系统分析

我们通过用例模型对系统的需求进行了完整的规格说明，这个规格说明将作为后续分析和设计的依据。而如何继续后续的分析过程也并没有一个统一的标准，甚至可以回到老路采用传统的结构化方法进行分析和设计；当然更好的选择显然还是继续采用面向对象方法。UML 用例分析技术则是一种典型的利用 UML 进行面向对象分析的技术，其主要思想来源于 RUP 分析设计工作流中的分析阶段，并适当地借鉴其它一些方法中的成功经验。通过本部分的学习能够掌握利用用例分析方法进行面向对象分析的基本过程和实践技能，并能够动手完成某一给定系统的分析模型。

1 架构分析

架构（Architecture）在面向对象的系统中扮演着越来越重要的角色，从某种意义上来说，面向对象的分析和设计都是以架构为中心进行的。在分析和设计的不同阶段，软件系统的架构被一步步细化和完善，最终形成一个规范的、稳定的、符合设计要求的架构模型。架构分析的过程就是定义系统高层组织结构和核心架构机制的过程。在早期迭代中，架构分析的主要目标是建立系统的备选架构，以用于组织后续的用例分析所获得的分析模型，该备选架构通过架构设计进行细化和调整，从而获得软件系统的基础架构。

本案例的早期架构选型时，采用流行的分层结构，其架构如图 1 所示。

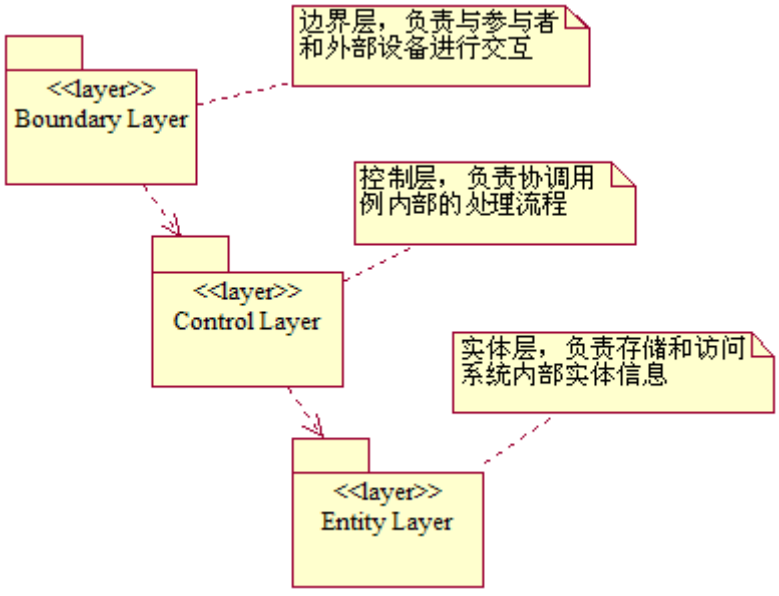

图 1 系统备选架构

在 B-C-E 三层架构的定义过程中，采用包图进行架构建模。包是一种通用的模型分组机制，为了有效地表示分层的概念需要引入构造型<<layer>>，并在层和层之间建立了依赖关系。通过 B-C-E 这三层划分系统三类处理逻辑，其中：

- 边界层(Boundary Layer)负责系统与参与者之间的交互；
- 控制层(Control Layer)处理系统的控制逻辑；
- 实体层(Entity Layer)管理系统使用的信息。

目前只是给出了这三个层次的基本定义，而各层的内部还没有任何元素，后续过程将对架构中的各个层次进行填充。

2 识别分析类

在对象系统中，系统的所有功能都是通过相应的类来实现的；因此首先需要从用例文档中找出这些可用的类，之后再将其所描述的系统行为分配到这些类中。面向对象的分析是对这个过程的第一次尝试，这是一个从“无”到“有”的跨越，也是整个分析过程中最难的一部分任务之一。而分析阶段所定义的类称之为分析类。

分析类代表了系统中具备职责和行为的事物的早期概念模型，这些概念模型最终会演化为设计模型中的类或子系统。分析类关注系统的核心功能需求，用来建模问题域对象。此外，分析类主要用来表现“系统应该提供哪些对象来满足需求”，而不关注具体的软硬件实现的细节。

架构分析中所定义的 B-C-E 三层备选架构为识别分析类提供了很好的思路，按照该备选架构，系统中的类相应地对应三个层次，即边界类、控制类和实体类。识别分析类的过程就是从用例文档中来定义这三类分析类的过程。

旅游申请系统中分析类较多，根据备选架构识别了三种分析类。图 2~图 4 分别给出了该系统的边界类、控制类和实体类的主视图，从图中可以看出该系统首次迭代所识别的全部分析类，这些分析类完全是按照前面所介绍的方法识别出来的。

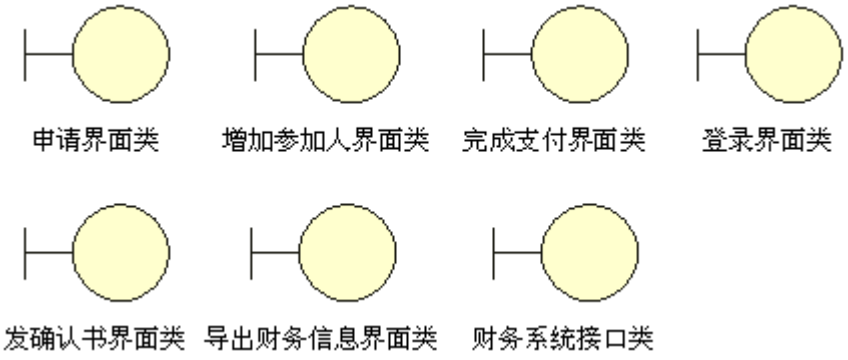

图 2 旅游申请系统初始边界类

本系统首次迭代共定义了 7 个边界类，其中包括 6 个界面类和 1 个系统接口类。由于首次迭代针对“管理参加人”用例只实现“增加参加人”子流程，因此其界面类也定义为“增加参加人界面类”；而“登录界面类”则同时处理前台服务员和收款员工两类用户的登录；此外，虽然时间参与者可能并不需要操作界面来启动系统（一般情况下是系统后台自动运行的业务），但为了后续分析的一致性，目前也定义一个“导出财务信息界面类”。

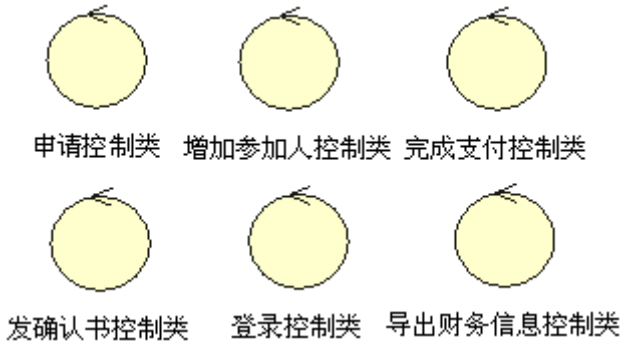

图 3 旅游申请系统初识控制类

控制类的定义比较简单，对应于首次迭代的 6 个用例，定义了 6 个控制类来封装相应用例的业务流程和逻辑规则。

图 4 旅游申请系统初始实体类

本系统目前初步定义了 7 个实体类，这些实体类与前面的关键抽象基本一致；所多出来的“用户”类是从登录用例中提取出来的，该类记录了系统的用户（包括前台服务员、收款员工和路线管理员等不同角色）信息，如用户名、密码等。关键抽象的提取更多的是凭借着对业务的理解和相关项目的经验；而此阶段对实体类的提取还可以按照前面所提到的名词筛选法来进一步明确并获得更多的实体类。

3 构造用例实现

目前所识别的分析类都是静态的描述，而为了确认所识别的分析类是否达成用例实现的目标，必须分析由这些类所产生的对象的动态行为，这就是构造用例实现的过程。在 UML 模型中，通过交互图来表示对象间的交互，它由一组对象和它们之间的消息传递组成。下面将利用构造用例实现的相关技术，详细讨论“旅游申请系统”中“办理申请手续”用例实现的构造过程。

图 5 给出了“办理申请手续-用例实现”的基本场景顺序图。该顺序图描述的基本流程如下：

前台服务员在申请界面上首先录入路线代码和出发日期（消息 1），界面类根据这些信息向控制类查询所要申请的旅游团和路线信息（消息 1.1），控制类执行查询请求，根据查询结果生成相应的旅游团对象（创建消息 1.1.1）和路线对象（创建消息 1.1.2），并将这两个对象关联起来（消息 1.1.3，即设定该旅游团对应的路线），最后返回旅游团对象（返回消息 1.1.4）；界面对象接收到返回结果后，进行刷新，从而显示所查询到的旅游团和路线信息。基本事件流的 1~4 步完成。

之后，用户确认需要申请该旅游团，前台服务员向界面录入用户的申请信息（消息 2），界面类将申请内容提交给控制类（消息 2.1），控制类针对申请信息的不同方面交由相应的实体类进行处理：首先生成一个申请对象（消息 2.1.1），并与旅游团对象关联（消息 2.1.2），表明该申请所对应的旅游团；之后，生成一个参加人对象（消息 2.1.3），来存储申请的责任人信息，并在申请对象中关联责任人信息（消息 2.1.4）；最后，控制类要求申请对象计算本次申请有关的支付信息（消息 2.1.5），申请对象根据自身的信息（包括申请的大人人数、小孩人数、申请日期等）和所关联的旅游团信息（包括旅游团的价格、出发日期等）来计算费用和订金等支付信息（消息 2.1.5.1），并生成支付明细对象来保存相应的结果（消息 2.1.5.2），并将结果返回给控制类（返回消息 2.1.5.3）。控制对象将本次申请的明细返回给界面后（省略了该返回消息），界面类进行刷新显示（消息 2.2）。

最后，用户根据系统计算出来的订金进行支付，服务员通过界面类将用户的支付信息录入到系统（消息 3），界面将支付结果提交给控制类（消息 3.1），控制类根据支付结果更新申请对象的状态（消息 3.1.1），同时申请对象也会把支付情况记录到支付明细对象中（消息 3.1.1.1）。

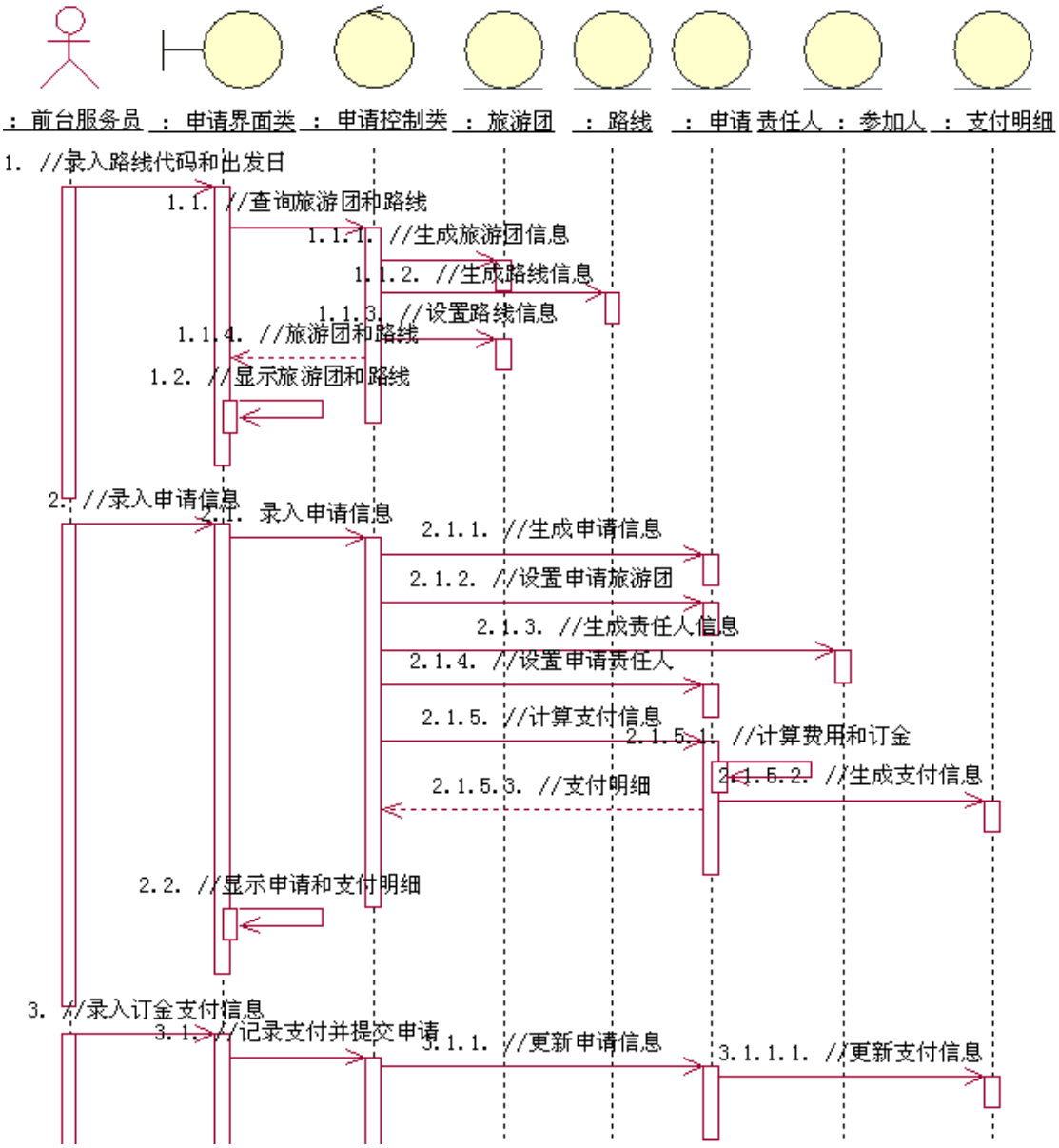

图 5 办理申请手续-用例实现的基本场景顺序图

针对该顺序图的绘制，还存在三个方面的问题需要进一步说明：

① 命名对象。正如前面所展示的顺序图，图中大部分对象都是匿名对象，并不需要指定特定的对象名称，只需要明确所属的类的类型，从而泛指所有通过该类所构造的对象。但在有些特殊场合下，为了特指该类某个特殊的对象，需要为该对象取特定的名字。在该图中，为“参加人”类定义了一个“责任人”对象，从而区分不同参加人的身份。这意味着，在当前办理申请手续-用例实现中，只需要记录那个作为责任人的参加人信息，而不需要维护其他普通参加人的信息。

② 设置对象间的关联。在该顺序图中，有诸如 1.1.3、2.1.2、2.1.4 等“设置某某信息”的职责，这类职责实际上是建立两个对象之间的关系。比如 1.1.3 职责是为旅游团设置其路线信息：通过控制类查询到了旅客所要申请的旅游团和路线，并分别存储到两个类中，但对于某个旅游团对象而言，应该要指定该旅游团是针对哪个路线的，即要建立旅游团类和路线类之间的关联关系，这个操作就是 1.1.3 所要达到的目标。同样的道理，2.1.2 建立申请和旅游团之间的关系，而 2.1.4 建立申请和责任人之间的关系。

③ “计算费用和订金”职责的处理。对于此类与业务逻辑相关的、涉及到多个对象的职责很

多时候都是由控制类来处理，但该顺序图中将该职责分配给申请类。而这种分配策略在此处是非常合适的，因为计算费用和订金所需的申请人数信息和旅游团费用信息都与申请类之间存在关系，即通过申请类可以明确地获得这些内容，因此按照专家模式，这是一个很合理的方案。

4 构造分析类图

通过构造用例实现验证了需求的可实现性：即可以利用识别出的分析类和它们之间的交互来达到用例的目标；这是整个分析过程最核心的工作。然而，对于面向对象的系统来说，所描述的这些交互最终都应该在相应的类中来定义并由类的对象来实现。虽然在构造用例实现的过程中已经获得了类的基本定义，但那是在一个个用例实现的基础上完成的，主要关注的是用例事件流的交互过程，而对单个类自身的特征和行为缺少统一的考虑。因此，下一阶段就需要将注意力集中到每一个分析类本身，在关注类自身定义的基础上再重新评估每个用例实现的需要。此外，一个类及其对象常常参与多个用例实现，此时更需要从类的整体角度去协调用例的行为。

构造分析类图的最终目标就是从系统的角度明确说明每一个分析类的职责和属性以及类之间的关系；并根据这些视图来描述和理解目标系统，从而为后续的设计提供基本的素材。

图 6 展示了“旅游申请系统”实体类类图（为保持图形的清晰，图中有些类的属性和操作并没有完全显示出来，如参加人类的操作等），该图中涉及到类、类的职责、属性和关系等内容都全部展示出来了。

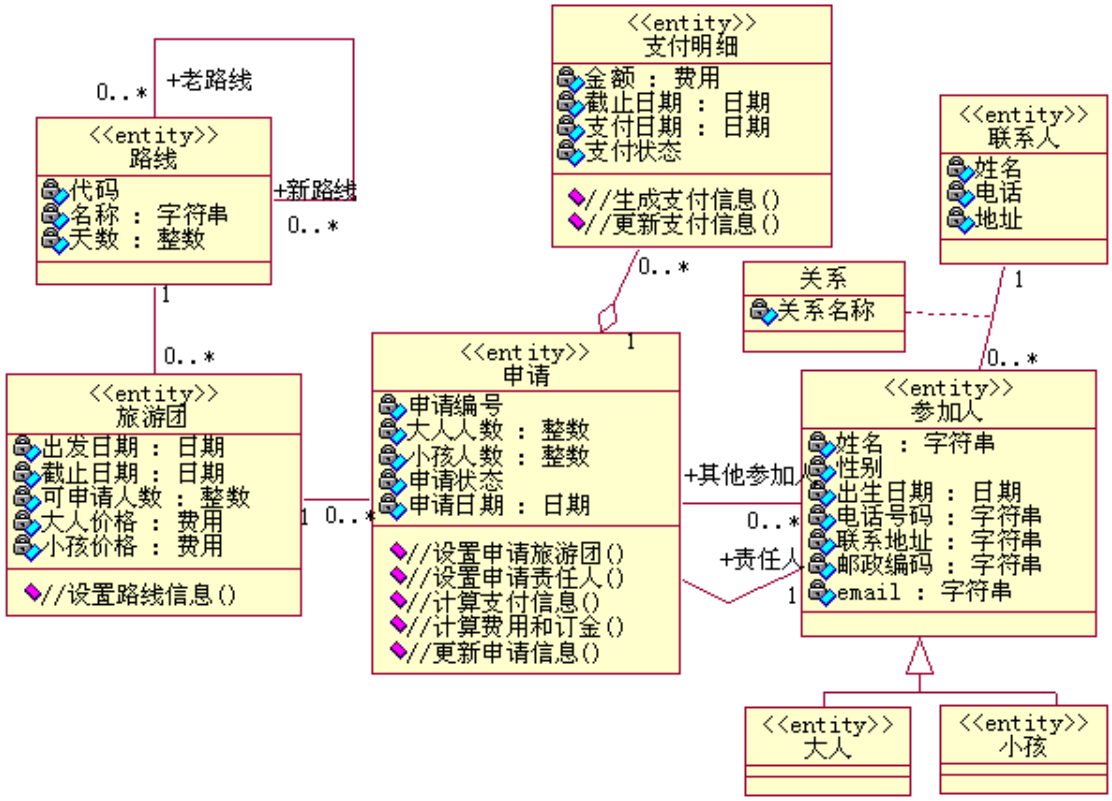

图 6 旅游申请系统实体类类图

需要说明的是，事实上整个分析阶段的重点就在于找出体现系统核心业务所需数据的实体类，而界面和业务逻辑细节分别由边界类和控制类隐藏；因此图 6 所展示的实体类类图就可以很好地反映“旅游申请系统”的分析成果。在有些面向对象分析方法中，分析阶段的工作就是找到这些实体类，由这些实体类即构成了系统概念模型这一最主要的分析成果。