主要内容

美团

- 领域驱动设计概述
- 几个概念的理解
- 几个实践问题的探讨

为什么领域驱动设计

美团

领域驱动设计

- ✧一种思维方式：关注点从工具回归到问题域本身
- ✧一组优先任务：以业务为先导
- ✧一种软件开发方法：系统拆解和集成方法、建模方法、分层架构、实现工具集

如何领域驱动设计

美团

战略设计

战术设计

从整体到局部

战略设计-统一语言

美团

- ✧**定义**：提炼领域知识的产出物，体现在两个方面，1)统一的领域术语，2)领域行为描述
- ✧**如何获取**：统一语言就是需求分析的过程，也是团队中各个角色就系统目标、范围与具体功能达成一致的过程
- ✧**强调统一**：无论是与领域专家的讨论，还是最终的实现代码，都使用相同的术语
- ✧**强调约束**：既要有内涵也要有外延

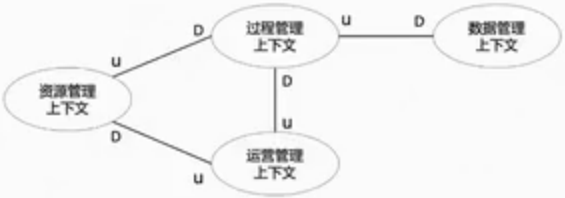
概念	英文	定义	约束	举例
资源	Resource	SO进行销售活动的对象		POI
公海	PublicSea	业务目标圈定的某类资源全集	和业务目标一一对应，比如交易公海是为了做交易的，推广公海是为了推广的。	业务目标是团购买单合作，圈定POI为到餐全国和综合52特的
战区类型	TheatreType	资源划分方式	依据资源具有属性划分	
战区	Theatre	为业务组织划分的资源范围	战区是对公海的划分，圈定一个业务组织单元的业务范围，该范围对其他业务组织单元具有排他性，战区之间是并列关系，且互不重叠，一个战区对应一个组织单元	orgunit：虚拟销售管理行政单元
业务组织	BizOrg	针对销售管理设置的组织结构	组织的划分依赖战区的划分	Glorg代替实现
私海	PrivateSea	在公海内个人圈定的资源集合	公海范围的排他性	普通私海

战略设计-子域和限界上下文

美团

总体设计思路：分治，避免大泥球

- ✧子域划分：确定逻辑边界
  - ✧核心域：重点关注，重点资源投入
  - ✧支撑域：专注于业务的某个方面
  - ✧通用域：用于整个业务系统
- ✧限界上下文：确定物理边界
  - ✧边界内概念含义统一
  - ✧不同边界内概念的关注点不同
  - ✧包含领域模型&系统实现
- ✧上下文映射图：集成
  - ✧九种映射关系

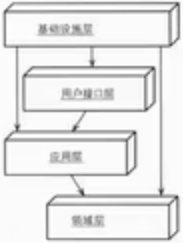


战术设计-设计过程

美团

总体设计思路：面向对象

概念建模 → 领域建模 → 框架设计

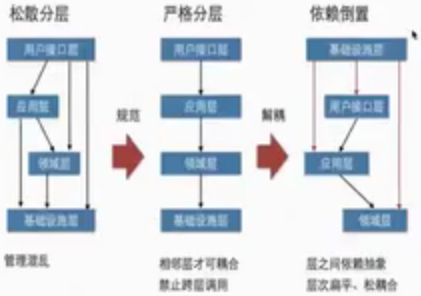


问题、业务、抽象 → 解决、技术、具体

战术设计-框架设计

美团

总体设计思路：分层、CQRS、EDA



主要内容

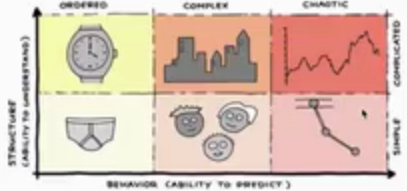
美团

- 领域驱动设计概述
- 几个概念的理解
- 几个实践问题的探讨

复杂度

美团

Jurgen Appelo 从理解力与预测能力两个维度分析了复杂系统理论



复杂的成因	复杂的控制
规模	分解
结构	层次结构
变化	抽象

复杂度

美团

复杂的控制方法在领域驱动设计中的体现

领域&领域驱动-为什么是“领域驱动”

美团

- 领域

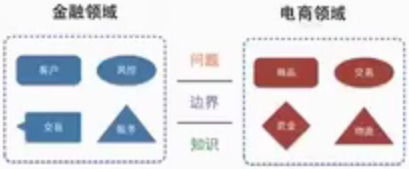
✧含糊定义：一个组织所做的事情以及其中所包含的一切（和“业务”的区别？）

✧问题+边界+知识

✧问题：具体业务目标

✧边界：范围的

✧知识：行业的，固定模式，内在规律



- 领域驱动

关注点由技术转向业务（问题）

以业务为先导（问题）

业务角色参与设计（问题）

统一语言（知识）

划分子域（知识）

限界上下文（边界）

面向对象建模（问题、知识）

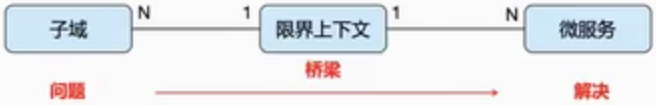
以上设计原则是对领域的回应，所以是领域驱动设计

子域&限界上下文&微服务

美团

限界上下文：不在于如何划分边界，而在于如何控制边界

- 领域逻辑层面：限界上下文确定了领域模型的业务边界
- 团队合作层面：限界上下文确定了开发团队的工作边界，建立了团队之间的合作模式
- 技术实现层面：限界上下文确定了系统架构的应用边界，保证了系统层和上下文领域层各自的一致性



✧限界上下文代表语义边界，所以逻辑层面可能多个子域在一个限界上下文内

✧从业务视角看通常一个限界上下文对应一个微服务，但考虑技术因素情况下可能对应多个服务

✧考虑高性能和稳定，一般将核心流程和非核心流程分离：query服务、offline服务、job服务

模型

美团

- ✧模型：抽象知识的表达
  - ✧抽象：非具体细节的
  - ✧知识：基本规律
  - ✧表达：描述和传达知识
- ✧模型长什么样子
  - ✧不同的描述目的和描述对象会有不同模型
  - ✧可以UML图、类图、DDD领域建模语言、代码与文档

✧领域模型是领域驱动设计的核心，也是从问题到解决的桥梁：认识-传达-指导开发


```
public class LoanBill {
    private RepayPlan repayPlan;
    private List<PeriodBill> periodBills;
}

public class RepayPlan implements Serializable {
    /**
     * 还款计划项
     */
    private List<RepayPlanItem> repayPlanItems;

    public RepayPlan(List<RepayPlanItem> repayPlanItems) {
        this.repayPlanItems = repayPlanItems;
    }
}
```

战术设计工具集

美团

✧值对象

- ✧可度量、不变性、相等性
- ✧直观表现是没有id
- ✧通常为度量单位、枚举等，比如币种、行政区等级

✧领域服务

- ✧是服务，不适合放在单个聚合和对象上的操作
- ✧是领域的，非应用服务
- ✧通常一个领域的计算/转换过程，或由多个领域对象协作产生的能力
- ✧比如利率计息算法、逾期等级计算方法

✧聚合根

- ✧满足对象操作的业务一致性规则

主要内容

美团

- 领域驱动设计概述
- 几个概念的理解
- 几个实践问题的探讨

划分领域

美团

✧业务本源
- ✧依据于业务的本质
- ✧根据目标、价值、维度划分

✧参考行业
- ✧相对成熟的领域：电商、金融、HR、财务……

✧参考业务对象（端）
- ✧业务的直接表象
- ✧B端：CRM、SCP……
- ✧C端：交易、履约……
- ✧M端：运营、财务……

✧参考业务组织
- ✧康威定律

✧一个常见的模式：纵+横
- ✧纵：根据业务目标和价值垂直划分
- ✧横：共性能力提取，划分通用域和支撑域

领域服务 & 应用服务

美团

✧战略层语境：

✧领域服务通常指相对聚焦的底层支撑域/通用域服务

✧应用服务通常指面向业务场景负责功能组装的服务

✧战术层语境：

✧领域服务指领域建模工具集中所指的“领域服务”

✧应用服务指面向场景的技术实现组装

✧在战术层语境，应用和领域仅仅是技术和业务的分离吗？能否参考战略层语境？

领域服务 & 应用服务

美团

信贷领域记账的场景例子

全额还款

- 还款金额试算和校验
- 隔日记账校验减免
- 计划缩期
- 还款冲抵
- 生成动账
- 生成状态转移
- 结清借据

财务还款

- 减免输入利息
- 减免输入罚息
- 还款冲抵
- 生成动账
- 生成状态转移
- 结清借据

逻辑的场景相关性

全额还款

- 还款金额试算和校验
- 隔日记账校验减免

财务还款

- 减免输入利息
- 减免输入罚息

应用服务

- 场景相关的
- 相对易变的

- 计划缩期
- 生成动账
- 结清借据

- 还款冲抵
- 生成状态转移

领域服务（行为）

- 场景无关的
- 相对稳固的
- 可复用的

✧战术层语境，应用服务可否包含面向场景的业务逻辑的组装？

领域层再分

美团

✧场景：多业务线&多产品，业务共性较多，但也存在部分差异

CRM领域不同业务线的资源管理

业务线	业务线A	业务线B	业务线C
资源范围	品类A全国	品类B31城	全品类全国
划分方式	地理单元划分	行政城市方式划分	自定义
管理单元	地理单元	城市（北京朝阳）	地理单元
组织	业务线A销售组织	业务线B销售组织	业务线A+C销售组织
组织和管理单元	组织最底层节点对应管理单元	组织最底层节点对应管理单元	组织最底层节点对应管理单元
私海	BD所属资源，唯一	BD所属资源，唯一	BD所属资源，唯一

信贷领域各类产品

领域层再分

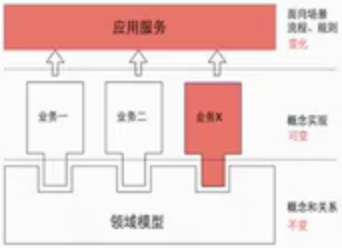
美团

✧方案：平台化思路，避免垂直烟囱，提人效降成本

✧问题：多业务/多产品带来领域的复杂

①抽象建模：抽象共性，泛化概念

②分离变化：分离抽象和具体



③分层的演进：领域实现层

富血对象落地的问题

美团

① 富血对象和spring依赖注入
——BeanHolder静态方法
——难以做到去spring

②大聚合根和性能问题
——懒加载+SPI模式
——难以做到去spring

③领域对象存储性能问题：增 or 删 or 改？
——根据业务逻辑判断，有侵入
——通过领域对象修改前后的比对判断

```
public class AccountLeafIdFactory {

    private static final Logger LOGGER = Log

    /**
     * 不做注入，静态方法调
     * 保证唯一
     */
    public static long genPeriodAllId(){
        int retryTime = 1;
        while (retryTime <= ZOOKE_RETRY_TIM
            try {
                Result result = staticLeaf.g
                if (result == null || !Statu
                    continue;
                }
                return result.getId();
            } catch (Exception e) {
                LOGGER.info(e.getMessage(),
                if (retryTime == ZOOKE_RETR
                    throw new LeafIdGenerate
                }
            } finally {
                retryTime++;
            }
        }
        throw new LeafIdGenerateException("L
    }
```

```
protected LocalDate getCurrentOrgDate() {
    Account account = getAccount();
    if (account == null) {
        throw new IllegalStateException("借据 [" + getId() + "] 获取当前机构日期异常，原因：Acc
    }
    OrganizationNo orgNo = account.getOrgNo();
    if (orgNo == null) {
        throw new IllegalStateException("产品 [" + account.getProductNo().getName() + "] 借
    }
    Organization organization = RepositoryHolder.getOrganizationRepository().get(orgNo);
    if (organization == null) {
        throw new IllegalStateException("产品 [" + account.getProductNo().getName() + "] 借
    }

    LocalDate currentOrgDate = organization.getCurrentOrgDate();
    if (currentOrgDate == null) {
        throw new CurrentOrgDateNotExistException("产品 [" + account.getProductNo().getName
    }
    return currentOrgDate;
}
```

DDD对clean code的再定义

美团

类别	规范	等级
统一语言	与统一语言（英文）一致的代码命名	强制
domain层	Domain仅包含领域模型定义的对象，且用plain object	强制
	Domain层不依赖spring的AOP和IOC等三方包	推荐
	Domain对象拒绝Getter、Setter、Constructor等注解	强制
	Domain对象行为拒绝setter、update、modify、save、delete等无明确业务含义的方法	强制
	值对象命名不用加上标识技术语言的Enum	强制
application层	application层拒绝XXXHandler、XXXProcessor、XXXContext等含义不明确的命名	强制
	区分命令和查询，命令推荐XXXCommandService，查询推荐XXXQueryService	推荐
infrastructure层	Repository的入参和出参除了原始数据类型，只能包含领域对象	强制
	Repository对外交互拒绝DTO、PO	强制
	对外接口访问的防腐层，统一命名为XXXAdaptor	建议
	禁止外部接口对象向上层透传	强制
事件	事件命名为事件+Event，且事件命名为动词过去时	强制

参考资料

美团

[1]《领域驱动设计》 Eric Evans

[2]《实现领域驱动设计》Vaughn Vernon

[3]《领域驱动设计实践（战略+战术）》 张逸

[4]《面向对象分析与设计》 Grady Booth，Kobert A. Maksimchuk等

[5]《大象 Thinking in UML》 谭云杰

[6] https://ipu.sankuai.com/ipu/courseware-detail/44800/3071

[7] https://ipu.sankuai.com/ipu/courseware-detail/47340/4407

美团

Q&A

更多技术干货
欢迎关注“美团技术团队”

招聘：

美团金融-资深Java技术专家

美团金融-支付架构师

邮箱：cuiyuanyuan02@meituan.com

美团

CODE A BETTER LIFE
一行代码 亿万生活


更多技术干货
欢迎关注“美团技术团队”
招聘：
美团金融-资深Java技术专家
美团金融-支付架构师
邮箱：cuiyuanyuan02@meituan.com
美团
CODE A BETTER LIFE


系统通道技术沙龙-嘉宾群
【后台开发&系统开发】岗位
正在热招
可以放在一页上吗
第一个议题问答：
1. 语问 红线 黑线的区别是什么呀
2. 可以结案例，把领域驱动设计DDD里的隐喻概念映射起来吗？
3. 能否把案由对象下面两个例子结合起来上面的理论讲一讲
4. 把DDD带入团队的关键步骤有那些

主会场

何正-美团技术专家

王吕松-美团技术专家

吴仁润-美团技术专家

主会场

何正-美团技术专家

王吕松-美团技术专家

吴仁润-美团技术专家

主会场

何正-美团技术专家

王吕松-美团技术专家

吴仁润-美团技术专家