MODEL EXPLORATION
Whirlpool

challenge model with new scenario

SCENARIO

- Tell a story: be concrete & specific
- Walk through and flesh out
- Refocus on hard part
- Refocus on core domain

Discover
the domain visually & collaboratively

Strategize
business-differentiating core-domains

Medium

Harvest & Document

- Collect reference scenarios
- Capture bits of model, with rationale
- Leave most ideas behind

Understand
business model and user needs

Discover
the domain visually & collaboratively

Coarse

CODE PROBE

Code
your bounded contexts with tactical patterns

- Experiment in code
- Refine language
- Add rigor
- Code cheap prototypes
- Script a scenario
- Try, then scrap, code changes

MODEL

Fine

Discover
the domain visually & collaboratively

Connect
sub-domains to form a loosely-coupled architecture

Valid with multiple sub-domains

- Propose a model
- Walk through scenarios as events & states
- Explore language
- Add rigor
- Make mistakes

domainlanguage.com/ddd/whirpool

NAME	STATE TRANSITIONS
1	

DESCRIPTION	
2	3

THROUGHPUT	AVG	MAX	ENFORCED INVARIANTS	CORRECTIVE POLICIES
COMMAND HANDLING RATE				
TOTAL NUMBER OF CLIENTS				
CONCURRENCY CONFLICT CHANCE		8	4	5

SIZE	AVG	MAX	HANDLED COMMANDS	CREATED EVENTS
EVENT GROWTH RATE				
LIFETIME OF A SINGLE INSTANCE				
NUMBER OF EVENTS PERSISTED		9	6	7

Name:

V5
github.com/ddd-crew/bounded-context-canvas

Purpose

What benefits does this context provide, and how does it provide them? Describe the purpose from a business perspective

Strategic Classification

Domain
- core
- supporting
- generic
- other?

Business Model
- revenue
- engagement
- compliance
- cost reduction

Evolution
- genesis
- custom built
- product
- commodity

Domain Roles

Role Types
- draft context
- execution context
- analysis context
- gateway context
- other

Inbound Communication

Collaborator

Messages

<Query> <Command> <Event>

Ubiquitous Language

Context-specific domain terminology

<Domain Term>
<definition>

Business Decisions

Key business rules, policies, and decisions

<Decision>

Outbound Communication

Messages

<Query> <Command> <Event>

Collaborator

Assumptions

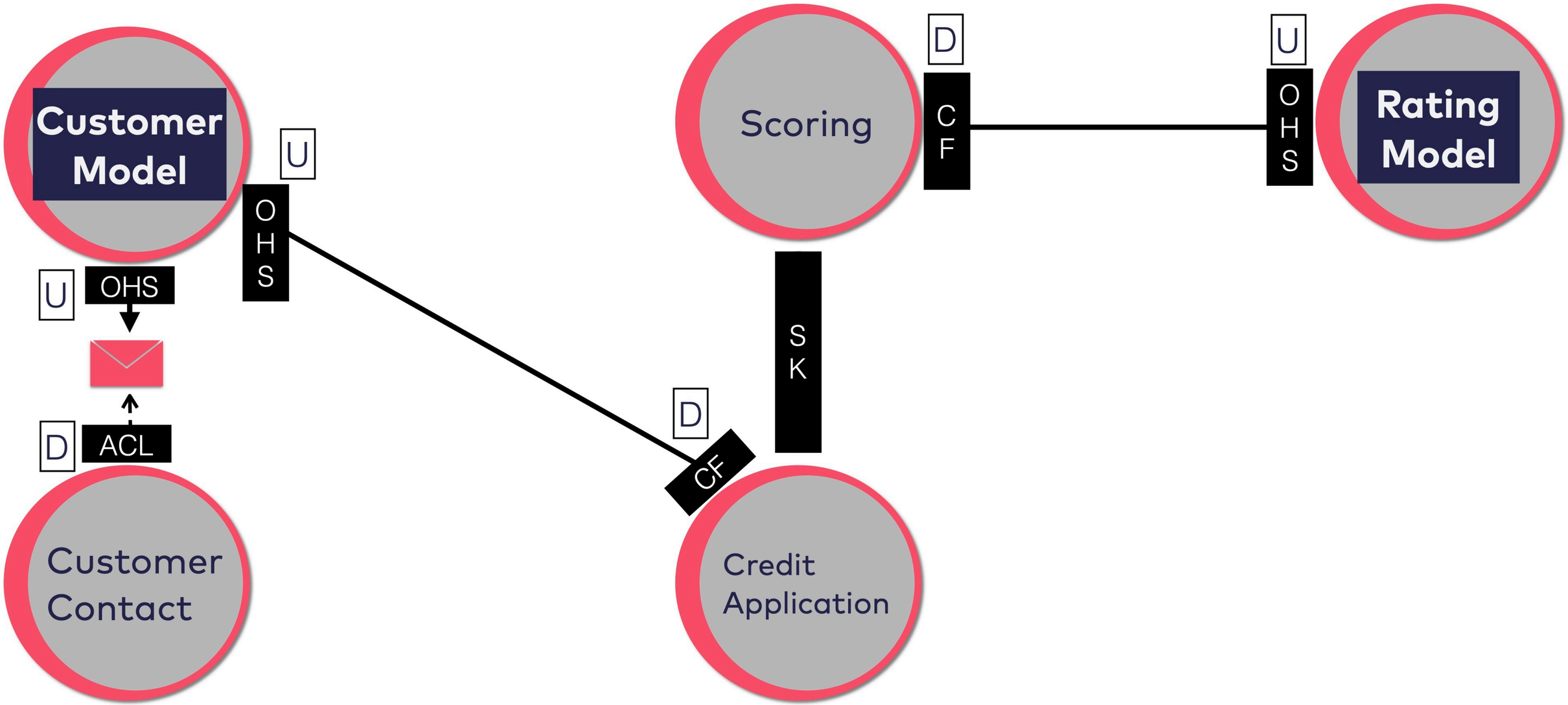
Describe which currently unverified assumptions went into this bounded context design. Make those assumptions explicit by documenting them here

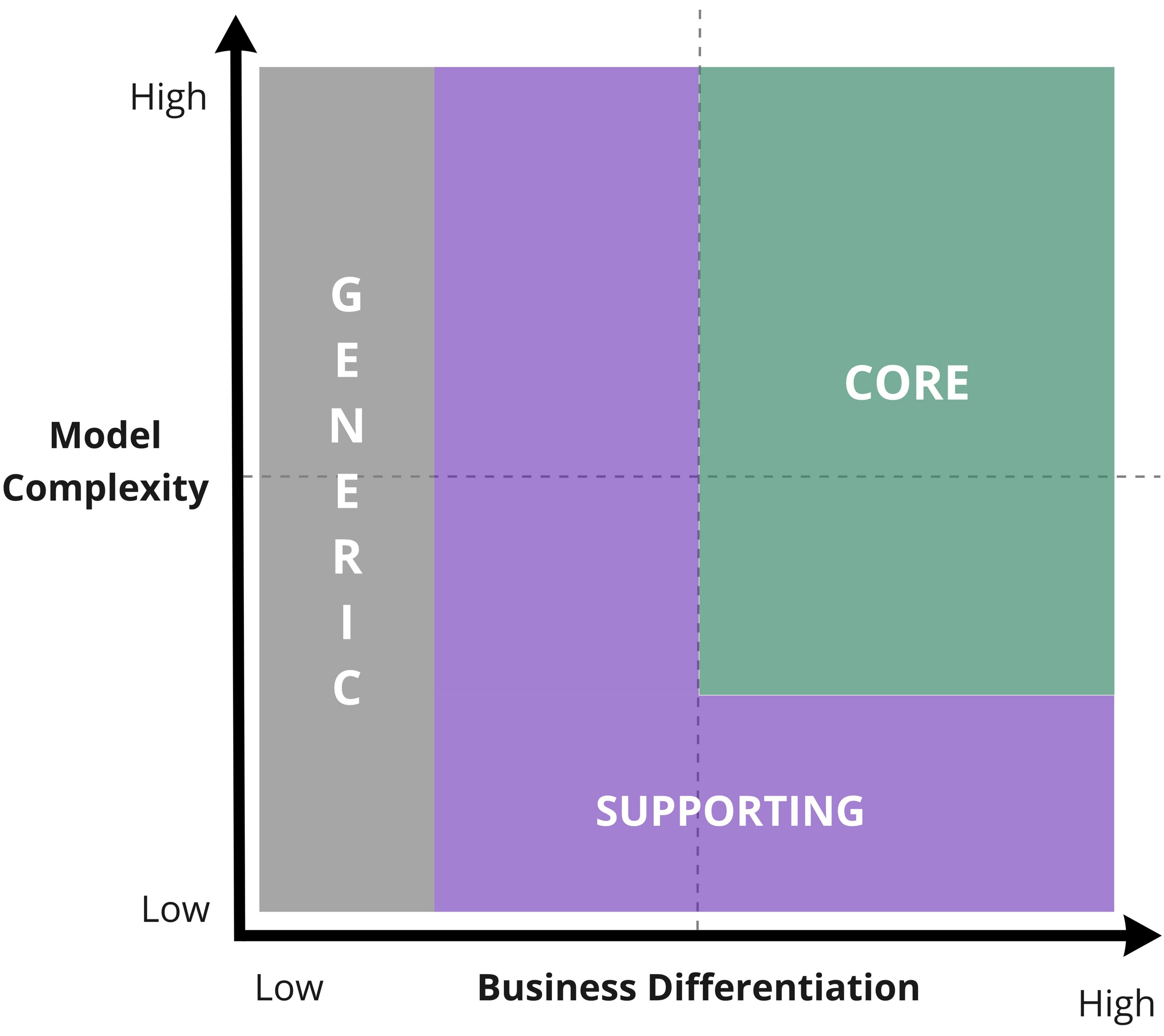
Verification Metrics

Describe metrics which can be used to (in)validate the current structure of this bounded context?

Open Questions

The model propagation from hell



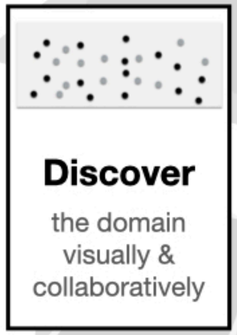

High
Model Complexity
Low
GENERIC
CORE
SUPPORTING
Low
Business Differentiation
High

Domain-Driven Design Starter Modelling Process

A starter process for beginners, not a rigid best-practice. DDD is continuous, evolutionary, and iterative design.

Understand

business model and user needs

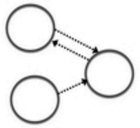


Discover

the domain visually & collaboratively

Decompose

the domain into sub-domains



Connect

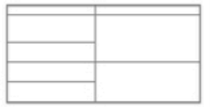
sub-domains to form a loosely-coupled architecture

Strategize

business-differentiating core-domains

Organise

teams around bounded contexts



Define

roles & responsibilities for bounded contexts

Code

your bounded contexts with tactical patterns

Notation

Actor / User / Persona

Bounded context

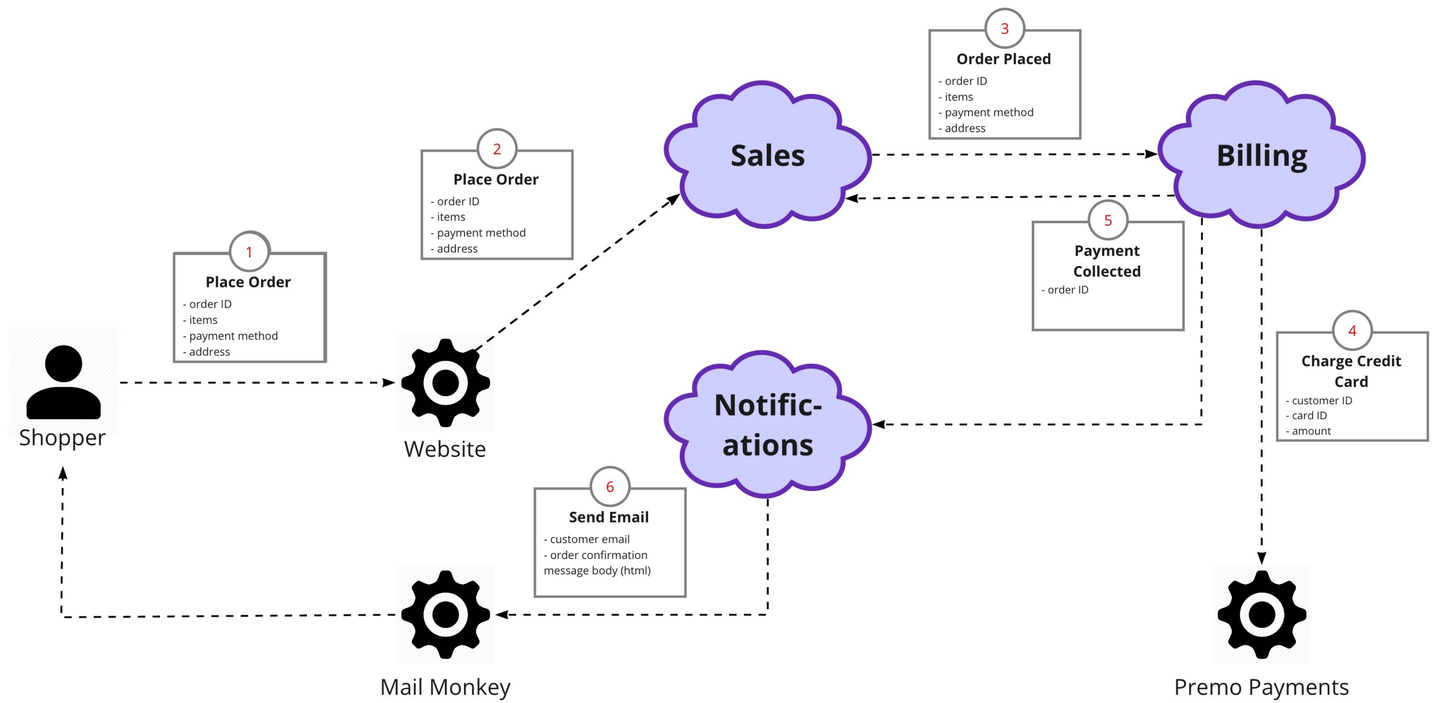
System

<n>
<name>
<data>
<data>
<data>

Message
n = order

Direction of message
From sender to receiver

Scenario: Placing an Online Order When All Items in Stock



Name

Model Traits

draft, execute, audit, enforcer, interchange, gateway, etc.

Description

Summary of purpose and responsibilities

Messages Consumed and Produced

Messages Consumed	Messages Produced
Commands Handled	Commands Sent
Events Handled	Events Published
Queries Handled	Queries Invoked

Strategic Classification

Domain	Business Model	Evolution
- Core	- Revenue	- Genesis
- Supporting	- Engagement	- Custom built
- Generic	- Compliance	- Product
- Other	- Cost reduction	- Commodity

Business Decisions

Key business rules, policies, and decisions

Dependencies and Relationships

Message Suppliers		Message Consumers	
Name	Relationship	Name	Relationship

Ubiquitous Language

Key domain terminology

Name:

V4
github.com/ddd-crew/bounded-context-canvas

Description

What benefits does this context provide,
and how does it provide them?

Strategic Classification

Domain
- core
- supporting
- generic
- other?

Business Model
- revenue
- engagement
- compliance
- cost reduction

Evolution
- genesis
- custom built
- product
- commodity

Domain Roles

Role Types
- draft context
- execution context
- analysis context
- gateway context
- other

Inbound Communication

Collaborator

Messages

<Query>

<Command>

<Event>

Ubiquitous Language

Context-specific domain terminology

<Domain Term>
<definition>

Business Decisions

Key business rules, policies, and decisions

<Decision>

Outbound Communication

Messages

Collaborator

<Query>

<Command>

<Event>


Buffer
ALLORA
TRAINER TRAVEL BOOKED
HOTEL SELECTED AND TRAINER ROOM RESERVED
WEATHER FORECAST SNAGES
HOTEL
B&B
FLIGHTS WEBSITE
TRAINS WEBSITE
CATERING COMPANY SELECTED

The Business Model Canvas

Designed for:	*Designed by:*	*Date:*	*Version:*

Key Partners

Who are our Key Partners?
Who are our key suppliers?
Which Key Resources are we acquairing from partners?
Which Key Activities do partners perform?

MOTIVATIONS FOR PARTNERSHIPS
Optimization and economy
Reduction of risk and uncertainty
Acquisition of particular resources and activities

Key Activities

What Key Activities do our Value Propositions require?
Our Distribution Channels?
Customer Relationships?
Revenue streams?

CATERGORIES
Production
Problem Solving
Platform/Network

Key Resources

What Key Resources do our Value Propositions require?
Our Distribution Channels? Customer Relationships?
Revenue Streams?

TYPES OF RESOURCES
Physical
Intellectual (brand patents, copyrights, data)
Human
Financial

Value Propositions

What value do we deliver to the customer?
Which one of our customer's problems are we helping to solve?
What bundles of products and services are we offering to each Customer Segment?
Which customer needs are we satisfying?

CHARACTERISTICS
Newness
Performance
Customization
"Getting the Job Done"
Design
Brand/Status
Price
Cost Reduction
Risk Reduction
Accessibility
Convenience/Usability

Customer Relationships

What type of relationship does each of our Customer Segments expect us to establish and maintain with them?
Which ones have we established?
How are they integrated with the rest of our business model?
How costly are they?

EXAMPLES
Personal assistance
Dedicated Personal Assistance
Self-Service
Automated Services
Communities
Co-creation

Channels

Through which Channels do our Customer Segments want to be reached?
How are we reaching them now?
How are our Channels integrated?
Which ones work best?
Which ones are most cost-efficient?
How are we integrating them with customer routines?

CHANNEL PHASES
1. *Awareness*
   How do we raise awareness about our company's products and services?
2. *Evaluation*
   How do we help customers evaluate our organization's Value Proposition?
3. *Purchase*
   How do we allow customers to purchase specific products and services?
4. *Delivery*
   How do we deliver a Value Proposition to customers?
5. *After sales*
   How do we provide post-purchase customer support?

Customer Segments

For whom are we creating value?
Who are our most important customers?

Mass Market
Niche Market
Segmented
Diversified
Multi-sided Platform

Cost Structure

What are the most important costs inherent in our business model?
Which Key Resources are most expensive?
Which Key Activities are most expensive?

IS YOUR BUSINESS MORE
Cost Driven (leanest cost structure, low price value proposition, maximum automation, extensive outsourcing)
Value Driven (focused on value creation, premium value proposition)

SAMPLE CHARACTERISTICS
Fixed Costs (salaries, rents, utilities)
Variable costs
Economies of scale
Economies of scope

Revenue Streams

For what value are our customers really willing to pay?
For what do they currently pay?
How are they currently paying?
How would they prefer to pay?
How much does each Revenue Stream contribute to overall revenues?

TYPES
Asset sale
Usage fee
Subscription Fees
Lending/Renting/Leasing
Licensing
Brokerage fees
Advertising

FIXED PRICING
List Price
Product feature dependent
Customer segment dependent
Volume dependent

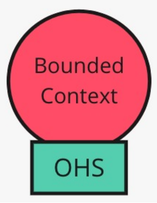
DYNAMIC PRICING
Negotiation (bargaining)
Yield Management
Real-time-Market

Context Map Cheat Sheet

Context Map Patterns

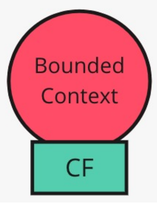
Open / Host Service

A Bounded Context offers a defined set of services that expose functionality for other systems. Any downstream system can then implement their own integration. This is especially useful for integration requirements with many other systems. Example: public APIs.



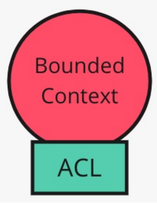
Conformist

The downstream team conforms to the model of the upstream team. There is no translation of models. Couples the Conformist's domain model to another bounded context's model.



Anticorruption Layer

The anticorruption layer is a layer that isolates a client's model from another system's model by translation. Only couples the integration layer (or adapter) to another bounded context's model but not the domain model itself.

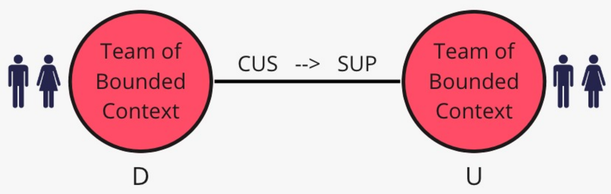


Shared Kernel

Two teams share a subset of the domain model including code and maybe the database. Typical examples: shared JARs, DLLs or a shared database schema. Teams with a Shared Kernel are often mutually dependent and should form a Partnership.

Customer / Supplier

There is a customer / supplier relationship between teams. The downstream team is considered to be the customer. Downstream requirements factor into upstream planning. Therefore, the downstream team gains some influence over the priorities and tasks of the upstream team.



Partnership

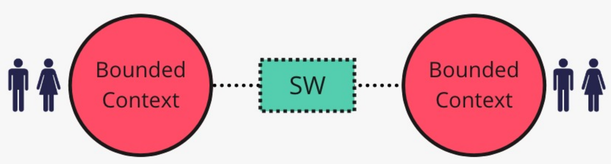
Partnership is a cooperative relationship between two teams. These teams establish a process for coordinated planning of development and joint management of integration.

Published Language

A Published Language is a well documented shared language between Bounded Contexts which can translate in and out from that language. Published Language is often combined with Open Host Service.Typical examples are iCalendar or vCard.

Separate Ways

Bounded Contexts and their corresponding teams have no connections because integration is sometimes too expensive or it takes very long to implement. The teams chose to go separate ways in order to focus on their specific solutions.



Big Ball Of Mud

A (part of a) system which is a mess by having mixed models and inconsistent boundaries. Don't let this lousy model propagate into the other Bounded Contexts. Big Ball Of Mud is a demarcation of a bad model or system quality.

Team Relationships

Mutually Dependent

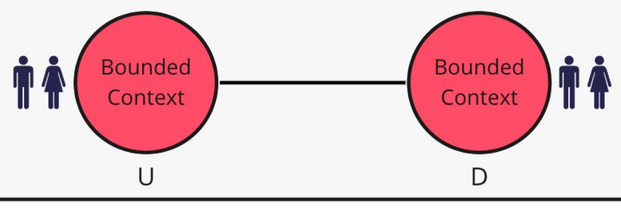
Two software artifacts or systems in two bounded contexts need to be delivered together to be successful and work. There is often a close, reciprocal link between data and functions between the two systems.

Free

Changes in one bounded context do not influence success or failure in other bounded contexts. There is, therefore, no organizational or technical link of any kind between the teams.

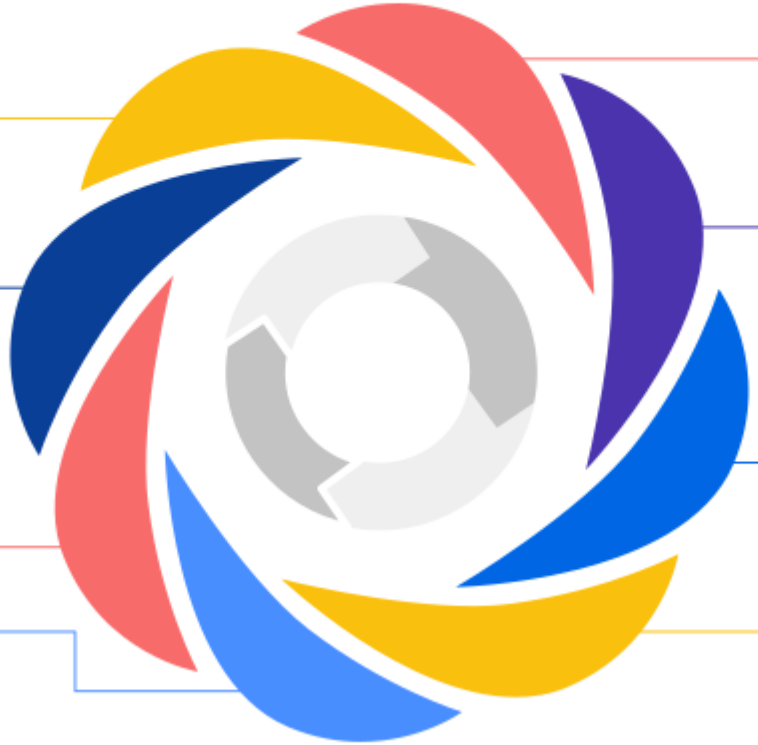
Upstream / Downstream

Actions of an upstream team will influence the downstream counterpart while the opposite might not be true. This influence can apply to code but also on less technical factors such as schedule or responsiveness to external requests.



Domain-Driven Design starter modeling process

A starter process for beginners, not a rigid best-practice. **DDD is continuous, evolutionary and iterative design.**



Align
Business model and user needs

Discover
The domain visually & collaboratively

Decompose
The domain into sub-domains

Connect
Sub-domains to form a loosely coupled architecture

Strategize
Business differentiating core-domains

Categorize
Your teams around Bounded contexts

Define
Roles & responsibilities for bounded contexts

Code
Your bounded context with tactical patterns

Source: https://github.com/ddd-crew/ddd-starter-modelling-process

Domain-Driven Design starter modeling process

A starter process for beginners, not a rigid best-practice. **DDD is continuous, evolutionary and iterative design.**

Align
Business model and user needs

Discover
The domain visually & collaboratively

Decompose
The domain into sub-domains

Connect
Sub-domains to form a loosely coupled architecture

Strategize
Business differentiating core-domains

Categorize
Your teams around bounded contexts

Define
Roles & responsibilities for bounded contexts

Code
Your bounded context with tactical patterns

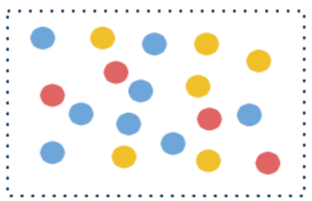
Source: https://github.com/ddd-crew/ddd-starter-modelling-process

Domain-Driven Design starter modeling process

A starter process for beginners, not a rigid best-practice.
DDD is continuous, evolutionary and iterative design.

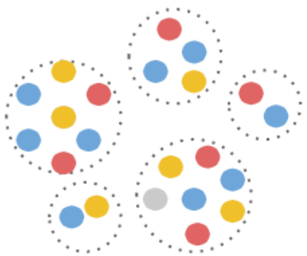
Align

Business model and user needs



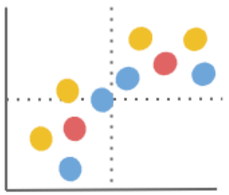
Discover

The domain visually & collaboratively



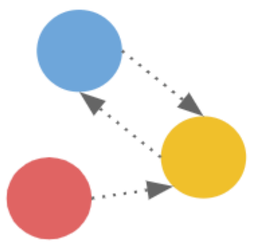
Decompose

The domain into sub-domains



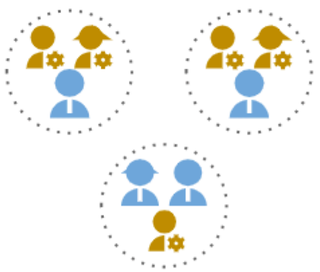
Strategize

Business differentiating core-domains



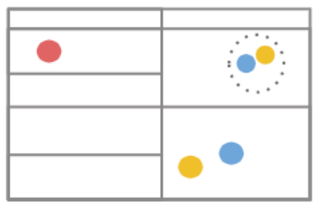
Connect

Sub-domains to form a loosely coupled architecture



Organize

Teams around bounded contexts

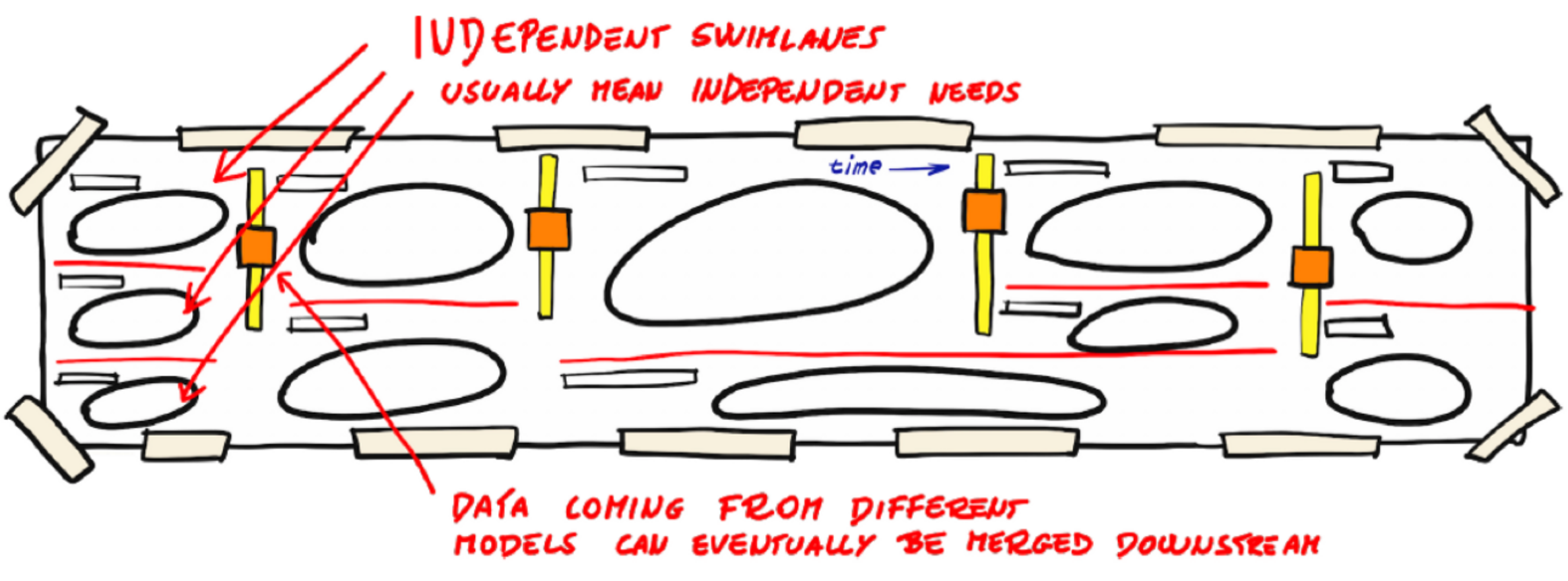


Define

Roles & responsibilities for bounded contexts

Code

Your bounded context with tactical patterns


INDEPENDENT SWIMLANES
USUALLY MEAN INDEPENDENT NEEDS
time
DATA COMING FROM DIFFERENT
MODELS CAN EVENTUALLY BE MERGED DOWNSTREAM