实验八 软件需求规格说明SRS（3）

实验目的：

1. 学习Petri网基本知识及如何应用Petri网建模

2. 练习用各种动态建模工具 (状态图、Petri网、数据流图、OCL逻辑等）对所负责的项目进行建模

3. 完善自己项目的SRS

实验内容：

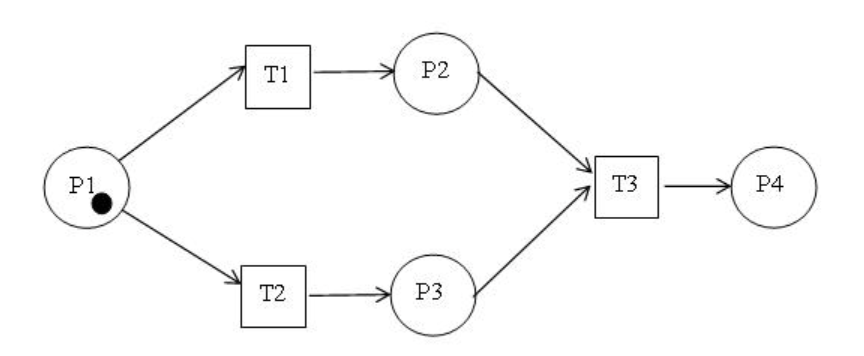
1. 阅读“SYSTEM MODELLING WITH PETRI NETS”，进一步学习Petri网知识，了解如何应用Petri网对系统进行建模（李雨桐）

**佩特里网**（英语：Petri net），是对离散并行系统的[数学](https://zh.wikipedia.org/wiki/%E6%95%B0%E5%AD%A6" \o "数学)表示。佩特里网属于[离散事件动态系统](https://zh.wikipedia.org/wiki/%E9%9B%A2%E6%95%A3%E4%BA%8B%E4%BB%B6%E5%8B%95%E6%85%8B%E7%B3%BB%E7%B5%B1" \o "离散事件动态系统)，是1960年代由[卡尔·亚当·佩特里](https://zh.wikipedia.org/wiki/%E5%8D%A1%E5%B0%94%C2%B7%E4%BA%9A%E5%BD%93%C2%B7%E4%BD%A9%E7%89%B9%E9%87%8C" \o ")发明的[[1]](https://zh.wikipedia.org/wiki/%E4%BD%A9%E7%89%B9%E9%87%8C%E7%B6%B2" \l "cite_note-1)，适合于描述异步的、并发的计算机系统模型。佩特里网既有严格的数学表述方式，也有直观的图形表达方式。

由于佩特里网能表达并发的事件，被认为是[自动化理论](https://zh.wikipedia.org/w/index.php?title=%E8%87%AA%E5%8A%A8%E5%8C%96%E7%90%86%E8%AE%BA&action=edit&redlink=1" \o "自动化理论（页面不存在）)的一种。研究领域趋向认为佩特里网是所有流程定义语言之母。

**经典佩特里网**

经典的佩特里网是简单的过程模型，由两种节点：库所和变迁，有向弧，以及令牌等元素组成的。



**结构**

佩特里网的元素：

**库所**（Place）圆形节点

**变迁**（Transition）方形节点

**有向弧**（Arc）是库所和变迁之间的有向弧

**令牌**（Token）是库所中的动态对象，可以从一个库所移动到另一个库所。

佩特里网的**规则**是：

有向弧是有方向的

两个库所之间或两个变迁之间不允许有弧

库所可以拥有任意数量的令牌

**行为**

如果一个变迁的每个**输入库所**（input place）都拥有数量足够的令牌时，该变迁即为**被允许**（enable）。一个变迁被允许时，变迁将**发生**（fire），**输入库所**（input place）的令牌被消耗，同时为**输出库所**（output place）产生令牌。

注意：

变迁的发生时是独立且唯一的，也就是说，没有一个变迁只发生了一半的可能性。

有两个或多个变迁都被允许的可能，但是一次只能发生一个变迁。这种情况下变迁发生的顺序没有定义。

如果出现一个变迁，其输入库所的个数与输出库所的个数不相等，令牌的个数将发生变化，也就是说，令牌数目不守恒。

佩特里网络是静态的，也就是说，不存在发生了一个变迁之后忽然冒出另一个变迁或者库所，从而改变Petri网结构的可能。

佩特里网的状态由令牌在库所的分布决定。也就是说，变迁发生完毕、下一个变迁等待发生的时候才有确定的状态，正在发生变迁的时候是没有一个确定的状态的。

两个变迁争夺一个令牌的情形被称之为**冲突**。当发生冲突的时候，由于佩特里网的时序是不确定的，因此具体哪个变迁得以发生也是不确定的。实际应用中，往往需要避免这种情形。用于描述现象的佩特里网也可能自然出现冲突，这表明我们对于变迁发生的条件没有完全了解。

多个弧连接两个节点的情况。在输入库所和变迁之间的弧的个数决定了该变迁变为被允许需要的令牌的个数。弧的个数决定了消耗／产生的令牌的个数。

**佩特里网的形式化定义**

一个经典的佩特里网由四元组（库所，变迁，输入函数，输出函数）组成。任何图都可以映射到这样一个四元组上，反之亦然。

被允许的形式化 变迁发生的形式化佩特里网 到变迁系统的映射 可达性图

佩特里网是一个三元组(P,T,F)

F(P X T)U(T X P)是弧的集合

**流程建模**

一个流程的**状态**是由在场所中的令牌建模的，状态的**变迁**是由变迁建模的。令牌表示事物（人，货物，机器），信息，条件，或对象的状态；库所代表库所，通道或地理位置；变迁代表事件，转化或运输

一个流程（Flow）有当前状态，可达状态，不可达状态。

**经典Petri网的局限性**

没有测试库所中零令牌的能力

模型容易变得很庞大

模型不能反映时间方面的内容

不支持构造大规模模型，如自顶向下或自底向上

**如何应用Petri网对系统进行建模**

1.确定系统的组成部分：首先，确定系统中的各个组成部分，包括对象、过程、事件等。这些部分将成为Petri网中的位置或变迁。

2.识别位置和变迁：将系统中的实体映射到Petri网的位置和变迁。位置表示系统中的状态，而变迁表示系统中的事件或操作。

3.定义初始状态：确定系统的初始状态，并在Petri网中标记相应的位置。

4.确定变迁触发条件：对于每个变迁，确定触发它所需的条件。这些条件可以是位置中的标记、时间限制等。

5.建立连接关系：使用有向边将位置和变迁连接起来，表示它们之间的关系。这些边称为弧，它们可以是单向的或双向的，表示从一个状态到另一个状态的转换。

6.定义变迁的动作：对于每个变迁，定义它所触发的动作或事件。这些动作可以包括状态的改变、标记的增加或减少等。

7.分析系统行为：使用Petri网模型进行分析，例如模拟系统的行为、检测死锁或冲突等。

8.验证模型：验证Petri网模型是否准确地反映了系统的行为，并对其进行调整或修正。

9.优化模型：根据需要，对Petri网模型进行优化，以提高系统的性能或效率。

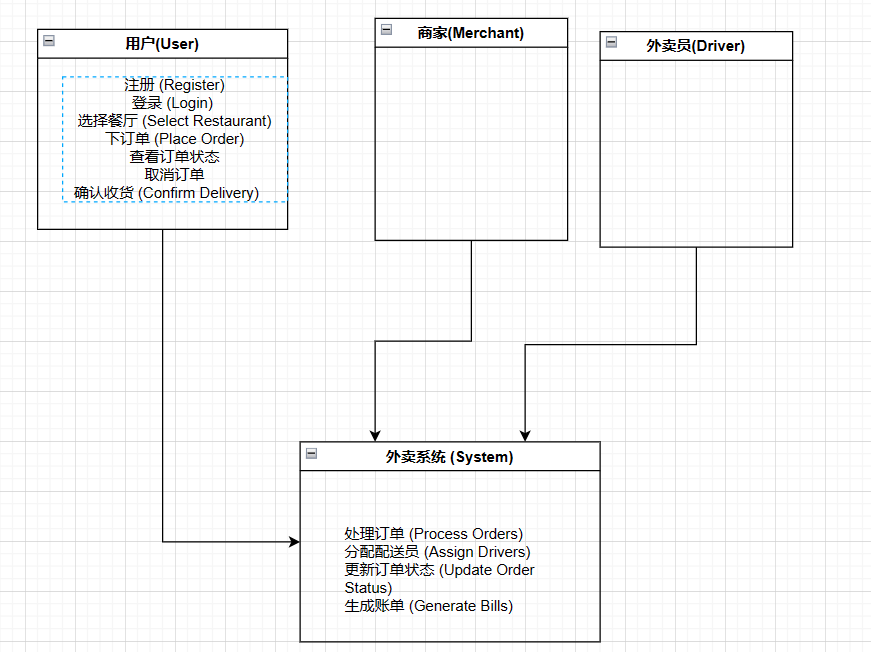
2. 针对各自负责项目的不同场景，练习用各种动态建模工具 (状态图(张勇）、Petri网（刘易）、数据流图（王攀）、OCL逻辑等（丁弘扬））建模需求，与用户沟通。

参考第4章，4.12 皮卡迪里电视广告售卖系统，分析、归纳、总结出符合实际的需求规格。

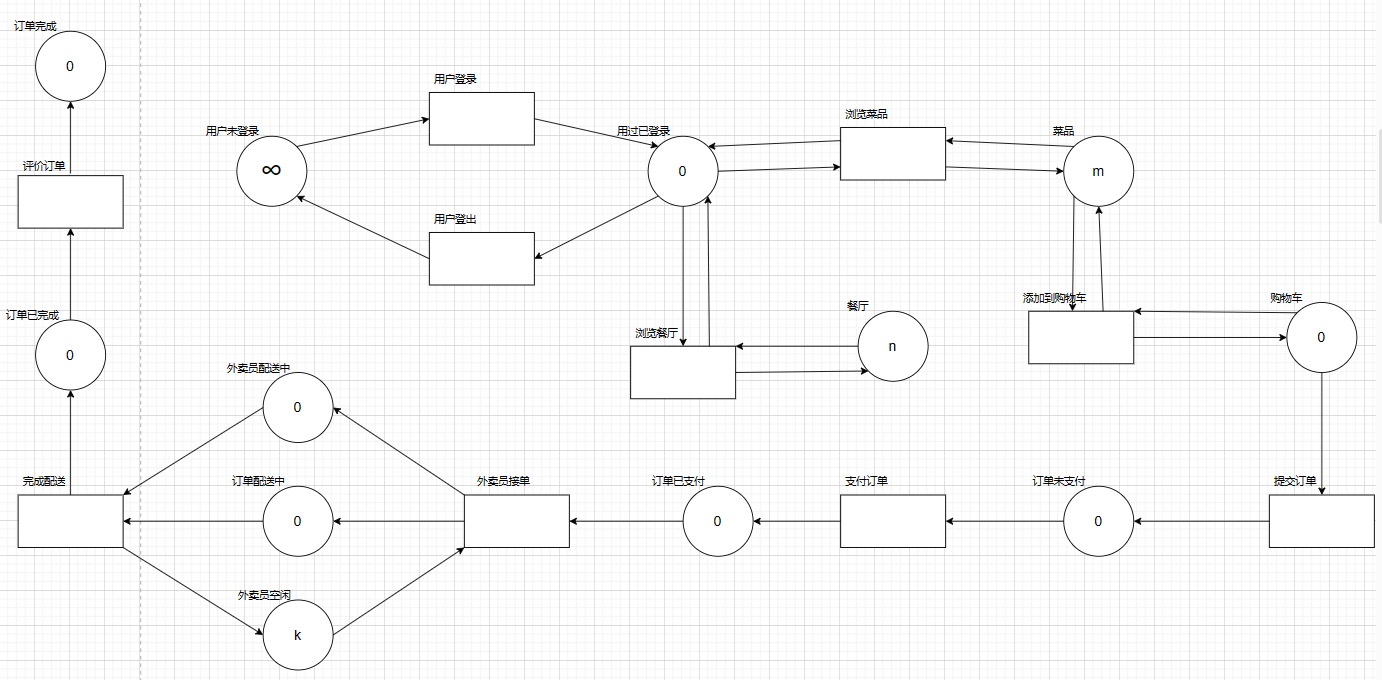
**状态图：**

外卖系统的状态图通常描述了系统中不同实体（如用户、商家、配送员）之间的交互以及系统在不同情境下的行为。

这个状态图显示了用户、商家和配送员之间的交互以及系统的行为。用户可以注册、登录、选择餐厅、下订单、查看订单状态和确认收货。商家可以处理订单和更新订单状态。配送员可以接受订单并更新订单状态。外卖系统负责处理订单、分配配送员、更新订单状态和生成账单等功能。



**Petri网**

****

**数据流图**

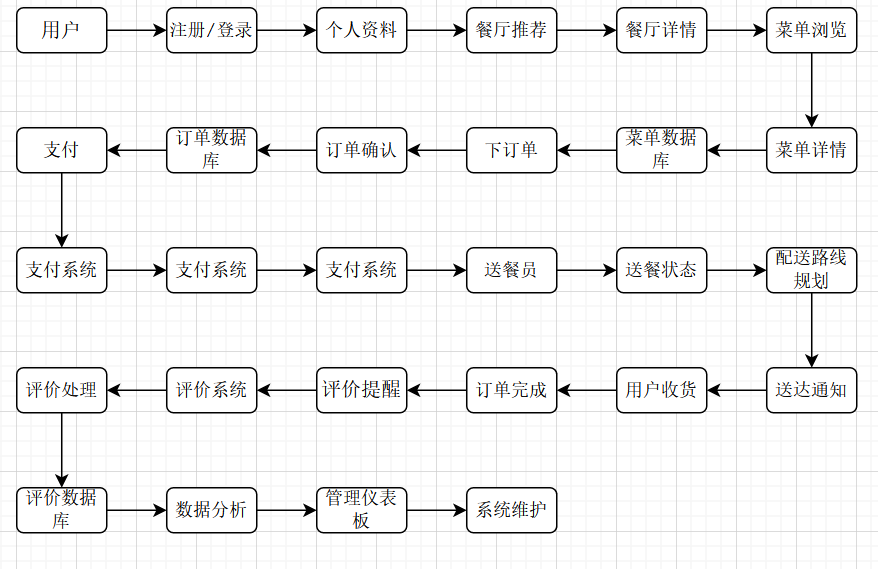
用户---->注册/登录---->个人资料---->餐厅推荐---->餐厅详情---->菜单浏览

---->菜单详情---->菜单数据库---->下订单---->订单确认---->订单数据库

---->支付---->支付系统 ---->支付系统---->支付系统---->送餐员---->送餐状态---->配送路线规划---->送达通知---->用户收货---->订单完成---->评价提醒

---->评价系统---->评价处理---->评价数据库---->数据分析---->管理仪表板

---->系统维护

****

**OCL逻辑**

1.定义业务实体和属性：

用户，餐厅，菜品，订单等。用户实体可能具有属性如用户名，电话号码，地址等。

2.识别约束条件：

一个用户只能有一个默认地址。

一个订单必须包含一个或多个菜品。

菜品的库存数量不能小于零。

餐厅的唯一标识

3.编写OCL约束表达式：

用户信息的真实性：

context User inv : self.name <> null and self.phoneNumber.isValid()

用户默认地址的约束：

context User inv : self.defaultAddress -> forAll ( a : Address | not exists ( anotherA:Address | anotherA <> a and anotherA.user = self))

账户安全：

context User inv ：self.password <> self.inputPassword

订单行为:

context Order inv : self.user.isValid() and self.quantity > 0

订单包含菜品的约束：

context Order inv : self.dishes->size() > 0

订单时间合理性：

context Order inv : self.orderTime.between(self.restaurant.openTime , self.restaurant.closeTime)

菜品库存数量的约束：

context Dish inv : self.stock > 0

餐厅的唯一标识：

context Order inv : self.restaurantId->isUnique()

餐厅状态约束：

context Restaurant inv: self.status in {‘Open’,’Closes’}

营业时间约束：

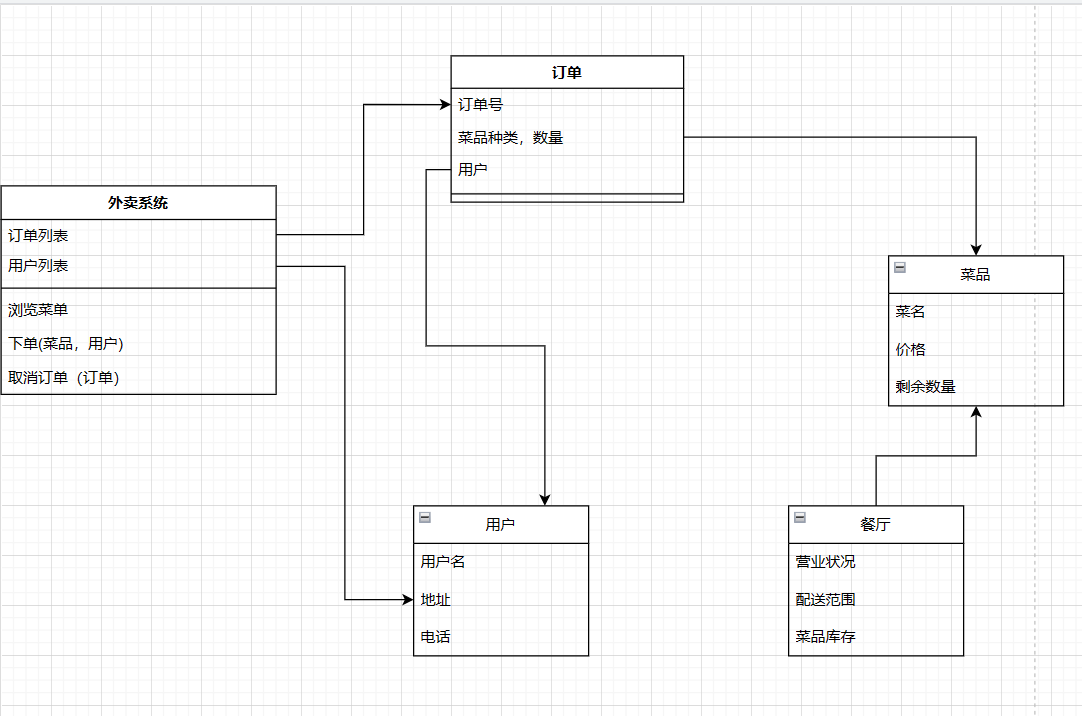
context Restaurant inv : selfopenTime<selfcloseTime

餐厅配送范围约束：

context Restaurant inv : allOrdersWithinDeliveryRange

餐厅库存限制：

context Dish inv : self.stock<=self.maxstock

****

3. 分工协作，用上面的工作补充完善SRS和所负责的项目。

项目跟踪，建立能反映项目及小组每个人工作的进度、里程碑、工作量的跟踪图或表，将其保存到每个小组选定的协作开发平台上，每周更新。