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

Introduction to Shared Understanding

Communication Challenges

Lost in Translation

- **Different vocabularies** between teams
- **Misinterpreted requirements** due to terminology
- **Implementation gaps** from misunderstood concepts
- **Endless clarification** cycles

Automotive Impact

Real-World Consequences

- **Feature delays** from misunderstood requirements
- **Integration failures** between vehicle systems
- **User experience issues** from inconsistent terminology
- **Safety risks** from ambiguous specifications

> **"The most disastrous thing that can happen to a software project is to have wrong people making key decisions."**
>
> *- Eric Evans, Author of Domain-Driven Design*

Automotive Software Complexity

Software Complexity Metrics

100M+

Lines of Code

150+

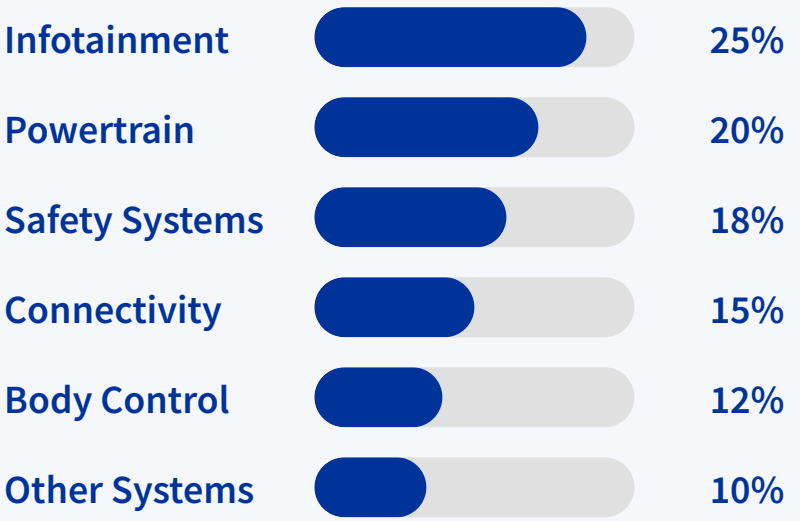
ECUs in Premium Vehicles

Multiple Domains

- **Powertrain** - Engine, transmission, battery management
- **Infotainment** - Media, navigation, connectivity
- **Safety** - ADAS, airbags, collision avoidance
- **Connectivity** - OTA updates, V2X communication

Domain Distribution

Software Code Distribution



Cross-Team Challenges

- **Integration complexity** between domains
- **Communication barriers** between specialized teams
- **Evolving requirements** across vehicle systems

Introduction to Ubiquitous Language

What is Ubiquitous Language?

Shared Vocabulary

- **Common terminology** between domain experts and developers
- **Precise meanings** for domain concepts
- **Consistent usage** across code and documentation
- **Evolution** through collaboration

Why It Matters

Key Benefits

- **Reduces ambiguity** in requirements
- **Improves collaboration** between teams
- **Creates alignment** between business and technical
- **Simplifies maintenance** of complex systems

In Automotive Software

Powertrain | Infotainment | Safety

Learning Goals

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

Key Concepts of Ubiquitous Language

Bounded Context

- **Explicit boundary** where a specific domain model applies
- **Linguistic boundary** for consistent terminology
- **Team alignment** with domain ownership

Shared Vocabulary

- **Common terms** used by all stakeholders
- **Precise meanings** for domain concepts
- **Consistent usage** in code and documentation

Language Evolution

- **Iterative refinement** through collaboration
- **Living documentation** that evolves with understanding
- **Versioned glossary** for terminology management

Automotive Example

Powertrain **Infotainment** **Safety** **Connectivity**

Creating and Evolving Ubiquitous Language

Evolution Process

Iterative Refinement

1. **Gather Terms**
   Interview domain experts, collect existing terminology
2. **Refine Meanings**
   Discuss with stakeholders, resolve ambiguities
3. **Model Concepts**
   Apply refined language to domain model
4. **Apply Consistently**
   Use in code, documentation, and discussions

Best Practices

Language Development

- **Collaborative Creation** - Involve all stakeholders
- **Document Glossary** - Centralized terminology reference
- **Consistent Usage** - Enforce in code and documentation
- **Regular Refinement** - Schedule periodic reviews

Automotive Success Factors

- Cross-Team Workshops
- Living Documentation
- Code-First Vocabulary
- Iterative Evolution

Analogies for Ubiquitous Language

Medical Field

- **Precise terminology** for body parts, symptoms, treatments
- **Universal understanding** between doctors, nurses, technicians
- **Reduced errors** from miscommunication

Legal Field

- **Specific language** for contracts, clauses, precedents
- **Shared vocabulary** across legal teams
- **Clear documentation** with precise terms

Scientific Research

- **Standardized terminology** for methods, results, conclusions
- **Peer review** with common language
- **Knowledge transfer** through precise communication

DDD Connection

Just as specialized fields develop precise terminology, **ubiquitous language** creates shared understanding between domain experts and developers in software

Automotive Examples of Ubiquitous Language

Powertrain

- **Torque**
  Rotational force, not "power" or "strength"
- **GearRatio**
  Precise ratio, not "gear setting"
- **ThrottlePosition**
  Position value, not "acceleration"

Infotainment

- **MediaSource**
  Source type, not "player" or "format"
- **Playlist**
  Ordered collection, not "song list"
- **NavigationRoute**
  Complete path, not "directions" or "map"

Safety Systems

- **SafetyZone**
  Defined area, not "safe space"
- **AlertLevel**
  Severity level, not "warning" or "alarm"
- **CollisionEvent**
  Specific occurrence, not "crash" or "impact"

Case Study: Mercedes-Benz MBUX System

MBUX Approach

Consistent Language

- **Unified terminology** across all interfaces
- **User-centric vocabulary** for all features
- **Shared understanding** between UX and engineering

DDD Implementation

Model Elements

- **UserInterfaceSession** - Tracks interaction state
- **MediaMetadata** , **Coordinates** - Value objects
- **InfotainmentAggregate** - Manages connected features

Integration Benefits

- **Clear boundaries** between vehicle systems
- **Consistent terminology** across vehicle functions
- **Simplified maintenance** through clear concepts

> **"MBUX demonstrates how a consistent, user-centric language creates a unified experience across complex vehicle systems."**
>
> *- Automotive UX Analysis*

Reverse Engineering: Audi MMI Interface

MMI System Analysis

Language Consistency

- **Media Control** - Consistent terminology across audio sources
- **Navigation** - Standardized route and destination terms
- **Connectivity** - Unified phone integration vocabulary

DDD Implementation

Model Elements

- **UserInterfaceSession** - Tracks interaction state
- **MediaMetadata**, **Coordinates** - Value objects
- **InfotainmentAggregate** - Manages connected features

Integration Patterns

- **Bounded Contexts** for different MMI modules
- **Consistent terminology** across MMI interfaces

Audi's MMI demonstrates clear separation of concerns with consistent terminology across infotainment, navigation, and vehicle controls

Brainstorming Puzzles: Creating Ubiquitous Language

1 EV Charging System

A system with multiple teams working on charging, payment, and battery management. Each team uses different terms for "charging session."

Create Ubiquitous Language

ChargingSession | PaymentTransaction | BatteryState

2 Vehicle Safety Features

Safety engineers use "brake assist" while UX designers use "emergency stop" for the same feature.

Resolve Terminology

AutomaticEmergencyBraking | CollisionAvoidance | ProximityDetection

3 Infotainment Controls

Users interact with media, navigation, and climate control through different interfaces with inconsistent terminology.

Unify Interface Language

UserInteraction | ControlMode | SystemState

Scenarios and Solutions: When Terminology Conflicts Arise

Cross-Team Conflicts

Different teams using inconsistent terminology for same concepts

Solution Approach

- **Glossary creation** with clear definitions
- **Regular alignment** meetings between teams
- **Bounded contexts** for different domains

Evolving Requirements

Business terminology changing during development

Solution Approach

- **Versioned glossary** with change tracking
- **Living documentation** that evolves with language
- **Domain expert workshops** for terminology refinement

Technical vs. Business

Developers using technical terms that business users don't understand

Solution Approach

- **Translation layer** between technical and business terms
- **User-facing documentation** with business terminology
- **Code comments** mapping technical to business terms

Self-Study Resources for Ubiquitous Language

Books

Domain-Driven Design
Eric Evans - **Ubiquitous Language** chapter

Implementing DDD
Vaughn Vernon - **Bounded Contexts** examples

DDD Distilled
Jimmy Nilsson - **Strategic Design** and language

Articles

Ubiquitous Language in Practice
Alberto Brandolini - **Practical examples** and patterns

Strategic Domain-Driven Design
Eric Evans - **Language patterns** and context mapping

EventStorming Guide
DDD-Crew - **Collaborative modeling** techniques

Online Resources

DDD Community
Forums, discussions, and **language examples**

Glossary Tools
Software for **creating and managing** ubiquitous language

Workshop Templates
Guides for **language creation** workshops

Practical Exercises

Terminology Mapping
Map conflicting terms between teams

Glossary Creation
Build shared vocabulary for a domain

Language Evolution
Document changes to terminology over time

Connecting to Day 2

Day 2 Topics

Knowledge Crunching

- **Working with domain experts**
- **Collaborative modeling** techniques
- **EventStorming** and Domain Storytelling

Self-Study Preparation

Key Concepts

- **Domain expert empowerment**
- **Communication models** between teams
- **Knowledge elicitation** techniques

Preparation Activities

- **Document existing terminology in your project**
- **Identify conflicting terms between teams**
- **Practice collaborative modeling techniques**

Summary of Ubiquitous Language

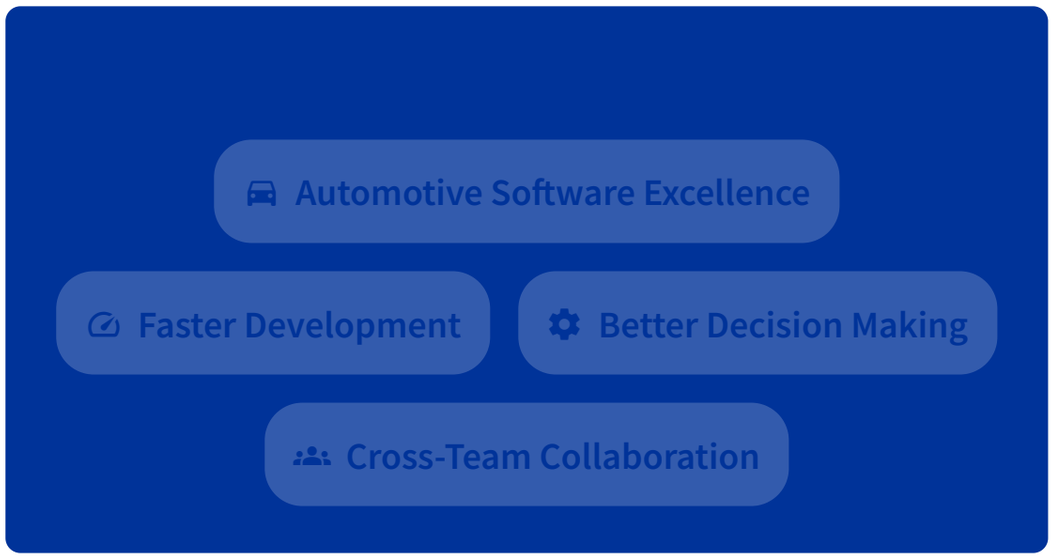
Key Takeaways

Shared Understanding

- **Common vocabulary** bridges communication gaps
- **Precise meanings** reduce ambiguity
- **Consistent usage** in code and documentation

Automotive Impact

- **Clear boundaries** between vehicle systems
- **Consistent terminology** across interfaces
- **Simplified maintenance** through clear concepts



Summary of Ubiquitous Language

Key Takeaways

Shared Understanding

- check_circle **Common vocabulary** bridges communication gaps
- check_circle **Precise meanings** reduce ambiguity
- check_circle **Consistent usage** across code and documentation

Automotive Impact

- check_circle **Clear boundaries** between vehicle systems
- check_circle **Consistent terminology** across interfaces
- check_circle **Simplified maintenance** through clear concepts

Connection to DDD

Building Blocks

- check_circle **Ubiquitous language** defines domain concepts
- check_circle **Bounded contexts** establish linguistic boundaries
- check_circle **Domain events** communicate across contexts

Value for Volkswagen Projects

Automotive Software Excellence

Faster Development

Better Decision Making

Cross-Team Collaboration