重庆交通大学信息科学与工程学院

大作业报告

论文题目	软件架构设计综述
课程名称	软件系统架构
专业班级	计算机 1601 班
学　　号	631607040116
姓　　名	何　德　华

任课教师　　王　家　伟

评语：			
成绩：			
评阅人：		**评阅时间**	

20　　年　　月

重庆交通大学信息科学与工程学院大作业任务书

<table>
<tr><td>课　程</td><td>软件系统架构</td><td>班级</td><td>2016 级 1-4 班</td><td>任课教师</td><td>王家伟</td></tr>
<tr><td>题　目</td><td colspan="2">软件架构设计综述</td><td>完成时间</td><td colspan="2">第 18 周</td></tr>
<tr><td>主要内容</td><td colspan="5">以“软件架构设计综述”为题写一篇论文（每位同学独立完成），字数不少于 5000 字。
1 论文主要内容必须包含：
（1） 软件架构的概念；
（2） 软件架构的目标；
（3） 软件架构的种类；
（4） 软件架构的设计方法及阶段；
（5） 架构设计每个阶段的内容；
（6） 软件架构设计的相关内容如风格、模式等。
2 论文的组成部分：
(1)摘要
(2)关键词
(3)目录
(4)正文
(5)参考文献</td></tr>
<tr><td>报告要求</td><td colspan="5">1、封面；
2、大作业任务书；
3、论文摘要
4、关键词
5、目录
6、论文正文
7、参考文献
注意：报告必须打印，每位同学打印报告后交给本班学习委员，学习委员集中交给任课教师，截止时间第 19 周前。</td></tr>
<tr><td>版面</td><td colspan="5">见附件中“论文模板”</td></tr>
</table>

摘要

关键字：软件系统架构；组件；

软件系统架构描述的对象是直接形成系统的抽象组件。 各个组件之间的连接是组件之间通信的清晰且相对详细的描述。 在实现阶段，这些抽象组件被细化为实际组件，例如特定类或对象。 在面向对象的世界中，组件之间的连接通常由接口实现。软件架构是构建计算机软件实践的基础。 与建筑师设计建筑项目的设计原则和目标一样，作为绘图员绘图的基础，软件架构师或系统架构师将软件架构作为满足不同客户需求的实际系统设计的基础。

目　　录

1. 软件架构的概念

1.1 介绍

软件架构将系统描述为计算组件及组件之间的交互。计算组件是泛指，可进一步划分为处理组件、数据组件、连接组件等，可以指子系统、框架、模块以及类等不同粒度的软件单元。决策派定义示例：软件架构包括以下一系列问题的重要决策：软件系统的组织、选择组成软件系统的结构元素和它们之间的接口以及如何组合这些元素，使它们合成为更大的子系统。软件系统的其他特性，例如使用、功能性、性能、弹性、重用、可理解性、经济和技术的限制及权衡以及美学等。一个更通俗易懂的决策列举：模块如何划分；各模块的职责为何；每个模块的接口如何定义；模块之间采用何种交互机制；开发技术如何选型；如何满足约束和质量属性需求；如何适应可能发生的变化。

1.2 观点

软件系统架构描述的对象是直接形成系统的抽象组件。 各个组件之间的连接是组件之间通信的清晰且相对详细的描述。在实现阶段，这些抽象组件被细化为实际组件，例如特定类或对象。 在面向对象的世界中，组件之间的连接通常由接口实现。软件架构是构建计算机软件实践的基础。 与建筑师设计建筑项目的设计原则和目标一样，作为绘图员绘图的基础，软件架构师或系统架构师将软件架构作为满足不同客户需求的实际系统设计的基础。

1.3 中心

结构化程序设计由 E.W.dijkstra 在 1969 年提出，是以模块化设计为中心，将待开发的软件系统划分为若干个互相独立的模块，这样使完成每一个模块的工作变单纯而明确，为设计一些较大的软件打下了良好的基础。由于模块互相独立，因此在设计其中一个模块时，不会受到其它模块的牵连，因而可将原来较为复杂的问题化简为一系列简单模块的设计。模块的独立性还为扩充已有的系统、建立新系统带来了不少的方便，因为我们可以充分利用现有的模块作积木式的扩展。按照结构化程序设计的观点，任何算法功能都可以通过由程序模块组成的三种基本程序结构的组合： 顺序结构、选择结构和

循环结构来实现。结构化程序设计的基本思想是采用"自顶向下，逐步求精"的程序设计方法和"单入口单出口"的控制结构。自顶向下、逐步求精的程序设计方法从问题本身开始，经过逐步细化，将解决问题的步骤分解为由基本程序结构模块组成的结构化程序框图；"单入口单出口"的思想认为一个复杂的程序，如果它仅是由顺序、选择和循环三种基本程序结构通过组合、嵌套构成，那么这个新构造的程序一定是一个单入口单出口的程序。据此就很容易编写出结构良好、易于调试的程序来。

2. 软件架构的目标

正如同软件本身有其要达到的目标，软件架构设计要达到如下的目标：1.可靠性：软件系统对于用户的商业经营和管理来说极为重要，因此软件系统必须非常可靠。2.安全性：软件系统所承担的交易的商业价值极高，系统的安全性非常重要。3.可扩展性：软件必须能够在用户的使用率、用户的数目增加很快的情况下，保持合理的性能。只有这样，才能适应用户的市场扩展得可能性。4.可定制化：同样的一套软件，可以根据客户群的不同和市场需求的变化进行调整。5.可伸缩：在新技术出现的时候，一个软件系统应当允许导入新技术，从而对现有系统进行功能和性能的扩展。6.可维护性：软件系统的维护包括两方面，一是排除现有的错误，二是将新的软件需求反映到现有系统中去。一个易于维护的系统可以有效地降低技术支持的花费。7.客户体验。软件系统必须易于使用。8.市场时机：软件用户要面临同业竞争，软件提供商也要面临同业竞争。以最快的速度争夺市场先机非常重要。

3. 软件架构的种类

- **分层架构**

  分层架构是使用最多的架构模式，通过分层使各个层的职责更加明确，通过定义的接口使各层之间通讯,上层使用下层提供的服务。分层分为：严格意义上的分层，一般意义的分层。严格意义的分层是 n+1 层使用 n 层的服务。而一般意义

的分层是上层能够使用它下边所有层的服务。领域驱动设计的分层定义：UI 层，UI 控制层，服务层，领域层，基础设施层。

- **MVC 架构**

MVC 架构相信做软件的都听说，主要是为了让软件的各部分松耦合，现在好多根据 MVC 思想构建的框架如：Spring MVC,Structs2,ASP.Net MVC 等。MVC 是 Model View Control 的简写，他的原理是什么那，比如拿 web 来举例吧。当一个 web 请求来了以后 View 接收这个请求，随即把请求转发给 Control 进行处理，Control 通过分析请求的类型等信息决定加载哪些 Model，当 Model 加载完成以后 Control 通知 Model 已经加载完毕，这是 View 就去读取 Model 数据进行显示自己。MVC 还有一个衍生架构叫 MVP,因为 MVC 的 View 跟 Control 和 Model 都有耦合关系所以为了解除 View 和 Model 之间的关系,View 不直接读取 Model 而是通过 Control 来转发 View 需要的数据。还有一个衍生架构叫 MVVP,就是增加了一个 ViewControl 的层，用来辅助视图的生成，这样 View 的功能更加简单只是用来显示不包含其它的功能，而且有了 ViewControl 使多视图或替换视图很方便。MVP 微软的 WPF 就是使用这种架构。

- **微内核架构**

微内核架构就是做一个稳定通用的内核，也就是给软件设计一个强劲的心脏。如果需要更多功能通过在内核外部再封装一层对软件进行扩充，微内核提供基本的接口供外部调用，这些接口一定要通用，并且提供事件的机制告诉外部内部发生的事件,这样就是内核与外部完全隔离。微软操作系统就是按照微内核设计的。我之前做了一个 Gis 组件当初思想也是这个样子的，但是当初不知道还有微内核架构，有了对微内核的深入理解会进一步完善那个 Gis 组件。

- **元模型架构**

元模型架构就是有元数据支撑的架构，现在使用的也很广泛，比如：ORM,.Net

类的设计等都是元数据支持的。元数据有自我描述性比如 ORM 会描述类对应数据库中的表属性对应数据库里的字段，还有 IOC 类中的引用需要注入哪个类等等都会通过元数据的形式实现。IOC 框架通过解析元数据信息使注入和被注入类只通过接口依赖，这样替换注入类很方便。元数据架构是很灵活的架构，可发展空间非常大，元数据架构会经常用反射技术或者动态代码生成技术。我之前做了一个 ORM 就是用到的元数据架构，我还想给 ORM 添加依赖注入面向切面编程等特性都很方便的。

- **管道-过滤器架构**

这个模式就像是工厂的流水线，生产原料通过流水线经过很多环节进行处理变成产品。软件也是一样的，网络 OSI7 层就是消息通过管道内部的很多步处理对消息进行加工过滤转换。再举一个例子，两家企业需要信息交换，但是企业的信息格式和描述规则都不相同，如果想达到交换必须经过处理，所以我们就得用管道过滤器模式，通过管道过滤器模式信息进入管道我们会在管道里添加各种处理功能，比如：数据验证，信息加密，信息解密，信息压缩，信息解压缩，格式转换等功能，对消息进行处理以符合我们要求的消息格式，而且如果需要添加一个新的处理只要把处理的功能插入到管道中即可，这样达到最大的灵活性。应用此模式的有：ASP.net 请求模型，Spring 对象构造，Structs 数据请求等。

4. 软件架构的设计方法及阶段

4.1 软件架构概述

软件架构的设计是指通过一系列的设计活动，获得满足系统功能性需求、，并且符合一定非功能性需求约束的软件架构模型。软件架构设计过程的本质在于：将系统分解成相应组成成分（如构件、连接件），并将这些成分重新组装成一个系统。现阶段，软件架构设计方法大多侧重对系统 NFR 的考虑，往往和软件架构分析方法结合使用，

希望能够在软件生命周期前期发现潜在的风险。根据软件架构设计方法所面向的目标不同，可以将软件架构设计方法分为面向单系统的架构设计方法、面向产品线的架构设计方法和同时支持单系统和产品线的架构设计方法。

4.2 主要设计方法

4.2.1 基于软件架构设计方法

基于架构的设计 ABD（Architecture Based Design）方法提供一个产生系统概念性架构的结构。概念性架构描述系统的主要设计元素以及它们之间的联系。概念性架构代表开发过程的第一步设计选择，为实现目标功能奠定一个至关重要的基础。ABD 方法确定系统的架构驱动因素（即那些影响架构的商业、质量和功能需求的组合）。一旦确定了系统的架构驱动因素，ABD 设计活动就开始。这并不意味者需要预先完成所有的需求分析、规格书编写工作。需求分析可以与 ABD 活动并行进行。有些应用不可能预先确定所有的需求，例如具有很长生命周期的系统或产品线，因此快速启动设计的可能性就非常重要。

4.2.2 基于属性驱动的设计方法

ADD（属性驱动设计）方法将一组质量属性场景作为输入，并使用对质量属性实现和架构设计之间关系的理解来设计架构。ADD 是一种定义软件体系结构的方法，该体系结构在软件必须满足的质量属性之上构建模块分解过程。它是一种递归分解过程，其中在每个阶段选择架构模式和策略以满足一组质量属性场景，然后分配函数以实例化由模式提供的模块类型。ADD 的结果是模块的体系结构分解视图和其他视图的前几个级别，而不是视图的所有细节都是通过 ADD 获得的。ADD 的结果是粗粒度的，并且 ADD 获得的架构与已经为实现准备的架构之间的差异在于需要做出更详细的设计决策。

ADD 方法的步骤如下：一个。从一组特定的质量方案和功能要求中选择体系结构驱动程序。

4.3 主要步骤阶段

- 预备架构阶段
- 概念设计阶段
- 细化架构阶段

5. 软件架构各阶段设计内容

（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。

6. 软件架构设计的相关内容如风格、模式等

6.1 软件架构的分层模式

分层模式可能是最著名的软件体系结构模式之一。许多开发人员使用它，却不知道它的名称。这样做的目的是将你的代码划分为“层”，其中每个层都有一定的责任，并向更高层提供服务。没有预先定义的层数，但你最常看到的是这些层：表示层或UI层、应用层、业务或域层、持久化或数据访问层以及数据库层。其思想是用户通过执行一些操作(例如，单击一个按钮)在表示层中启动一段代码。然后，表示层调用底层，即应用层。然后我们进入业务层，最后，持久化层将所有内容存储在数据库中。因此，较高层依赖并调用较低层。

6.2 软件架构风格

架构风格也被称为架构模式，换句话说是架构方面的套路。也可以说是前人在架构方面总结出来的，用以解决特定问题的方法。下面来看下官方定义：架构风格是用来描述某一特定应用领域软件系统的组织方式的惯用法，反映了众多系统所具有的结构和语义特性，指导如何使用构件构造一个完整的系统。常用的架构风格。包括：管道过滤器风格、面向对象风格、分布式架构、SOA风格、微服务风格等等。