Domain-Driven Design – Grundlagen und Praxis

Fachliche Komplexität durch Modelle, klare Grenzen und gemeinsame Sprache beherrschbar machen

01 INTRO

Domain-Driven Design setzt dort an, wo die eigentliche Schwierigkeit eines Softwaresystems nicht in der Technologie, sondern in einer komplexen Fachdomäne liegt. Softwaremodell und fachliches Verständnis werden dabei gemeinsam entwickelt.

Der Kurs vermittelt strategisches und taktisches Domain-Driven Design und verbindet beide Ebenen mit praktischer Softwareentwicklung. Im Mittelpunkt stehen Ubiquitous Language, Bounded Contexts, Domänenmodelle und die kontrollierte Integration unterschiedlicher fachlicher Modelle.

DAUER

3 Tage

FORMAT

Präsenz · Live Online · Inhouse

VORKENNTNISSE

Praktische Erfahrung mit Softwareentwicklung und objektorientiertem Design

02 TRAININGSPROGRAMM

Inhalte

Warum Domain-Driven Design?

- fachliche Komplexität
- Domain
- Subdomain
- Modell
- Wissen
- Zusammenarbeit mit Fachexperten
- sinnvoller Einsatz

Ubiquitous Language

- gemeinsame Sprache
- Begriffe
- Mehrdeutigkeiten
- Sprache im Code
- Sprache in Tests
- Modelle weiterentwickeln
- Wissen sichtbar machen

Strategisches Design

- Core Domain
- Supporting Subdomain
- Generic Subdomain
- Priorisierung
- Make or Buy
- Investitionsentscheidungen
- fachliche Grenzen

Bounded Contexts

- Modellgrenzen
- gleiche Begriffe, unterschiedliche Bedeutung
- Verantwortlichkeiten
- Datenhoheit
- organisatorische Grenzen
- technische Grenzen
- Context Mapping

Beziehungen zwischen Contexts

- Partnership
- Shared Kernel
- Customer und Supplier
- Conformist
- Anti-Corruption Layer
- Open Host Service
- Published Language

Entities und Value Objects

- Identität
- Werte
- Lebenszyklus
- Immutability
- Invarianten
- Modellierungsentscheidungen
- Beispiele

Aggregates

- Konsistenzgrenzen
- Aggregate Root
- Invarianten
- Referenzen
- Transaktionen
- Größe
- typische Fehlkonstruktionen

Services, Repositories und Factories

- Domain Services
- Application Services
- Repository
- Factory
- Verantwortlichkeiten
- Infrastruktur
- Domänenmodell sauber halten

Domain Events

- fachlich relevante Ereignisse
- Modellierung
- Entkopplung
- Reaktionen
- Integration Events
- Eventual Consistency
- Event Storming einordnen

Domain-Driven Design und Architektur

- Layered Architecture
- Hexagonal Architecture
- Ports and Adapters
- CQRS einordnen
- Persistenz
- technische Frameworks
- Domäne schützen

Externe Systeme integrieren

- fremde Modelle
- Anti-Corruption Layer
- Adapter
- Übersetzung
- APIs
- Events
- Fehlergrenzen
- Modellhoheit

Modellierungs-Workshop

- Domäne untersuchen
- Sprache entwickeln
- Subdomains identifizieren
- Bounded Contexts schneiden
- Modell entwickeln
- Integration definieren
- Entscheidungen diskutieren

03 DURCHFÜHRUNG

Rahmen und Organisation

Dauer	3 Tage
Durchführung	Präsenz oder Live Online, vorzugsweise als Inhouse-Schulung
Schwerpunkt	Inhouse-Schulungen für Unternehmen und Teams
Sprache	Deutsch, englische Fachbegriffe und Dokumentation
Unterlagen	Kursunterlagen, Beispiele und Übungsaufgaben
Technik	Eigener Rechner für die praktischen Übungen

> **Inhouse-Schulungen**
> Eine eigene Fachdomäne kann als Grundlage für den Modellierungs-Workshop dienen.

04 ZIELGRUPPE

Für wen ist der Kurs gedacht?

Softwareentwickler, Softwarearchitekten, technische Leads und Business Analysts, die komplexe Fachdomänen in Software abbilden.

> **Voraussetzungen**
> Praktische Softwareentwicklung und Grundlagen objektorientierten Designs.

05 LERNZIELE

Was Sie nach dem Kurs können

Die Teilnehmer können fachliche Modelle entwickeln, Bounded Contexts identifizieren, taktische Bausteine des Domain-Driven Design einsetzen und unterschiedliche Domänenmodelle kontrolliert integrieren.

06 FAQ

Häufige Fragen

Ist Domain-Driven Design eine Softwarearchitektur?

Nein. Domain-Driven Design ist primär ein Ansatz zur Modellierung komplexer Fachdomänen. Bestimmte Architekturformen unterstützen seine Umsetzung.

Braucht jedes Projekt Domain-Driven Design?

Nein. Bei geringer fachlicher Komplexität kann es unnötigen Aufwand verursachen.

Werden Microservices benötigt?

Nein. Bounded Contexts können beispielsweise auch innerhalb eines modularen Monolithen umgesetzt werden.

UC IT Service · Ulrich Cuber · kontakt@uc-it.de
https://www.uc-it.de/coaching/domain-driven-design/