ONE PAGE SUMMARY

IT & Software › Software Architecture › Domain Driven Design

Introduction to Domain Driven Design

Mastering Domain Driven Design for Effective Software Solutions

DEFINITION

> Domain Driven Design (DDD) is an approach to software development that focuses on modeling complex systems based on the business domain. It emphasizes collaboration between technical and domain experts to create a shared understanding.

KEY CONCEPTS

Ubiquitous Language

A common language developed by both developers and domain experts to ensure clear communication.

Bounded Context

A defined boundary within which a particular model is applicable, helping to manage complexity.

Entities

Objects that have a distinct identity and lifecycle, representing key business concepts.

Value Objects

Objects that describe attributes but do not have a distinct identity, often used for measurement.

Aggregates

Clusters of related entities and value objects that are treated as a single unit for data changes.

EXAMPLES

→ A shopping cart as an aggregate containing items (entities) and total price (value object).

→ A user profile as an entity that includes personal details (value objects) like name and address.

MEMORY TIPS

★ Think of DDD as 'Designing Domain Dynamics' to remember its focus on the business domain.

★ Use the acronym 'EVA' for Entities, Value Objects, and Aggregates to recall key building blocks.

COMMON MISTAKES

✗ Confusing entities with value objects by overlooking identity.

✗ Neglecting to define bounded contexts, leading to unclear model boundaries.

QUICK RECAP

Domain Driven Design focuses on aligning software models with business needs. Key concepts include ubiquitous language, bounded contexts, and aggregates, which help manage complexity and enhance collaboration.