Domain Driven Design

Duration 3 day(s) (DDD-03)

Business Domain Oriented Design

Description

Software design on a complex business domain often faces the same recurring problems: - Fragile and rigid code, aging poorly, expensive in maintenance, and difficult to evolve - Difficult or impossible transmission of knowledge, especially in a context of regular turnover - Absence of capitalization on the knowledge of the profession - Loss of credibility and trust in the application The DDD approach proposes to solve these problems by attacking this complexity head-on: the domain model is the core of the software, whether from the point of view of the architecture, the naming of the components, or the effort made. This training exposes the essential concepts of DDD. Its red thread is the improvement of an existing design, as more and more advanced DDD bricks are introduced.

Goals

- Avoid recurring problems caused by data or technique-centric conceptions
- Put IT technical debt under control
- Design maintainable, scalable, and capitalizable software
- Use a pragmatic and proven design approach
- Homogenize the design with the vocabulary of the DDD patterns
- Validate a design or an achievement according to simple criteria
- To speak a unified language between requirements, design, and implementation: the Ubiquitous Language
- Combine the DDD approach and Agility
- Consider DDD in the context of Microservices architectures, Event sourcing / CQRS, NoSQL, REST

Public

- Developers
- Architect
- Analyst
- Project Manager

Prerequisites

- Object Oriented Programming Experience

Structure

50% Theory, 50% Practice

Program

From technology-centric design to craft-centric design

- Business and modeling domain
- Design issues: the usual suspects, causes and consequences
- The essential proposals of the DDD
- Presentation of the red thread: the room reservation application

Business concepts at the heart of the design: Tactical DDD bricks

- Put the technical debt under control by using the simple bricks of DDD: Value Objects and Entities
- Clearly mark the business code of interaction frameworks (web, ..) with Application Services
- Avoid duplication and dispersion of business logic with Domain Services
- Avoid pollution of business code by persistence and infrastructure issues with Repositories and Infrastructure Services

Master the complexity: the advanced tactical DDD

- Know the definition and implications of Transactional Consistency and Eventual Consistency
- To master the complexity induced by the complex graphs of associations between objects, with the Aggregates
- Cut the software into Business Modules
- Separate concerns with Domain Events
- Analysis Patterns: reusable domain models

Integrate software and teams: the Strategic DDD

- The Hexagonal Architecture: the Domain Model as the core of the software
- Upstream and downstream teams: Context Mapping patterns
- Implement: Integration Styles

Overview: DDD and architecture

- CQRS and Event Sourcing
- DDD and NoSQL
- DDD and Microservices
- DDD and REST