

Domain-Driven Design Essentials

Duration: 5 Days

Description

Domain-Driven Design Essentials is a comprehensive five-day online training program for technical teams seeking to analyze complex business problems, create effective domain models and translate those models into maintainable software solutions.

Domain-Driven Design is not merely a coding technique. It is a software design and problem-solving approach that helps development teams and domain experts build a shared understanding of the business. By aligning software models with business concepts, DDD enables teams to manage complexity and create systems that can evolve with changing requirements.

Participants explore strategic Domain-Driven Design concepts, including problem space, solution space, subdomains, Ubiquitous Language, Bounded Context and Context Mapping. Tactical design building blocks such as Entities, Value Objects, Aggregates, Domain Services, Domain Events, Factories and Repositories are then translated into code through practical exercises.

The training concludes with a hands-on coding workshop in which participants model a real-world business problem and implement the resulting domain model using C#.

Learning Outcomes

- Explain the purpose and principles of Domain-Driven Design
- Analyze complex business problems from a domain perspective
- Distinguish between problem space and solution space
- Identify Core, Supporting and Generic Subdomains
- Create a Ubiquitous Language shared by technical and domain experts
- Define effective Bounded Context boundaries
- Model relationships between contexts through Context Mapping
- Use Event Storming to explore business processes and domain events
- Apply strategic and tactical DDD patterns
- Translate a domain model into working C# code

Audience

- Intermediate and senior software developers
- Backend developers

- Software and solution architects
- Technical leads and engineering leads
- Technical teams collaborating with business analysts and domain experts
- Teams developing microservices or modernizing enterprise applications
- Professionals working on systems with complex business rules

Outline

1. Introduction to Domain-Driven Design

- What is Domain-Driven Design?
- The purpose and core principles of DDD
- Software development problems addressed by DDD
- Common misconceptions about Domain-Driven Design
- Why teams may feel intimidated by DDD
- Good, bad and effective software design
- The role of DDD in the software development lifecycle

2. Understanding the Domain

- What is a business domain?
- The relationship between business problems and software models
- Discovering domain knowledge
- Collaboration between domain experts and development teams
- Domain modeling fundamentals
- Continuously refining the domain model
- Maintaining alignment between the model and the code

3. Problem Space and Solution Space

- Understanding the problem space
- Understanding the solution space
- Separating the business problem from its technical implementation
- Discovering and clarifying business requirements
- Modeling the domain before selecting a technical solution
- Connecting problem-space insights to solution-space decisions

4. Subdomain Analysis

- What is a subdomain?
- Breaking a complex domain into manageable areas
- Core Domain
- Supporting Subdomain

- Generic Subdomain
- Prioritizing subdomains according to business value
- Connecting organizational capabilities with subdomains

5. Ubiquitous Language

- What is Ubiquitous Language?
- Creating a shared language between business and technical teams
- Defining domain concepts clearly and consistently
- Using the shared language in analysis, documentation and code
- Identifying ambiguous or conflicting terminology
- Evolving the Ubiquitous Language together with the domain model

6. Strategic Domain-Driven Design

- The purpose of strategic design
- Decomposing large and complex systems
- Identifying domain and model boundaries
- Aligning business capabilities with software boundaries
- The impact of strategic decisions on software architecture

7. Bounded Context

- What is a Bounded Context?
- Defining the boundaries of a domain model
- Understanding how the same term can have different meanings across contexts
- The relationship between Bounded Contexts and subdomains
- Criteria for defining effective context boundaries
- Problems caused by unclear or incorrect context boundaries

8. Context Mapping

- What is a Context Map?
- Modeling relationships between Bounded Contexts
- Evaluating team and system boundaries
- Sharing data and models between contexts
- Identifying integration points
- Translating Context Mapping results into architectural decisions

9. Event Storming

- What is Event Storming?
- Using Event Storming for collaborative domain discovery
- Identifying Domain Events

- Exploring commands, actors and business rules
- Bringing domain experts and technical teams together
- Using Event Storming outputs to identify Bounded Contexts

10. Tactical Domain-Driven Design

- The purpose of tactical design
- Representing the domain model in code
- Placing business rules in the appropriate layer
- Rich Domain Model principles
- Relationships between tactical design building blocks
- Avoiding an Anemic Domain Model

11. Entities and Value Objects

- What is an Entity?
- Identity and lifecycle concepts
- What is a Value Object?
- Equality by value
- Immutability principles
- Differences between Entities and Value Objects
- Selecting the appropriate modeling component

12. Aggregates and Aggregate Roots

- What is an Aggregate?
- The Aggregate Root
- Protecting business rules and invariants
- Defining Aggregate boundaries
- Managing relationships between Aggregates
- Avoiding unnecessarily large Aggregates
- Transaction and consistency boundaries

13. Domain Services and Application Services

- What is a Domain Service?
- Modeling domain behavior that does not belong to an Entity or Value Object
- Responsibilities of an Application Service
- Separating domain and application concerns
- Placing business rules in the appropriate service

14. Domain Events

- What is a Domain Event?

- Modeling significant changes within the domain
- Creating and publishing Domain Events
- Reducing unnecessary dependencies between components
- Connecting Domain Events with business processes

15. Factories, Repositories and Modules

- The purpose of the Factory pattern
- Managing complex object creation
- What is a Repository?
- Separating the domain model from data access mechanisms
- Defining Repository boundaries
- Using Modules to organize the domain model

16. Domain-Driven Design and Clean Architecture

- Introduction to Clean Architecture
- The position of the domain layer within the architecture
- Directing dependencies toward the core domain
- Separating domain, application and infrastructure responsibilities
- Developing a technology-independent domain model
- Building maintainable and testable software

17. Coding Workshop

- Discovering the domain through a real-world business problem
- Analyzing the problem space and solution space
- Identifying subdomains and the Core Domain
- Creating a Ubiquitous Language
- Defining Bounded Context boundaries
- Applying strategic and tactical DDD concepts
- Translating the domain model into C# code
- Implementing Entities, Value Objects and Aggregates
- Implementing Repositories and Domain Services
- Live coding and solution review

Prerequisites

- Participants are expected to have at least intermediate-level programming knowledge.
- Experience with a commonly used object-oriented programming language is sufficient.
- Basic C# knowledge is recommended because the practical exercises are

implemented in C#.

- Visual Studio or Rider, Docker and C# 8.0—or the compatible version specified before the course—must be installed on participants' computers.