

# 快速理解和拆分复杂业务

## 领域驱动设计

殷湘

# 大纲

- 交织 - 软件设计的挑战
- 蓝图 - 领域驱动设计
- 规律 - 事件风暴
- Workshop – 业务拆解建模实战

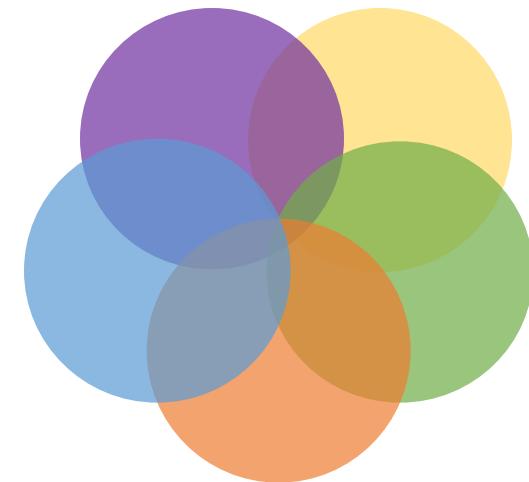
# 交织

软件设计面临的挑战



# 软件设计面临的挑战

- 如何快速理解复杂业务？
- 如何设计复杂系统架构？
- 如何拆解复杂业务，分而治之？



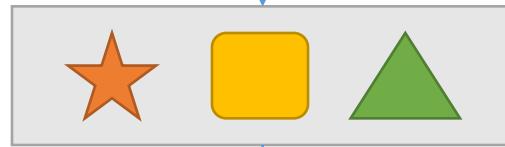
# 微服务如何拆分？

按层拆分？

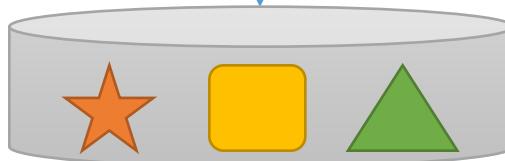
展现层



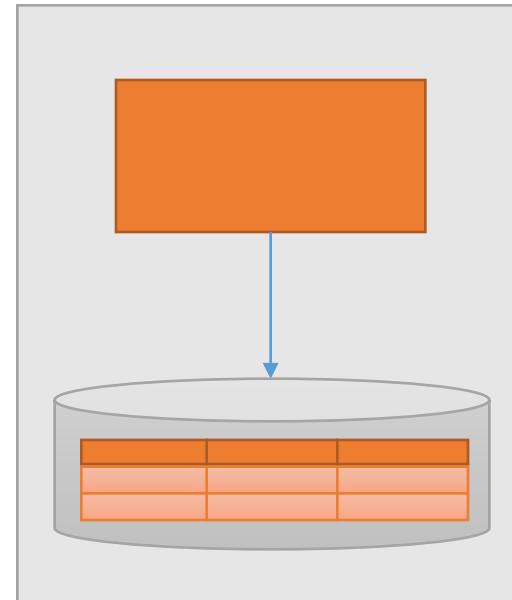
服务层



数据层



按表拆分？



# 微服务：多小才算“微”？

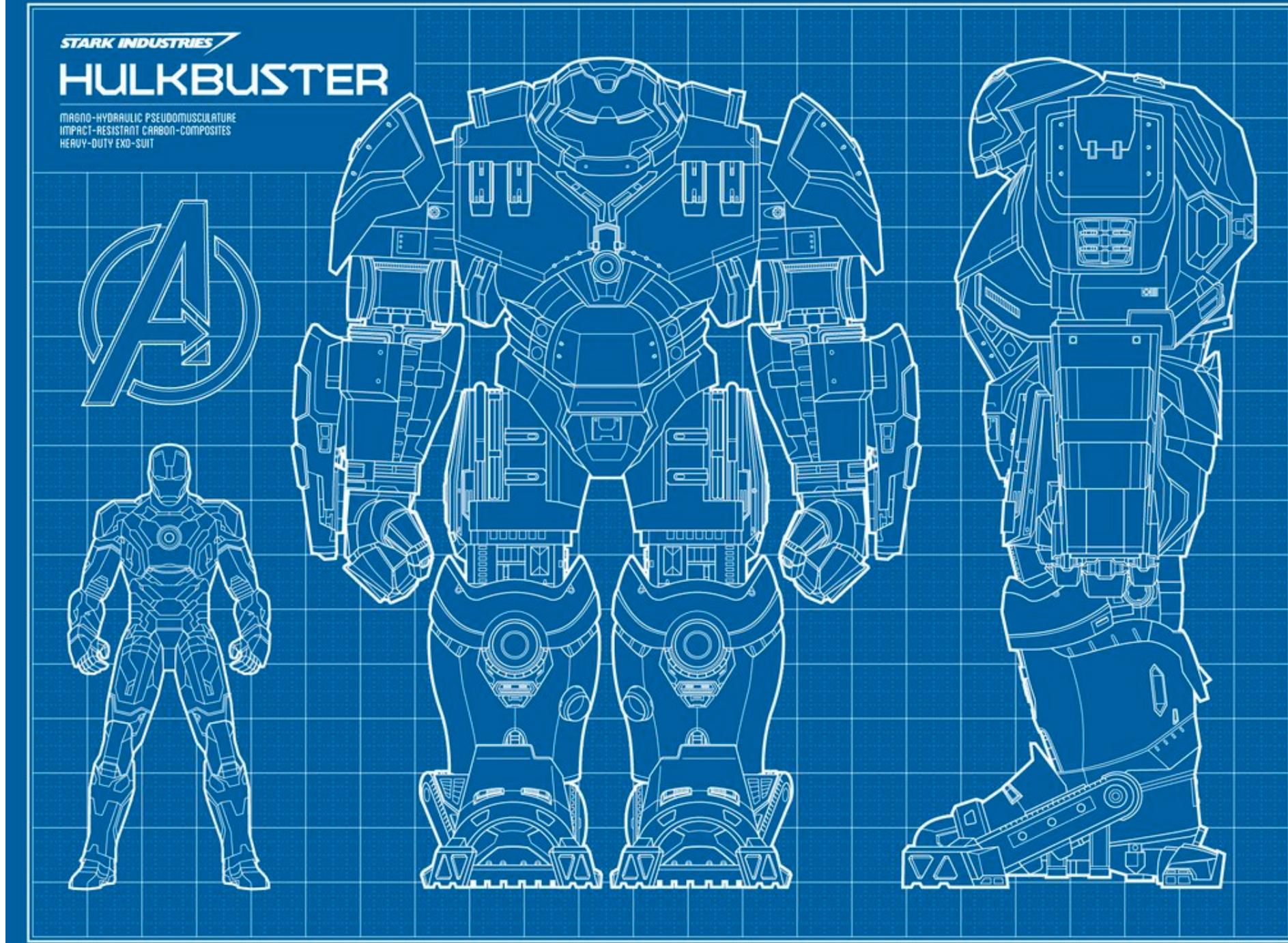
- 2 pizza team (按食量)
- 六周内完成重写 (按时间)
- 只做一件事 (按任务)
- < 300 行代码 (按代码量)

# 微服务：多小才算“微”？

- 2 pizza team (按食量)
  - 六周内完成重写 (按时间)
  - 只做一件事 (按任务)
  - < 300 行代码 (按代码量)
- 尺寸真的重要吗？是否有理可循？

蓝图

领域驱动设计

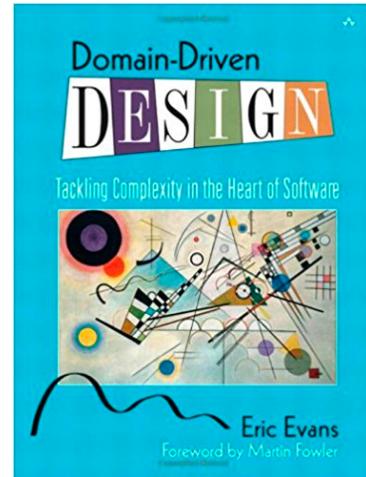


# 领域驱动设计 (DDD: Domain Driven Design)

- 软件的复杂度主要来自业务的复杂，而非技术
- 用围绕业务概念来构建领域模型的方式来控制业务的复杂性

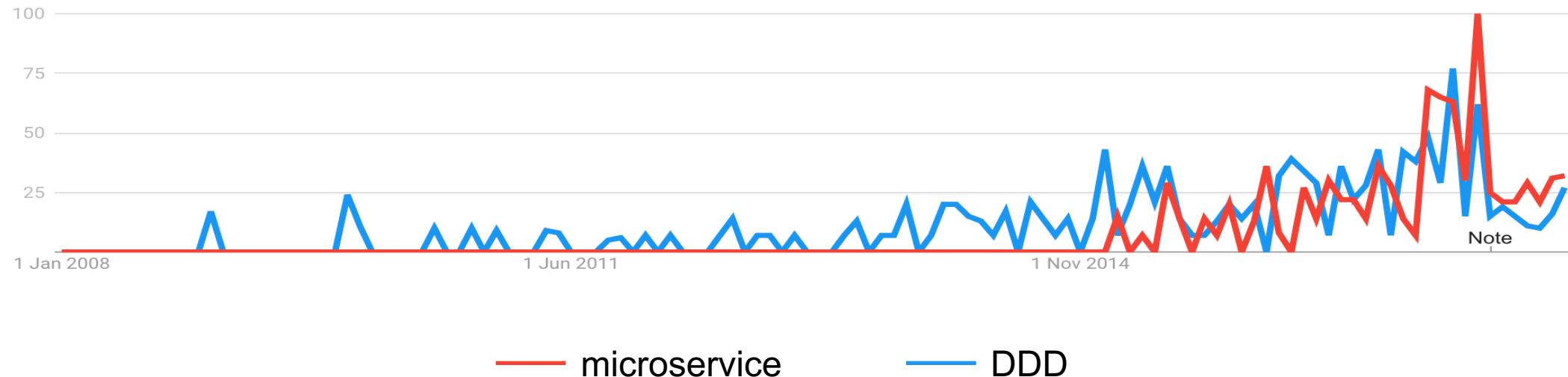
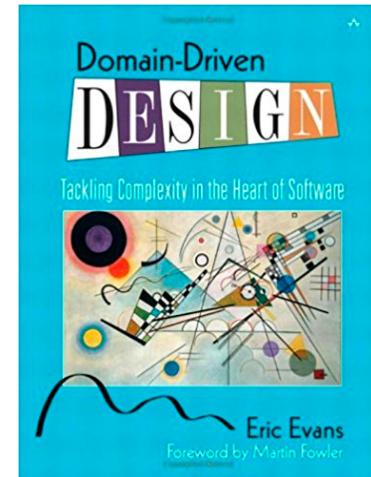
# 为什么DDD最近几年才火起来？

- Domain Driven Design, Eric Evans, 2003



# 为什么DDD最近几年才火起来？

- Domain Driven Design, Eric Evans, 2003



**DDD和微服务到底有什么联系？**

**Microservice is defined as a loosely-coupled, service-oriented architecture with bounded context**

“微服务就是有限界上下文的松耦合SOA架构”

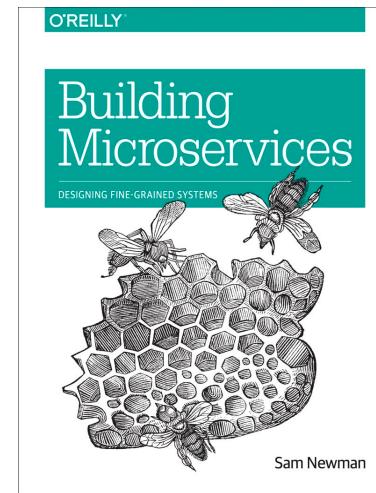


Adrian Cockcraft

VP - AWS  
Cloud Architect - Netflix

# Microservices should cleanly align to bounded contexts

“微服务应该与限界上下文对齐”

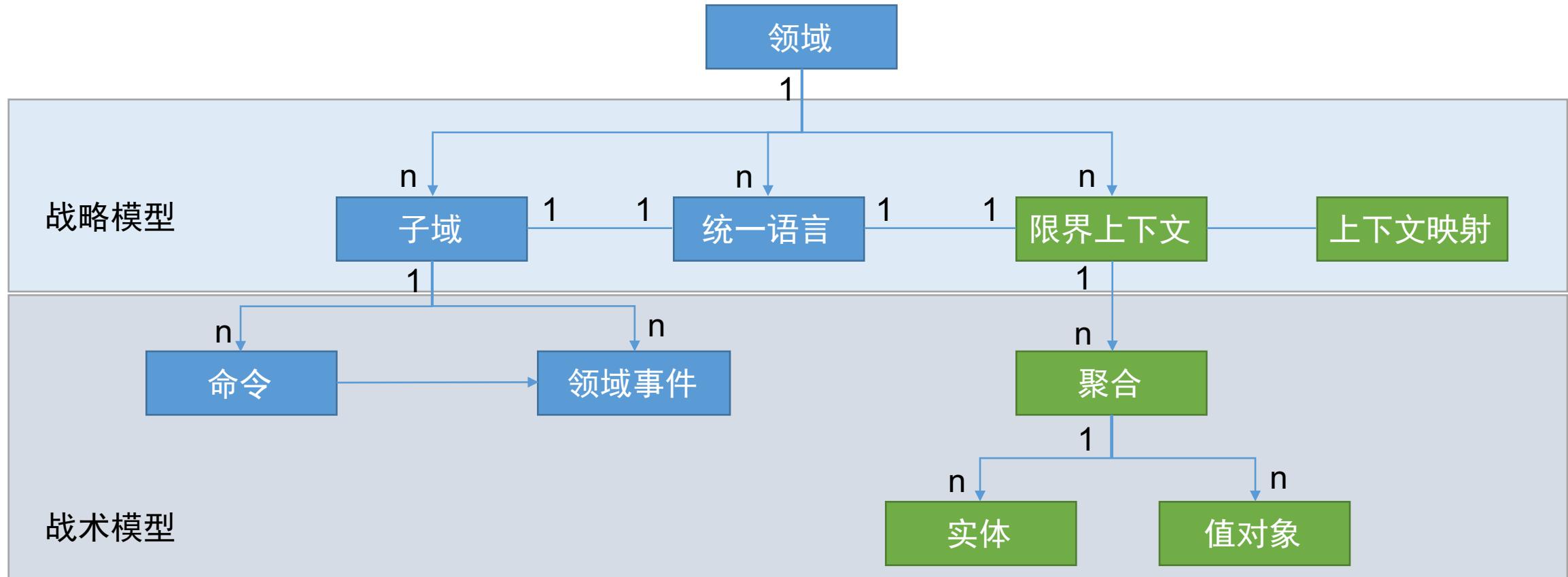


Sam Newman

Consultant  
ThoughtWorks

**限界上下文是什么？**

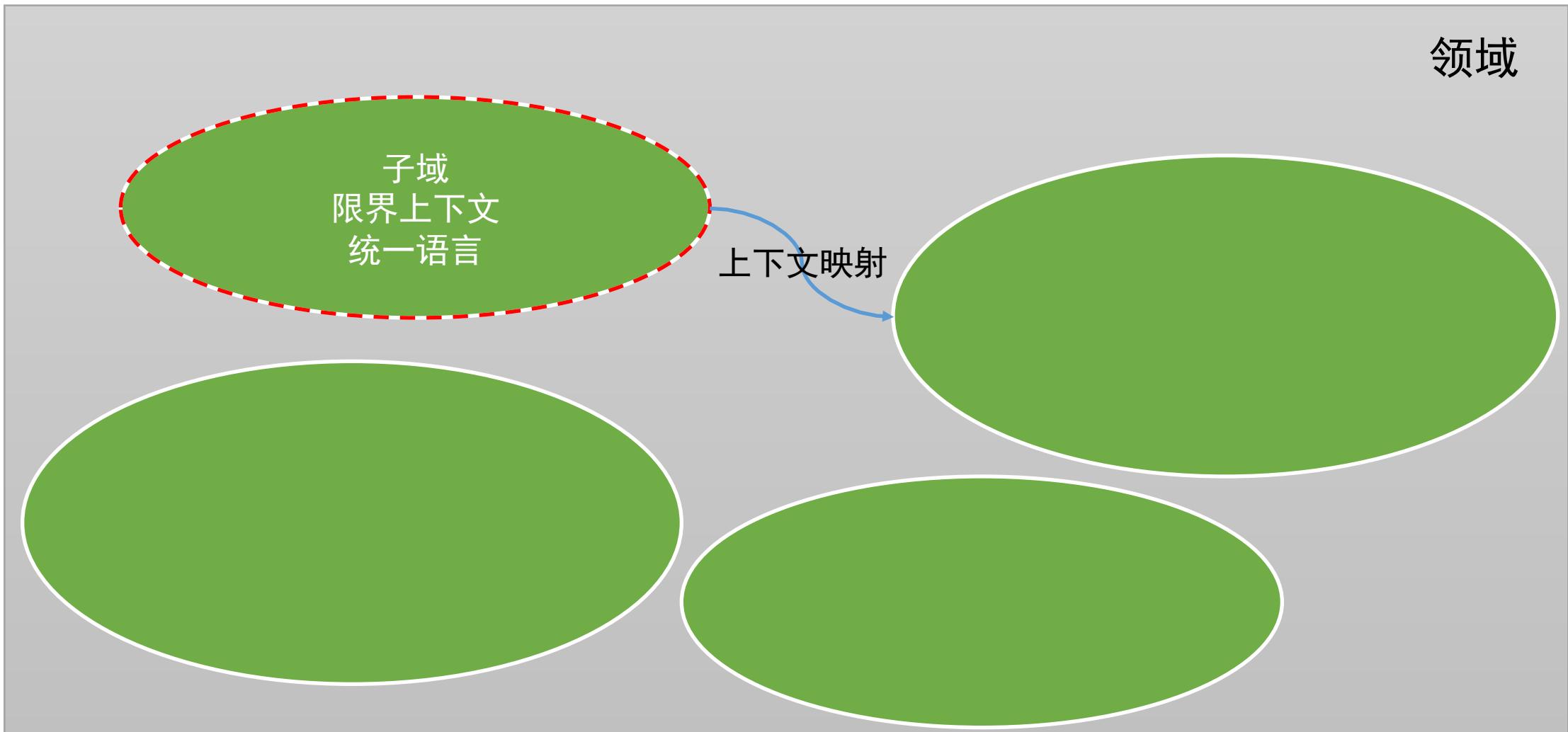
# DDD核心概念：145



问题空间: 业务领域

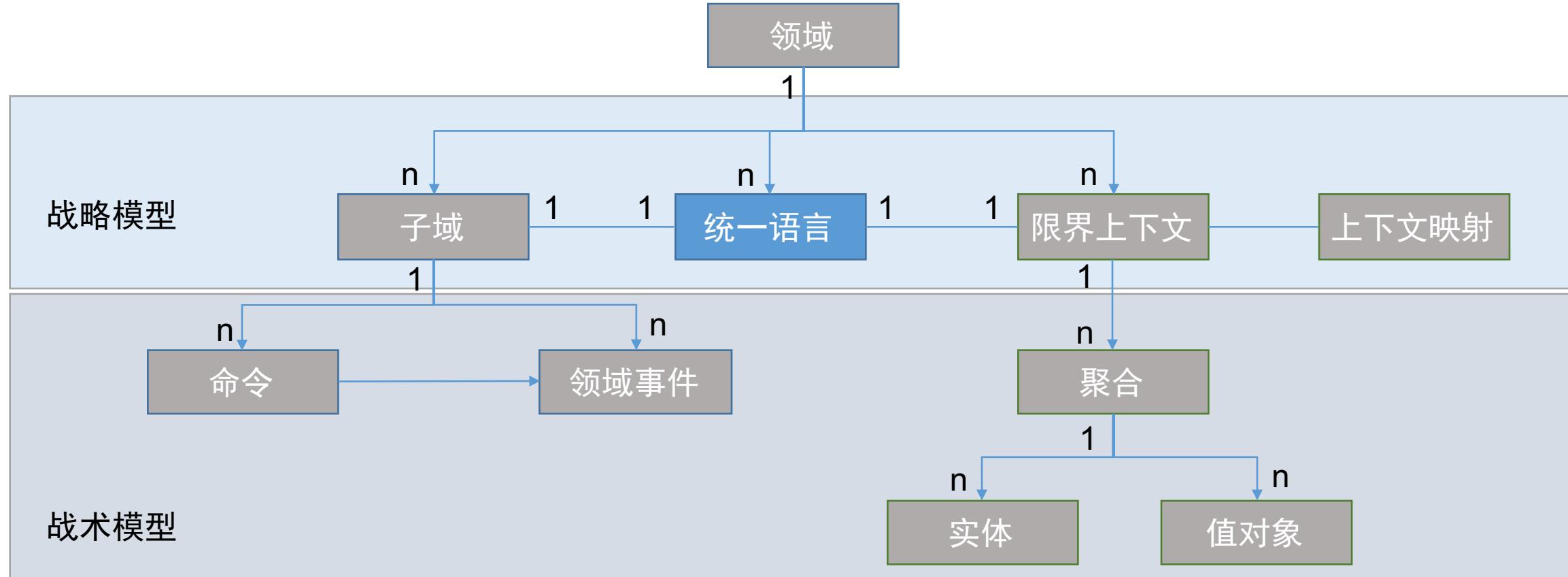
解决方案空间: 软件模型

# 战略模型：领域的全局观



# **领域(Domain)**：公司所从事的行业

- 金融
- 地产
- 保险
- 电商

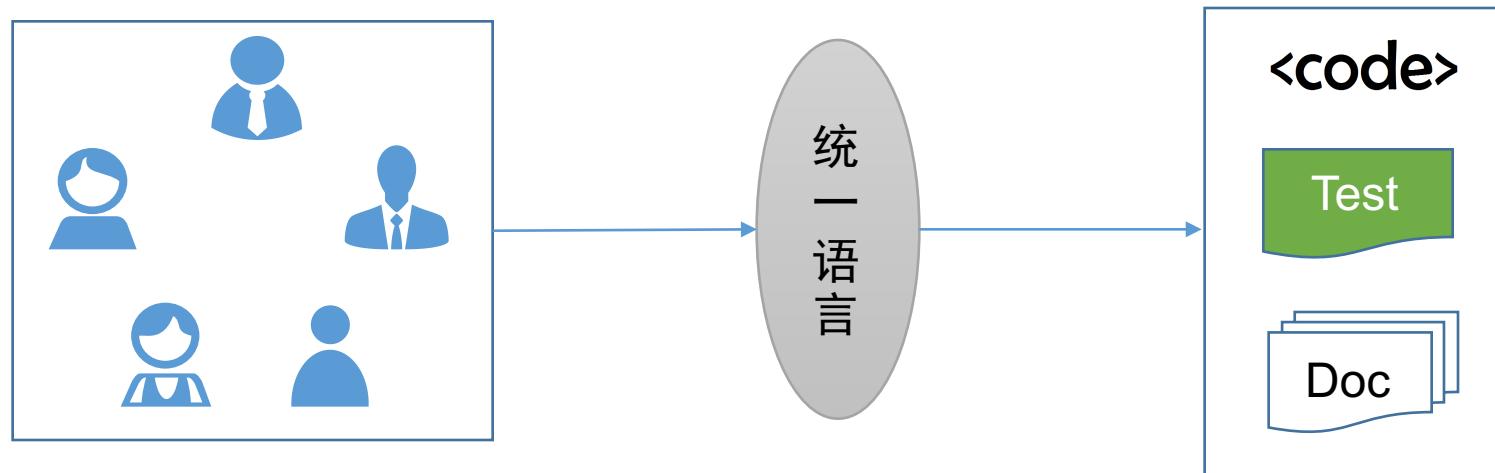


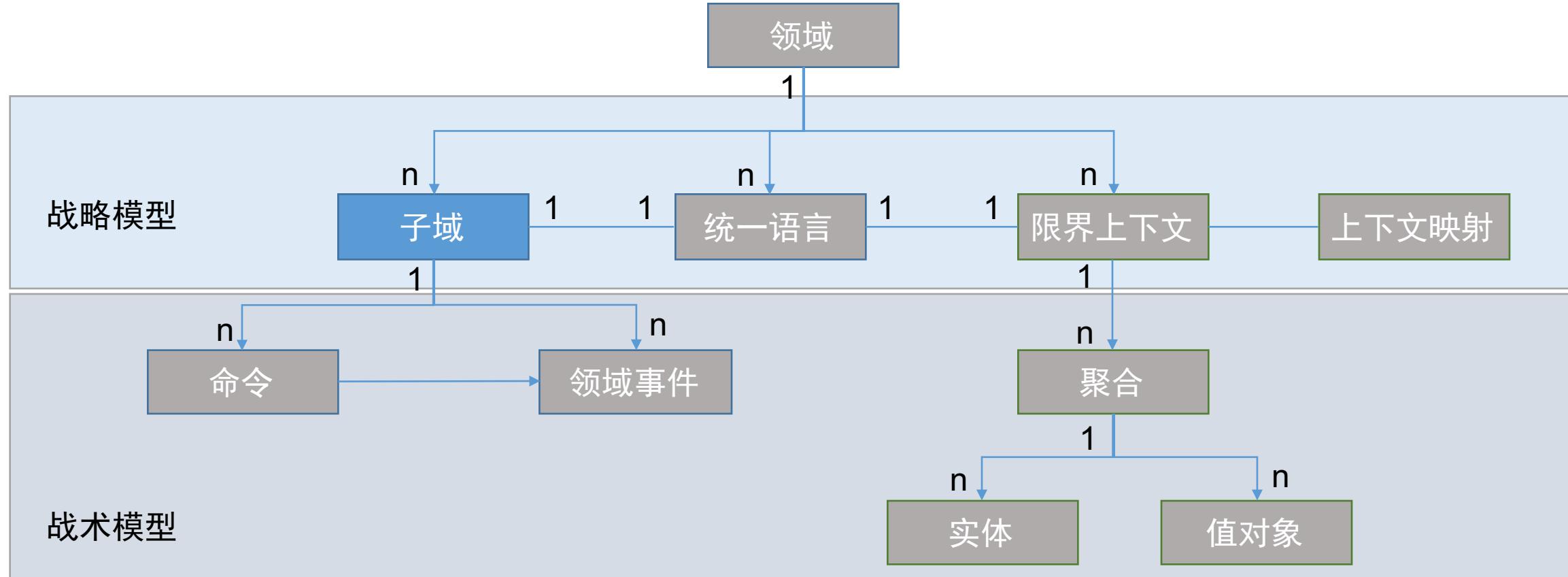
# 统一语言 (ubiquitous language)

- 没有歧义领域模型术语

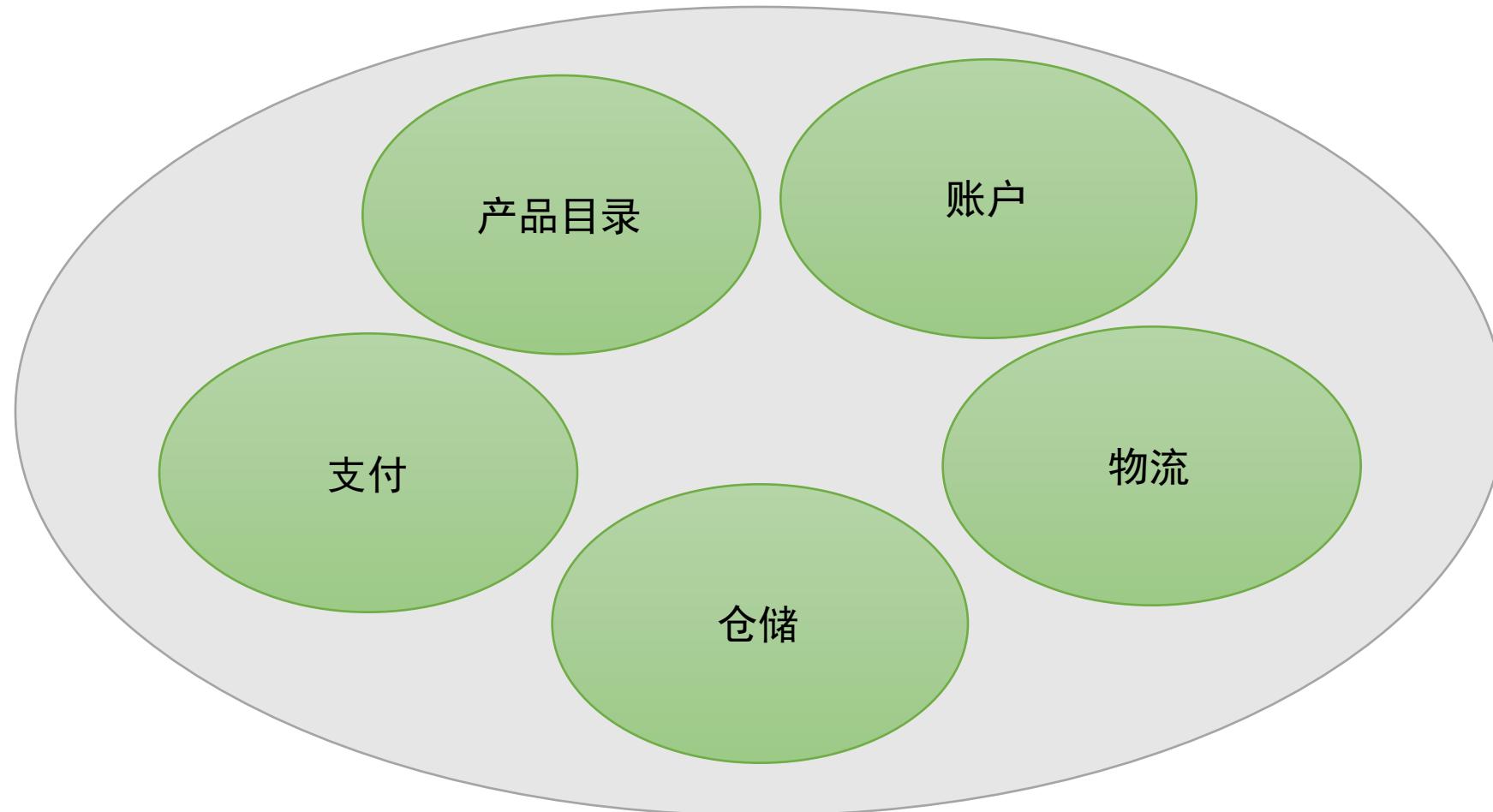


# 统一语言



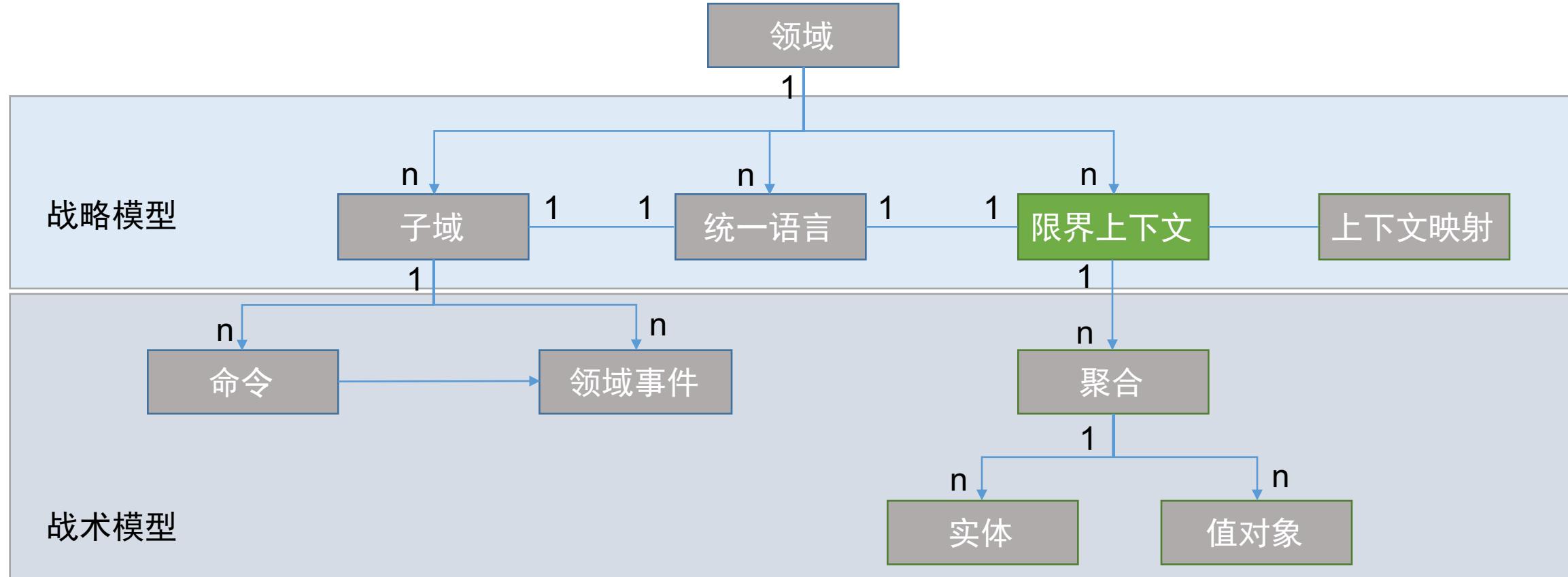


# 子域(subdomain)：以电商平台为例



# 子域的类型 (以京东为例)

- 核心域(core domain): 物流 (核心竞争力所在)
- 支撑域(supporting domain): 支付 (核心竞争力支撑)
- 通用域(general domain): 第三方认证 (可购买可外包)



# 限界上下文(bounded context)

- 限定子域边界的上下文
- 限界上下文 1 : 1 子域
- 分解业务复杂度
- 降低业务耦合度

# 如何确定限界上下文？

- 唯一的统一语言
- 单一职责原则 (高内聚): 业务关注点
- 一个团队

# 如何确定限界上下文？

- 用户、产品目录、订单都涉及用户，他们属于同一限界上下文吗？

产品目录

订单

用户

# 如何确定界限上下文？

- 用户、产品目录、订单所指的用户一致吗？

产品目录

订单

用户

# 如何确定限界上下文？

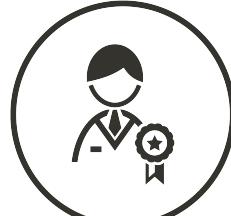
- 用户呈现的不同方面

 **产品目录**

- 性别
- 生日
- 购物历史
- 优惠券
- 地域

 **订单**

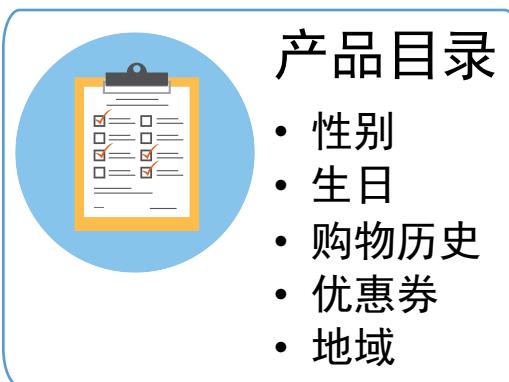
- 姓名
- 联系方式
- 收货地址

 **用户**

- 用户名
- 姓名
- 性别
- 联系方式
- 生日

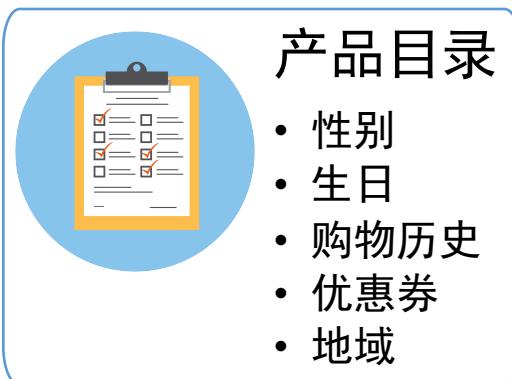
# 如何确定界限上下文？

- 用户呈现的不同方面
- 不同业务人群关心的问题 => **不同原因导致变化**



# 如何确定限界上下文？

- 用户呈现的不同方面
- 不同业务人群关心的问题 => 不同原因导致变化
- 单一职责：一个限界上下文，同一业务人群，相同变化原因（**关注点分离**）



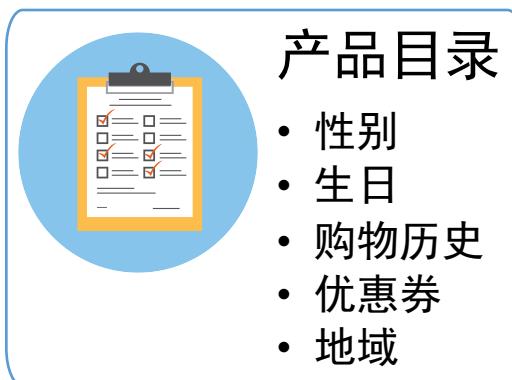
派送问题



身份问题

# 如何确定限界上下文？

- 语义不能统一
- 不同职责
- 不同限界上下文



派送问题



身份问题

# 如果共用同一用户数据有什么问题？

产品目录

订单

用户

- 用户名
- 姓名
- 性别
- 生日
- 联系方式
- 收货地址
- 购物历史
- 优惠券
- 地域

# 如果共用同一用户数据有什么问题？

- 如果订单需要修改用户数据结构怎么办？

产品目录

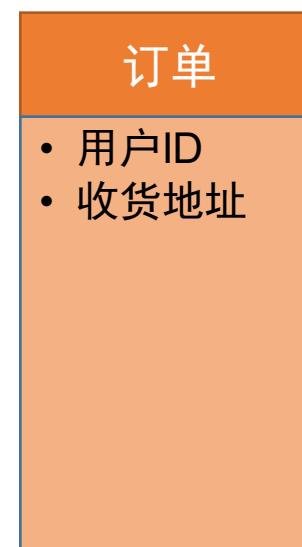
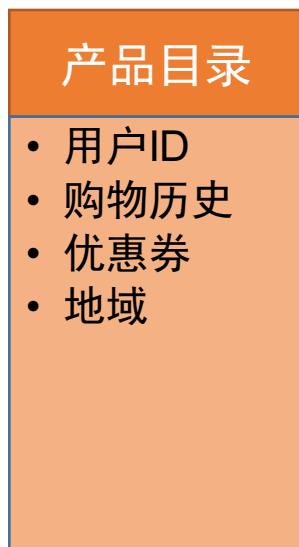
订单

用户

- 用户名
- 姓名
- 性别
- 生日
- 联系方式
- 收货地址
- 购物历史
- 优惠券
- 地域

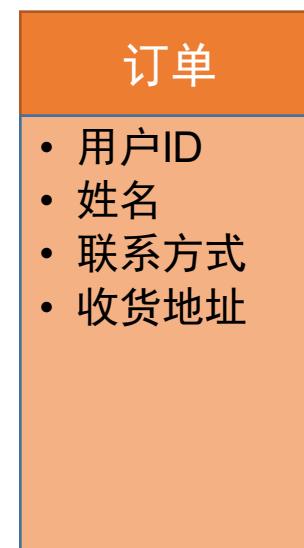
# 上下文之间用户数据如何映射？

- 如果用户服务不可访问，其他服务可独立运行吗？



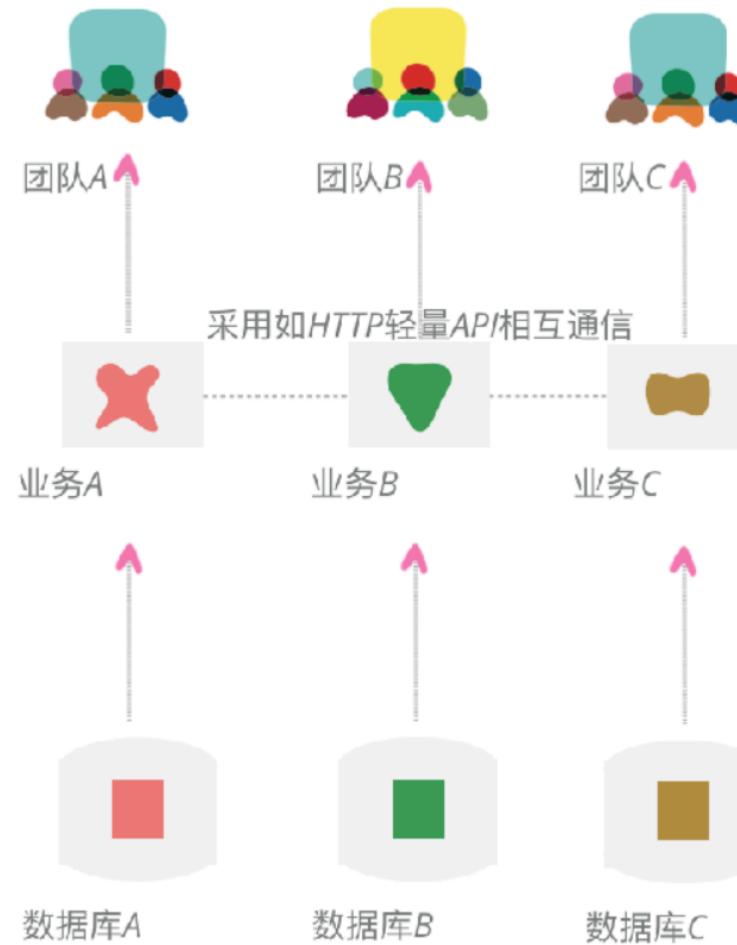
# 上下文之间用户数据如何映射？

- 如果用户服务不可访问，其他服务可独立运行吗？
- 用户服务推送更新，产品目录和订单服务接受消息，更新用户信息

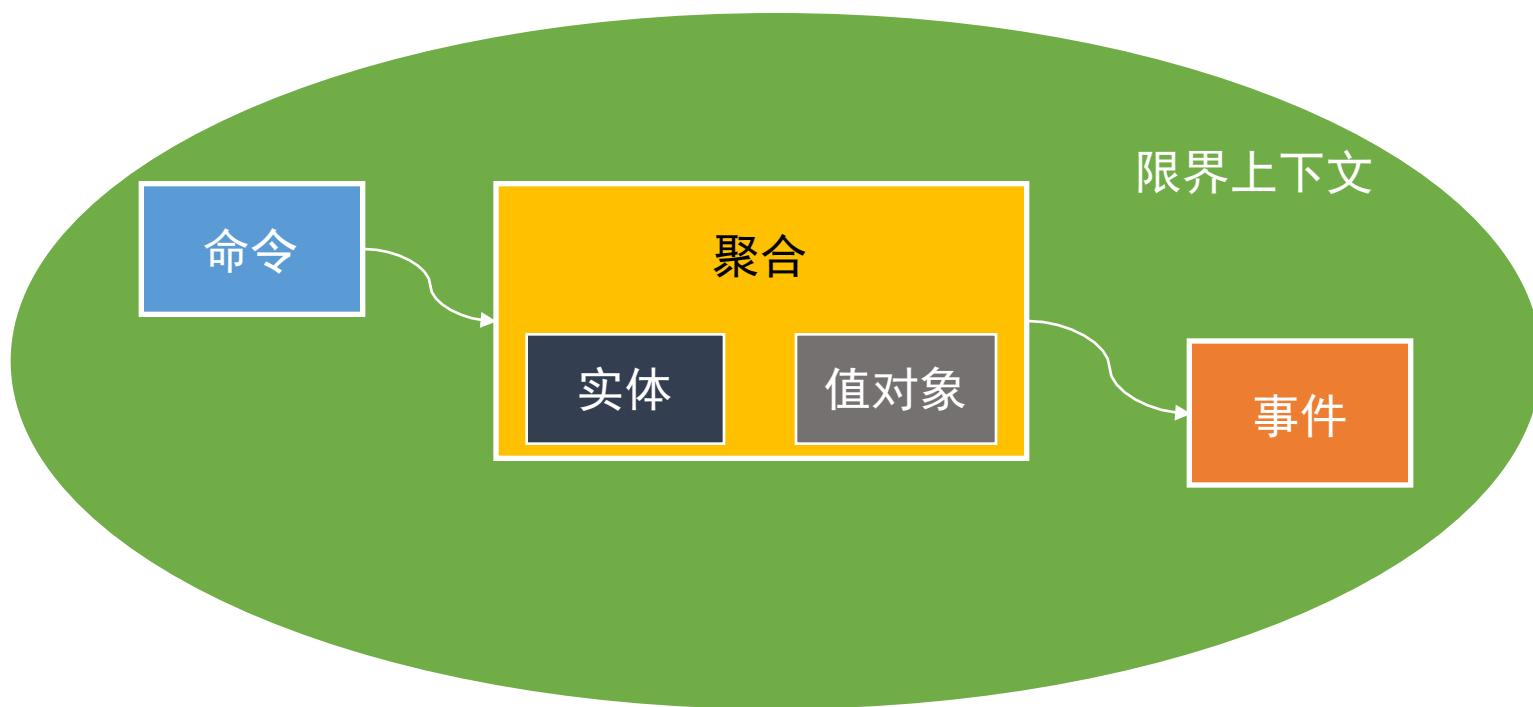


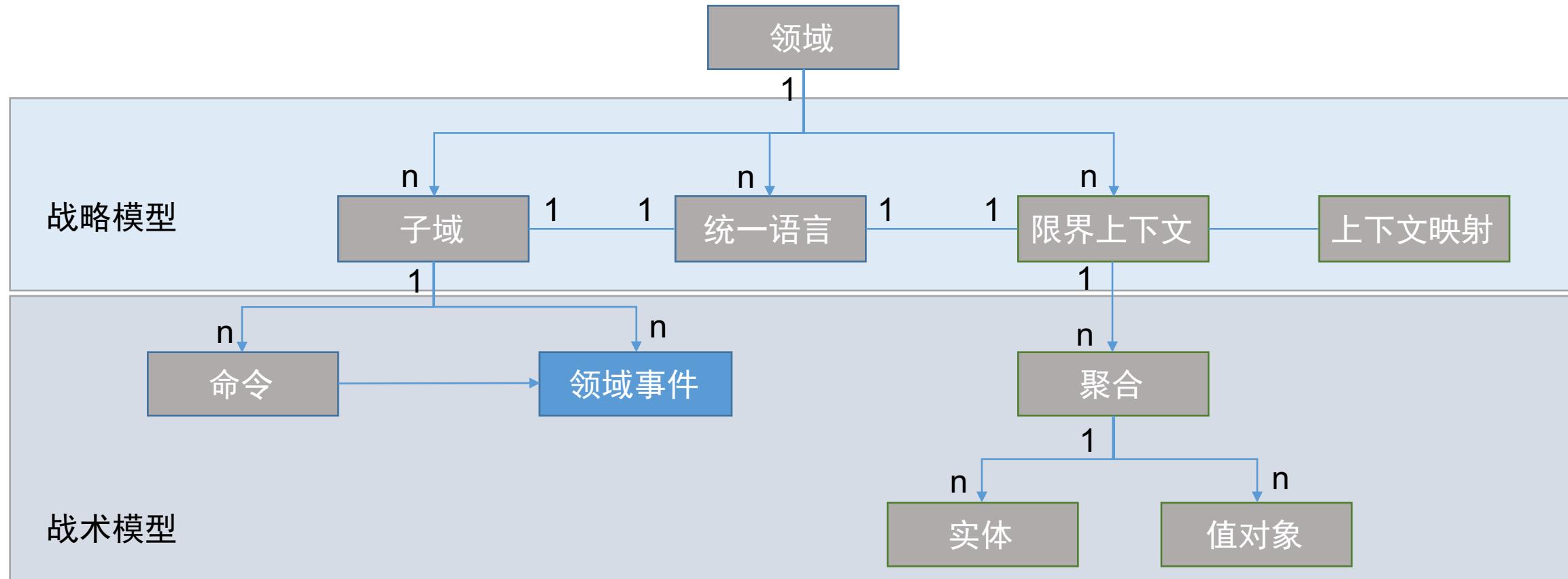
# 微服务：如何拆分？多小才算“微”？

- 微服务 1:1 限界上下文
- 跟子域业务范围有关
- 尺寸不是首要因素
- 微服务
  - 应用边界
  - 数据边界



# 战术模型：限界上下文的内部细节





# 领域事件(domain event)

- 业务关心的
- 已发生的
- 有时序顺序的



## WHERE ARE DOMAIN EVENTS COMING FROM?

MAYBE AN ACTION  
STARTED BY A USER



MAYBE THEY'RE COMING  
FROM AN EXTERNAL SYSTEM

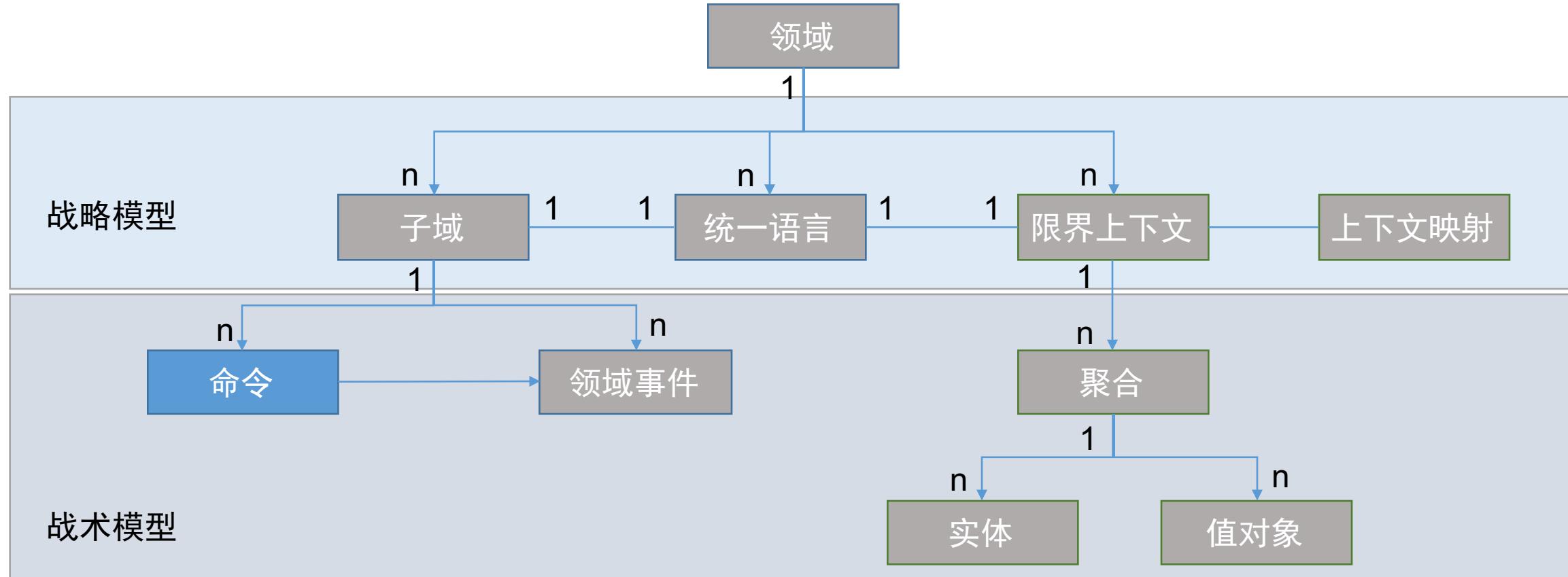


MAYBE THEY'RE JUST THE  
RESULT OF TIME PASSING



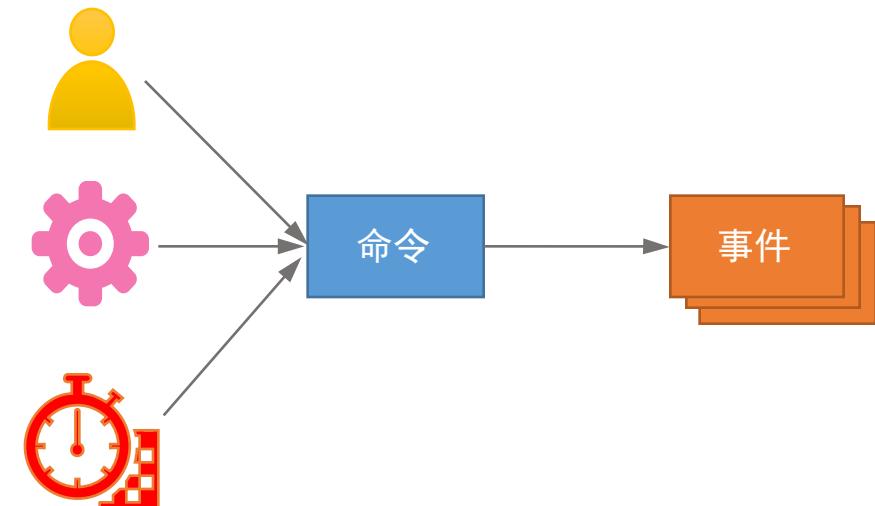
OR MAYBE, THEY'RE JUST  
THE CONSEQUENCE  
OF ANOTHER DOMAIN EVENT

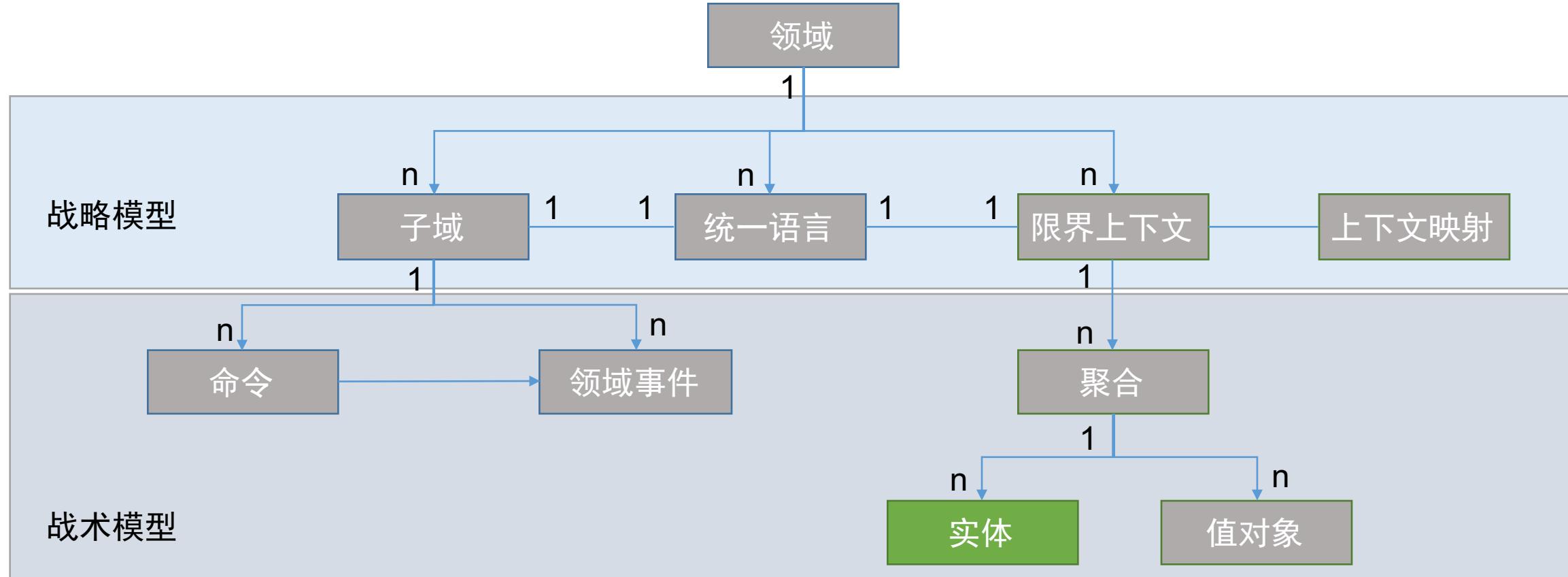




# 命令(command)

- 用户从UI触发的操作
- 外部系统触发
- 定时任务





# 实体(entity)

- 实体

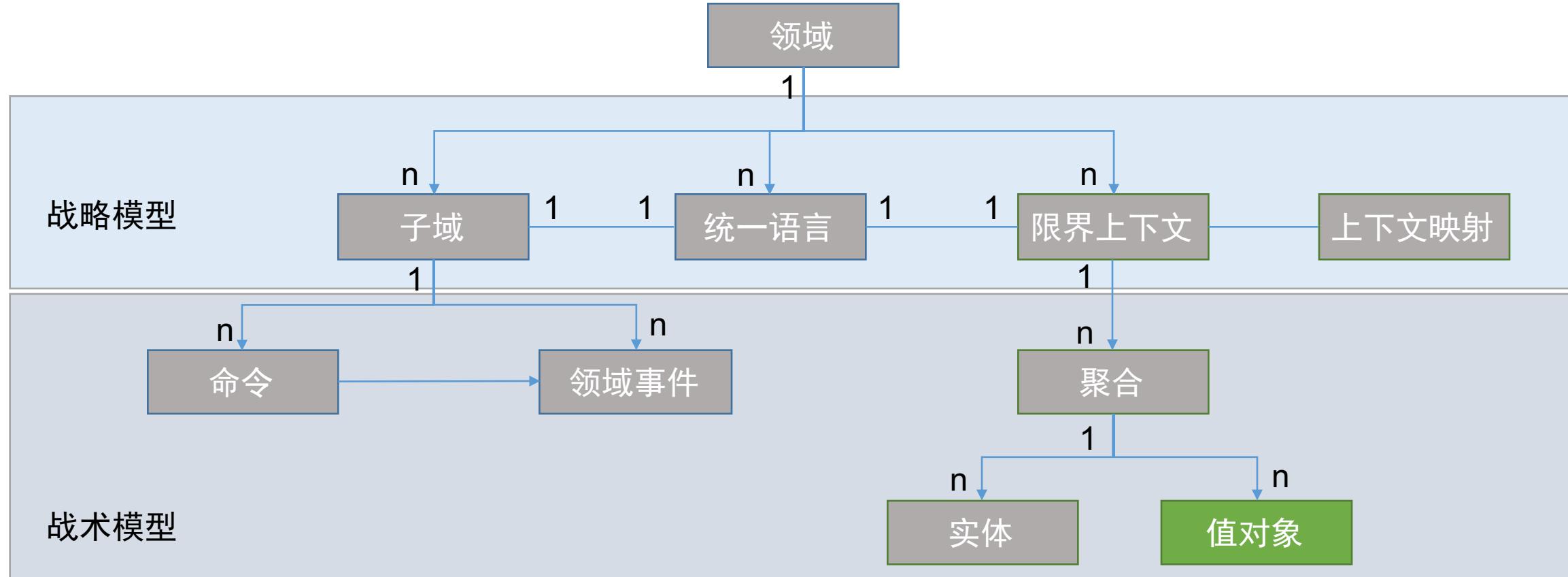
- 有生命周期 (创建、修改、删除)
- 有唯一标识
- 通过唯一表示辨别处于生命周期不同阶段的同一实体

用户子域

User
• id
• name
• avatar
• contact

手机子域

Phone
• id
• name
• color
• weight



# 值对象(value object)

- 值对象
  - 没有生命周期
  - 没有唯一标识
  - 通过属性判断是否同一值对象
  - 模型复杂度低
  - 建模时优先考虑是否能用值对象
- 例如：电影院订票系统里，座位是实体还是值对象？

订单子域

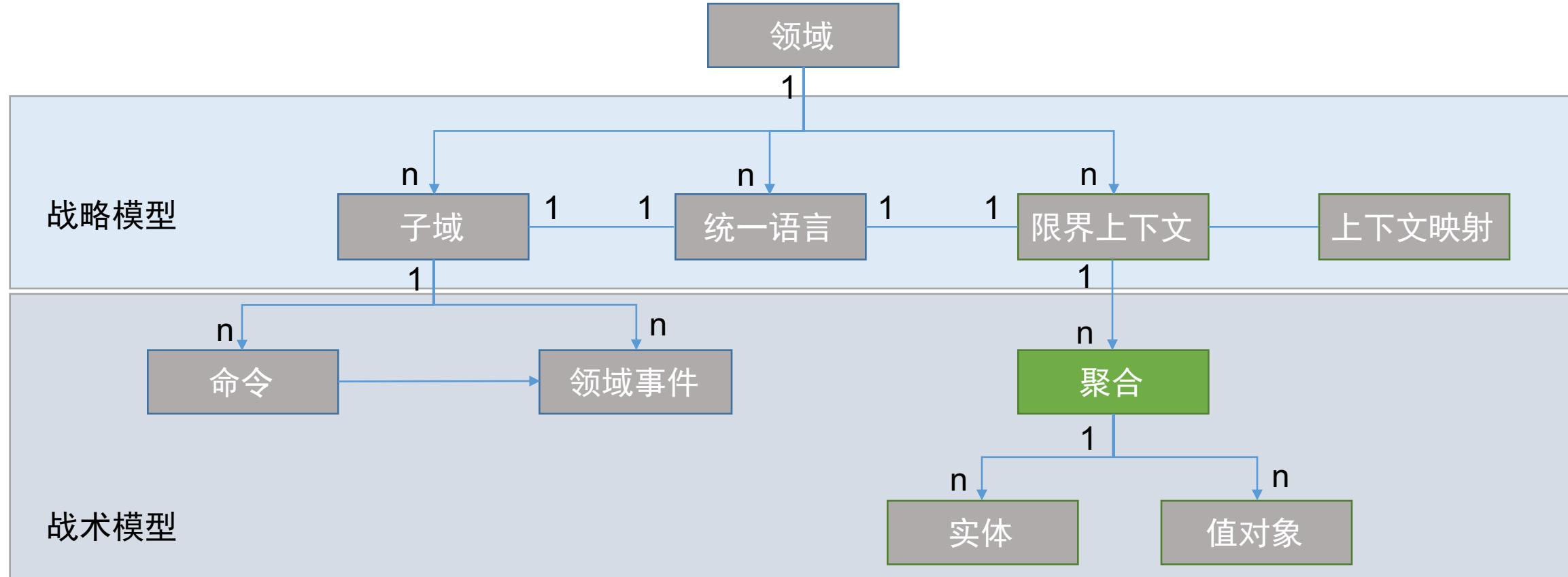
Money

- amount
- currency

手机子域

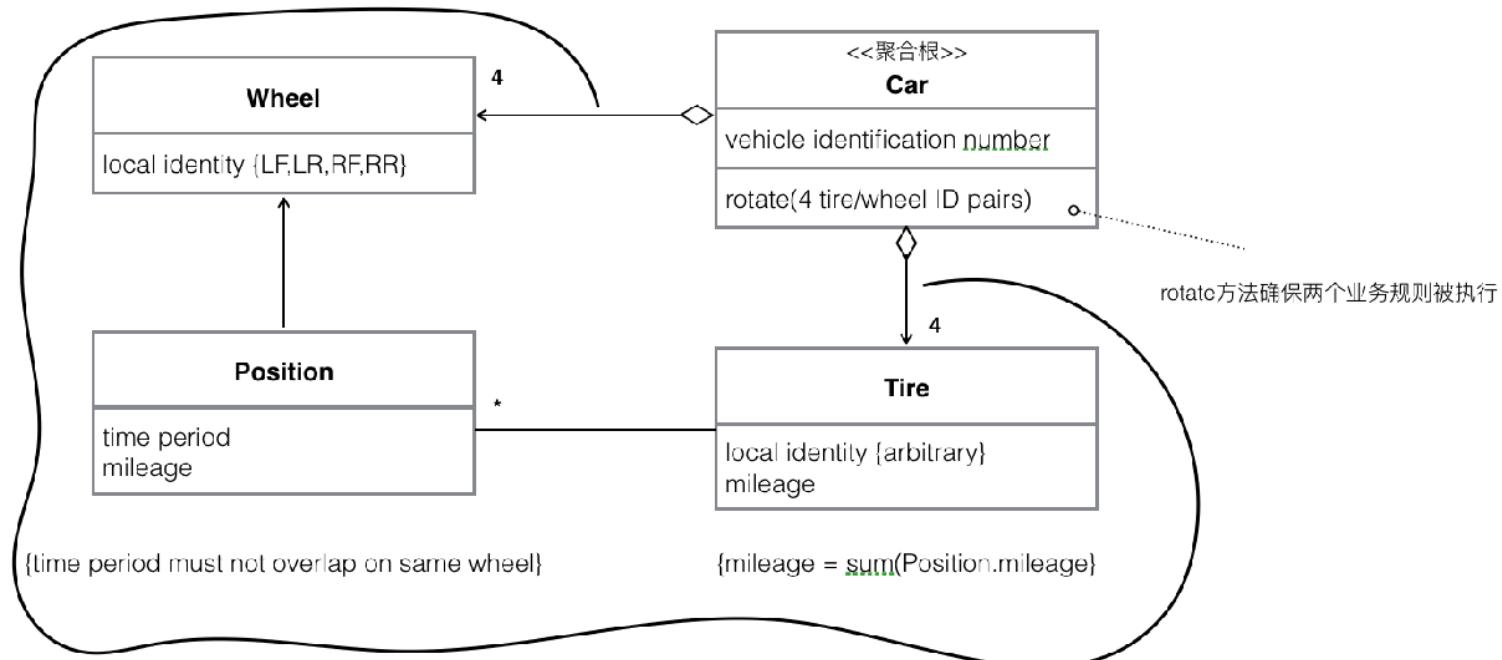
Color

- red
- green
- blue
- alpha



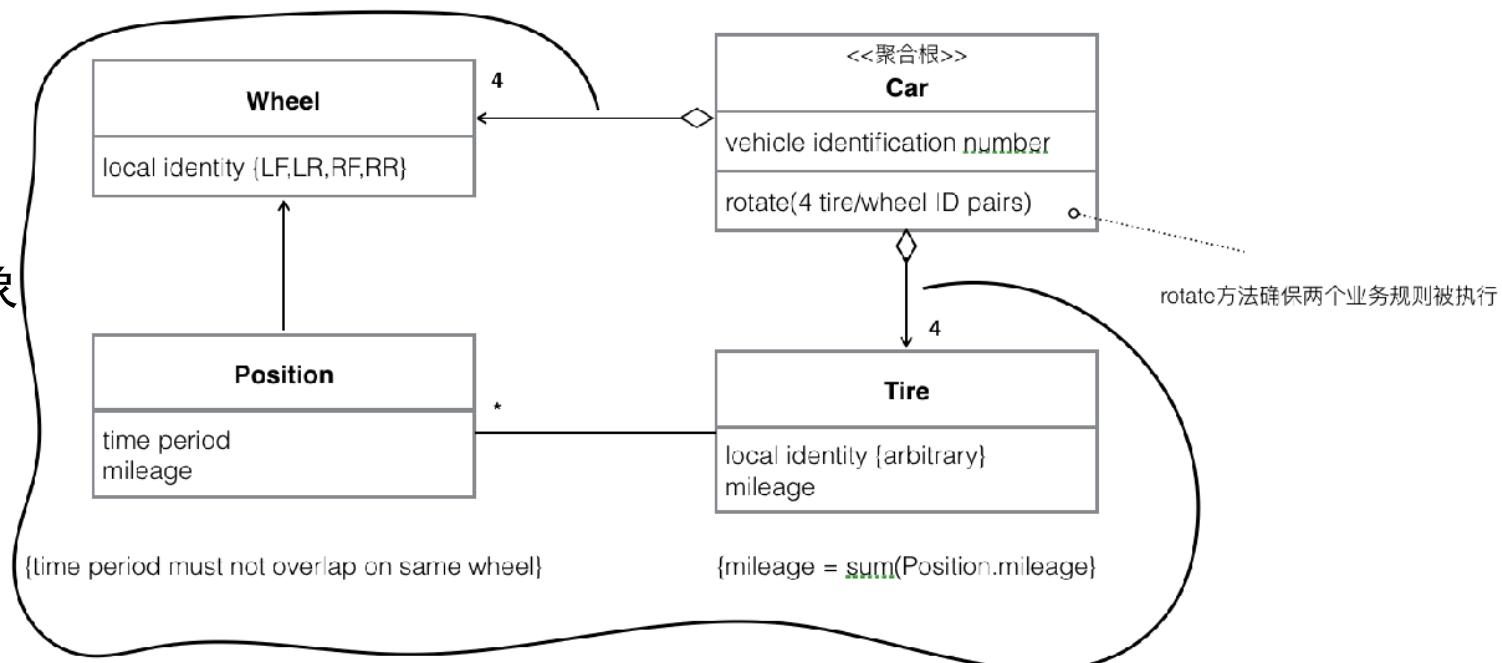
# 聚合(aggregate)

- 一组生命周期相关领域对象
- 确保业务规则在领域对象的各个生命周期得以执行
  - 聚合内保证业务不变性
- 与事务一一对应
  - 聚合内保证事务一致性
  - 聚合间保持最终一致性

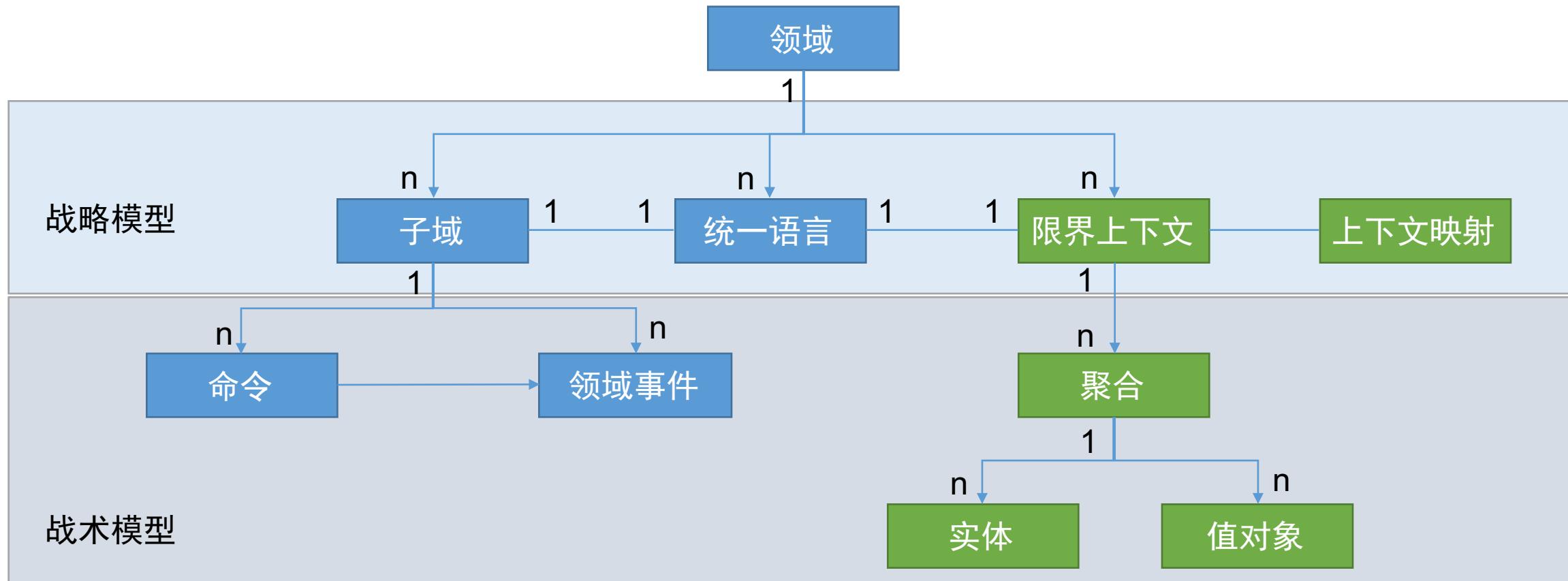


# 聚合根(aggregate root)

- 聚合对外暴露的实体
- 聚合根有全局标识
- 聚合内实体只能有局部标识
- 只能通过聚合根修改边界内对象
- 只能通过聚合根导航到边界内对象
- 简化复杂的对象网络



# DDD核心概念：145



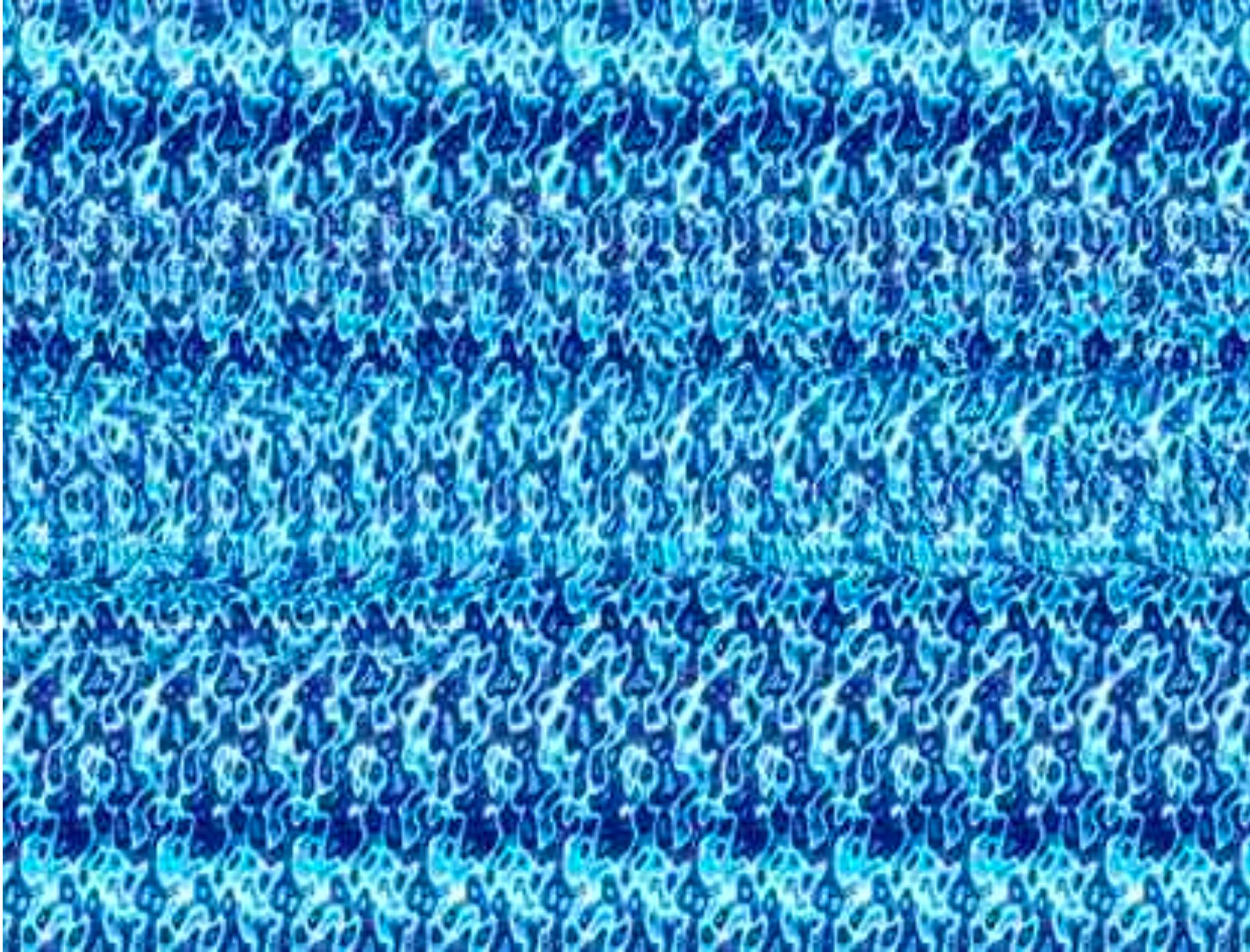
问题空间: 业务领域

解决方案空间: 软件模型

诸多概念，何从下手？

# 规律

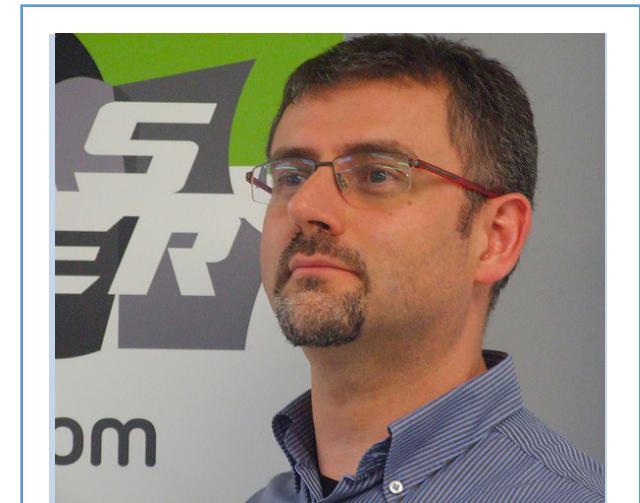
事件风暴



# 事件风暴 (Event Storming)



以workshop的方式  
快速探索复杂业务领域



由Alberto Brandolini发明，  
经过DDD社区和团队实践

# 事件风暴 (Event Storming)

投入

让带着问题的人和拥有答案的人共聚一堂，构建模型

---

强大

可以让实践者理解复杂业务模型的时间从数周缩短到几小时

---

高效

跟DDD的实现模型高度一致，并能快速发现聚合和限界上下文

---

简单

标记都很简单，没有复杂的UML

---

有趣

参与者乐在其中，从中的发现超乎预期

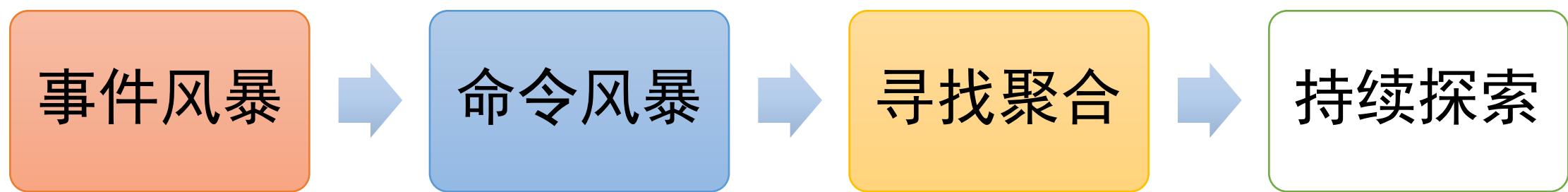
---

# 活动准备

- 正确的人
  - 领域专家
  - 架构师
  - 研发/测试人员等关键角色
- 开放空间
  - 有足够的空间将事件流可视化
  - 让大家能在白板前交互讨论
- 即时贴
  - 至少三种颜色

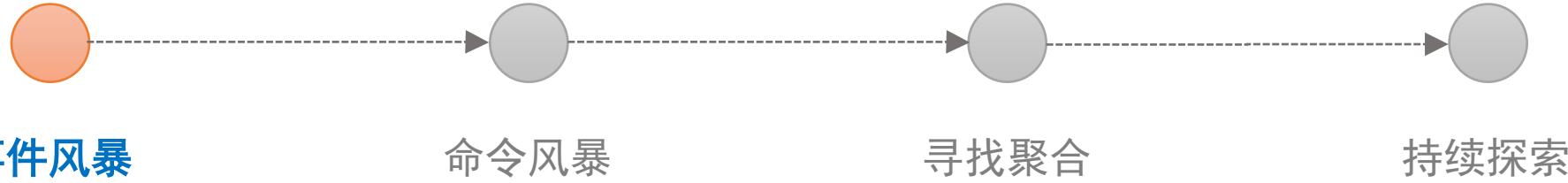


# 事件风暴步骤：橙蓝黄白

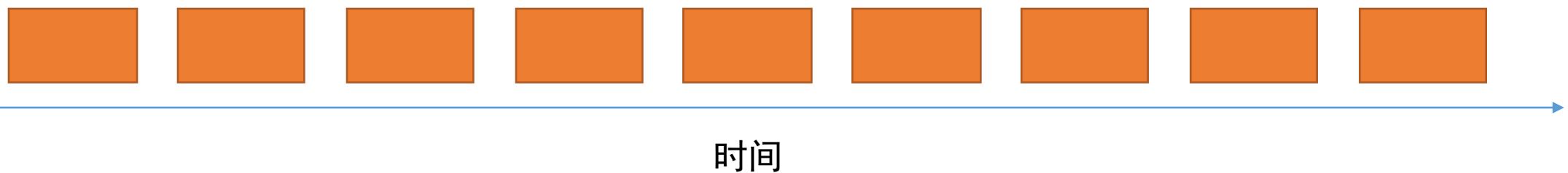


# 事件风暴

- 事件风暴
- 命令风暴
- 寻找聚合
- 持续探索

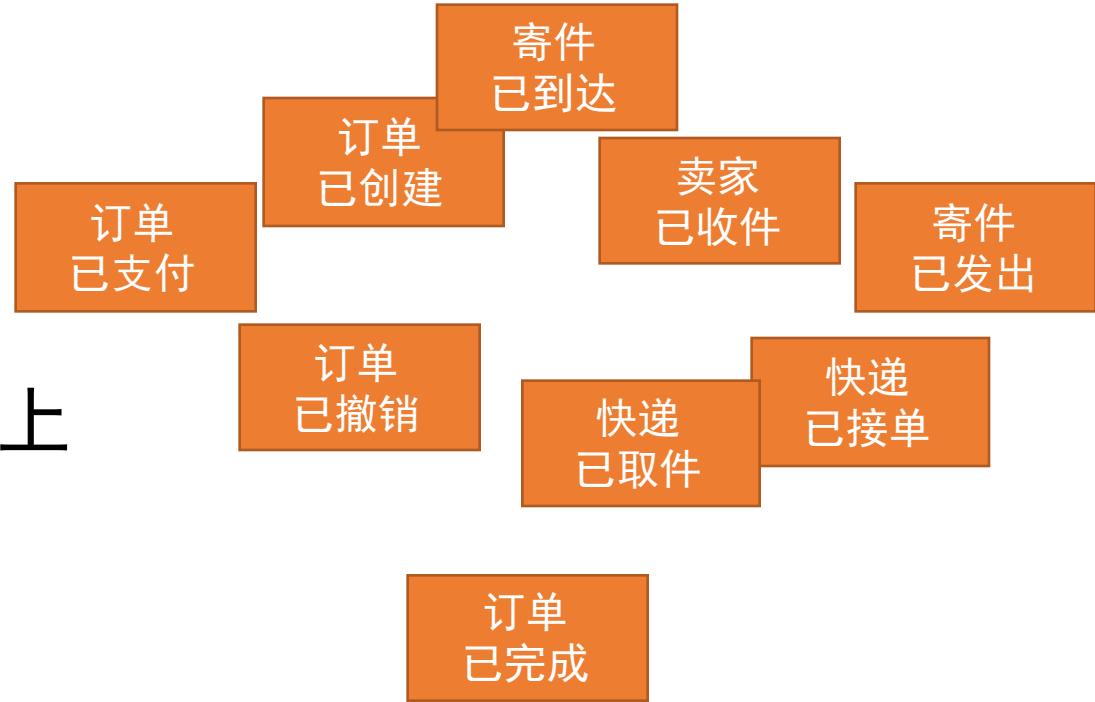


1. 头脑风暴出领域事件
2. 对领域事件进行排序



# 头脑风暴

- 领域专家介绍业务
- 参与者可任意提问
- 参与者将领域事件写在橙色即时贴上
- 每个即时贴一个事件
- 事件格式：“XX已XX”(订单已创建)



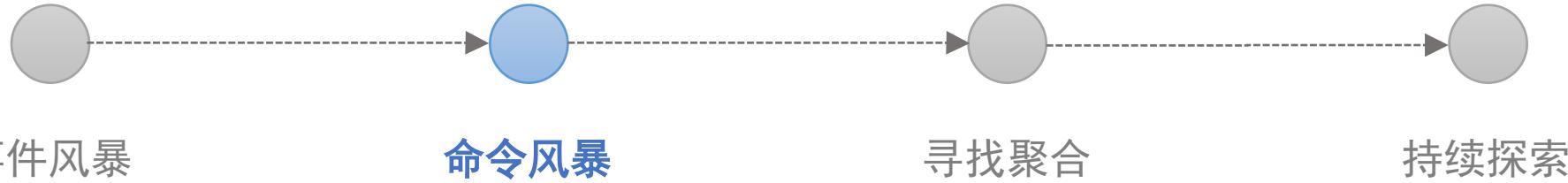
# 事件排序

- 参与者将自己的事件贴纸贴在白板上
- 事件从左到右按时间顺序排序
- 不同参与者需保证事件相对顺序



# 事件风暴

- 事件风暴
- 命令风暴
- 寻找聚合
- 持续探索



- 将命令写在蓝色即时贴上
- 将命令贴在所产生的事件左边
- 命令可能产生多个事件



# 事件风暴

- 事件风暴
- 命令风暴
- 寻找聚合
- 持续探索



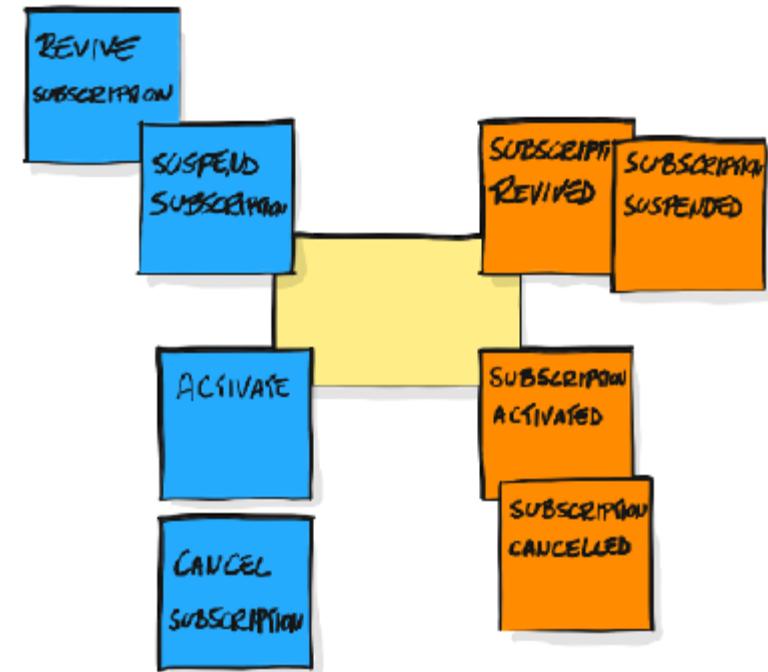
事件风暴

命令风暴

寻找聚合

持续探索

- 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
- 通过业务上的统一概念，识别出分布在时间轴不同位置的同一聚合
- 使用黄色即时贴标记聚合



聚合接受命令，产生事件



事件风暴

命令风暴

寻找聚合

持续探索





事件风暴

命令风暴

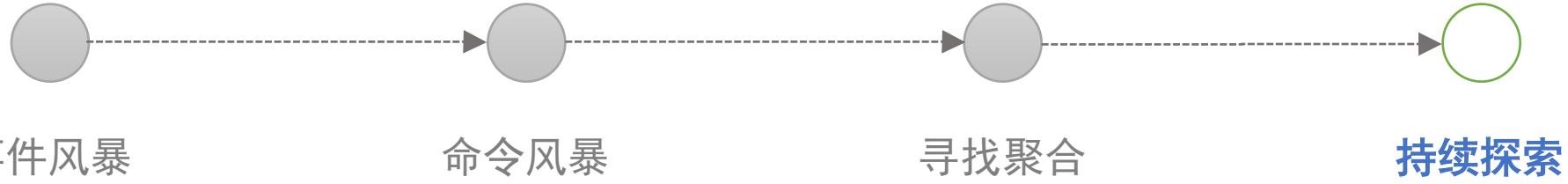
寻找聚合

持续探索



# 事件风暴

- 事件风暴
- 命令风暴
- 寻找聚合
- 持续探索



- 根据语义和业务关注点确定限界上下文
- 探索子域：实体、值对象
- 拟定关键验收测试
  - 行为驱动开发 (BDD: behavioral driven development)

# 事件风暴：规律

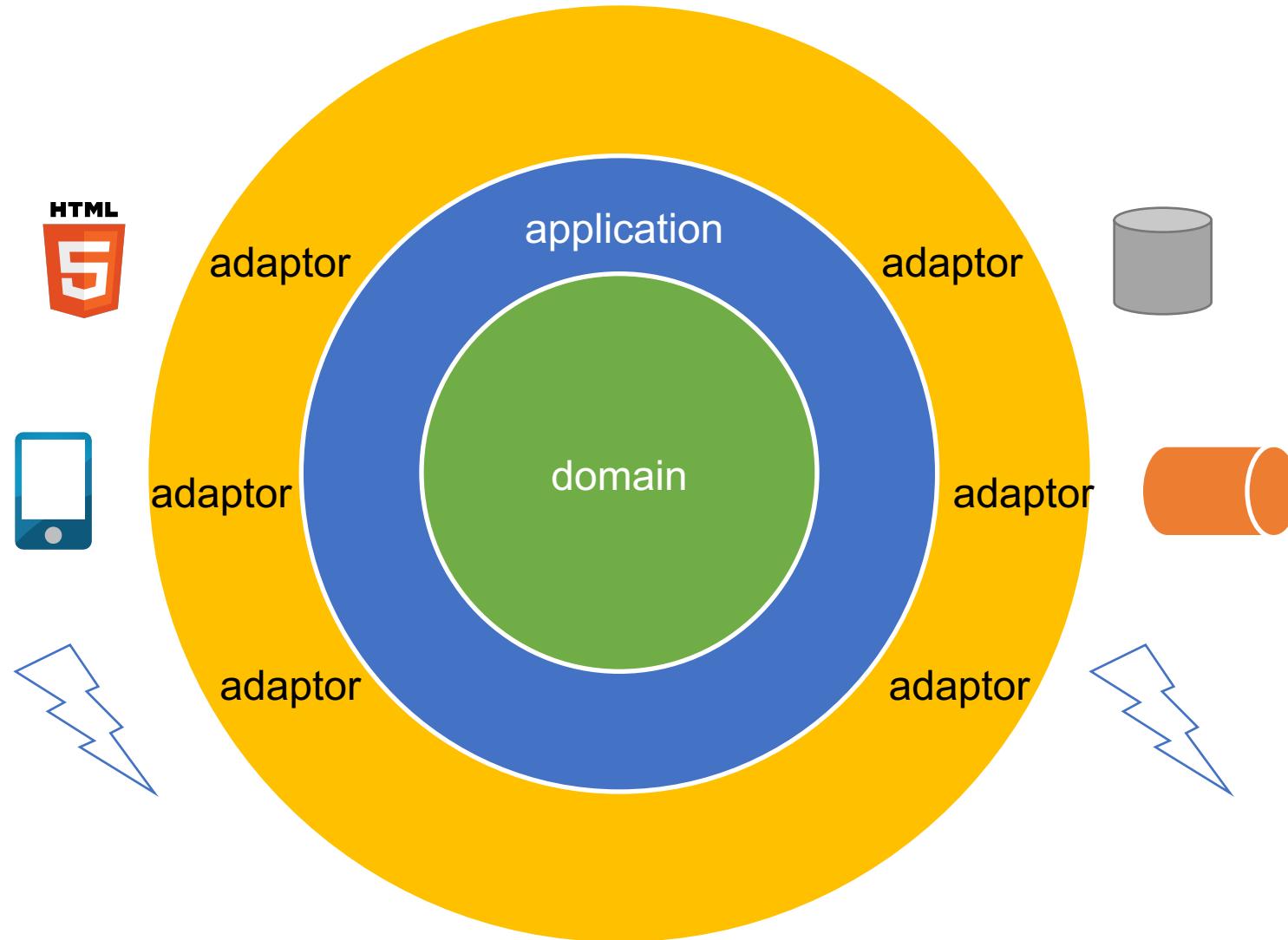
- 通过时间顺序梳理事件
- 通过事件反溯命令
- 通过事件和命令归纳聚合
- 通过语义和业务关注点确定限界上下文



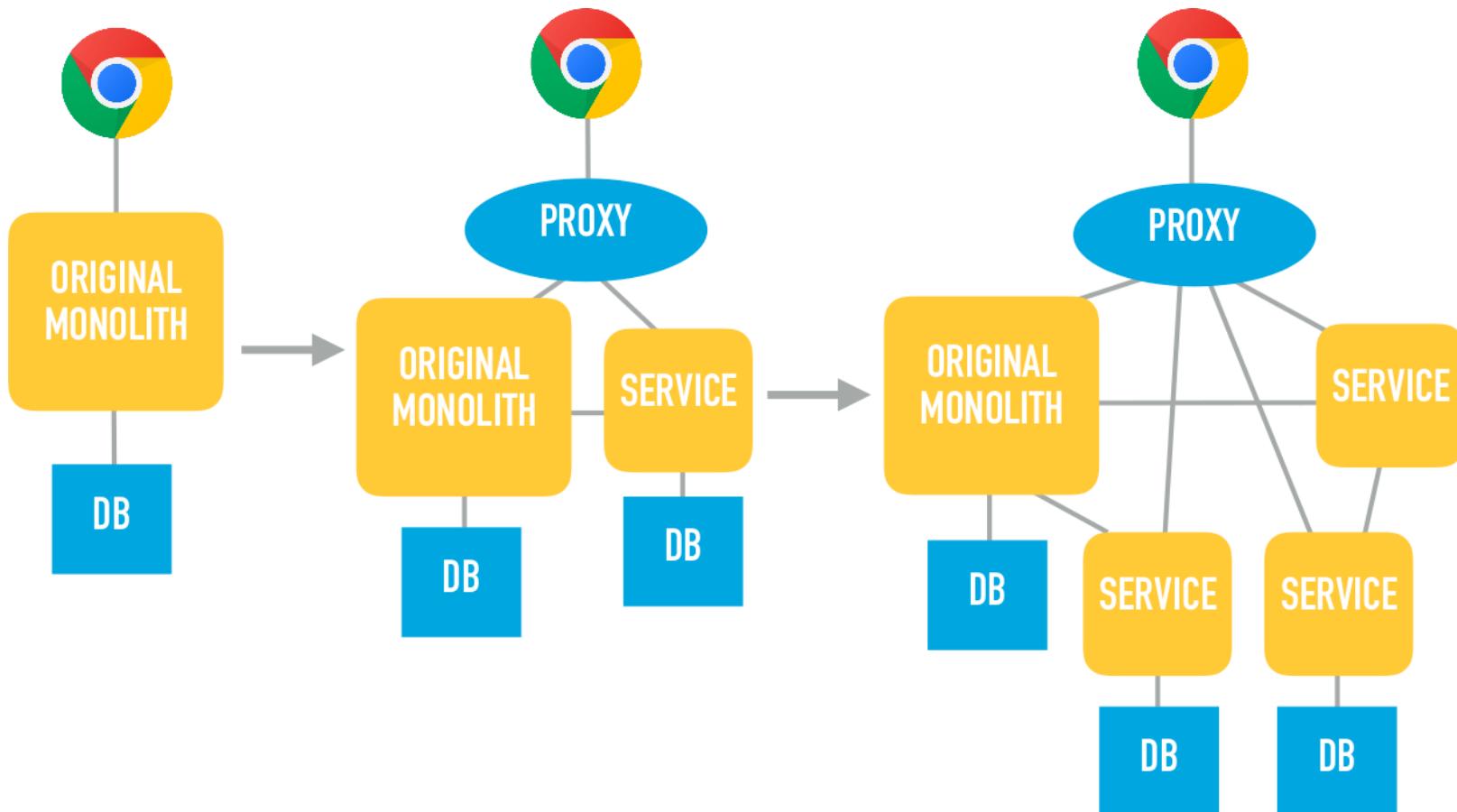
# 注意事项

- 所有人参与
- 不评价
- 将当前没有答案的问题列在玫红色贴纸上，贴在相关业务附近

# 基于DDD的服务实现



# 系统重构: 绞杀者模式



# Workshop

业务拆解建模实战

**Thank you!**