

## 核心域 core domain

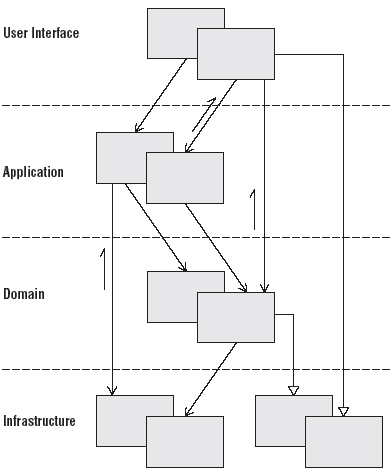
## 通用域 common domain

## 支撑域 support domain

## 统一语言Ubiquitous Language

一个团队使用一种语言，如果同一个对象表述为不同的名称，那这两个不同名称属于两个不同的子域。

## 分层与架构Layered Architecture



用户接口

领域层

防腐层

应用层

基础设施层

PO层

clean架构

六边形架构

\*用户界面层（或表示层）负责向用户显示信息和解释用户指令

这里的用户可以是另一个计算机系统，不一定是使用用户界面的人

\*应用层 定义软件要完成的任务，并且指挥表达领域概念的对象来解决问题。

这一层所负责的工作对业务来说意义重大，

也是与其他系统的应用层进行交互的必要渠道。

应用层要尽量简单，不包含业务规则或者知识，只为下一层中的领域对象协调任务，

分配工作使他们互相协作。它没有反映业务情况的状态，但可以具有另外一种状态，

为用户或程序显示某个任务的进度

\*领域层（或模型层） 负责表达业务概念，业务状态信息以及业务规则。

尽管保存业务状态的技术细节是由基础设施层实现的，

但是反映业务情况的状态是由本层控制并且使用的。

领域层是业务软件的核心

\*基础设施层 为上面各层提供通用的技术能力：为应用层传递消息，

为领域层提供持久化机制，

为用户界面层绘制屏幕组件，等等。

基础设施层还能够通过架构框架来支持4个层次间的交互模式

PO层将领域对象进行持久化

## 对象Object

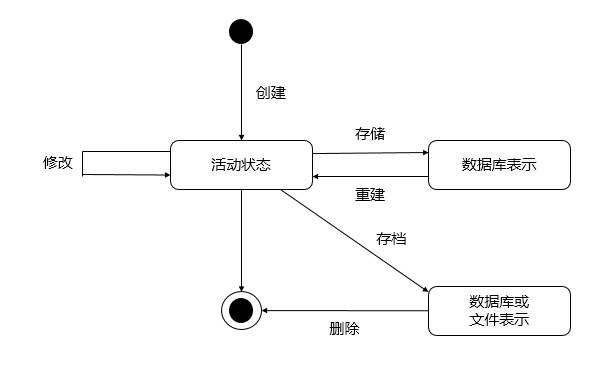
VO

View object 视图层对象，展示层对象（页面传或接）

DO

Domain object 领域对象（表达业务）

领域对象生命周期



PO

Persistent object 持久化对象（持久到存储）

## 聚合根Aggregate Root

聚合根具有全局唯一标识

聚合根之间通过id来引用

聚合根设计的要尽量小

一个聚合根配一个仓储

聚合根到聚合根通过ID关联

聚合根到实体直接对象引用

聚合根到值对象直接对象引用

## 仓储Repositories

仓储接口总接收聚合根或返回聚合根

帮助管理领域对象生命周期的中间和结束

仓储只提供三个基础接口

根据唯一标识查找

getById

保存或更新

save

移除

Remove

## 工作单元UnitOfWork

工作单元是一种数据访问模式，他是用来维护一个已经被业务修改（如增加，删除和更新）的业务对象组成的列表。它负责协调这些业务对象的持久化工作及并发问题。

工作单元可以理解为带事务控制的一个方法，在这个方法中有对一张或多张表的操作。全部操作成功时事务自动提交，只要有一张表处理失败或有异常发生时事务回滚。

也就是类似于在以前MVC模式中带@Transactional的注解的方法。

管理数据库连接和事务

跟踪实体更改，并把更改保存到数据库

被定义在领域层，但在基础设施层实现

## 规范与说明Specification

提供仓储扩展查询

## 工厂Factory

负责创建领域对象

Customer tom=new Customer();

Order order=OrderFactory.CreateOrder(tom);

Assert.IsNotNull(order);

FACTORY与REPOSITORY的关系是：FACTORY负责处理对象生命周期的开始，而REPOSITORY帮助管理生命周期的中间和结束。从领域驱动设计的角度来看，FACTORY和REPOSITORY具有完全不同的职责。FACTORY负责制造新对象，而REPOSITORY负责查找已有对象。

## 值对象ValueObject

没有唯一标识，起到描述作用

## 实体Entity

具有生命周期

只在聚合根内有本地唯一标识（这个唯一标识不能是基本数据类型）

可以放业务逻辑比如判断之类的

## 领域服务DomainService

可以放业务逻辑比如判断之类的

## 模块Module

## 应用服务ApplicationService

只包含工作流控制，没有业务逻辑（如判断）

## 基础设施服务infrastructureService

## 领域事件

领域服务或领域实体一起实现不属于单个实体的业务规则