Domain Driven Design

Un viaggio inaspettato

Facciamo un gioco?

"Ho l'idea del secolo" (cit.)

Sono il responsabile di una multinazionale leader nel settore del commercio all'ingrosso. Voglio creare un gestionale che mi permetta di:

Gestire il mio magazzino

Gestire le promozioni che verranno fatte all'interno

Interfacciarsi con la cassa per gestire la vendita dei miei prodotti

Gestire il supporto dei clienti in caso di richieste di informazioni o di resi

Pronti? Via!

Cosa ho capito di DDD…

Cosa è DDD?

Domain-driven design (DDD) is an approach to developing software for complex needs by deeply connecting the implementation to an evolving model of the core business concepts.

https://dddcommunity.org/learning-ddd/what_is_ddd/

Cosa NON è DDD?

NON è un Silver Bullet!

NON è una tecnologia

NON è (solo) Repository e Aggregate

DDD in 3 parole (o poco più)

Ubiquitous Language

Bounded Contexts

Entità / Value Objects

Aggregate

Domain Event

Repository

Context Mapping

Ubiquitous Language

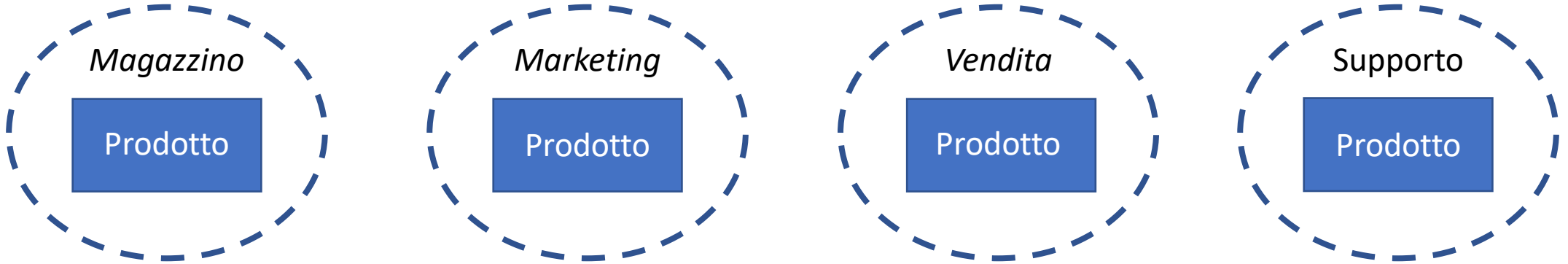
common, rigorous language between developers and users. This language should be based on the Domain Model used in the software - hence the need for it to be rigorous, since software doesn't cope well with ambiguity.

https://martinfowler.com/bliki/UbiquitousLanguage.html

Bounded Contexts

In caso di ambiguità nel nostro Ubiquitous Language possiamo provare a dividere il nostro dominio in contesti diversi. **Ognuno con il proprio linguaggio!**

Sono Bounded in quanto **isolati e non dipendenti tra di loro**.



Entità e Value Objects

Sono Entità tutti quegli attori del mio sistema che hanno una identità definita, espongono uno stato e possono avere dei comportamenti. Lo stato di una entità è modificabile a seguito di un'azione che avviene nel mio sistema.

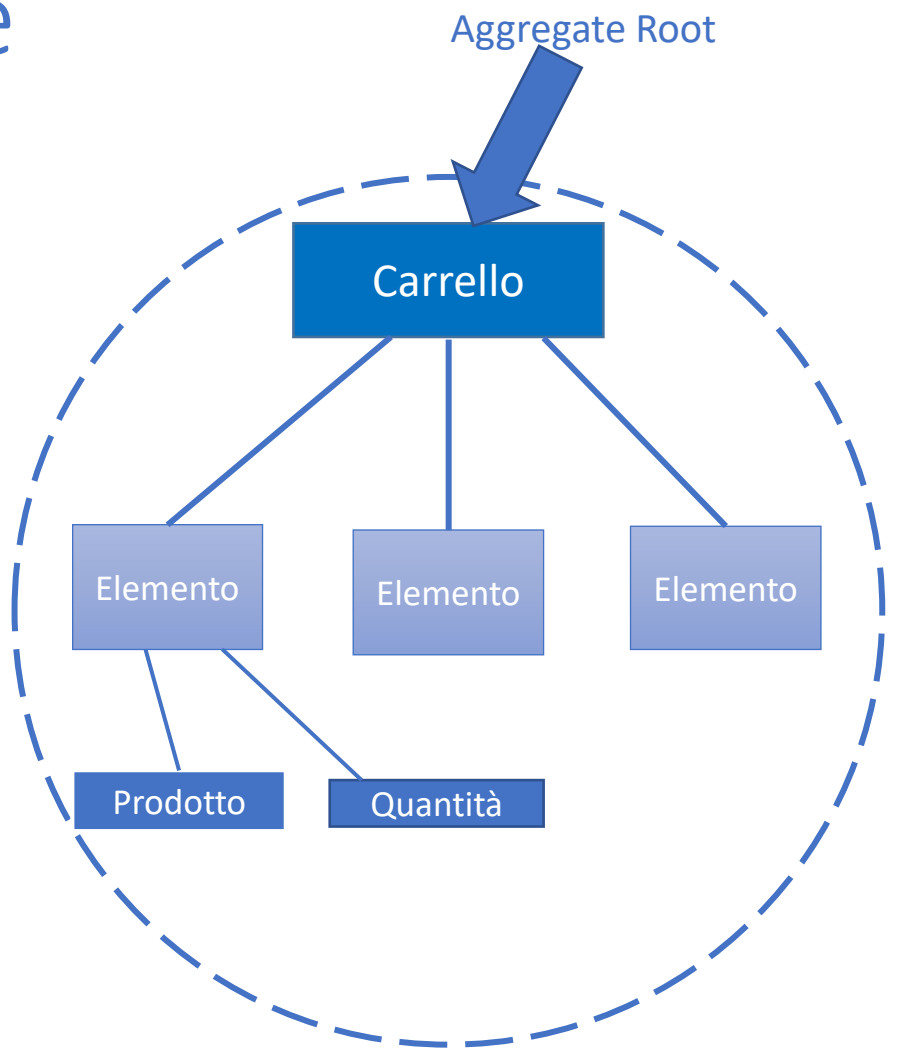
Sono Value objects tutte quei componenti che sono **immutabili**. Ogni modifica che avviene su un value object genera sempre una nuova istanza di esso.

Aggregate

Un **Aggregate** rappresenta un grafo di oggetti collegati tra di loro secondo la logica di dominio.

Espone *"both data and behavior"* e viene creato utilizzando delle **factory**. I comportamenti rappresentano i cambi di stato dell'Aggregate.

Tutta la gestione dell'Aggregate passa attraverso un singolo attore definito **Aggregate Root**



Domain Event

Un Domain Event rappresenta un'azione accaduta e che è **significativa per il mio dominio**.

NON TUTTI GLI EVENTI SONO DOMAIN EVENT!

Vengono generati dai comportamenti dell’Aggregate.

Repository

Ha il compito di persistere gli aggregate del dominio

Identifica una singola transaction sulle operazioni eseguite

Lavora direttamente SOLO sull'Aggregate root

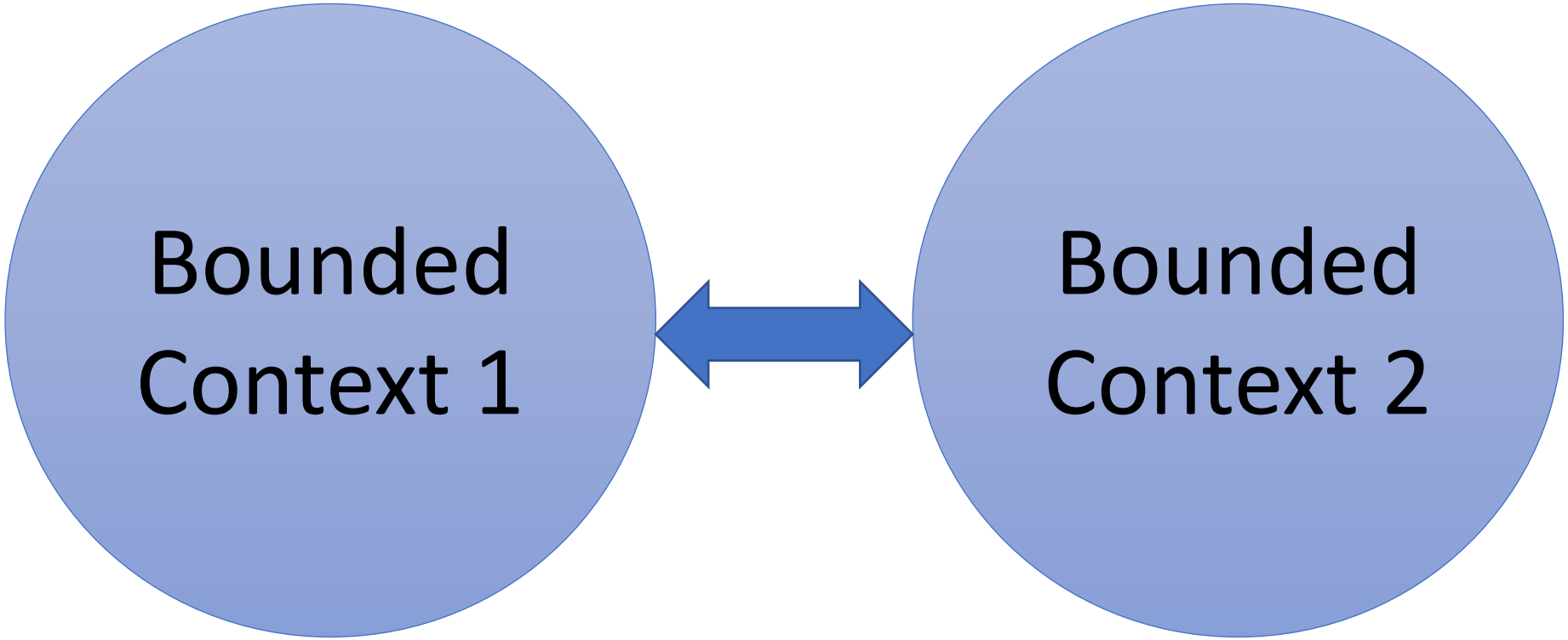
Context Mapping

Essendo isolati i Bounded Context non contengono riferimenti diretti tra di loro.

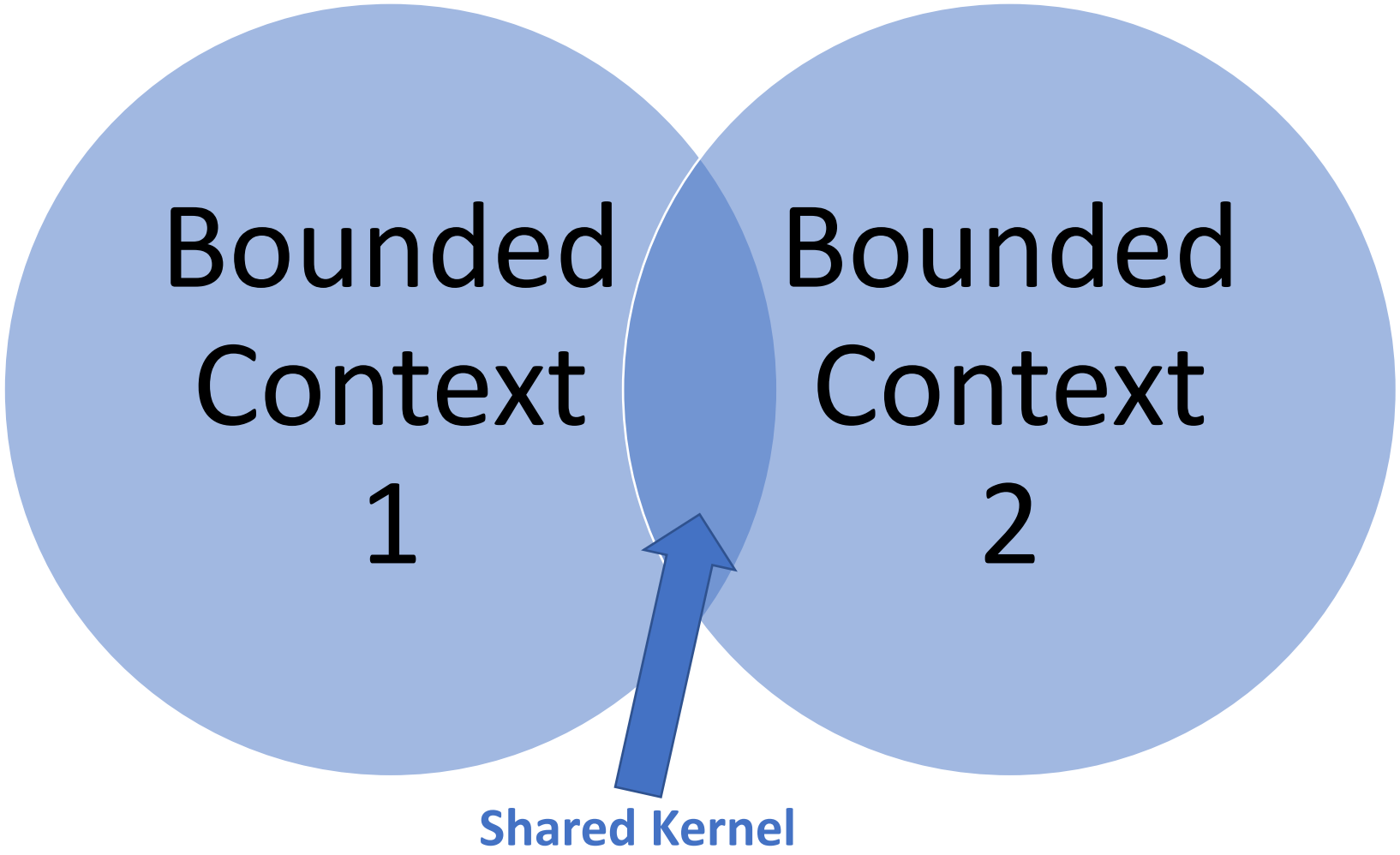
Esistono strategie di comunicazione, definite **Context Mapping**

Downstream - Upstream	Mutually dependent	Free

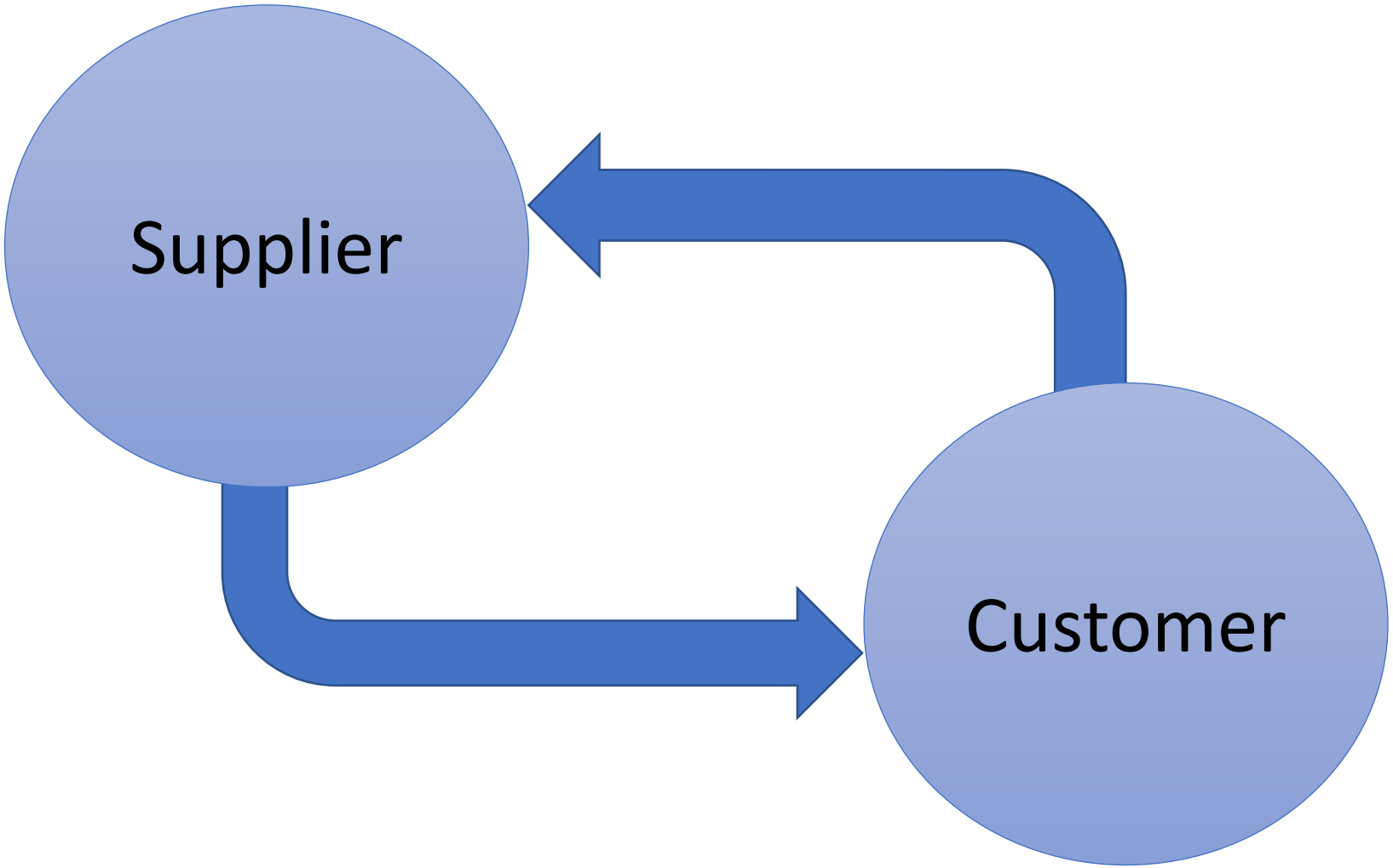
Partnership



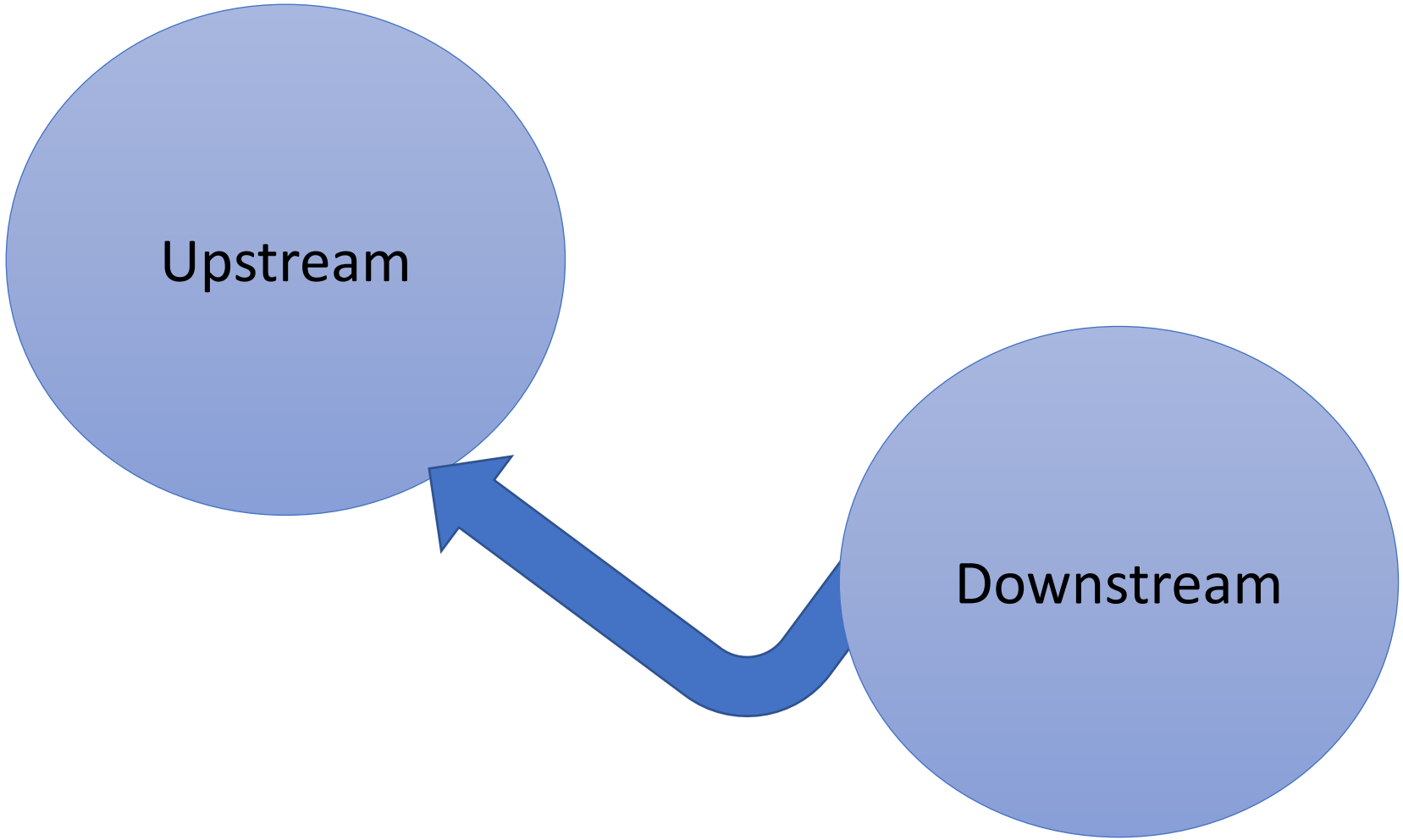
Shared Kernel



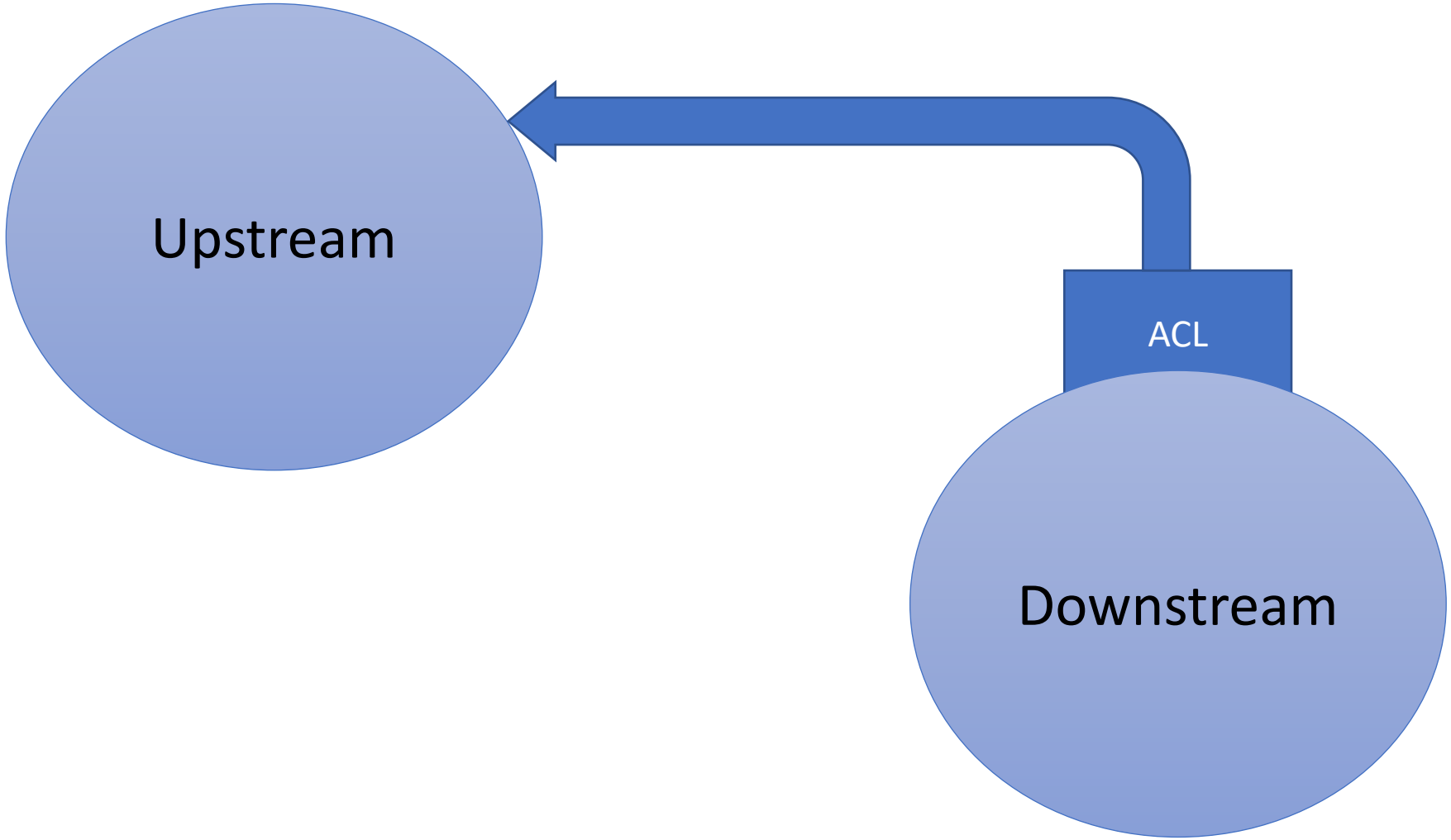
Customer - Supplier



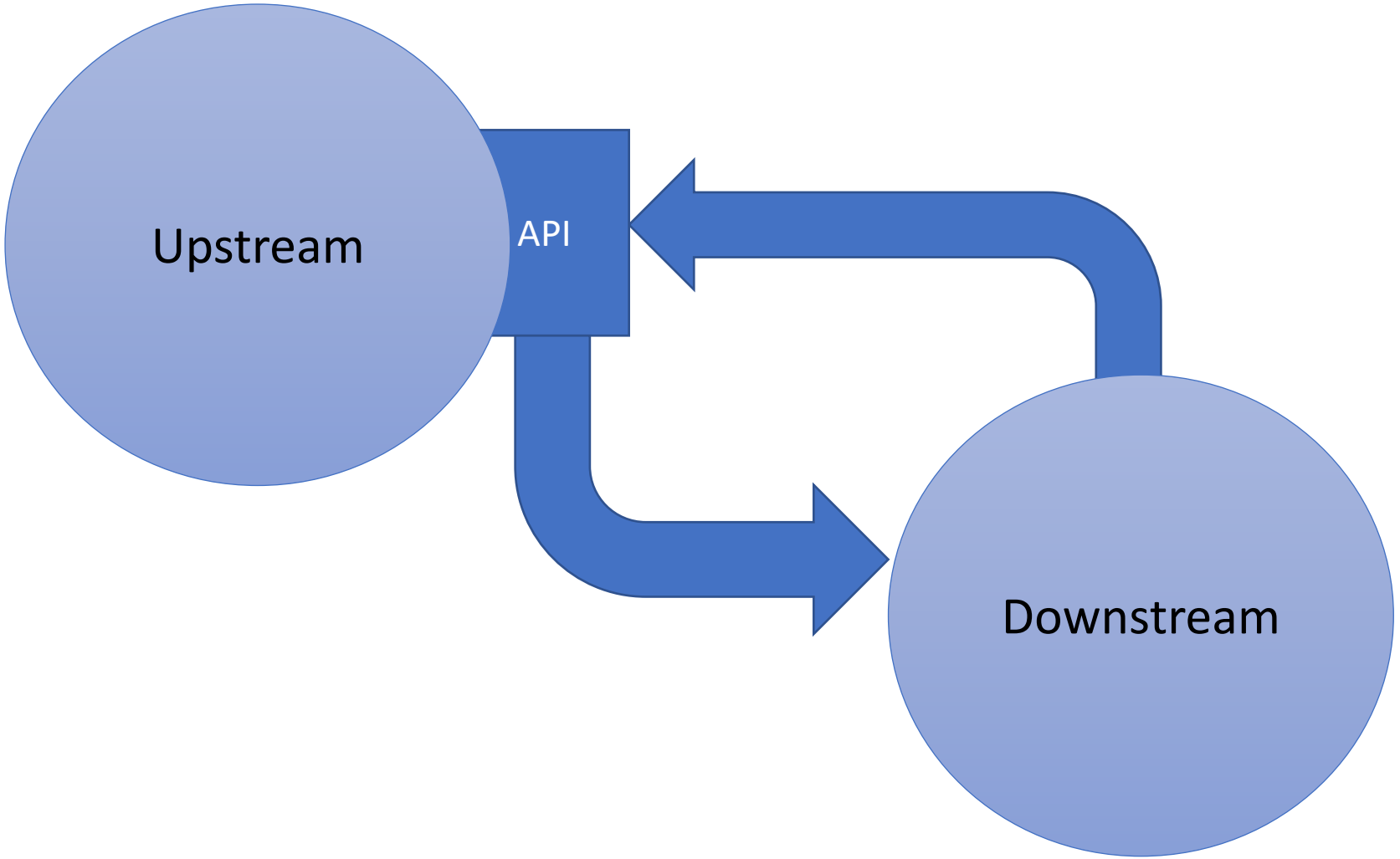
Conformist



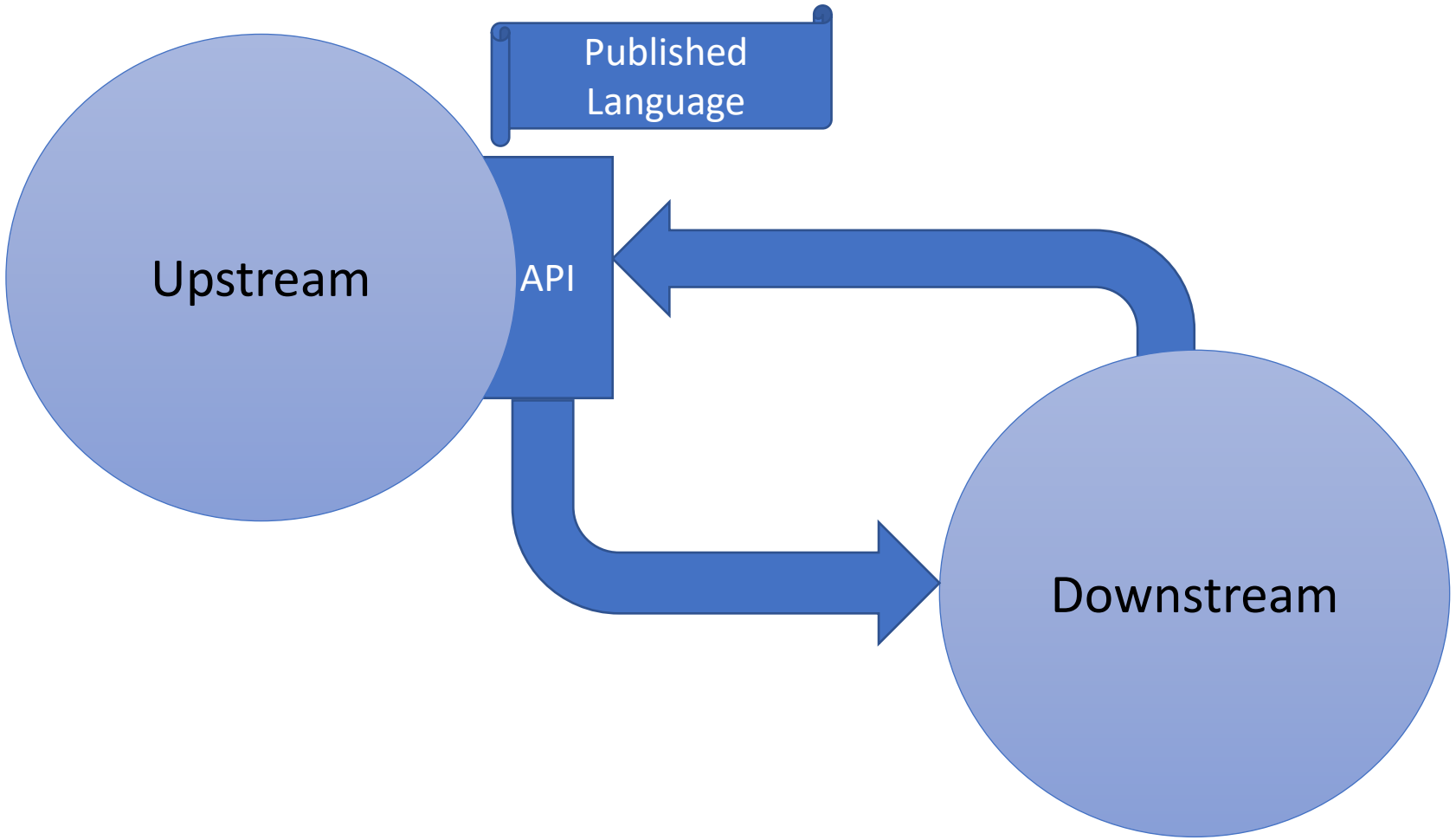
Anti-Corruption Layer



Open Host



Published Language



Quindi adesso possiamo rivedere il nostro gestionale?


Not. Yet.

Ci siamo fatti tutte le domande?

Qual è la giacenza minima sotto cui deve partire una richiesta di ricarica del prodotto?

Quali sono le informazioni necessarie per aprire una richiesta di assistenza?

Quali sono le regole in base a cui viene applicato uno sconto?

Cosa succede quando viene accettata una richiesta di reso?

DDD & Friends: CQRS

Basato su CQS (Command Query Separation) di Bertrand Mayer

Un componente o modifica uno stato oppure ritorna dei dati

Separare scrittura da lettura ci permette di modellare i singoli componenti ottimizzandoli in base alla necessità

DDD & Friends: Event Sourcing

Invece di salvare l'ultimo stato noto di un elemento, salvo tutti gli eventi che hanno portato a questo stato

Quando recupero l'oggetto replico tutti gli eventi che sono avvenuti per ricostruire l'ultimo stato noto

Spesso viaggia in coppia con CQRS

Grazie!

Alberto Mori
Software Developer @ Able Tech s.r.l.

https://twitter.com/albx87
https://github.com/albx
albi.mori@gmail.com
http://www.morialberto.it/