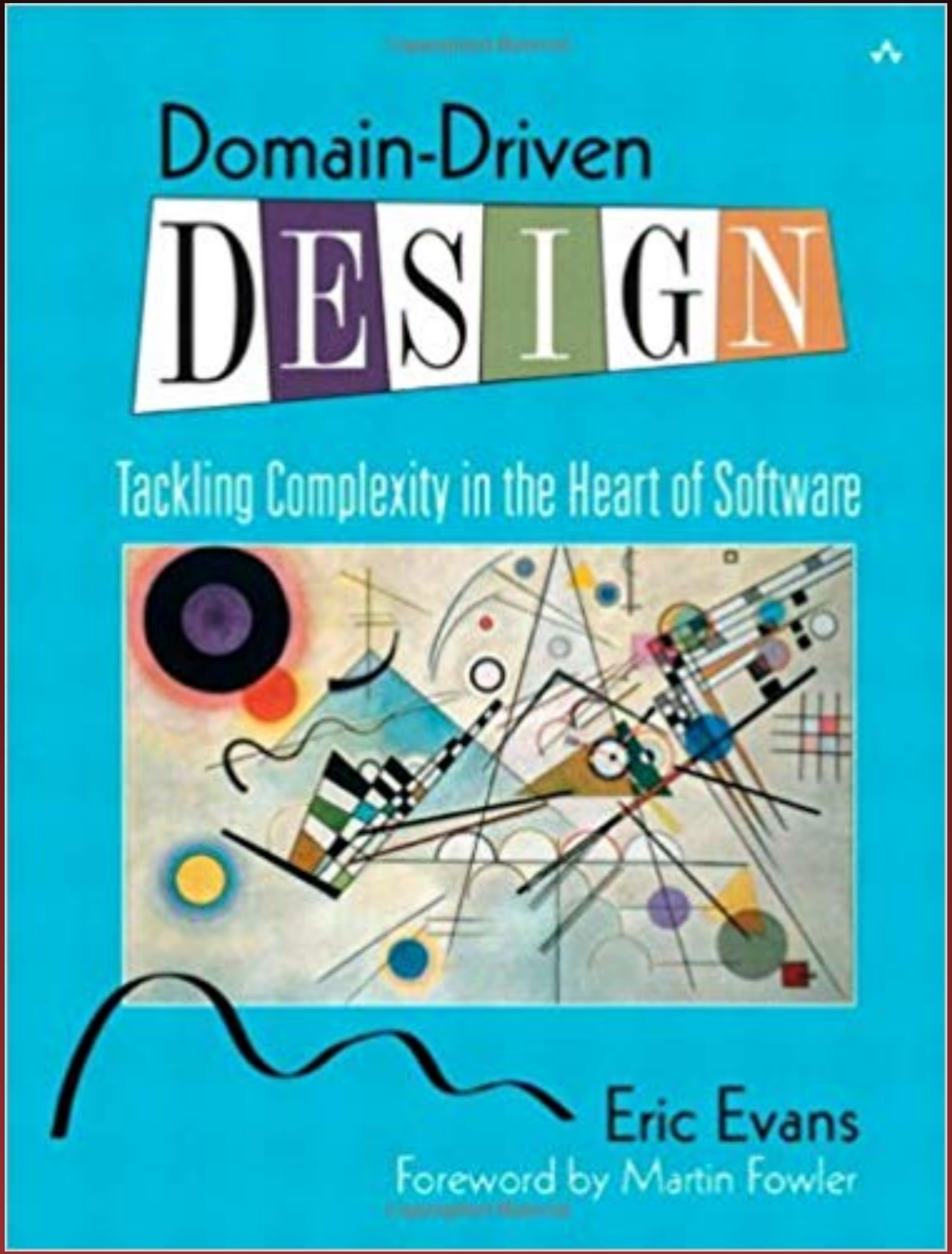
Spring Boot Microservices

Beginner to Guru

Deconstruction Strategies

Domain Driven Design

- **Domain Driven Design** - is a methodology to bring clarity to complexity
- aka - DDD
- Some call DDD an extension to Object Oriented Programming
- DDD concepts can be used to model complex systems
- DDD Concept is accredited to Eric Evans from his 2003 book -
    - "Domain-Driven Design: Tackling Complexity in the Heart of Software"
- See lesson resources for detailed PDF about DDD from Eric Evans.



Domain-Driven DESIGN

Tackling Complexity in the Heart of Software

Eric Evans

Foreword by Martin Fowler

DDD - Definitions

- Following are definitions from Eric Evens 'Domain-Driven Design Reference", 2015
  - Via *Creative Commons Attribution License*
- **Domain** - A sphere of knowledge, influence or activity. The subject area to which the user applies a program is the domain of the software.
- **Model** - A system of abstractions that describes the selected aspects of a domain and can be used to solve problems related to that domain
- **Ubiquitous Language** - A language structured around the domain model and used by all team members within a bounded context to connect all the activities of the team with the software.

DDD Definitions

- **Context** - The setting in which a word or statement appears that determines its meaning. Statements about a model can only be understood in a context.
- **Bounded Context** - A description of a boundary (typically a subsystem, or the work of a particular team) within which a particular model is defined and applicable.

Building Blocks

- **Entities** - Not a traditional Object. Represent a thread of identity that runs through time and often across distinct representations.
- **Value Objects** - Some objects describe or compute some characteristic of a thing. Immutable object, with attributes - but no identity
- **Domain Events** - Something happened that domain experts care about. An object that is used to record a discrete event related to model activity
- **Services** - Sometimes it just isn't a thing. Some concepts are not natural to model as objects.

Building Blocks

- **Aggregates** - are a cluster of entities and value objects
- **Repositories** - Query access to aggregates express in the ubiquitous language. Like a specialized service
- **Factories** - Like OOP, factories create aggregates.

DDD for Microservice Design

- Think in Bounded Contexts -
  - **Bounded Context** - A description of a boundary (typically a subsystem, or the work of a particular team) within which a particular model is defined and applicable.
- A bounded context will help you contain complexity
- Contexts will define common terminology
- DDD Bounded Contexts helps you with organization
- DDD Building blocks help you with defining implementation details

Example

- Warehouse Management System - ie software to run a large warehouse. Receives orders, selects inventory, ships products
- Some 'microservices' in terms of bounded contexts might be:
  - Inventory
  - Order Allocation
  - Manifest (shipping interface with parcel carriers)
  - Labor Tracking
  - Returns

SPRING FRAMEWORK

GURU