重庆交通大学信息科学与工程学院

大作业报告

论文题目	软件系统架构综述
课程名称	软件系统架构
专业班级	计算机科学与技术 1601 班
学　　号	631607040104
姓　　名	刘强
任课教师	王家伟

评语：			
成绩：			
评阅人：		评阅时间	

2019 年 06 月

重庆交通大学信息科学与工程学院大作业任务书

课程	软件系统架构	班级	2016 级 1-4 班	任课教师	王家伟
题目	软件架构设计综述		完成时间	第 18 周	
主要内容	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				
报告要求	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。**				
版面	见附件中“论文模板”				

摘要

软件架构又被称为软件构件结构 SCI(Software Component Infrastructure)或软件体系结构，是对软件系统整体结构的刻画。当下，随着软件的规模变大变复杂，对整个系统结构的设计显得尤为重要，对于大规模的软件系统，系统结构设计比真正的编码阶段更重要。如果没有做好系统架构，可能会导致大量修改代码，甚至是重构系统。

关键字：软件；架构；软件系统架构；

目　录

1. 软件架构的概念

1.1 介绍

软件架构是一个易于理解的概念，并且大多数工程师直观地感受到，特别是在一点经验的情况下，但很难精确定义。特别是，很难在设计和架构之间划清界限 – 架构是设计的一个方面，集中于某些特定功能。

在“软件架构简介”中，认为软件架构是一个涉及问题的设计层面：“超越计算的算法和数据结构；设计和指定整个系统结构成为一种新的问题结构问题包括总体组织和全球控制结构；通信，同步和数据访问协议；设计元素的功能分配；物理分布；设计元素的组成；扩展和性能；以及设计备选方案之间的选择。

但是建筑不仅仅是结构;IEEE 架构工作组将其定义为“其环境中系统的最高级概念”。它还包括系统完整性，经济约束，美学考虑和风格的“契合”。它不仅局限于内向，而是在整个系统的用户环境和开发环境中考虑整个系统 – 一个外向的焦点。

1.2 架构观点

我们选择在多个架构视图中表示软件架构。每个体系结构视图都针对特定于开发过程中的利益相关者的一组特定问题：用户，设计人员，管理人员，系统工程师，维护人员等。

这些视图通过展示如何将软件体系结构分解为组件以及组件如何通过连接器连接以生成有用的表单来捕获主要的结构设计决策[PW92]。这些设计选择必须与要求，功能和补充以及其他约束相关联。但这些选择反过来又对需求和未来设计决策产生了进一步的限制。

1.3 架构中心

虽然上面的观点可以代表系统的整个设计，但架构只关注一些特定的方面：

1）模型的结构 - 组织模式，例如，分层。

2）基本要素 - 关键用例，主要类，共同机制等，而不是模型中存在的所有元素。

3）一些关键场景显示了整个系统的主要控制流程。

4）该服务，捕获模块化，可选功能，产品线方面。

本质上，架构视图是整个设计的抽象或简化，其中通过将细节放在一边使重要特征更加明显。在推理时，这些特征很重要：

1）系统演变 - 进入下一个开发周期。

2）在产品线的上下文中重用体系结构或部分体系结构。

3）评估补充质量，例如性能，可用性，可移植性和安全性。

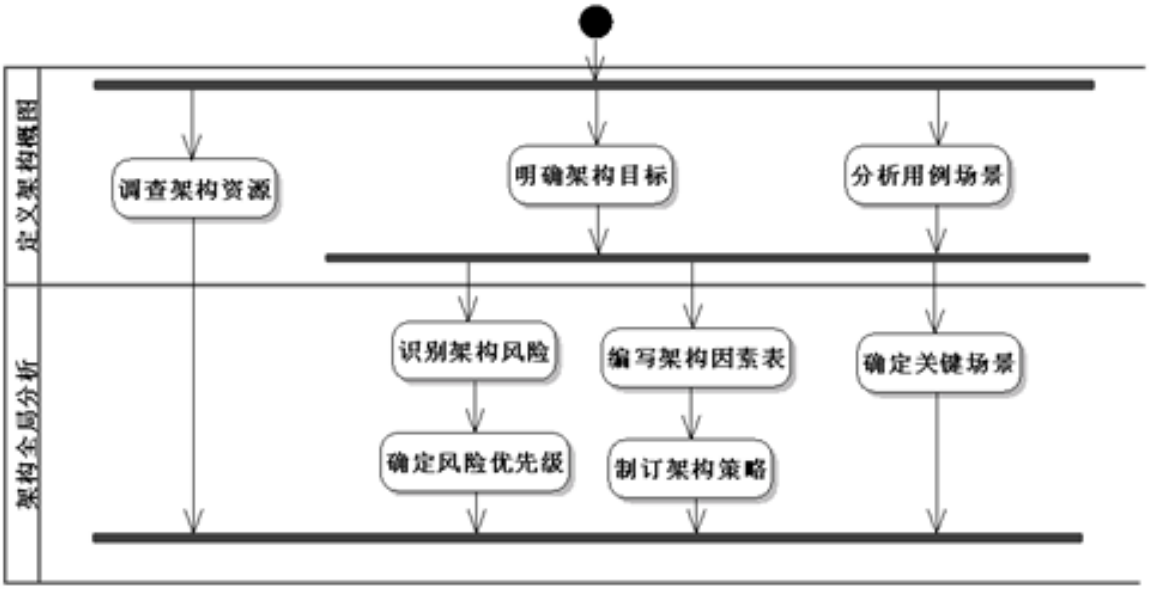
4）将开发工作分配给团队或分包商。

5）关于包括现成组件的决定。

6）插入更广泛的系统。

2. 软件架构的目标

在 MMN 的宏观视图中，包括了三个过程环节：定义架构概图、架构全局分析以及构建概念模型。这是一个循序渐进的过程，是系统架构整体分析的逐步细化。这个过程的关键是找准架构分析的切入点。这正是定义架构概图所要解决的问题。定义架构概图需要明确架构目标、调查架构资源和分析用例场景。这三个活动可以是并行的，至少彼此之间是相互影响、相互作用的。如下图所示：



虽然这些活动是并行的，但从一开始明确架构目标才是最佳的选择，因为架构目标是整个架构过程所要努力达到的方向。不了解架构目标，搭建出来的系统架构再好，也可能不符合客户的需求。架构目标来源于需求，主要指客户或其他利益相关人提出的项目（产品）愿景。愿景表达了客户的目标以及对系统的期望。从愿景中我们可以获得许多架构分析所需要知道的知识，例如明确客户最期望达到的目标，以此可以确定场景与风险的优先级；了解客户的不同目标，可以由此识别系统客户的不同角色，明确不同的利益相关人的态度。

3. 软件架构的种类

3.1 软件架构的类型

1） 软件架构设计考虑系统结构和要求，以获得成功的系统架构。

2） 重点关注那些有助于您创建架构的事情。

3.2 良好架构的重要特征

1） 架构应该尝试满足多个利益相关者的要求。

2） 它应该处理功能和质量要求。

3） 它应该实现所有用例，场景和隐藏实现细节。

3.3 软件架构的类型

3.3.1 业务架构

业务体系结构定义了企业内的业务，治理，组织和关键业务流程的策略，这种类型的体系结构侧重于业务流程的分析和设计。

3.3.2 应用架构

应用程序体系结构用作单个应用程序系统的蓝图，它们与组织业务流程的交互和关系。

3.3.3 信息架构

信息架构定义逻辑和物理数据资产和数据管理资源。

3.3.4 信息技术架构

IT 体系结构定义了构成组织整体信息系统的硬件和软件构建块。

4. 软件架构的设计方法及阶段

4.1 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 软件架构的设计方法

4.2.1 基于架构的设计方法

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。ABD 方法的基础是以下思想：

1） 基于耦合和内聚的思想进行功能分解；

2） 通过选择架构风格实现质量和商业需求；

3） 使用软件模板描述特定类型的软件系统。

ABD 方法的步骤如下：

1） 功能分解；

2） 选择架构风格；

3） 为架构风格分配功能；

4） 精化模板；

5） 验证功能；

6） 生成并发视图；

7） 生成部署视图；

8） 验证质量场景；

4.2.2 属性驱动的设计方法

属性驱动的设计 ADD (Attribute Driven Design)方法把一组质量属性场景作为输入，利用对质量属性实现与构架设计之间的关系的了解，对构架进行设计。

ADD 是一种定义软件构架的方法， 该方法将模块分解过程建立在软件必须满足的质量属性之上。它是一个递归的分解过程，其中在每个阶段都选择构架模式和战术来满足一组质量属性场景，然后对功能进行分配，以实例化有该模式所提供的模块类型。

ADD 的结果是架构的模块分解视图和其他视图的最初的几个层次，不是视图的所有细节都是通过 ADD 得到。

ADD 的结果是粗粒度的，由 ADD 得到的构架和已经为实现做好准备的构架之间的区别是：需要做出更详细的设计决策。

ADD 方法的步骤如下：

1）从具体的质量场景和功能需求集合中选择构架驱动因素。

2）选择满足构架驱动因素的构架模式。

3）验证用例和质量场景并对其进行求精，使它们成为子模块的限制。

对需要进一步分解的每个模块重复上述步骤。

4.3 软件架构的阶段

软件架构有三个阶段：

1） 预备架构阶段

2） 概念设计阶段

3） 细化架构阶段

5. 架构设计每个阶段的内容

5.1 预备架构阶段

5.1.1 需求结构化

可以全面理解需求的各个层次、各个方面，更为分析需求之间关系、识别遗漏需求、发现延伸需求奠定基础。

5.1.2 分析约束影响

对架构设计而言，来自方方面面的约束性需求中潜藏了大量风险因素。因此，必须主动分析约束影响，识别架构影响因素，以便在架构设计中引入相应决策，尽早识别风险。

约束对架构设计有重大影响，约束分类直接体现“这些约束来自哪些涉众”。将约束分为 4 类：业务环境、使用环境、构建环境影分别考虑客户、用户和开发方 3 类涉众，技术环境和 3 类涉众均有关。

5.1.3 确定关键质量

在需求结构化的基础上，确定关键质量着重完成两项任务：

1） 根据系统所在领域的特点及系统规模等因素，确定架构设计重点支持哪些质量属性；

2） 分析上述质量属性之间的制约关系，第一时间指定权衡折中的具体策略。

5.1.4 确定关键功能

架构设计不能把全部功能作为驱动因素，应该以需求结构化为基础，确定关键功能，将从“所有功能”中筛选出对架构设计影响最大的一个功能子集。

功能需求的数量相当大，一般通过选取不到 20%的典型功能需求（不是唯一标准），进行

有重点的深入分析来带动架构设计。如果再考虑到需求变更的问题，在架构设计期间 80%的功能需求的变更都不会对架构设计的推进造成冲击。

5.2 概念架构阶段

5.2.1 概念架构设计步骤

1）初步设计

基于关键功能，借助鲁棒图进行以发现职责为目的的初步设计。

必要性：初步设计并不是总是需要的，复杂系统需要

目标：发现职责。勿需展开架构设计细节。因 PA 阶段未对系统做任何“切分”。

作用：是后续架构设计工作的基础。初步设计识别出了职责，是后续高层分割方案的基础。因为每个“高层分割单元”都是职责的承载体，而分割的目的也恰恰在于规划高层职责模型。

2）高层分割

对系统这个黑盒子进行高层切分，如将复杂系统分为多个二级系统，或者直接切分系统为具体的子系统。

3）考虑非功能需求

概念架构并不等于理想架构。不能只考虑功能需求，还必须要考虑非功能需求。

5.3 细化架构阶段

架构设计图是一种设计架构、描述架构的核心手段和方法。通过架构视图作为分而治之的手段，使架构师可以分别专注于架构的不同方面、相对独立分析和设计不同子问题。软件架构设计之细化架构设计提供 2 视图设计法和 5 视图设计法两种设计方法。

5.3.1 2 视图设计方法

在多种架构视图中，最常用的是逻辑架构视图和物理架构视图，采用二者进行细化架构设计称为 2 视图设计方法。

软件的逻辑架构规定了软件系统是由哪些逻辑元素组成的，以及这些逻辑元素之间的关系。具体而言，组成软件系统的逻辑元素可以是逻辑层、功能子系统、模块。

软件的物理架构规定了组成软件系统的物理元素，这些物理元素之间的关系，以及它们部署到硬件上的策略。

逻辑架构视图和物理架构视图两个视图的设计交替进行、迭代展开。逻辑职责的划分逐步清晰，促进了物理分布设计；反之亦然。

5.3.2 5 视图设计方法

和 2 视图方法相比，5 视图方法适合更大型的软件，它更全面地覆盖了架构设计的各个方面。架构师可以站在 5 个不同的“思维立足点”、分而治之、分别设计，具体方法描述如下：

逻辑视图设计是面向对象或结构化，目标是设计职责划分和职责间协作，例如分模块、分层、划分垂直功能子系统，为模块、层、子系统定义接口。开发视图设计是面向文件，目标是设计程序单元和程序单元组织，例如，开发语言选型、应用程序框架选择、编译依赖关系等。运行视图设计是面向控制流，目标是设计控制流和控制流组织，例如，多进程技术、多线程技术、中断服务程序等物理视图设计是面向节点，目标是设计物理节点和物理节点拓扑，例如，PC 机、服务器、单片机的机型与拓扑连接等数据视图设计是面向表或文件，目标是设计数据存储格式和持久化设计，例如，关系数据库、实时数据库、文件等。

6. 软件架构设计的相关内容如风格、模式等

6.1 软件架构的风格

软件体系结构或体系结构视图具有称为体系结构样式的属性，其减少了可供选择的可能形式的集合，并且对体系结构施加一定程度的一致性。样式可以由一组模式定义，或者通过选择特定组件或连接器作为基本构建块来定义。对于给定的系统，可以通过 RUP 中的项目特定指南工作产品提供的体系结构样式指南中捕获一些样式作为体系结构描述的一部分。风格在架构的可理解性和完整性方面起着重要作用。

6.2 软件架构的模式

软件架构的模式是现成的形式，可以解决重复出现的架构问题。架构框架或架构基础架构（中间件）是一组组件，您可以在其上构建某种架构。在主要的软件开发开始之前，我们必须选择一个合适的体系结构，它将为我们提供所需的功能和质量属性。因此，在将它们应用到我们的设计之前，我们应该了解不同的体系结构。

目前常见的软件架构模式有 10 种：分层模式，客户端 – 服务器模式，主从设备模式，管道 – 过滤器模式，代理模式，点对点模式，事件总线模式，模型 – 视图 – 控制器模式，黑板模式，解释器模式。

参考文献

[1] 李志杰 软件架构设计方法综述