**“卓越软件架构与设计”课程内容**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | 主题 | **内容概要** | | **第一单元 架构与架构视图** | 软件架构是针对软件系统、子系统以及模块层次的设计过程，包括如何组织系统组件，管理组件之间关系以及指导设计的基本原则。  1. 架构的定义  （1）架构的概念与定义  IEEE对架构的定义。对组件的理解，对自治组件与服务的分析；组件与环境的关系。  案例分析：大型银行与保险系统的客户核心系统的架构演化，通过分析系统集成、部署以及系统间通信的方式，深入对架构的理解。  Kruchten对架构的定义。提出了架构风格的概念，并体现了架构演化的思想。结合Spring、Linux的设计思想，初步建立架构风格的基础知识。  （2）识别可重用的架构资源  介绍架构资源元模型，分析开发期资源和运行期资源。通过案例介绍参考架构、架构风格与架构模式在软件架构过程中的运用。  案例分析：商业智能SaaS平台系统中，架构对整个系统设计的指导意义。以真实案例剖析对可重用资源的运用。例如在商业智能SaaS平台中，选择了SaleForce作为参考架构，对系统架构设计产生的影响。  2、架构视图  （1）分而治之的设计原则  软件系统的复杂度主要来自其规模。规模越大，复杂度越高，降低复杂度的最有效途径就是缩小规模，解决之道就是“分而治之”，核心思想是关注点分离。  在架构设计中，通常运用“视图”的概念对整个系统进行分解。通过不同涉众不同角色的不同观察视角，对系统进行划分。  （2）架构视图实例讲解  a. Christine提出的架构视图剖析，分别包括概念视图、模块视图、执行视图与代码视图。  案例分析：图像处理系统的架构视图。对图像处理系统进行系统分析，分别建立该系统的概念视图、模块视图、执行视图与代码视图。  b. RUP 4+1视图  介绍RUP 4+1视图的组成以及各个视图之间的关系。  案例分析：LSVT 4+1视图架构规格说明书  一个真实的案例，展现了一个完全按照4+1视图进行设计的软件系统架构。 | | **第二单元 架构风格与参考架构** | 1、REST架构风格  REST描述了Web作为一个分布式超媒体的应用，相互链接的资源通过交换代表资源状态的表述来进行通信。它 是WEB系统架构运用最为广泛的架构风格。 案例分析：订单管理系统的REST架构。通过案例讲述如何在架构设计中运用REST架构。 2、基于消息的分布式架构 分布式架构是企业软件系统主要采用的一种架构风格，通过使用基于消息的中间件完成消息的发送与接收，从而实现系统之间的集成，以及业务处理的异步模型。 案例：医疗卫生知识库系统。通过引入消息队列改善系统架构的质量。 3、数据为中心的软件架构 一般的数据管理系统都分为三个步骤：Data Ingestion、Data Storage与Data Processing。在大数据处理中，这种模型体现得更为明显。所有的软件系统都离不开数据处理。此外，本节内容还会讲解MapReduce架构风格，剖析MapReduce的架构原理和最佳实践。 案例：Twitter数据管理与分析，运用Apache HDFS、Flume、HIVE，完成对Twitter数据的管理与分析。 4、基于云的软件架构 基于云的软件架构可以分为三个层次，即SaaS，PaaS和IaaS。本节主要讲解基于元数据的多租户SaaS架构。  案例：商业智能SaaS平台的整体架构 | | **第三单元 架构模式与应用实践** | 1、分层架构模式与实践  讲解经典的软件分层架构。  2、微内核架构模式与实践  微内核模式是架构模式中极为重要的一种模式，尤其是它划分功能子集为核心功能子集的设计思想非常重要，但它的重要性却常常被人忽略。  案例：商业智能SaaS平台与Spring框架对微内核模式的运用  3、管道-过滤器架构模式与实践  若要实现数据处理的良好可扩展性，有效降低数据处理的算法复杂度，就需要运用管道-过滤器模式。  案例：商业智能SaaS平台的核心算法  4、MVC架构模式与实践  MVC架构模式是最常用的架构模式，体现了关注点分离的架构原则。在介绍MVC模式的同时，还将深入分析与比较MVC模式与MVP模式之间的异同。  案例：Ruby On Rails的MVC架构  5、CQRS架构模式与实践  CQRS模式即命令查询职责分离模式，是DDD中基于事件的读写分离架构模式。将业务逻辑建模为状态机模型，并利用松散耦合的命令与事件机制，采用异步模型改善系统整体性能。  案例：会议注册与管理系统的CQRS架构 | | **第四单元 架构核心原则与设计策略** | 1、设计简单的架构  （1）设计的意图：通过清晰地表达设计意图，以简化系统的整体架构，并有利于设计者与开发者之间的沟通。  案例分析：Ruby On Rails的架构设计  （2）保证系统足够小，促进恰如其分的架构设计  （3）分而治之与关注点分离：遵循“关注点分离”的架构原则，将架构的分离策略分为纵横分离与内外分离。  （4）如何化繁为简：通过专注实体的职责，整体分析系统目标、范围与核心概念；并通过抽象简化简化整个系统的模型。  （5）设计的演进：通过具体案例讲解如何运用风险驱动进行架构的设计与演化。  2、设计一致的架构  （1）设计风格的一致性：合理运用前面讲到的架构风格，针对不同场景做出正确的架构决策。分析基于约束对架构风格的驱动设计方式。  （2）概念的一致性：如何通过业务分析甄别领域概念，对整个系统的领域逻辑进行建模，并利用抽象简化模型。通过引入DDD的概念，以确定一致的设计概念，以便于开发团队的沟通和协作。  （3）解决方案的一致性：要求在系统设计时，确保针对同一个问题域采用一致的解决方案。  （4）路线图：确定产品线，制定路线图，作为保证一致架构的高层蓝图。  3、设计自治的架构  （1）最小完备特征：设计最小完备的对象、组件与服务。通过识别设计特征来判断实体是否符合最小完备原则。  （2）自我履行特征：设计智能有意识的对象，可以通过从调用端和提供行为的一方双向地分析，以改善职责的分配。  （3）稳定空间特征：合理的封装，维持实体的边界，并提供开放空间以满足系统整体架构的可扩展性。划定软件系统的逻辑边界与物理边界，保证系统的可重用性。 （4）独立进化特征：通过引入契约和接口，制定规范，以设计具有可进化性的系统。在设计时，考虑新旧版本的兼容性，为系统建立防腐层隔离变化。 | | **第五单元 MMN架构设计过程** | MMN架构设计过程是指对系统架构从宏观、微观与纳米层面的整体设计过程。这是一个迭代和演进的设计过程，通过自顶向下结合自下而上的方式，对整个软件系统进行分析与设计，保证整个软件系统满足功能需求与质量属性。  1、宏观视图的架构因素与设计过程  （1）定义架构概图：包括调查架构资源，明确架构的目标，根据架构目标做出重要的设计决策，并分析主要的用例场景，以建立一个粗略的架构概图。  案例分析：企业应用套件的架构概图  （2）架构全局分析：识别架构风险，并确定风险优先级。然后根据识别出来的风险编写架构因素表，制订具体的架构策略。同时确定整个系统的关键场景。  案例分析：远程访问的架构策略  （3）构建概念模型：确定技术框架与技术选型，识别并分析软件产品的设计约束，从而确定架构风格，并根据具体场景运用架构模式。  案例分析：CIMS架构概念模型  （3）建立系统的逻辑视图和物理视图  案例分析：集团智能信息辅助系统的架构概念模型  2、微观视图的架构因素与设计过程  （1）细化逻辑视图：进行领域分析，确定系统的应用逻辑架构与业务逻辑架构，并设计整个系统的模块视图；  案例分析：汽车零售管理系统领域分析  案例分析：商业智能SaaS平台逻辑架构  （2）细化物理视图：执行运行时分析，确定通信机制和资源分配，并以此为根据设计部署视图和执行视图  案例分析：燃气集团解决方案  3、纳米视图的架构因素与设计过程  （1）构建设计模型：讲解职责驱动设计，通过角色、职责与协作完成对象的职责分配，并通过识别变化点，利用抽象对变化进行封装，以及合理运用设计模式。  案例分析：数据分析器  案例分析：商业智能SaaS平台引擎设计  （2）代码视图：包括确定部署组件、配置管理、持续集成等与代码级别有关的内容。  案例分析：集团智能信息辅助系统的代码视图  案例分析：商业智能SaaS平台的代码视图 | | **第六单元 大型软件系统体系架构** | 1、在线零售商集成解决方案  整个系统牵涉到电子商务、库存管理、呼叫中心、邮件服务等多个系统的集成。该解决方案通过运用分布式系统的最佳实践，运用基于消息的中间件，对系统进行整体设计，使得系统能够高质量地支撑在线零售商的核心业务。   2、银行保险客户核心支撑系统真实案例，是某大型金融集团的客户核心支撑系统，需要支持的业务系统多达数十个，且具有不同的业务，部署在不同的平台。如何通过合理地设计，运用ESB和REST对整个系统进行集成。 | |