The CPSA® Advanced Level Module DDD

Day 1: Foundations - Domain, Model & Ubiquitous Language

iSAQB® Training Course in Domain-Driven Design

22 December 2025

VOLKSWAGEN GROUP INDIA

Course Overview: 6-Day Training Journey

Day 1

Foundations

Domain, Model & Ubiquitous Language

22 Dec 2025

Day 2

Knowledge Crunching

Collaborative Modeling with Domain Experts

05 Jan 2026

Day 3

Implementation

From Model to Technical Implementation

06 Jan 2026

Day 4

Architecture

Model in Application Architecture

07 Jan 2026

Day 5

Strategic Design 1

Cutting and Distinguishing Models

08 Jan 2026

Day 6

Strategic Design 2

Context Mapping & Integration

09 Jan 2026

Day 1 Agenda: Foundations - Domain, Model & Ubiquitous Language

Introduction to DDD

What is Domain-Driven Design and why it matters for automotive

Domains, Software & Models

Understanding connections between business domains and software

Ubiquitous Language

Creating shared vocabulary between domain experts and developers

DDD Building Blocks

Entities, Value Objects, Aggregates, Services, Repositories, Factories

Automotive Applications

Applying DDD concepts to **powertrain**, **infotainment**, and **safety** systems

Hands-on Activities

Domain modeling exercises and mini-project

Learning Goals: Day 1

1-1

Domain Connections

Explain connections between **domains**, **software**, and **models**

1-2

Ubiquitous Language

Understand role of **ubiquitous language** in domain modeling

1-3

DDD Building Blocks

Explain DDD building blocks (**Entities**, **Value Objects**, **Aggregates**)

1-4

Block Connections

Explain connections between building blocks

What is Domain-Driven Design?

DDD Concept

Strategic Approach

Design software as **precise**, **transparent**, and **transformable** representation of business domain

Shared Understanding

Creates **common language** between domain experts and developers

> *"Domain-Driven Design tackles complexity at the heart of software, focusing on the core domain and its logic."*

Why for Automotive?

Software Complexity

Modern vehicles contain **100+ million lines of code** across multiple domains

Multiple Domains

Infotainment, **safety**, **powertrain**, connectivity, autonomous driving

Cross-Team Collaboration

Different teams work on **interconnected** vehicle subsystems

100M+	150+	5+
Lines of Code	ECUs in Premium Vehicles	Software Domains

LG 1-1: Connections between Domains, Software, and Models

Core Concepts

Domain

Sphere of **knowledge**, **activity**, or **influence**

Software

Implementation that serves the domain

Model

Abstraction of domain concepts

Automotive Application

Powertrain Domain

- **Domain:** Engine components, fuel systems, transmission
- **Model:** Engine class, FuelSystem class, Transmission class
- **Software:** Powertrain control module implementation

Key Principle

Software must **serve the domain**, not the other way around

LG 1-2: Role of Ubiquitous Language in Domain Modeling

Ubiquitous Language

Common Terminology

Shared vocabulary between domain experts and developers

Bridge Communication

Eliminates **translation** between business and technical teams

Living Documentation

Language **evolves** with domain understanding

Automotive Application

Infotainment System

- **Terms:** MediaSource, Playlist, NavigationRoute
- **Consistent:** Used in discussions, code, and documentation

Powertrain Domain

- **Terms:** TorqueCurve, GearRatio, ThrottlePosition
- **Precise:** Clear meaning for engineers and developers

LG 1-3: DDD Building Blocks

Entity

Objects with **distinct identity** that track state over time

Automotive Example:
Vehicle, Customer, Order

Value Object

Immutable objects defined by attributes, not identity

Automotive Example:
VIN, Speed, Temperature

Aggregate

Cluster of related objects treated as a **single unit**

Automotive Example:
Vehicle with Components

Service

Stateless operations that don't naturally fit entities

Automotive Example:
PaymentProcessor, DiagnosticService

Repository

Mediates between domain and **data mapping** layers

Automotive Example:
VehicleRepository, CustomerRepository

Factory

Encapsulates **complex object creation** logic

Automotive Example:
VehicleFactory, OrderFactory

LG 1-4: Connections between Building Blocks

Key Relationships

Aggregates Contain

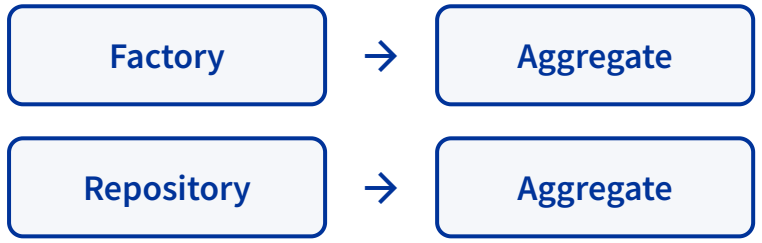
Aggregates contain **Entities** and **Value Objects**

Repositories Manage

Repositories manage **Aggregates**

Factories Create

Factories create complex **Entities** and **Aggregates**



Automotive Example

Vehicle Aggregate

- **Entity:** Vehicle (with unique VIN)
- **Value Objects:** Speed, Temperature, FuelLevel
- **Service:** DiagnosticService (stateless)

Vehicle Factory

- Creates Vehicle with proper validation
- VehicleRepository persists and retrieves

Key Concepts with Automotive Examples

Vehicle Domain

- **Entity:** Vehicle with unique VIN
- **Value Object:** LicensePlate, Color
- **Aggregate:** Vehicle with Components
- **Repository:** VehicleRepository

Infotainment System

- **Entity:** MediaSession, NavigationRoute
- **Value Object:** MediaSource, Coordinates
- **Service:** MediaStreamingService
- **Ubiquitous Language:** Playlist, Route, VoiceCommand

Powertrain Domain

- **Entity:** Engine, Transmission
- **Value Object:** TorqueCurve, GearRatio
- **Factory:** PowertrainFactory
- **Ubiquitous Language:** ThrottlePosition, FuelInjection

Activities & Exercises

Define Automotive Domain Boundaries

1. Identify **key domains** in vehicle systems
2. Define **scope** and boundaries
3. Map **interactions** between domains

Create Ubiquitous Language

1. Gather **domain terminology**
2. Create **shared vocabulary**
3. Define **glossary** for vehicle systems

Model Domain Concepts

1. Identify **entities** and **value objects**
2. Group into **aggregates**
3. Create **class diagrams**

Mini-Project: Automotive Domain Model

Project Steps

1 Define Domain

Choose an automotive domain (e.g., **infotainment**, **powertrain**, **safety**)

2 Identify Building Blocks

List **entities**, **value objects**, and **aggregates**

3 Create Ubiquitous Language

Define **shared terminology** for domain concepts

4 Model Relationships

Show connections between **building blocks**

Example: Infotainment System

Entities

- MediaSession (unique ID)
- NavigationRoute (unique ID)

Value Objects

- MediaSource (attributes only)
- Coordinates (attributes only)

Ubiquitous Language

- Playlist, Route, VoiceCommand, MediaControl

Resources & Tools

Learning Platforms

DDD-Crew
Free resources, patterns, and examples

Domain-Driven Design Weekly
Weekly video series on DDD concepts

DDD Community
Discussions, Q&A, and case studies

Modeling Tools

PlantUML
Text-based UML diagrams for domain models

Miro
Collaborative whiteboard for modeling

EventStorming.com
Tools and techniques for event storming

Books & Blogs

Domain-Driven Design
Eric Evans - The foundational DDD book

Vaughn Vernon's Blog
Practical DDD implementation strategies

DDD in Practice
Real-world case studies and patterns

Connecting to Day 2

Day 2: 05 January 2026

Day 2 Topics

Knowledge Crunching

Working with domain experts to extract knowledge

Collaborative Modeling

EventStorming , Domain Storytelling, User Story Mapping

Knowledge Elicitation

Interviewing, observation, field observation, apprenticing

Review Before Day 2

Key Concepts

- Ubiquitous Language principles
- Building Blocks relationships
- Automotive domain examples

Self-Study Materials

- EventStorming guide
- Collaborative modeling videos
- DDD community discussions

Summary & Q&A

Day 1 Key Points

- **Domain-Driven Design** creates precise, transparent representations of business domains
- **Ubiquitous Language** bridges communication between domain experts and developers
- **Building Blocks** (Entities, Value Objects, Aggregates) form the foundation of DDD
- DDD helps manage **automotive complexity** across multiple vehicle systems

Questions & Discussion

Let's clarify any concepts and discuss how DDD applies to your specific automotive projects at Volkswagen