《软件架构设计》学习报告：

**什么是软件架构（Software Architecture**）？

软件架构是指在一定的设计原则基础上，从不同角度对组成系统的各部分进行搭配和安排，形成系统的多个结构而组成架构，它包括该系统的各个组件，组件的外部可见属性及组件之间的相互关系。组件的外部可见属性是指其他组件对该组件所做的假设。

软件架构设计就是从宏观上说明一套软件系统的组成与特性。

软件架构设计是一系列有层次的决策 ，比如：功能与展现的决策；技术架构的决策；自主研发还是合作；商业软件还是开源软件 。

说白了就和盖房子一样，卧室设计成什么样，客厅设计成什么样，厕所设计成什么样，上述中的“卧室”，“客厅”，“厕所”就相当于软件中的各个模块，软件架构确定局部模块采用什么技术，确定整体采用哪种技术将他们统一起来。

**为什么要进行软件架构设计？**

计算机科学和程序设计的飞速发展，使得软件设计应用到从航空航天到日常生活的方方面面。单个人开发一段小程序的做法早就过时，大范围协作的工程化时代随即到来。

随着大范围协作的效率问题和软件复杂度的爆炸式增长，管理和技术方面的各种不确定性也爆发性增加，导致软件开发的质量无法得到有效保证，周期和成本无法得到有效控制。

人们一直在寻求找到这些问题的解决办法。然而 Fred Brooks 在 1975 年出版的软件工程圣经《人月神话》中说，没有（能解决所有问题的）银弹（There is no silver bullet）。

自此，人们发展了项目研发过程管理来控制管理活动的不确定性，同时也发展了软件架构设计方法来控制技术方面的不确定性。

进而在实践中不断的总结和改进，用于有效指导和最大程度的保障软件开发的质、周期和成本。

通俗来说，现在不是单个代码英雄的时代，现在的软件不可能一个人独立完成，那就得协作，协作就会出现一系列问题，如何进行管理，如何使开发的质量得到保证，如何不至于软件开发延期，这就需要引入软件架构设计来解决。

**六大基本原则**

架构设计最重要的就是**编程思想，**在日常开发中我们需要遵从六大基本原则，分别是：

单一职责原则(SRP Single Responsibility Principle)

定义：对一个类来说，应该仅有一个引起它变化的原因。简单来说就是模块的划分需要做到职责清晰，不要既能满足A需求，又能满足B需求，这样耦合太强，不利于后期的扩展和维护。

开闭原则(COP Open Close Principle)

定义：软件中的对象（类、模块、函数等）应该对于扩展是开放的，但是对于修改是封闭的。简单的说就是程序中类应该是易于扩展的，而不是一有什么新的需求或者变化就去修改原来的代码，这样很容易出现问题。

里氏替换原则(LSP Liskov Substitution Principle)

定义：在基类出现的地方，其子类也可以出现，并且不会出现错误。简单的说就是利用了Java面向对象思想中的继承、多态的特性

依赖导致原则(DIP Dependence Inversion Principle)

定义：高层次的模块不依赖于低层次的模块的实现细节。简单来说就是依赖抽象（高层次模块）而不依赖具体实现（低层次模块）。

接口隔离原则(ISP InterfaceSegregation Principles)

定义：不依赖不需要的接口，从而容易重构、更改和重新部署。

迪米特原则(最少知识原则 LKP/LOD Least Knowledge Principle)

定义：一个类应该对自己需要耦合或者调用的类知道的最少，类的内部如何实现与调用者或者依赖者没关系，调用者或者依赖者只需要知道它需要的方法即可。

在平时编程中，我用到的单一职责原则和开闭原则比较多，其他原则接触的相对来说比较少，并没有特别注意。

**架构设计的六大步骤**

通过阅读相关的开发案例，架构设计大致可以分为6个步骤，依次是需求分析、领域建模、确定关键需求、概念架构设计、细化架构设计、架构验证。

（1）需求分析

我们知道，需求分析的目标是找出功能、质量和约束这三个方面的要求。首要工作是沟通以获取需求，然后是确定非功能性需求、确定系统目标、建立用例模型等，最终以需求说明书的形式作为产出物。

（2）领域建模

领域建模的目标是构建业务领域模型，业务决定功能，功能决定模型。领域建模主要工作是与业务领域专家一起，整理和掌握软件功能和非功能要求的业务数据、业务流程等。

（3）确定关键需求

具体而言，确定关键需求工作包括：为了确定关键功能而进行的功能需求和约束需求的研究；为了确定关键质量而进行的质量需求和约束需求研究。

（4）概念架构设计

具体包含五项工作：决定如何划分顶级子系统、架构风格选型、开发技术选型、二次开发技术选型、集成技术选项。

（5）细化架构设计

从逻辑架构、开发架构、运行架构、物理架构、数据架构五个方面出发，对模块划分、接口定义、领域模型、技术选型、控制流程、硬件分布、软件部署、存储格式等内容进行详细设计。

（6）架构验证

对后续工作产生重大影响且造成返工代价很高的任何工作，都应该安排原型测试和评审工作。同时，进行必要的软件技术选型验证工作。

当然这也不是绝对的，基本上是这个流程，即可完成对某一项目的架构设计。