Domain-Driven Design

with Functional Programming Paradigm

Quick History

- 2003, Domain-Driven Design by Eric Evans
- 2010, DSLs in Action
- 2015, Microservice

DDD

是

- 设计哲学
- 沟通指引
- 方案讨论基础

不是

- 编程框架
- 代码规范
- 解决方案

DDD 是否过时?

领域规划
LAYERED ARCHITECTURE

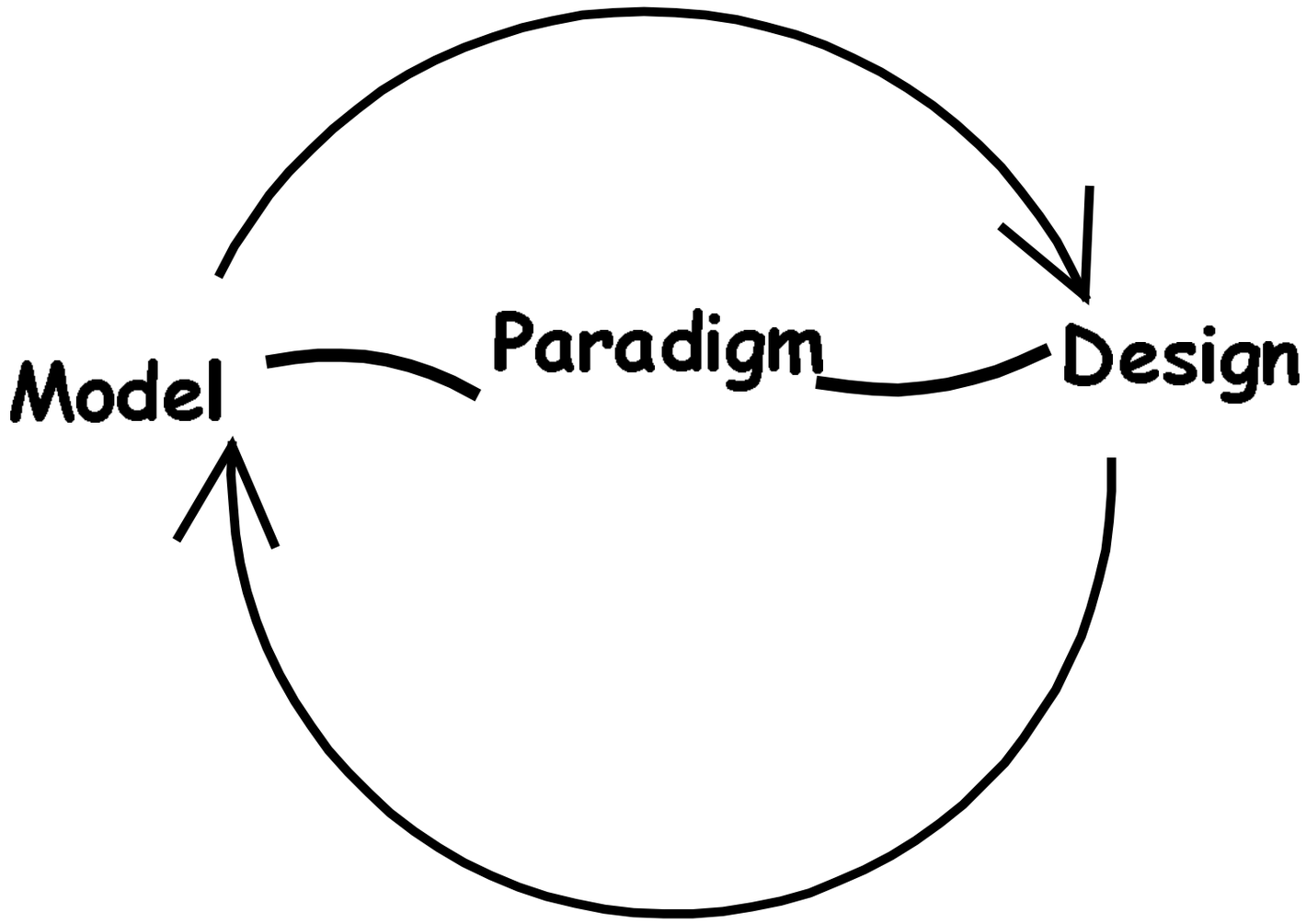
模型设计
Entities / Value Objects / Services

运行架构
Aggregates / Repositories / Factories

新的领域与需求
大数据/机器学习/区块链....

新一代的语言与范式
Scala/Java 8/Golang/Rust/ TypeScript……

新的框架
Akka/Vert.x/Spark/ React……



编程范式影响设计

（OOP 不是唯一选择）

概念对应与FP

Value Object	Tuple, i.e (A, B) / Class
Entity	Immutable Class / Data Type
Entities	Stream of Immutable Data
Aggregates	Collection Monads
Services	IO Monads
Bounded Context	Package
Factories	Functions

业务流程 = 函数组合

- 大部分时候，流程中的各种变化离不开以下基础函数
  - a -> b, 由 a 获得 b
  - [a] -> [b]，由 a 的容器 获得 b 的容器
  - a -> [b]，由 a 获得 b 的容器
  - [a] -> b，由 a 的容器获得 b

- FP 有各种函数组合的方式
  - Higher-Order Functions
  - Map/FlatMap/Fold
  - Kleisli
  - Applicative
  - Functor
  - Monad……

Free Monad 实现
分层结构

- 业务逻辑(Domain Layer) —— AST
- 运行环境(Application Layer) —— Interpreter
- 数据(Persistence Layer) —— Actor

Demo Code in Scala

(akka http + cats + akka)

https://github.com/neomaclin/movie-ticket-reservation/tree/free-monad-b

Q & A