团队技术进阶之路
—以DDD为切入点

黄晓辉 | 2017.12

你能了解并收获什么？

- 将DDD引入团队的正确时机和实施路径
- 一次组织变革的实战经验
- DDD实践中的那些坑
- DDD方法论的总结

团队概况

- 20+ 开发人员，4个开发小组
- 面向客户的后端业务系统开发
- Java 技术栈，Restful API
- 2位同事具有DDD的实践经验

问题&背景

- 团队没有形成对业务模型的一致理解
- 业务逻辑bug率居高不下
- 代码可读性不高，新人上手困难
- 遭遇技术瓶颈，需要新的方法论

解决思路

- “程序的可读性、可扩展性、正确性”是**目标**
- “团队开发能力的提升”是**关键**
- “一套好的架构模式和方法论”是**装备**
- “规划合理的可操作的实施路径”是成功的**保障**

为什么选择DDD作为切入点？

- 相对成熟的方法论体系，实践案例和技术交流社区
- 团队中有有经验的同事，可以“老带新”
- 适合业务系统开发团队：专注于业务逻辑的正确性

如何开始？

— 领域模型的大讨论

形式：
召集项目骨干，组织多次讨论会，收集想法，对现有业务进行建模

发现的问题：
1.过于强调DDD的理论意义，就Value Object，Entity，aggregate等概念产生很多分歧；
2.过于强调领域模型的命名，而忽视了对内涵的理解

取得的进展：
1.激发了团队的兴趣，大家开始审视工作中遇到的各类业务问题
2.加深了对现有业务的理解

事后回顾&总结
建立全局共识很重要，即：领域建模不应该放在第一步，领域划分才是首要任务。完成领域划分后，再在每个子领域中建立模型和统一语言，这样可以避免很多不必要的分歧。

POC

— 将讨论的结果付诸实践

形式：

2名有DDD经验的开发+1名有技术潜力的开发 参与，对一个小业务模块尝试重构

发现的问题：

1.对Vo，Entity，Service，Repository，Factory的功能定位不清晰

2.尝试使用**六边形架构**，但选择的系统粒度过小，导致效果不佳

取得的进展：

1.加深了实现层面的理解

2.制定了编码规范

3.引入了基于Spock的BDD实践

事后回顾&总结

1. 理论和实践是有差距的

2. 人的认知能力的提高需要一个过程，只有亲身实践才能加深认知

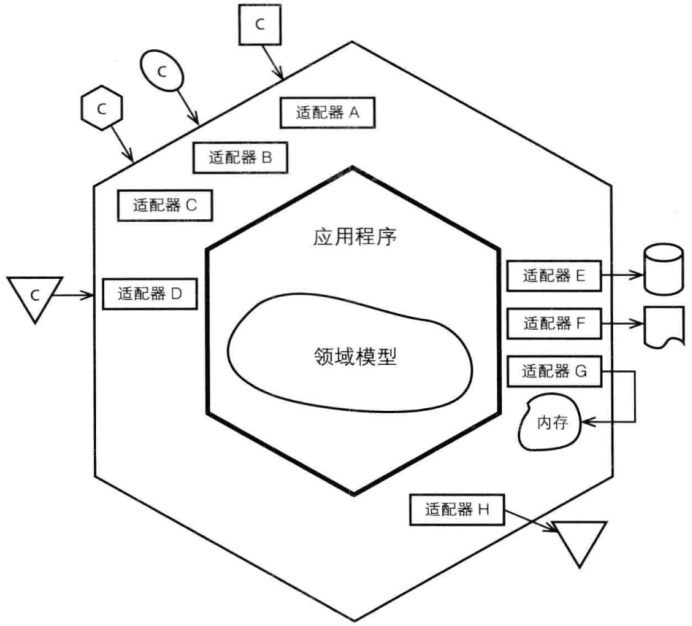


图4.4六边形架构也称为端口与适配器。对于每种外界类型，都有一个适配器与之相对应。外界通过应用层API与内部进行交互。

融入日常开发

— 将实践扩大到业务开发团队

形式：
在整个开发团队中进行推广学习，并在一个业务开发团队中开始使用

发现的问题：
1.新业务开发和DDD实践并行，导致项目进度放缓
2.对经验不足的成员来说，还是比较难于上手

取得的进展：
1.整个团队开始学习和实践

事后回顾&总结
1. 很多实现层面的问题并没有彻底解决（例如：Repository的定位，Aggreate的边界），导致在实践过程中还需要花费很多时间讨论，从而导致一些新手认识上的混乱
2. 编码规范的保证需要能够自动化，节省人工审查的成本

反思

停一停，看一看，想一想

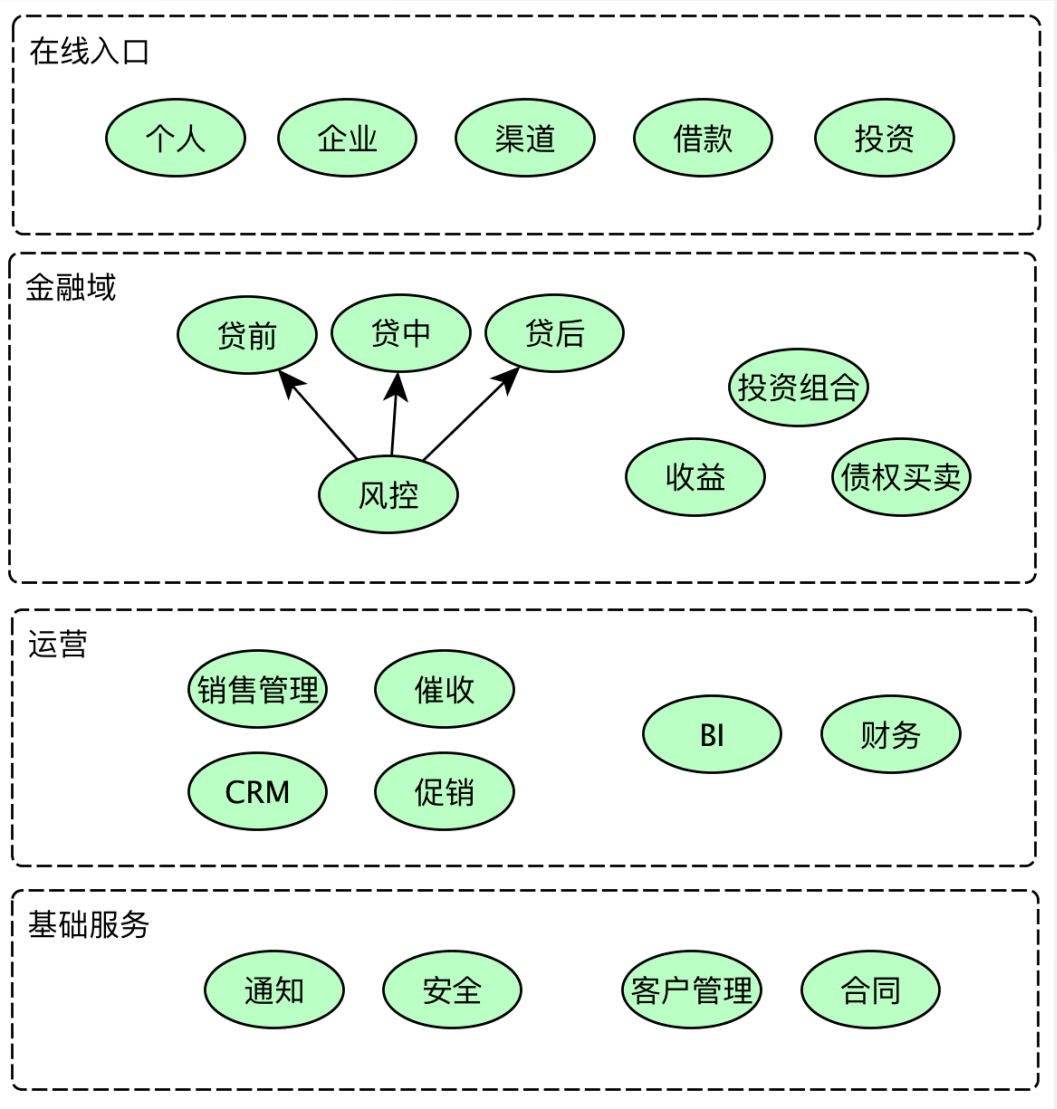
- 对子域区别对待：通用子域，支撑子域，核心子域，需投入不同的精力和人员
- 对开发人员的要求要做区分：不同level的人作不同的要求
- 单独立项，确保对业务进度无影响
- 太关注战术，而忽略了DDD在战略层面的意义：**战略要明确，战术要灵活**

康威定律

组织架构调整的依据

- 业务领域划分，决定了组织结构的划分
- 在组织内，定义其专属的代码库，组件、系统、迭代方式和发布周期
- 微服务到底应该多“微”？
  受业务复杂度和组织大小的影响，一般来说，4~8人的团队大小比较合适

业务域划分（示例）：



重新出发

- 成立专项小组，专注于基于DDD的业务重构
- 制定编码规范，并构建自动化质量检测器
- 完善对象的职责，角色和关系界定：**Bo，Controller，Service，Repository，Entity，Po，Mapper，Config，Request，Interceptor，Util，Client，Spec**
- 明确架构的层次，并确定每一层的职责：**接入层，业务层，服务层，基础设施层**

技术总结

- 大多数聚合都应该是“小聚合”
- 六边形架构适用于系统之间；DB不应被看做外部依赖
- 只对你的核心业务域使用DDD
- 贫血模式 & 充血模式 之争：要具体业务具体分析。例如：API 是“被动型”的业务模式，所以只有Service是主动的，充血的，其他辅助类都是“贫血”的
- 基础设施层的核心作用是：实现技术逻辑与业务逻辑的分离，至于具体的实现方式是次要的。

管理总结

- 团队变革的核心是人，人的认知改变需要时间
- 渐进式、迭代式，不要强求一蹴而就
- 不要迷信技术，人和技术的优化配置才能开发出最经济有效的软件产品
- 团队技术进阶的切入点选择：“跳一跳能够得着”
- 制定规范的重要性：让共识得以延续，让管理者专注于“例外情况”

方法论总结

- DDD的战略意义大于战术意义，它强调的是”理解业务，做正确的事”
- 实践DDD的最高境界是“忘掉DDD”：合适的设计才是最好的设计，**没有最佳实践**
- 除了DDD，还有很多事情要做，如：技术架构，数据架构，性能，安全等等

谢谢！