点餐系统项目文档

小组成员 刘燕 吕杉 蔡福兵

指导教师 龚伟

起止时间 2015 年 1 月-2015 年 1 月

重庆师范大学软件工程系

摘要

餐饮传统的点菜方式是纯手工操作，由服务员记录顾客点的菜，在具体工作中容易出现一下问题：手写单据字迹潦草从而导致上错菜、传菜分单出错现象严重、加菜和查账程序较繁琐。处理特殊口味有遗漏和偏差、客人催菜遗忘现象较繁琐、计算账单易出错、不方便人员管理等等。

内容目录

第1部分 需求

1.1. 项目驱动

1.1.1. 项目目标

1.1.1.1. 项目背景

餐饮传统的点菜方式是纯手工操作，由服务员记录顾客点的菜，在具体工作中容易出现以下问题：手写单据字迹潦草从而导致上错菜、传菜分单出错现象严重、加菜和查账程序较繁琐。处理特殊口味有遗漏和偏差、客人催菜遗忘现象较繁琐、计算账单易出错、不方便人员管理等等。

1.1.1.2. 项目目标

目标：提高客人的点餐速度、简化点餐操作和方便餐馆信息管理。

好处：方便餐馆管理和客人能够快速点餐。

测量标准：在餐馆人员管理方面操作简单，客人点餐效率更高，出错率下降。

1.2. 利益相关者

1.2.1. 客户

餐馆经理，决定是否接受产品。

1.2.2. 顾客

餐馆经理，决定是否购买产品。

1.2.3. 其他利益相关者

服务员、厨师、客人

核心团队：

需求获取者：蔡福兵、吕杉、刘燕

分析者：刘燕、吕杉、蔡福兵

设计者：吕杉、蔡福兵、刘燕

1.2.4. 产品的直接操作用户

餐馆经理：负责整个餐馆的人

服务人员：为客人服务直接和客人交流的人

客人：来餐馆就餐的客人

厨师：为客人做出美味饭菜的人

1.3. 强制的限制条件

1.3.1. 解决方案的限制条件

描述：产品应该使用 windows 系统和安卓系统。

理由：客人使用手持设备点餐更快捷；管理者使用熟悉的 windows 系统将会方便管理餐馆信息。

验收标准：产品经过测试人员测试，可以在 windows 上面正确运行，在安卓上面可以准确下订单。

1.3.2. 预期的工作地点环境

产品将用在小型餐饮店中，它必须保证无差错并且能够实时处理客人的订单。

产品有可能安装在手持设备上面，可以通过手持设备进行点餐。

产品安装在 windows 系统上面，管理者方便管理餐馆信息。

1.3.3. 进度计划条件限制

最后的期限：产品应该在 2015 年 6 月之前完成。

理由：客户要求。

影响：如果没有完成产品，将会超过预算。

1.4. 命名惯例和定义

1.4.1. 定义利益相关者在项目中使用的所有术语，包括同义词

下为术语表：

menu（菜单）	包含一个及以上 **dishInformation** 的列表。dishInformation（菜品信息），包含菜名、对应图片、菜品编号、所属菜系、用料、口味、制作方法、评分、单价。
菜品状态	表示一道菜品是否已经烹饪，如果烹饪完成，标记为完成，否则，标记为未完成。
order(订单)	包含 **Customer** 选中的菜品和对应的菜品数量、每个菜品的单价、菜品总数、价格总计、**桌号**、生成日期和时间、订单号、订单状态。

Customer（顾客）	来餐馆用餐的客人,通过点餐系统点餐，生成**订单**。
Chef（厨师）	烹饪来自 **Customer** 的**订单**上的菜的人，可以对**菜品状态**进行修改。
Waiter（服务员）	协助 **Customer** 进行点餐下**订单**的人。
桌号	在餐馆内部对每个桌子进行唯一编号，以确定 **Customer** 的位置。
评价用餐	**Customer** 对本次用餐的菜的喜好程度进行打分（5 分制）。
订单状态	表示**订单**是否已经支付。

1.5. 相关事实和假定

1.5.1. 事实

来餐馆用餐的人需要下**订单**，**Chef** 需要根据 **Customer** 的**订单烹饪**菜。

1.5.2. 业务规则

来餐馆的 **Customer** 用餐之前要下**订单**，希望 **Chef** 和 **Waiter** 能准确理解 **Customer** 的意思，做出美味的饭菜。

1.5.3. 假定

Customer 可能需要变更**订单**。

Chef 收到的**订单**需要和 **Customer** 下的**订单**必须一致。

订单信息必须真实准确。

Waiter 确认**订单**之后 **Chef** 必须立即收到**订单**。

1.6. 工作的范围

1.6.1. 工作的上下文范围

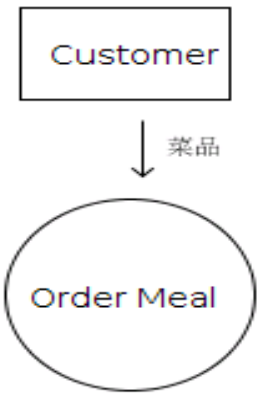

图 1: 点餐上下文范围

1.6.2. 工作切分

下为工作切分表:

事件名称	输入和输出	BUC 小结
1. **Customer** 想要点餐	菜品（入）	生成**订单**，烹饪菜

1.6.3. 确定业务用况

1.6.3.1. Order Meal 业务用况模型浏览

1. **参与者类别**

   1. 图 2 给出了“Order Meal”业务用况模型中的所有参与者

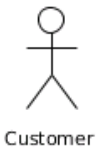

图 2: “Order Meal”中的参与者

2. **Customer**

Customer 在“Order Meal”上执行事务。参与者可以点餐下**订单**并支付**订单**。

2. **用况类别**

1. 主要用况

图 3 给出了“Order Meal”业务用况模型中的主要用况。

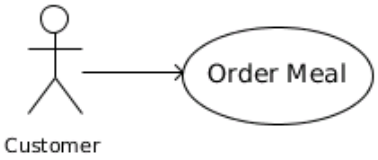

图 3: “Order Meal”中的主要业务用况

2. 简略描述

该用况描述了 **Customer** 如何使用 Order Meal 来点餐。

1.6.3.2. 业务用况描述——Order Meal

1. 简略描述

该用况描述了 **Customer** 如何使用 Order Meal 来点餐。

2. 用况图

参见图 4

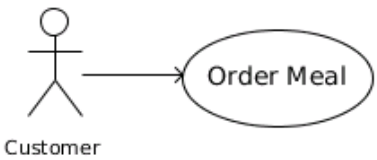

图 4: Order Meal 业务用况的用况图

3. 前置条件

- 餐馆正在营业。

4. 基本流

    1. 当参与者 **Customer** 选择点菜时，启动用况。
    2. **Customer** 浏览**菜单**并选择喜欢的菜品。
    3. Order Meal 记录下 **Customer** 选择的菜，生成**订单**。
    4. **Customer** 选择支付**订单**。
    5. Order Meal 记录**订单**已经支付。
    6. 用况终止。

5. 成果

    Customer 点餐后生成**订单**并且已经支付完**订单**。

1.7. 业务数据模型和数据字典

1.7.1. 数据模型

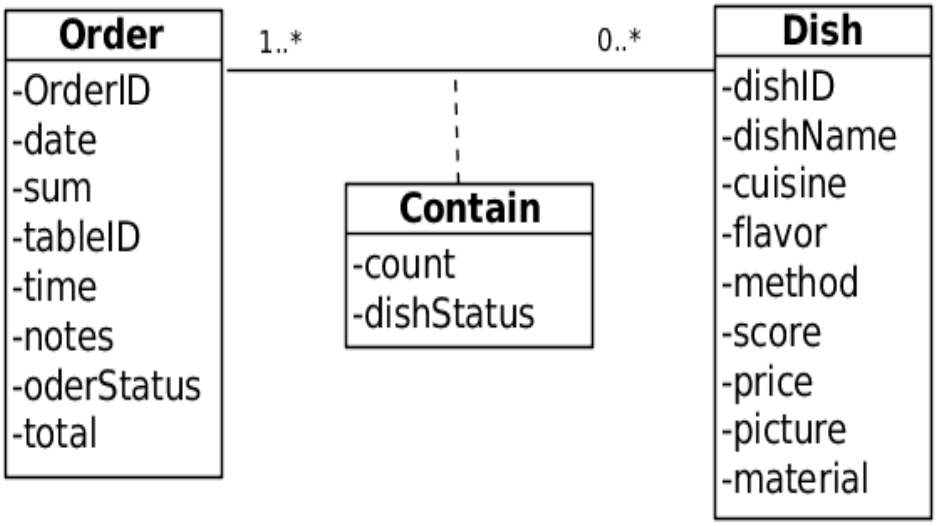

图 5: 数据模型

1.7.2. 数据字典

下为数据字典表:

名称	定义	类型
Dish	菜单上面的菜。保存了菜品名字、编号、所属菜系、口味、制作方法、得分、单价的详细信息	Class
Order	记录一个 **Customer** 在餐馆点餐后的信息。保存了订单号、日期时间、菜品总数、桌号、总价、订单状态的信息	Class
Contain	保存订单上面每个菜品的数量和状态	Association/Class
dishName	菜品的名字	Attribute/element
dishID	菜品的编号	Attribute/element
cuisine	菜系：川菜、苏菜、鲁菜、浙菜、湘菜、闽菜、粤菜、皖菜	Attribute/element
flavor	口味，包括酸、甜、苦、辣	Attribute/element
method	烹饪一道菜的方法	
score	**Customer** 评分后的平均得分	Attribute/element
price	每个菜品的单价	Attribute/element
dishStatus	状态表示当前菜品是否完成，由 **Chef** 来判断后更改	Attribute/element
notes	备注，对**订单**的特殊说明信息	Attribute/element
material	烹饪这道菜品所用到的原料	Attribute/element
data	生成**订单**的日期年/月/日 时/分/秒）	Attribute/element
time	生成**订单**的时间（ 时/分/秒）	Attribute/element
sum	菜品总数量	Attribute/element
tableID	**桌号**，确认 **Customer** 的位置	Attribute/element
count	记录对应菜品的数量	Attribute/element
total	**订单**价格总计	Attribute/element
orderStatus	**订单**状态，表示是否已经支付	Attribute/element
picture	菜品对应的图片	Attribute/element

1.8. 产品范围

1.8.1. 产品边界

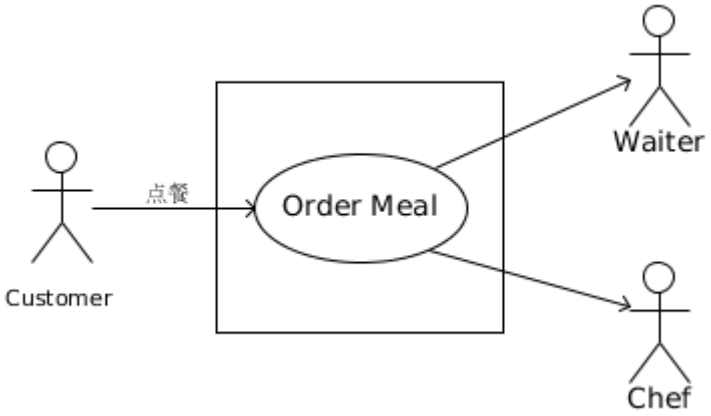

图 6: 为产品用况Order Meal 的边界图

Order Meal 产品用况图，展示了产品边界（矩形框）之外的参与者 Customer。这个产品用况回溯到 Order Meal(BUC)。

1.8.2. 产品用况清单

下为产品用况情况:

PUC 编号	**PUC 名称**	**参与者**	**输入和输出**
1	Order Meal	**Customer**、**Waiter**、**Chef**	菜品（入） **桌号**（入） **订单**（出）

1.8.3. 单个产品用况

1.8.3.1. Order Meal 产品用况模型浏览

1. **参与者类别**

图 7 给出了“Order Meal”产品用况模型中的所有参与者

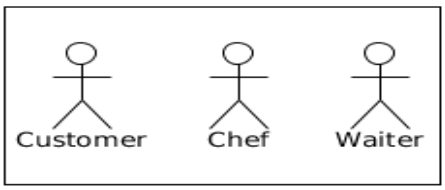

图 7: 产品用况“Order Meal”的所有参与者

1. **Customer**：

Customer 在 Order Meal 上点餐，最后支付**订单**。

2. **Chef**：

Chef 为 Order Meal 提供服务。**Chef** 负责根据**订单**烹饪菜给 **Customer** 和对**订单** 中菜品状态的更改。

3. **Waiter**：

Waiter 协助 **Customer** 点餐，在 **Customer** 完成**订单**之后，输入**桌号**来确认**订　　单**。

2. 用况类别

1. 主要用况

图 8 给出了“Order Meal”产品用况模型中的主要用况。

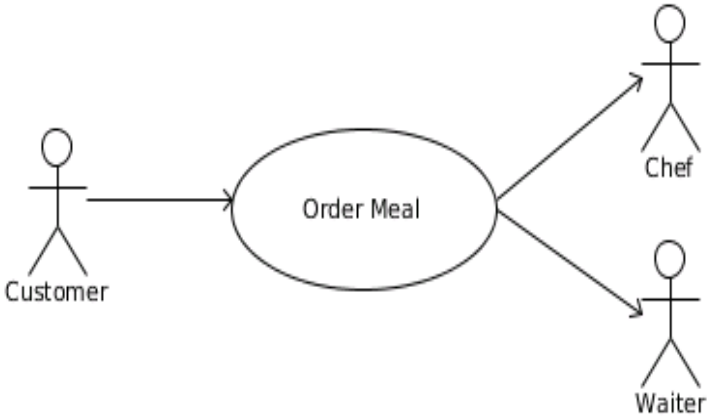

图 8: 产品用况“Order Meal”的主要用况

2. 简略描述

该用况描述参与者 **Waiter** 协助参与者 **Customer** 如何使用 Order Meal System 点餐，

最后系统将**订单**发送给参与者 **Chef** 处理。

1.8.3.2. 产品用况描述——Order Meal

1. 简略描述

   该用况描述参与者 **Waiter** 协助参与者 **Customer** 如何使用 Order Meal System 点餐,最后系统将**订单**发送给参与者 **Chef** 处理。

2. 用况图

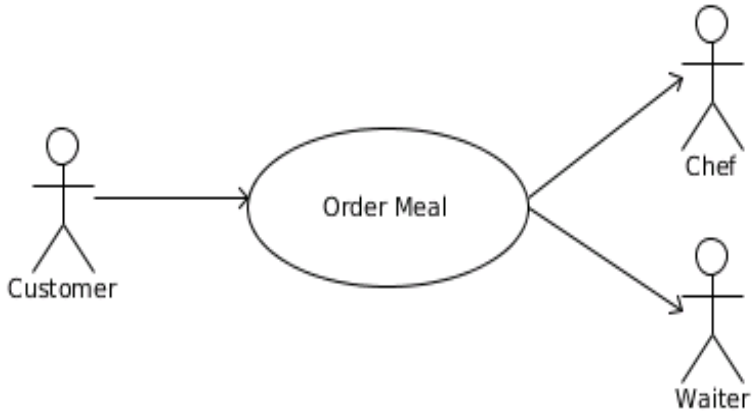

图 9: "Order Meal"的用况图

3. 前置条件

   - 各个点餐系统设备的通信正常，可以正常使用。

4. 基本流

   1. 当参与者 **Customer** 想要点餐时，开始启动用况。
   2. 系统 Order Meal System 显示**菜单**。

      {选择菜品}

   3. **Customer** 浏览**菜单**并选择菜品。
   4. 系统显示 **Customer** 选择的菜品的信息和所有菜品的总价。
   5. **Customer** 确定选择完成。

{完成下单}

6. 系统显示**桌号**列表，提示参与者 **Waiter** 选择**桌号**。

7. **Waiter** 选择**桌号**。

8. 系统生成**订单**并发送**订单**信息给厨房的系统。

9. 参与者 **Chef** 标记**菜品状态**为完成。

10. 系统记录变更后的**菜品状态**。

{买单}

11. **Customer** 选择支付**订单**。

12. 执行子流*“买单”*。

{完成买单}

13. 用况终止。

5. 备选流

特定备选流:

*A1. 选择**评价用餐***

在**{完成买单}**扩展点上，如果 **Customer** 想要**评价用餐**，则:

1. **Customer** 选择评价。

2. 系统显示**订单**上面的菜品，提示 **Customer** 评价。

3. **Customer** 选择分数。

4. 系统记录分数，并更新菜品的平均得分。

5. 事件流恢复下一步。

普适备选流

A2. 菜品排序

在事件流的任何位置，当 **Customer** 想要以不同的方式查看**菜单**上的菜品时，则:

1. **Customer** 选择排序方式（口味、推荐、评分高低）。

2. 系统按该方式进行排序显示。

3. 事件流恢复到中断之前的位置。

有界备选流:

A3. 查询已选菜品

在事件流**{选择菜品}**和**{完成买单}**扩展点之间的任何位置，当 **Customer** 想要查看已选菜品时，则:

1. **Customer** 选择查看**订单**。
2. 系统显示**订单**信息。
3. **Customer** 选择返回。
4. 事件流恢复到中断之前的位置。

A4. 取消菜品

在事件流**{完成下单}**和**{买单}**扩展点之间的任意位置，如果 **Customer** 想要取消菜品，则:

1. **Customer** 选择查看**订单**。
2. 系统显示**订单**信息。
3. 如果 **Cusotmer** 想取消的菜品没有被 **Chef** 标记为完成，则:
    a. **Customer** 选择取消菜品。
    b.系统提示确认取消。
    c.如果 **Customer** 选择确认。
        I.系统取消**订单**上 **Customer** 选择取消的菜品。
        II.系统更新当前**订单**信息。
4. 如果 **Cusotmer** 想取消的菜品被 **Chef** 标记为完成
    a.系统提示 **Customer** 不能取消菜品。
5. 事件流恢复到中断之前的位置。

A5. 添加菜品

在事件流**{完成下单}**和**{买单}**扩展点之间的任意位置，如果 **Customer** 想要添加菜品，则：

1. **Customer** 浏览**菜单**并选择菜品。
2. 系统添加菜品到**订单**中，并更新显示。
3. 事件流恢复到中断之前的位置。

6. 子流：

S1. *买单*

1. 系统显示**订单**。
2. **Waiter** 确认支付**订单**。
3. 系统更改**订单状态**为已支付，存档并打印**订单**。

   {完成买单}
4. 恢复到下一步。

7. 后置条件：

- 系统产生一条**订单**记录，并提醒了 **Chef** 有新**订单**产生。
- 系统没有产生任何新**订单**。

1.9. 功能需求与非功能性需求

下为功能性需求与非功能需求：

需求类型	描述	理由	验收标准	来源
功能需求	“Order Meal”应该能够显示菜单	以供“Customer”浏览菜单	菜单正常显示出所有菜品	用况“Order Meal”
功能需求	“Customer”应该能够根	以供“Cus-	菜单上的	用况

	据“Order Meal”的菜单选择菜品	tomer”选中菜品	菜品“Customer”可以选择	“Order Meal”
功能需求	“Order Meal”能够计算并显示“Customer”选择的菜品总价	方便“Customer”能够查询当前选择的菜品总价	“Customer”选择菜品后，能够查看总价	用况“Order Meal”
功能需求	“Waiter”能够选择相应“Customer”的桌号	协助“Customer”生成订单	订单上面能够查看到Waiter桌号	用况“Order Meal”
功能需求	“Order Meal”能够发送订单到厨房的设备	使“Chef”能够浏览订单并根据订单烹饪菜	“Chef”能够浏览到订单内容	用况“Order Meal”
功能需求	“Chef”能够修改订单上面菜品的状态	为了让“Customer”能够标记当前出现的菜是否已经烹饪	Chef能够修改菜品状态	用况“Order Meal”
功能需求	“Customer”能够查看订单信息	使“Customer”能够确认自己的消费信息	“Customer”能够看到自己本次消费的订单信息	用况“Order Meal”
功能需求	“Customer”能够取消选择的菜品	“Customer”不喜欢之前选择的菜品，想要取消	“Customer”能够取消已经选择的菜品	用况“Order Meal”
功能需求	“Order Meal”能够计算所有“Customer”的评分的平均值	每个菜品可以有“Customer”评分的	对某一个菜品进行评价后，有能改变	用况“Order Meal”

		平均值	该菜品的评分结果	
功能需求	“Customer”能够添加菜品	“Customer”想要添加菜品	订单上面已经添加了“Customer”的菜品	用况“Order Meal”
观感需求	产品应该吸引人	让使用者有很好的视觉体验，喜欢用我们的产品	有70%的人在第一次使用该产品时都想去喜欢上了它	
易用性需求	未经培训的人第一次尝试使用该产品时，它应该易于使用	这是一个新产品，我们都愿意使用它	使用者中90%的人在第一次使用该产品时，能在自己的独立情况下完成点餐	
易用性需求	产品应该用起来令人愉快	我们希望人们喜欢产品的使用，这样他们就会继续使用我们的产品	使用我们产品的人越来越多	
性能需求	响应应该足够快，实时性高	避免因产品反应过慢，通信不实时，使“Customer”产生厌烦	在95%的情况下，响应时间将不超过0.5s，在其他情况下不超过2s	
容量需求	产品能够支持500个使用者使用	避免使用人数过多导致	产品能够支持500个	

		服务器无法正常工作	“使用者同时使用	
可靠性和可访问性	产品的无故障运行时间应该达到99%	避免经常崩溃是用户不愿再使用产品	产品的无故障运行时间应该达到99%	
产品化需求	产品应该能够由未培训过的用户安装，不需要参考独立打印的安装指南	避免产品安装过于复杂用户不会安装而放弃使用	99%的用户第一次下载安装的时候能够成功安装	
安全需求	产品应该确保用户的信息只能被授权的用户访问	避免“用户的信息被未授权的人访问	产品应该确保用户的信息只能被授权的用户访问	

第2部分 分析

2.1. 提取分析类图

2.1.1. 用况“Order Meal”类图

2.1.1.1. 基本流分析参与者与系统响应

事件流	
参与者输入	系统响应
系统响应	
	系统显示菜单
Customer 选择菜品	
	系统显示选择的菜品
Customer 确认完成选择	
	系统显示桌号
Waiter 选择桌号	
	系统生成订单，发送订单给厨房系统
Chef 标记菜品状态为完成	
	系统记录菜品状态
Customer 选择支付订单	
	系统显示订单
Waiter 确认支付订单	
	系统更改订单状态为已支付，并保存订单

2.1.1.2. 提取类

边界类：点餐界面（OrderMealUI）、烹饪界面（CookUI）。

控制类：点餐（OrderMeal）。

实体类：菜品（Dish）、订单（Order）。

2.1.2. 用况“Order Meal”协作图

根据参与者与系统的响应，得到用况的协作图：

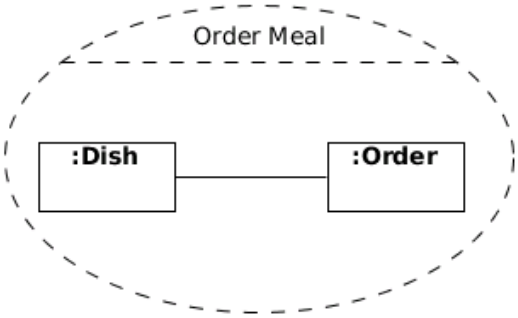

图 10: “Order Meal”的协作图

2.1.3. 用况“Order Meal”基本流的通信图

由协作图根据系统和参与者之间的交互得到得到通信图：

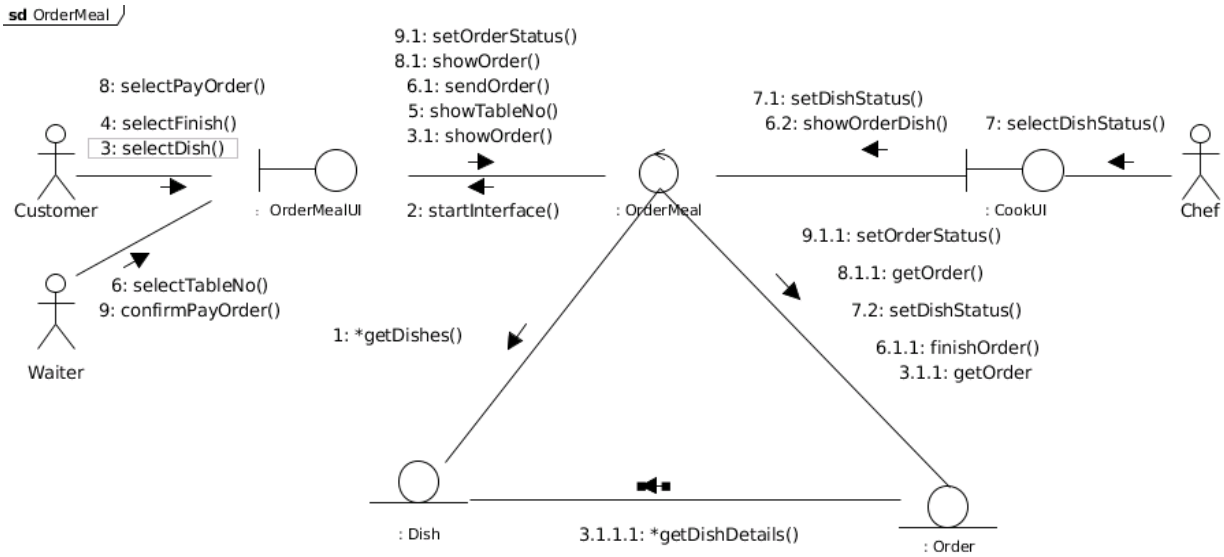

图 11: 用况“Order Meal"基本流的通信图

2.1.4. 顺序图

2.1.4.1. 用况“Order Meal”基本流的顺序图

将图 11 转换为顺序图为：

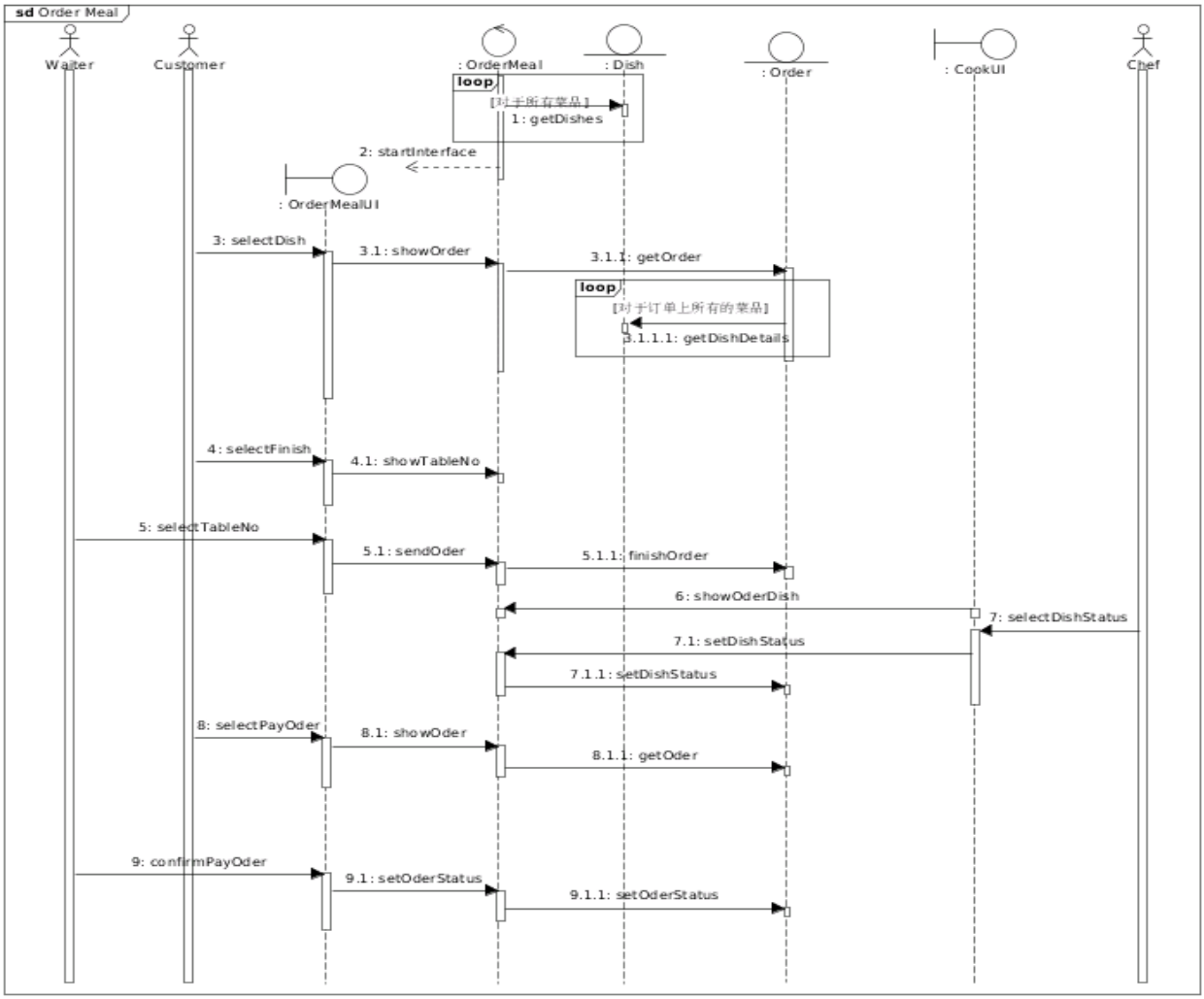

图 12: 用况“Order Meal”基本流的顺序图

2.1.4.2. 顺序图管理

将图 12 使用交互使用来管理顺序图，“买单”被顺序图“买单”引用。

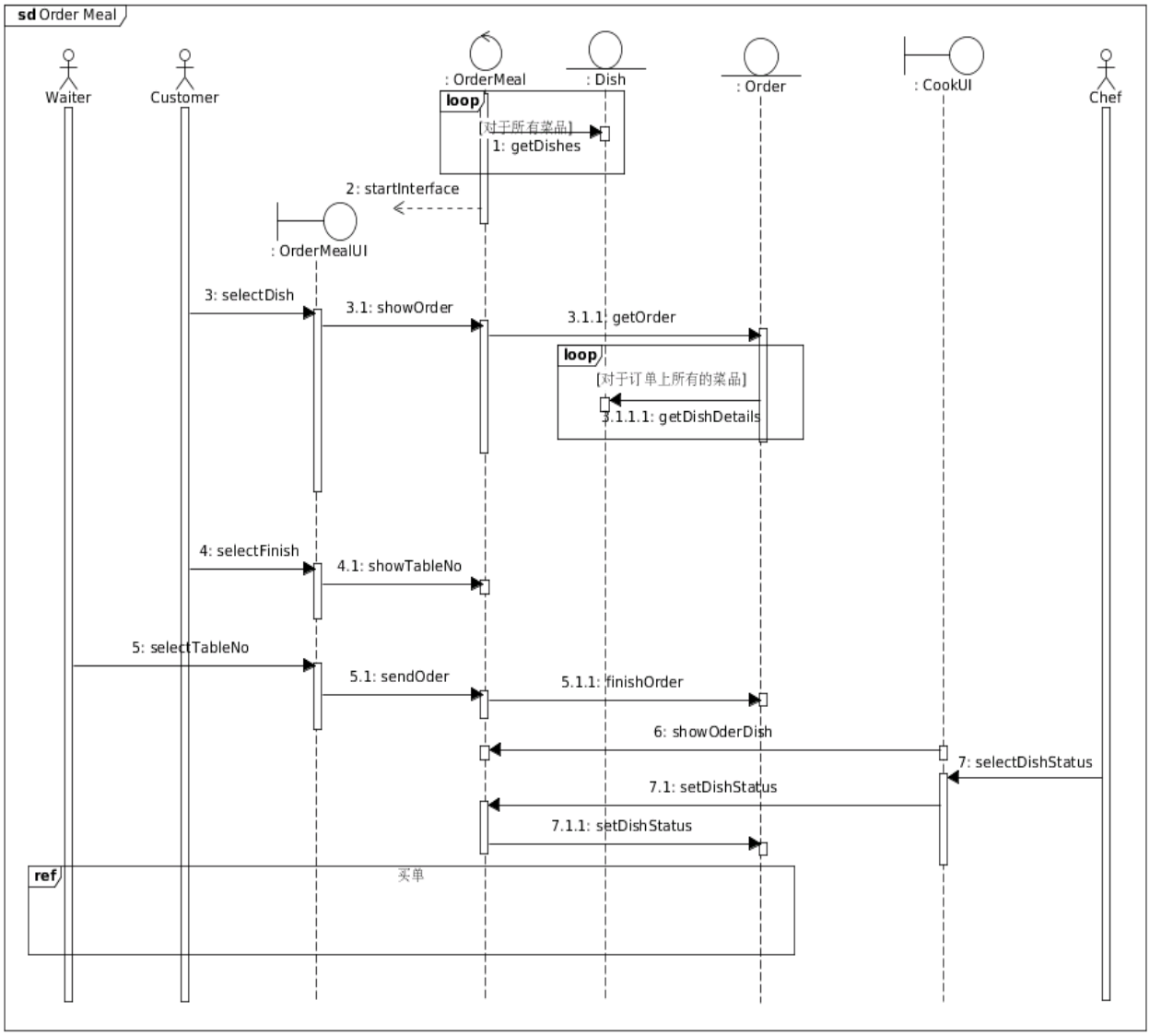

图 13: 带有交互使用的用况“Order Meal”的顺序图

“买单”的顺序图:

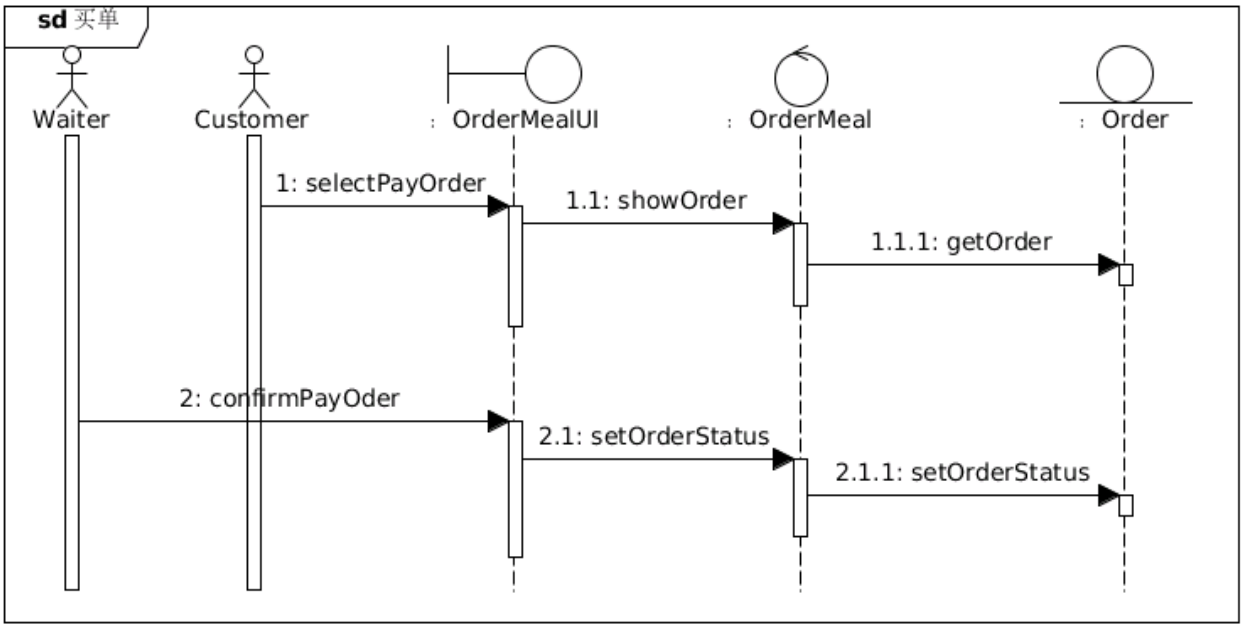

图 14: 交互片段“买单”的顺序图

2.1.4.3. 备选流顺序图

备选流评价用餐的顺序图:

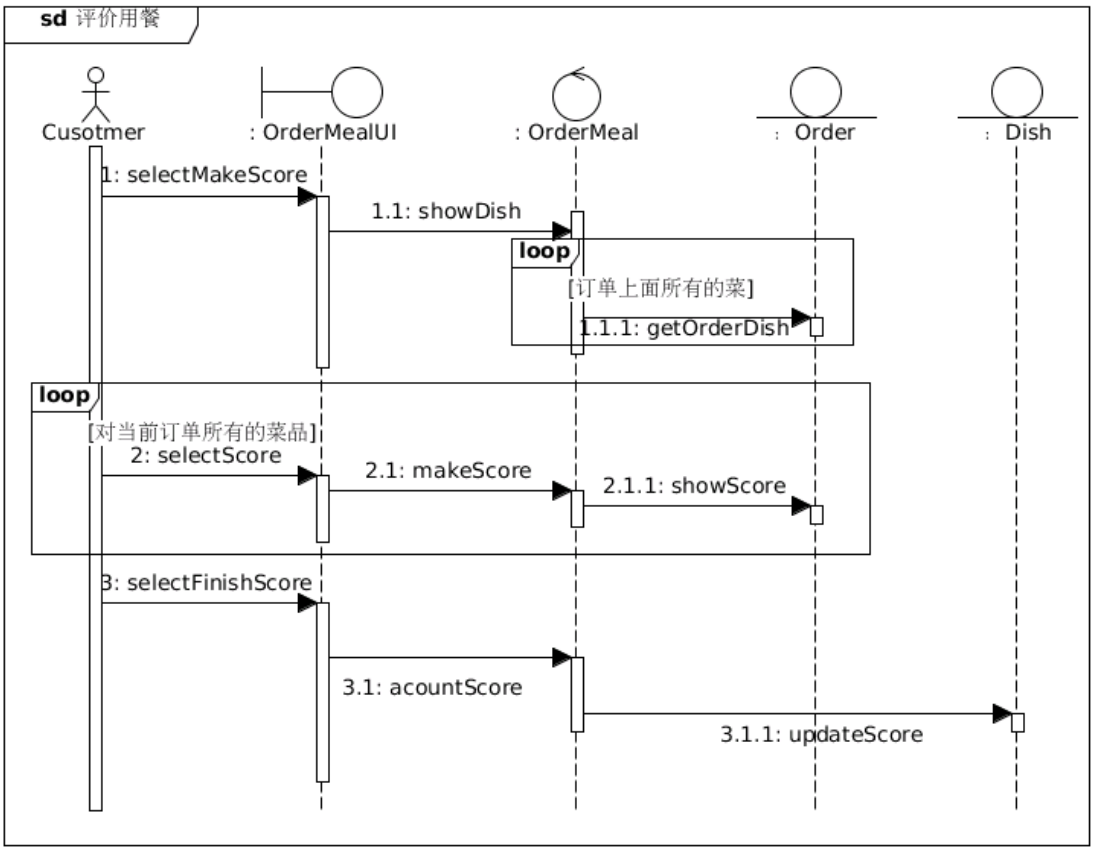

图 15: 备选流”评价用餐“的顺序图

选择菜品排序方式的顺序图:

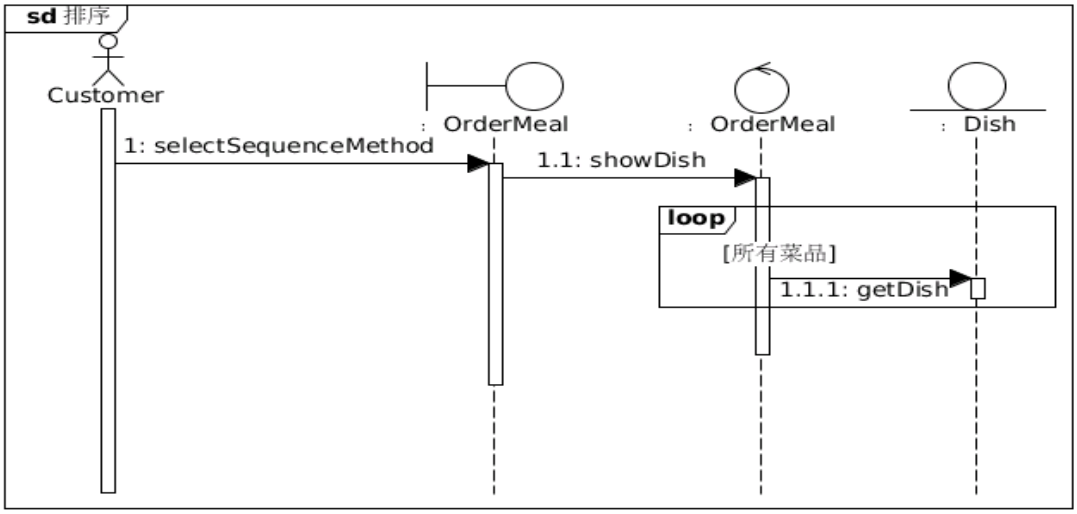

图 16: 备选流“菜品排序“的顺序图

查询已选菜品的顺序图:

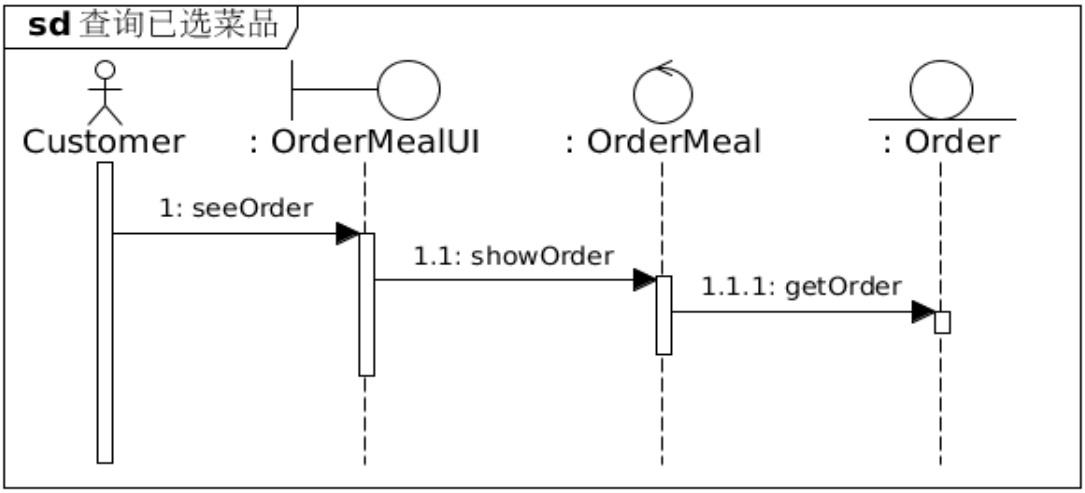

图 17: 备选流“查询已选菜品”的顺序图

取消菜品的顺序图:

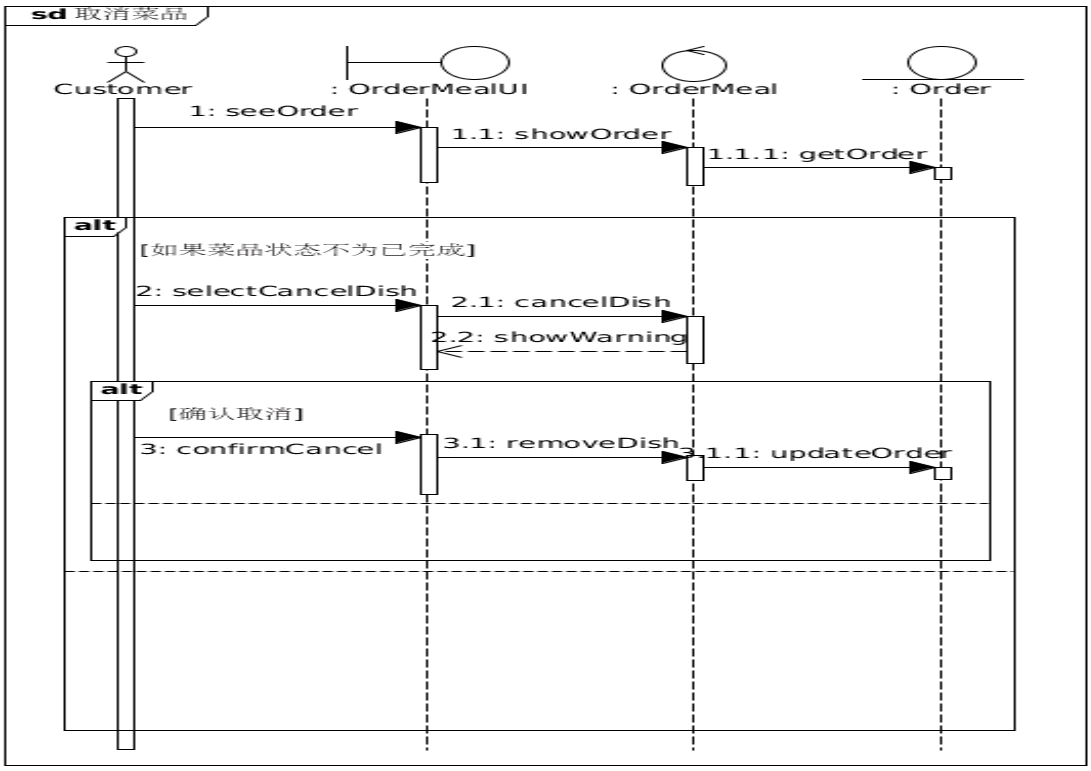

图 18: 备选流“取消菜品”的顺序图

添加菜品的顺序图:

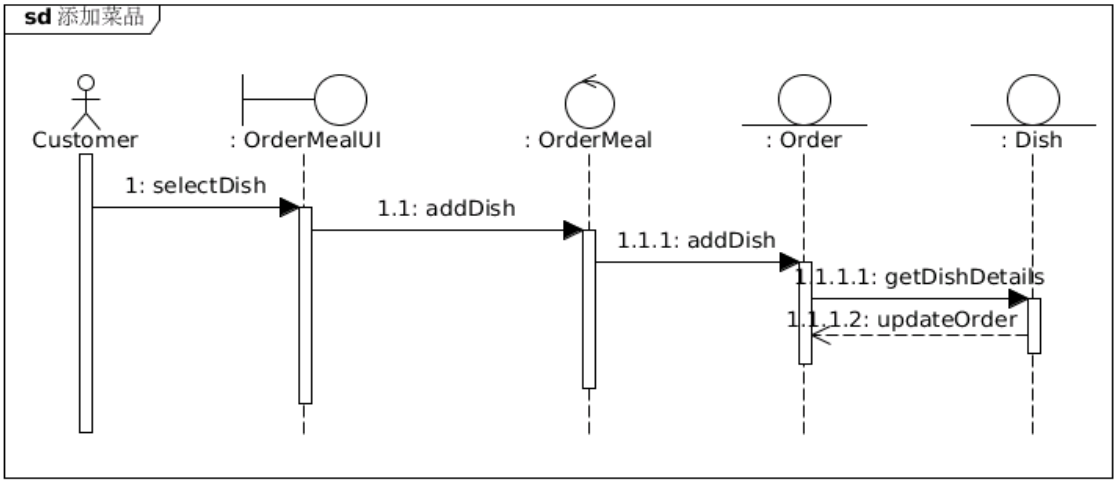

图 19: 备选流“添加菜品”的顺序图

2.1.5. 整合类图

图 20: 类图

2.1.6. 状态机

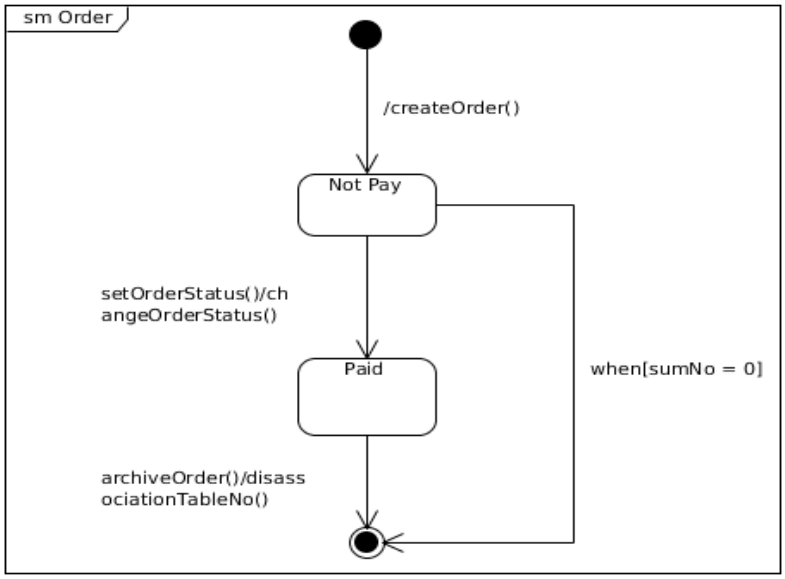

图 21: Order 类的状态机图

2.1.7. 操作规范

上下文： Dish

操作规范：getDish()

操作意图：返回菜品的菜名及该菜品的平均得分

操作签名：Dish::getDish() dishName::String，score::int

逻辑描述(前置条件和后置条件)：

前置条件： dish->exists

后置条件： result = dish.dishName, dish.score

其他调用操作：无

传递给其他对象的事件：无

特性集合：无

对异常的响应：未定义

非功能性需求：未定义

操作规范：getDishDetails()

操作意图：返回菜品的名字和价格

操作签名：Dish::getDishDetails() dishName::String, price::int

逻辑描述(前置条件和后置条件)：

前置条件： dish->exists

后置条件： result = dish.dishName, dish.price

其他调用操作：Order.updataOrder

传递给其他对象的事件：无

特性集合：无

对异常的响应：未定义

非功能性需求：未定义

操作规范：updataScore()

操作意图：更新本次订单中的每个菜品的平均得分

操作签名：Dish::updataGrade(score)

逻辑描述(前置条件和后置条件)：

前置条件： dish->exists

score >= LOWESTSCORE **and**

score <= HIGHTESTSCORE

后置条件： 更新了本次订单中所包含的每个菜品的平均得分

其他调用操作：无

传递给其他对象的事件：无

特性集合：无

对异常的响应：未定义

非功能性需求：未定义

操作规范：getDishes()

操作意图：返回菜品的名字、价格、图片以及风味

操作签名：Dish::getDishes() dishName::String, price::String, picture::String,

flavor::String

逻辑描述(前置条件和后置条件):

前置条件： dish->exists

后置条件： result = dish.dishName, dish.price, dish.picture, dish.flavor

其他调用操作：无

传递给其他对象的事件：无

特性集合：无

对异常的响应：未定义

非功能性需求：未定义

上下文： Order

操作规范：getOrder()

操作意图：返回订单信息

操作签名：Order::getOrder() order::Order

逻辑描述(前置条件和后置条件):

前置条件： order->exists

后置条件： result = order->all（有问题）

其他调用操作：Dish.getDishDetails

传递给其他对象的事件：无

特性集合：无

对异常的响应：未定义

非功能性需求：未定义

操作规范：setOrderStatus()

操作意图：设置订单的状态(未支付、已支付)

操作签名：Order::changeOrderStatus(orderNo, orderStatus)

逻辑描述(前置条件和后置条件):

前置条件： order->exists

后置条件： order.orderStatus = orderStatus

其他调用操作：无

传递给其他对象的事件：无

特性集合：无

对异常的响应：未定义

非功能性需求：未定义

操作规范：updataOrder()

操作意图：对订单进行更新

操作签名：Order::updataOrder(orderNo)

逻辑描述(前置条件和后置条件)：

前置条件： order->exists

后置条件： 订单中的菜品更新

其他调用操作：无

传递给其他对象的事件：无

特性集合：无

对异常的响应：未定义

非功能性需求：未定义

操作规范：showGrade()

操作意图：显示某个菜品的平均得分

操作签名：Order::showGrade() score::Dish

逻辑描述(前置条件和后置条件)：

前置条件： dish->exists

后置条件： dish.score = score

其他调用操作：无

传递给其他对象的事件：无

特性集合：无

对异常的响应：未定义

非功能性需求：未定义

操作规范：getOrderDish()

操作意图：获取订单的某个菜品的菜品名和平均评分

操作签名：Order::getOrderDish() dishName:String, score::int

逻辑描述(前置条件和后置条件)：

前置条件： dish->exists

后置条件： result = dish.dishName, dish.score

其他调用操作：无

传递给其他对象的事件：无

特性集合：无

对异常的响应：未定义

非功能性需求：未定义

操作规范：changeDishStatus()

操作意图：更改菜品的状态（烹饪已完成、烹饪未完成）

操作签名：Order::changeDishStatus(dishName, dishStatus)

逻辑描述(前置条件和后置条件)：

前置条件： dish->exists

后置条件： dish.dishStatus = dishStatus

其他调用操作：无

传递给其他对象的事件：无

特性集合：无

对异常的响应：未定义

非功能性需求：未定义

操作规范：finishOrder()

操作意图：生成一个订单

操作签名：Order::finishOrder()

逻辑描述(前置条件和后置条件):

前置条件：order->exists

后置条件：创建了一个新订单

其他调用操作：无

传递给其他对象的事件：无

特性集合：无

对异常的响应：无

非功能性需求：无

第3部分 设计

3.1. 物理设计和逻辑设计

3.1.1. 硬件平台和软件平台

硬件平台：台式计算机，嵌入式设备。

软件平台：windows 系统，安卓系统。

- 需要使用中间件，允许对象通过局域网与其他对象进行通信。
- 使用 C/C++/Qt 编写程序，则使用关系型数据库 MySQL。
- 系统将使用打印机，使用 C 语言编写本地接口方法。
- 提供特定功能的对象之间的交互，使用通信图进行设计。
- 数据输入界面布局按照区域设计，这些区域为将要创建或更新的对象提供数据--录入订单信息；录入菜单信息。
- 从特定的硬件或者其他系统发送或者接收命令和数据的性质确定，打印机输出，相关数据内容。

  设计开始于硬件和软件决定之后。

3.1.2. 系统设计和详细设计

系统设计在系统设计和详细设计两个层面，设计活动在系统嗯架构和企业整体架构的上下文中发生。

企业架构：使用独立的点餐系统。企业希望简化点餐操作，优化点餐流程，提高用户点餐的便捷和趣味性，提升用户体验；提高客人的点餐速度、点餐信息及时准确更新，和方便餐馆信息管理。服务员为客人提供最便捷的服务，厨师为客人烹饪菜品，并和经理决定菜品信息。装配较高性能的台式电脑，搭载 Window7 系统，路由器若干(假设），采购用户体验逐渐提升的安卓系统设备是很好的选择。

系统架构：处理在企业架构所提供的框架内的单个系统或者一组相关系统的架构。这些子系统之间的结构和关系是系统架构师的领域。使用独立的点餐系统。

详细设计：设计独立的元素来匹配架构，主要考虑对象和类的设计，也解决用户界面和数据库设计。

- 系统设计：设计影响系统整体，最重要的方面是系统架构。点餐要求系统各个组件协同工作，实时更新点餐信息，采用系统使用 MVC 风格架构。

- 详细设计：传统上,详细设计是关于设计输入,输出,进程和文件或者数据库结构;系统的这些相同的方面也必须要在面向对象系统中设计,但是他们将以类的方式组织。在项目的分析阶段,业务中的概念以类的形式被识别并且细化,用况将被确认和描述。这些被包含到类图中的类反映了业务需求但是他们仅仅包含了非常简化的类的视图,这些类用于处理用户或系统的接口,数据存储和以及总体协调其它类。这些类会被加入到设计中,细节的详细程度取决于新系统所使用的硬件和软件平台。--结合类图，关注输入输出，进程，文件数据库的设计，结合系统硬件。

3.2. 设计的质量和目标

标准：完成的应用程序是否高质量。

评估：使用一些标准确定设计是否符合目标。

3.2.1. 目标和约束

设计者需要达到开发早期设定的目标。

- 功能性：系统能够完成点餐过程，提供文档方便实际操作。
- 有效性：在长时间的情况下，系统依然能够正常运行。
- 经济性：系统成本应当在适合的范围内。
- 可靠性：不易出现硬件和软件故障，维护数据完整性。--保护运营信息。
- 安全性：抵御外部人员的恶意攻击和内部人员的为授权使用。
- 灵活性：能处理随时间而改变的业务需求。--能处理在不同时间段的订单处理。
- 通用性：系统的通用程度，包括可移植性。
- 可构建性：清晰，无不必要复杂内容的设计方便编写代码。物理设计与开发语言特性相关，面向对象语言不一定有相同的特征（特性和操作的可见性等），处理多重继承的能力，基本语言中列举的集和和链接等的使用类的可用性。--清晰的设计，从个层次方便代码的编写。
- 客观理性：允许项目经理评估实现各种子系统设计的工作量，为子系统提供相关的字包含性。
- 可维护性：高的设计降低维护成本。
- 可用性：涉及各方面，系统都是满足需求的。
- 可重用性：面向对象的系统很多特征便于提高可重用。继承，类，组件，设计模式重

用。

3.2.2. 设计折中

设计约束来自项目上下文背景，用户需求。客户和开发人员的协调需要折中协调。

3.2.3. 设计中的可衡量目标

项目需要达到可衡量的目标。

3.3. 系统架构

- 系统设计活动定义了进行详细设计的上下文环境(context)。
- “以架构为中心的”系统设计的一个主要部分是定义系统的架构。
- 为设计上的权衡设置优先级，确定子系统和主要部件，确定任何内在的并发性，将多个子系统分配到对应的处理器上，确定将要使用的设计模式，选择用于人机交互的，策略和标准，选择一个数据管理策略，指定代码开发标准，生成系统测试计划，确定实现需求。

3.3.1. 架构

- 系统架构师是代表客户工作的，部分角色是理解客户的业务,以及这种业务如何被一个信息系统最好地支持。
- 系统架构师处理全局整体的问题，信息系统的架构是系统的一个高层视图:它根据主组件及其连接方式进行建模;通常不会处理系统的详细设计,尽管它可能会设置详细设计所使用的标准。
- 架构提供对业务的灵活支持，专注将解决问题。
- 业务分析师归档每个需求，系统分析师考虑用况和其他需求转换为类模型支持属性职责和操作，设计转换分析模型的每一个方面到一个最终将是下这些需求的设计模型。

3.3.2. 架构风格

- 划分子系统：分层分区。具有共同属性的系统元素组合在一起，封装了职责的一致集合。子系统不是对象也不是函数，而是一组相关类关联操作事件和约束，并有一个与其他系统之间的定义清晰的小型接口 。--系统将为多个参与者提供不同的视图，共同完成点餐。并同时为多个客人提供服务。

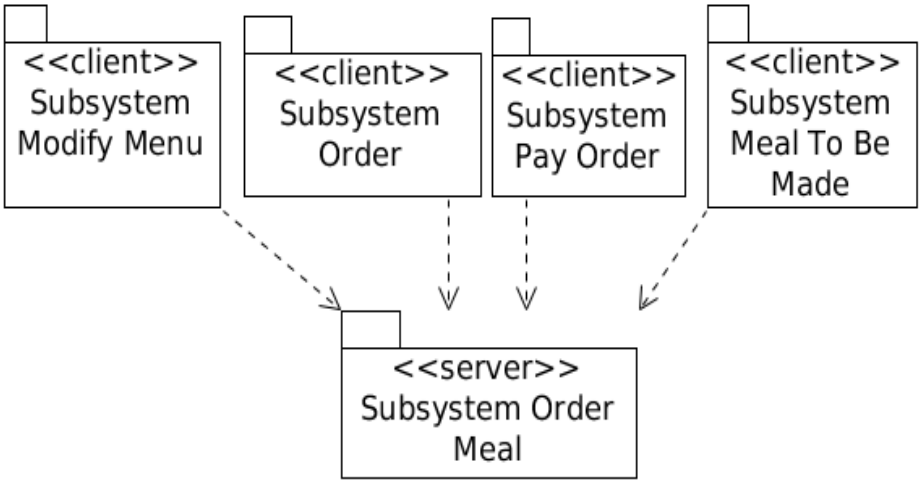

图 22: 初步子系统划分图

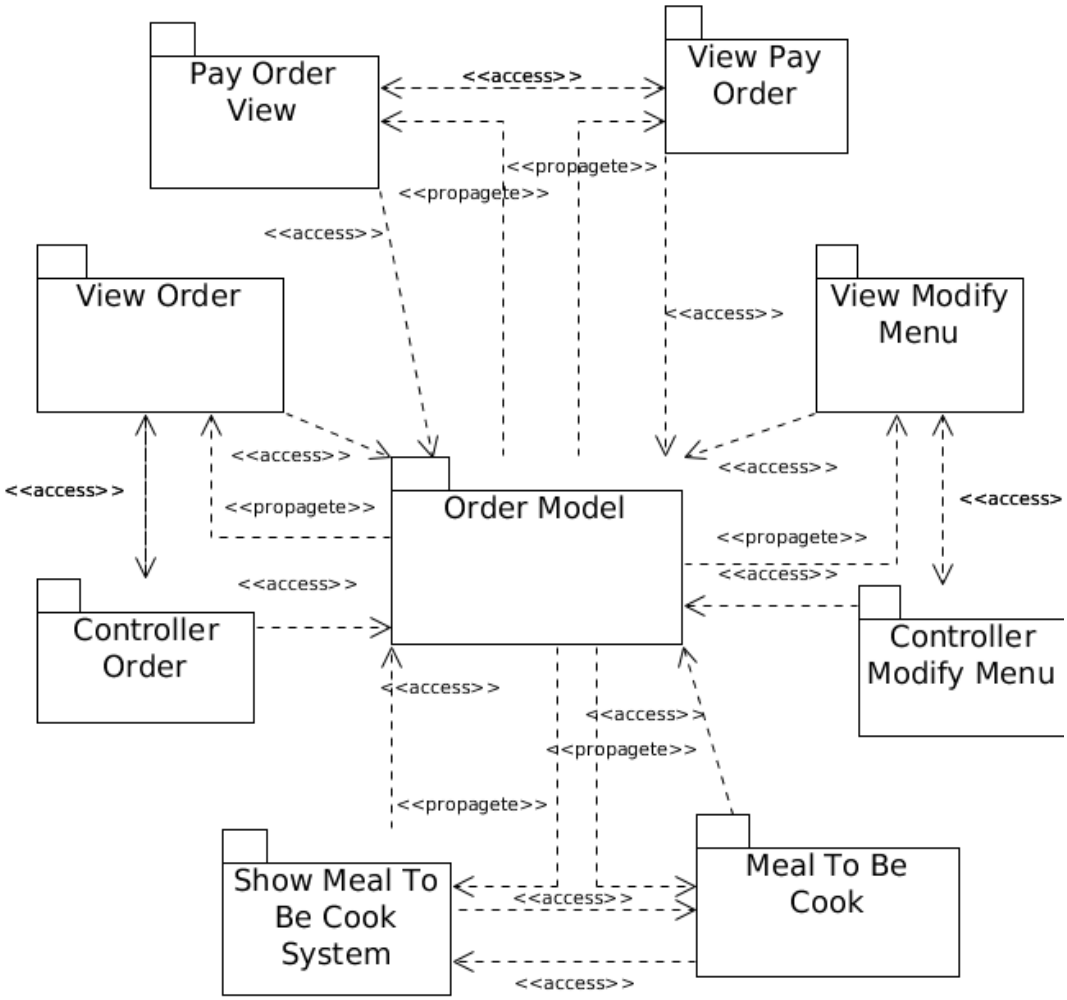

图 23: MVC 架构图

- MVC 风格：模型构成主要功能，视图表示用户界面，控制器管理模型和视图更新。支持通过不同界面风格表示的用户需求。用户界面灵活多变包含了相同的核心功能，任何核心的功能的改变，都需要改变所有的界面子系统。能确保所有视图都保持最新这一需求。

- 组件职责

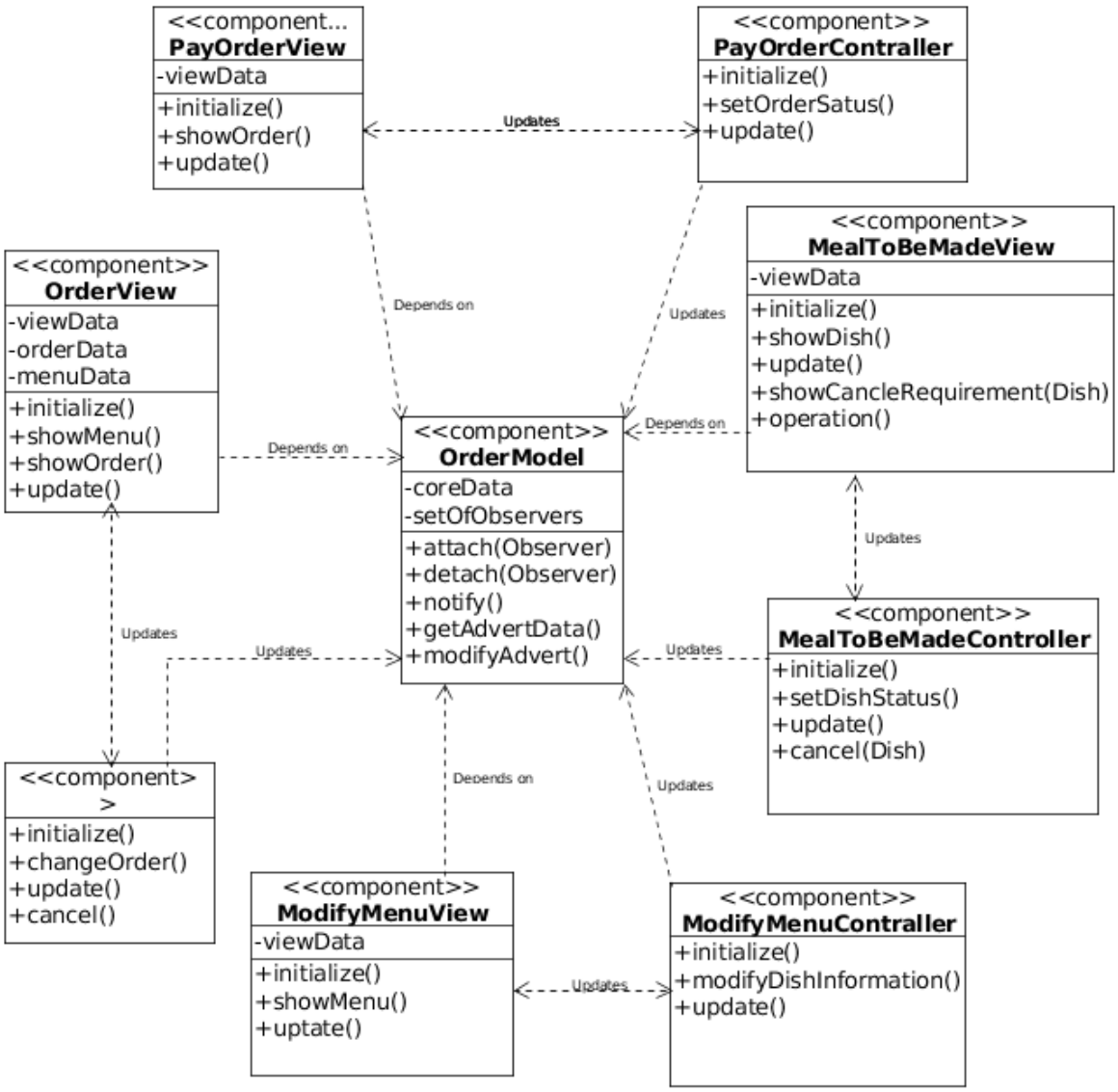

图 24: Order Meal 系统的组件职责

3.3.3. 并发性

用例同时响应 Customer/Waiter，Chef 等参与者的操作，每个事件触发不同控制，多个子系统划分到不同的不同的硬件单元。访问数据库由独立的数据库管理系统来处理。

3.3.4. 分配子系统

Order Meal 服务器子系统将分配到较高性能的台式计算机，其他子系统分配到手持设备

或 pc 上，可以满足硬件资源，成本和性能要求。

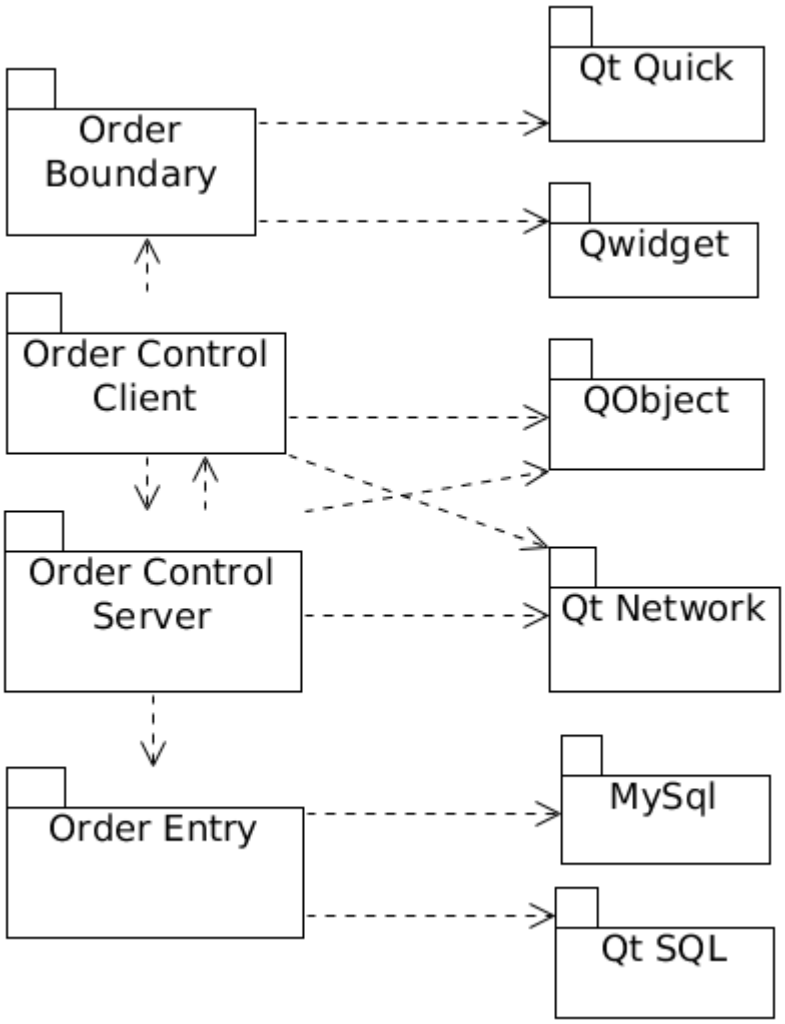

图 25: Oder Meal 软件架构包图