# 大学生海报设计竞赛系统

## 需求分析和设计

### 需求分析

在系统工程及软件工程中，需求分析指的是在创建一个新的或改变一个现存的系统或产品时，确定新系统的目的、范围、定义和功能时所要做的所有工作。需求分析是软件工程中的一个关键过程。在这个过程中，系统分析员和软件工程师确定顾客的需要。只有在确定了这些需要后他们才能够分析和寻求新系统的解决方法。

在软件工程的历史中，很长时间里人们一直认为需求分析是整个软件工程中最简单的一个步骤，但在过去十年中越来越多的人认识到它是整个过程中最关键的一个过程。假如在需求分析时，分析者们未能正确地认识到顾客的需要的话，那么最后的软件实际上不可能达到顾客的需要，或者软件无法在规定的时间里完工。[[1]](#footnote-1)

所以，在动手设计、开发软件系统之前，首先需要了解一下软件需求的背景。

#### 背景

为了展示大学生的艺术才华，激发其创新意识和实践能力，我们决定开发一套大学生海报设计竞赛系统。大学生可以创建或加入队伍进行竞赛、查看赛事、查看作品、分享作品、点赞作品等。管理员可以发布竞赛、设置获奖、发布新闻、数据统计等。评委可以查看作品，设置分数，设置评语等。

根据背景介绍，可以确定海报设计竞赛系统需要解决以下几个方面的问题：

表1‑1业务前景表

|  |  |
| --- | --- |
| **编号** | **目标** |
| P01 | 大学生可以创建或加入队伍进行竞赛、查看赛事、查看作品、分享作品、点赞作品等。 |
| P02 | 管理员可以发布竞赛、设置获奖、发布新闻、数据统计等。 |
| P03 | 评委可以查看作品，设置分数，设置评语等。 |

确定业务前景是需求分析的第一步，它给我们确定了所要开发软件系统的范围。通常情况下，后续所有工作都应该围绕业务前景来开展，凡是和业务前景无关的工作都不予考虑，除非调整业务前景。

为了方便，我们给这个管理系统取一个英文缩写名称：PDCS，也就是Poster Design Competition System系统。

#### 用例分析

对于复杂的系统，通常还需要涉众分析、业务建模等工作。由于本系统非常简单，所以这里直接进行用例分析[[2]](#footnote-2)。用例分析解决三个问题：什么“人”会和系统交互？用系统做什么？和系统之间的交互过程是怎么样的？需要注意的是，回答这三个问题时，必需**完全站在系统使用者的角度来看**，绝对不要考虑系统实现方面的细节问题。

这里用“系统使用者”这个术语，而不是用户，是因为使用系统的不一定是人，还有可能是和系统交互的其他系统，所以在用例分析中将其叫做“角色[[3]](#footnote-3)（Actor）”。PDCS有三个角色“大学生”、“评委”和“管理员”。UML[[4]](#footnote-4)建模工具中，表示角色的图标如图1‑1所示：

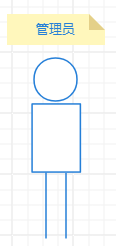


图1‑1 Actor图标

根据前面的背景描述（通常还需要和相关的人员[[5]](#footnote-5)交流）可以确定大学生会使用系统完成以下操作：“查看赛事”、“查看作品”、“分享作品”、“点赞作品”、“创建或加入队伍”。这些需要的操作就叫做“用例（Use Case）”，表示Use Case的图标如图1‑2所示：

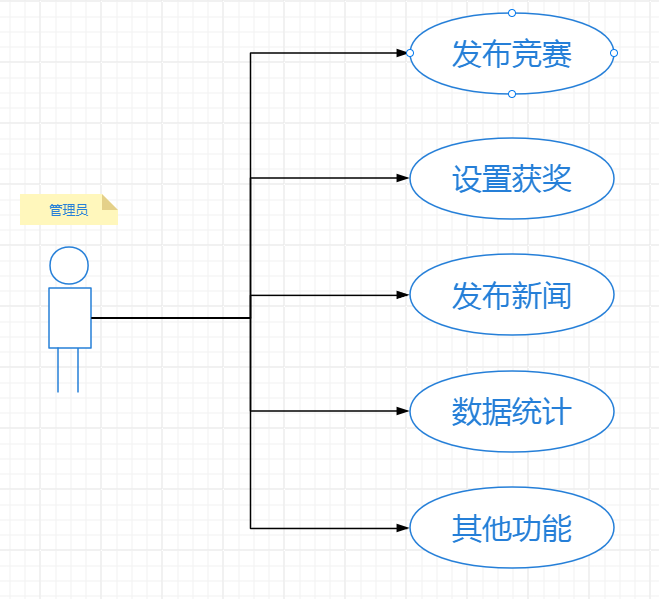


图1‑2 Use Case图标

Use Case是UML中非常重要的概念，其定义为：在不展现一个系统或子系统内部结构的情况下，对系统或子系统的某个连贯的功能单元的定义和描述。其实Use Case就是对系统功能的描述，不过一个Use Case描述的是整个系统功能的一部分，**这一部分一定要是在逻辑上相对完整的功能流程**。在使用UML的开发过程中，需求是用Use Case来表达的，界面是在Use Case的辅助下设计的，很多类是根据Use Case来发现的，测试实例是根据Use Case来生成的，包括整个开发的管理和任务分配，也是依据Use Case来组织的。[[6]](#footnote-6)

虽然同一个Use Case可能被多个Actor所使用，但为了避免遗漏，我们在识别和分析Use Case时，要**对于每个Actor来逐个进行**。将Actor和Use Case联系起来，才构成最终的Use Case图，并没有只有Use Case的Use Case图。

PDCS是大学生海报设计竞赛系统，角色是“读者”和“管理员”。具体的Use Case图如图1‑3所示。



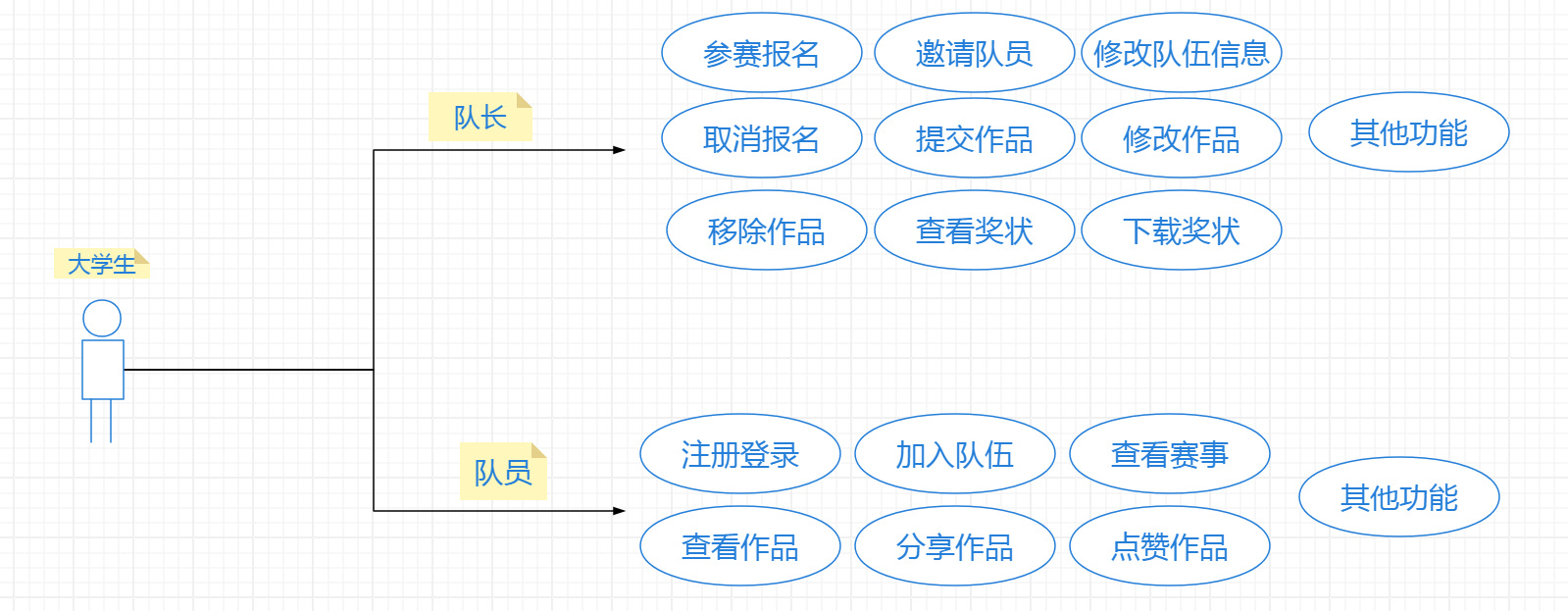
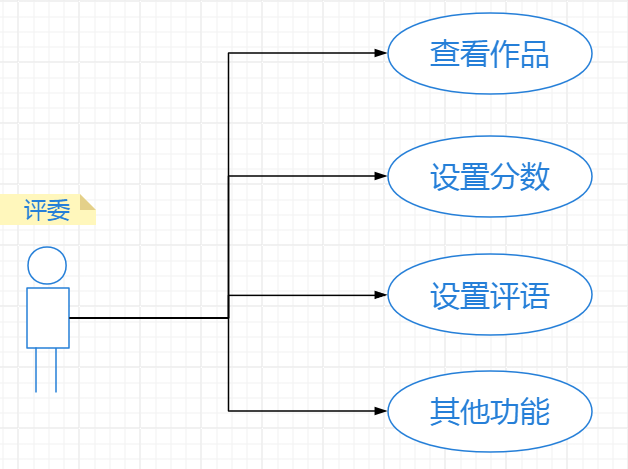
`

图1‑3 PDCS用例图

显然，用例图只回答了本节的前两个问题，对于第三个问题的回答需要通过对Use Case的描述来完成。Use Case的描述可以用各种方法，但在UML中，通常用两种图来描述Use Case，它们就是顺序图（Sequence Diagram）和协作图（Collaboration Diagram）。当系统比较简单明确时，我们也可以略过这部分描述，直接在设计阶段给出细化后的内容。

### 概要设计

概要设计的主要任务是把需求分析得到的用例图转换为软件结构和数据结构。设计软件结构的具体任务是：将一个复杂系统按功能进行模块划分、建立模块的层次结构及调用关系、确定模块间的接口及人机界面等。数据结构设计包括数据特征的描述、确定数据的结构特性、以及数据库的设计。显然，概要设计建立的是目标系统的逻辑模型，与计算机无关。

这里强调一下，在实际应用软件工程时不要拘泥于理论完整性。也就是说，只要能够把问题描述清楚，就没有必要给出软件工程方法论中所列举的全部“交付件”。这一点，以后不再赘述。

#### 界面设计

一般来说，概要设计首先得完成模块的划分，建立模块的层次结构和关系，然后为每个模块设计界面。但软件系统非常简单时，我们可以把界面设计和模块设计结合在一起：当我们为每个模块都设计了界面，那么系统的模块也就确定了。

界面设计就是人机界面设计，也就是UI（User Interface）设计，是指对软件的人机交互、操作逻辑、界面美观的整体设计。好的UI设计不仅让软件变得有个性有品味，还让软件的操作变得舒适、简单，是一门很大的学问。

概要的界面设计，最方便的工具就是纸和笔，当然也可以使用一些软件工具。Web应用系统首页的要素通常包括，横幅标题、导航链接、快捷菜单、内容展示。下面给出一个简单的PDCS首页界面示意图如图1‑4所示。



图1‑4首页界面设计示意图

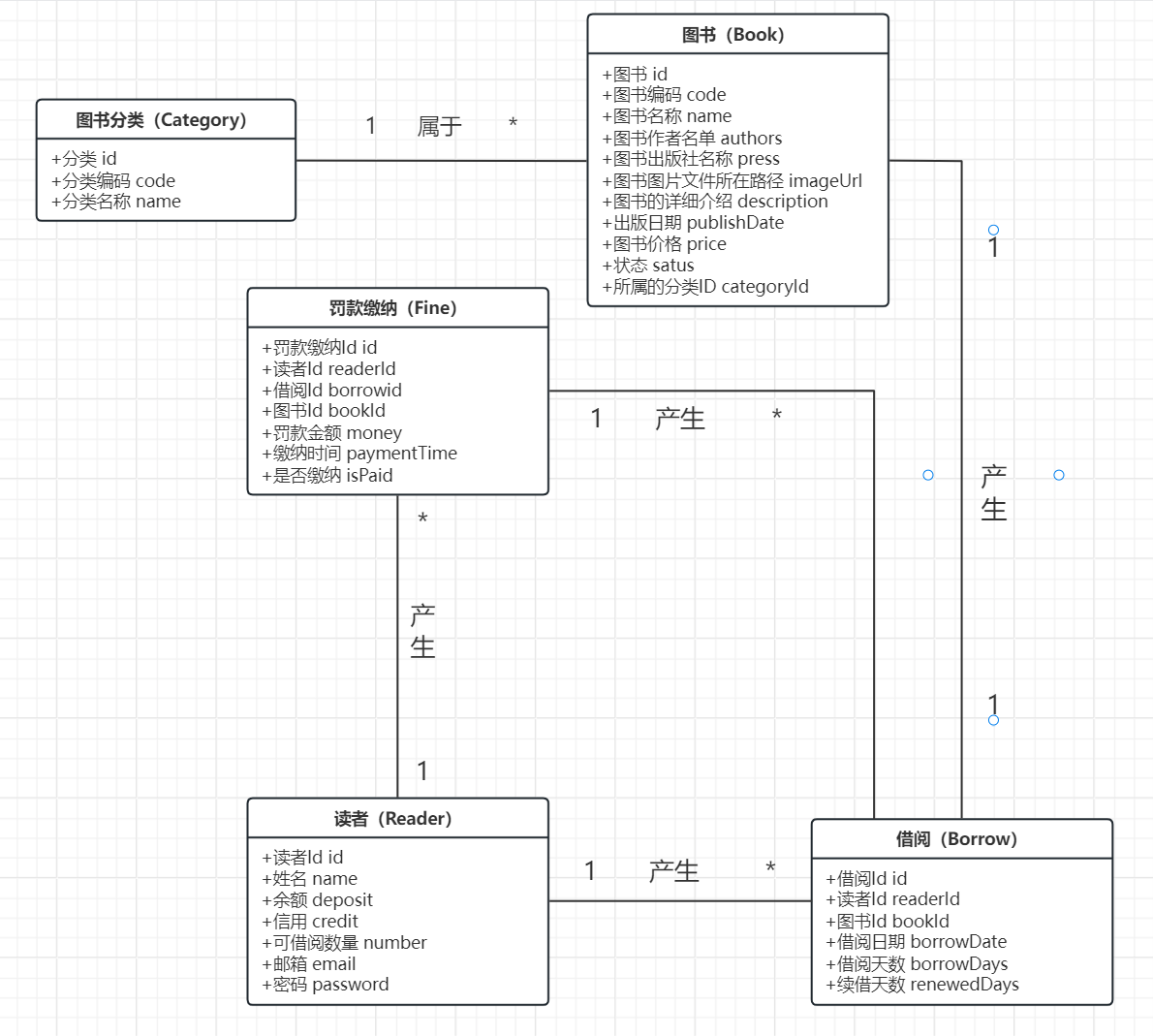
首页包含五大部分：1）左上方的横幅标题，2）左边是分类链接菜单，3）主区域是图书的列表，4）其上方提供的快速查找资料搜索框。

本节只给出了典型的一个界面设计，我们将在后面进一步细化、实现。其它模块的界面设计请大家自行补充完善。

### 数据库设计

#### 数据库实体类图

根据前面的需求分析和系统设计，使用实体类图进行数据库概念模型。（）

图1‑5 实体类图

#### 数据关系模型

SSLS的关系数据库中只需要5张表，这里给出了图书分类和图书表的设计，请大家补充其他表的设计：

表1‑2图书分类表Category

|  |  |  |  |
| --- | --- | --- | --- |
| **字段名** | **类型** | **属性** | **说明** |
| id | Int | PK，IDENTITY | 分类Id。(自增型) |
| code | Varchar(50) | NOT NULL | 分类编码，用于快速录入。 |
| name | Varchar(50) | NOT NULL | 分类名称。 |

表1‑3图书表Book

|  |  |  |  |
| --- | --- | --- | --- |
| **字段名** | **类型** | **属性** | **说明** |
| id | Int | PK，IDENTITY | 图书ID。(自增型) |
| code | Varchar (50) | NOT NULL | 图书编码，用于快速录入。 |
| name | Varchar (100) | NOT NULL | 图书名称。 |
| authors | Varchar (100) | NOT NULL | 图书作者名单。 |
| press | Varchar (100) | NOT NULL | 图书出版社名称。 |
| imageUrl | Varchar (100) | NULL | 图书图片文件所在的路径。 |
| description | Varchar (1000) | NULL | 图书的详细介绍。 |
| publishDate | DateTime | NULL | 出版日期。 |
| price | Decimal (10,2) | NULL | 图书价格。 |
| satus | Varchar(50) | NULL | 状态。未在库、在库、外借。 |
| categoryId | Int | FK,NOT NULL | 所属的分类ID。 |

表1‑4读者表Reader

|  |  |  |  |
| --- | --- | --- | --- |
| **字段名** | **类型** | **属性** | **说明** |
| id | Int | PK，IDENTITY | 读者Id。(自增型) |
| name | Varchar(50) | NOT NULL | 姓名 |
| deposit | Decimal (10,2) | NOT NULL | 余额 |
| credit | Int | NOT NULL | 信用 |
| number | Int | NOT NULL | 可借阅数量 |
| email | Varchar(50) | NOT NULL | 邮箱 |
| password | Varchar(50) | NOT NULL | 密码 |

表1‑5 借阅表Borrow

|  |  |  |  |
| --- | --- | --- | --- |
| **字段名** | **类型** | **属性** | **说明** |
| id | Int | PK，IDENTITY | 借阅Id。(自增型) |
| readerId | Int | FK,NOT NULL | 读者Id |
| bookId | Int | FK,NOT NULL | 图书Id |
| borrowDate | DateTime | NOT NULL | 借阅日期 |
| borrowDays | Int | NOT NULL | 借阅天数 |
| renewedDays | Int | NOT NULL | 续借天数 |

表1‑6 罚款缴纳表Fine

|  |  |  |  |
| --- | --- | --- | --- |
| **字段名** | **类型** | **属性** | **说明** |
| id | Int | PK，IDENTITY | 罚款缴纳Id。(自增型) |
| readerId | Int | FK,NOT NULL | 读者Id |
| borrowId | Int | FK,NOT NULL | 借阅Id |
| bookId | Int | FK,NOT NULL | 图书Id |
| money | Decimal (10,2) | NOT NULL | 罚款金额 |
| paymentTime | DateTime | NOT NULL | 缴纳时间 |
| isPaid | Bit | NOT NULL | 是否缴纳 |

接下来，大家可以进行数据库设计。数据库名称这里确定为PDCS。数据库表的实现可以使用SQL脚本完成，也可以使用图形界面设计完成。请大家设计完成后，在文档中补充SQL脚本。创建数据库表的SQL脚本如下：  
CREATE DATABASE PDCS;  
USE PDCS;  
SET FOREIGN\_KEY\_CHECKS=0;  
  
DROP TABLE IF EXISTS `admin`;  
CREATE TABLE `admin` (  
 `AdminID` int(11) NOT NULL AUTO\_INCREMENT,  
 `AdminName` varchar(255) NOT NULL,  
 `AdminNumber` varchar(255) NOT NULL,  
 `AdminPsd` varchar(255) NOT NULL,  
 PRIMARY KEY (`AdminID`),  
 UNIQUE KEY `AdminNumber` (`AdminNumber`)  
) ENGINE=InnoDB DEFAULT CHARSET=utf8;  
  
DROP TABLE IF EXISTS `awards`;  
CREATE TABLE `awards` (  
 `AwardID` int(11) NOT NULL AUTO\_INCREMENT,  
 `WorkID` int(11) NOT NULL,  
 `AwardName` varchar(255) NOT NULL,  
 `CompetitionID` int(11) NOT NULL,  
 PRIMARY KEY (`AwardID`),  
 KEY `WorkID` (`WorkID`),  
 KEY `CompetitionID` (`CompetitionID`),  
 CONSTRAINT `awards\_ibfk\_1` FOREIGN KEY (`WorkID`) REFERENCES `works` (`WorkID`),  
 CONSTRAINT `awards\_ibfk\_2` FOREIGN KEY (`CompetitionID`) REFERENCES `competitions` (`CompetitionID`)  
) ENGINE=InnoDB DEFAULT CHARSET=utf8;  
  
DROP TABLE IF EXISTS `competitions`;  
CREATE TABLE `competitions` (  
 `CompetitionID` int(11) NOT NULL AUTO\_INCREMENT,  
 `CompetitionName` varchar(255) NOT NULL,  
 `CompetitionDate` date NOT NULL,  
 `RegistrationDeadline` date NOT NULL,  
 `CompetitionTypeID` int(11) DEFAULT NULL,  
 `CompetitionTypeName` varchar(255) DEFAULT NULL,  
 PRIMARY KEY (`CompetitionID`),  
 KEY `fk\_competition\_type` (`CompetitionTypeID`),  
 CONSTRAINT `fk\_competition\_type` FOREIGN KEY (`CompetitionTypeID`) REFERENCES `competitiontypes` (`CompetitionTypeID`)  
) ENGINE=InnoDB DEFAULT CHARSET=utf8;  
  
DROP TABLE IF EXISTS `competitiontypes`;  
CREATE TABLE `competitiontypes` (  
 CompetitionTypeID` int(11) NOT NULL AUTO\_INCREMENT,  
 `TypeName` varchar(255) NOT NULL,  
 PRIMARY KEY (`CompetitionTypeID`)  
) ENGINE=InnoDB DEFAULT CHARSET=utf8;  
  
DROP TABLE IF EXISTS `judges`;  
CREATE TABLE `judges` (  
 `JudgeID` int(11) NOT NULL AUTO\_INCREMENT,  
 `JudgesName` varchar(255) NOT NULL,  
 `JudgesNumber` varchar(255) NOT NULL,  
 `JudgesPsd` varchar(255) NOT NULL,  
 PRIMARY KEY (`JudgeID`),  
 UNIQUE KEY `JudgesNumber` (`JudgesNumber`)  
) ENGINE=InnoDB DEFAULT CHARSET=utf8;  
  
DROP TABLE IF EXISTS `participant`;  
CREATE TABLE `participant` (  
 `participant\_id` int(11) NOT NULL AUTO\_INCREMENT,  
 `participant\_name` varchar(255) NOT NULL,  
 `participant\_number` varchar(50) NOT NULL,  
 `participant\_psd` varchar(50) NOT NULL,  
 `participant\_address` varchar(255) NOT NULL,  
 PRIMARY KEY (`participant\_id`)  
) ENGINE=InnoDB DEFAULT CHARSET=utf8;  
  
DROP TABLE IF EXISTS `teams`;  
CREATE TABLE `teams` (  
 `TeamID` int(11) NOT NULL AUTO\_INCREMENT,  
 `TeamName` varchar(255) NOT NULL,  
 `CaptainID` int(11) NOT NULL,  
 `Member1ID` int(11) DEFAULT NULL,  
 `Member2ID` int(11) DEFAULT NULL,  
 `Member3ID` int(11) DEFAULT NULL,  
 PRIMARY KEY (`TeamID`),  
 KEY `CaptainID` (`CaptainID`),  
 KEY `Member1ID` (`Member1ID`),  
 KEY `Member2ID` (`Member2ID`),  
 KEY `Member3ID` (`Member3ID`),  
 CONSTRAINT `teams\_ibfk\_1` FOREIGN KEY (`CaptainID`) REFERENCES `participant` (`participant\_id`),  
 CONSTRAINT `teams\_ibfk\_2` FOREIGN KEY (`Member1ID`) REFERENCES `participant` (`participant\_id`),  
 CONSTRAINT `teams\_ibfk\_3` FOREIGN KEY (`Member2ID`) REFERENCES `participant` (`participant\_id`),  
 CONSTRAINT `teams\_ibfk\_4` FOREIGN KEY (`Member3ID`) REFERENCES `participant` (`participant\_id`)  
) ENGINE=InnoDB DEFAULT CHARSET=utf8;  
  
DROP TABLE IF EXISTS `works`;  
CREATE TABLE `works` (  
 `WorkID` int(11) NOT NULL AUTO\_INCREMENT,  
 `WorkName` varchar(255) NOT NULL,  
 `CompetitionID` int(11) NOT NULL,  
 `TeamID` int(11) NOT NULL,  
 `Likes` int(11) DEFAULT '0',  
 `Comments` text,  
 `ImageURL` varchar(255) DEFAULT NULL,  
 PRIMARY KEY (`WorkID`),  
 KEY `CompetitionID` (`CompetitionID`),  
 KEY `TeamID` (`TeamID`),  
 CONSTRAINT `works\_ibfk\_1` FOREIGN KEY (`CompetitionID`) REFERENCES `competitions` (`CompetitionID`),  
 CONSTRAINT `works\_ibfk\_2` FOREIGN KEY (`TeamID`) REFERENCES `teams` (`TeamID`)  
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

1. http://zh.wikipedia.org/wiki/需求分析 [↑](#footnote-ref-1)
2. 用例分析可以用在业务建模阶段，也可以用于系统建模阶段，甚至用在非计算机领域。由于系统过于简单，这里并不明确区分。 [↑](#footnote-ref-2)
3. Actor翻译成“角色”是有争议的，建议直接用英文。 [↑](#footnote-ref-3)
4. Unified Modeling Language (UML)又称统一建模语言或[标准建模语言](http://baike.baidu.com/view/3458435.htm)，是始于1997年一个OMG标准，它是一个支持模型化和软件系统开发的图形化语言。 [↑](#footnote-ref-4)
5. 标准的称呼为“涉众Stakeholder”，包括可能的用户、系统开发出资人等等所有相关的人员。 [↑](#footnote-ref-5)
6. http://baike.baidu.com/Use case [↑](#footnote-ref-6)