完成一个比较复杂的项目后，终于有空看看书了，这次决定将架构设计的方法论进行一次系统的学习，借助温昱大师的《一线架构师》一书。我将把这次学习分成三部分，分别是概论&预架构阶段&非功能目标的方法论、概念架构阶段、细化架构阶段。此外，今天看到老妈很喜欢的大幂幂了，真心很美，继续加油学习了，为成为一名合格的程序员而努力。

* 架构师的4个困惑：

|  |  |  |
| --- | --- | --- |
| 4个实际问题的困惑 | 将系统划分模块，如何更合理？ | 细化架构阶段 |
| 大系统架构设计，如何起步？ | 概念架构阶段 |
| 总觉得需求很糟糕，影响了架构设计 | 预架构阶段 |
| 非功能需求重要，但如何设计 | 非功能目标的方法论 |

软件开发在中国也已经有了超过30年的发展，在当前情况下，软件架构的知识体系已经建立，其方法论已经出现，借用温大师的话“架构设计是质疑驱动的”。

需求 = 功能 + 质量 + 约束（架构设计的上下文）

* 架构设计是多阶段、多视图的

|  |  |
| --- | --- |
| 阶段1 | 把握需求特点，确定架构驱动力 |
| 阶段2 | 根据重大需求，确定概念架构 |
| 阶段3 | 细化架构设计，关注不同视图（4+1视图） |

作为架构师，首先面对的风险是需求，既要关注功能需求，还要平衡质量属性，且不能遗漏约束性需求。这部分最重要的是引入一个新的观念，就是二维的需求观，可以通过需求层次-需求方面矩阵来表示。其**应用法则**为：从上到下、从左到右；重点是质量属性遗漏。

|  |  |  |  |
| --- | --- | --- | --- |
|  | 功能 | 质量 | 约束 |
| 业务级需求 | 业务目标 | 快、好、省 | 技术性约束、法规性约束、标准性约束  技术趋势、竞争因素与对手  遗留系统集成、分批实施 |
| 用户级需求 | 用户需求 | 运行期质量 | 用户群特点、用户水平、多国语言 |
| 开发级需求 | 行为需求 | 开发期质量 | 开发团队技术水平、分布情况、磨合程度、业务知识  管理：保密要求、产品规划  安装、维护 |

预架构阶段需要的活动有：需求结构化、分析约束需求、确定关键质量、确定关键功能。在实践中，只要有了明确的业务需求、全面的用户需求和典型的行为需求，就可以开始软件架构设计了。通常来说，不同需求影响架构的不同方面，如下表所示。

|  |  |  |
| --- | --- | --- |
| 操作 | 基本原理 | 对架构设计的影响 |
| 功能 | 功能是发现职责的依据 | 每个功能都是由一条“职责写作链”完成的，架构师通过为功能规划职责协作链、将职责分配到子系统、为子系统界定接口、确定基于结构的交互机制，来推动架构设计进行 |
| 质量 | 质量是完成架构设计的动力 | 基于当前架构设计中间结果，进一步考虑具体质量要求，进行细化、调整  质量和功能功能影响项目 |
| 约束 | 约束对架构设计的影响分为几类 | 直接决定设计决策的约束（系统运行与Unix）  转化为功能需求的约束（本银行遵循人民银行标准引出“利率调整”功能）  转化为质量需求的约束（操作人员水平较差引出易用性需求） |

确定关键质量的五大原则：分类合适+必要扩充、考虑多方涉众、检查性思维(checklist)、识别矛盾+划定优先级、严格程度符合领域与规模特点。接下来首先介绍识别矛盾的工具，**质量属性关系矩阵,**这部分最重要的就是权衡。

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 性能 | 安全性 | 持续可用性 | 可互操作性 | 可靠性 | 鲁棒性 | 易用性 | 可测试性 | 可重用性 | 可维护性 | 可扩展性 | 可移植性 |
| 性能 |  |  |  | - | - | - | - | - |  | - | - | - |
| 安全性 | - |  |  | - |  |  | - | - | - |  |  |  |
| 持续可用性 |  |  |  |  | + | + |  |  |  |  |  |  |
| 可互操作性 | - | - |  |  |  |  |  |  |  |  | + | + |
| 可靠性 | - |  | + |  |  | + | + | + |  | + | + |  |
| 鲁棒性 | - |  | + |  | + |  | + |  |  |  |  |  |
| 易用性 | - |  |  |  |  | + |  | - |  |  |  |  |
| 可测试性 | - |  | + |  | + |  | + |  |  | + | + |  |
| 可重用性 | - | - |  | + | - |  |  | + |  | + | + | + |
| 可维护性 | - |  | + |  | + |  |  | + |  |  | + |  |
| 可扩展性 | - | - |  |  | + |  |  | + |  | + |  | + |
| 可移植性 | - |  |  | + |  |  | - | + | + | - | + |  |

通常来说，对于不同类型的系统，要求的严格程度也是不同的，比如。

|  |
| --- |
| 经验： 项目，3-5项；产品，5-7项；平台，7-9项 |
| 例如  银行项目：易用性、安全性  银证产品：易用性、安全性、互操作性、可扩展性、可维护性  金融平台：安全性、互操作性、持续可用性、性能、可扩展性、可维护性、可重用性、可管理性、开发性 |

最后是这部分内容的核心，即如何确定关键功能，可以借助4条启发规则，核心功能、必做功能、高风险功能和独特功能，案例如下所示。

|  |  |
| --- | --- |
| 运行期质量属性  可伸缩性：木有上限  性能：即强调速度，又强调吞吐量  易用性：最便捷的选择方式  安全性：数据安全  持续可用性：不停机  互操作性：含公司各系统间互操作  开发期质量：  可扩展性 | 终端用户功能  检索/下订单/发货（核心功能）  评价系统（必做功能）  多维度关联信息（必做功能）  最快的全库搜索（必做功能、高风险功能）  管理员功能  灵活的打折设置  频率极高的新货上架 |

B2C平台的6个关键业务链



Tip:关键需求决定架构，其他需求验证架构

首先用一个小笑话来发送一下，项目管理中的4拍型领导：决策时拍脑袋—就这么定了；指挥时拍胸脯—保证没问题；失误时拍大腿—我怎么没想到；追查时拍屁股—老子不干了。

这部分内容虽然很少但也很重要，“目标-场景-决策“表。

|  |  |  |
| --- | --- | --- |
| 目标 | 场景 | 决策 |
| 业务需求和约束 | 项目成员，在客户现场，能够访问PM系统  PM系统的用户，分别在windows，linux，mac上工作 | B/S架构  B/S架构 |
| 互操作性（质量属性） | HR系统，已有员工信息，不再输入而是导入  PM系统的用户也要管理项目文档 | 支持从HR系统导入数据  PM接入GIT |
| 跨平台（质量属性） | DBMS多种类型  服务器跨平台 | ORM  JAVA |

参考资料

1. 温昱. 一线架构师实践指南[M]. 北京:电子工业出版社, 2011.