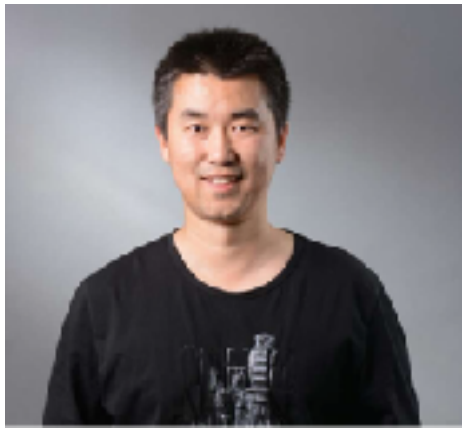
微服务架构与领域驱动设计

王磊

关于我



华为 - 2012技术专家

ThoughtWorks - Lead Consultant

Sybase - Tech Leader

- 丰富的持续交付/微服务架构/DevOPS经验
- 《微服务架构与实践》作者
- 《DevOps实践指南》译者
- 中国首批EXIN DevOps Master教练
- 西安DevOps Meetup 联合发起人
- 《消费者驱动契约测试-Pact》译者
- 《使用SpringBoot/Cloud构建微服务》视频作者(StuQ)

01 微服务架构与DDD

02 领域驱动设计的核心

03 基于事件风暴的DDD实践

什么是微服务架构

微服务架构

Microservices - the new architectural style

Martin Fowler, Mar 2014

微服务架构是一种架构模式，它提倡将单一应用程序划分成**一组小的服务**，服务之间互相协调、互相配合，为用户提供最终价值。

每个服务运行在其**独立的进程中**，服务与服务间采用**轻量级的通信机制**互相协作（通常是基于HTTP协议的RESTful API）。

每个服务都围绕着具体业务进行构建，并且能够**被独立的部署**到生产环境、类生产环境等。

微服务架构

以**持续交付**为核心

基于**DevOps**

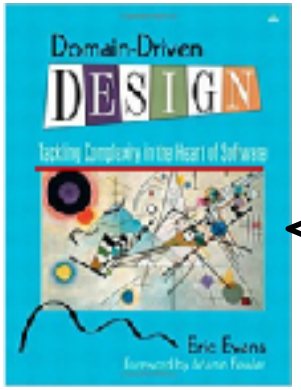
的**演进式架构**

A thousand Hamlets in a thousand people’s eyes.

Shakespear

什么是领域驱动设计

领域驱动设计



领域驱动设计是一种设计方法，围绕**业务概念**构建**领域模型**，并通过分离技术实现的复杂性，从而控制软件演化的复杂度。

Tacking Complexity in the heart of software

领域驱动设计解决的两个核心问题：

1.业务架构如何合理的设计划分?

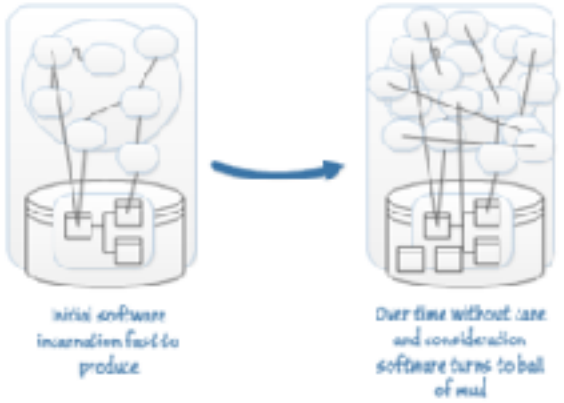
2.技术架构与业务架构保持一致?

微服务架构与领域驱动设计?

微服务架构与领域驱动设计

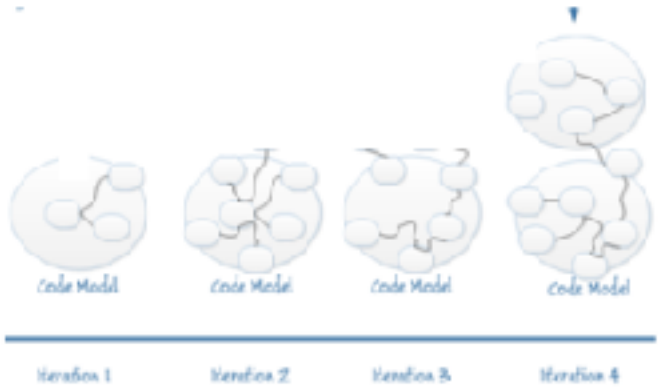
- **能有效支撑演进式架构**

项目1：
没有使用领域模型，对设计没有重视
✓ 迅速完成第一个版本
✓ 一年后陷入困境



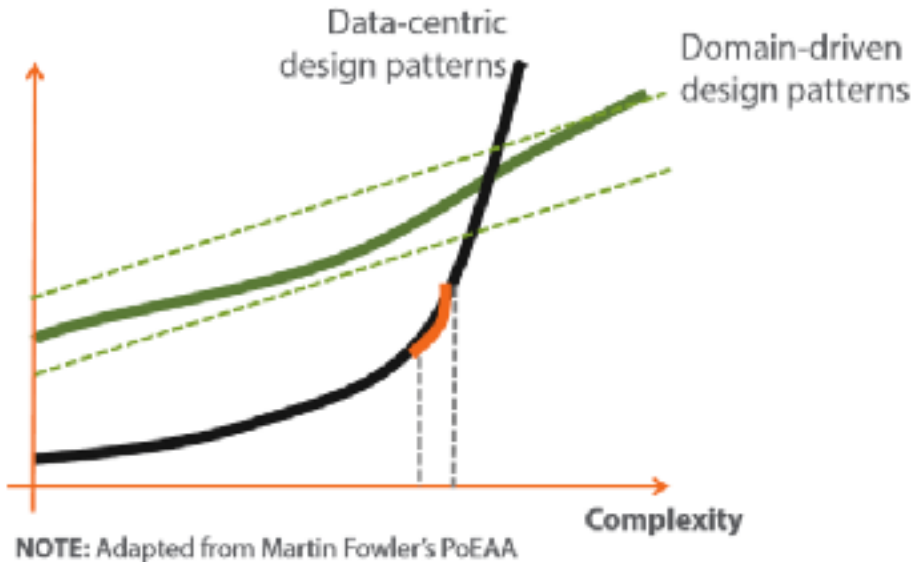
项目2：
业务专家进行高瞻远瞩的全局分析建模
✓ 分析模型过于复杂
✓ 分析模型与程序设计脱节

项目3：
重视领域模型，迭代开发完善模型
✓ 提供灵活性和扩展性
✓ 根据团队对领域的不断理解，深化领域模型



微服务架构与领域驱动设计

- 能有效降低复杂软件的维护成本

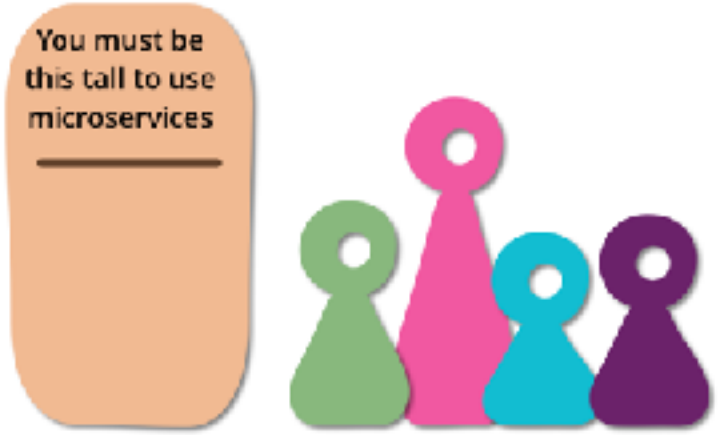


随着时间的推移，采用领域驱动设计比采用以数据中心设计的软件复杂度要低得多。

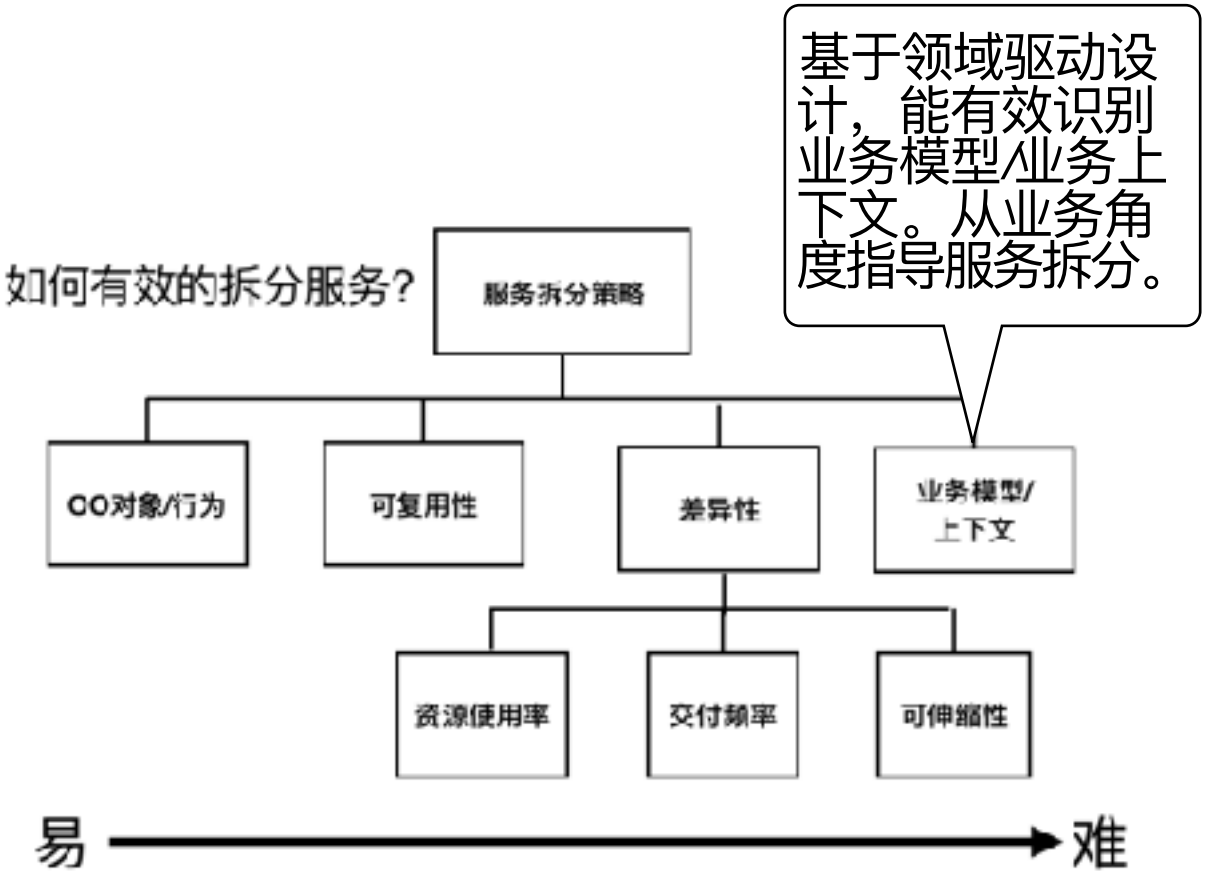
《Pattern of EEA》

微服务架构与领域驱动设计

- **能有效指导服务的设计与划分**



- **分布式的复杂度**
  - 网络因素(带宽、超时)
  - 数据一致性
- **运维成本高**
  - 环境配置(Provisioning)
  - 部署/监控
- **微服务的治理**
  - 依赖管理
  - 服务依赖
- **组织结构调整**
  - 团队文化
  - 沟通成本



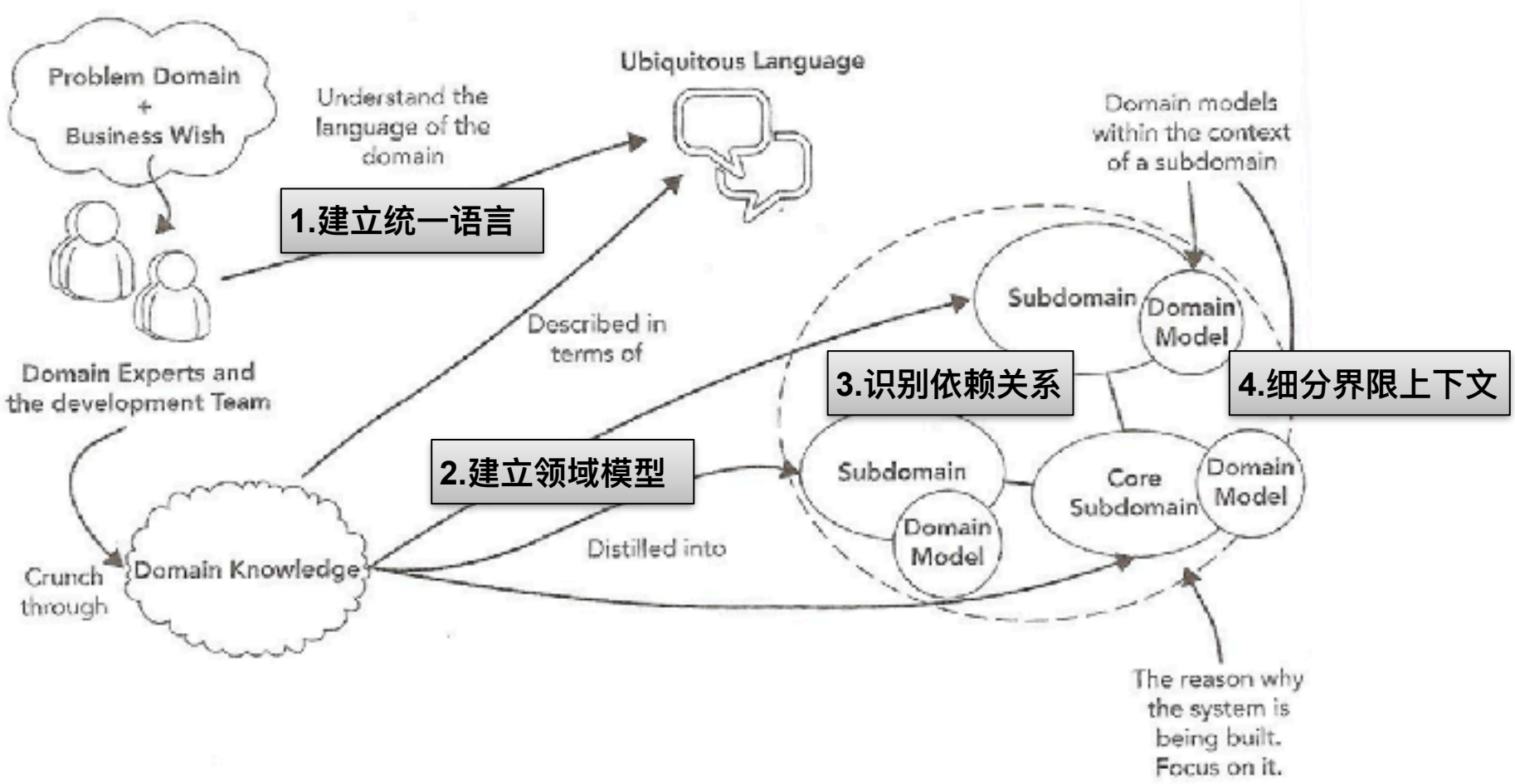
01 微服务架构与DDD

02 领域驱动设计的核心

03 基于事件风暴的DDD实践

领域驱动设计的核心

将业务架构和系统架构对应起来，建立针对业务变化的高响应力架构



领域驱动设计的理论基础

Strategic-战略建模

Ubiquitous Language - 统一语言

- 使用一致的业务描述语言

Domain&Subdomains - 领域&子域

- Core Domain
- Supporting Domain
- Generic Domain

Bounded Contexts - 业务上下文

- 定义领域模型的应用范围和其上下文

Context Mapping - 上下文映射

- 负责不同上下文之间的协作
- Shared Kernel
- Anti-Corruption Layer

Tactical-战术建模

Entity - 实体对象

- 一个对象通过ID被唯一标识

Value Object - 值对象

- 通过值确定对象的等价性

Repository - 存储机制

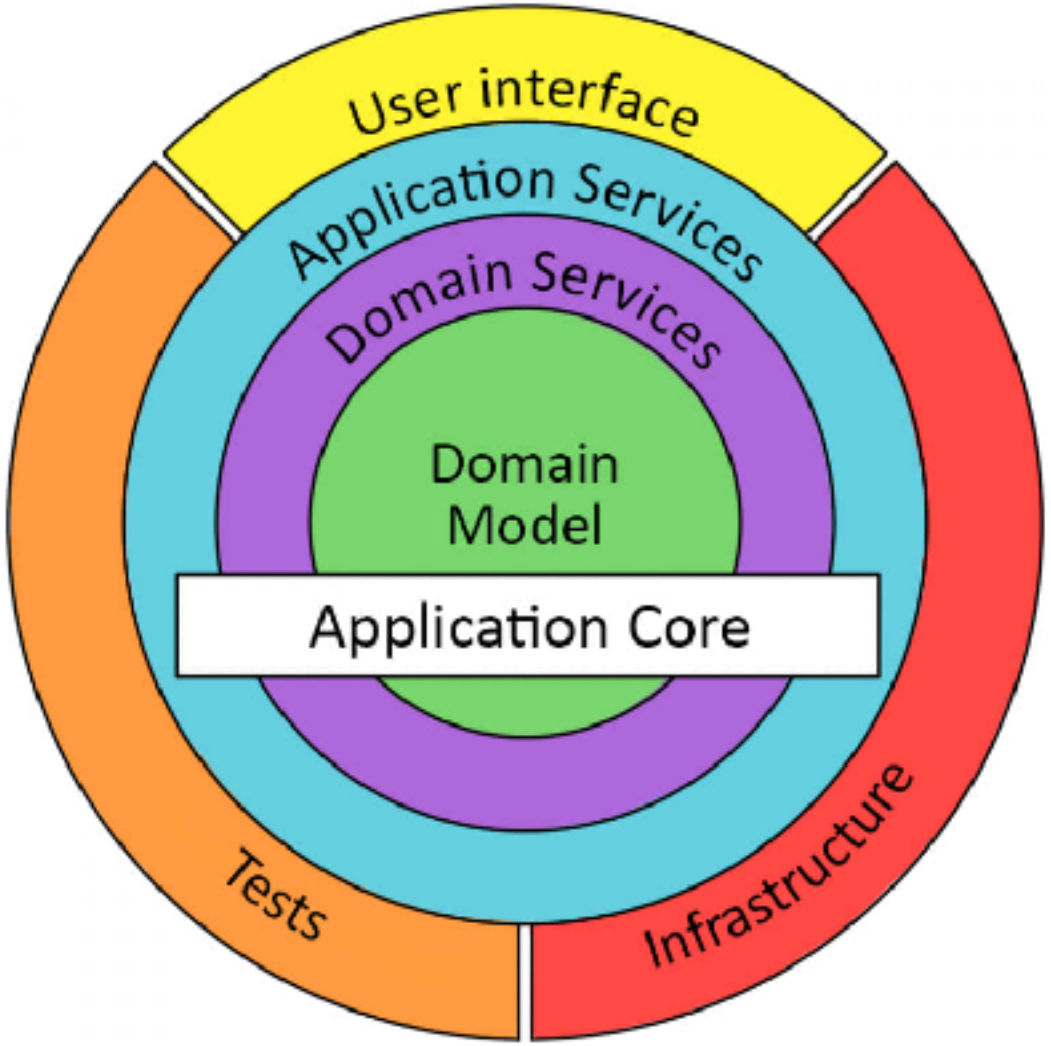
Aggregate - 聚合(根)

- 一组实体对象和值对象的集合
- 外部对聚合的访问通过聚合根

Factories - 对象工厂

Domain Services - 业务逻辑

领域驱动设计的架构



Onion架构

Domain Model

- 核心的领域模型，包括Entity/ValueObject/Aggregate

Domain Services

- 领域模型的业务逻辑

Application Services

- 同应用相关的接口适配

User Interface

- 用户接口相关的部分

Infrastructure

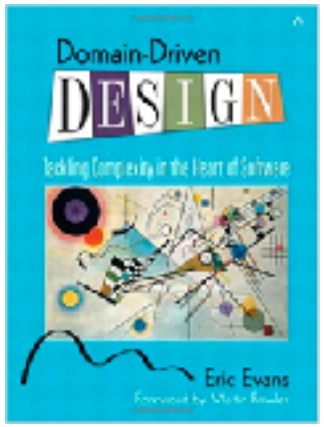
- 数据相关部分：WebService/DB/File
- 支撑相关部分: 环境相关配置

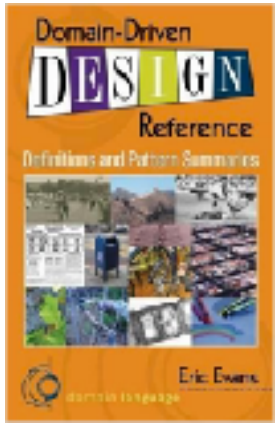
01 微服务架构与DDD

02 领域驱动设计的核心

03 基于事件风暴的DDD实践

如何落地领域驱动设计?



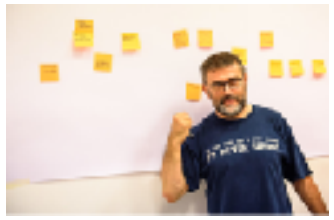


Eric Evan

Vaughn Vernon

Event Storming

Event Storming是一种快速探索复杂业务领域的方法：



Alberto Brandolini

Effective: 可以让实践者在数小时内理解复杂业务模型

Engaging: 带着问题的和拥有答案的人一起来构建模型

Efficient: 能快速发现界限上下文以及相关的聚合根等

参与人员：业务人员，领域专家，技术人员，架构师，测试人员等关键角色

开放空间：足够的空间将业务事件流可视化，让人们可以互相讨论

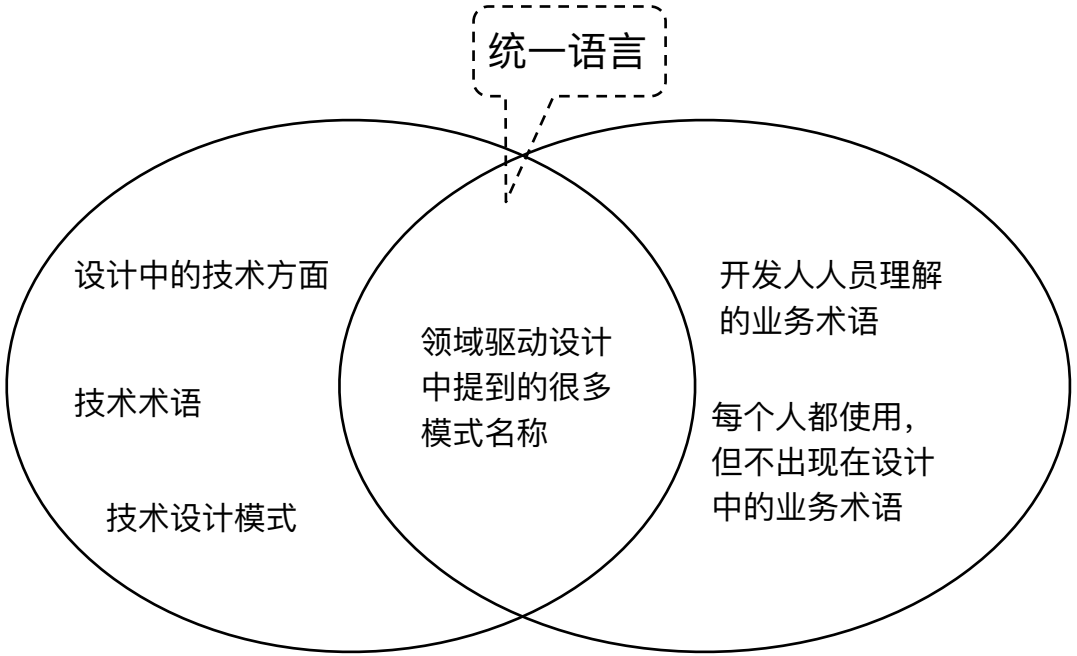
彩色即时贴：使用不同颜色标识

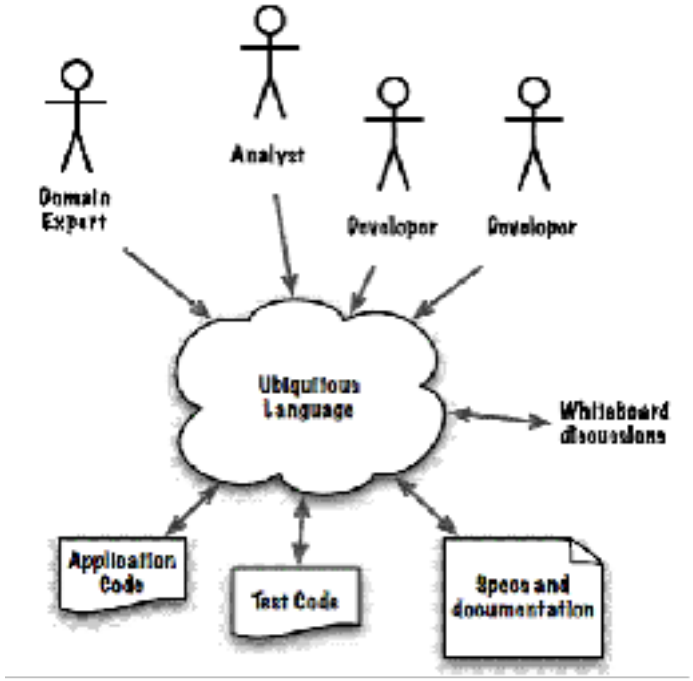
如何使用Event Storming

①	②	③	④	⑤	⑥
统一语言	**业务流梳理**	**寻找事件**	**寻找命令**	**寻找聚合**	**划分子域&界限上下文**
顺畅的沟通	理解业务	发生了什么?	谁引起的?	通过什么产生?	分解系统（划分微服务）

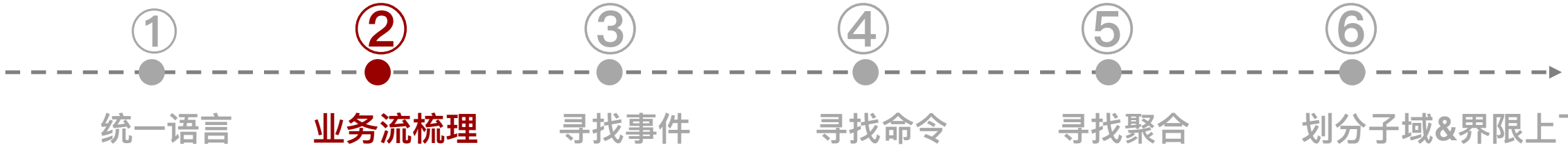
统一语言

① 统一语言　② 业务流梳理　③ 寻找事件　④ 寻找命令　⑤ 寻找聚合　⑥ 划分子域&界限上下文





业务流梳理



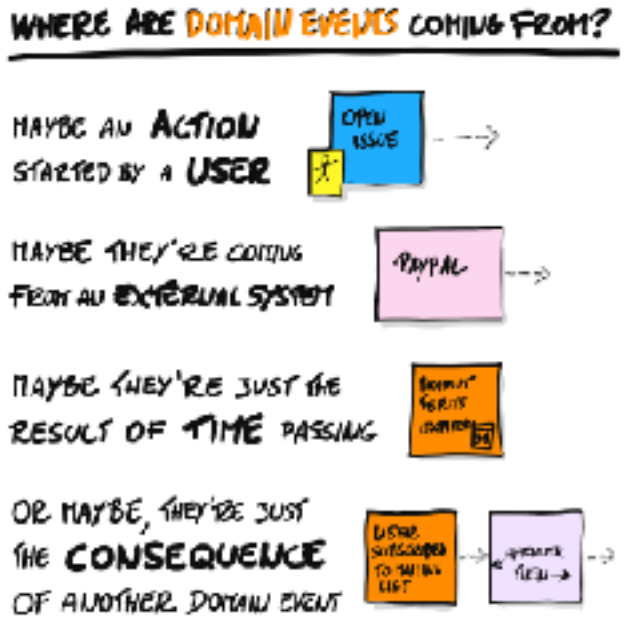
领域专家介绍业务，参与者可以任意提问，大家在理解业务的基础上梳理出业务流。

寻找事件

① 统一语言 — ② 业务流梳理 — ③ **寻找事件** — ④ 寻找命令 — ⑤ 寻找聚合 — ⑥ 划分子域&界限上下文

什么是(领域)事件?

- 任何的业务都会以数据的形式留下足迹。我们对于数据的追溯可以通过对事件的追溯来完成。当把这些事件按照时间顺序排列起来，几乎可以清晰的推测出在过往的一段时间内到底发生了深数据变化。



参与过程:

- 根据对业务的理解，将领域事件写在橙色贴上，每个即时贴代表一个事件
- 事件按照从左到右按时间顺序排列，不同参与者的事件需保证相对顺序
- 事件采用用“*xx*已*xx*”的格式，如“订单已创建”

注意要点:

- 业务流程中发生的事件
- 用“已发生”时态描述
- 有时间顺序

寻找命令

① 统一语言 ② 业务流梳理 ③ 寻找事件 ④ 寻找命令 ⑤ 寻找聚合 ⑥ 划分子域&界限上下文

什么是命令?

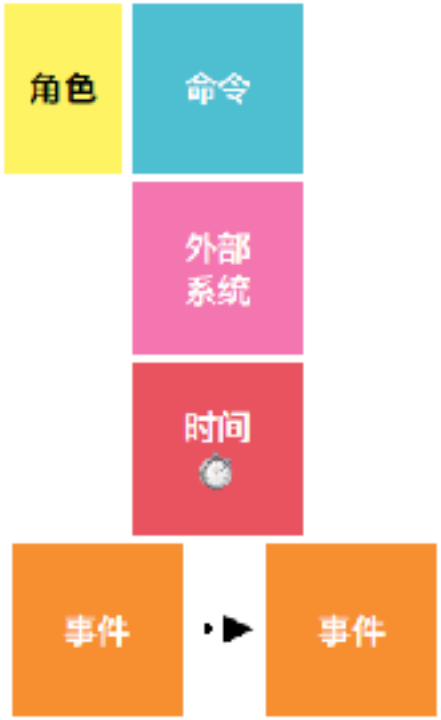
- 命令代表了外部系统或者用户触发的动作、以及内部的定时行为

参与过程:

- 将命令写在**蓝色即时贴上**
- 将命令贴在所产生的事件旁边
- 有的命令可能产生多个事件
- 识别出触发命令的外部系统和角色

注意要点:

- 用户从UI界面进行的操作
- 外部系统触发
- 定时任务



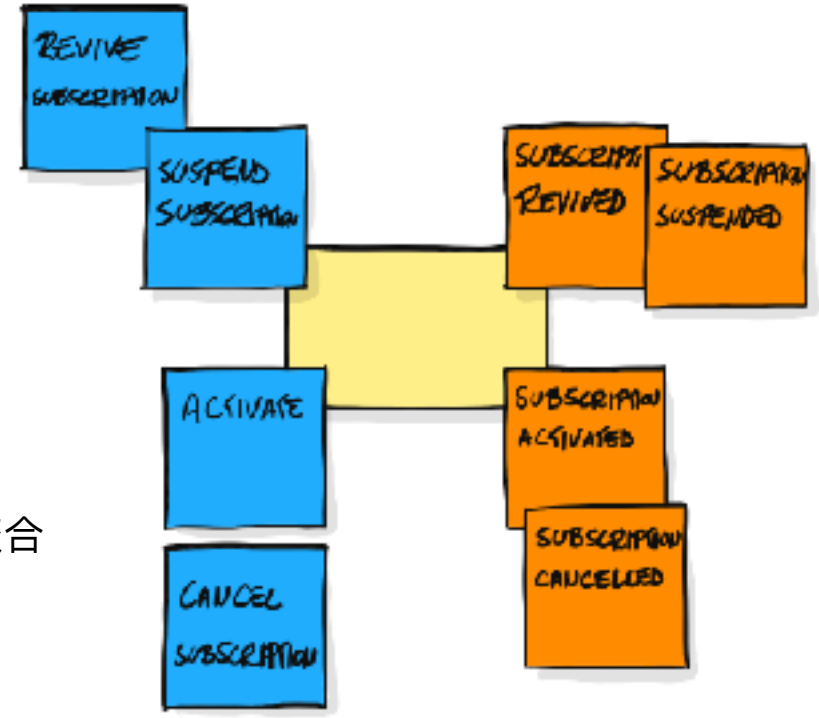
寻找聚合

① 统一语言 ② 业务流梳理 ③ 寻找事件 ④ 寻找命令 ⑤ 寻找聚合 ⑥ 划分子域&界限上下文

什么是聚合?

聚合是一组相关领域对象的集合。

- 聚合内相关对象保证数据一致性
- 只能通过聚合根修改边界内的对象
- 聚合根有唯一的全局标识



参与过程:

- 对命令和事件进行划分找到聚合边界
- 利用聚合定义进行确认
- 识别出分布在时间轴不同位置的同一个聚合
- 对聚合使用大的黄色即时贴进行标记

注意要点:

- 聚合接受命令，产生事件

划分子域&限界上下文

划分限界上下文

- 合并业务意义结合紧密的聚合
- 合并业务不确定性较高的聚合

限界上下文 ≈≈ 微服务

01 微服务架构与DDD

02 领域驱动设计的核心

03 基于事件风暴的DDD实践

Thank You

Website: http://servicecomb.incubator.apache.org/

Gitter: https://gitter.im/ServiceCombUsers/Lobby