软件工程的所有的环节和步骤，都是有其对应的意义的，至少我讲的这个流程里是这样的。如果是那些比较老旧，虚头巴脑的一些东西，不好落地的，过于复杂一些东西，现在暂时这个阶段用不着的一些东西，我都没讲。

概要设计 -> 架构师必备能力，设计出来一套系统长什么样子的，跑起来大概是什么样子的

概要设计评审 -> 让前天team的架构师来把一下关，尽量避免你犯了什么大错

详细设计 -> 是你手下的小弟必备的能力，小弟必须得去对自己负责的那个模块或者子系统，完成对应的一个详细设计。

P7技术专家，架构师的角色，概要设计；P6高工，带1~3人的小团队，负责一个独立子系统的详细设计，也可能是某个子系统的概要设计；P5初中级的工程师，一般就是干活儿的主力，他们负责对某个小型子系统或者是子系统中的一些模块，完成对应的详细设计

你必须强力push你手下的小弟，把系统设计的非常详细，参照我们的详细设计规范来。详细设计文档，需要详细到什么程度呢？照着文档就可以写代码。

数据库、缓存、MQ -> 设计好；接口 -> 设计好；实现类图（面向对象设计的功底，设计模式运用的功底）-> 设计好；完成上面3个步骤之后，将每个功能的运行流程都设计出来，写一些文字，复杂的画一些活动图

当然，如果你的小弟水平很low，可能刚开始还是得你自己来设计详细设计文档，让小弟参照着文档来做。但是你必须得培养小弟可以独立完成一些模块的详细设计。

详细设计评审 -> 就是你作为架构师，你手下的小弟完成了所有的详细设计之后，你就需要开会，评审会议，你仔细看你每个小弟的设计文档，你需要结合着那个模块对应的业务需求，来仔细反复思考，他们的详细设计有没有什么地方会有问题的

1、数据库设计

你要重点看一看数据库里面的索引设计

我们在这里，因为是讲课，没必要将一些CRUD的一些东西讲的太细，可能耗时太长了，大量的时间可能带着大家在设计CRUD的一些东西

所以说我这里之前给大家讲过了，就是我们对数据库设计这块，就给出了ER模型，数据库的物理设计，没搞，索引，主键，外键，还没弄。因为现阶段比较简单，所以我们可以到一边开发的时候，一边根据我们写了什么SQL，来现场设计和增加索引。

但是在实际的项目中，必须要求你的小弟在这一块，在完成了数据库ER模型、接口、实现类、运行流程的设计之后，必须要结合自己要执行的SQL操作，完成数据库的物理索引的设计。

数据库的设计这块，一方面你是看一看表名、字段名的命名，备注，你是否看得懂；数据库ER实体之间的关系是否正确；数据库整个设计，凭自己的感觉去看看，有没有什么问题；看看每个表都建立了哪些索引，让召开评审的同学说一说每个索引的用处。

2、接口设计

主要不是架构师看的，同一个团队内部有互相调用关系的几个同学，互相之间要去看看对方的接口设计，能否满足自己的需求。包括，针对前端的接口，需要前端同学过来参加这个评审，让前端看看这个接口能否满足他的需求

我们之前为了讲课方便，所以对面向前端的接口，都没有设计，主要是面向前端的接口，大多数都是CRUD的一些操作，设计接口，需要耗费大量的精力来编写输入输出对的json格式。很麻烦，很浪费时间。

但是，在实际的项目中，肯定是要求你手下的小弟，必须将所有的接口都给设计好的。然后让要调用他的接口的同学来评审，看接口是否ok

3、开发架构

实现类图，你作为一个架构师，你需要去review你手下的小弟的类设计。

（1）面向对象的设计

也就是说，我们现在还都是一些比较简单的CRUD的一些操作，没有太多所谓的面向对象的设计，大多数可能给都是一些controller、service、dao、mapper的设计。

但是到了我们课程的项目后期，大量的复杂技术架构出来，面向对象的设计，就很重要。

如果这块没有做良好的面向对象的设计，最后写出来的代码基本上就是一坨屎。面向过程去设计，最后可能一个复杂的模块，就两三个类，每个类里面就几个方法，大量的业务逻辑耦合在了一个方法里面。最后一眼看过去，就是一个方法里面一坨屎。

面向对象是很重要的，如果你设计的好的话，那么你的代码的结构是很清晰的，代码是面向对象的，比如老鹰类，翅膀类，鸟巢类，天空类 -> 老鹰类组合了翅膀类，基于翅膀类去飞，平时基于鸟巢类去休息，可以飞到天空类里面去 -> 老鹰类里面有十几个方法，飞，休息

（2）设计模式的运用

跟面向对象的设计有关系的，很多时候你为了面向对象去设计，让代码有清晰的结构，就可能会用这个设计模式。

如果我们不用任何的设计模式，看起来就是一个类，这个类里有一堆的方法，每个方法有一堆恶心的逻辑。

命令模式+模板方法模式+工厂方法模式，在对应的操作过来的时候，我们就是去找对应的工厂创建一个对应的库存更新命令出来，执行这个库存更新的命令，这个库存更新命令会按照模板方法定义的模板，来依次执行多个步骤逻辑

我觉得很多同学所在的团队，或者是公司，哪怕有这一些流程，执行的也不一定很好。设计，不是写文档，设计就是设计，体现的是你的系统设计的功力，文档只不过记录设计的载体。不要说我在写文档，公司里有规范，我必须得写，硬生生凑一堆没有生气的死板的文档出来，这个文档都不好好写，就是满足公司的规范。

你绝对是不可能成为一个架构师的，我这次讲解的这些流程，都是架构师必须要掌握的东西。而且大家要理解你做的每件事情背后的意义。

实现类图的设计，我们之前做了一点简化，就是对于那些CRUD相关的，或者是业务逻辑相关的类，我们都没有去设计对应的类图。这个在实际的应用中，是可以接受的，哪怕是在公司的实际工作环境中，这个是ok的。controller -> service -> dao -> mapper。

但是我们对那些重要的实现类图，体现出有一定面向对象设计含量的，和设计模式运用含量的相关的东西，我们都给出了实现了类图

我们通常而言，建议说，对于CRUD，也可以设计类图，只不过是设计简单类图。但凡有一定复杂程度的业务流程和业务场景，建议给出详细的面向对象的设计类图；只要能用设计模式的，就给出设计模式的实现类图。

4、运行流程

实现类图和运行流程，是我在详细设计里面，最最最最重视的两块东西

很多同学之前做详细设计，可能主要就是做数据库设计，接口设计 -> 恰恰相反，我不觉得这两块会有很大的问题，数据库设计和接口设计，重要 -> 这两块一般来说不是系统设计真正出问题的一个重灾区

哪怕你手下的小弟写的有问题，数据库表设计的一塌糊涂，接口设计的一塌糊涂；一般来说都能给指正出来，让他去修改的

很多公司，很多团队，尤其是一些中小型公司，创业型公司，流程和规范不完善的公司，他们做详细设计文档，只写数据库设计和接口设计 -> 类设计，运行流程的设计，基本是泛泛而过，或者根本就不做

实现类图不设计，代码是一坨屎 -> 运行流程不设计，你的小弟的详细设计文档就是对你来说是个黑盒，你根本不知道他要写哪些类，这些类之间的协作和运行流程是什么 -> 可能会导致最后你的小弟写出来的代码跟你想象的不太一样

运行流程不设计好的话，那么你作为一个架构师，根本是无法把控你的系统

哪怕你的概要设计做的再好，根本没法落地，到你的小弟落地的时候，写一坨屎出来，最后功能可以跑通，但是他那个可悲的代码，完全没有办法呗别人理解。甚至可能他写出来的业务流程是错的。直到你集成测试，code review的时候，才会去发现。

对那些重要的，复杂的流程，我要求必须是要活动图

对于一些不复杂的流程，可以手写文字版本的运行流程就可以了

但是对于一些核心的，复杂的，重要的业务流程，我们需要去画活动图，通过活动图体现出来一个正确的，清晰的系统运行的流程

5、详细设计的重要性总结

重中之重，一句话，你务必确保说，你作为一个架构师，看了你手下的小弟设计出来的详细设计文档，连你都知道怎么写代码，可能那个代码不用你去写，但是你看着这个文档，绝对知道怎么去写这个代码

做到这个程度，详细设计文档就成功了