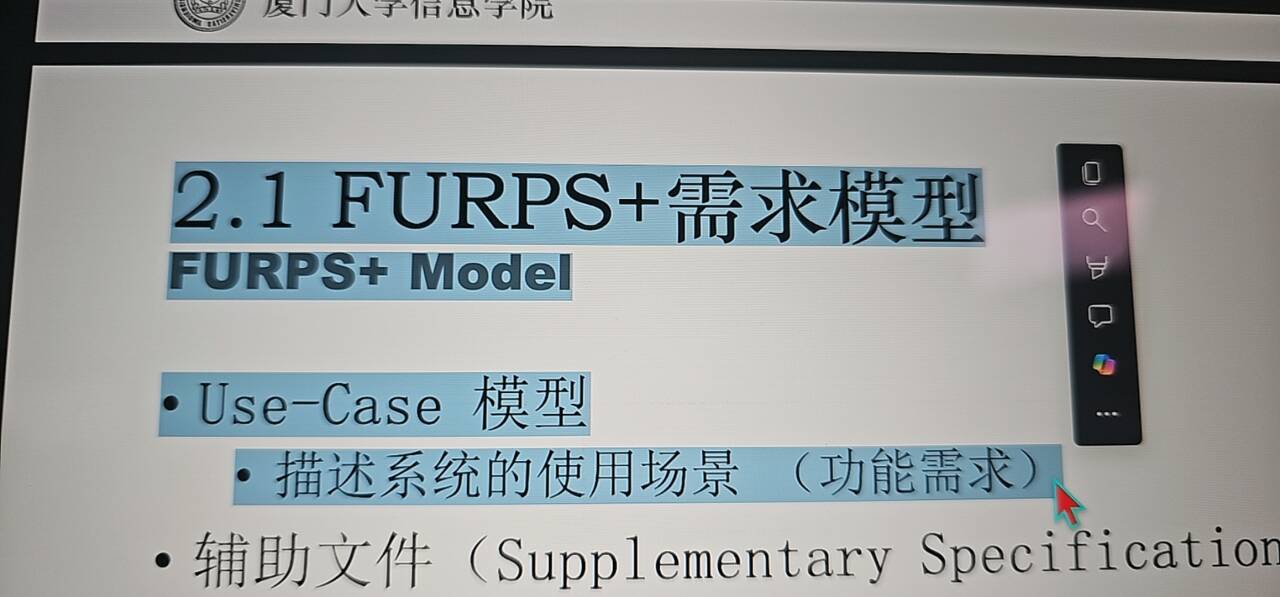


Functional：用例模型，是一种用于描述系统功能需求的工具，通过从用户的角度展示系统的功能及其执行者，帮助开发团队理解系统的功能需求



Usability: 使用需求主要是讲用户如何使用这个系统？（app，网页，微信小程序等等方式，做多端的软件每一端的代码都不一样，这就是成本）

Reliability：不是软件的可靠性，是整体的可靠性（交付给用户的是一个整体），单纯考虑软件的可靠性是没有意义的

性能：在有限的硬件资源下，通过软件来实现尽可能多的完成需求，好的软件设计可以减少大量的硬件开销

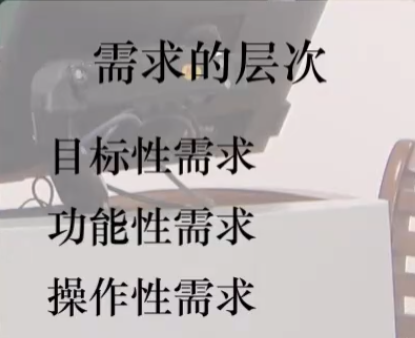
可支持性：前端，适配各种设备；后端，国际化，支持各种地区的语言来使用等等

实现：比如不在windows上实现而是在Ubuntu上实现，资源的限制

接口：外围的接口，如支付牌照，开发票要跟税务局的接口去对接

法律：功能的合法性，用户信息的保密等

—————————————————————————————————————



目标性需求：比如电商平台要的就是流量，有流量才能赚钱

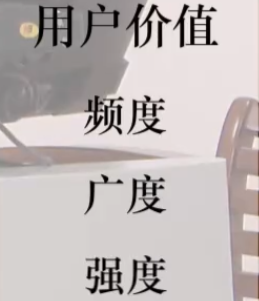
功能性需求：就是要怎么获得流量，用什么样的功能获得流量，如对接抖音微信等吸引流量；针对性的做功能，比如做垂直电商

操作性需求：不同的人有不同的操作需求，电商，顾客，后端人员的操作需求都不同

—————————————————————————————————————

需求优先级：

从用户价值和公式价值来衡量

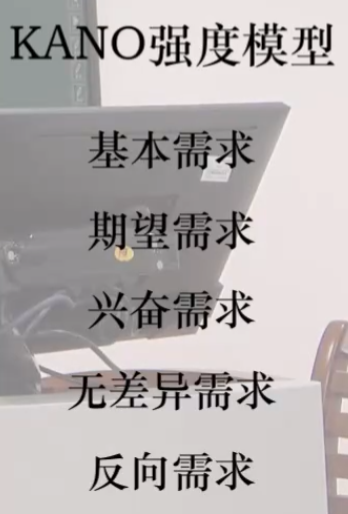


频度：用户使用的频率方面，比如电商平台

用户的浏览商品的频度会比看订单信息的频度高，频度高的功能会有更高的要求，就可以提升它的重要性资源分配更多，可以保证用户的使用体验更好

广度：各种各样的用户，用电商来说顾客是最多的，优先满足顾客的使用需求就是一个好的选择

强度：



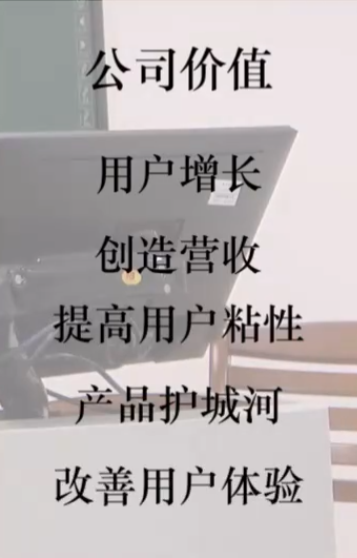
基本需求：用户能买东西

期望需求：用户希望能得到更多的有用的功能

兴奋需求：对于每一个用户做个性化的推荐，也就是给用户构建一个信息茧房，让他在自己的舒适区内更愿意频繁的去下单

无差异需求：有没有无所谓

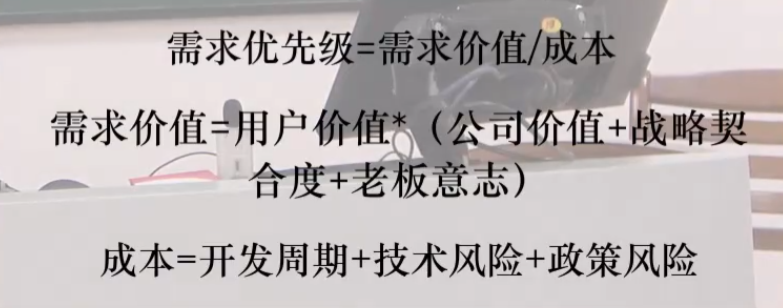
反向需求：做了以后反而不好



用户增长：功能对用户增长有帮助

创造营收：做的功能能提升用户的订单转化率

提升用户粘性：用户看的越久订单转化率越高

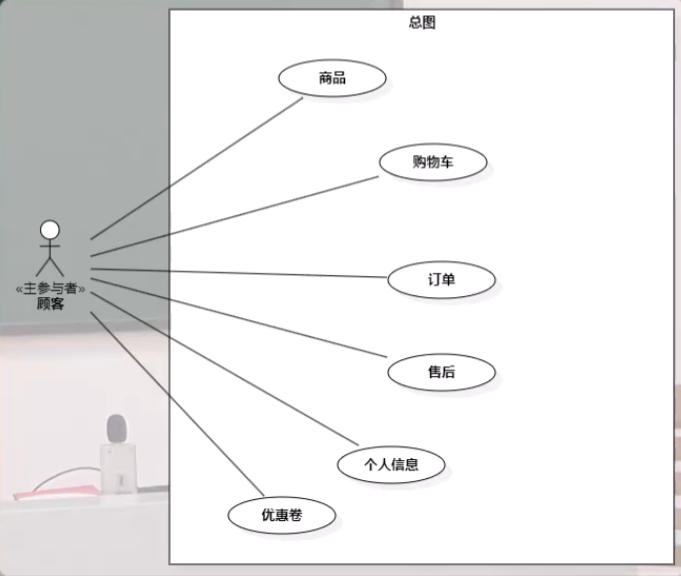


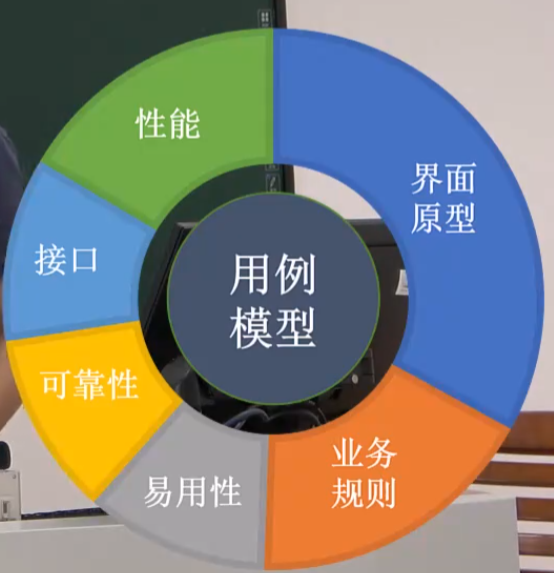
—————————————————————————————————————

用例模型：需求模型中比重最大最重要的部分



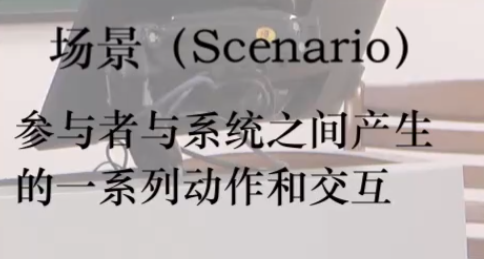
用例图

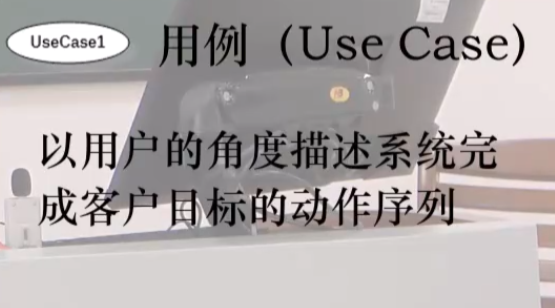




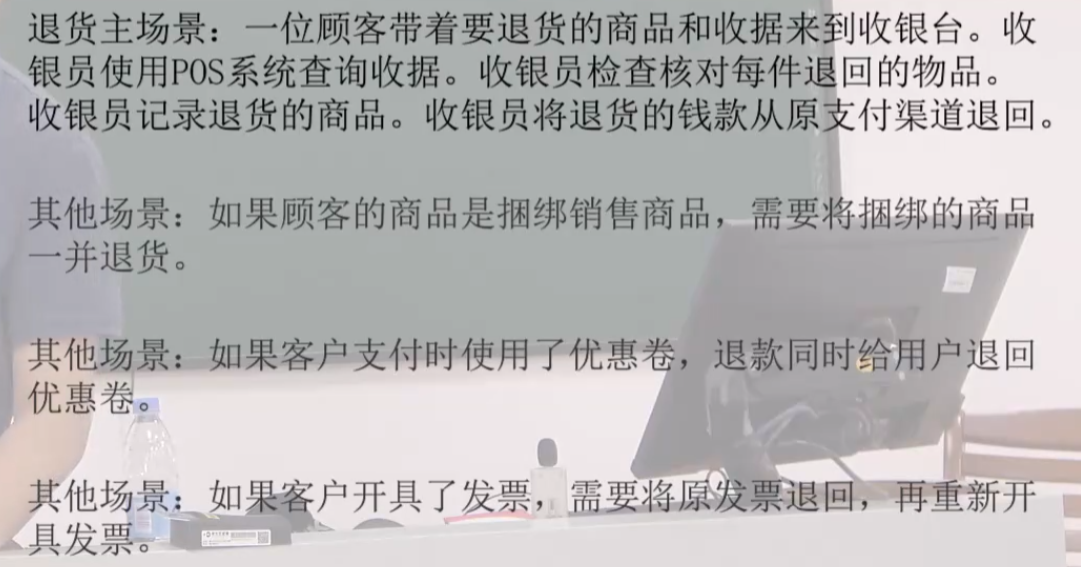
用例的构成：



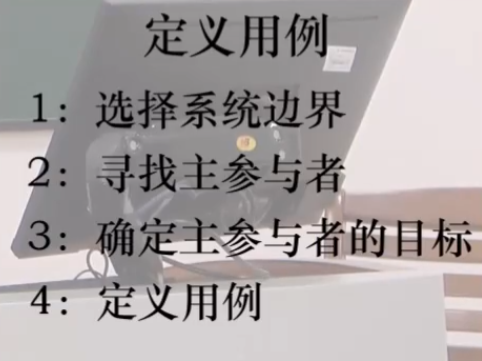


场景的集合

举例：



—————————————————————————————————————

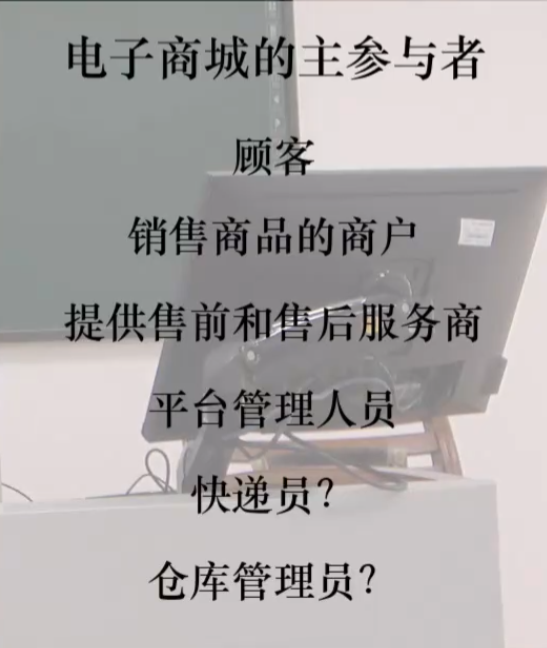


1.只管卖的方面，不管仓库，进出货之类的

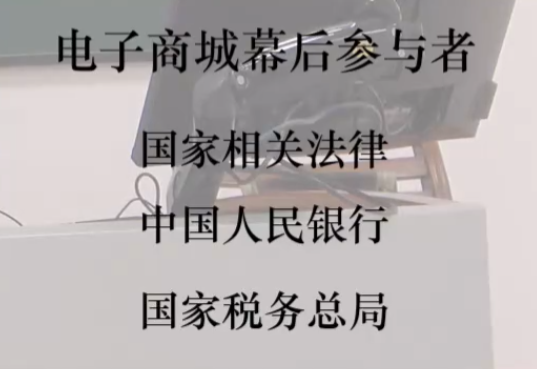
3.主参与者的目标（即用例的出发点）



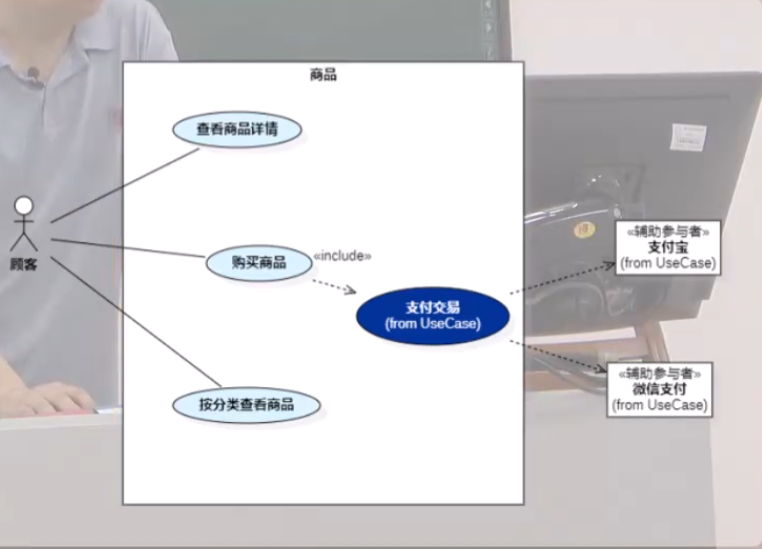
确认主参与者的例子：

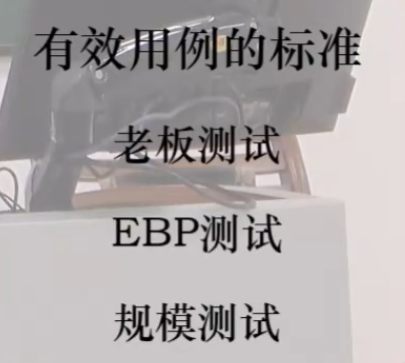
一边确定主参与者一边划定边界

确认辅助参与者的例子：



主操作者，用例，辅助操作者，边界的关系：





**老板测试**：老板问：“你整天都干什么了？”，你回答：“登录系统。”老板会满意吗？，那么该用例不会通过老板测试，这意味着该用例与达到可量化价值的结果没有太大关系。

**EBP测试**：EB**P**即基本业务过程（一个人于某个时刻在一个地点所执行的任务，用以响应业务事件。 该任务能够增加可量化的业务价值，并以持久状态留下数据。

**规模测试**：用例由单独的活动或步骤组成，相反，用例通常包含多个步骤，在详述式用例通常需要3～10页文本。

—————————————————————————————————————

用例的级别



**1. 用户级别（User-Goals）浅蓝色**

* 核心：**以用户目标为导向**，描述用户为完成**具体任务**（如 “登录系统”“提交订单”）与系统的交互。
* 特点：粒度适中，直接对应用户的核心需求，是用户视角下最直观的用例类型，能明确 “用户通过系统要达成什么”。

**2. 汇总级别（Summary Level）白色**

* 核心：**对多组相关用例的 “概括性组织”**，展现系统的**宏观业务流程 / 功能模块**（如 “电商平台的‘购物全流程’” 可汇总 “选商品”“加购物车”“支付” 等多个用户级用例）。

多个用户之间都存在的关系形成一个汇总

* 特点：偏向 “高层概览”，帮助团队快速理解系统的核心业务逻辑如何组织，无需关注细节。

**3. 子功能级别（Subfunctions）深蓝色**

* 核心：**聚焦系统内部的细粒度子功能 / 交互步骤**，是对 “用户级别用例” 的**细节分解**（如 “用户登录” 可拆分为 “输入账号”“密码加密验证”“生成会话令牌” 等子功能级用例）。

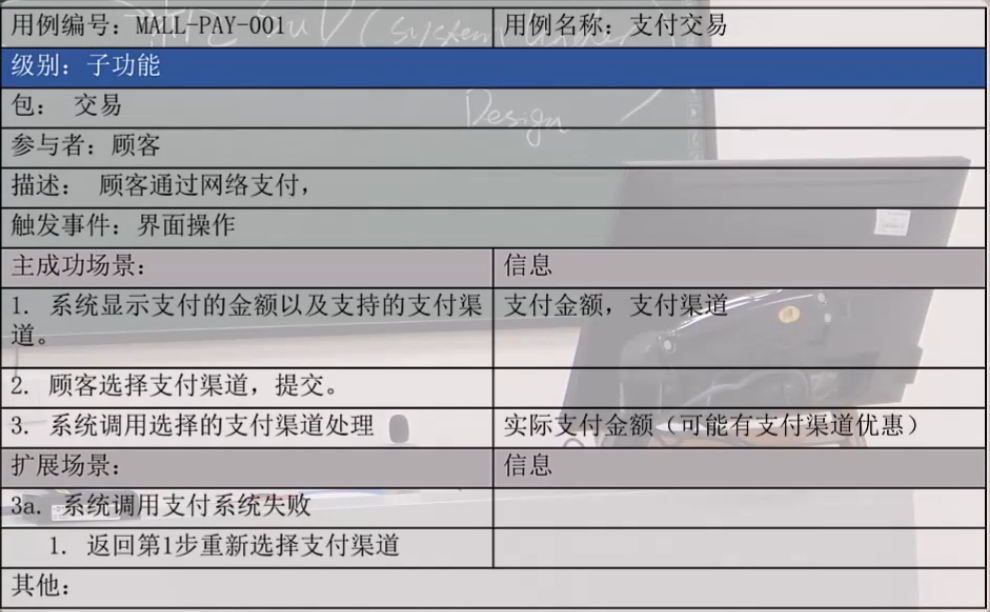
多个用户会重复用到的功能形成子功能

* 特点：粒度最细，用于**详细设计与开发阶段**，指导具体功能模块的实现，确保用户级的 “大目标” 能通过一个个微观子功能的协作完成。

—————————————————————————————————————

用例的格式：





用例编号：用来追踪用例的设计

用例名称

级别：浅蓝色为用户级别（白色是汇总级别，深蓝是子功能级别）

包：归属于的汇总用例

主成功场景，扩展场景：一个用例是多个场景组成的

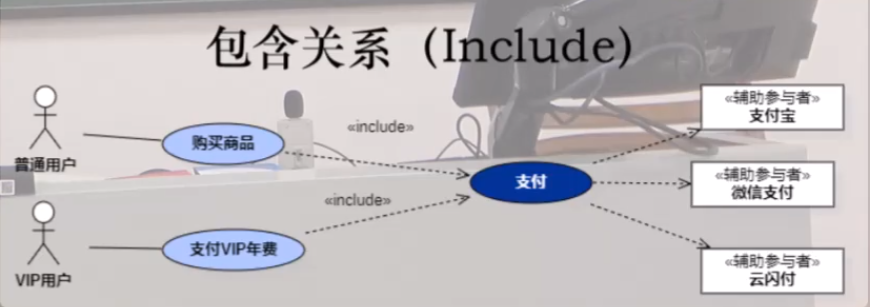
主成功场景：happy path 用户顺利的购买商品的一个过程

扩展场景：主成功场景的分支可能

写用例的关键是俩点，第一：用例中人和系统的交互（场景）（行为需求），第二：交互了什么信息（信息需求），不需要描述用例的内部实现，用例步骤不要太长，不超过九步

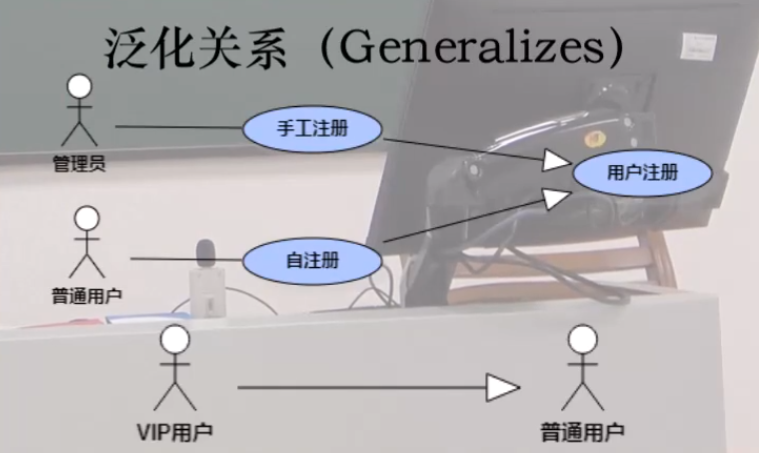
—————————————————————————————————————

用例的关系：



虚线箭头：表示依赖的线

<<stereotype>>（相当于盖章，改变原来符号的关系）：比如<<include>>是包含关系用在主成功场景，<<extend>>用在扩展场景



泛化：指在用例图的元素之间存在着继承关系（继承父类的特性）

如VIP用户继承与普通用户，z在此之上多了特点的专属用例

—————————————————————————————————————

系统交互建模

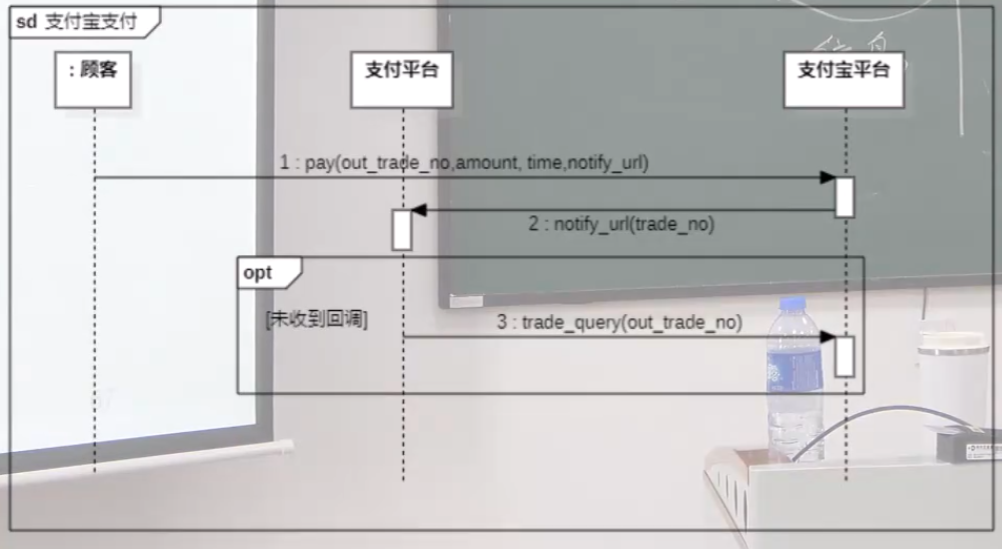
用例从人（主参与者）与系统之间的交互中得来，因为辅助参与者属于用例的内部实现，所以一般不会出现在用例的文字中，所以系统顺序图（描述系统和辅助参与者信息和需求的流动）应运而生。

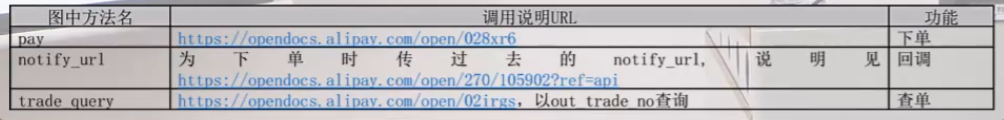
用例描述的是人和系统的行为以及信息的需求

信息：人输入到系统或者系统展现出来的，行为：主要是人在面对系统返回的信息所做的抉择，或者系统返回给人内容时的反馈方式

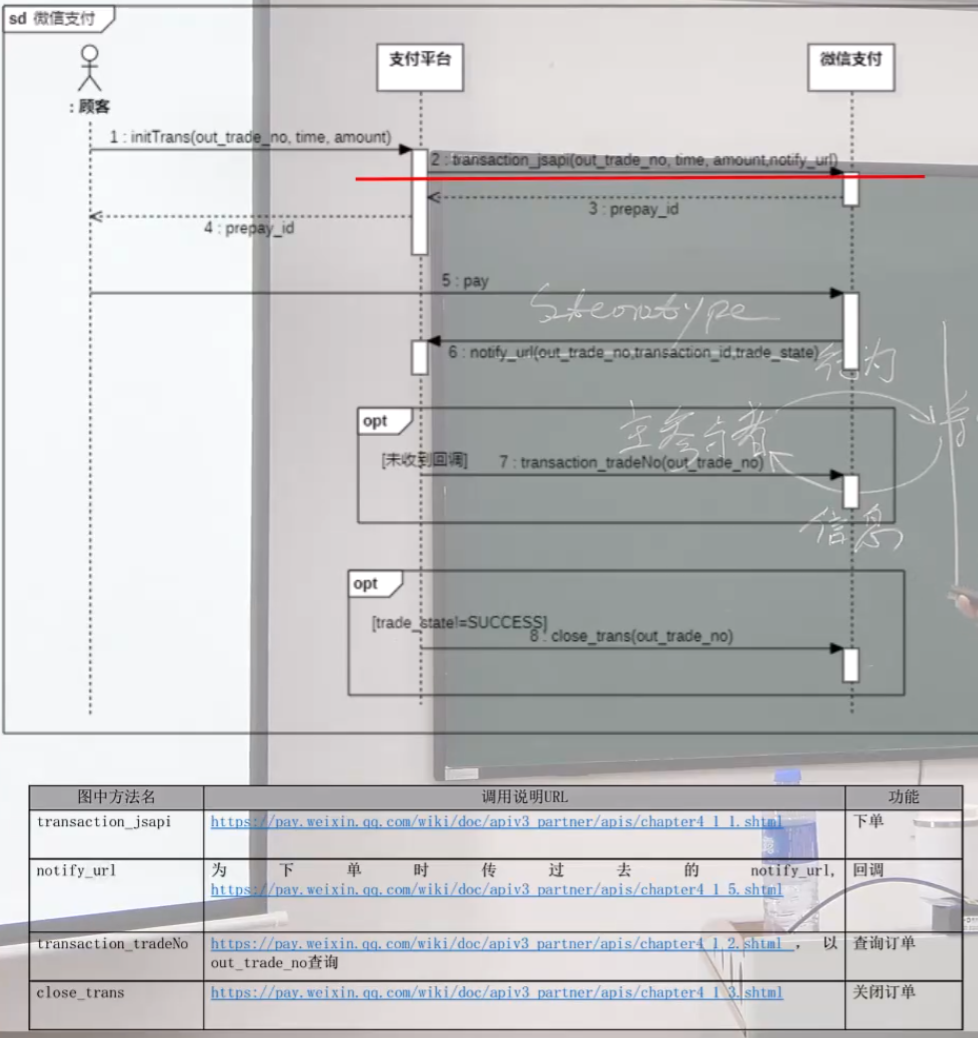
支付的系统顺序图：

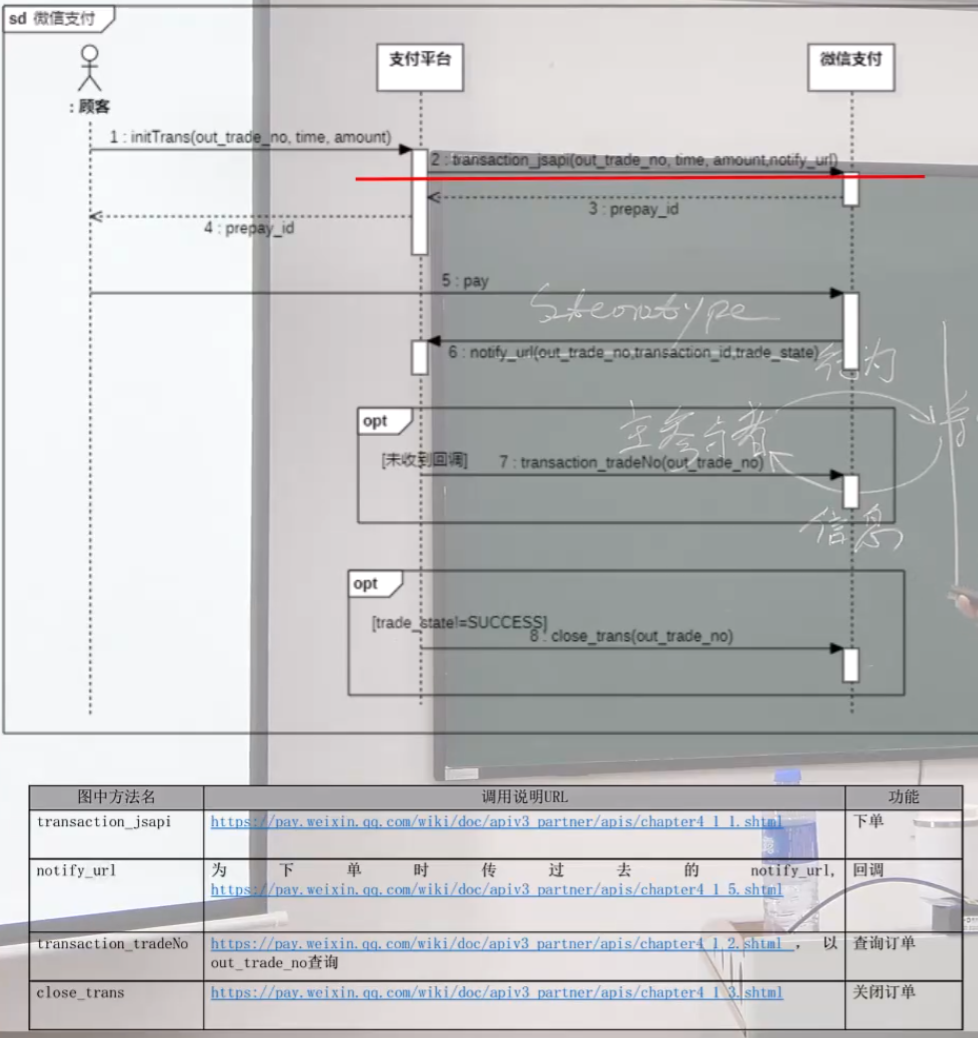
支付宝





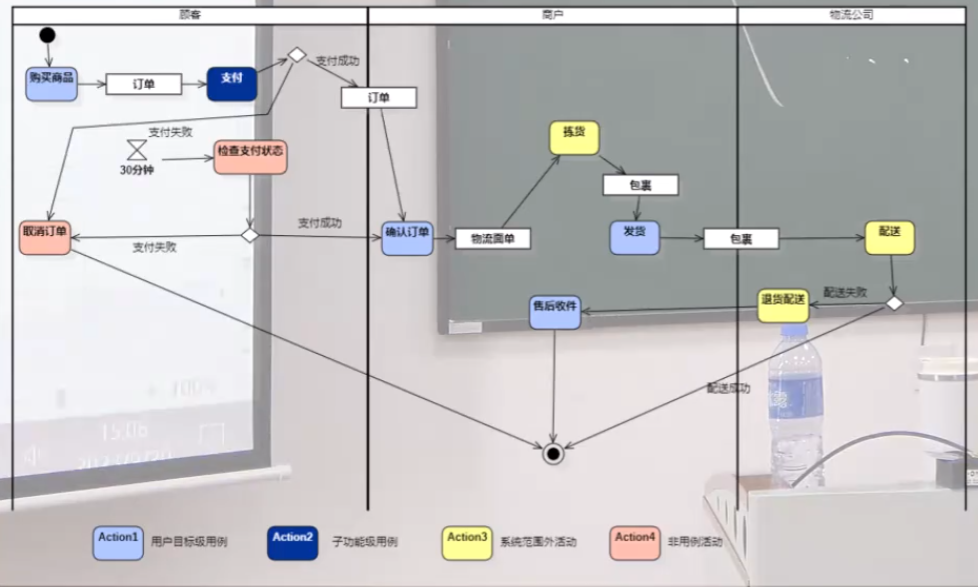
微信





—————————————————————————————————————

业务流程建模



使用活动图（activity diagram）来建模，表示最基本的用例，用例外的东西，系统的东西等等

活动图的基本动作action（圆角方框）原来表示用例之间的关系/分类/级别

对象节点object node（方框）表示一个用例结束后会有一个标志性的东西，如订单传递给下一个用例

决策节点decision node（菱形）表示在一个用例中做的决策产生不同的结果，根据结果转到不同的用例