1. 软件架构设计的What&Why？

What？

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，组建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。

why？

业务需求层出不穷；软件系统越来越复杂；参与的人越来越多；共性和特殊性的问题越来越多；技术发展日异月新；……

2. 软件架构师

2.1. 软件架构师是干什么的

|  |  |
| --- | --- |
| 介于需求与开发的中间人 | 良好的沟通能力 |
| 能够统领全局的大牛 | 良好的大局观 |
| 能够将需求转换为技术 | 洞悉前沿与市场嗅觉 |
| 能够为软件研发提供指导 | 见多识广的大牛 |
| 需要全面思考软件系统方方面面的问题 | 缜密地思考问题 |
| 能够攻关和搞定重要技术难题 | 公司可信赖的支柱 |

2.2. 架构师的素质

|  |  |
| --- | --- |
| 全局思维 | 从业务、市场，到技术实现；  从软件的过去、现在，到将来；  从外部客户，到内部研发；  从软件研发，到硬件部署；  从功能实现，到运行效率。 |
| 战略思维 | 在所在行业的发展战略；  在业务领域的发展战略；  在技术方向的发展战略；  在潜在市场的发展战略。 |
| 前瞻思维 | 市场趋势的发展动向；  前沿技术的发展动向；  竞争对手的发展动向；  合作伙伴的发展动向。 |
| 抽象思维 | 各项业务需求：抽象成功能模块；  各项功能的实现：抽象成软件架构。 |
| 逆向思维 | 假如不实现会怎样？  假如没搞定会怎样？  假如没有它会怎样？  假如被延期会怎样？ |

2.3. 架构师的分类

随着行业和社会的发展，架构师的定义和分类越来越广泛和细分，广泛和细分其实并不矛盾，如果“广泛”是x轴，“细分”是y轴，则二维坐标系x和y轴中间的任一点就是一种架构师类别。但总体来说，或目前来说，集合业界的大致认知，总结如下：

|  |  |  |
| --- | --- | --- |
| No. | 分类 | 描述 |
| 1 | 解决方案架构师 | 与客户探讨业务需求，将业务、市场，与技术、产品结合起来，为客户提供解决他们需求的方案。 |
| 2 | 系统架构师 | 也称应用架构师。最终确认和评估系统需求，并将业务转换为技术，为研发人员制订核心框架与技术规范 为研发工作澄清技术细节并扫清技术障碍 。 |
| 3 | 平台架构师 | 这里的平台其实包括两个平台，一个是系统平台，也就是负责搭建多个系统整合的系统应用平台；另外一个其实是基础平台，是专门负责搭建基础技术平台；两者其实区别蛮大，也经常容易被从业人员混淆。举个简单例子，金蝶有平台架构师一职，但是金蝶BOSS应用和金蝶中间件两者招聘的对象和技术要求是截然不同的。 |
| 4 | 业务架构师 | 业务架构其实已经开始脱离技术层面了，但是它要求架构师有跨越多系统的大局观，去整合和组织不同系统的技术平台与交互模式。其实这个职位的未来也就是CIO了。 |
| 5 | 网络架构师 | 过去，我们可能听的最多的是网络工程师。不错，一个优秀的网络架构师必须有足够的网络技术基底，并且它的关注点也是系统的基础架构。比如说如果搭建并优化集群环境，如果构建基于云计算的系统应用与部署等等。它对于像淘宝、腾讯这样的互联网公司是极其重要的。 |
| 6 | 移动架构师 | 移动互联网的迅猛发展横向和纵向都细分出了很多新的职责和岗位，移动架构师的职责和作用日益重要，既要整体和全局考虑整个前后端的软件系统架构，又要重点深入移动客户端的架构设计的方方面面，既要有跨平台思维，又要拿捏好原生和混合开发的尺度，另外移动应用的特点，导致移动架构师必须要比传统系统架构师更加注重非功能性的质量属性。 |
| 7 | 前端架构师 | 这也是移动互联网的迅猛发展而细分出来的新的职责和岗位，这里的前端特指网站开发中的前端，主要考虑前端呈现层的设计（HTML/CSS/JS/AJAX/RIA/…），跨浏览器设计等等。 |
| 8 | ... | ... |

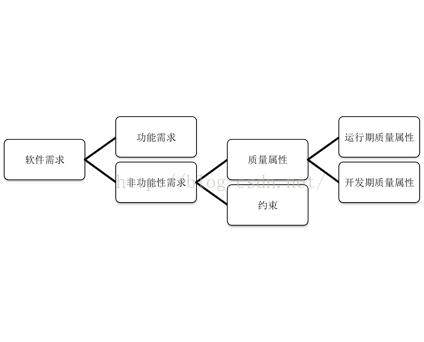
3.  软件需求

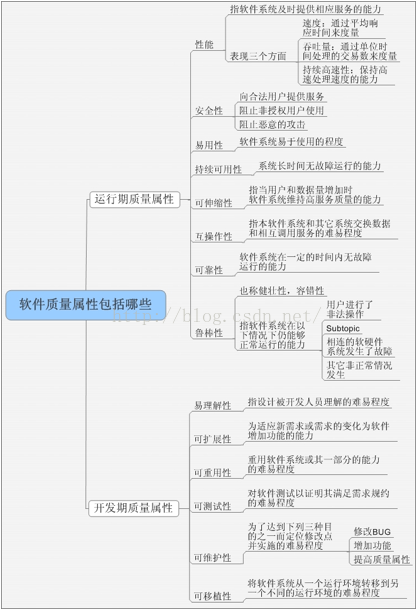
软件架构设计中常说需求驱动架构设计，可见需求在整个架构设计中起到了关键指引和方向的作用，如果以目标导向为原则，则需求的满足和实现就是架构设计的终极目标。  
OK，在进行架构设计之前，我们先看下软件需求。

3.1.  软件需求怎么来的（软件需求的过程）

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 阶段 | 需求阶段 | 说明 | 什么人做什么事 | | 可能产生的文档 | |
| 客户方 | 实施方 | 客户方 | 实施方 |
| 1 | 业务需求 | 来自客户的原始需求，背景描述，业务诉求和期望目标。 | 项目负责人或接口人描述需求或提供客户方需求文档。 | 销售或售前接触客户，听取和记录需求。 | 工作说明书（SOW），或邀标书 | 售前的方案建议书 |
| 2 | 用户需求 | 通常是在问题定义的基础上进行用户访谈、调查，对用户使用的场景进行整理，从而建立从用户角度的需求。 | 项目负责人或接口人接受访谈和调研。 | 需求分析师或项目经理等进行需求调研。 |  | 调研计划、用户需求调研提问库、调研日志、用户需求说明书 |
| 3 | 软件需求 | 从系统实现的角度描述的需求。开发人员（设计和分析人员）在业务需求、用户需求的基础上形成的。 |  | 需求分析师或项目经理、架构师等讨论和细化需求，编写需求文档 |  | SRS（Software Requirement Specification）软件需求规格说明书 |

3.2.  软件需求的分类：





另外，也有McCall软件质量模型：



注：在架构设计中，对非功能性需求的重视程度，也会影响架构设计的好与劣；但也要平衡过渡设计和适可而止的关系。

4. 软件架构的过程

业界软件架构设计的方法论很多，各有各自的应用场景和特点，下文结合ADMEMS（Architecture Design Method has been Extended to Method System）架构设计方法论说明软件架构的过程：

|  |  |  |  |
| --- | --- | --- | --- |
| 架构阶段 | 目标 | 方式方法 | 现实工作场景 |
| 预架构阶段 | 全面理解需求；需求结构化，摒弃“需求列表”，建立二维需求观（ADMEMS矩阵）。 | 使用ADMEMS矩阵方法，捋清需求间关系和发现衍生需求。 | 1、与人：与项目经理、需求分析师等内部需求人员了解需求；与客户了解需求（不建议架构师做需求分析师角色）。 2、与物：了解《需求规格说明书》等需求文档。" 3、对需求有什么问题，反馈给售前或销售，可能会参与拜访客户或电话会议。 4、销售或售前有时会要求提供一个大致的工作量，以便他们初步评估项目可行性。 |
| 概念架构 | 高层组件及其关系 | 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（不是必须）。 2、高层分割，将复杂系统切分为多个二级系统或多个子系统。 3、考虑非功能需求，采用ADMEMS推荐的目标-场景-决策表。 | 1、参与内部讨论：项目可行性分析、讨论，从需求、技术、人力、风险等角度提供建议。 2、项目投标准备：参与投标团队的技术方案编写，编写系统架构章节，解决招标书上技术问题的问答。 3、参与项目讲标：作为讲标团队成员参与项目讲标，负责技术问答环节的应对。 |
| 细化架构 |  | 5视图法 | 在项目概要设计阶段，进行架构设计，制定规范和约定，为详细设计提供指导。 |
| 实现 | 详细设计 编码实现 | 架构设计形成详细设计文档 | 在项目实现阶段，对开发人员提供规范指引和技术支持。 |

注：架构设计的过程和内容的不是固定不变的，现实中，比如售前提供解决方案中，很多时候需要架构师提供细化架构中才会深思的逻辑架构、物理架构等，这时候，架构师就需要有螺旋思维和跳跃思维的方式，就像武功中，招式是死的，人是活的，要学会灵活运用。

5.  软件架构设计的方式方法

5.1.  多视图法

多视图方法是业界广泛认同的一种架构设计思路，包括：  
● SEI的3视图法：  
模块视图、组件-连接器视图、分配视图。  
● 西门子的4视图法：  
概念视图、模块视图、代码视图、执行视图。  
● RUP的4+1视图法：  
用例视图、逻辑视图、开发视图、进程视图、物理视图。  
● 联邦企业架构框架：  
技术架构视图、信息架构视图、应用架构视图、业务架构视图。  
● ……

5.2.  5视图法



5视图法分析的意义：

● 全面分析软件系统方方面面的问题  
● 尽早地发现和排除项目风险与不确定因素  
● 从不同角度去展现要设计的软件系统  
● 为项目进行中不同的干系人提供指导：  
    -- 逻辑架构描述系统功能，并指导系统测试  
    -- 开发架构规范软件的层次及代码风格  
    -- 数据架构指导数据库的设计  
    -- 运行架构定义了一些关键过程的设计  
    -- 物理架构明确软件如何部署与实施

5.3.  5视图法设计步骤

两种观念：

|  |  |
| --- | --- |
| 观念 | 设计步骤 |
| 观念一 | 顺序进行：  1、逻辑架构。  2、开发架构  3、数据架构  4、运行架构  5、物理架构 |
| 观念二 | 5个视图是穿插进行设计的，对复杂系统而言，根本不可能将逻辑视图设计完了后再考虑其它视图。 |
| 笔者观念 | 观念一和观念二的情况在现实状况中都可能存在，要根据具体情况具体分析，但总体而言，5视图穿插进行思考更有利于思考的全局性和完整性。 |

5.4.  如何进行5视图法的设计

以下5视图表格中的工具和方法每个架构师或略有差异，以下仅为参考。

5.4.1.  逻辑架构

逻辑架构的重点是考虑软件功能性需求。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| No. | 考虑的方面 | 产出物 | 工具 | 说明 |
| 1 | 系统功能划分为几个子系统与功能模块？ | 系统功能树 | 树型结构图 |  |
| 2 | 向什么用户提供什么样的功能？ | 用例模型 | UML用例图 | 体现用户和行为 |
| 3 | 每个功能都是怎样的操作流程与分支？ | 用例描述 | 用例描述表 | 含输入输出、事件流分析；  不要有界面描述 |
| UML活动图 | 进行业务流程分析，包括泳道图 |
| 4 | 如何通过界面与用户交互？怎样交互？ | 鲁棒分析 | 鲁棒图 | 通过对“用例描述表”进行原文分析法拣出名词和动词 |
| 5 | 应当设计哪些类与界面？怎样设计？ | 领域模型 | UML类图 |  |
| 6 | 与哪些外部系统接口？怎样接口？ | 接口描述 | UML类图 |  |

**5.4.2.**  **开发架构**

开发架构重点关注的是开发编码实现方面的问题。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| No. | 考虑的方面 | 产出物 | 工具 | 说明 |
| 1 | 分层结构设计 | 分层架构图（开发架构图） | 各种绘图工具 | 好的分层结构支持自动化测试 |
| 2 | 开发技术选项 | 开发语言  开发框架  开发工具 |  | 考虑商用产品、开源框架、自研框架 |
| 3 | 模块划分 | 源码工程；Project目录结构；  分包(分库) |  |  |
| 4 | 开发规范 | 开发/编码规范文档； |  |  |
| 5 | 软件质量属性 | 分析和决策结果 |  | 考虑运行期和开发期软件质量属性，并权衡利弊进行决策。 |

**5.4.3.**  **数据架构**

数据架构不仅仅要考虑开发中涉及到的数据库，实体模型，也要考虑物理架构中数据存储的设计。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| No. | 考虑的方面 | 产出物 | 工具 | 说明 |
| 1 | 数据是集中还是分布存储的？如何考虑分布式存储？ | 数据架构图 |  |  |
| 2 | 领域模型到数据库表的转换？表结构关系的设计？ | 逻辑模型  物理模型  ER图 | Power Designer  Visio |  |
| 3 | 实体如何设计？充血模型和贫血模型？ | UML类图 |  |  |
| 4 | 使用什么数据库？关系型还是非关系型？ | 选型结果 |  |  |

|  |  |
| --- | --- |
| 关系型数据库 | 非关系型数据库（NoSQL） |
| Oracle（首次发行：1980年）  MySQL（首次发行：1995）  MS SQL Server（首次发行：1989）  PostgreSQL（首次发行：1989）  IBM DB2（首次发行：1983）  Microsoft Access（首次发行：1992）  Sybase ASE（首次发行：1987）  SQLite（首次发行：2000）  …… | MongoDB（首次发行：2009）  Cassandra（首次发行：2008）  Apache CouchDB  Hbase  Redis  db4o  BaseX  …… |

**5.4.4.**  **运行架构**

运行架构关注的不再是全局而是局部，着重关注那些关键点与难点，常常需要技术攻关与预研。主要考虑控制流、通讯机制、资源争用、锁机制、同步异步、并发、串行，同时也要考虑质量属性。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| No. | 考虑的方面 | 产出物 | 工具 | 说明 |
| 1 | 运行：同步vs.异步；并发vs.串行 | 考虑开发架构中代码的实现。 |  |  |
| 2 | 交互：对象间交互；状态转换 | 考虑开发架构的合理性，到类、到接口、到代码。 |  |  |
| 3 | 质量：安全；可靠；可伸缩 | 考虑开发架构的合理性 |  |  |
| 4 | 性能：响应时间；吞吐量 | 估算：  在线人数、并发人数；  每秒事务量；  响应时间。 |  |  |

**5.4.5.**  **物理架构**

物理架构主要考虑硬件选择和拓扑结构，软件到硬件的映射，软硬件的相互影响。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 考虑的方面 | 产出物 | 工具 | 说明 |
| 1 | 网络方面：网络拓扑；网络设备；安全机制 | 拓扑图  安全规范 |  |  |
| 2 | 性能方面：可靠性、可伸缩性 | 需要什么样设备性能 |  |  |
| 3 | 部署方面：集中式还是分布式；组件部署 | 部署图 |  |  |

**6.**  **一个思考，谁驱动了架构设计？**

需求驱动了架构设计？

质量属性了驱动了架构设计（ADD）？

领域驱动了架构设计（DDD）？

风险驱动了架构设计？

质疑驱动了架构设计？

……

到底是谁驱动了架构设计？我们以船舶设计建造为例，来看这些问题：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 目标和结果 à | 问题 à | 回答 à | 小结 | |
| 设计和制造一艘远洋货轮，能经历数月海上颠簸和长途跋涉，并保证货物的安全和完整，最后能顺利抵达目标港口。 | 我们为什么要设计和制造这样的大船？ | 市场有这个需求；  获利很丰厚；  解决就业；  …… | 不管是外部需求还是内部的需求，都是需求。不正是需求驱动吗？ | |
| 如何保证货船能长途航行并经受波涛和风雨？ | 船要造的结实； | 鲁棒性 | 这些不正是质量属性驱动设计吗？ |
| 引擎设计的强劲； | 性能 |
| 船的燃料充足； | 持续可用性 |
| 装备卫星电话、GPS、雷达等设备 | 互操作性 |
| 货物和人员要安全 | 安全性 |
| 船舶设计师怎么设计这样的船舶？ | 现在都会通过计算机进行船舶的CAD和3D模型设计。 | 是不是佷似领域模型驱动了设计？ | |
| 造船一定有很多风险吧？ | 那是，比如订货方客户有时提出改装船舶的意见（需求变化的风险）；有时某些工艺成品率不稳定（质量风险）；等等。 | 所有可能的风险点在设计时都要考虑到，做好预案，才能保证架构设计的可行性和灵活性，风险驱动了架构设计。 | |
| 上面怎么提了那么多问题，其实还有很多问题，比如…… | 嗯，你现在有不少疑问了。 | 不断的质疑架构设计中可能存在各种问题，有质疑才有思考，才有解决方案，从而推动架构设计的不断完善。 | |

**总结：**

**以上几个方面都能驱动架构设计，并不是零和的规则，而是一个立方体从不同方向看的问题。以上方面有些是指导思想，有些是行动方式，有的兼而有之，阐述方式看似不同，终极目标还是造出大船（实现需求），造出好船（质量属性），怎么造（领域模型），造的顺利（风险控制），挑不出毛病（架构师自己先质疑问题并解决了）。**

**7.**  **软件架构设计的误区**

● 高开高走落不到实处

● 理想与现实需要折中

● 遗漏关键性约束与非功能需求

● 为虚无的未来埋单而过度设计

● 过早做出关键性决策

● 客户说啥就是啥成为酱油哥

● 埋头干活儿缺乏前瞻性

● 架构设计还要考虑系统可测性

● 架构设计不要企图一步到位

**8.**  **部分参考资料**

温昱的《一线架构师实践指南》