3.3.1-架构设计指引

Keep It Simple Stupid (保持简单和直接)!

架构设计应该足够简单直接，做到恰如其分，不多也不少！

简单设计也是极限编程实践中的一个重要实践。

恰如其分在设计思想上就是要要确立“小即是美”的思想，**整体架构设计需要遵循“完整、灵活和轻盈”。灵活体现在**能快速地响应变化；**轻盈**体现在一切都刚刚好，没有任何臃肿多余的设计，**完整**体现在简单而不简陋，该有的设计和产物没有残缺，符合“麻雀虽小，五脏俱全”的标准。

良好的软件架构的可以帮助我们：

- 让整个团队跟随一个清晰的技术愿景和技术路线图。
- 对不同层次的干系人提供关于系统的完整、清晰、一致的视图。
- 技术领导力和更好的协调
- 统一的技术术语和模型，更方便团队的交流，减少彼此的误解。
- 识别和控制风险
- 结构良好的代码，为应对变化建立坚实的基础。

1、如何做到“恰如其分”?

关于架构设计，一个最主要的争论是要做到什么地步的预先设计才是刚刚好？传统的瀑布模式强调事无巨细的架构设计与详细设计，甚至写出所有的伪代码。而敏捷方法推崇“浮现式设计”。

其实，敏捷方法从未说过不做架构设计，也没有宣布过 不写文档 。

对于开发之前的预先架构设计，我们认为需要根据项目的实际情况进行取舍，做好能够很好的支持开发工作，控制风险就是刚刚好，具体的一些经验如下：

多少预先设计是太少了?

如果有以下的任何情况，说明预先的架构设计做的不够，如果贸然启动开发会带来巨大风险：

- 不了解系统的边界是什么，在哪里?
- 团队成员对系统的"Big Picture"没有形成共识
- 无法交流整体愿景
- 团队成员对需要做的事情不清楚或感到茫然，不知道该如何开始编码
- 没有考虑非功能需求或质量属性
- 没有考虑环境约束如何影响软件（部署环境等）
- 没有考虑过技术风险
- 尚未确认重大问题的对策
- 没有考虑关注点分离，适当的分层、可修改性，可测试性等
- 团队成员使用的技术选型不一致
- 团队成员对于解决方案是否管用没信心

多少预先设计太多了?

- 很多信息，太多太细的文档
- 在文档中编写太多的伪代码
- 太多图表
- 过于死板，开发人员感觉缺少灵活性，编码变成翻译工作
- 序列图，非常多的乏味的序列图
- 太长的设计时间，还没开发项目截止时间已经快到了

2、架构设计原则

软件产品的技术架构应参考以下原则进行设计，对于无法满足的情况需要在架构评审时进行明确说明。

	原则	说明
1	**结构清晰**	系统纵向的分层和横向的分模块结构清晰合理，层次分明，概念清晰一致

2	**SOLID**	通过有效运用以下六个原则实现逻辑结构和程序结构的清晰灵活，能够应对变化： • 单一职责原则（Single Responsibility Principle - SRP） • 开放封闭原则（Open Closed Principle - OCP） • 里氏替换原则（Liskov Substitution Principle - LSP） • 最少知识原则（Least Knowledge Principle - LKP） • 接口隔离原则（Interface Segregation Principle - ISP） • 依赖倒置原则（Dependence Inversion Principle - DIP）
3	**N+1设计**	关键组件永远不少于两个冗余，通常三个。不引入任何单点故障风险
4	**无状态设计**	只有当业务确实需要的、没有任何其他方案替代时才使用状态
5	**配置外置**	发布包中不能包含任何环境相关的配置信息，应通过环境变量等方式注入
6	**功能开关**	能够不停机打开或关闭任何发布的功能
7	**故障隔离**	实现故障隔离设计，通过断路保护避免故障传播和交叉影响
8	**异步**	只有在绝对必要的时候才进行同步调用
9	**监控设计**	在设计阶段就必须要充分考虑如何监控，无监控不上线；
10	**日志流**	仔细设计日志的输出，把日志当作事件流输出和使用
11	**弹性伸缩**	支持弹性伸缩，启动迅速、优雅终止、面对突然伸缩时保持健壮。
12	**水平可扩展**	扩展方法要选择水平扩展非垂直升级：永远不要依赖更大，更快的基础设施
13	**自动化**	对开发、测试和生产环境都进行了完整的规划设计，尽一切可能实现各个环境实现发布和变更自动化。
14	**不锁定**	使用商品化的通用标准硬件，尽量不绑定到特殊的硬件厂商或硬件产品
15	**慢半拍**	只用已经证明确实好用的，成熟的技术
16	**成本最优**	如果是不擅长的、非核心的、不会带来竞争优势的组件的则考虑直接购买选用成熟产品

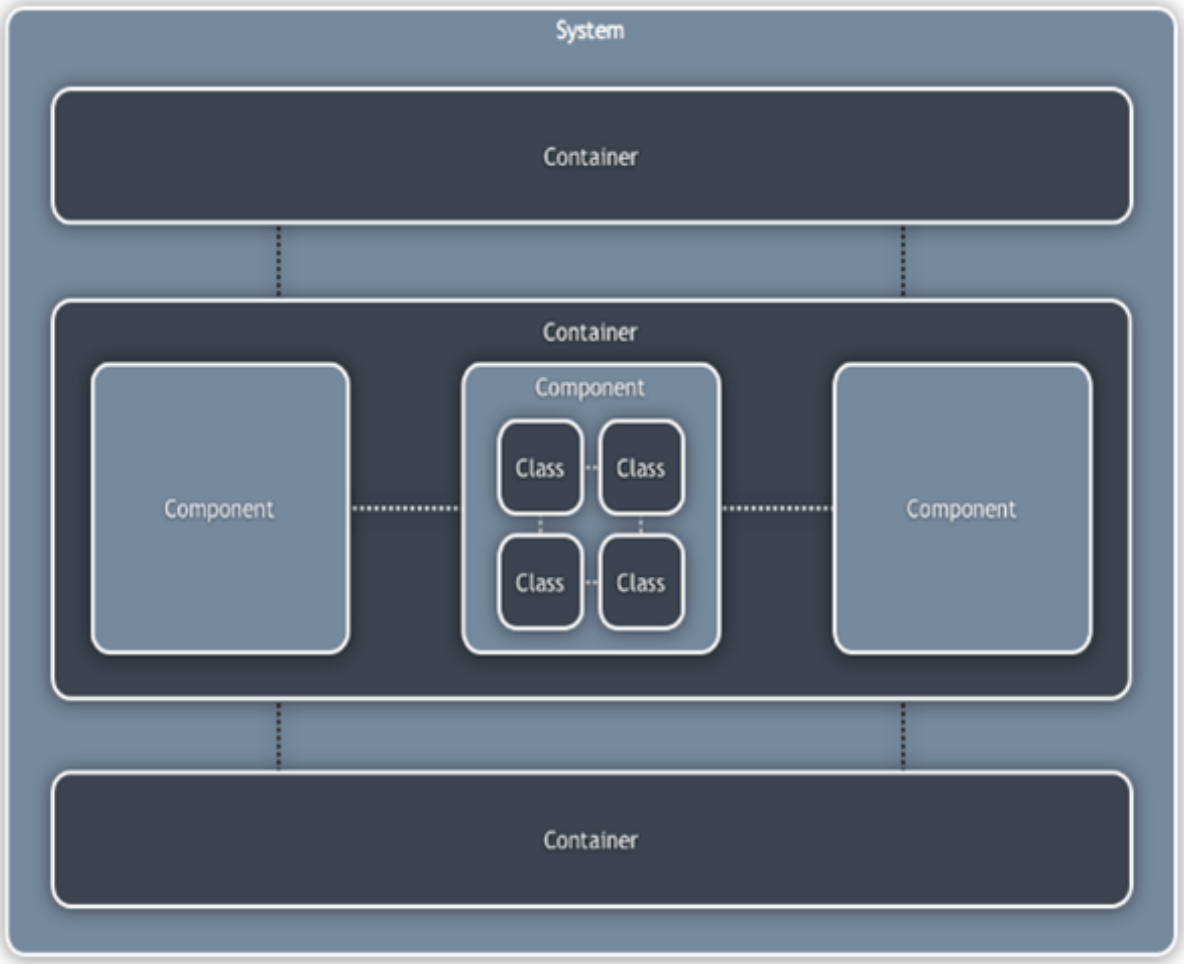
3、设计方法

整体架构设计方法遵循C4架构设计方法。

C4指的是：

系统的架构 = 上下文 (Context) + 类 (Class) + 组件 (Component) + 容器 (Container)

- **Context**
  - 总体蓝图设计、系统的职责、环境上下文，外部关系
- **Class**
  - 领域模型设计：说明系统的核心务逻辑、核心概念类、模块划分设计
- **Component**
  - 系统的逻辑实现设计，包括重要组件、数据结构、程序结构
- **Container**
  - 系统物理实现设计，部署结构以及技术选型



基本的架构设计流程可分为如下六个步骤：

1、 **深入理解需求**，即需要设计一个什么样的系统，该系统在生产环境与其他系统和外部实体是如何交互的？功能需求和非功能需求到底是什么？架构师应主动参与到需求分析和需求评审的过程。

2、 **概念设计(Context,Class)**，对业务进行分析，梳理出系统的业务领域（模块、子系统）、重要的概念（对象、数据实体）、以及概念之间的关系。并形成领域模型，具体可参考《领域驱动设计》；

3、 **逻辑设计(Class,Component)**，根据概念设计的领域模型，设计出系统的逻辑模型（组件或服务、以及组件内部的程序包、对象、数据库结构等逻辑对象），并按照组件概要说明基本结构。包括

- 类设计
- 如涉及数据库，要进行物理数据模型设计，推荐使用PowerDesigner进行设计。
- 接口设计
- 技术分层设计：进行分层次的技术架构设计，例如展示层、业务逻辑层、DAO层等。

4、 **部署设计(Component,Container)**，包括

- 部署设计，设计组件如何部署到运行环境。将运行环境抽象为一个个容器（节点），说明逻辑组件或服务如何部署到容器中，以及容器之间的通讯关系。
- 环境设计，详细设计系统所使用的sit,uat,prod等环境，以及对应的自动化部署方案；
- 安全架构设计，详细设计系统的信息安全架构；
- 非功能需求的方案设计：针对可靠性、安全性、高可用性、以及运维监控需求等进行方案论证与设计。

5、技术选型

- 技术选型，根据系统的逻辑设计和部署设计选择合适的操作系统、数据库、第三方软件、SDK,开源程序包等；

6、架构评审

- 进行技术架构评审，基本过程参见本文第5章节。

4、设计流程

在一个敏捷的团队里，人人都是架构师，软件架构设计应该由研发团队整体负责，

同时为了协调架构设计相关的事情，会由SDM或一名资深的SDE主导架构设计，主导者需要承担如下职责：

- 深入理解需求和业务目标，架构师应该在深入理解业务需求和目标，并且能够更抽象一个层次，把握会导致需求变动的各种因素。
- 设计软件
- 控制技术风险
- 管理架构演化，软件的架构是动态的，会不断演化。
- 与其他成员一起编写代码（**注意软件研发团队中没有不写代码的架构师**）
- 保证质量

在一个研发项目过程中，基本的架构设计流程如下：

阶段	设计工作	说明
立项	技术可行性研究	协作产品经理完成产品的技术可行性研究
启动	初步设计	完成初步的架构技术选型以及关键技术的研究，控制技术风险 主要交付物为 2.2-模板-初步设计方案.pptx
定义	架构设计	完成系统的整体性架构设计，并完成架构评审 主要交付物为 2.3-模板-系统架构设计说明书模板.docx
实施	详细设计以及架构设计的变更	根据架构设计完成详细设计，并在迭代过程中对架构设计进行补充和优化 必要时可以修改架构设计 主要交付物为 2.4-模板-系统部署方案模板.docx 2.4-模板-系统接口规格说明书模板.docx
验收	设计交付文档的归档	完成架构设计相关文档的验收归档

5、架构评审

5.1 评审流程

基本评审流程如下：

1、产品研发团队完成架构设计后进行自查，自查时基于架构设计检查清单（参见 2.2-模板-项目质量保证计划.xlsx）进行内部自查；

2、向PMO提交架构评审申请，并在提交评审申请时提交自查结果，以及相关的设计交付文档；研发团队需提前提交申请，并至少为评审人员预留1-3个工作日（取决于架构和业务的复杂度）用于了解和评估架构设计的内容；

3、PMO收到评审申请后，会对提交的交付物进行初步评估，如果质量不能满足要求会直接驳回评审申请。如果初步评估通过，则会协调技术委员会，并与研发团队约定评审会议时间和会议形式。

4、评审人员主持评审会议，进行评审后给出评审意见。

5.2、管控原则

基本管控原则如下：

1、架构评审是分配研发环境和资源的前提，没有通过架构评审将不能申请研发资源；

2、在验证测试和部署发布的环节也会进行检查，没有通过架构评审的原则上不允许进入验证测试环节，更不能在生产环境发布；

3、对每次评审，**每个项目有两次提交申请的机会，如果两次评审都不通过，项目风险等级直接调整为最高级，项目组应停止其他工作进行紧急处理。**