

Discovering the Domain Architecture through DDD



Dino Esposito

AUTHOR

@despos

www.software2cents.wordpress.com

Key Points

Ubiquitous Language

Definition and
Discovery

Bounded Context

Definition and
Discovery

Context Map

Design of Top-level
Architecture



Ubiquitous Language

Ubiquitous Language—**What's That?**

- **Vocabulary of domain-specific terms**
 - Nouns, verbs, adjectives, idiomatic expressions and even adverbs
- **Shared by all parties involved in the project**
 - Primary goal of avoiding misunderstandings
- **Used in all forms of spoken and written communication**
 - Universal language of the business as done in the organization

Motivation



Definition

Natural language,
not artificial

Comes out of
interviews and
brainstorming

Iteratively composed
and refined along the
way

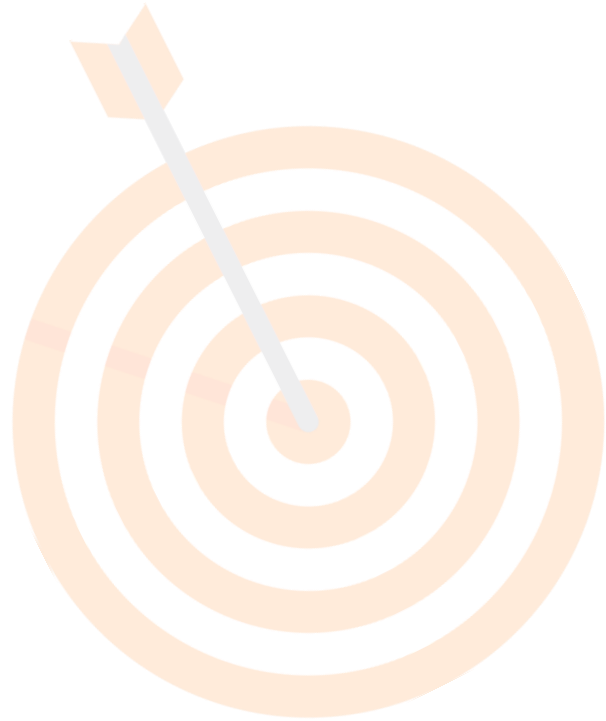
Unambiguous and
fluent

Meets
expectations of
domain experts

Meets
expectations of
technical people



Ubiquitous \approx Used Everywhere



- User Stories & RFC
- Meetings
- Emails
- Technical Documentation
- Schedule
- **Source code** ←

Structure



List of terms saved to Office documents

- Glossary of terms fully explained
- Made available to everyone
- Part of the project documentation

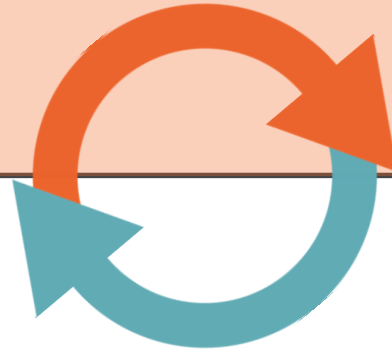


Continuously updated

- Responsibility of the development team

“Use the model as the backbone of a language.”

Eric Evans



- **Discovering** the ubiquitous language
- leads you to **understand** the business domain
- **PS:** Any model that works.
Not necessarily an object-oriented model.



Start from User Requirements

Noun

Verb

As a registered customer of the I-Buy-Stuff online store, I can redeem a voucher for an order I place so that I don't actually pay for the ordered items myself.

Registered Customer

Redeem

Voucher

Order

Place

Pay

Ordered Items

- **Voucher** is the domain name.
- Synonyms like coupon or gift card are **not** allowed.

UBIQUITOUS LANGUAGE

Words and verbs that truly reflect the semantics of the business domain.



At Work Defining the Ubiquitous Language

Delete the booking



Cancel the booking

Submit the order



Checkout

Update the job order



Extend the job order

Create the invoice



Register/Accept the invoice

Set state of the game



Start/Pause the game



No Ambiguity
No Synonyms

*Extra threshold costs
should be
emphasized in the
user account*

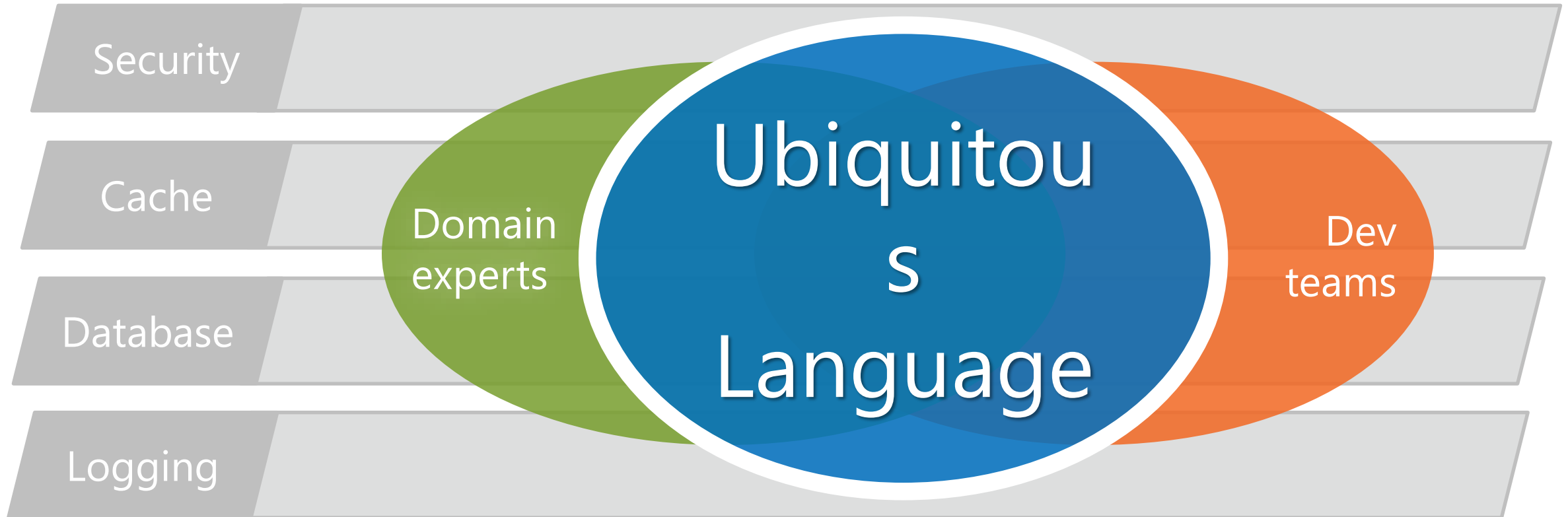


What's really meant here?

- Show costs when the user logs in?
- List costs in a detail page?
- Mark extra costs in the bill?
- Other? Specify ...

Different concepts named differently.
Matching concepts named equally.

How Much Is Technical?



Tips from the Trenches



Scenarios Where UL Is Key to Have

Really

Really a lot of domain logic tricky to digest

- Ensures all relevant terms are understood
- No other term is used to indicate same/similar concepts



Business logic not completely defined

- Business is young and growing with the system (i.e., startup)
- Domain logic discovered along the way

Ubiquitous Language in Code

Missing a point is creating a bug.

- Classes
- Members
- Namespaces

Naming
convention

Extension
Methods

- Domain-specific
language
- C#, Kotlin

- Code assistants
- Refactoring tools
- Static checkers

Tools

Agnostic

- No mandated
technology
- No mandated
paradigm




The ubiquitous language
changes indefinitely.

Bounded Context—**What's That?**

- **Delimited space where an element has a well-defined meaning**
 - Any elements of the ubiquitous language
- **Beyond the boundaries of the context, the language changes**
 - Each bounded context has its own ubiquitous language
- **Business domain split in a web of interconnected contexts**
 - Each context has its own architecture and implementation

Motivation

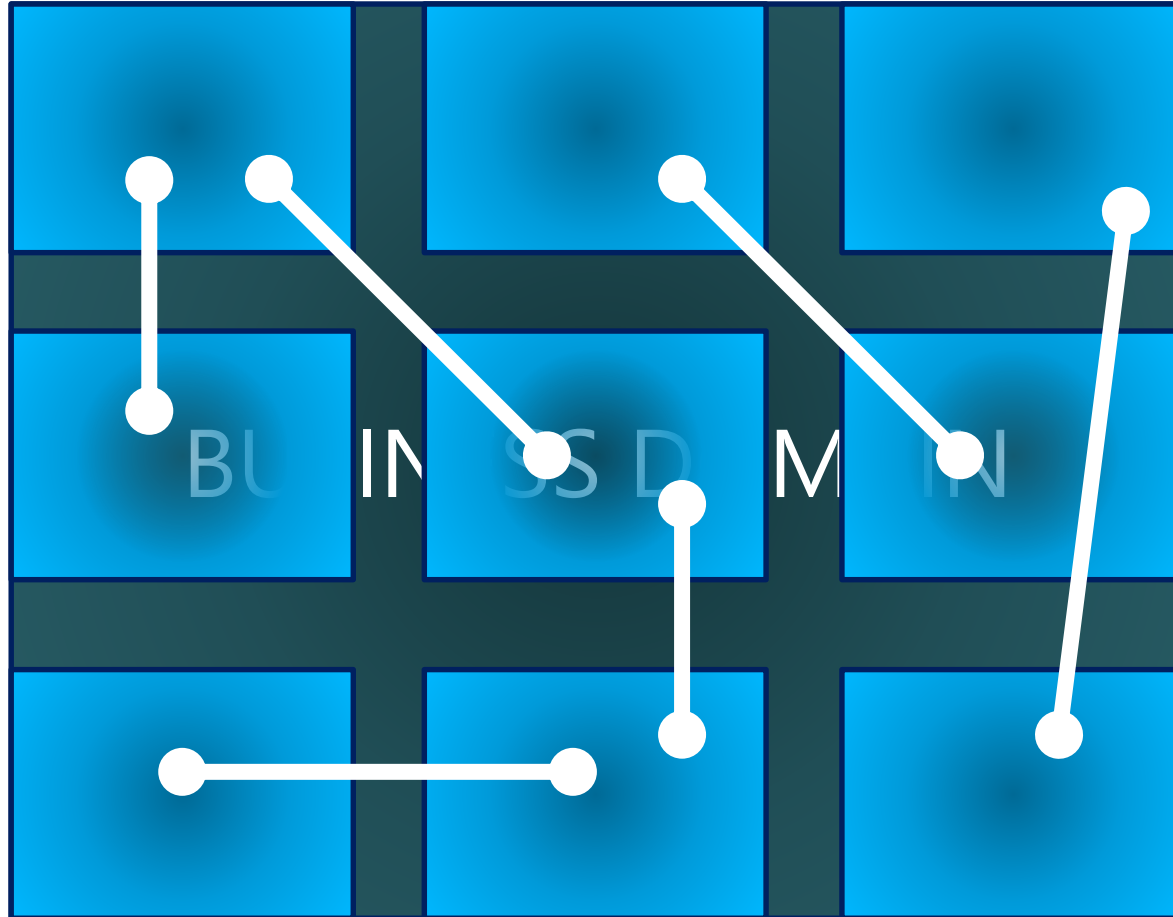


Remove
ambiguity and
duplication

Simplify design
of software
modules

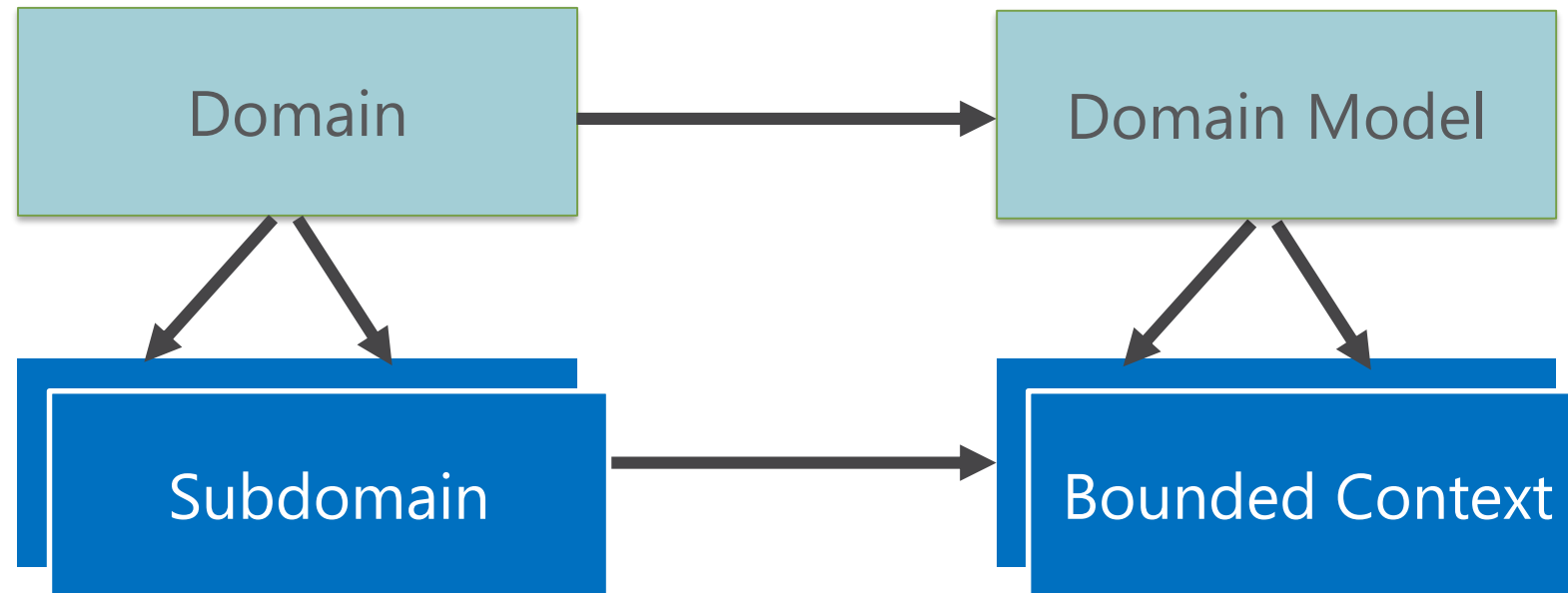
Integration of
external
components

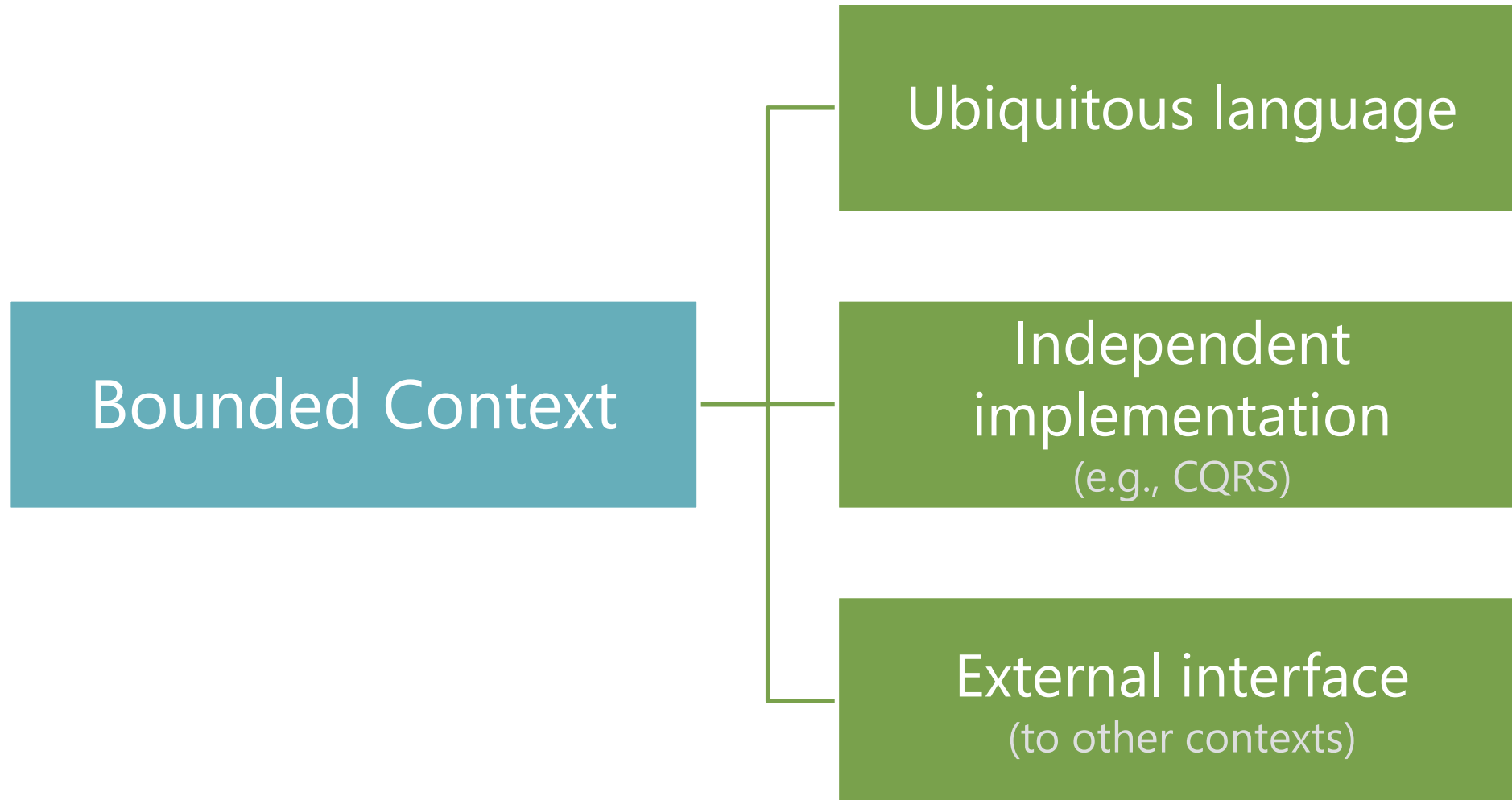
Domain and Contexts



Problem Space

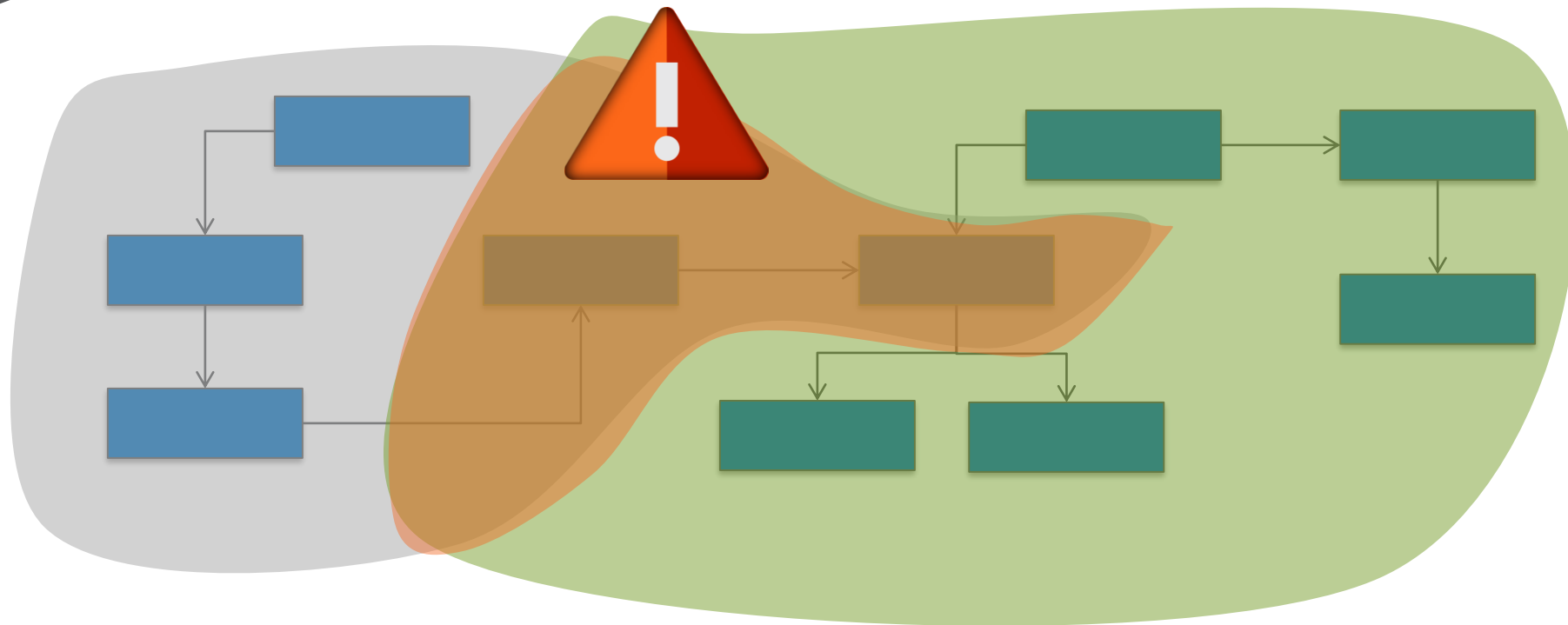
Solution Space







Integrity of the model at risk

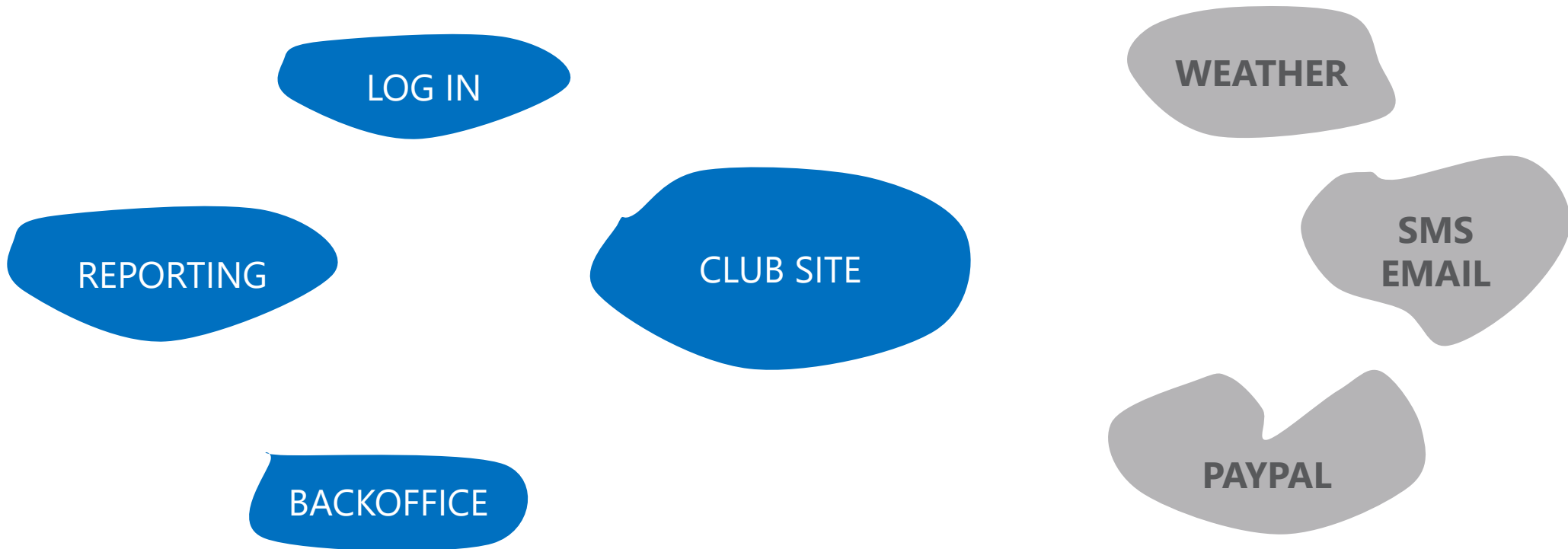


Integrity at Risk

- Same term that means differently to different people
- Same term to indicate different elements
- Dependency on external subsystems
- Dependency on legacy code
- Functional areas of the application that are better treated separately

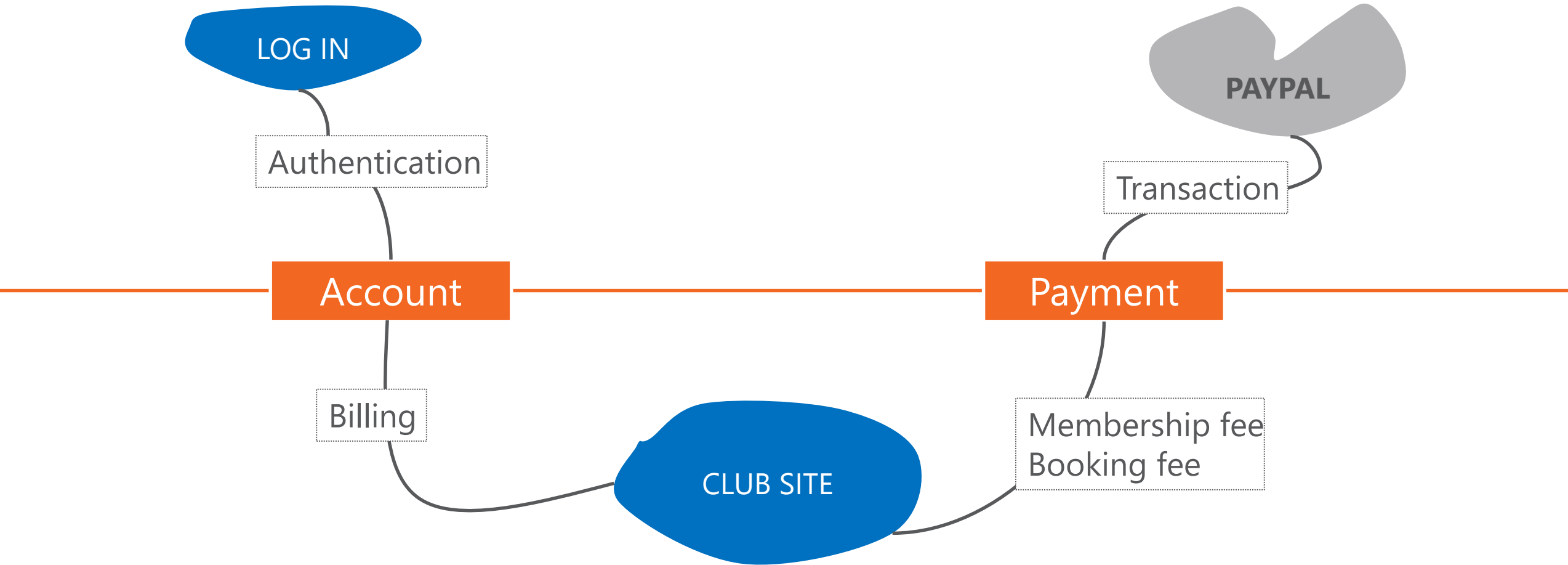


Booking application for a sporting club

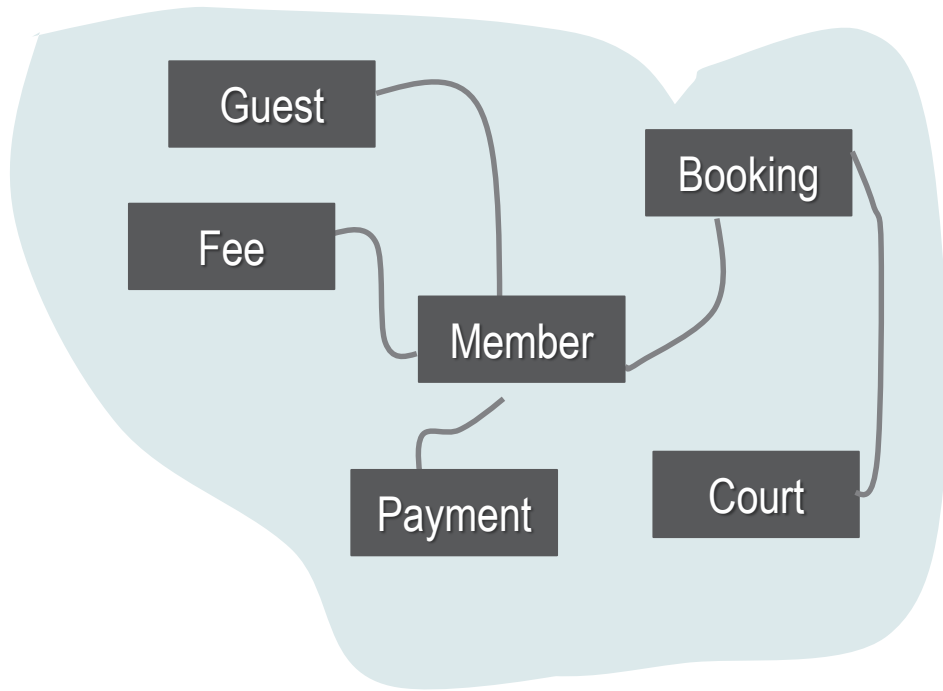


Same term that is used to indicate different things

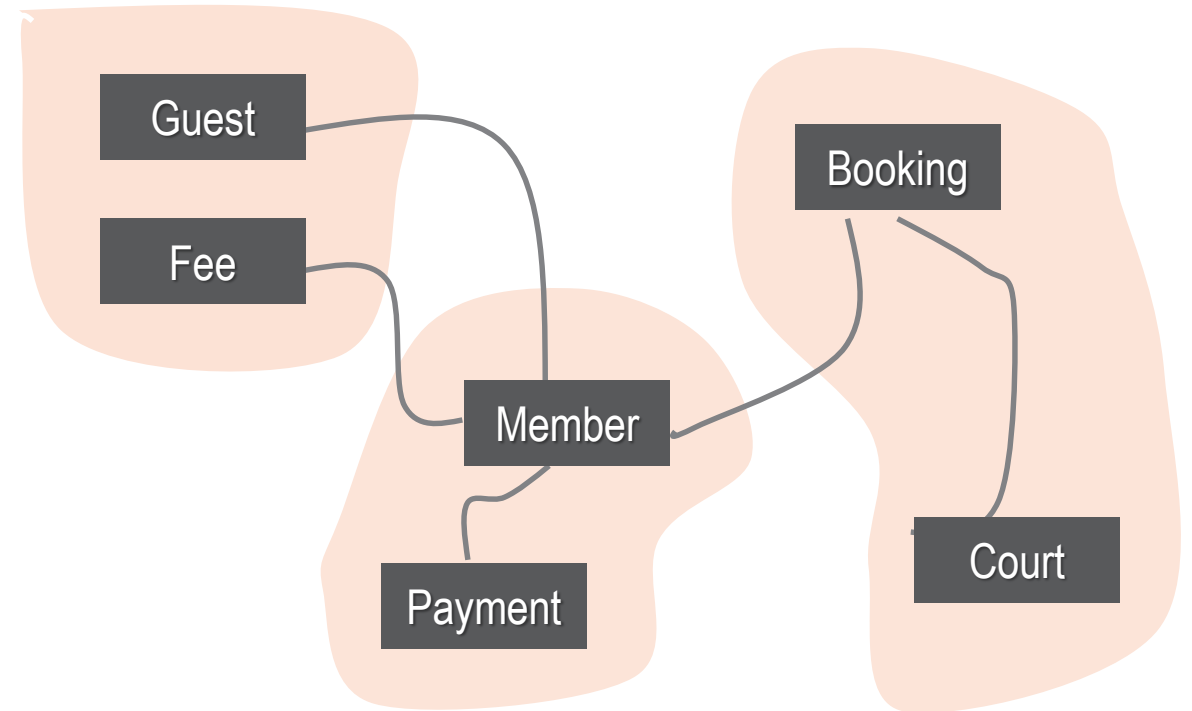
Same term that indicates different things to different people



Contours of the Bounded Context

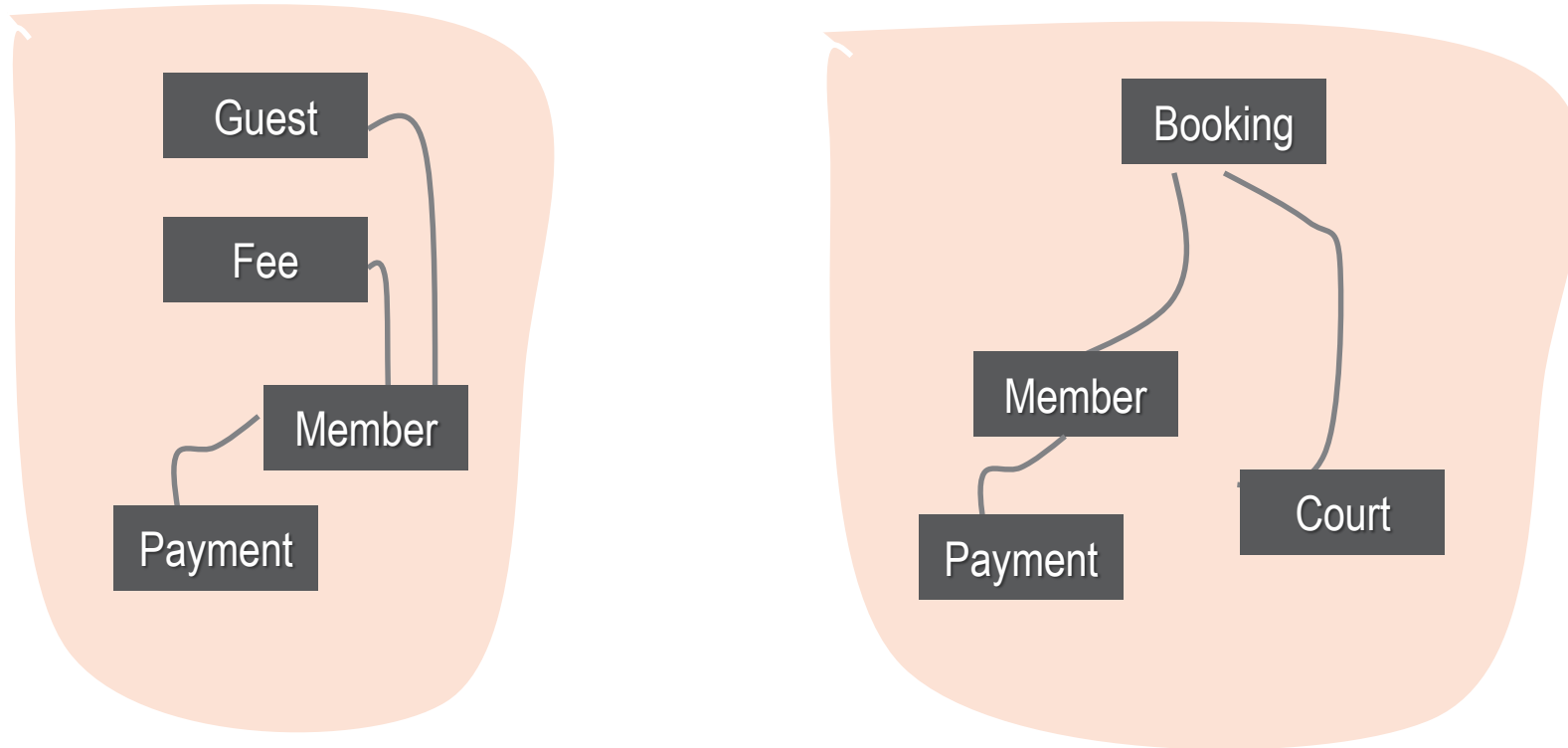


Single Model



**Bounded Context
with Shared Kernel**

Contours of the Bounded Context



**Self-contained
bounded contexts**

Web of Bounded Contexts

**Systems may
result from the
composition of
multiple
contexts**

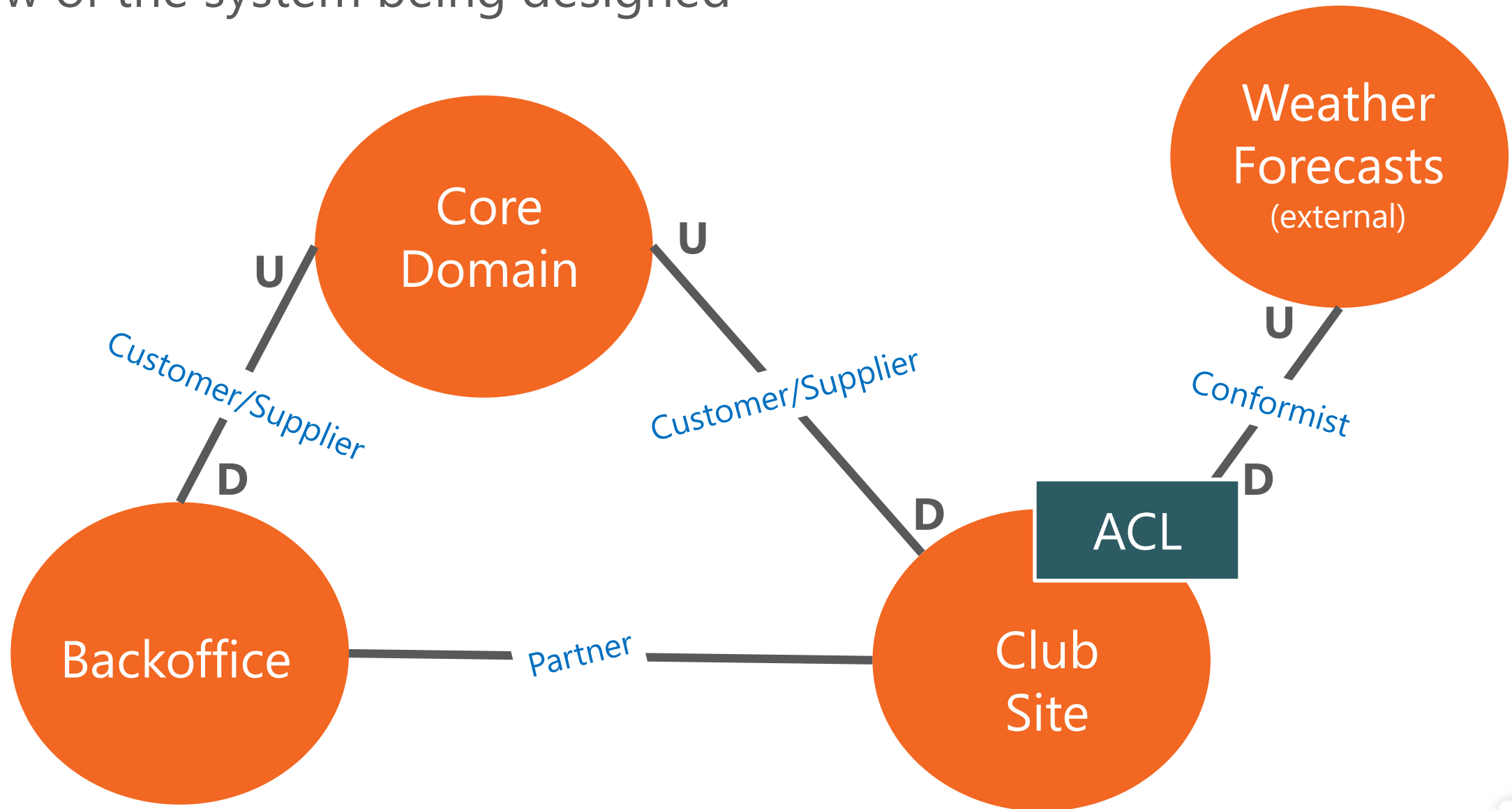
Example:

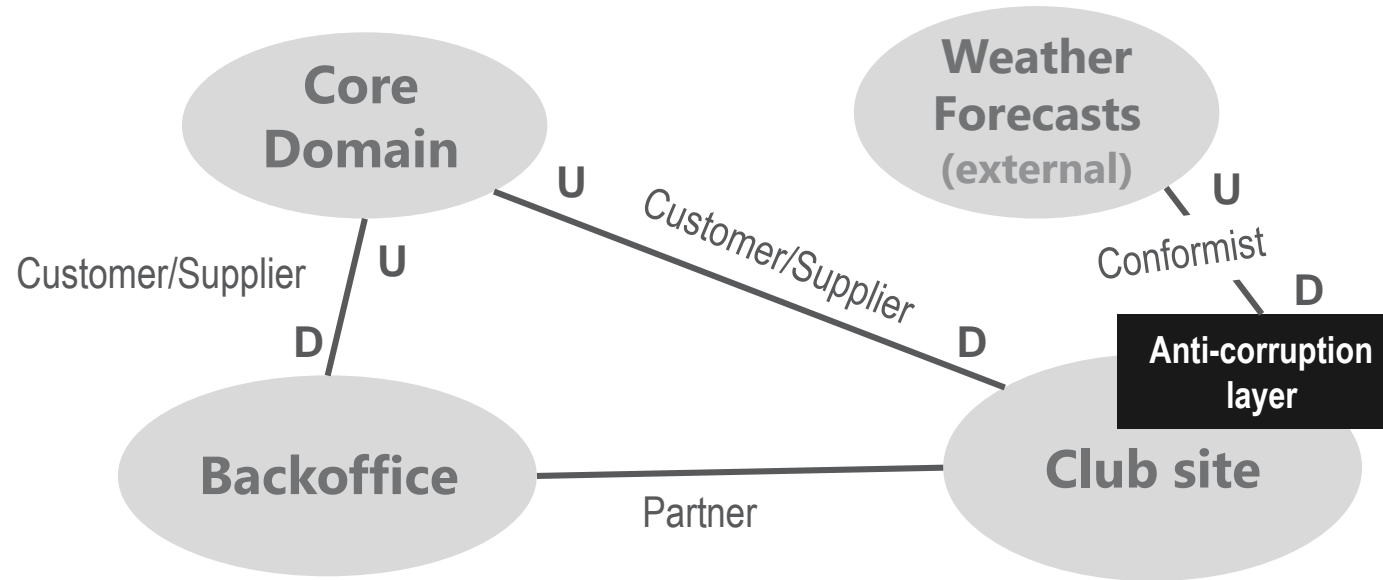
Web store
Accountability
Delivery &
shipment

**Number of
bounded
contexts often
reflects
physical
organization**

One bounded
context for each
business
department

Context map is the diagram that provides a comprehensive view of the system being designed





Direction of relationship

Upstream context influences **downstream** context

Aspects being influenced: binaries, schedule, request-for-changes

Relationships

Conformist

- Downstream context depends on upstream context
- No negotiation possible

Customer/Supplier

- Customer context depends on supplier context
- Chance to raise concerns and have them addressed in some way

Partner

- Mutual dependency between the two contexts

Shared Kernel

- Shared model that can't be changed without consulting teams in charge of contexts that depend on it.

Anti-corruption Layer

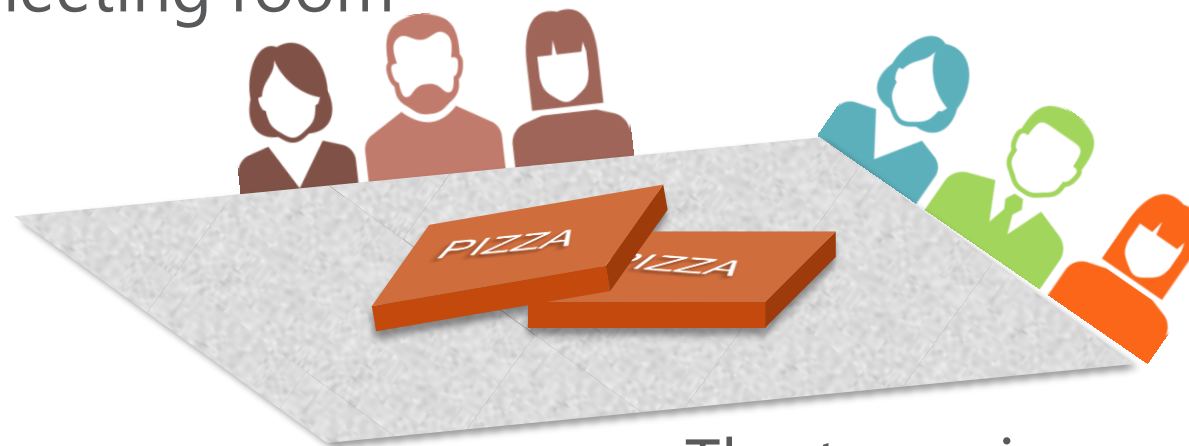
- Additional layer giving the downstream context a fixed interface no matter what happens in the upstream context



Event Storming

Exploring a business domain starting from observable domain events

Developers and domain experts
together in a meeting room



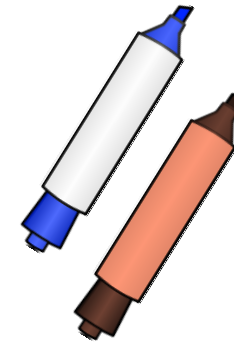
The two-pizza rule sets the right
number of invited people



Event Storming

Exploring a business domain starting from observable domain events

Necessary equipment



How It Works

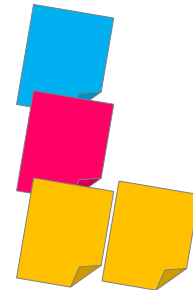
Identify relevant domain events

- Use a sticky note of a given color to put events on the wall



Find what causes the event

- **User action?** Add a sticky note of a different color
- **Asynchronous event?** Add a sticky note of a different color
- **Another event?** Add another sticky of same color on top



Look at the modeling surface as a timeline

- Add notes with markers



Facilitator



Leads the
meeting

Starts the meeting
asking questions

Sticks first notes on
the wall to show the
way

Guides the
modeling
effort

Asks question to
better understand the
emerging model

Ensures ideas are
represented
accurately

Keep focused and
moves ahead

Benefits

Comprehensive
vision of the
business domain

Bounded contexts
and aggregates in
each context

Types of users in
the system

Where UX is critical

Aggregate
handles commands
and controls
persistence

Personas who runs
commands and
why

Sketches of
relevant screens

More Information

Just search for **Event Storming**