘者如果提交了职位应聘申请之后就不能够在申请其他职位了，因此需要修改该应聘者的应聘状态来确认这一点，因此需要修改应聘者个人的状态信息。最终，我们得到了如图 4-27 所示的健壮图。

在我们进行健壮性分析的过程中，我们一方面需要仔细阅读用例文档，一方面还需要比对着领域模型中对对象之间关系的描述，这样才能确保健壮性分析的检查和验证的作用得到发挥。

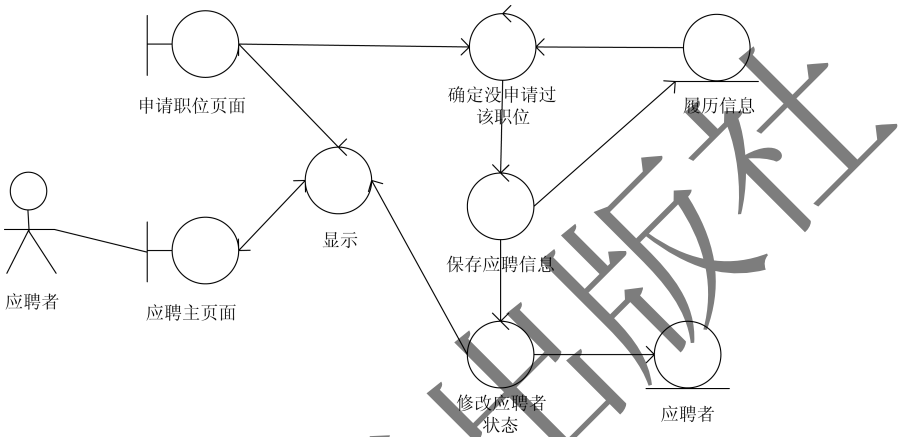

图 4-27　申请职位用例健壮图

4.7.3　构建动态模型

构建动态模型中的顺序图是根据健壮性分析中得到的页面和实体对象直接对应到顺序图中的边界对象和实体对象中。对于健壮图中的抽取出来的控制对象只是对动作的一个提取，并不能直接对应到顺序图中对应的控制对象中。这个分配的过程是我们在构建顺序图的主要任务。

图 4-28 是申请职位的顺序图。

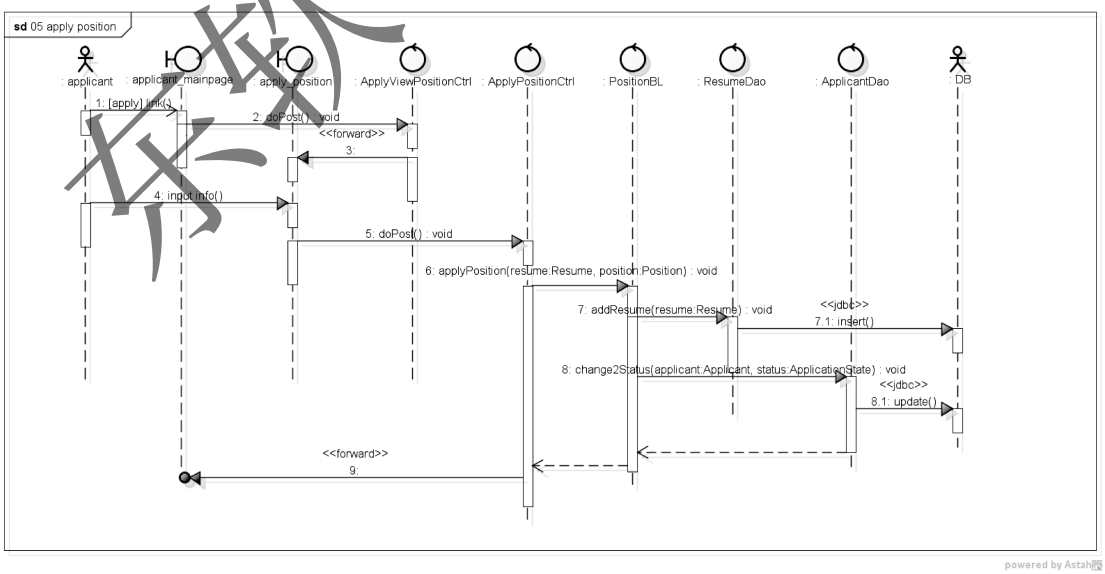

图 4-28　申请职位用例对应的顺序图

4.8 知识拓展

4.8.1 抽取候选类的其他方法

抽象领域模型中的候选类除了使用名词抽取法之外，还有其他两种方法。

(1)重用和修改现有的模型。

这实际上是首要、最佳且最简单的方法。如果条件许可，通常构建领域模型可以从这一步开始。在许多领域中，由于领域存在的稳定性，都存在已发布的、绘制精细的领域模型和数据模型。这些领域包括库存、金融、卫生等等领域。

(2)使用分类列表。

分类列表是通过对某个领域中的一些相关内容进行分类整理，以启发对领域中相关对象的认知。大家可以通过《UML 面向对象建模与设计》中介绍的概念类候选列表来开始创建领域模型。表中包含大量值得参考的常见类别，其中强调的是业务信息系统的需求。该准则还建议在分析时建立一些优先级。表 4-10 取自图书管理系统项目。

表 4-10　　概念类分类列表

概念类的类别	示例
业务交易 准则：十分关键(涉及金钱)，所以作为起点	借阅，归还 预订
交易准则 准则：交易中通常会涉及项目	图书 借书证
与交易或交易相关的产品或服务 准备：(产品或服务)是交易的对象	借书记录
与交易相关的人或组织的角色：用例的参与者 准则：我们通常要知道交易所涉及的各方	资料管理员、拣书者、藏书者 院图书馆管理系统
交易的地点；服务的地点	资料室
重要事件，通常包含我们需要记录的时间或地点	借阅记录、归还记录、催还列表
物理对象 准则：特写是在创建控制软件或进行仿真时非常有用	条码扫描仪
事务的描述	图书介绍、图书评价
类别：描述通常有类别	图书类别
事务(物理或信息)的容器	资料室、个人藏书室
容器中的事物	条目
其他协作的系统	院图书馆管理系统
金融、工作、合约、法律材料的记录	图(藏)书列表，统计报表
金融手段	
执行工作所需的进度表、手段、文档等	晒书计划表、图书推荐表

这里，没有什么正确的列表。我们可以根据自己的经验对上述列表进行调整和修改，这里面的分类主要是起到启发的作用。列表中的抽象事务和领域词汇在一定程度上是随意搜集的，但都是建模者认为重要的。

4.8.2　领域驱动设计(DDD)

Eric Evans 在《领域驱动设计——软件核心复杂性应对之道》(Addison－Wesley 2004，已由清华大学出版社翻译出版)一书中提出了“领域驱动设计(简称 DDD)”的概念。领域驱动设计是基于面向对象分析与设计技术，对技术架构进行了分层规划，同时对每个类进行了策略和类型的划分。

领域模型是领域驱动的核心。根据 DDD 的设计思想，业务逻辑由大量相对小的领域对象(类)组成，这些类是现实领域的业务对象映射，每个类是相对完整的独立体，具备自己在领域中的状态和行为。业务逻辑不再集中在几个大型的类上，而是由这样许多的细粒度的类组成。

领域驱动设计很好的遵循了关注点分离的原则，提出了成熟、清晰的分层架构。同时对领域对象进行了明确的策略和职责划分，让领域对象和现实世界中的业务形成良好的映射关系，为领域专家与开发人员搭建了沟通的桥梁。在领域驱动设计中，领域对象是核心，每个领域对象都是一个相对完整的内聚的业务对象描述，所以可以形成直接的复用。同时设计过程是基于领域对象而不是基于数据库的 Schema，所以整个设计也是可以复用的。领域模型适合具备复杂业务逻辑的软件系统，对软件的可维护性和扩展性要求比较高。不适用简单的增删改查业务。

基于领域驱动的设计，保证了系统的可维护性、扩展性和复用性，在处理复杂业务逻辑方面有着先天的优势。

4.9　本章小结

构建领域模型是系统分析师初步以面向对象的思想审视要构建系统的阶段。它抽取了业务领域中的关键概念，帮助项目小组成员更快地理解和深入到系统当中。领域模型中概念的抽取可以从已经建立的成果中抽取，也可以从大量的用户业务文档中进行抽取，总之这里将涉及到的信息量将会很大，但是根据 RUP 过程中使用到的原则，没有必要在构建领域模型时花费更多的时间，建议在创建初步领域模型时规定特定的时间。无论如何，你都无法使其十全十美，因此，应快速建立领域模型，并期望在以后的工作中将其逐步完善。

采用面向对象的观点从系统分析转换到设计阶段，健壮性分析起到了一种桥梁的作用。通过绘制健壮图，从用例描述中抽取边界对象、控制对象，并核实和补充实体对象，在此基础之上绘制序列图将会更加完善，使得分析到设计转换更加顺畅。动态模型包括顺序图、协作图、活动图和状态图，其中最重要的一种动态模型就是序列图，通过构建序列图完成为对象分配职责的工作，同时补充和完善系统类图。交互模型的构建使得整个系统设计进一步细化。

4.10 强化练习

1. 什么是领域模型？它与数据模型之间的区别是什么？

2. 请说出构建领域模型的基本步骤。

3. 请根据下面对图书管理系统的文字描述构建初步的领域模型。

(1)借书。图书管理员输入读者的借书证。系统首先检查借书证是否有效，若有效，对于第一次借书的读者，在读者账户文件上建立档案。否则，查阅读者账户，检查该读者所借图书是否超过 10 本，若已达到 10 本，拒借，未达 10 本，办理借书(检查图书目录中可借数量是否大于 D，修改图书详情中对应书籍的图书状态为已借，修改图书目录中的可借数量，修改读者账户中已借书数量，并将读者借书情况记录到图书借阅文件中)。

(2)还书。图书管理员获得欲还图书，并从读者账户文件和图书借阅文件中读出与读者有关的记录，查阅所借日期，如果超期(3 个月)作罚款处理，并记录到图书借阅文件中。否则，修改图书详情、读者账户、图书目录和图书借阅文件。

(3)查询。系统可根据图书管理员的查询请求，通过读者账户文件、图书目录等文件查询读者情况、图书借阅情况及图书库存情况，打印各种统计表。

4. 请根据下面对大学注册系统的文字描述构建初步的领域模型。

(1)大学的每个学位都设置了多门必修课和多门选修课。

(2)每门课程都处于给定的级别并有学分值。

(3)同一门课程可以是多个学位的组成部分。

(4)每个学位都规定了完成学位所需要的最低总学分值。例如，包括必修课在内，计算机技术学士需要 68 学分。

(5)学生可以对提供的课程进行组合，形成适合自己的学习计划，并完成这些课程就能获得他们所注册的学位。