**用例图的关系**

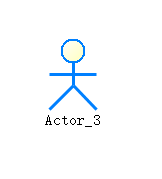
用例图描述的是参与者所理解的系统功能，主要元素是用例和参与者，是帮助开发团队以一种可视化的方式理解系统的功能需求。这时处于项目初始，分析用户需求的阶段，不用管怎么实现具体的功能，只要能向客户形象化的表述项目的功能就行。

用例图有四个部分：用例（Use Case), 参与者（Actor),系统边界，关系。

## 参与者（Actor)

参与者是与系统交互的人或物。首先当然包括我们的开发系统用户，除此之外，与我们开发的系统有关联的其他系统也算是参与者。

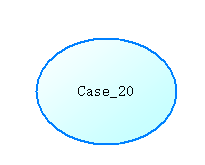
在UML图中我们用一个小人表示。



## 用例（Use Case)

用例是参与者可以感受到的系统服务或功能单元。我理解的就是用户可以使用我们开发的项目去做的任何事情

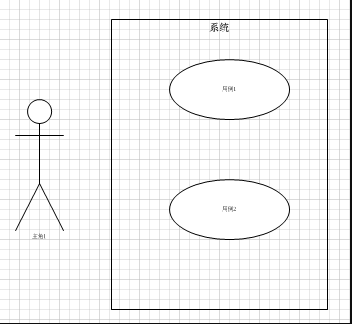
任何用例都不能在缺少参与者的情况下独立存在，同样，任何参与者也必须要有与之关联的用例。在UML图中我们用椭圆表示：



## 系统边界

指系统与系统之间的界限。把系统边界以外的同系统相关联的其他部分称为系统环境。

在UML图中我们用一个矩形表示。



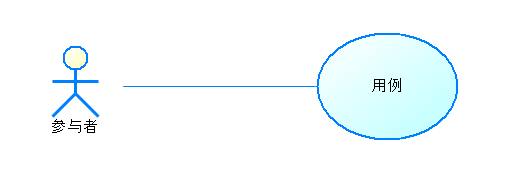
## 关系

用例图中的关系有4种：关联，泛化，包含和扩展。

### 关联

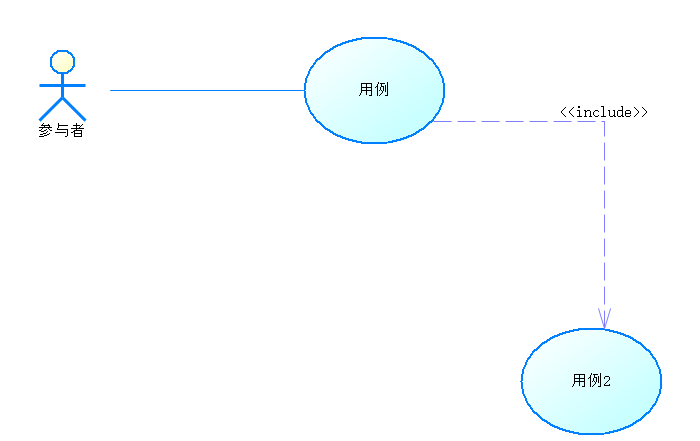
表示参与者和用例之间的交互。为通信途径，任何一方都可发送或可接收消息。

箭头指向：指向消息接收方。在UML中用直线表示



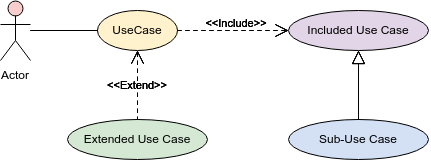
### 包含

包含关系用来把一个较复杂的用例所表示的功能分解成较小的步骤。包含用例是必须的，如果缺少包含用例，基用例就是不完整的。包含关系最典型的应用就是复用。这种情况类似与在过程设计语言中，将程序的某一段算法封装成一个子过程，然后在从主程序中调用这一子过程（这么说好像懂了点）  
在UML中，包含关系用带箭头的虚线段加《include》表示，箭头指向被包含的用例。  
如下图用例和用例2的关系是包含(include)



### 扩展

扩展关系是指用例功能的延伸。与包含关系不同的是，扩展用例是可选的，如果缺少扩展用例。不会影响到基用例的完整性。在UML中，扩展关系用带箭头的虚线段加《extend》表示，要注意的是箭头指向基用例。  
如下图用例和用例3是扩展关系



### 泛化

用例的泛化指的是一个父用例可以被特化形成多个子用例，用我们熟悉的语言来说就是继承关系。在UML中，泛化关系用空心箭头表示，箭头指向的是父用例。

