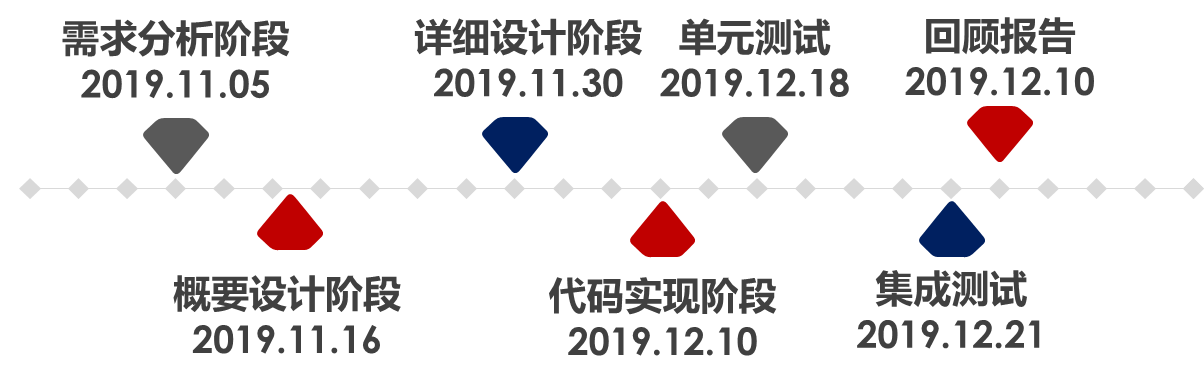
# 一、课程设进展回顾

## 1.1时间线回顾



## 1.2需求分析阶段

1.2.1数据描述

静态数据：概念类图、ER图

动态数据：活动图、状态机图、数据流图、时序图

数据字典

1.2.2功能需求

功能需求：用户目标分析、用例图

功能描述：用户用例详述、管理员用例详述

1.2.3性能需求

1.2.4运行需求

## 1.3概要设计阶段

1.3.1术语介绍

1.3.2总体设计

总体结构和模块外部设计

系统逻辑架构设计

系统物理架构设计

1.3.3功能分配与接口设计

模块划分

内部接口

外部接口

1.3.4数据结构与程序关联设计

1.3.5运行设计

1.3.6出错处理

1.3.7网络设计

## 1.4 controller API设计阶段

根据需求对每一模块进行API设计，同时分内部模块和外部模块进行分类，整理成文档，编码成Swagger界面

## 1.5详细设计阶段

为每个模块完成的功能进行具体的描述要把功能描述转变为精确的、结构化的过程描述。 根据模块的API功能画出每一个API函数的分层时序图，以便于代码的实现。

1.5.1具体模块需求分析

1.5.2模块的API说明

1.5.3模块的静态模型：分层类图及其包含的方法

1.5.4模块的动态模型：分层时序图、状态机图

1.5.5模块的设计说明

## 1.6后端实现阶段

根据详细设计阶段的API分层时序图，和完善的类图进行每一层的代码设计与实现。同时编写本地测试用例进行单元测试。设计与应用集成方法，将项目部署到自己租用的服务器上。

## 1.7开发成本——人时

需求分析阶段：3天\*4时\*5人（人时）

概要设计阶段：4天\*4时\*5人（人时）

Controller API设计阶段：2天\*6时\*5人（人时）

详细设计阶段：2天\*4时\*5人（人时）

后端实现阶段：5天\*19时\*5人（人时）

总计：715人时

# 二、系统设计和实现中存在的问题及解决方案

**问题和解决方案如下：**

**（1）沟通：**

成员只有始终保持沟通讨论，保证接收到最新的、一致的项目需求信息，才能使得开发工作顺利进行，避免出现信息交换不及时而导致的返工。对于沟通，结合实际来说，如果需求分析师不能将变更的需求信息及时传递给UI设计人员，就会导致不符合用户需求的设计，更会使开发人员写出无用的代码，这必然导致重设计、重编码，甚至会延误整体项目进度。对于讨论，尤其是像我这样缺少经验的项目经理，不论是制定计划，还是工作量识别，都必须向有经验的同事请教，接受正确的建议，才能得到合理的安排。

**（2）项目范围：**

功能边界要清晰 项目经理以需求文档为依据，将项目范围及边界清晰罗列，是把控项目开发进度的先决条件。 对于oomall系统来说，其功能并不复杂，且开发周期短，所以在确定项目范围并进行任务细化时，可精确到接口、页面。把一个大任务分解成一个个的小任务的好处是，可以帮助我们更加精确的估计出它们的工作量，并暴露出很多可能一时无法想到的工作量，也可以保证后续进行项目开发过程的状态跟踪，更加精确。

**（3）时间计划：**

人员分配要合理，以前总认为写计划比写代码容易的多，其实恰恰相反。一份合理的项目计划需要经过思考、沟通、权衡、询问、倾听的过程，要知道，用来分析解决问题需要花费的时间，远远大于单纯的写代码时间。 项目进度计划必须将分解出来的小任务，综合考虑时间、难易程度、人员能力，估出工作量并进行合理分配。

**（4）代码开发：**

功能验证要同步，当日的开发任务结束后，作为项目经理应该对现有开发成果做验证，即对已完成的功能做验证，及时发现缺陷或其他问题，次日找对应的开发人员做修复。

# 三、从软件工程角度分析课程设计过程和存在的问题

**1、开发工具**

使用SpringBoot Mybatis FreeMarker Maven Git等工具

**2.过程框架**

本次开发中，采用增量模型的过程框架，针对课程设计中不断变更的需求，一步步交付可执行模块，多次迭代后获得最终版本。

1. **开发方法**

本次课程作业采用敏捷开发方法是一种从1990年代开始逐渐引起广泛关注的一些新型软件开发方法，是一种应对快速变化的需求的一种软件开发能力。相对于“非敏捷”，更强调程序员团队与业务专家之间的紧密协作、面对面的沟通（认为比书面的文档更有效）、频繁交付新的软件版本。能够很好地适应需求变化的代码编写和团队组织方法，也更注重软件开发中人的作用。敏捷建模（Agile Modeling，AM）的价值观包括了XP的四个价值观：沟通、简单、反馈、勇气，此外，还扩展了第五个价值观：谦逊。

敏捷开发特点：

(1)人和交互 重于过程和工具。

(2)可以工作的软件 重于求全而完备的文档。

(3)客户协作重于合同谈判。

(4)随时应对变化重于循规蹈矩。

**4.软件质量**

  软件属性包括功能属性和质量属性，但是，软件架构（及软件架构设计师）重点关注的是质量属性。因为，在大量的可能结构中，可以使用不同的结构来实现同样的功能性，即功能性在很大程度上是独立于结构的，架构设计师面临着决策（对结构的选择），而功能性所关心的是它如何与其他质量属性进行交互，以及它如何限制其他质量属性。课程设计中我们对微服务的切分，模块职责的界定进行了详细讨论，根据高内聚低耦合原则，在可行范围下以软件质量优先进行开发。

**5.开发效率**

理想情况中是多--单位时间产量高，快--同样的产量所需的时间少，好--质量高，省--省钱，省时，省资源，团队的通力合作能使得软件的开发效率更高。