领域驱动设计的价值和挑战

架构篇-张晓宁

- 微观架构设计为了解决什么问题
- 分析几种架构模型
- 丽呈架构模型如何设计
- 展示几种不同抽象程度的代码实现

微观架构设计为了解决什么问题

- 商业价值是软件系统的核心价值，如何隔离底层技术细节，让开发这更聚焦业务

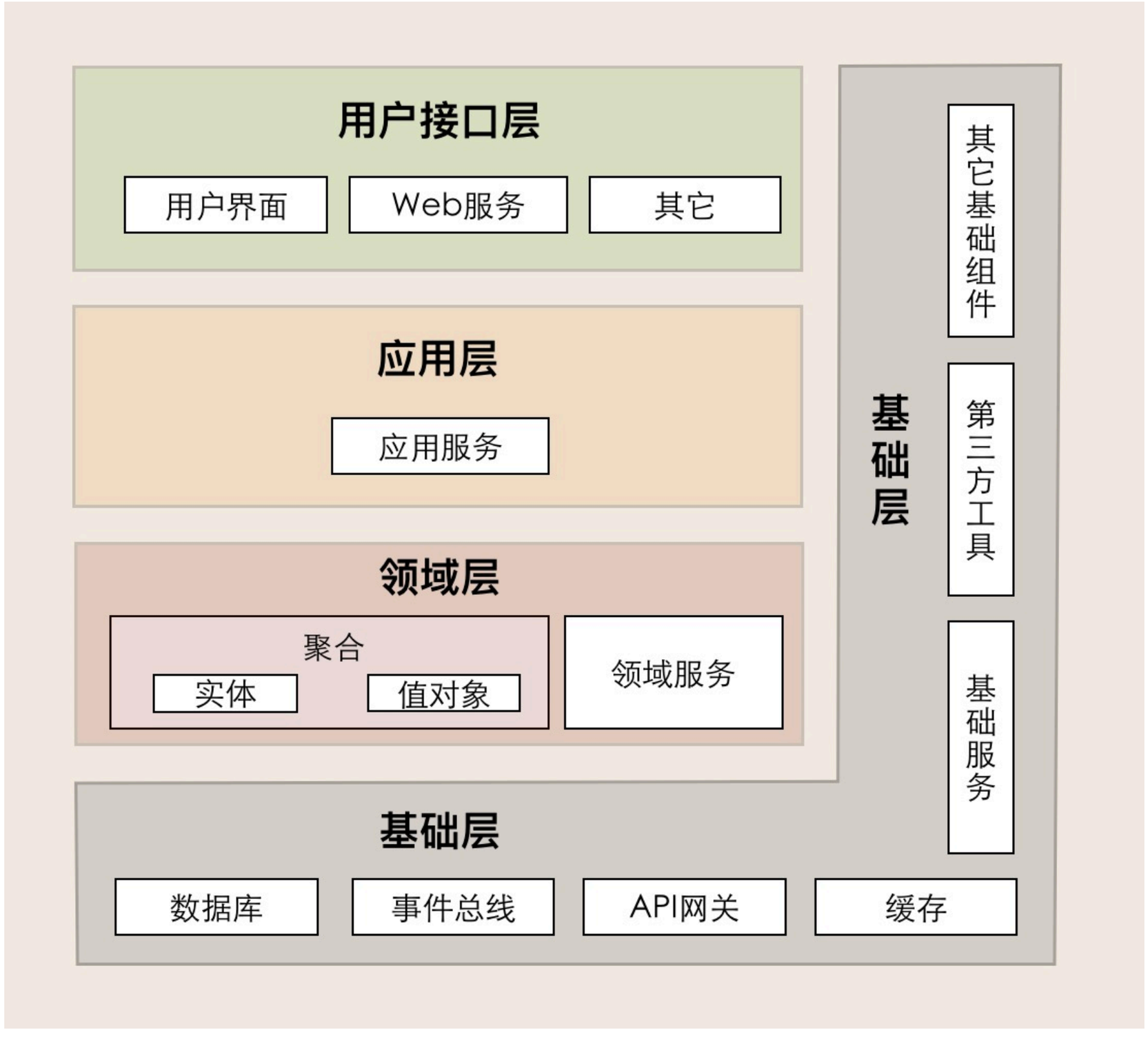
- 一个企业的核心能力应该是高度稳定的，架构也是如此，由于软件系统的易变性，如何隔离稳定态和敏捷态

- 体量决定架构，那么随着体量的增长，如何让系统可以自然演进，从创新业务快到核心业务稳的转换

分析几种架构模式

- DDD简单分层架构
- DDD洋葱架构
- DDD六边形架构
- CQRS模式
- 三种架构对比

DDD简单分层架构


用户接口层
用户界面
Web服务
其它
应用层
应用服务
领域层
聚合
实体
值对象
领域服务
基础层
数据库
事件总线
API网关
缓存
基础层
其它基础组件
第三方工具
基础服务

DDD简单分层架构

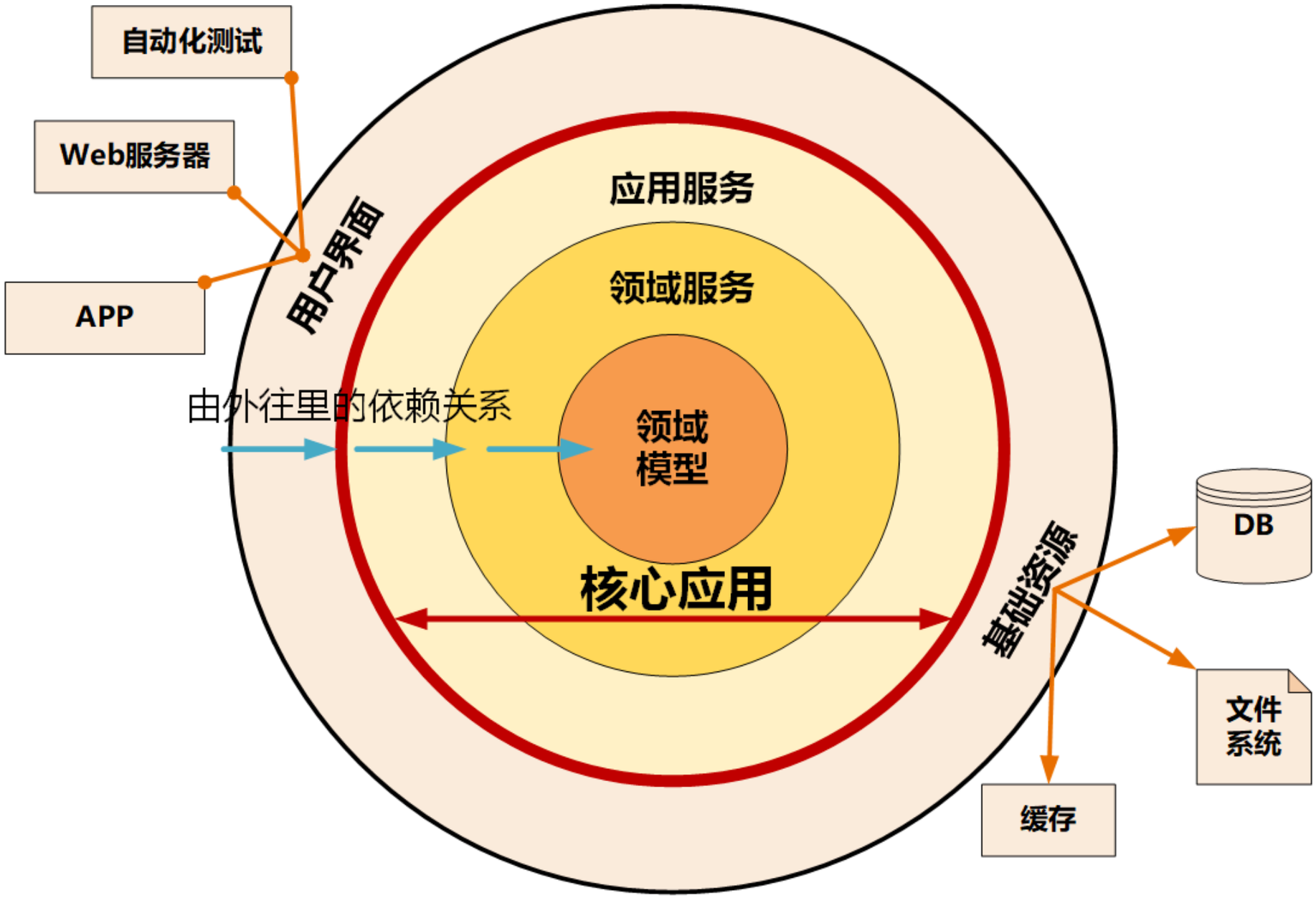
- 用户接口层：与用户按双方约定好的协议进行通信，这里的用户可以是UI、程序、自动化测试、批处理脚本等

- 应用层：是真正对一个完整业务用例的编排，但不实现具体的业务逻辑。可以认为是微观的解决方案，更关心业务场景

- 领域层：软件系统的核心价值所在，实现高内聚的业务逻辑。可以认为是微观上的业务功能，更关心业务逻辑

- 基础层：贯穿所有层，提供网络通信、数据持久化等基础技术服务，这层完成所有技术细节

DDD洋葱架构


自动化测试
Web服务器
APP
用户界面
应用服务
领域服务
领域
模型
核心应用
由外往里的依赖关系
基础资源
DB
文件
系统
缓存

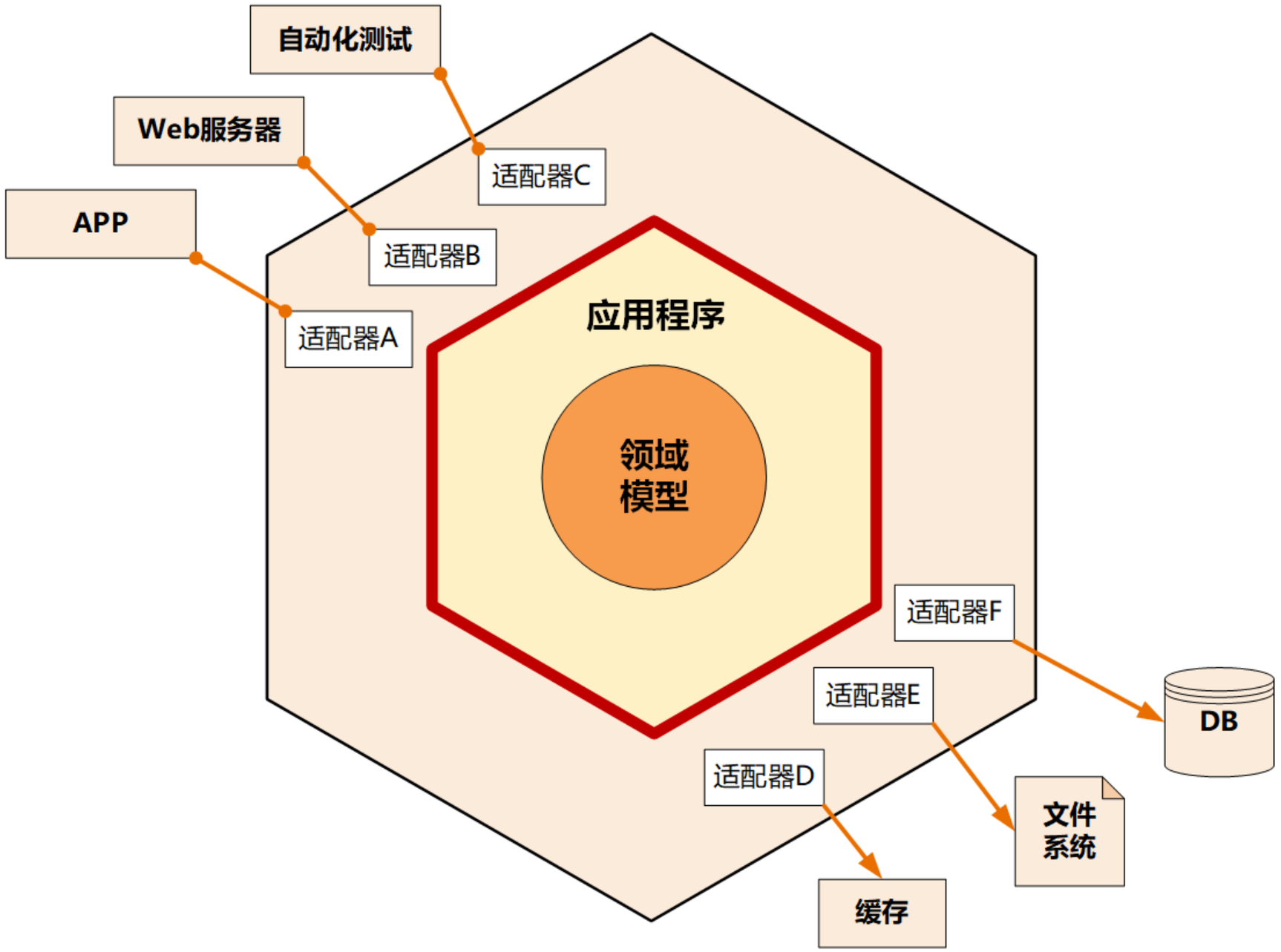
DDD洋葱架构

- 主要强调依赖关系，外层只能依赖与之相邻的内层，且内层不可依赖外层

- 作为严格分层架构的同时，更进一步的将业务代码模型和技术细节分离

- 可以得到一个高度抽象的业务代码模型，同时也对编码工作造成了很大的局限性

DDD六边形架构

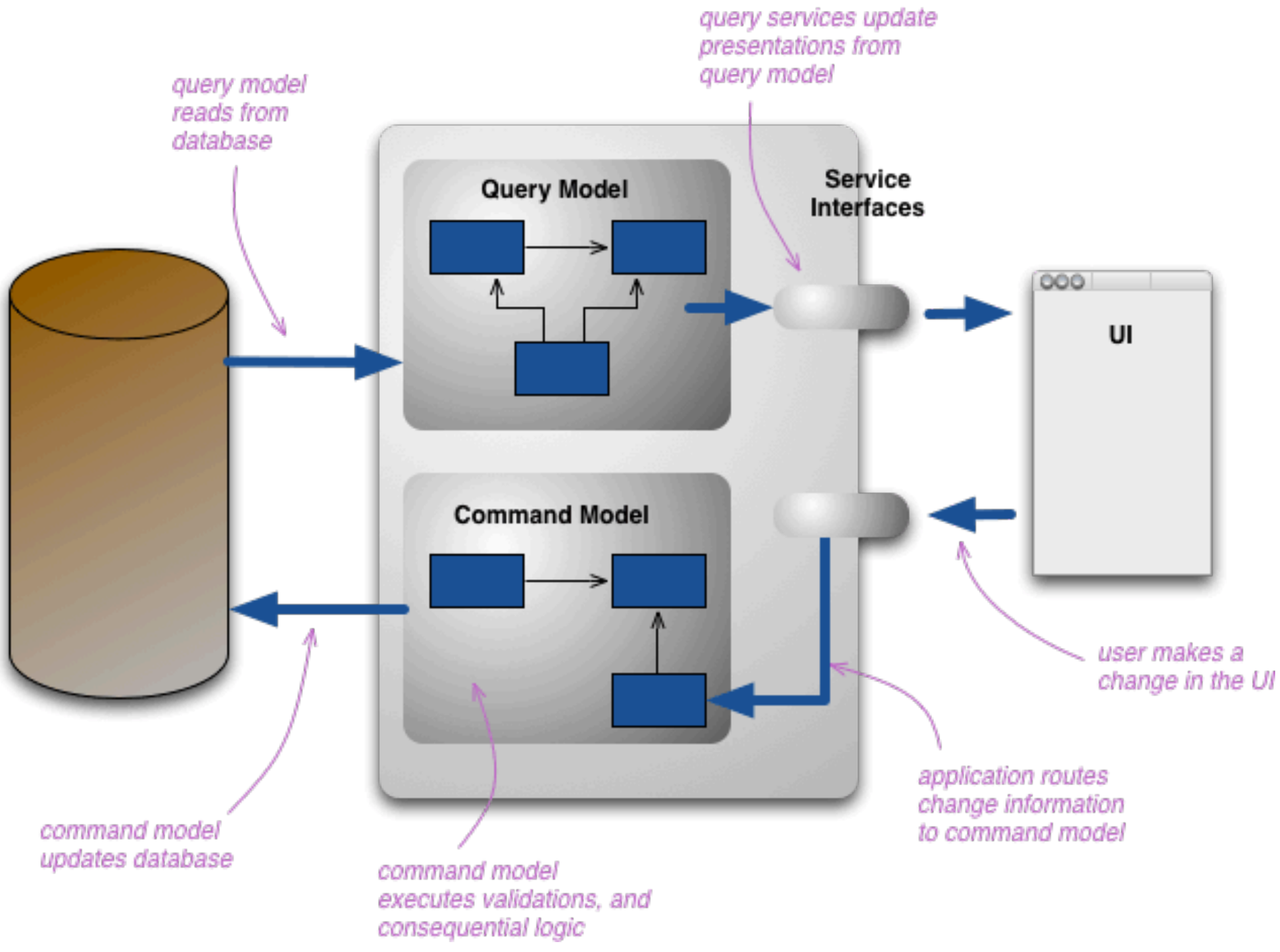

自动化测试
Web服务器
APP
适配器C
适配器B
适配器A
应用程序
领域
模型
适配器F
适配器E
适配器D
DB
文件
系统
缓存

DDD六边形架构

- 又叫端口适配器架构，强调的是通过适配器，将业务逻辑和技术细节隔离

- 可以认为是洋葱架构的超集，是比洋葱架构更高的业务抽象。

CQRS模式


query model
reads from
database
query services update
presentations from
query model
Query Model
Service
Interfaces
UI
Command Model
user makes a
change in the UI
command model
updates database
command model
executes validations, and
consequential logic
application routes
change information
to command model

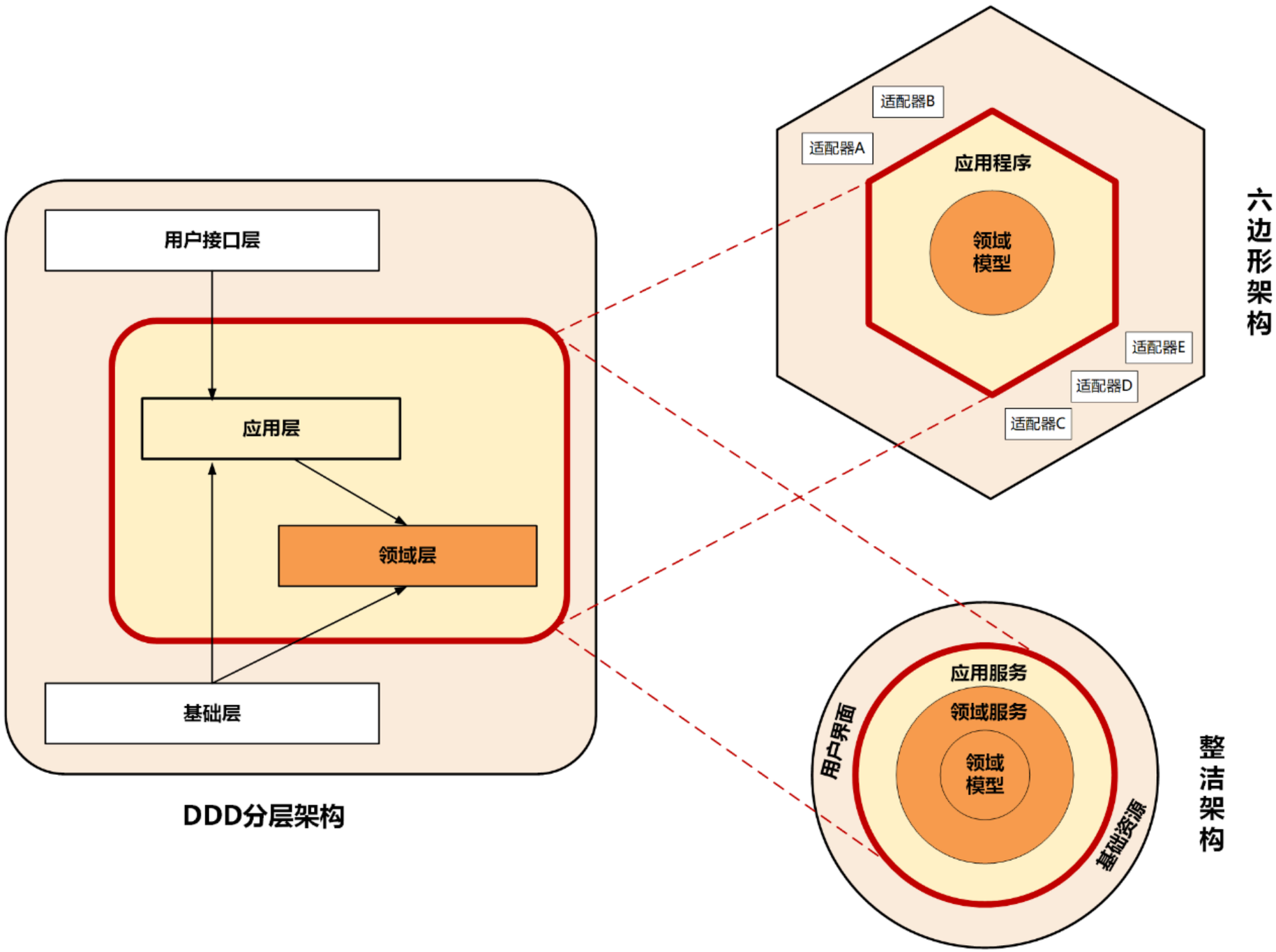
CQRS模式

- CQRS（Command Query Resposibility Segregation，命令查询职责分离）

- 通过将展示模型和领域模型分离，隔离业务流程和查询展示复杂度

- Command是技术角度的写，Query是技术角度的读，我更喜欢叫做业务角度的领域模型展示模型职责分离

三种架构对比


用户接口层
应用层
领域层
基础层
DDD分层架构
适配器B
适配器A
应用程序
领域
模型
适配器E
适配器D
适配器C
六边形架构
应用服务
领域服务
领域
模型
用户界面
基础资源
整洁架构

三种架构对比

- 三种架构的核心思想都是，寻找业务代码模型和技术细节隔离尺度的平衡点，隔离技术复杂度和业务复杂度

- 目的是为了清晰的定义逻辑边界，保障领域层的可移植性和业务模型关联性，以实现可伸缩的演进式架构

- 无论是严格分层架构，还是松散分层架构，无论领域层的抽象程度如何，它都应是不可越过的“耦合底线”

三种架构对比

“在分层架构中，我们将领域模型和业务逻辑分离出来，并减少对基础设施、用户界面甚至应用层逻辑的依赖，因为它们不属于业务逻辑。将一个复杂的系统分为不同的层，每层都应该具有良好的内聚性，并只依赖于比其自身更低的层。”

–Eric Evans

丽呈架构模型如何设计

- 将工程分别定义为敏捷态和稳定态，敏捷态工程和稳定态工程差异化对待

- 稳定态严格按照DDD设计方法执行战略战术设计，代码模型严格遵循高度抽象设计规范

- 敏捷态工程在不违反基本原则下可以做“抽象设计降级”

- 随着敏捷态的业务变的稳定和重要后，迁移至稳定态进行规范，既是演进和沉淀的过程

展示几种不同抽象程度的代码