论文中的共性问题

1. 用例模型及用例描述（或用例规约）、

用例模型非常重要的使用场景之一是捕获、分析及验证需求，是需求分析员与客户交流的媒介，通过阅读需求规格说明书，用户才能知道需求分析员是否准确理解实际业务需求、是否遗漏了实际业务需求，所以用例模型一定是站在用户的角度构建，采用用户可理解的语言来表述。这部分内容主要存在以下问题：

1. 用例粒度不能太细

用例与系统功能不是一一对应的，用例模型着重刻画业务功能（用户关心的是系统是否涵盖了所有业务）。除了业务功能，系统还存在辅助功能。

对于大部分数据表的增、删、改、查操作，无需写成独立的用例。如添加用户、删除用户、修改用户，不管是添加、删除还是修改操作，在用户看来，都是管理用户，在用例模型中可以采用“管理用户”一个用例来表达。

1. 用例之间的关系需准确

一个用例模型中，大部分用例之间不存在任何关系。

用例之间的主要关系包含：include和extend。

include代表的是，一个用例要完成，被包含用例必须被执行。include关系的使用场景是，多个用例中包含一些共同的步骤，为了避免写多次，将共同的步骤提取出来形成一个用例，该用例不具备独立价值。所以要慎用include关系，实际上用例之间很少这种关系。



机构后台不是用例。假设机构后台是用例，包含关系表达的是，要完成机构后台这个用例，除了机构后台包含的步骤，还必须完成资质申请、机构详情和修改密码的步骤，这显然不符合逻辑。

extend代表的是，一个用例完成后，扩展用例是可选动作，扩展用例具有独立价值。如何“查询航班”用例执行完后，可以“打印航班信息”，是否打印是可选项，这种情况下，两个用例之间是扩展关系。

1. 用例描述不是使用说明书

还处于需求分析阶段，概要设计都还未开始，更不用说编码实现，用例描述中事件流的描写不应涉及界面，如何实现还不清楚，是开发一个桌面程序、还是Web程序、还是APP都有可能。

此外，事件流描述的是参与者与系统的交互过程，是交替的关系。如：发布养成心得的正常事件流

1. 会员请求发布养成心得
2. 系统显示养成心得发布界面
3. 用户填写心得
4. 系统保存心得
5. ER图
6. 不是所有用户角色都是实体

很多同学将“管理员”识别为实体，理由是管理员可以添加用户、可以做很多操作。出现这种情况还是因为从系统功能的角度进行数据建模，最简单检查方式是查看表之间是否存在这种关系。

1. 没有实体图

论文中画实体图，首先没有实体图这一说，只有实体联系图（ER），实体图可以用关系模式替代，不要随便创造一些名词。

1. 层次功能框图

为了便于在word文档中展示，可以采用层叠方式绘制，尽量画一个图，而不是画很多个功能图。

登录

注册

\*\*\*\*\*\*系统

后台

前台