**案例编号**：UM003

**案例名称**：关于事件处理的用例关系构建

**案例目标**：

通过此案例，让学生理解三种用例关系和如何构建这三种用例关系。

**案例描述**：

用例之间有三种关系，分别是：包含include、扩展extends和泛化generalization。本案例通过三个场景下的用例关系构建，让学生理解三种关系的用法和意义，以及构建这三种关系的方法和语法规则。

**Keyword**：包含、扩展、泛化、用例关系

**系统描述**：

本案例所属系统为相对抽象的事件处理系统。在系统中，拥有对事件的处理、资源的分配和紧急事件的呼叫等功能。为了安全性，用户可以通过密码或者指纹进行系统登陆。在登陆系统后，可以开始一个事件，然后对事件进行管理，比如编辑事件名称、参考地图或其他工具为事件分配资源等。如果遇到紧急事件，可以进行紧急呼叫并采取措施进行管理。

**案例建模过程**：

第一个场景：事件的管理

1. 分析用例。“用户开始一个事件，对事件进行管理，参考地图为事件分配资源”，从这段话可以提取出开始事件OpenIncident、分配资源AllocateResources，开始一个事件或者分配资源，都需要参考一个工具——地图，因为两个用例都需要用到这个用例“查看地图”，故将这个用例分离出来，作为单独一个用例查看地图ViewMap。
2. 考虑OpenIncident和AllocateResources与ViewMap的关系：OpenIncident和AllocateResources都需要用到ViewMap，并且如果没有ViewMap，OpenIncident和AllocateResources也是无效的。这种情况下就是《include》关系。
3. 把三个用例的关系用UML语法规范画下来。

第二个场景：紧急事件的报告

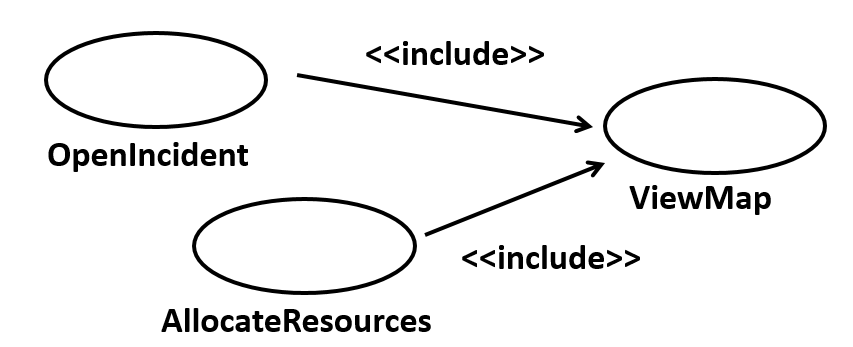
1. 分析用例。“遇到紧急事件，可以进行紧急呼叫并采取措施进行管理”，从这段话提取出工作人员可以对紧急事件进行汇报，所以就有紧急事件报告ReportEmergency用例。存在一种情况，当事件过于紧急时，工作人员可以进行呼救，故有Help用例。
2. 考虑ReportEmergency和Help用例之间的关系。可以肯定它们不是include关系，因为ReportEmergency没有Help用例也可以存在，而Help是在ReportEmergency基础上存在的，只有在有紧急事件时，才有可能产生呼救这个用例，这种情况下就是《extends》关系。
3. 把这两个用例的关系用UML语法规范画下来。

第三个场景：用户验证

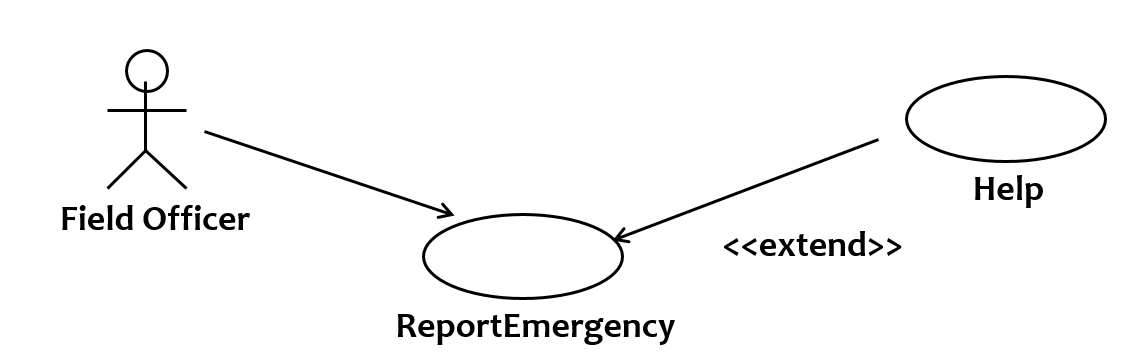
1. 分析用例。“用户可以通过密码或者指纹进行系统登陆”，从这段话可以提取出验证用户ValidateUser、检查密码CheckPassword和检查指纹CheckFingerprint三个用例。
2. 考虑以上三个用例之间的关系。相对于include和extends，泛化关系相对比较容易的找到，检查密码和检查指纹都属于对用户进行验证，故CheckPassword和CheckFingerprint都是ValidateUser的子用例，所以它们是Generalization泛化关系。
3. 把三个用例的关系用UML语法规范画下来。

**案例结果**：

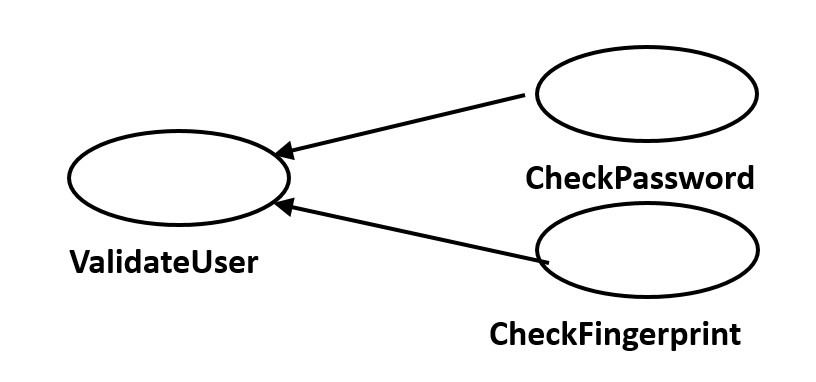
第一个场景：



第二个场景：



第三个场景：



**案例总结**：

本案例通过一个抽象的事件处理案例描述了用例之间的关系，通过这个案例我们需要明白如下几点：

1. 泛化(generalization)：泛化关系是一种继承关系，子用例将继承基用例的所有行为，关系和通信关系，也就是说在任何使用基用例的地方都可以用子用例来代替。泛化关系在用例图中使用空心的箭头表示，箭头方向从子用例指向基用例。
2. 扩展(extend)： extend关系是对基用例的扩展，基用例是一个完整的用例，即使没有子用例的参与，也可以完成一个完整的功能。extend的基用例中将存在一个扩展点，只有当扩展点被激活时，子用例才会被执行。 extend关系在用例图中使用带箭头的虚线表示(在线上标注<<extend>>)，箭头从子用例指向基用例。
3. 包含(include)： include为包含关系，当两个或多个用例中共用一组相同的动作，这时可以将这组相同的动作抽出来作为一个独立的子用例，供多个基用例所共享。因为子用例被抽出，基用例并非一个完整的用例，所以include关系中的基用例必须和子用例一起使用才够完整，子用例也必然被执行。include关系在用例图中使用带箭头的虚线表示(在线上标注<<include>>)，箭头从基用例指向子用例。