Domain Driven Design

Ubiquitous Language, Entities, Aggregates, Events, Services, Repositories, Contexts, Strategic Design

Every enterprise application has a domain – the actual area where the application delivers business value to real users. The database, user interface framework, messaging infrastructure, etc. are just tangential to what the application is really about. In the past, engineering teams concentrated so much on SQL database tables and GUI form layout that the handling of the domain was swamped and lost in the mix.

In contrast, modern software engineering teams rightly place much greater focus on the domain layer of the project and incorporate domain driven design as a central pillar of their project development strategy. Its substantial benefits become very clear on larger projects and on projects that evolve over many iterations. This course explores all the patterns that underlie domain driven design with the goal that at the end of it, attendees will be fluent in DDD and can move from being participants in, to contributors to future projects that incorporate an important domain model. What's above (e.g. the UI) the domain model and what is below (e.g. database, messaging) may well change frequently over iterations, but the domain layer itself will have longevity, so it is extremely important to get it right.

Target Audience

Object-oriented developers, architects and product managers who wish to build robust domain models as the central core of their applications.

Prerequisites

Good all-round experience of software engineering and product development

Contents of One-Day Training Course

Overview

What is domain driven design?
What is its role in the larger software development ecosystem

Layering

Interaction (or UI) layer
Application (or command) layer
Domain layer
Infrastructure layer
Think of the domain as the middle of a sandwich rather than a slice of a pyramid

Ubiquitous Language

"All singing from same hymn sheet"
Identifying a common terminology and set of meanings that all stakeholders can use
Language of the domain (so non-techies can easily understand it)

Entities and Value Objects

Important role of identity
What do we need to identify (and how)
How do we attach values to identities

Aggregates

Boundaries and associations between groupings of entities and value objects
Controlled access

Domain Events

State changes
What is happening inside the domain model and exposing this to outside

Factories & Services

Constructing and supplying entities
Segregating specific responsibilities
Integration with dependency injection

Repositories

Connecting to a database (e.g. ORM) with an aggregate access service
Creating and calling queries
Role of testing

Bounded Context

What is is inside and outside the scope of a domain model
Importance of boundaries & multiple models (good fences make good neighbors)

Supple Design

Intention revealing interfaces
Side-effect free functions
Assertions
Conceptual contours
On-going model evolution

Strategic Design

Core domain
Segregated core
Abstract core
Dependencies – managing relationships between large project subsystems

Large-Scale Projects

Review of layering
Published language and internals
Extensibility
Flexible architecture for longer lifecycles
Handling large systems

Project

The combined use of many domain driven design ideas inside a larger project – including creating the domain model and its use from other layers