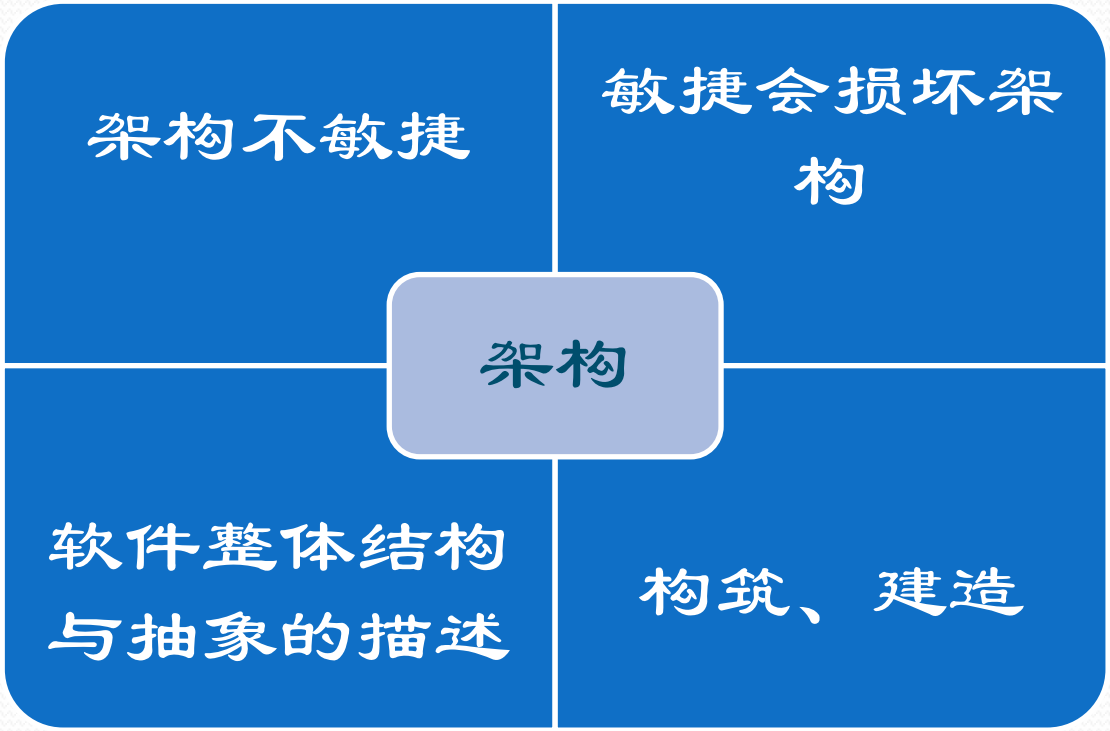
敏捷架构设计

汪亮亮

主要内容

- 架构和软件架构
- 敏捷架构设计时机
- 软件架构技术债务
- 敏捷架构师

什么是架构？



架构定义

架构

- 难以扭转的决策

软件架构

- 软件的高级形式和流程，与问题领域无关但与预期的用户体验有关

架构是宽泛的

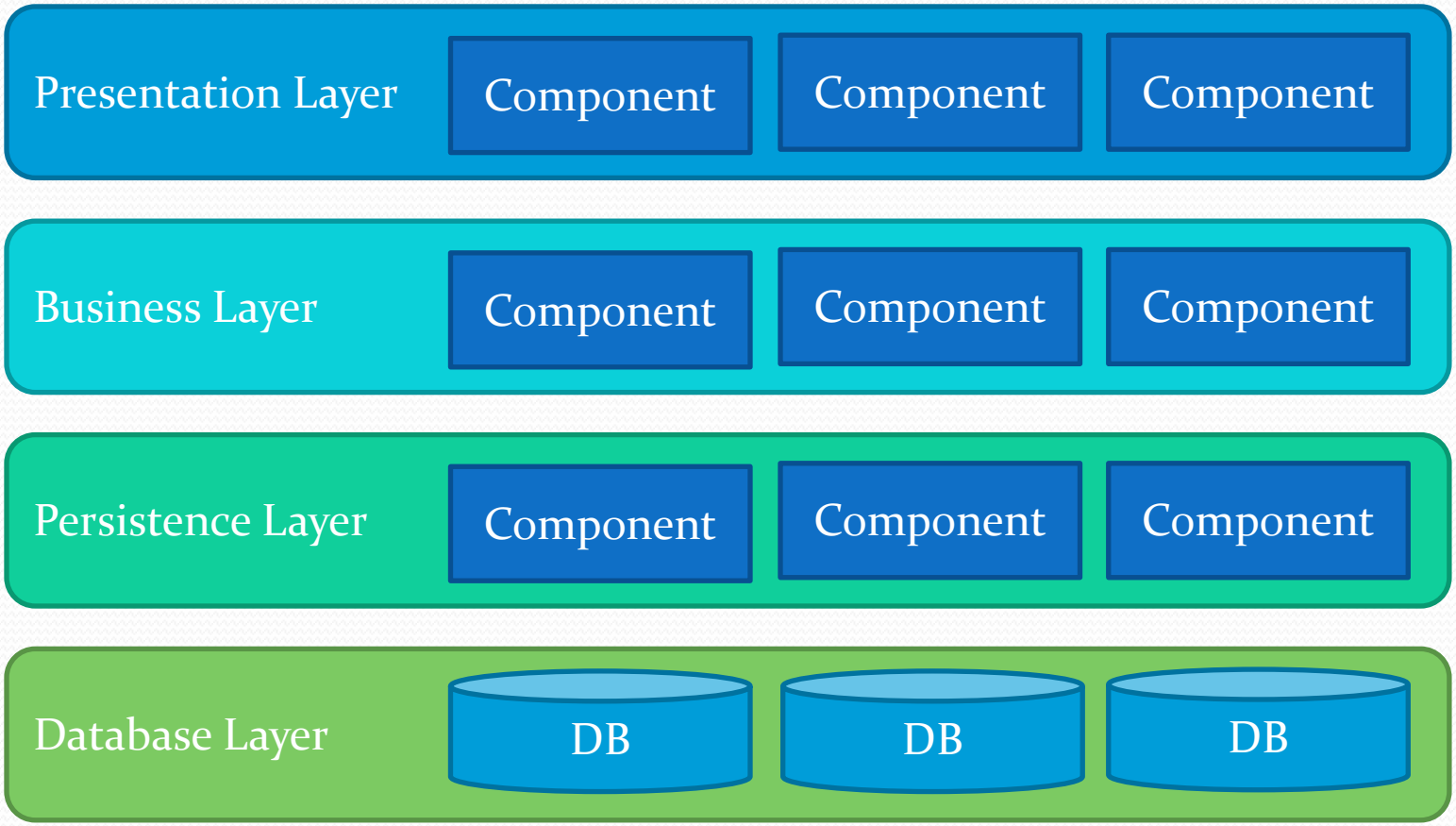
- 架构反映一
  里的关键性
- 如果您没有
  决策的

例子：编程语言是一种非常重要的决策。试想在项目的第一天，您选择了Java，3个月后，您又觉得Go可能是更好的语言选择。您已经花了3个月时间编写Java，不可能快速扭转这一决策。

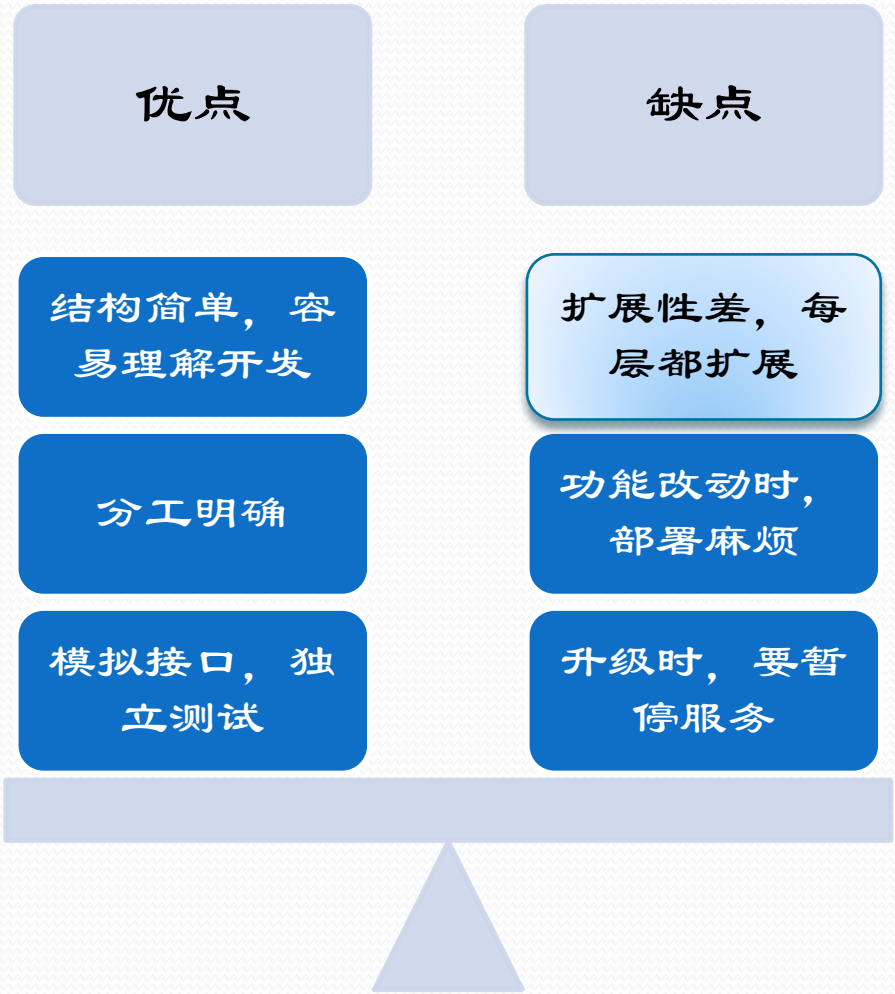
软件架构是具体的

- 分层架构
- 事件驱动架构
- 微核架构
- 微服务架构
- 云架构

分层架构

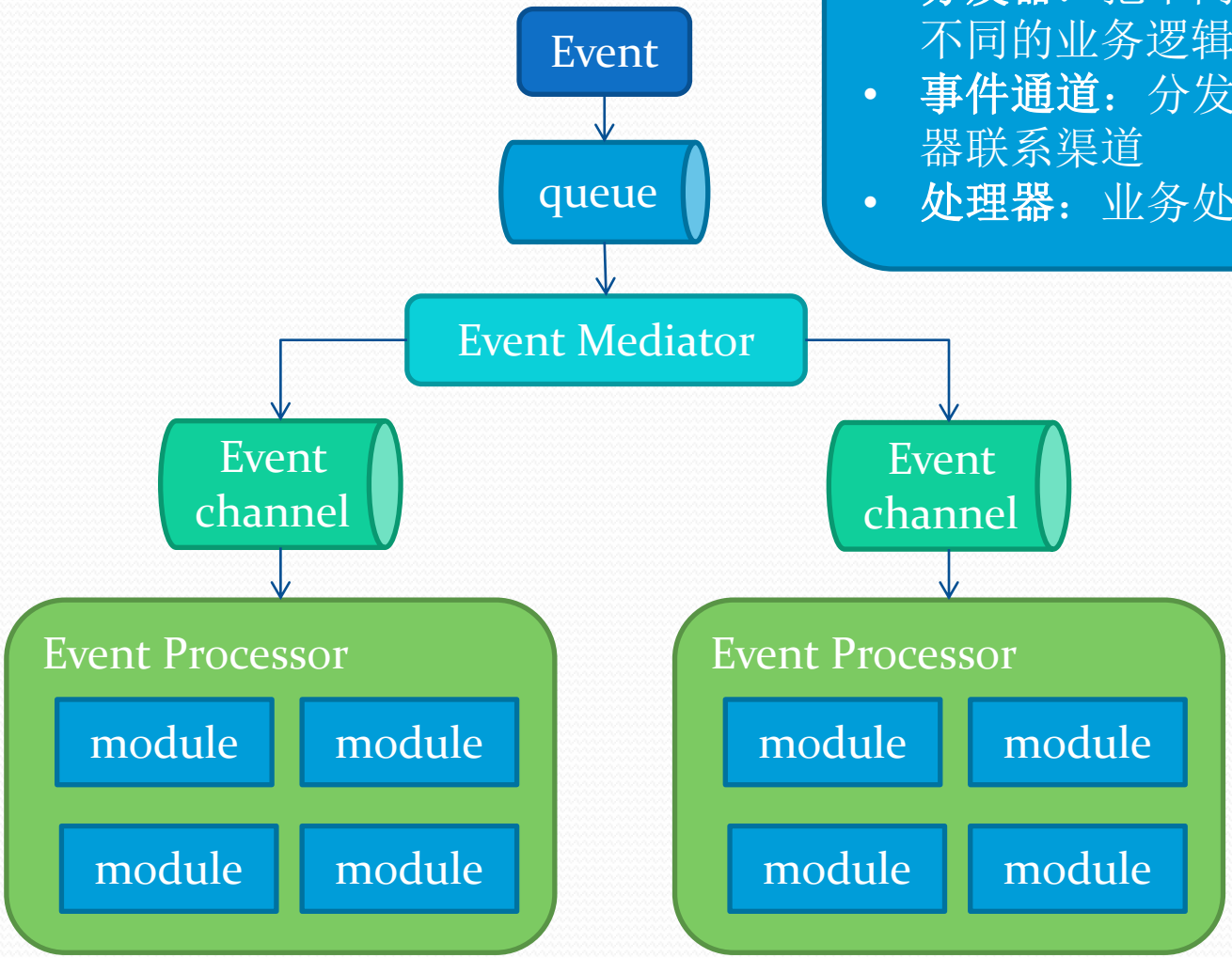

Presentation Layer
Component
Component
Component
Business Layer
Component
Component
Component
Persistence Layer
Component
Component
Component
Database Layer
DB
DB
DB

分层架构

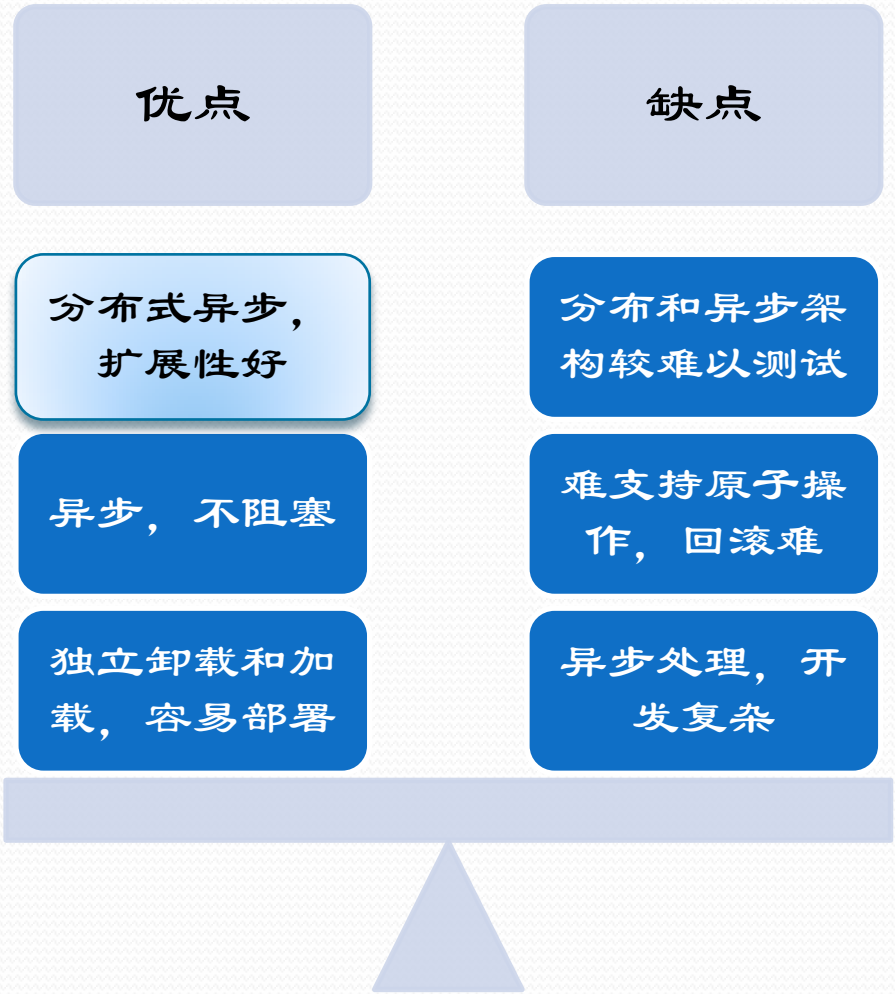

优点
缺点
结构简单，容易理解开发
扩展性差，每层都扩展
分工明确
功能改动时，部署麻烦
模拟接口，独立测试
升级时，要暂停服务

事件驱动架构

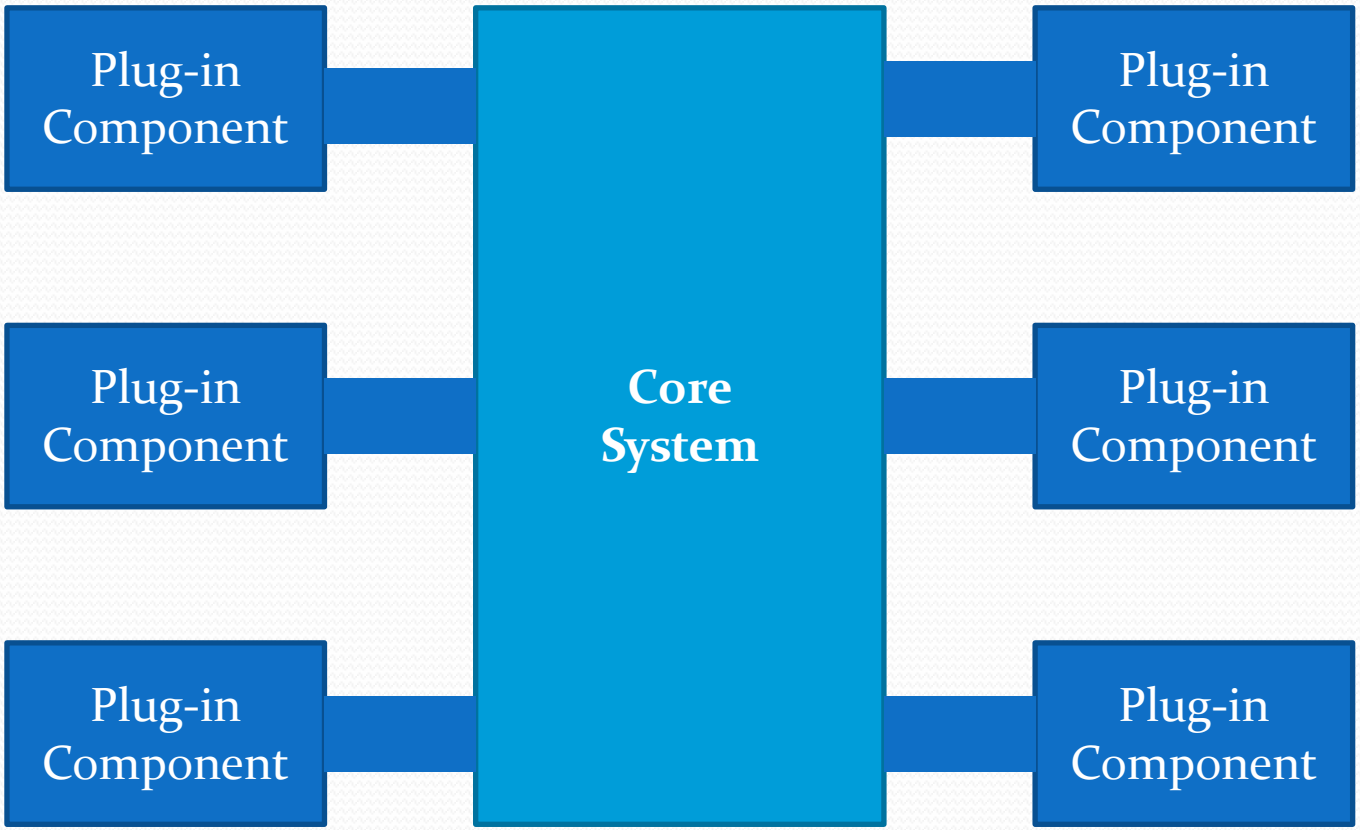
- **事件队列**：接受事件入口
- **分发器**：把不同事件分发到不同的业务逻辑单元
- **事件通道**：分发器和处理器联系渠道
- **处理器**：业务处理逻辑单元



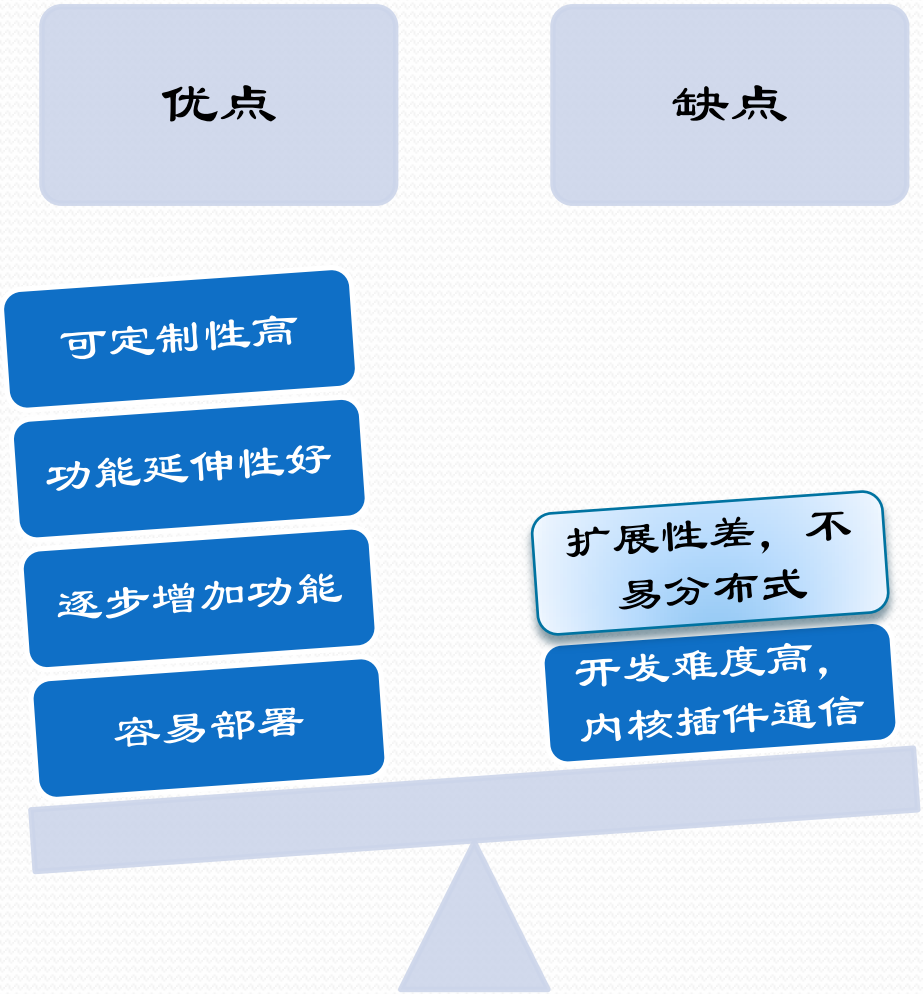
事件驱动架构


优点
缺点
分布式异步，扩展性好
分布和异步架构较难以测试
异步，不阻塞
难支持原子操作，回滚难
独立卸载和加载，容易部署
异步处理，开发复杂

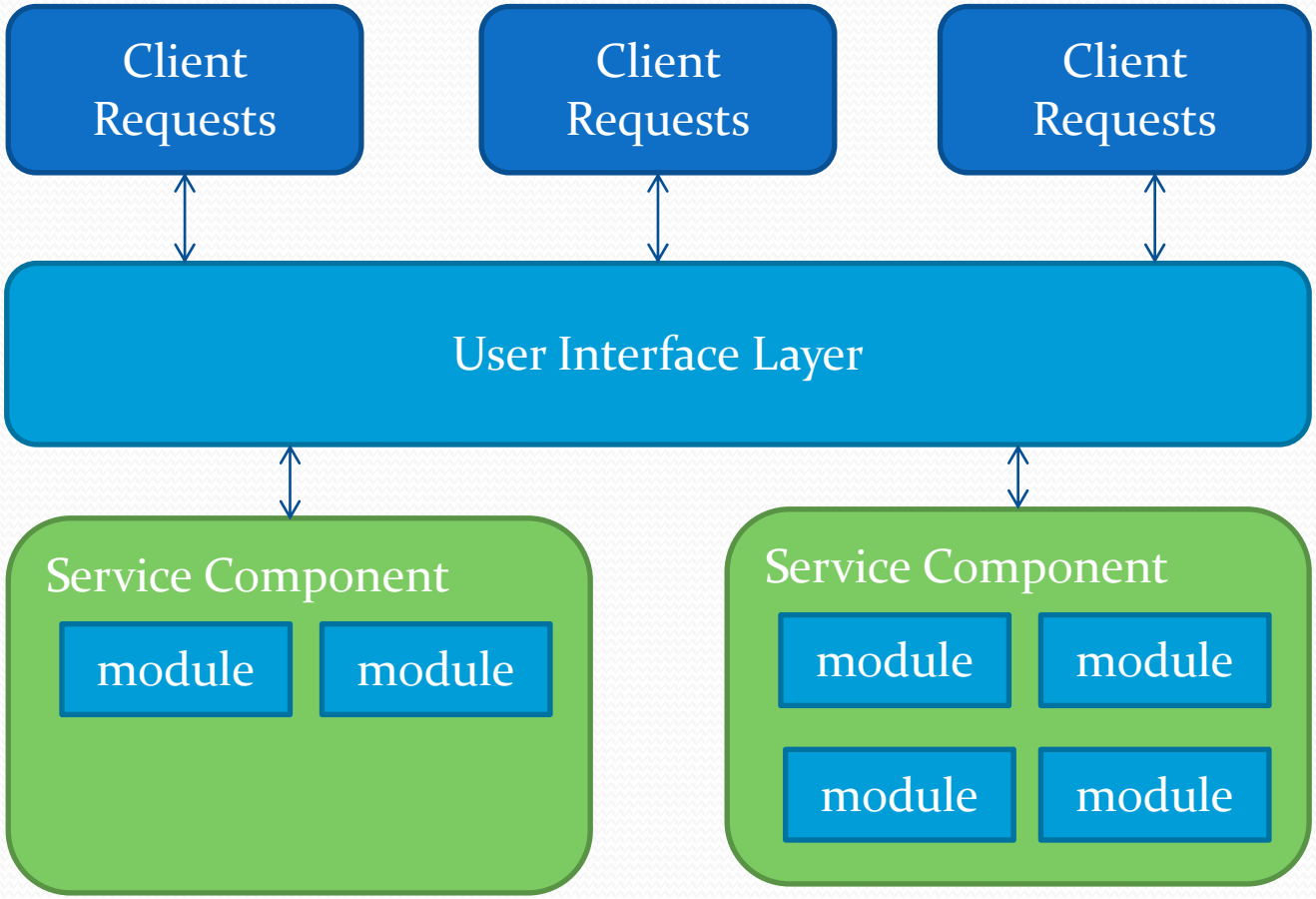
微核架构


Plug-in Component
Plug-in Component
Plug-in Component
Core System
Plug-in Component
Plug-in Component
Plug-in Component

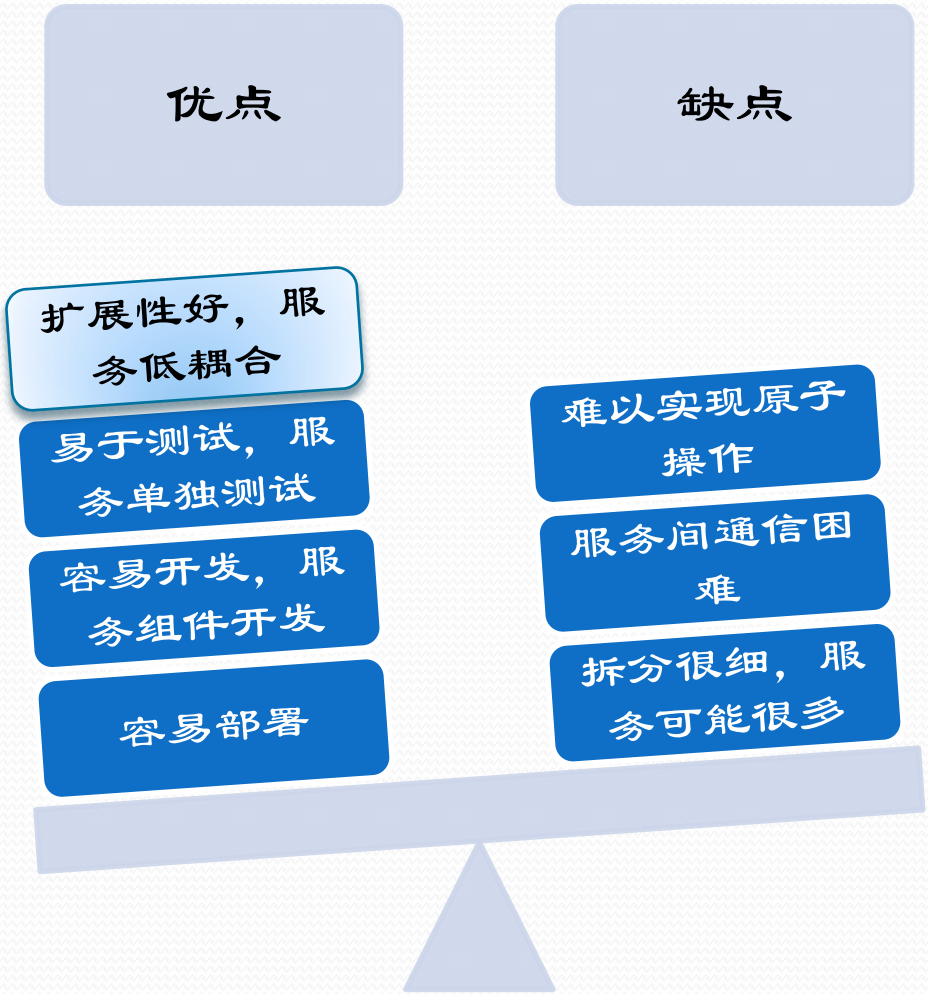
微核架构



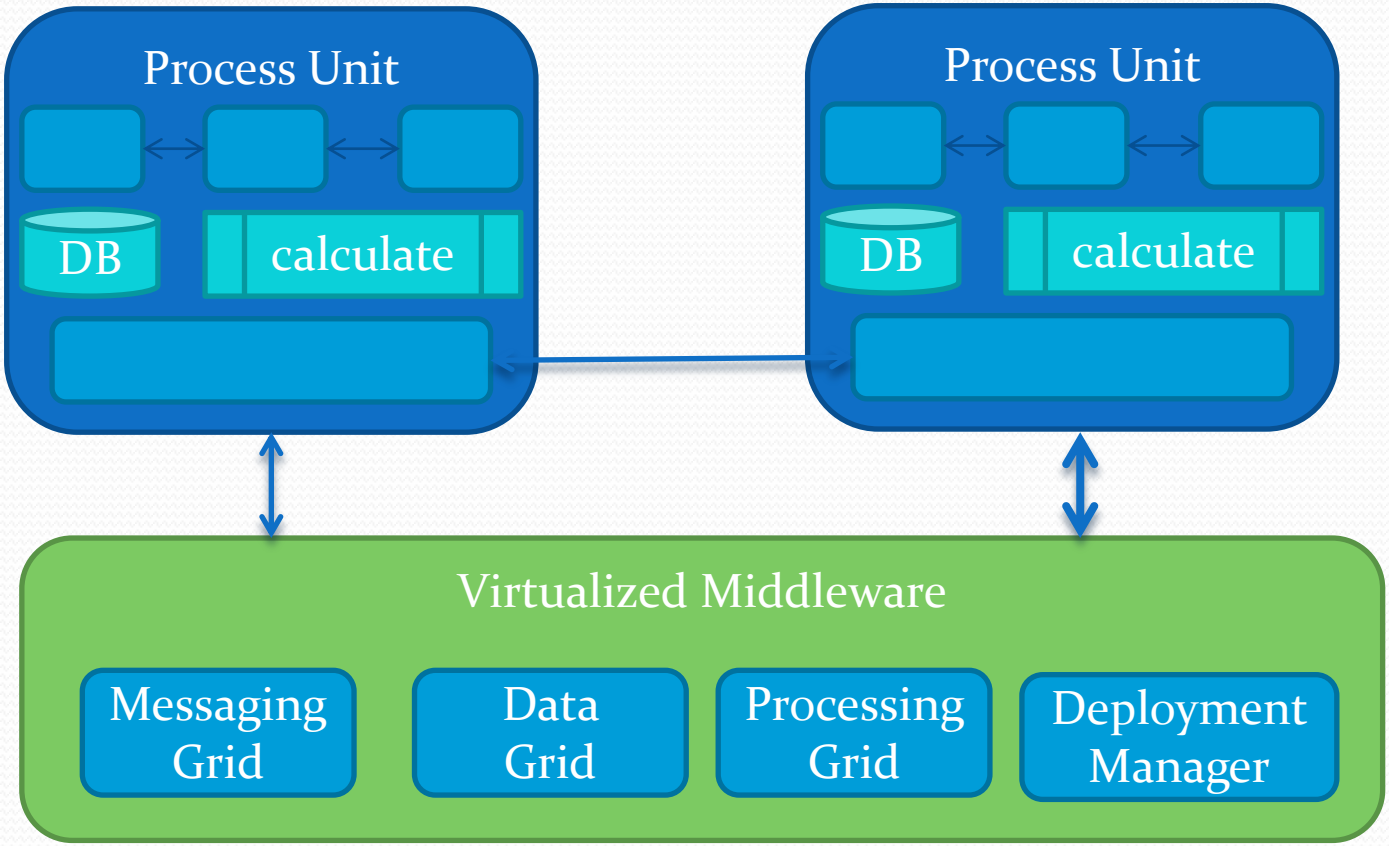
微服务架构


Client Requests
Client Requests
Client Requests
User Interface Layer
Service Component
module
module
Service Component
module
module
module
module

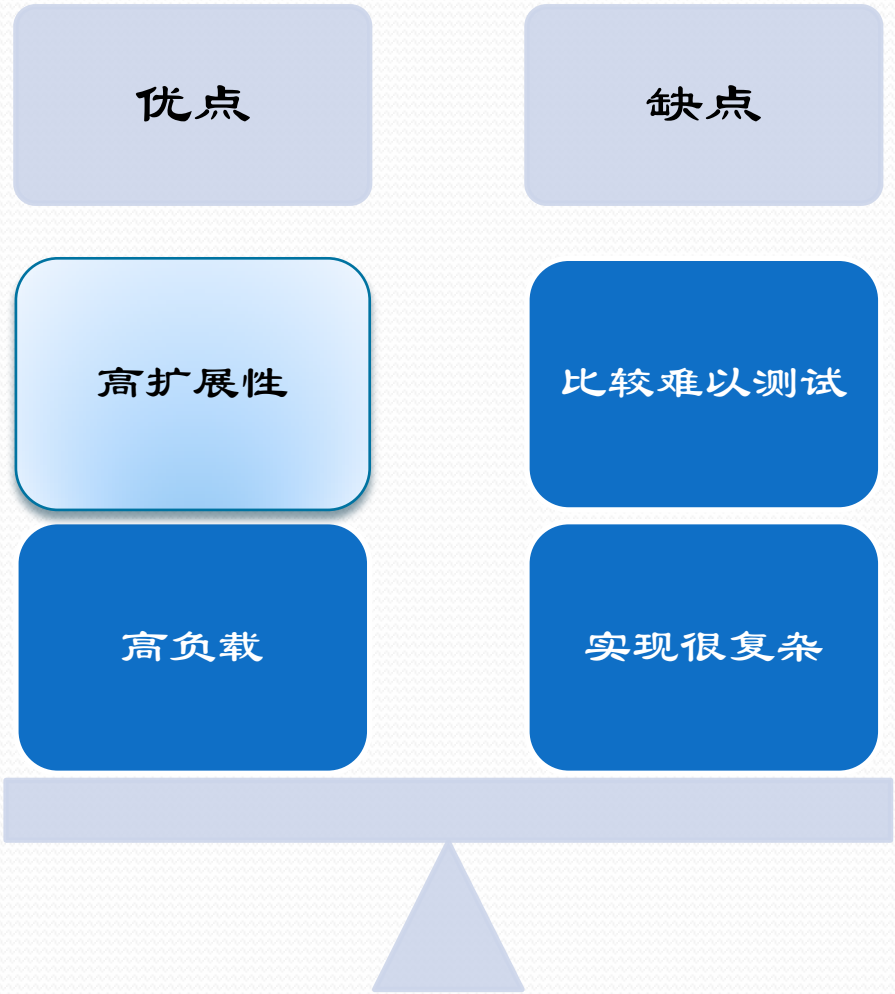
微服务架构


优点
缺点
扩展性好，服务低耦合
易于测试，服务单独测试
容易开发，服务组件开发
容易部署
难以实现原子操作
服务间通信困难
拆分很细，服务可能很多

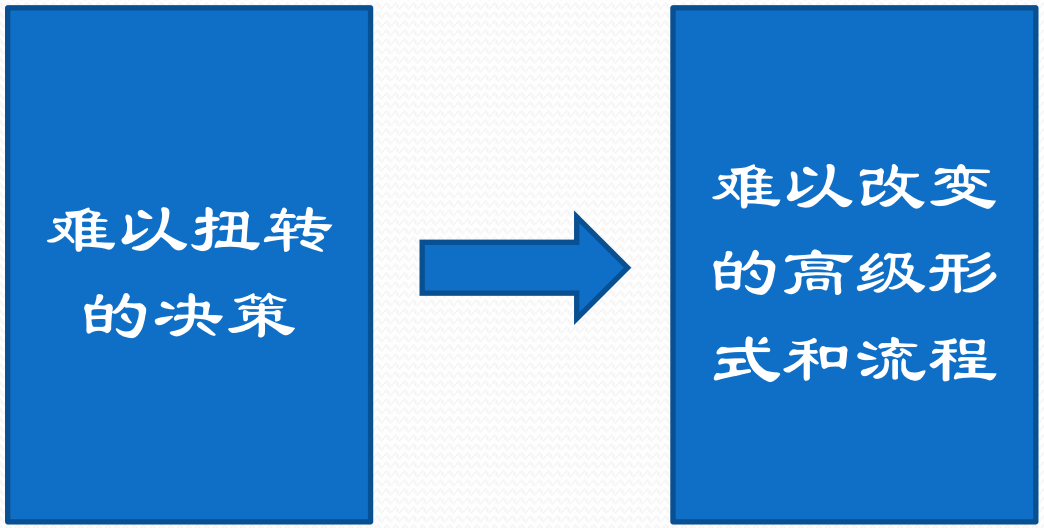
云架构


Process Unit
DB
calculate
Process Unit
DB
calculate
Virtualized Middleware
Messaging Grid
Data Grid
Processing Grid
Deployment Manager

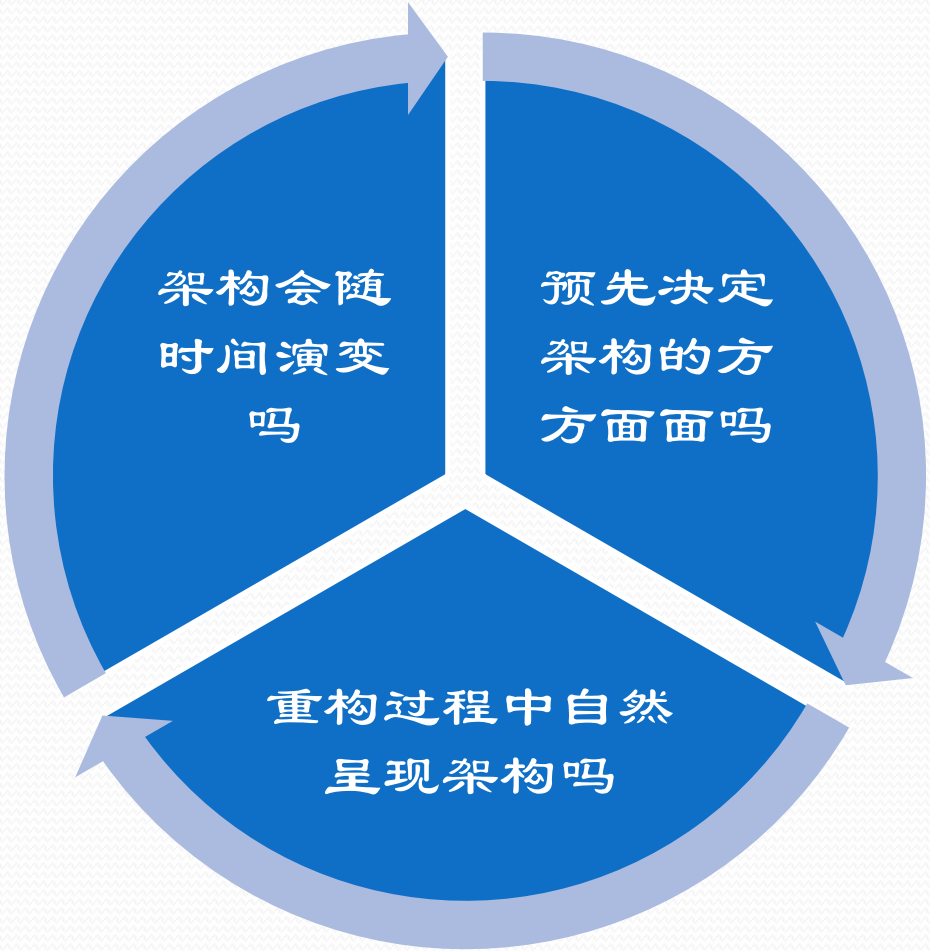
云架构



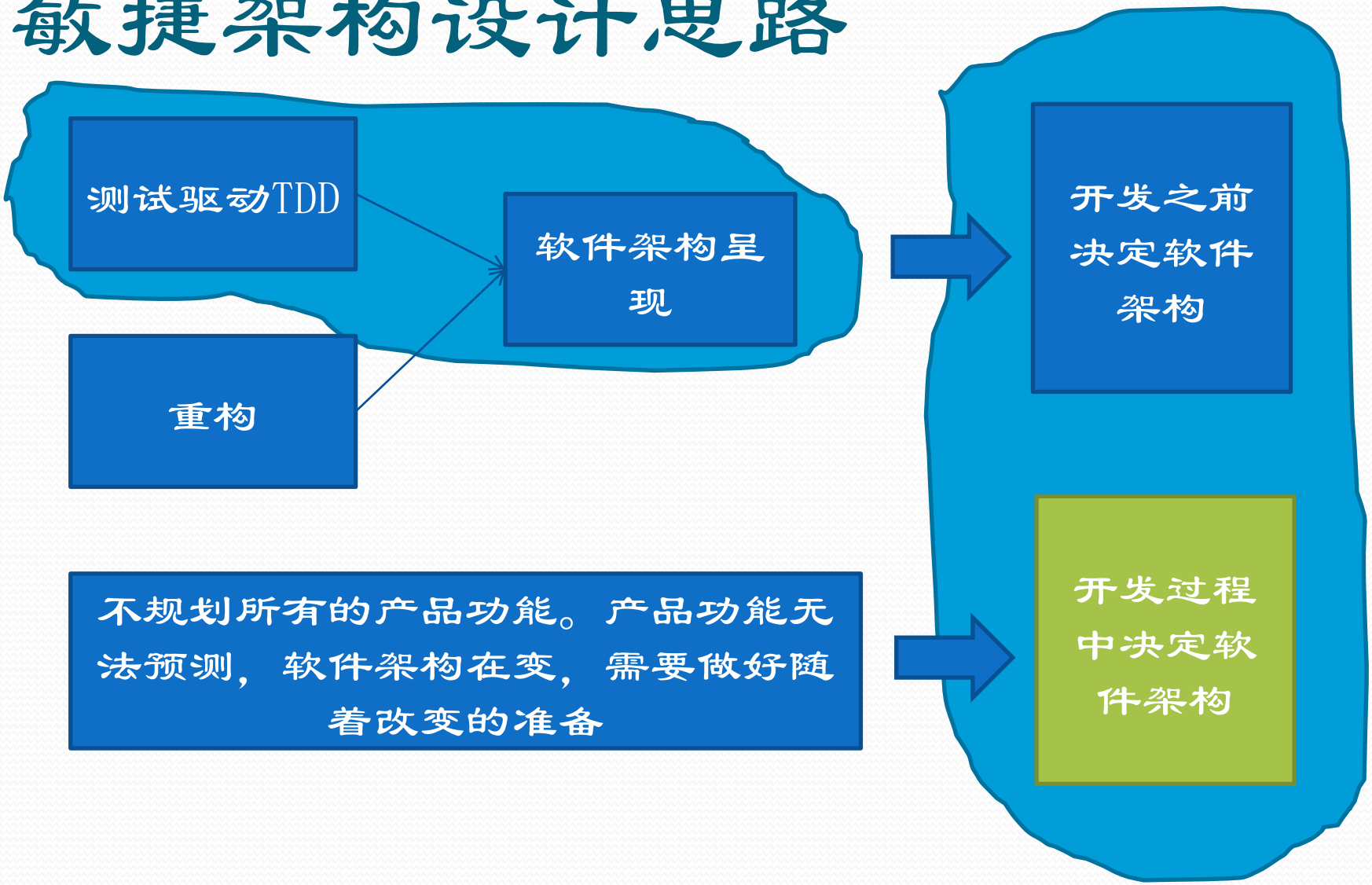
软件架构也是架构



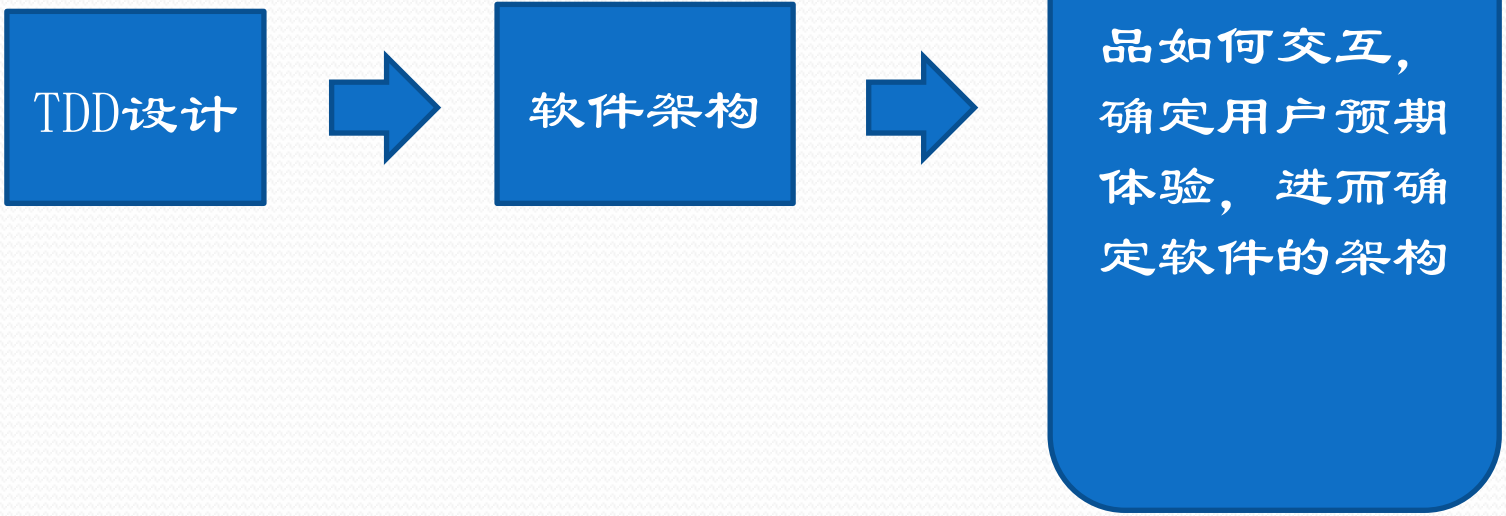
敏捷架构何时开始



敏捷架构设计思路


测试驱动TDD
重构
软件架构呈现
开发之前决定软件架构
开发过程中决定软件架构
不规划所有的产品功能。产品功能无法预测，软件架构在变，需要做好随着改变的准备

开发前构建架构



石头剪刀布游戏-初版需求

故事：

Feature: Play

Scenario: Rock v . Scissors

- Given player one throws rock
- And player two throws scissors
- Then player one winds

并未提供充足的有关用户预期体验的信息。

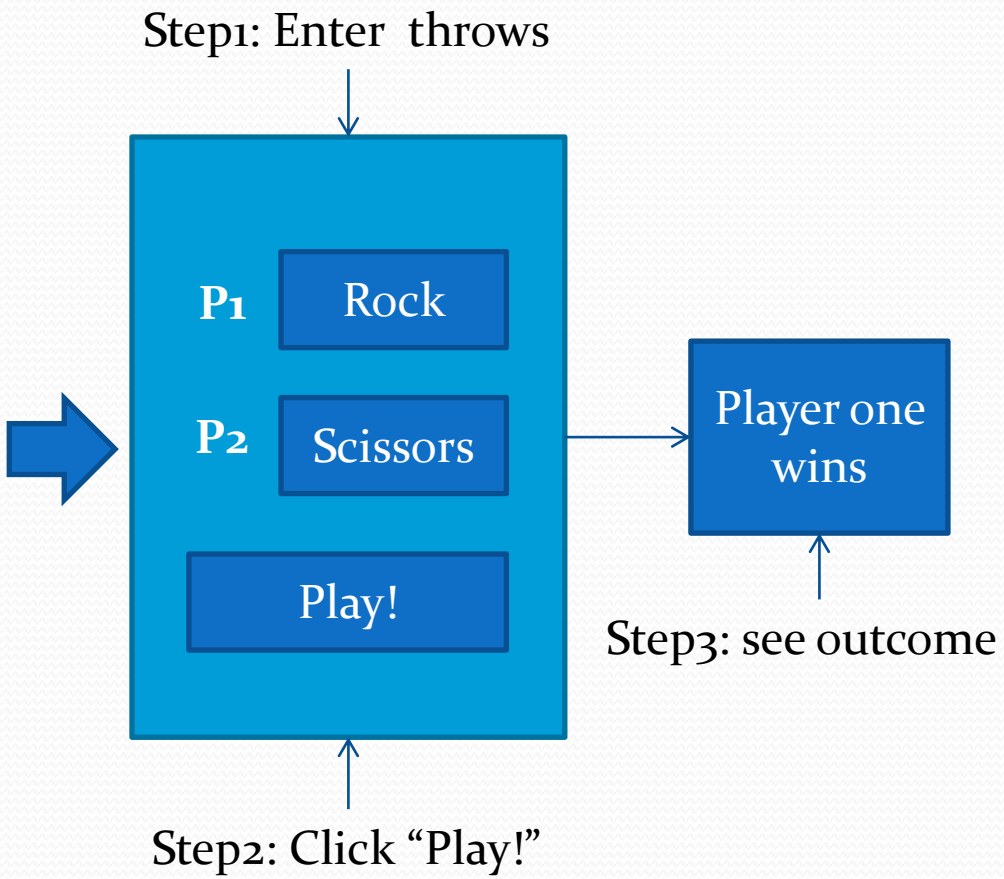
- 玩家一和玩家二出石头和剪刀，动作输入后查看赢家？
- 是使用应用玩游戏？

石头剪刀布游戏-升级需求

Feature: Play

Scenario: Rock v . Scissors

- Given player one throws rock in real life
- And player two throws scissors in real life
- When one of them enters the throws into the application
- Then the application tells them that the player one wins

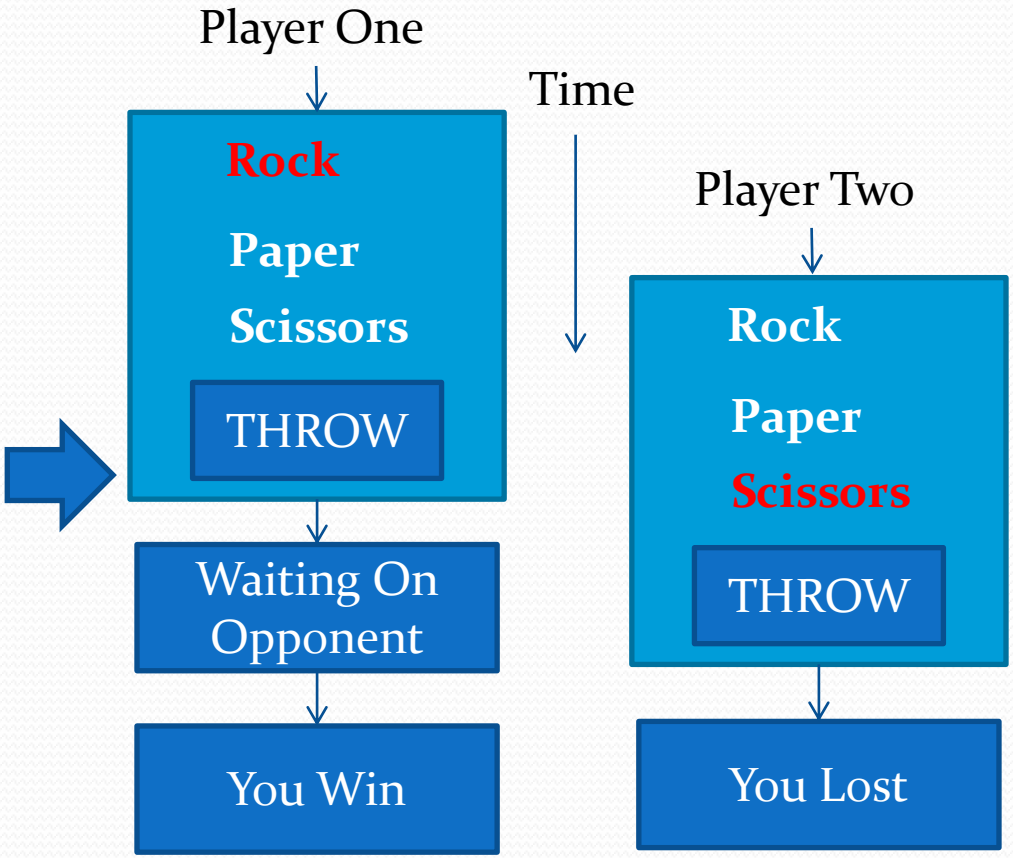


石头剪刀布游戏-变更需求

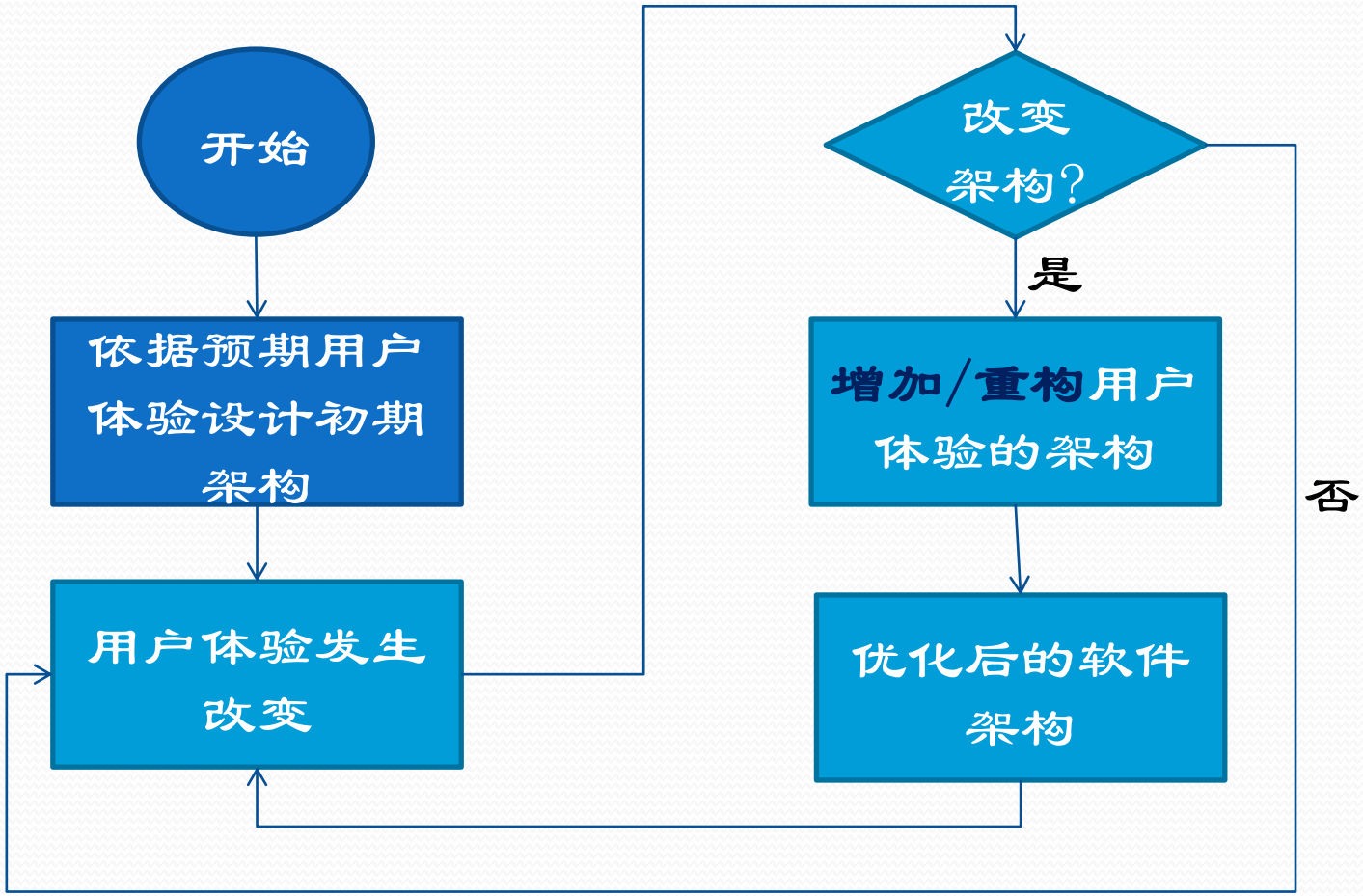
Feature: Play

Scenario: Rock v . Scissors

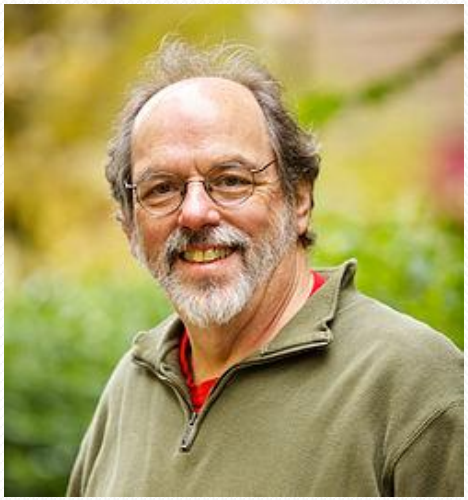
- Given player one submits a “rock” throw
- When player two submits a “scissors” throw
- Then player one should see that he or she won
- And player two should see that he or she lost



开发中构建架构



架构债务



Ward CunningHam 沃德●坎宁安
计算机程序员和Wiki概念的发明者。设计模式和敏捷软件方法的先驱之一。

开发团队在设计或架构选型时，从短期效应的角度选择易于实现的方案，但从长远看，这种方案会带来更消极的影响，亦即开发团队所欠的债务。

架构债务定义

Martin Fowler 马丁●福勒

“敏捷软件开发宣言”创作者之一，ThoughtWorks公司的首席科学家。

架构债务类似于金融债务，会产生利息。利息指由于鲁莽的设计决策导致未来的开发中付出更多努力的后果。可以继续支付利息，也可以重构设计将本金一次付清。虽然一次性付清本金需要代价，却可以降低未来的利息。

技术债务四

技术开发总是精益求精。唯一不足是开发中又发现更优的技术方案，有技术[illegible]比如Google[illegible]

没有人知道好标准是什么，做事情由自己性子来，无组织、流程、标准。

[illegible]技术有追求，迫于交[illegible]力不得不用临时拙[illegible]的方案代替。国内很多大公司都这样。

拙劣团队

团队中有人清楚好的架构，但是管理混乱，没时间设计，敷衍了事。

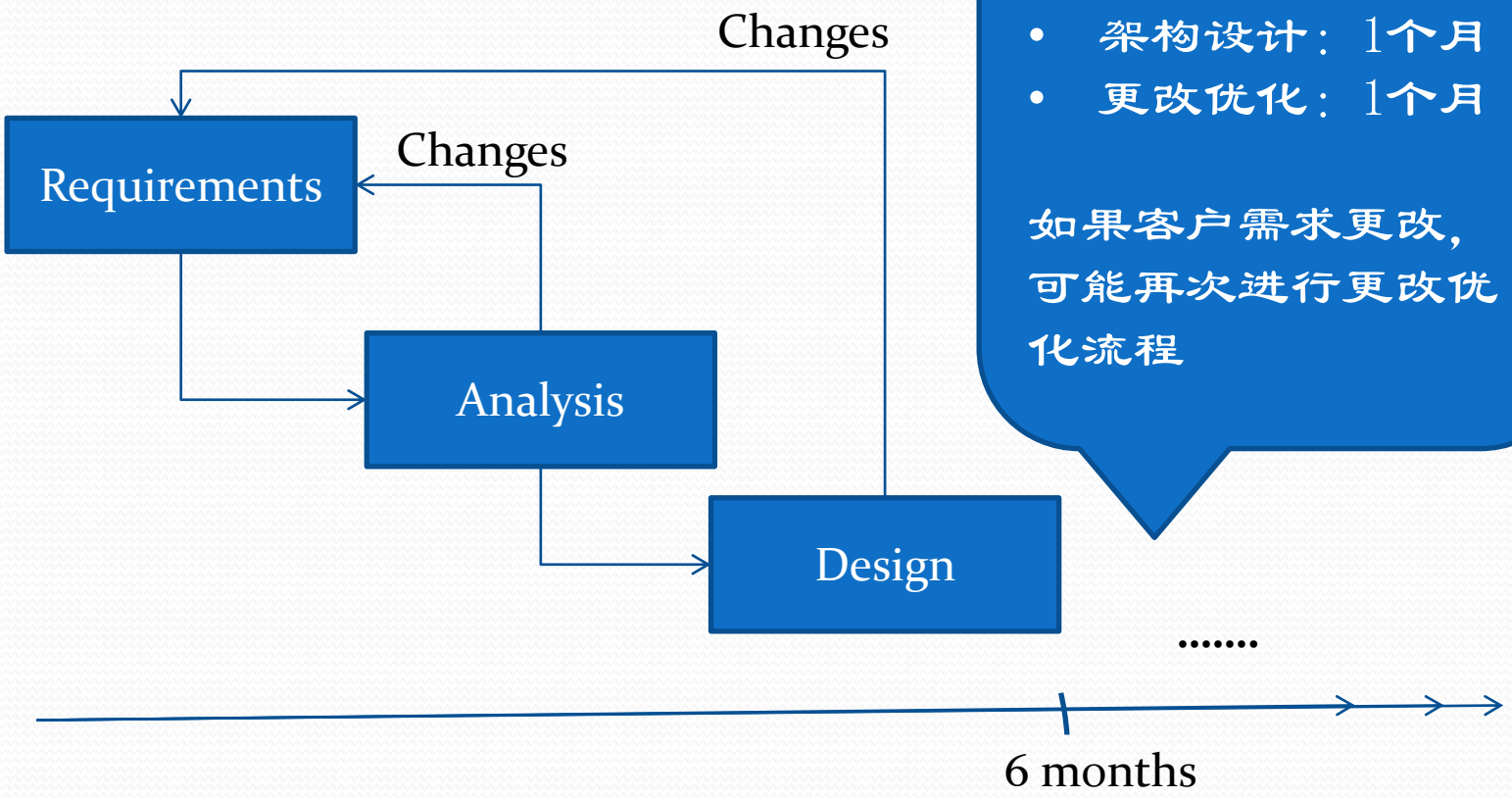
混乱团队

解决技术债务

无论哪种团队，都会产生技术债务，
不要心存侥幸心理！

系统重构

瀑布式开发



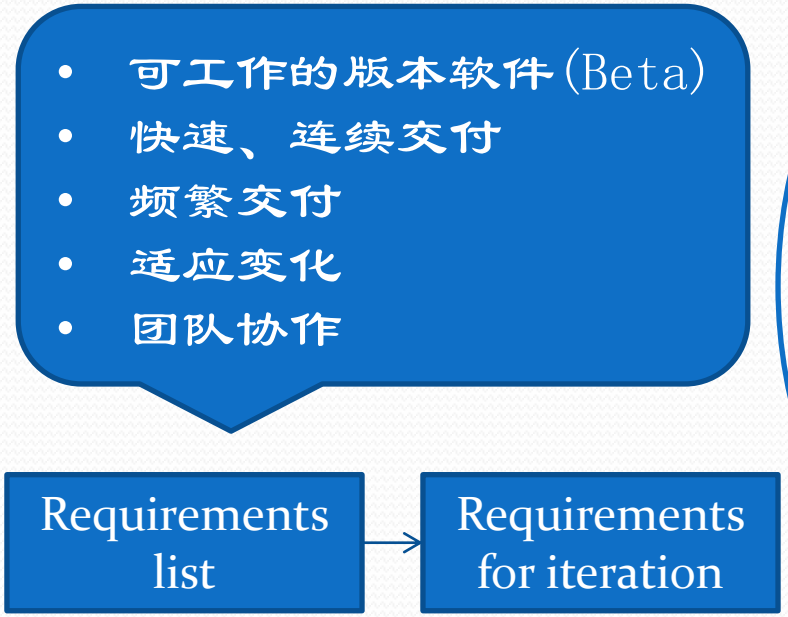
6人，2年的小项目

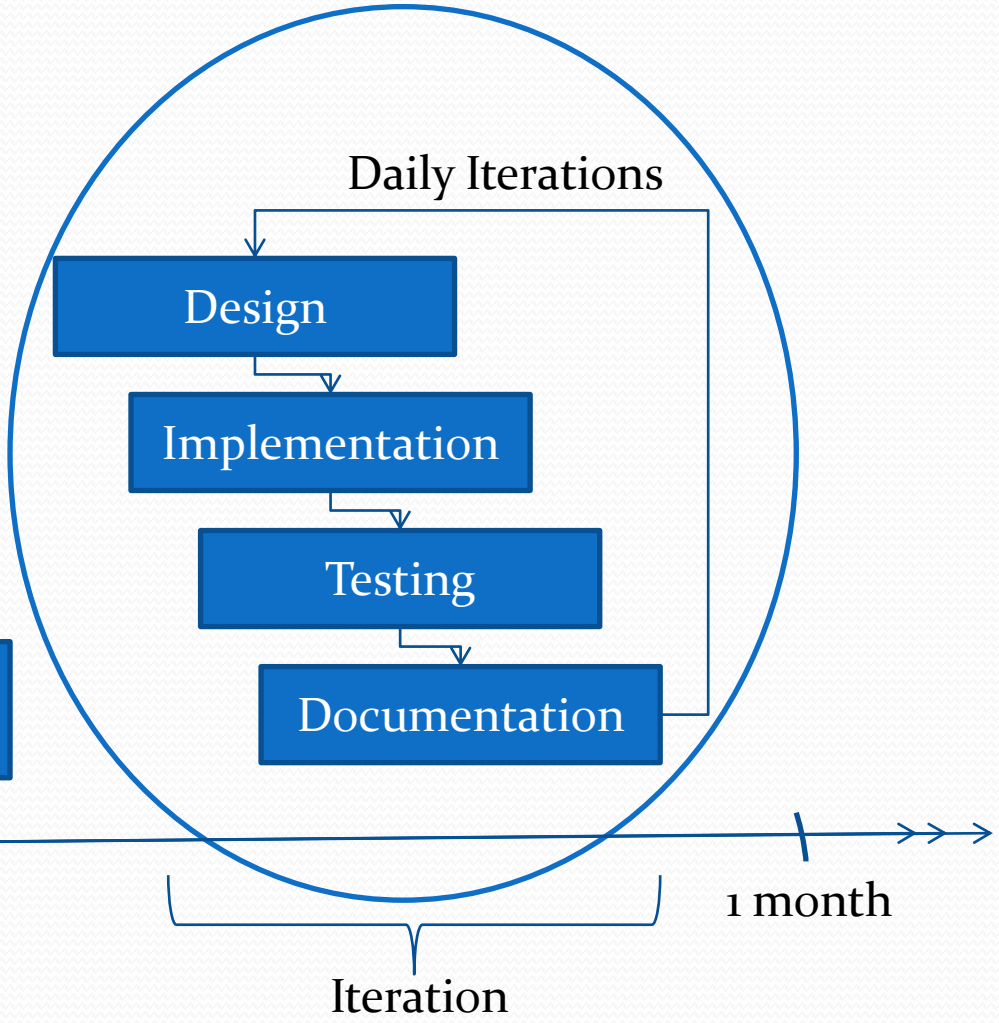
- 需求记录：3个月
- 分析：1个月
- 架构设计：1个月
- 更改优化：1个月

如果客户需求更改，可能再次进行更改优化流程

敏捷开发

- 可工作的版本软件(Beta)
- 快速、连续交付
- 频繁交付
- 适应变化
- 团队协作

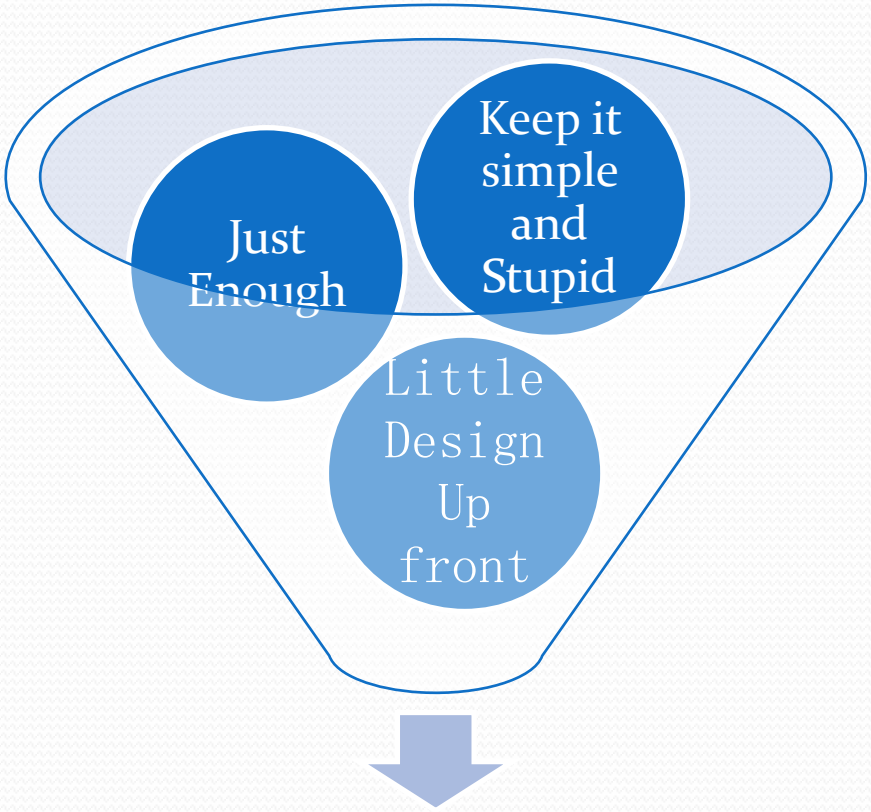




传统&敏捷架构师

关注大格局	创建原型 明智决策
遵从性导向	关注可持续性
绘制蓝图	愿景和当前保持平衡
不参与实战	参与团队开发

敏捷架构设计原则



演进架构调整、重构、
TDD、持续集成

谢谢