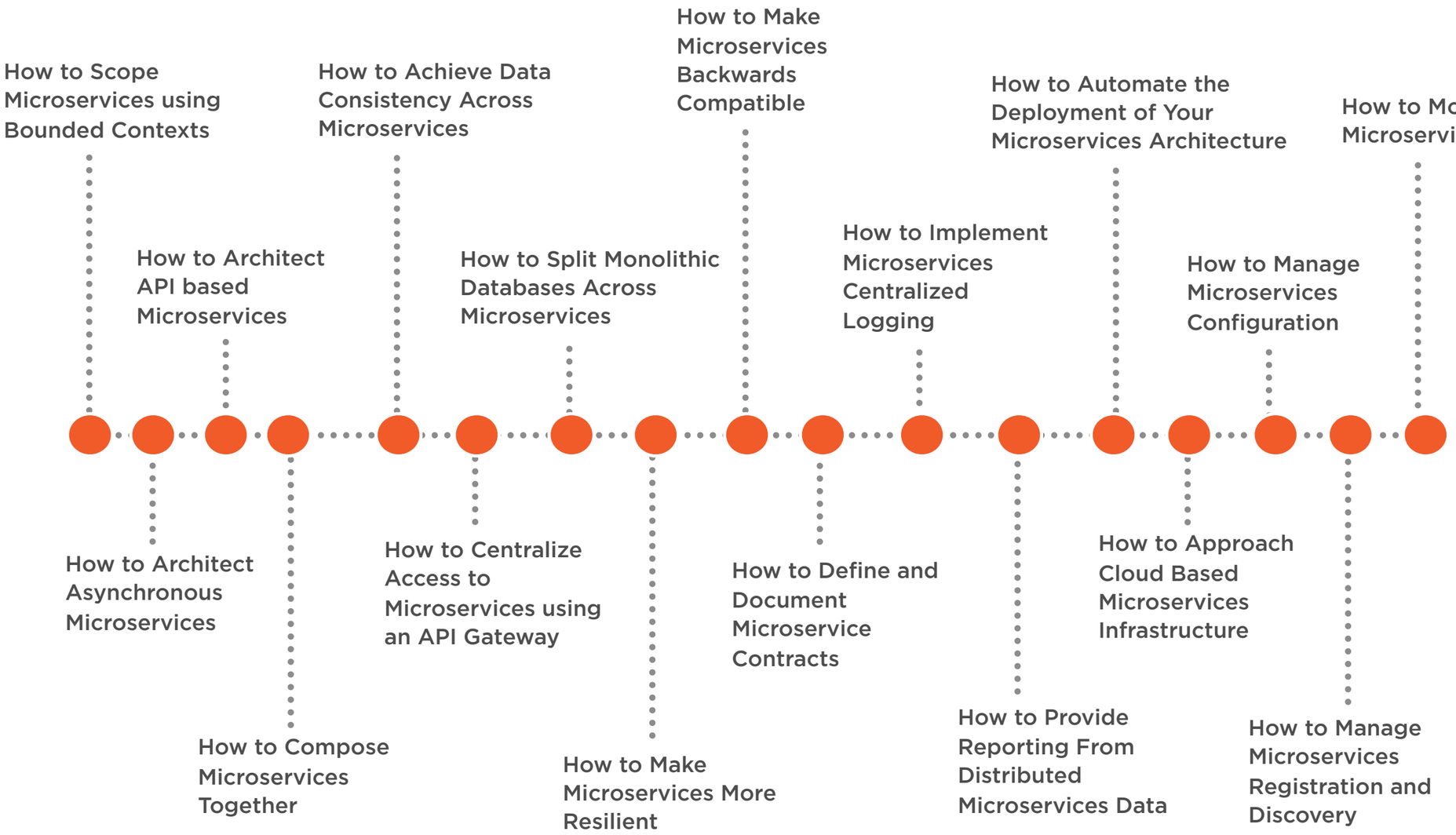
Microservices Architectural Design Patterns Playbook

HOW TO SCOPE MICROSERVICES USING BOUNDED CONTEXTS

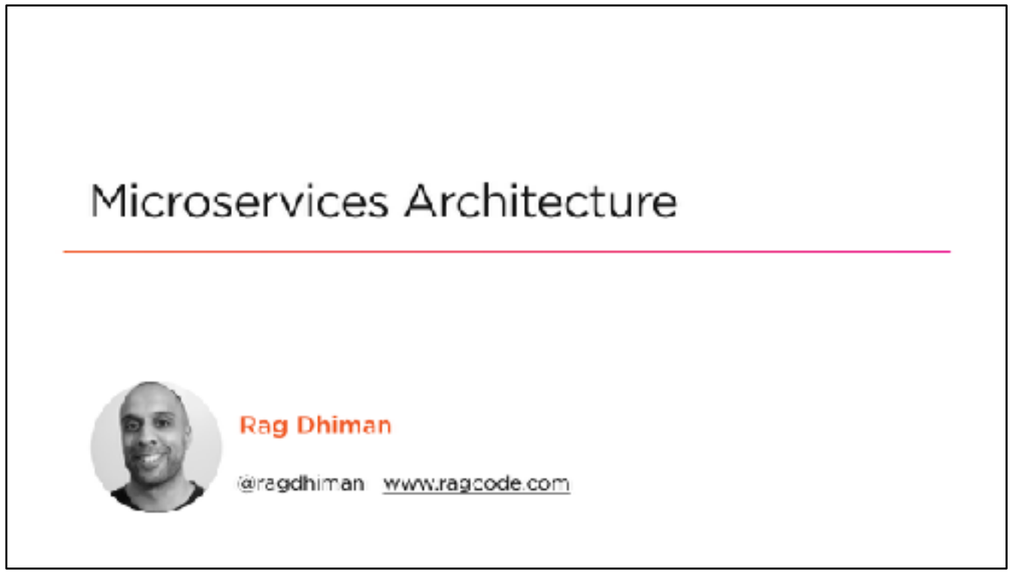
Rag Dhiman

@ragdhiman www.ragcode.com

Microservices Architectural Design Patterns Playbook



Microservices Architectural Design Patterns Playbook



Overview

Introduction

- Bounded Context
- Ubiquitous Language

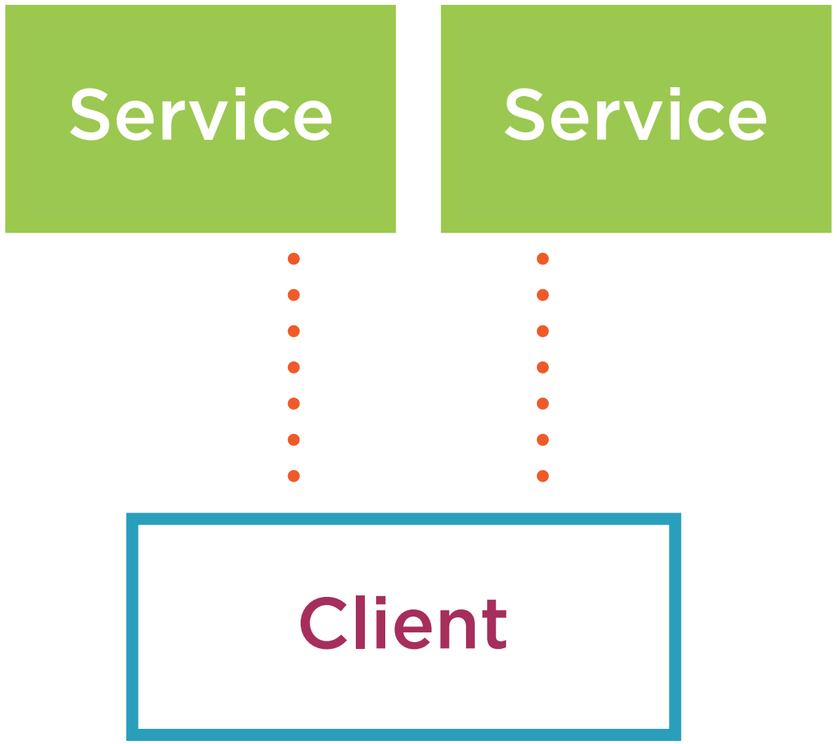
Unbounded Approach to Microservices

Using Bounded Contexts for Microservices

Aggregation

Introduction

Introduction



Competitive software

What is micro?

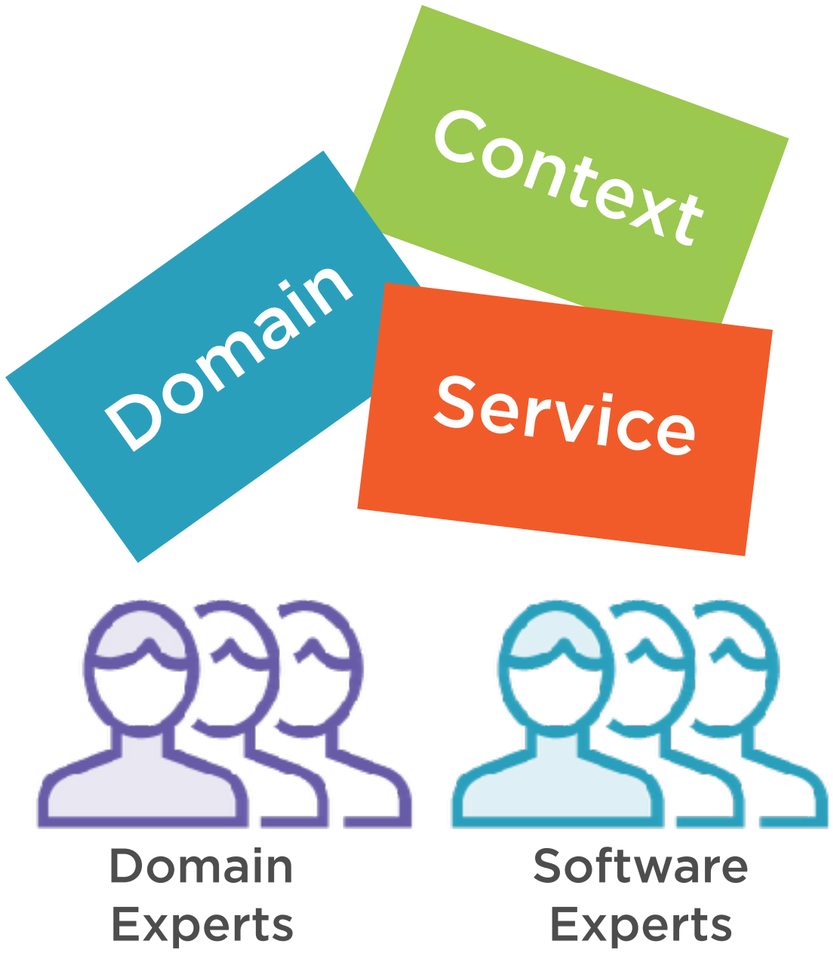
Design vs. scrum

Architects and developers

Bounded context for scope

- Domain Driven Design
- Eric Evans
- Strategic design pattern
- Tool for the whole process

Domain Driven Design



Software that models real-world domains

- Domain experts and software developers

Many tools and techniques

Concept of bounded contexts

Domain consists of multiple bounded contexts

- Each BC represents a domain function

Bounded context for design encourages:

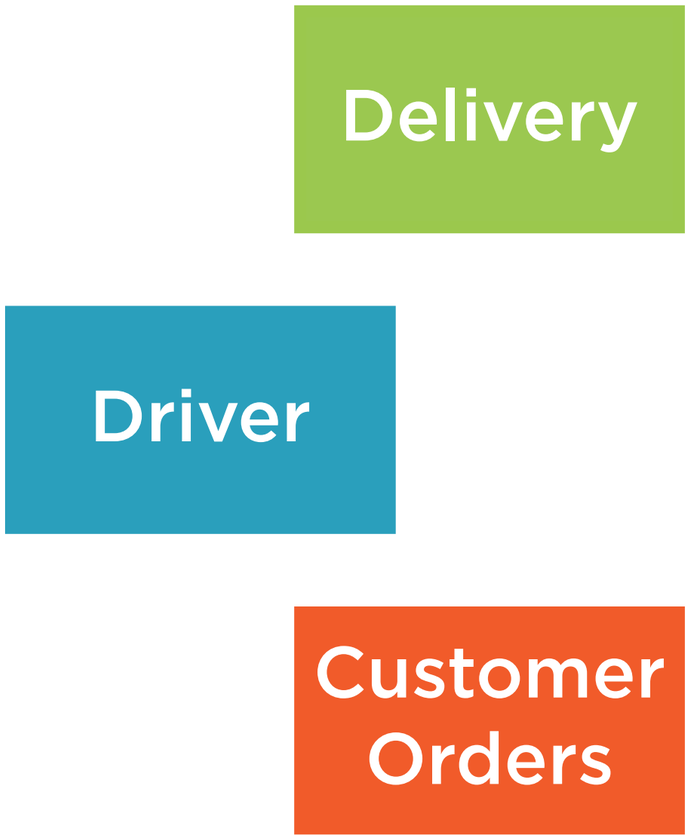
- Loose coupling

- High cohesion

Bounded Context

A specific responsibility enforced by an explicit boundary

Bounded Context



Starts off as a core domain concept

Internal models (supporting concepts)

Each BC has an explicit interface

Shared models for communication

Microservice = Bounded context

- Belongs to a team
- Own repository and data store
- Contracts

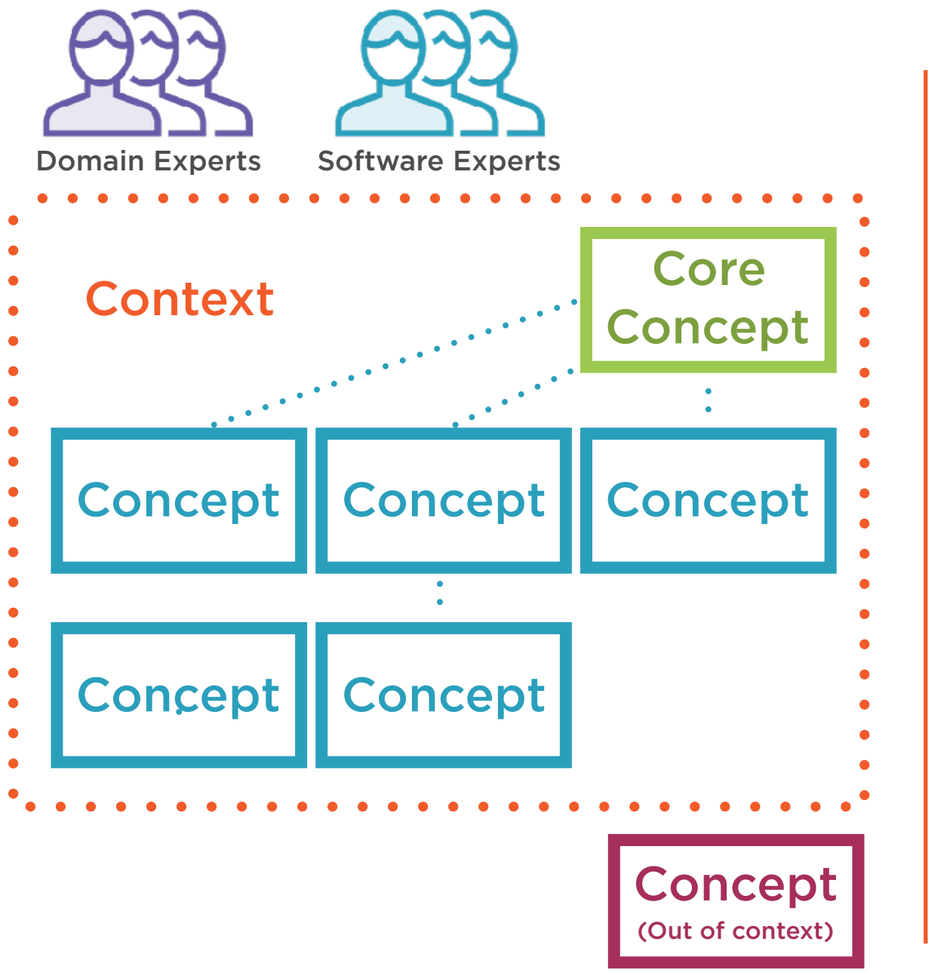
Logical bounded context

Out of context

Ubiquitous Language

Language that belongs to a specific domain function (bounded context). Also used by all team members to connect all the activities of the team with the software.

Bounded Context and Ubiquitous Language



Core concept defines the language

- Ubiquitous language
- Natural core language

Used as a bounded context filter

- Concepts in context
- Concepts out of context

How to define the language

- Domain experts
- Software developers

Unbounded Approach to Microservices

Core Aspects of Your Domain

Driver

Transport

Shifts

Annual leave

Customer Orders

Orders

Returns

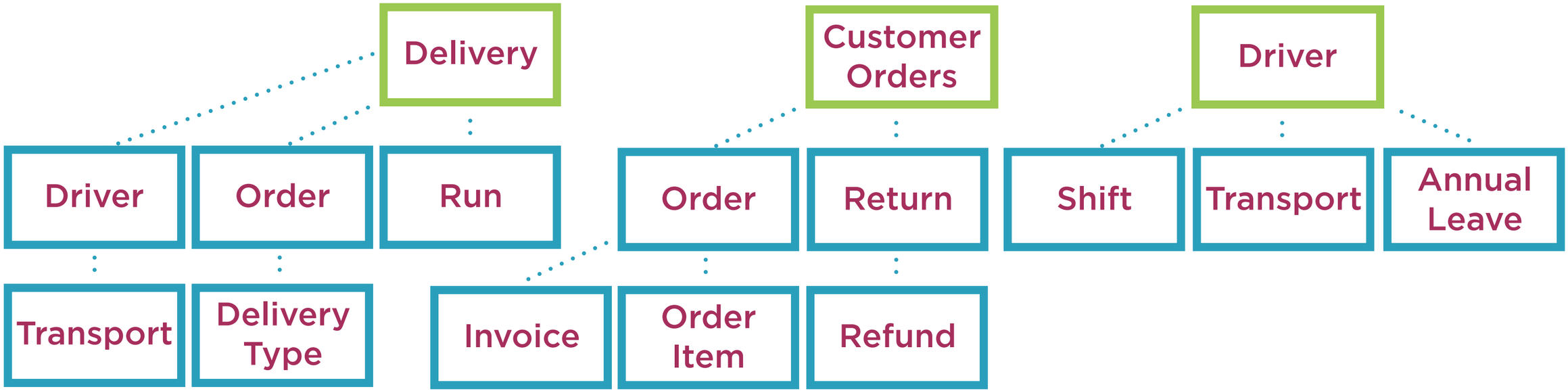
Delivery

Drivers

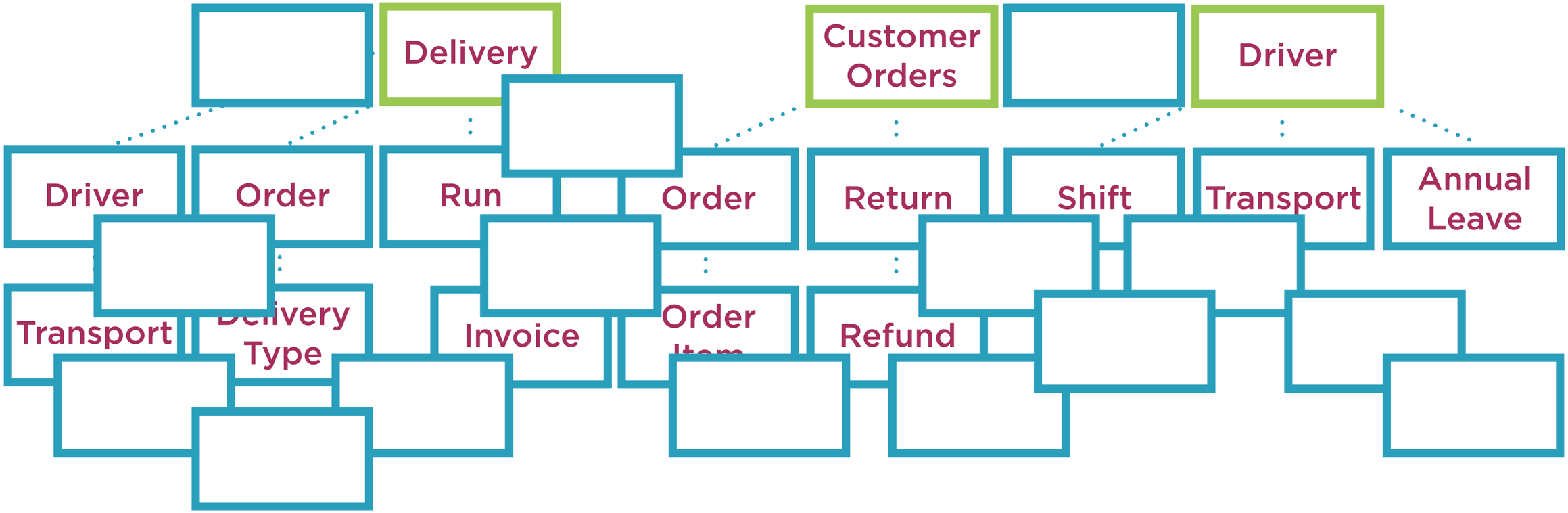
Orders

Runs

Core Concepts with Supporting Concepts

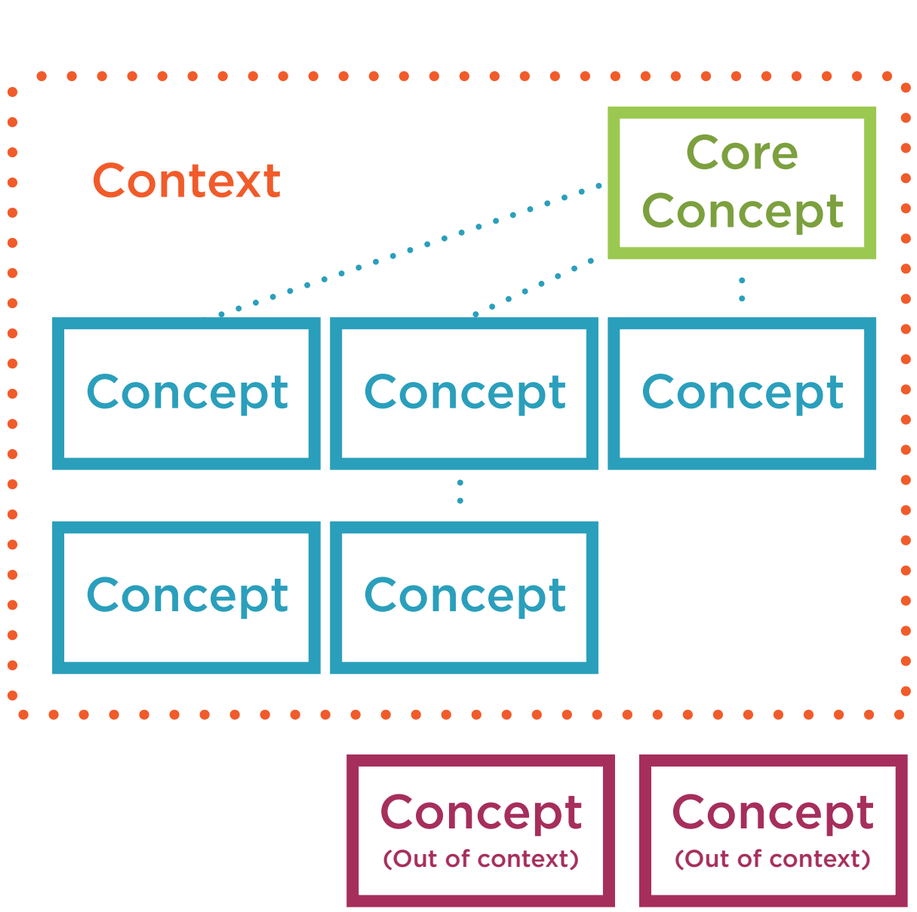


Unbounded Context



Using Bounded Contexts for Microservices

Bounded Contexts as a Technique



Key aspects\concepts of the domain

Concepts form bounded contexts

Core concept forms ubiquitous language

Rename supporting concepts\models

- Rename to natural UL terms

Move out of context concepts\models

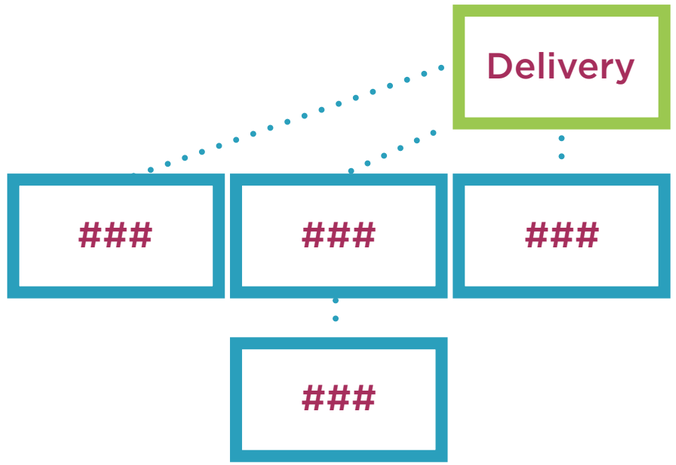
- Not part of the ubiquitous language

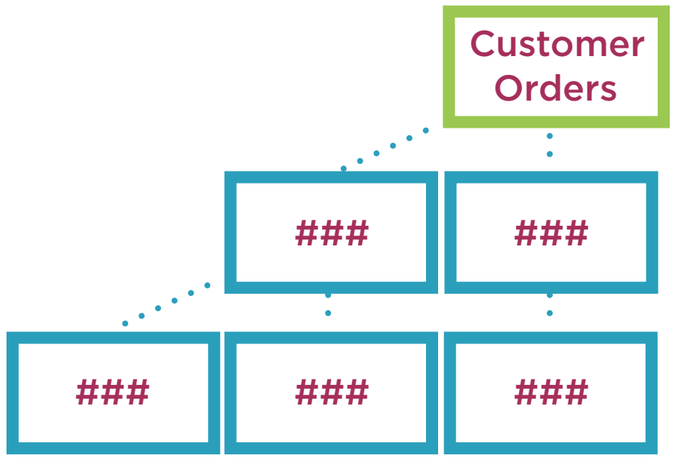
Remove out of scope concepts

Single concepts indicate integration

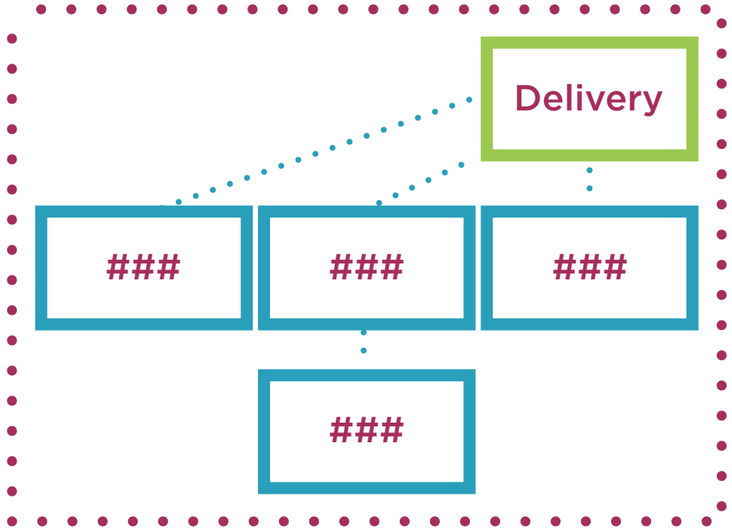
- Integration with other bounded contexts

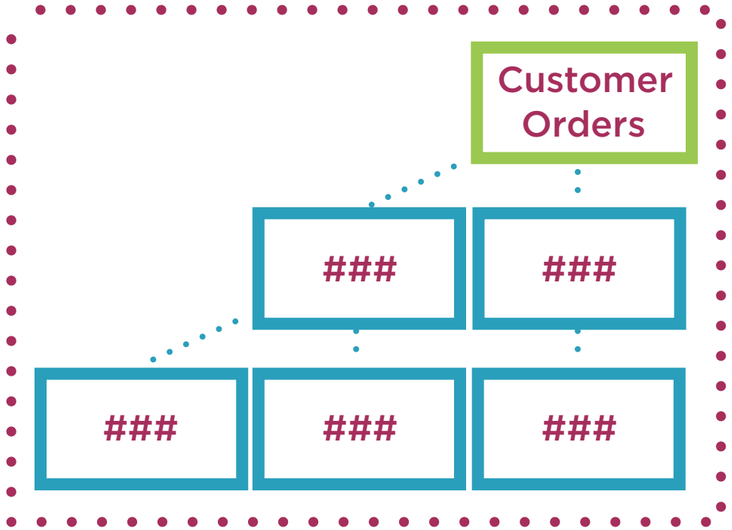
Identifying the Core?



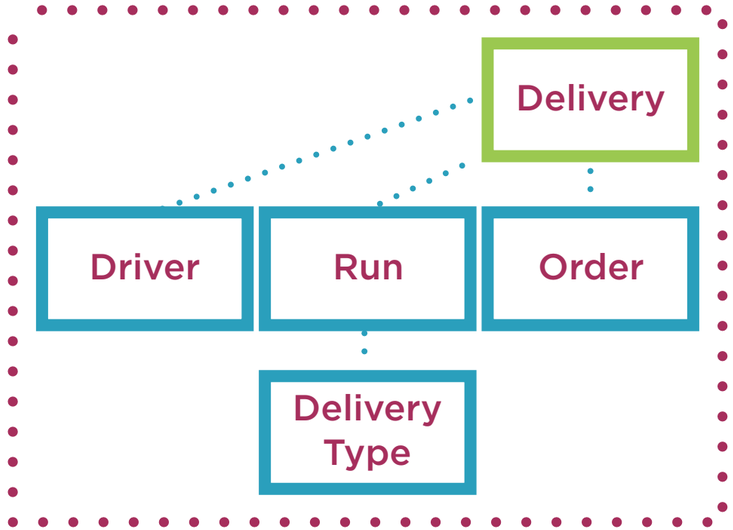


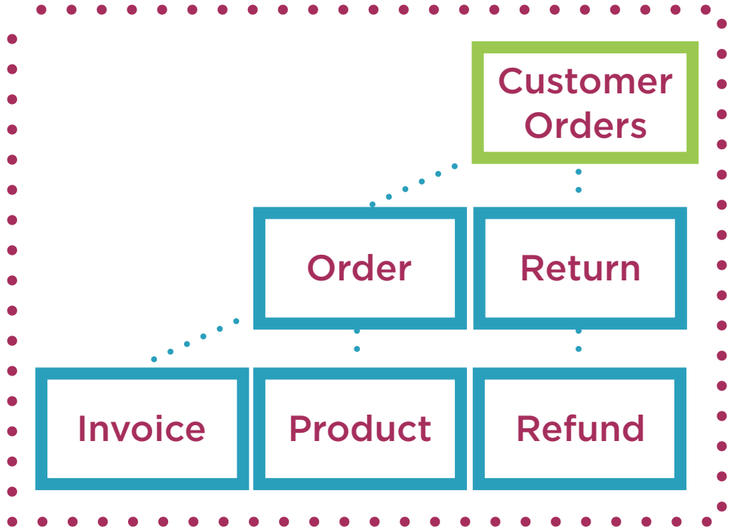
Identifying The Core?



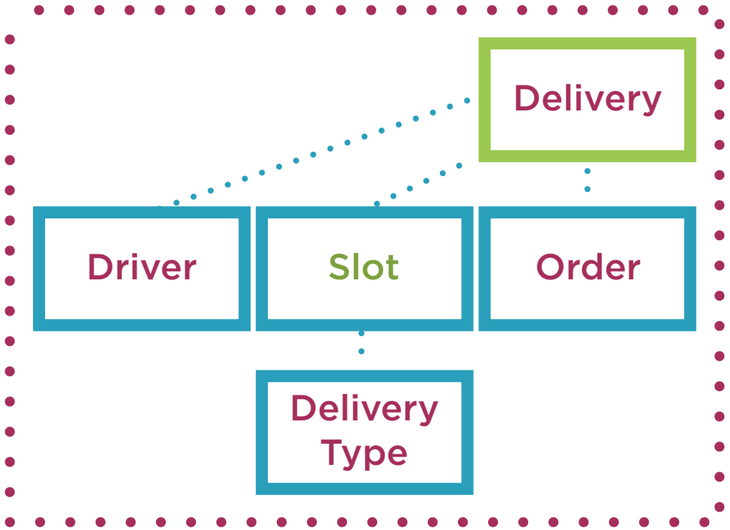


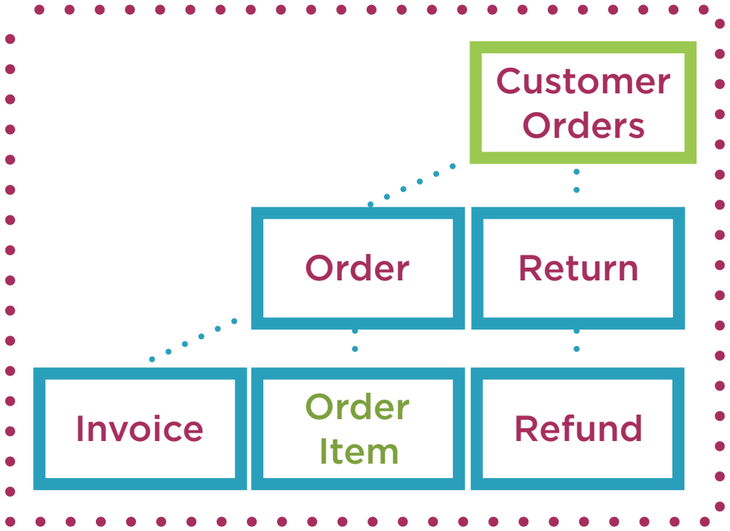
Core Defines the Ubiquitous Language



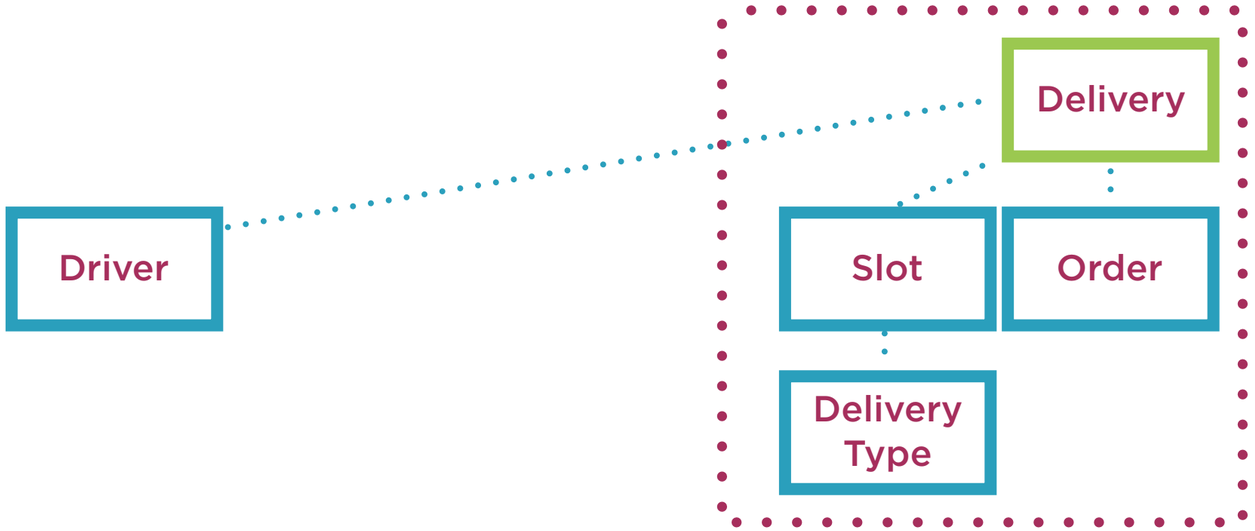


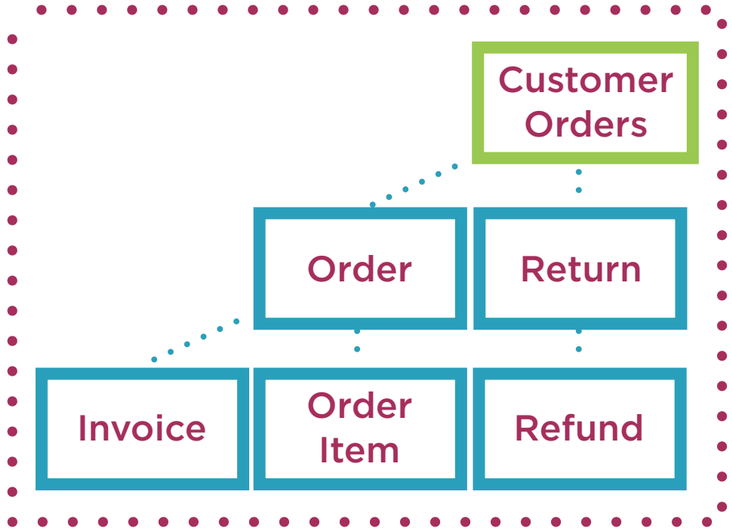
Rename Concepts



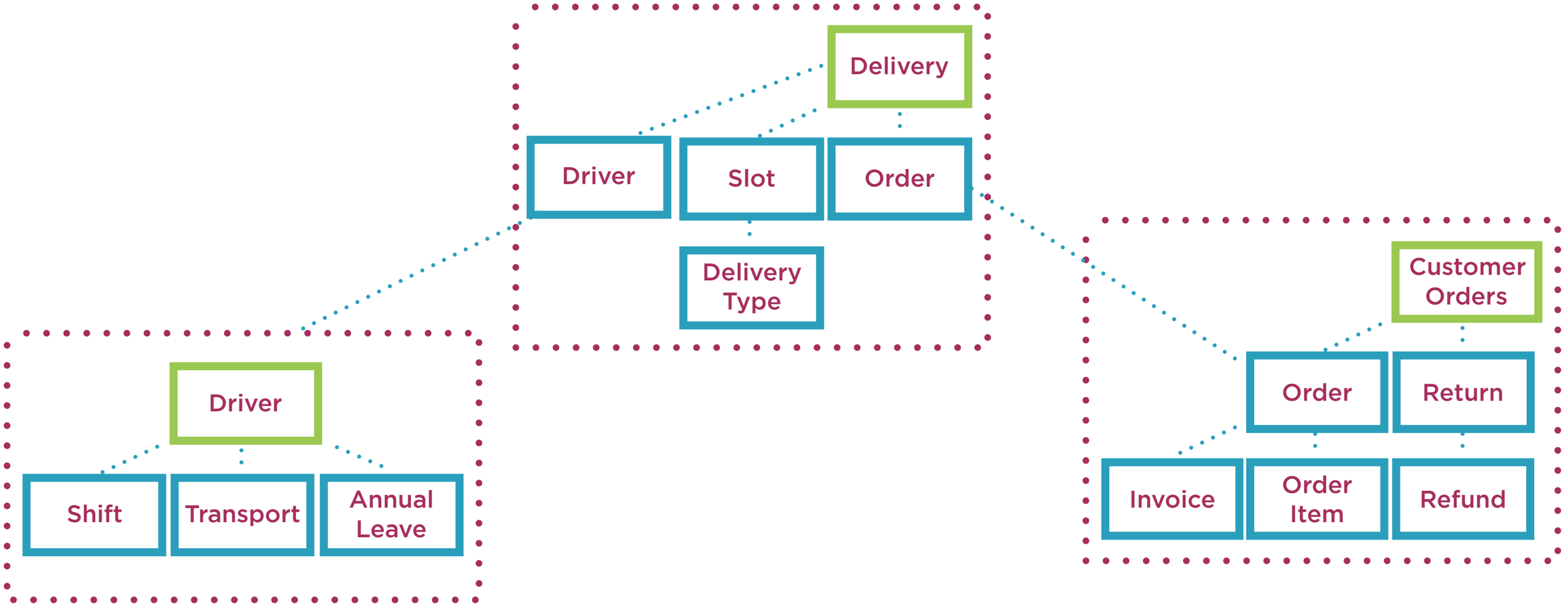


Out of Context Concepts

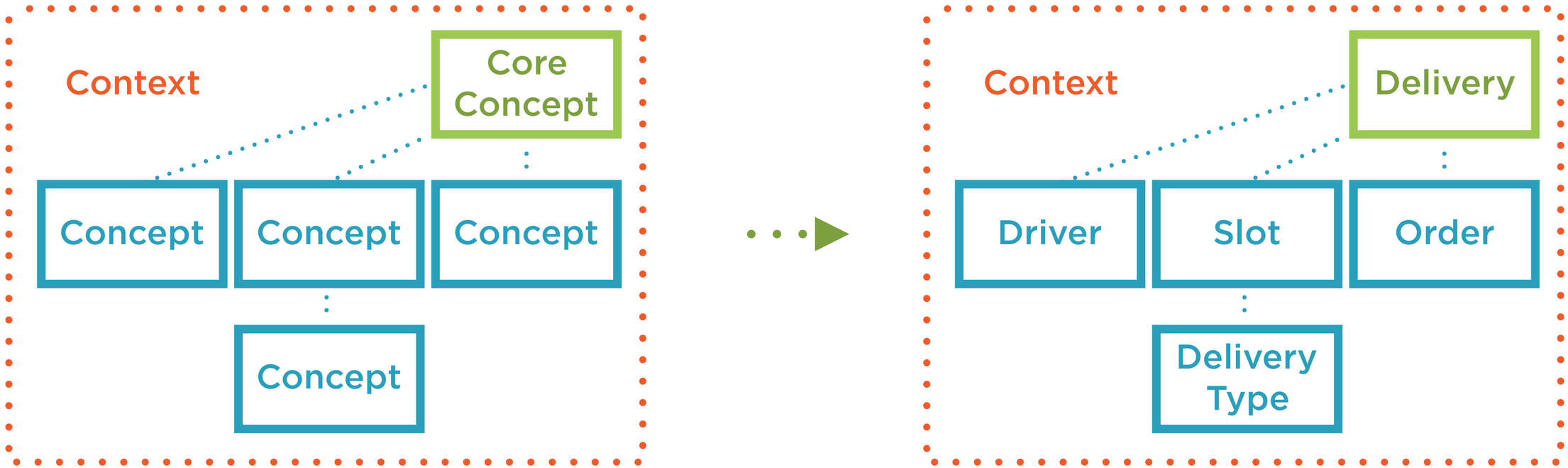




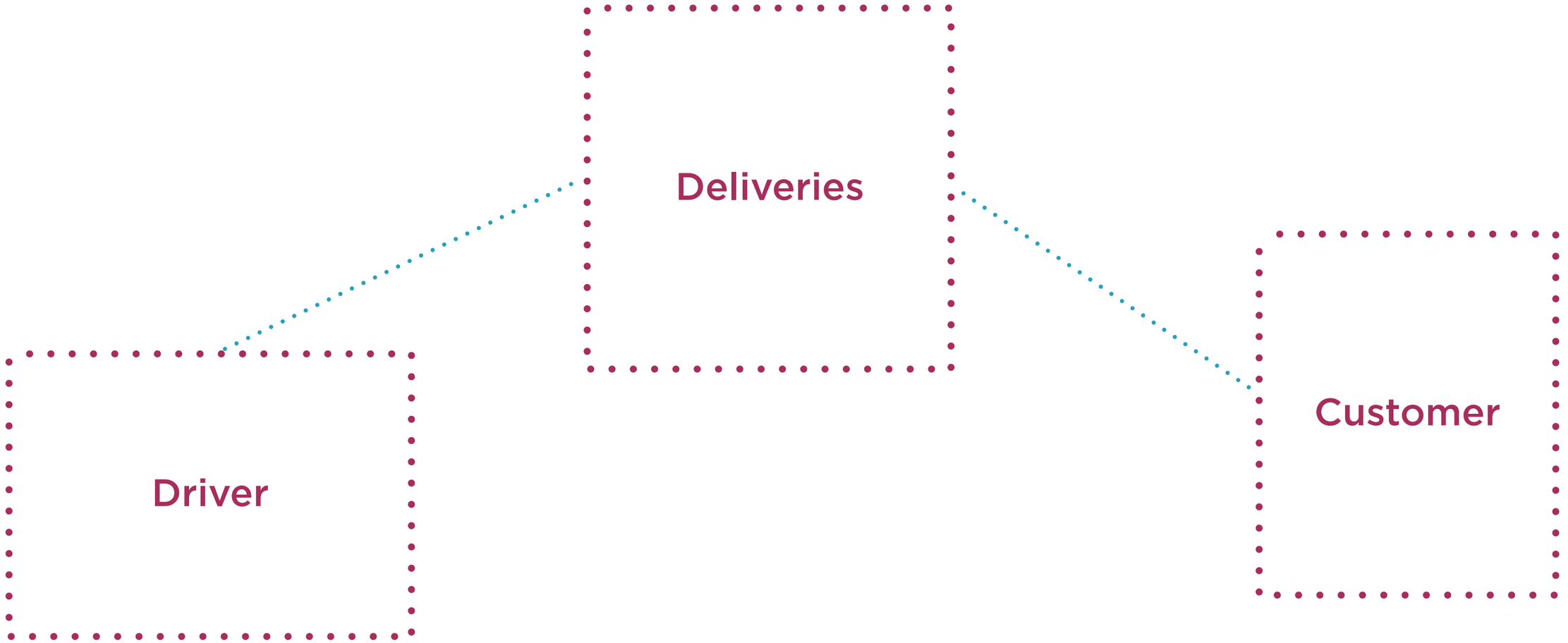
Single Concepts Indicate Integration



Bounded Context to Microservice

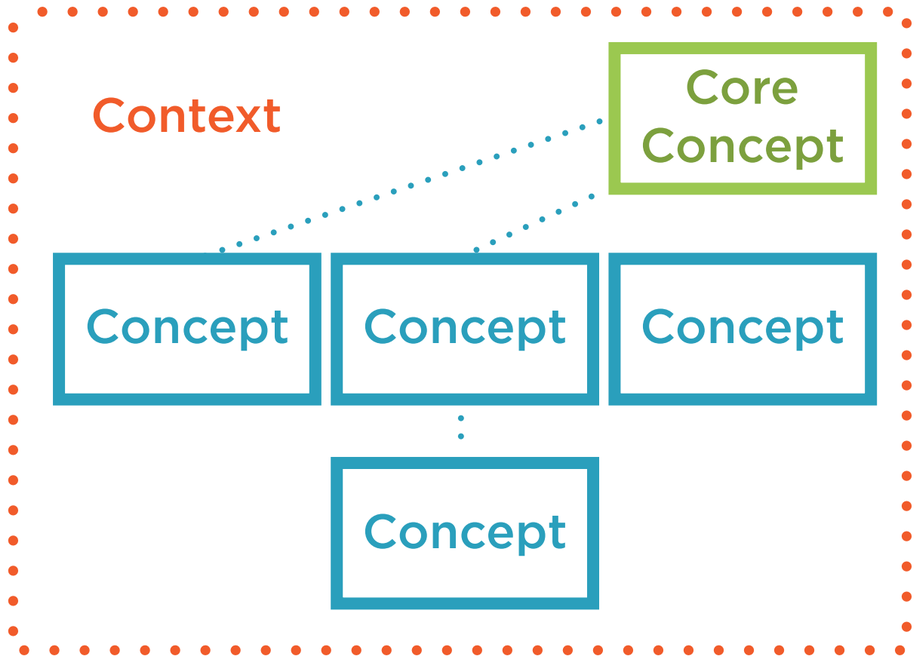


Bounded Contexts Become Microservices



Aggregation

Aggregation



Combining services

Used in addition to decomposing

- Bounded context method

Reasons for

- Reporting
- Enhanced functionality
- Usability for clients
- Performance

When to decompose or aggregate?

Summary

Introduction

- Bounded Context
- Ubiquitous Language

Unbounded Approach to Microservices

Using Bounded Contexts for Microservices

Aggregation

Microservices Architectural Design Patterns Playbook