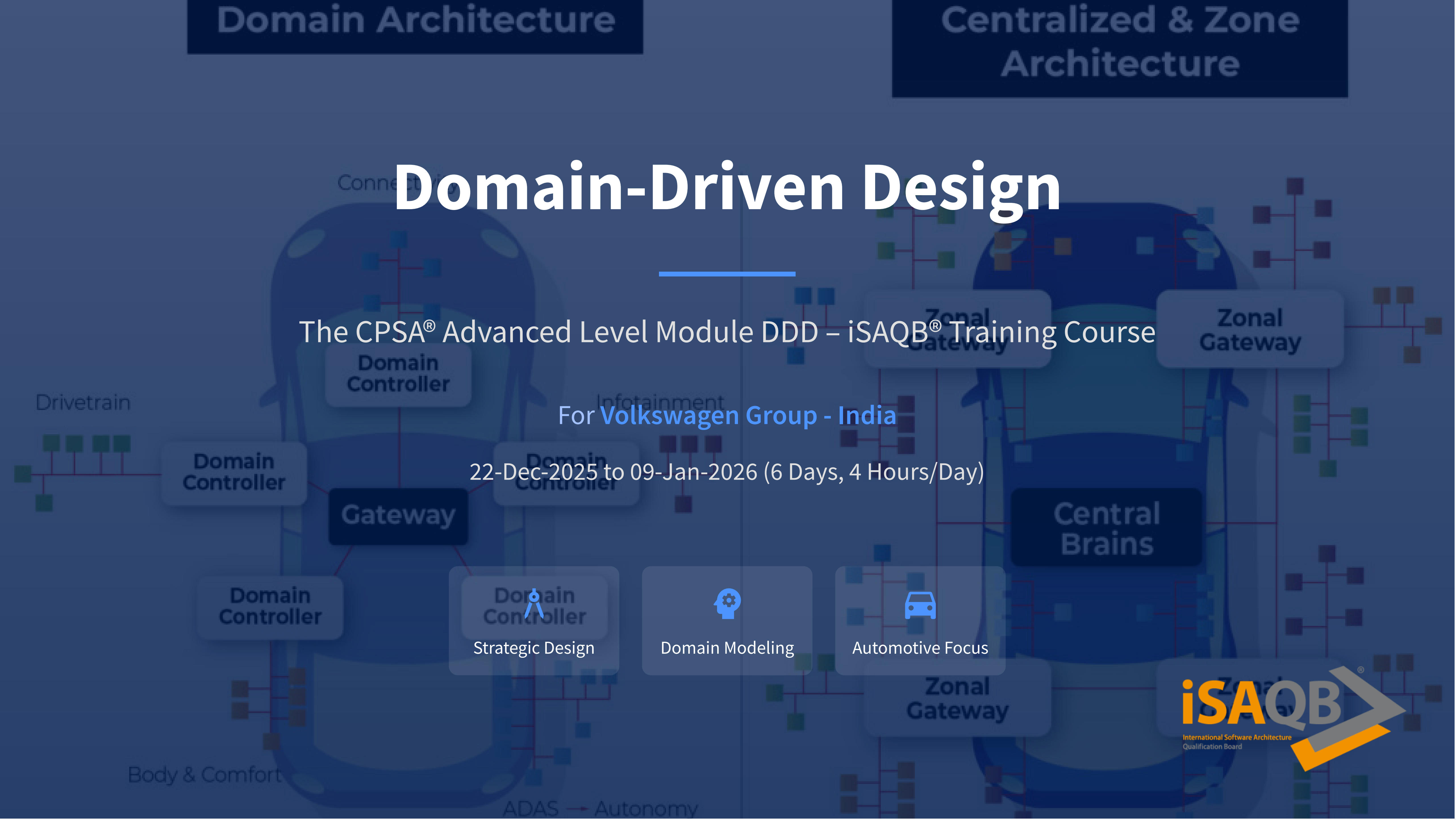


Domain-Driven Design

The CPSA® Advanced Level Module DDD – iSAQB® Training Course

For **Volkswagen Group - India**

22-Dec-2025 to 09-Jan-2026 (6 Days, 4 Hours/Day)

Strategic Design

Domain Modeling

Automotive Focus

iSAQB®
International Software Architecture Qualification Board

What is Domain-Driven Design?

Strategic Approach

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

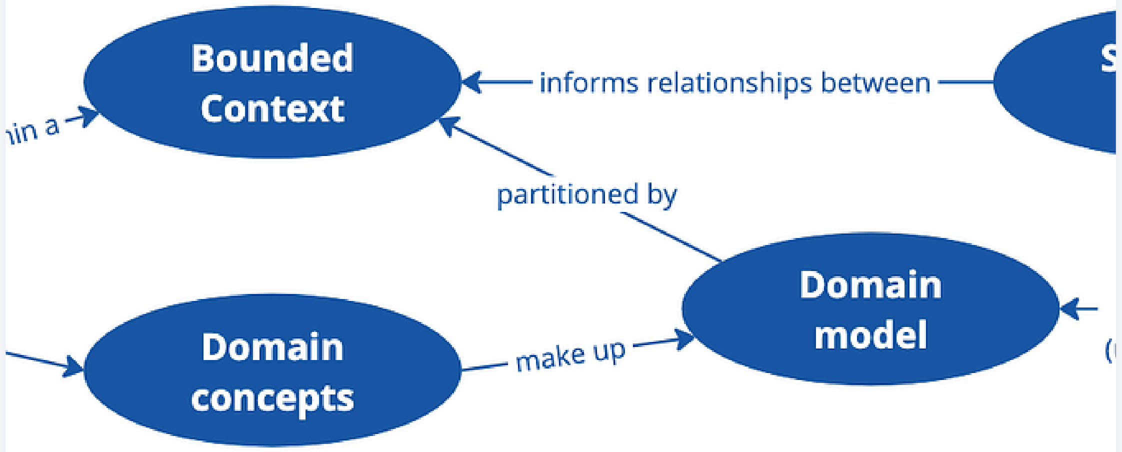
Shared Understanding

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

Why for Automotive?

Modern vehicles contain **100+ million lines of code** across multiple domains (infotainment, safety, powertrain)

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



iSAQB Certification Pathway

Foundation Level

Completed ✓

Fundamental knowledge of software architecture principles, methods, and techniques

- ✓ Architecture Fundamentals
- ✓ Quality Attributes
- ✓ Design Patterns

→

Advanced Level

Current Focus

Specialized modules for in-depth knowledge in specific areas of software architecture

- DDD (Current)
- Cloud Architecture
- Security
- Embedded Systems

Technical Competence	Methodical Competence	Communicative Competence	Total Credits
0	20	10	30

Training Journey Overview

6-Day Progressive Learning Path

24 hours of instruction (4 hours/day)

1
22-Dec-2025

Foundations

Domain, Model & Ubiquitous Language

- Entities
- Value Objects
- Aggregates

2
05-Jan-2026

Knowledge

Knowledge Crunching & Collaborative Modeling

- EventStorming
- Interviews
- Observation

3
06-Jan-2026

Implementation

From Model to Technical Implementation

- SOLID
- Factories
- Repositories

4
07-Jan-2026

Architecture

Model in Application Architecture

- Hexagonal
- CQRS
- BDD

5
08-Jan-2026

Cutting and Distinguishing Models

- Bounded Contexts
- Subdomains
- Core

6
09-Jan-2026

Strategic 2

Context Mapping & Integration

- Customer/Supplier
- ACL
- Domain Events

Foundations — Knowledge — Implementation — Architecture — Strategic

DDD Building Blocks

Core components that form the foundation of **Domain-Driven Design**

Entity

Objects with distinct identity that track state over time

Example: Vehicle, Customer, Order

Value Object

Immutable objects defined by attributes, not identity

Example: VIN, Speed, Temperature

Aggregate

Cluster of related objects treated as a single unit

Example: Vehicle with Components

Service

Stateless operations that don't naturally fit entities

Example: PaymentProcessor, DiagnosticService

Repository

Mediates between domain and data mapping layers

Example: VehicleRepository, CustomerRepository

Factory

Encapsulates complex object creation logic

Example: VehicleFactory, OrderFactory

Key Relationships

- → Aggregates contain Entities and Value Objects
- → Repositories manage Aggregates
- → Factories create complex Entities

Day 1: Foundations - Domain, Model & Ubiquitous Language

1 22-Dec-2025

Foundations: Domain, Model & Ubiquitous Language

Learning Goals

✓ **LG 1-1**
Explain connections between **domains, software, and models**

✓ **LG 1-2**
Understand role of **ubiquitous language** in domain modeling

✓ **LG 1-3**
Explain DDD building blocks (Entities, Value Objects, Aggregates)

✓ **LG 1-4**
Explain connections between building blocks

Concepts & Activities

Domain & Domain Model
Software as representation of expert knowledge

Ubiquitous Language
Common terminology for experts and developers

Activities
- Define automotive domain boundaries
- Create ubiquitous language for vehicle systems
- Model domain concepts using class diagrams

Skills & Automotive Application

Skills Being Built

DDD Foundations · Domain Modeling · Communication · Technical Translation

Automotive Applications

Powertrain Domain
Common terminology for engine components

Infotainment System
Shared language for media interfaces

Safety Systems
Consistent terms for safety features

Day 2: Knowledge Crunching - The Path to the Model

2

05-Jan-2026

Knowledge Crunching: The Path to the Model

Learning Goals

✓ **LG 2-1 to 2-3**

Empower **domain experts**, select suitable contacts, communicate effectively

✓ **LG 2-4 to 2-6**

Use **modeling techniques**, conduct interviews, apply observation

✓ **LG 2-7 to 2-9**

Overview of **Collaborative Modeling**, conduct workshops

✓ **LG 2-10**

Understand **agility** as foundation of DDD

Concepts & Activities

Domain Expert Empowerment

Leveraging expert knowledge effectively

Collaborative Modeling

EventStorming, Domain Storytelling, User Story Mapping

Activities

- Workshop-style collaborative modeling
- Role-playing interviews with experts
- Analyzing automotive domain through observation

Skills & Automotive Application

Skills Being Built

Communication | Collaboration | Domain Analysis | Workshop Facilitation

Automotive Collaborative Modeling

EventStorming
Map vehicle manufacturing events

Domain Storytelling
Visualize customer ordering process

Expert Interviews
Extract powertrain domain knowledge

Day 3: From Model to Implementation

3 06-Jan-2026

From Model to Implementation

Learning Goals

✓ **LG 3-1**
Extend domain model with technical building blocks

✓ **LG 3-2**
Model interfaces for domain classes

✓ **LG 3-3**
Account for interactions between implementation and model

✓ **LG 3-4**
Argue why DDD is worthwhile for complex business logic

Concepts & Activities

SOLID Principles
Single Responsibility, Open/Closed, Liskov Substitution

Technical Building Blocks
Repositories, Factories, Aggregates

Activities
• Refactor domain model into technical components
• Design interfaces and technical layers
• Debate benefits of DDD for complex logic

Skills & Automotive Application

Skills Being Built

Technical Implementation
Design Principles
Refactoring
DDD Justification

Automotive Implementation

Vehicle Aggregate
Implement with components and invariants

Repository Pattern
Vehicle and component data management

Factory Pattern
Complex vehicle configuration creation

Day 4: The Model in Application Architecture

4

07-Jan-2026

The Model in Application Architecture

Learning Goals

LG 4-1

Design a **ports & adapter architecture** for domain model

LG 4-2

Formulate correlations between **DDD and BDD**

Key Takeaway

Integrate domain model into larger system architecture

Concepts & Activities

Hexagonal Architecture

Ports & Adapters pattern for isolation

CQRS

Command-Query Responsibility Segregation

Activities

- Design hexagonal architecture for domain model
- Compare DDD and BDD approaches
- Map domain model to architectural layers

Skills & Automotive Application

Skills Being Built

Architectural Design

System Integration

Pattern Application

DDD vs BDD

Automotive Architecture

Telematics System

Hexagonal architecture for data isolation

Infotainment

CQRS for media playback vs. settings

BDD Integration

Behavior specs for safety features

Day 5: Strategic Design 1 - Cutting and Distinguishing Models

5

08-Jan-2026

Strategic Design 1: Cutting and Distinguishing Models

Learning Goals

✓ **LG 5-1 to 5-2**
Identify **symptoms** of large models, assess **cross-team models**

✓ **LG 5-3**
Move from **problem to solution space**

✓ **LG 5-4 to 5-5**
Distill **core**, describe **Bounded Contexts** in Context Map

Concepts & Activities

Problem vs Solution Space
Distinguishing domain problems from technical solutions

Subdomain Classification
Core, Supporting, Generic subdomains

Activities
- Analyze large system and identify problems
- Define Bounded Contexts and relationships
- Create Context Map for complex system

Skills & Automotive Application

Skills Being Built

Strategic Thinking

System Decomposition

Context Mapping | Architectural Vision

Automotive Strategic Design

Vehicle Platform
Bounded contexts for powertrain, infotainment

Core Domain
Identify differentiating features for VW

Context Map
Visualize relationships between vehicle systems

Day 6: Strategic Design 2 - Context Mapping

6

09-Jan-2026

Strategic Design 2: Context Mapping

Learning Goals

✓ **LG 6-1 to 6-2**

Use interfaces for **customer/supplier teams**, design **Open Host Service**

✓ **LG 6-3 to 6-4**

Isolate model with **Anticorruption Layer**, reuse elements in **Shared Kernel**

✓ **LG 6-5 to 6-6**

Understand **Separate Ways**, use **Domain Events** for communication

Concepts & Activities

Customer/Supplier

Relationships between upstream and downstream contexts

Integration Patterns

OHS, ACL, Shared Kernel, Separate Ways

Activities

- Design inter-context communication strategies
- Map system integrations using context patterns
- Design Anticorruption Layer for external systems

Skills & Automotive Application

Skills Being Built

Inter-context Design | Integration Strategies | Communication Patterns | System Boundaries

Automotive Integration Patterns

Powertrain → Vehicle

Customer/Supplier relationship

Diagnostics System

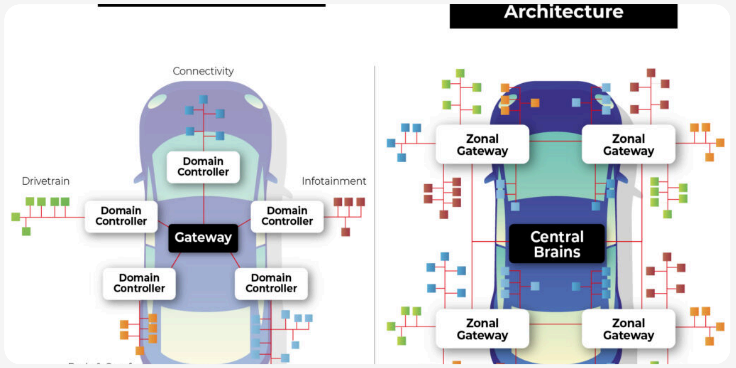
Open Host Service for multiple clients

Manufacturing Events

Domain Events across systems

Connecting DDD to Automotive Domain

Automotive Software Complexity



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

Multiple Domains
Infotainment, powertrain, safety, connectivity, autonomous driving

Cross-team Collaboration
Different teams work on different vehicle subsystems

DDD Applications in Automotive

Bounded Contexts
Separate contexts for **infotainment**, **powertrain**, **safety**, **connectivity**

Context Mapping
Customer/Supplier relationships between safety and powertrain

Domain Events
Event-driven communication between vehicle subsystems

Ubiquitous Language
Common terminology for engineers, developers, and domain experts

Applying to Volkswagen Projects

Project Integration
Apply DDD to your specific Volkswagen projects

Key Applications
- Infotainment Systems
- EV Battery Management
- Autonomous Driving
- Connected Services
- Fleet Management

> *"DDD helps Volkswagen teams create clear boundaries between vehicle subsystems while ensuring effective communication between them."*

Benefits
Reduced complexity, better maintainability, improved collaboration between teams

End Outcomes & Next Steps

Skills Acquired

Strategic Design
Decompose complex systems into manageable contexts

Tactical Implementation
Translate domain models into technical solutions

Collaboration
Bridge communication between domain experts and developers

Domain Modeling · Context Mapping · Event-Driven Design · Hexagonal Architecture

Value for Volkswagen Projects

Automotive Software Excellence
Design complex vehicle systems with clear boundaries

Faster Development
Reduce integration complexity between vehicle subsystems

Better Decision Making
Strategic approach to system architecture decisions

Cross-Team Collaboration
Effective communication between specialized teams

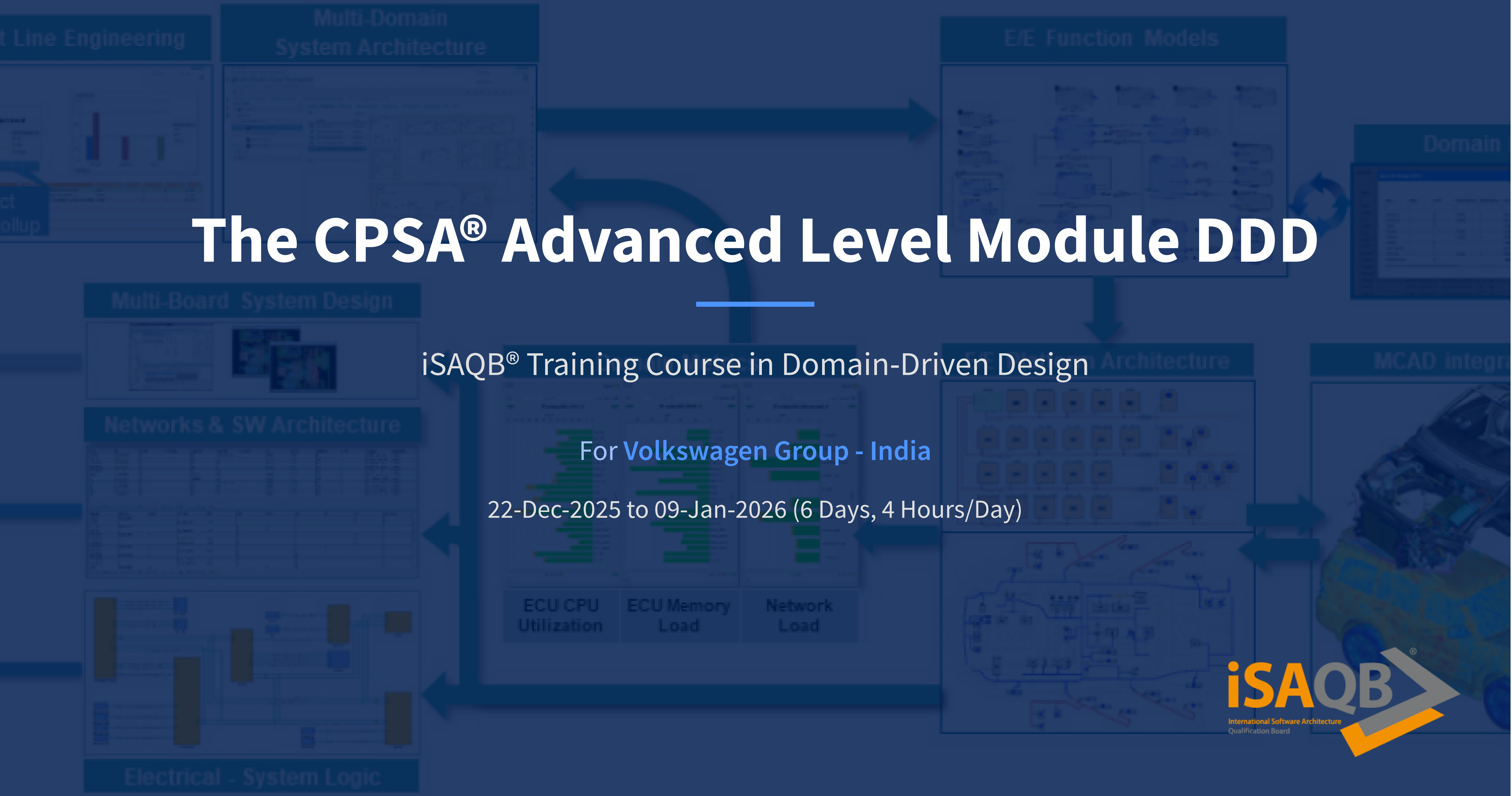
Next Steps

Apply Knowledge
Implement DDD concepts in your current Volkswagen projects

Share Learnings
Mentor colleagues and establish DDD practices in teams

Continue Learning
Explore advanced DDD patterns and automotive case studies

Pursue CPSA-A Certification Exam



The CPSA® Advanced Level Module DDD

iSAQB® Training Course in Domain-Driven Design

For **Volkswagen Group - India**

22-Dec-2025 to 09-Jan-2026 (6 Days, 4 Hours/Day)

Course Overview: Domain-Driven Design

What is DDD?

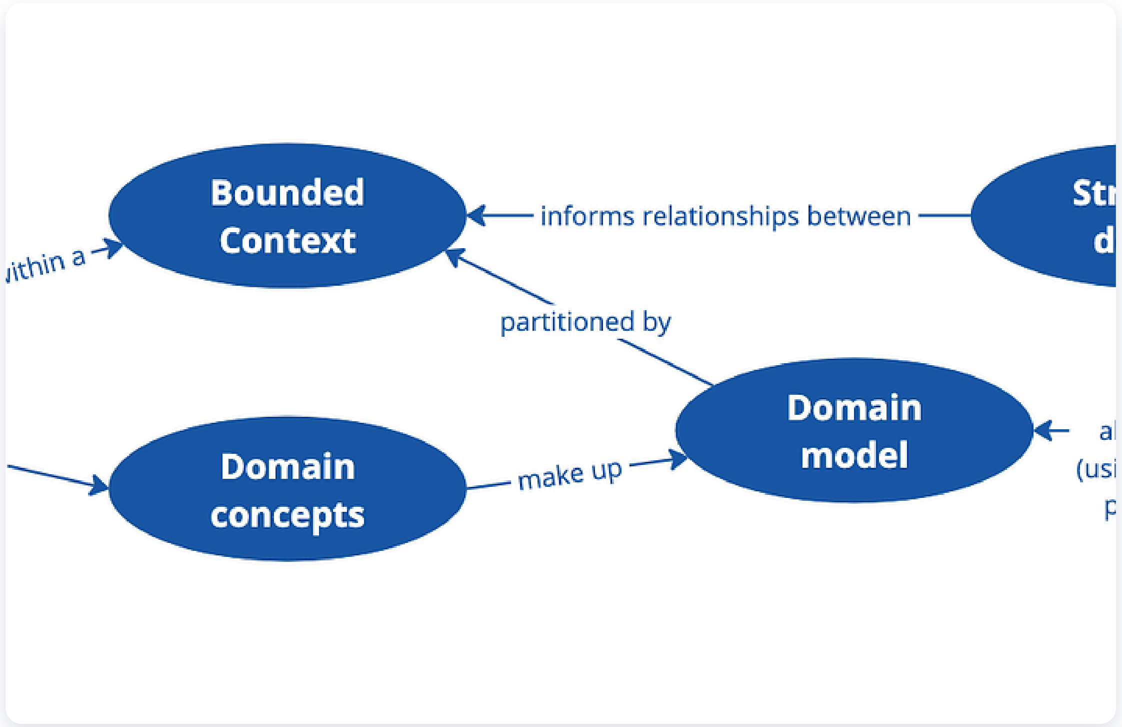
A strategic approach to software design that focuses on **complex business domains** and creates a shared understanding between technical teams and domain experts.

Why for Automotive?

Modern vehicles contain **100+ million lines of code** across multiple domains (infotainment, safety, powertrain), requiring precise modeling of complex interactions.

Value for Architecture

Creates **precise, transparent, and transformable** representations of domains, enabling better communication, reduced complexity, and more maintainable systems.



iSAQB Certification Pathway

Foundation Level

Completed ✓

Provides fundamental knowledge of software architecture principles, methods, and techniques.

- ✓ Architecture Fundamentals
- ✓ Quality Attributes
- ✓ Design Patterns

Advanced Level

Current Focus

Specialized modules for in-depth knowledge in specific areas of software architecture.

- DDD (Current)
- Cloud Architecture
- Security
- Embedded Systems
- And more...

Technical Competence	Methodical Competence	Communicative Competence	Total Credits
0	20	10	30

Training Schedule and Flow

6-Day Training Journey

4 hours per day • Building from foundations to strategic design

This training provides **24 hours** of instruction, exceeding the recommended **17 hours** for deeper dives, more exercises, and comprehensive coverage.

22-Dec-2025
Day 1
Foundations
Domain, Model & Ubiquitous Language

05-Jan-2026
Day 2
Knowledge Crunching
Collaborative modeling with domain experts

06-Jan-2026
Day 3
Implementation
From model to technical implementation

07-Jan-2026
Day 4
Architecture
Model in application architecture

08-Jan-2026
Day 5
Strategic Design 1
Cutting and distinguishing models

09-Jan-2026
Day 6
Strategic Design 2
Context Mapping

Foundations — Knowledge — Implementation — Architecture — Strategic

Day 1: Foundations - Domain, Model & Ubiquitous Language

1

22-Dec-2025

Foundations: Domain, Model & Ubiquitous Language

Learning Goals

LG 1-1
Explain connections between **domains, software, and models**

LG 1-2
Understand role of domain-specific terminology in building **ubiquitous language**

LG 1-3
Explain building blocks of DDD (Entities, Value Objects, Aggregates, etc.)

LG 1-4
Explain connections between building blocks

Concepts Covered

Domain & Domain Model
Software as representation of expert knowledge

Ubiquitous Language
Common terminology for experts and developers

Building Blocks
Entities, Value Objects, Aggregates, Services

Technical Components
Factories, Repositories, Domain Events

Activities & Skills

Activities

- Define a simple domain and its boundaries
- Create a ubiquitous language for a given domain
- Model basic domain concepts using class diagrams

Skills Being Built

DDD Foundations · Domain Modeling · Communication · Technical Translation

Day 2: Knowledge Crunching - The Path to the Model

2 05-Jan-2026

Knowledge Crunching: The Path to the Model

Learning Goals

- ✓ **LG 2-1 to 2-3**
  Empower **domain experts**, select suitable contacts, communicate effectively
- ✓ **LG 2-4 to 2-6**
  Use **modeling techniques**, conduct interviews, apply observation methods
- ✓ **LG 2-7 to 2-9**
  Overview of **Collaborative Modeling**, select approaches, conduct workshops
- ✓ **LG 2-10**
  Understand **agility** as foundation of DDD

Concepts Covered

- **Agile & Evolutionary Modeling**
  Iterative refinement of domain models
- **Domain Expert Empowerment**
  Leveraging expert knowledge effectively
- **Collaborative Modeling Methods**
  EventStorming, Domain Storytelling, User Story Mapping
- **Knowledge Elicitation**
  Interviewing, observation, field observation, apprenticing

Activities & Skills

Activities

- Workshop-style collaborative modeling sessions
- Role-playing interviews with domain experts
- Analyzing a domain through observation

Skills Being Built

Communication · Collaboration · Domain Analysis · Workshop Facilitation

Day 3: From Model to Implementation

3 06-Jan-2026

From Model to Implementation

Learning Goals

LG 3-1
Extend domain model with **technical building blocks** (Repositories, Factories)

LG 3-2
Model **interfaces** for domain classes

LG 3-3
Account for **interactions** between implementation and model

LG 3-4
Argue why DDD is worthwhile for **complex business logic**

Concepts Covered

Cohesion & Coupling
Principles for maintainable software design

SOLID Principles
Single Responsibility, Open/Closed, Liskov Substitution

Dependency Management
Avoiding cyclical dependencies, Law of Demeter

Technical Building Blocks
Repositories, Factories, Aggregates

Activities & Skills

Activities

- Refactoring domain model into technical components
- Designing interfaces and technical layers
- Debating benefits of DDD for complex logic

Skills Being Built

Technical Implementation

Design Principles

Refactoring

DDD Justification

Day 4: The Model in Application Architecture

4 07-Jan-2026

The Model in Application Architecture

Learning Goals

LG 4-1

Design a **ports & adapter architecture** for the domain model

LG 4-2

Formulate correlations and distinctions between **DDD and BDD**

Key Takeaway

Integrate domain model into larger system architecture effectively

Concepts Covered

Hexagonal Architecture

Ports & Adapters pattern for isolation

CQRS

Command-Query Responsibility Segregation

Layered Architecture

Traditional architectural approach

Dependency Injection

Inversion of Control for loose coupling

Activities & Skills

Activities

- Design hexagonal architecture for domain model
- Compare and contrast DDD and BDD approaches
- Map domain model to architectural layers

Skills Being Built

Architectural Design

System Integration

Pattern Application

DDD vs BDD

Day 5: Strategic Design 1 - Cutting and Distinguishing Models

5

08-Jan-2026

Strategic Design 1: Cutting and Distinguishing Models

Learning Goals

- **LG 5-1 to 5-2**
  Identify **symptoms** of large models, assess **cross-team models**
- **LG 5-3**
  Move from **problem to solution space**
- **LG 5-4**
  Distill the **core** of a system
- **LG 5-5**
  Describe **Bounded Contexts** in a Context Map

Concepts Covered

- **Problem Space vs Solution Space**
  Distinguishing domain problems from technical solutions
- **Subdomain Classification**
  Core, Supporting, Generic subdomains
- **Bounded Context**
  Explicit boundaries for domain models
- **Context Map**
  Visualizing relationships between contexts

Activities & Skills

Activities

- Analyze large system and identify problems
- Define Bounded Contexts and relationships
- Create Context Map for complex system

Skills Being Built

Strategic Thinking

System Decomposition

Context Mapping

Architectural Vision

Day 6: Strategic Design 2 - Context Mapping

6 09-Jan-2026

Strategic Design 2: Context Mapping

Learning Goals

✓ **LG 6-1 to 6-2**
Use interfaces for **customer/supplier teams**, design **Open Host Service**

✓ **LG 6-3 to 6-4**
Isolate model with **Anticorruption Layer**, reuse elements in **Shared Kernel**

✓ **LG 6-5**
Understand when to divide models with **Separate Ways**

✓ **LG 6-6**
Use **Domain Events** for communication between contexts

Concepts Covered

Customer/Supplier
Relationships between upstream and downstream contexts

Open Host Service (OHS)
Public API for multiple client contexts

Anticorruption Layer
Translation layer between contexts

Domain Events
Asynchronous communication between contexts

Activities & Skills

Activities
- Design inter-context communication strategies
- Map system integrations using context patterns
- Design Anticorruption Layer for external systems

Skills Being Built

Inter-context Design

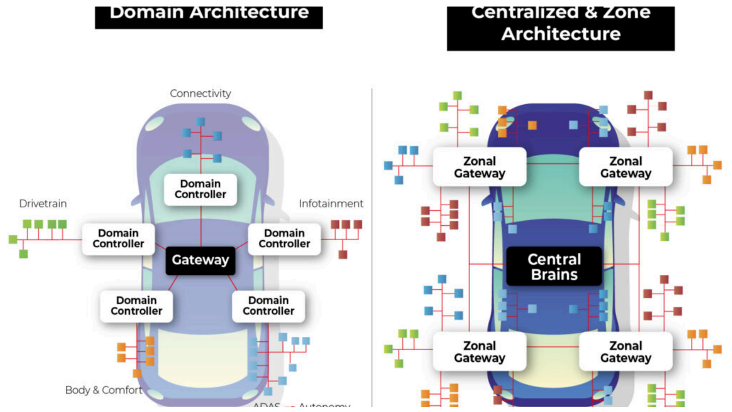
Integration Strategies

Communication Patterns

System Boundaries

Connecting DDD to the Automotive Domain and Volkswagen Projects

Automotive Software Complexity



Growing Complexity

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

Multiple Domains

Infotainment, powertrain, safety, connectivity, autonomous driving

Cross-team Collaboration

Different teams work on different vehicle subsystems

DDD Applications in Automotive

Bounded Contexts

Separate contexts for **infotainment**, **powertrain**, **safety**, **connectivity**

Context Mapping

Customer/Supplier relationships between safety and powertrain

Domain Events

Event-driven communication between vehicle subsystems

Ubiquitous Language

Common terminology for engineers, developers, and domain experts

Applying to Volkswagen Projects

Project Integration

Apply DDD to your specific Volkswagen projects

Key Applications

- Infotainment Systems
- EV Battery Management
- Autonomous Driving
- Connected Services
- Fleet Management

> *"DDD helps Volkswagen teams create clear boundaries between vehicle subsystems while ensuring effective communication between them."*

Benefits

Reduced complexity, better maintainability, improved collaboration between teams

End Outcomes & Next Steps

Skills Acquired

Strategic Design
Decompose complex systems into manageable contexts

Tactical Implementation
Translate domain models into technical solutions

Collaboration
Bridge communication between domain experts and developers

Domain Modeling · Context Mapping · Event-Driven Design · Hexagonal Architecture

Value for Volkswagen Projects

Automotive Software Excellence
Design complex vehicle systems with clear boundaries

Faster Development
Reduce integration complexity between vehicle subsystems

Better Decision Making
Strategic approach to system architecture decisions

Cross-Team Collaboration
Effective communication between specialized teams

Next Steps

Apply Knowledge
Implement DDD concepts in your current Volkswagen projects

Share Learnings
Mentor colleagues and establish DDD practices in teams

Continue Learning
Explore advanced DDD patterns and automotive case studies

Pursue CPSA-A Certification Exam