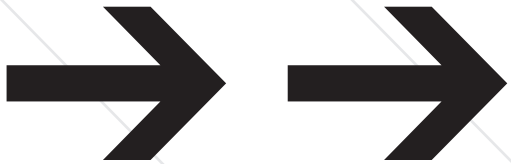
DDD

Domain Driven Design - The way back to OO!

Felipe Rodrigues de Almeida

TOPICS



THE PROBLEM

Where are we missing the way?

- What is important? What is urgent?
- Domain Models vs Manager Models (MOP)
- Bad Communication
- Architects thinking only about infra-structure
- We have lost real Object Orientation

THE CONCEPT

What is DDD about?

Domain means “a sphere of knowledge, influency and activity “ of a situation.

So, Domain Driven Design means, design (a software) driven (oriented) by it knowledge, influency and activity.

THE CONCEPT

What is DDD about?

Above all, DDD is about to take the domain (a sphere of knowledge, influency and activity) and represent it in objects.

DDD is the way back to OO!!!

THE CONCEPT

Why to use DDD?

More than just a concept, DDD is a process to find out how to build a system and also to define how to treat specific business issues.

THE CONCEPT

Why to use DDD?

Using DDD as a process you will be able to:

- really use all those things your teacher said you could when you were learning OO.
- avoid the Manager(Service) Model (MOP), which requires knowledge about a specific implementation. This kind o design is composed only by services.
- improve the comunication between your team and the customer.

THE CONCEPT

Where are we missing the way? (Fixes)

- What is important? What is urgent?

The business is important and the customer too. Let`s give some attention to business and customer comunication!

Important aspects cannot be left behind.

Where are we missing the way? (Fixes)

- Domain Models vs Manager Models(services) MOP

Domain Models treat the business like objects, services and compositions, concerned about the way things are connected in the real world.

Developing software is not about creating makers.

Where are we missing the way? (Fixes)

- Bad Comunication

The words you use when you are trying to explain a situation to your customer, who doesn't know the system, are responsible for misunderstanding. Talking with him using his business terms, will make for better communication.

Use the customers words to communicate!

Where are we missing the way? (Fixes)

- Architects thinking only about infra-structure

Architects are “freak“ people that usually think about technical stuff and forget what is important to the customer.

We have to think about business too.

Using DDD design will focus on business situations. The internal transfer of knowledge is easier since communication will be improved as well.

Where are we missing the way? (Fixes)

- We have lost real Object Orientation

Object orientation is about the use of objects and their interactions to design software.

At design time, we have to find objects from the real world, to use for the specific case.

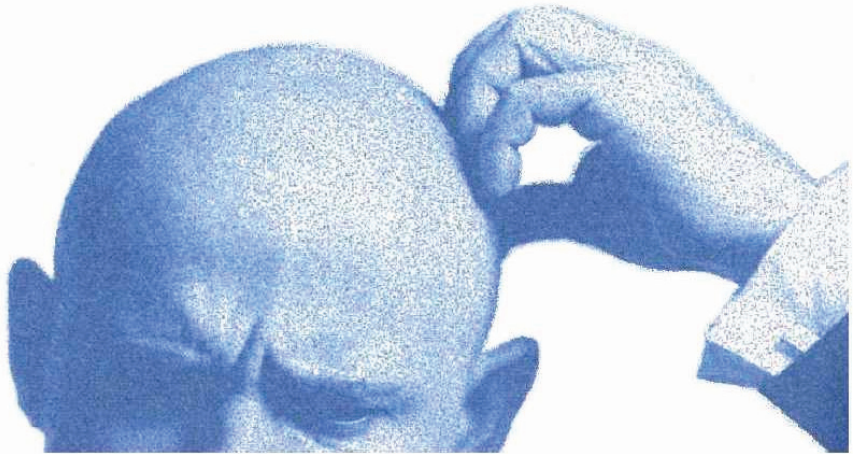
Each domain model is singular and is not applicable to another model.

THE MEANING

Wait a Minute!

You said: *"Each domain model is singular and is not applicable to another model."*

If I am in an enterprise, why should I have more than 1 model to describe a single "thing", perhaps even more that 1 model per application. As a proponent of DRY, this doesn't make sense to me.



Layered Architecture

A layered architecture at heart, is a way to separate concerns.

Domain model depends on some infrastructure, but not on a specific impementation.

THE CONCEPT

Presentation Layer

Responsible for interacting with the user, getting information from the user and showing information to the user.

The presentation layer usually depends on the application layer or Data Source Layer!

THE CONCEPT

Application Layer

Works as delivery, coordinating tasks which the system is responsible for doing. This layer has no business rules and is only responsible for delegating work to domain objects. Its objects do not have state to reflect business situation, but can have state of a task's progress.

Consists of a set of decorators, delegating calls to domain layer.

THE CONCEPT

Data Source Layer

A set of generic technical capabilities, providing support for higher layers.

Here goes the most technical things!

THE CONCEPT

Domain Layer

Represents the concepts of business information and business situation.

A set of generic technical capabilities, providing support for higher layers.

This layer is the heart of business software!

Domain Objects

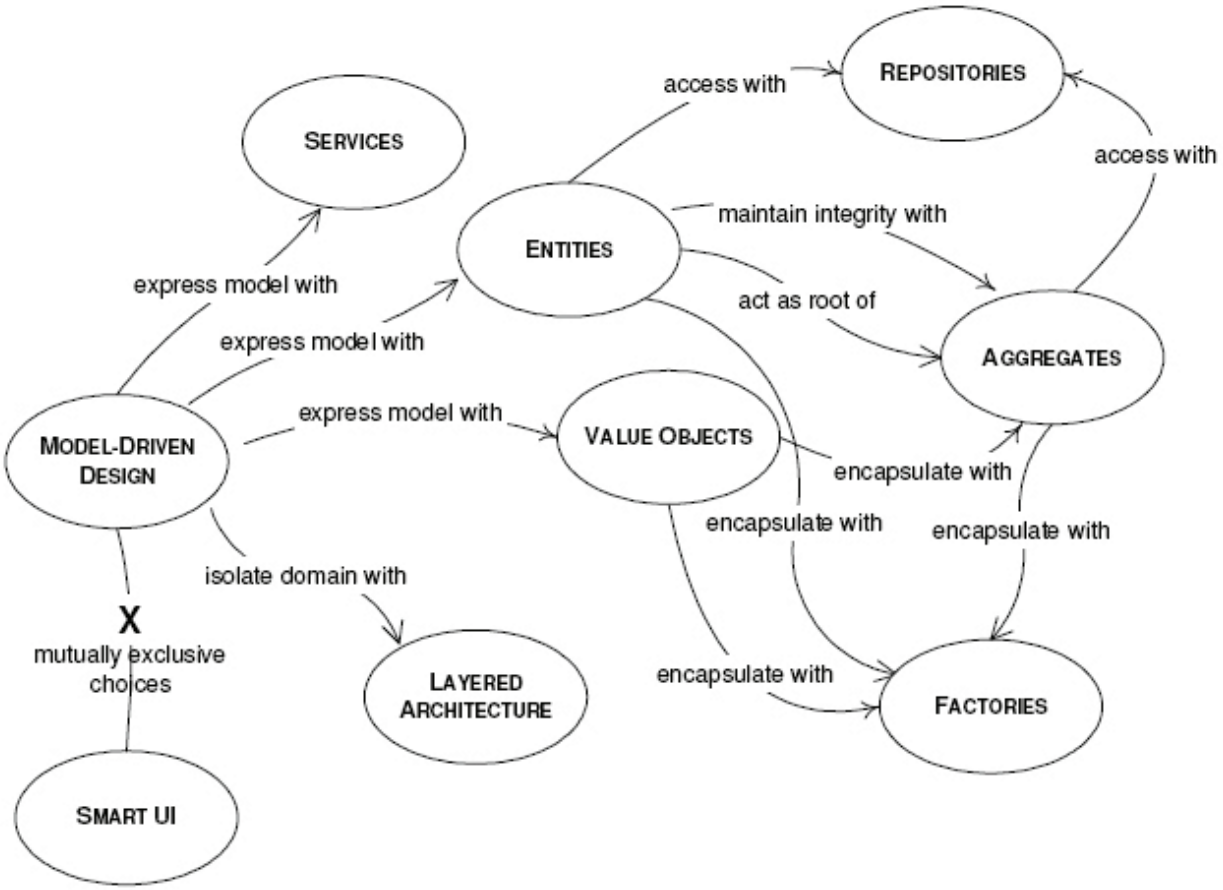
Domain Objects are instances of real entities which hold any knowledgenemt, influency or activity of a situation.

We can figure out which domain objects we need, by talking to domain experts.

Focusing on key words will help to define domain objects.

THE WAY BACK

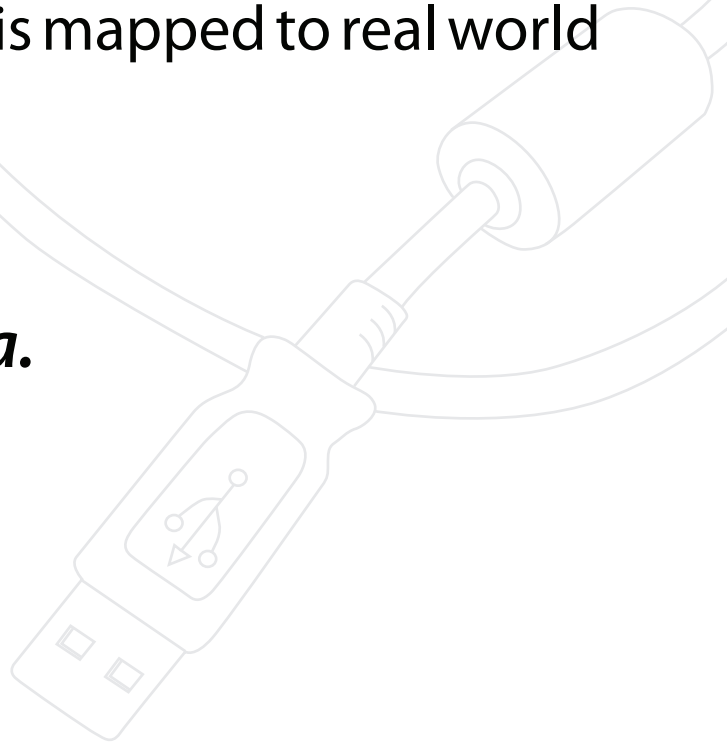
Domain Objects



Entities

Any object which has an identity. Usually is mapped to real world objects.

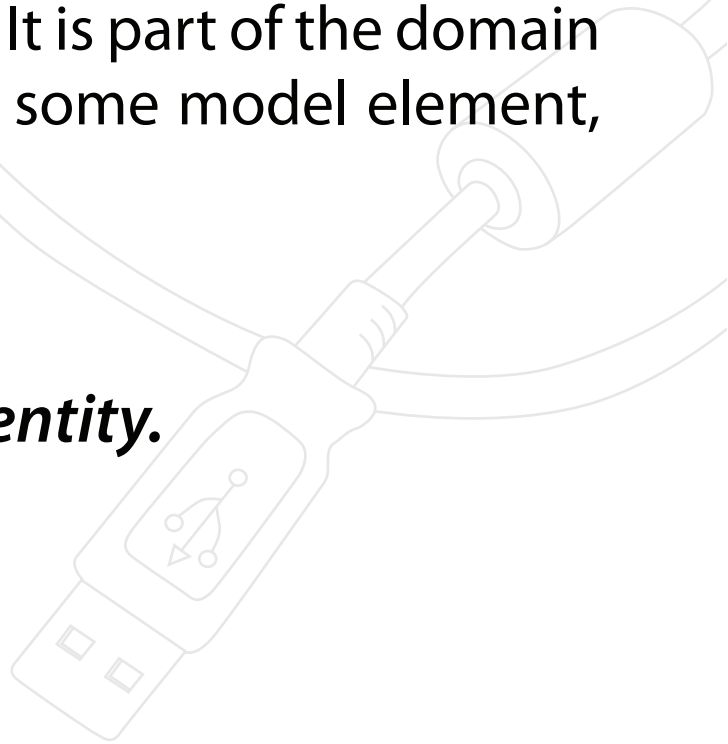
Unique set of data.



Value Objects

Used to describe characteristic of things. It is part of the domain as much as entities, but is always under some model element, since it has no identity.

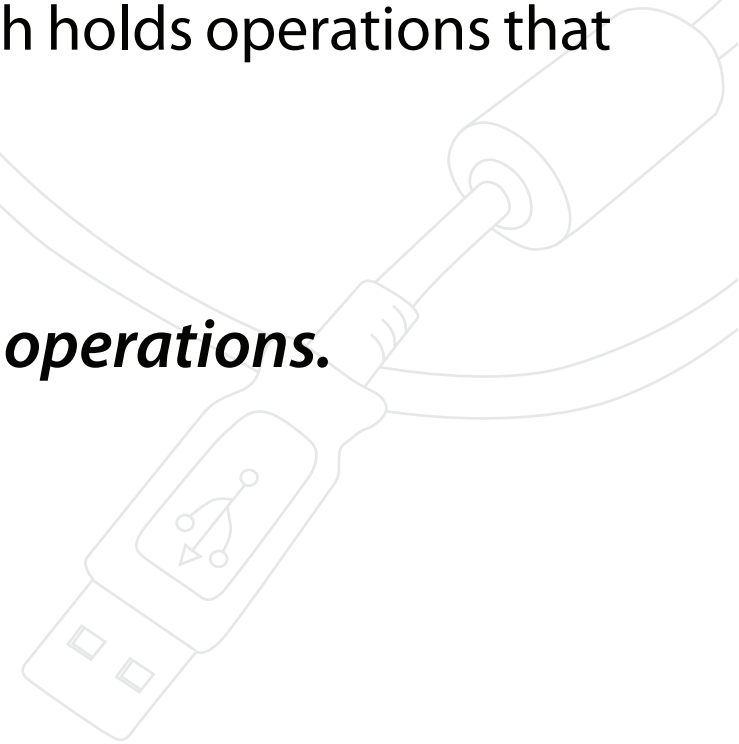
Set of data without identity.



Services

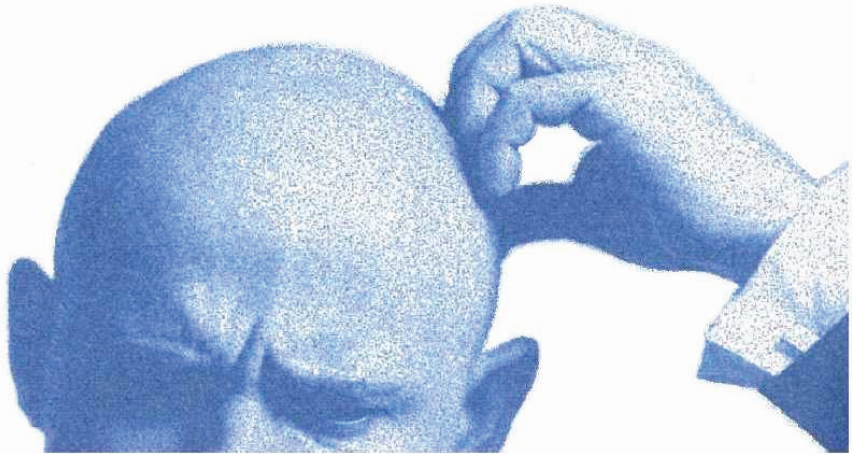
Services are a standalone interface, which holds operations that don't fit in any object in the model.

It is not a thing, but a set of operations.



Wait a Minute!

You've said earlier that DDD is better than service/manager models, but here you're saying that DDD uses services. Make sure that you describe how this type of service is different from the other.



THE WAY BACK

Modules

Organize your domain objects by modules, then whenever you need anything you know where it is.

Don't drive your modules by infrastrucure.

Objects Life Cycle

Once you have decided the main objects of your system, you then know about its life cyle. How will it interact in the model? Who is able to use it?

Aggregate, Factory and Repository patterns stand for objects life cycle.

The point is avoid showing internal complexity to the client.

THE WAY BACK

Aggregates

A set of objects with a unique interface for external calls. Indicated for cases where you need keep consistent a set of objects.

Aggregates always have a root responsible for being the interface for others elements.

THE WAY BACK

Factories

Responsible for creation of domain objects or even an entire aggregate.

Required only when creation becomes too complicated.

Repositories

Works as a “repository“ for domain objects. Provides a simple interface for complex persistence operations. Can encapsulate several data sources for a object.

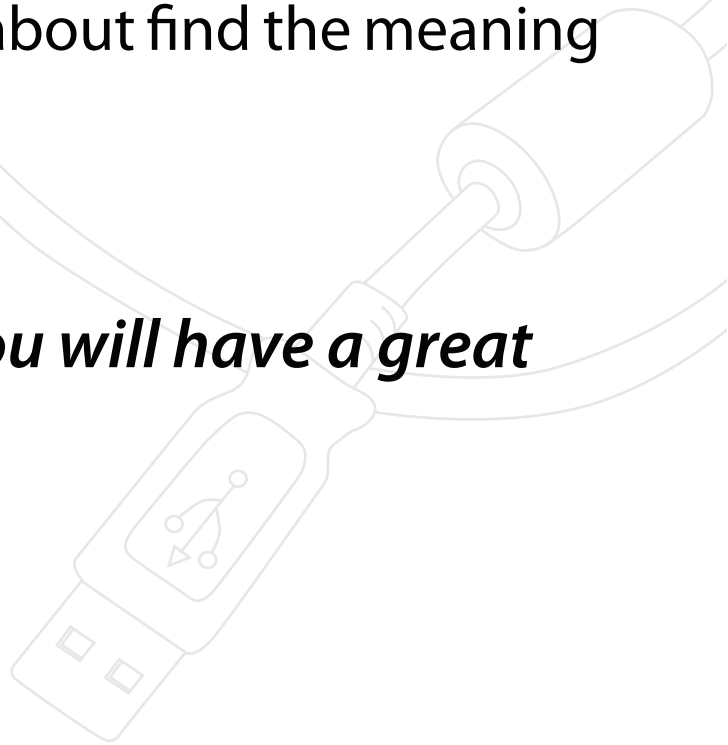
Make a repository whenever you find a type that needs global access.

THE MEANING

Meaning what?

All software has a meaning. Modeling is about find the meaning of software.

Find the meaning, express it and you will have a great model.



THE MEANING

Ubiquitous Language

Created by the team, maintained by the team and for the team. Ubiquitous Language has to be concerned about the business in question.

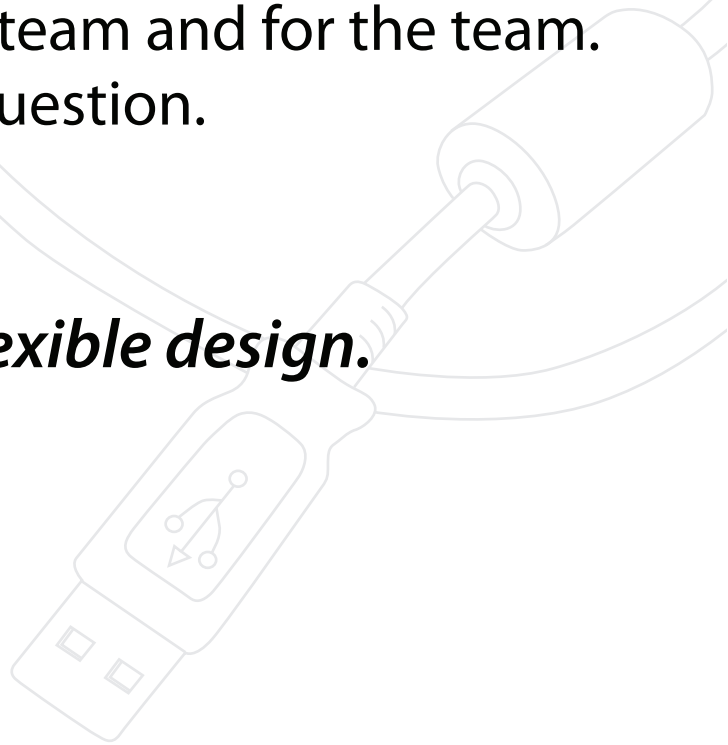
Works like a dictionary, composed by the terms taken from Domain Experts.

THE MEANING

Supple Design

Created by the team, mainteined by the team and for the team.
Have to concern about the business in question.

A set of concepts to make a flexible design.



Supple Design - Some Patterns

- Intention-Revealing Interfaces
- Standalone Classes
- Conceptual Contours

THE MEANING

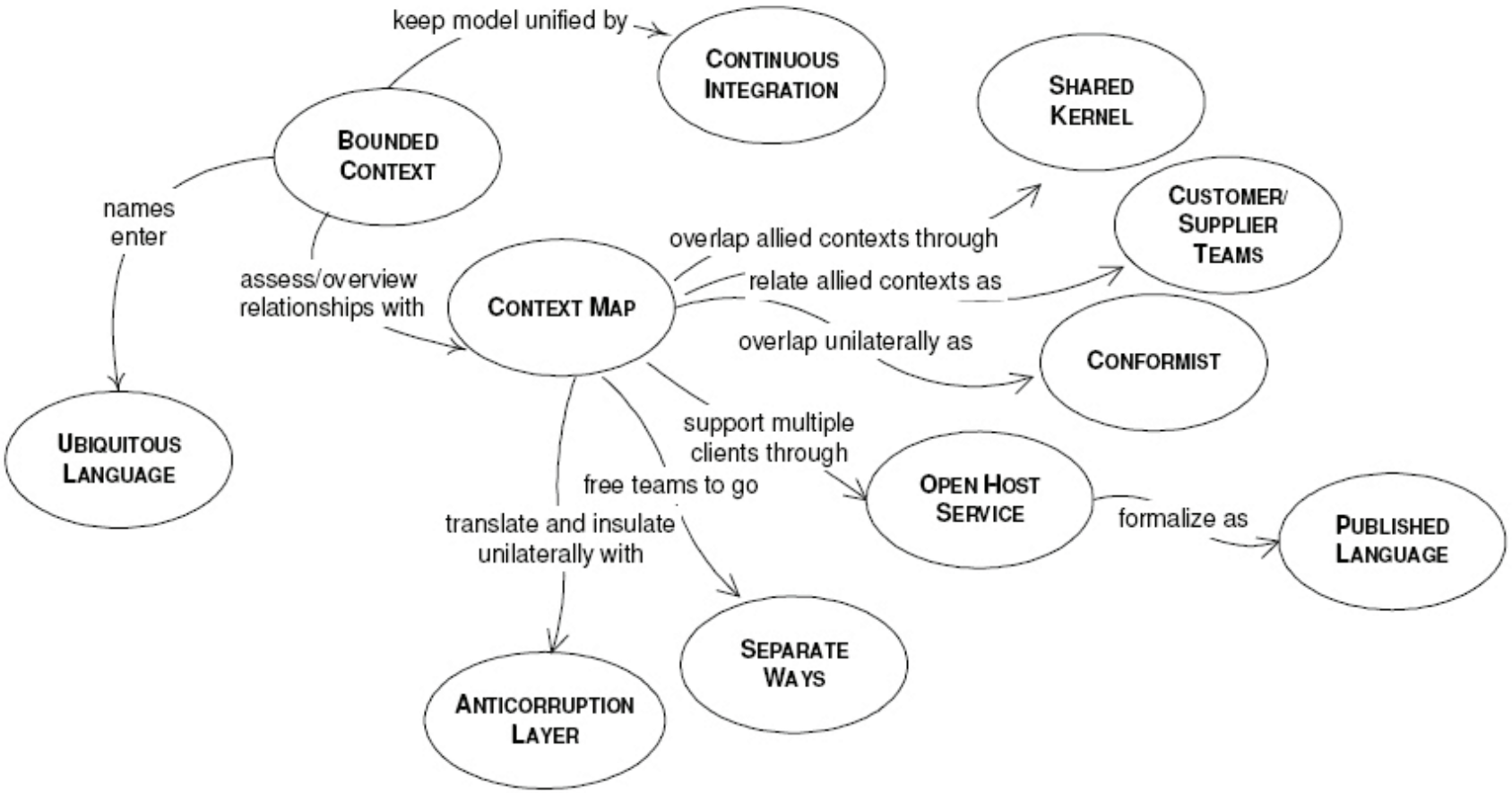
Strategic Design

A set of good pratices to achieve a good Domain Model. These patterns can works as a check list to improve the design quality.

If you want you can express these patterns as documents.

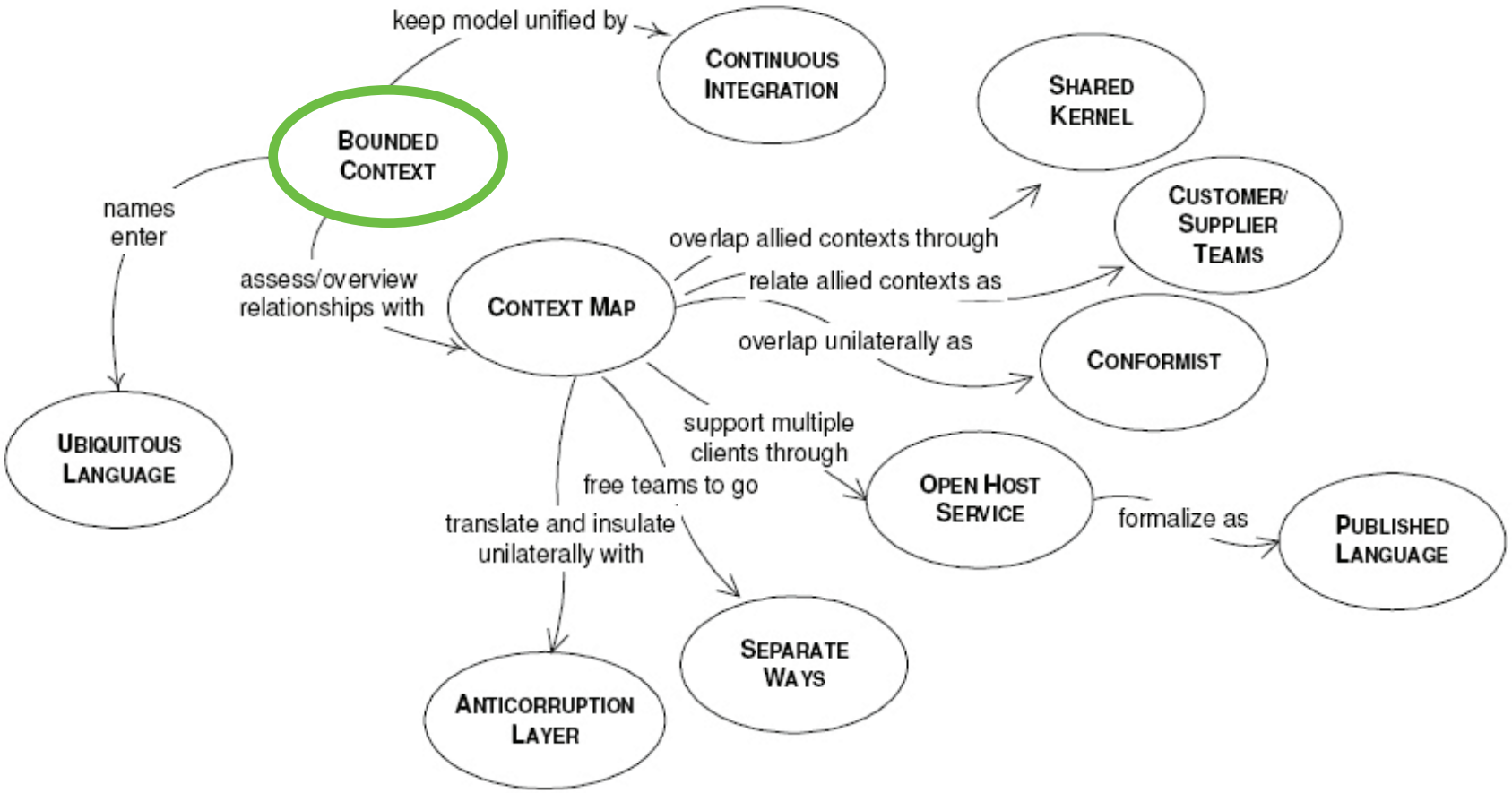
THE MEANING

Strategic Design



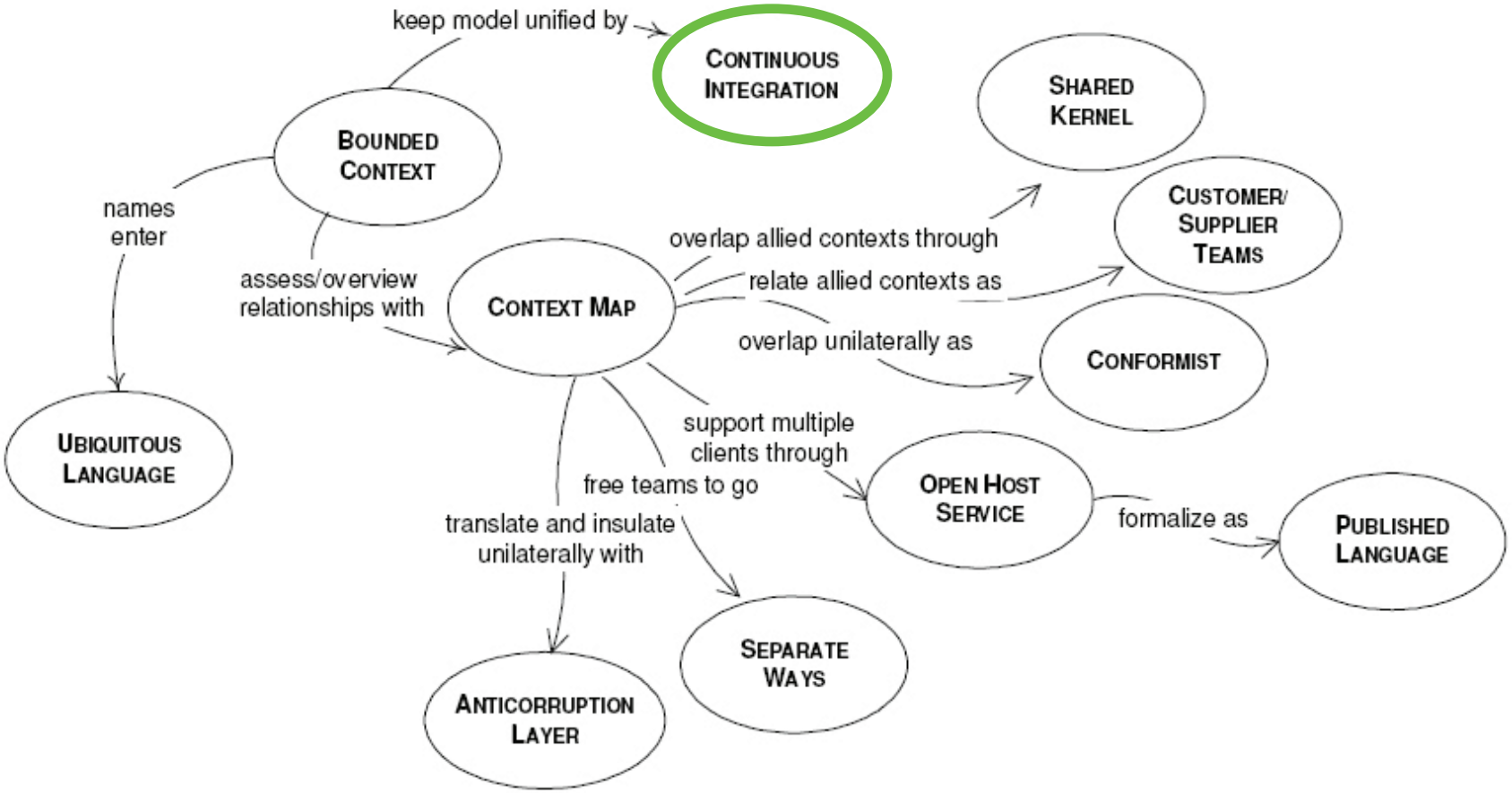
THE MEANING

Strategic Design



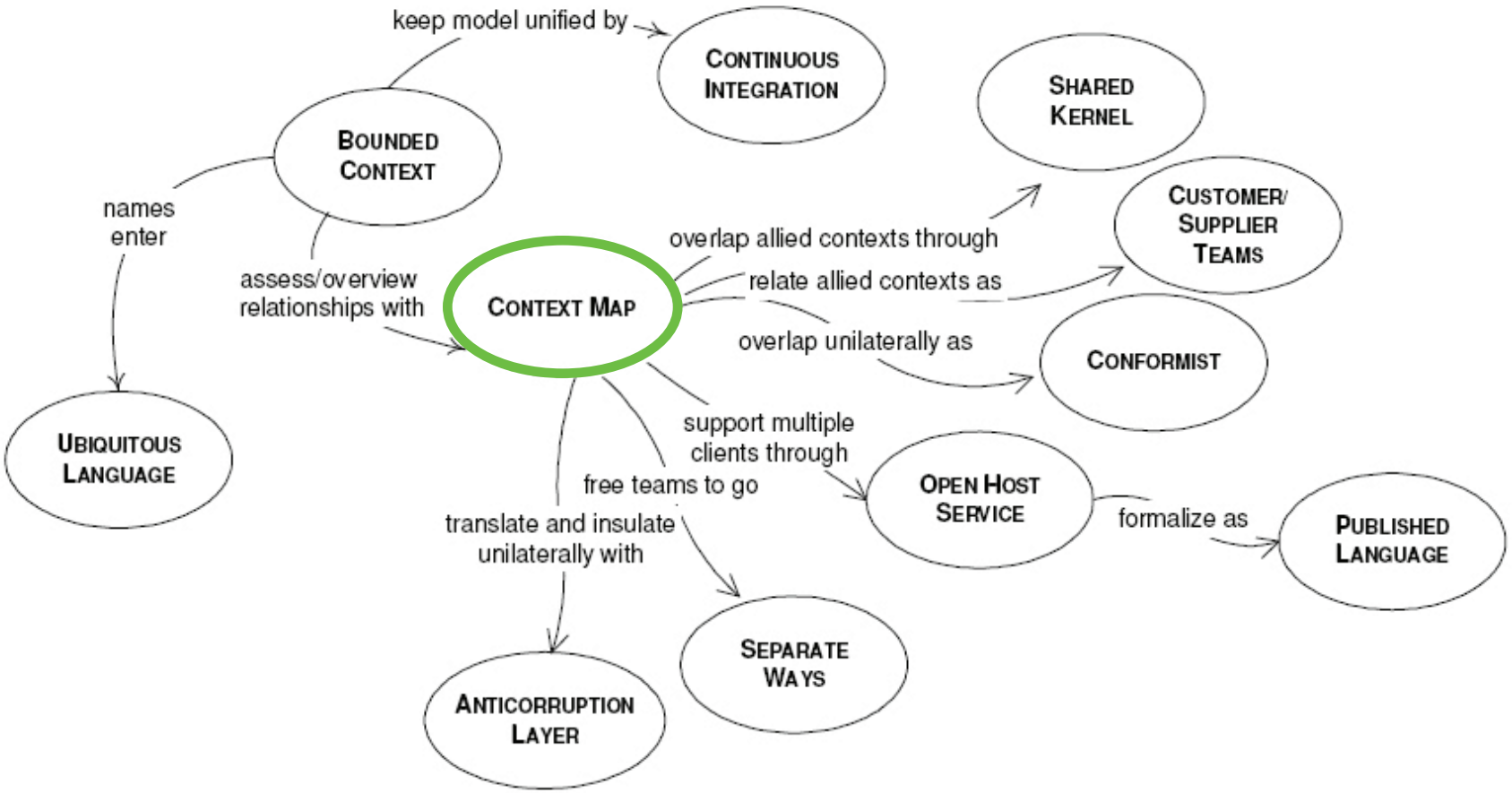
THE MEANING

Strategic Design



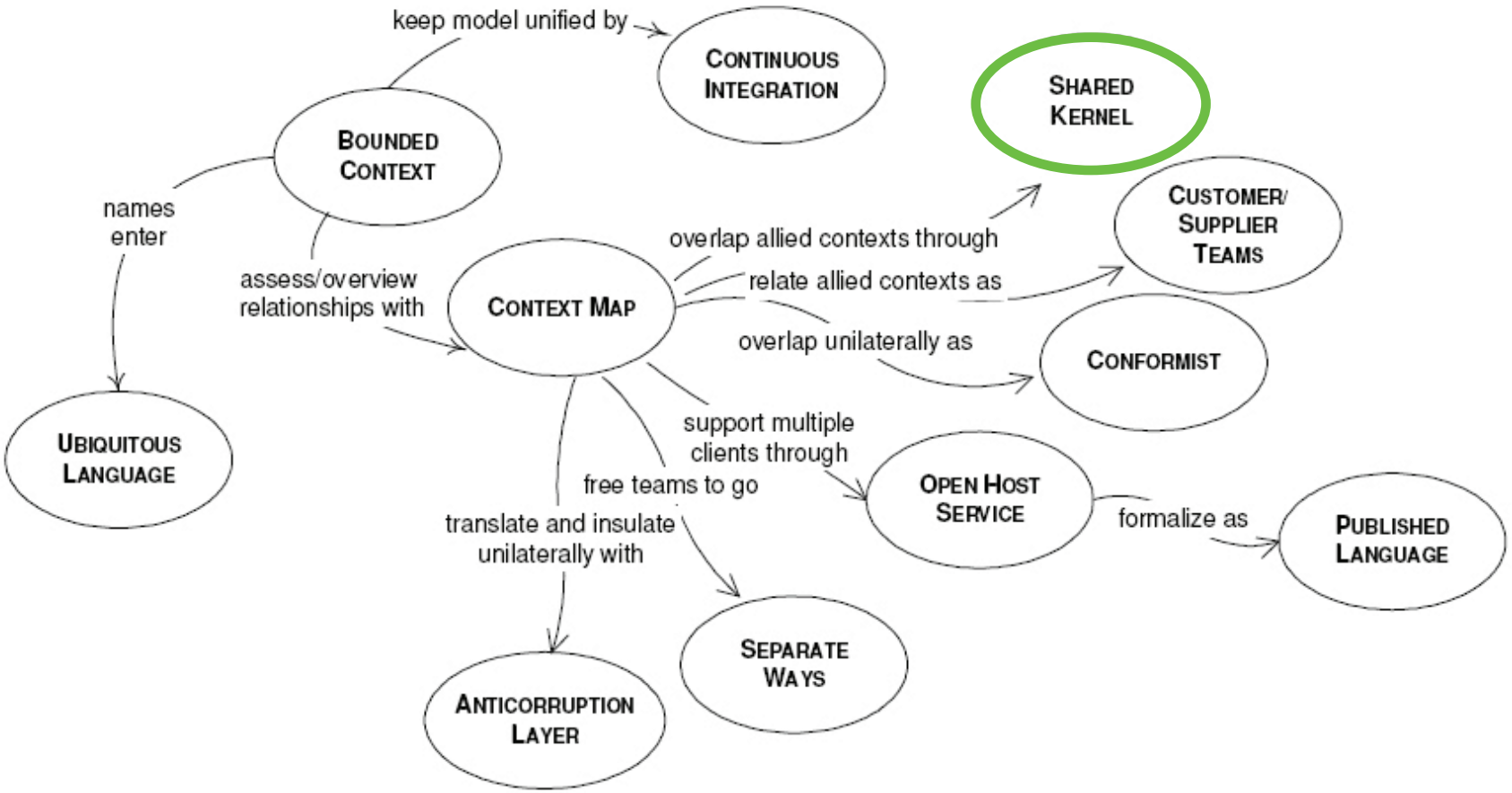
THE MEANING

Strategic Design



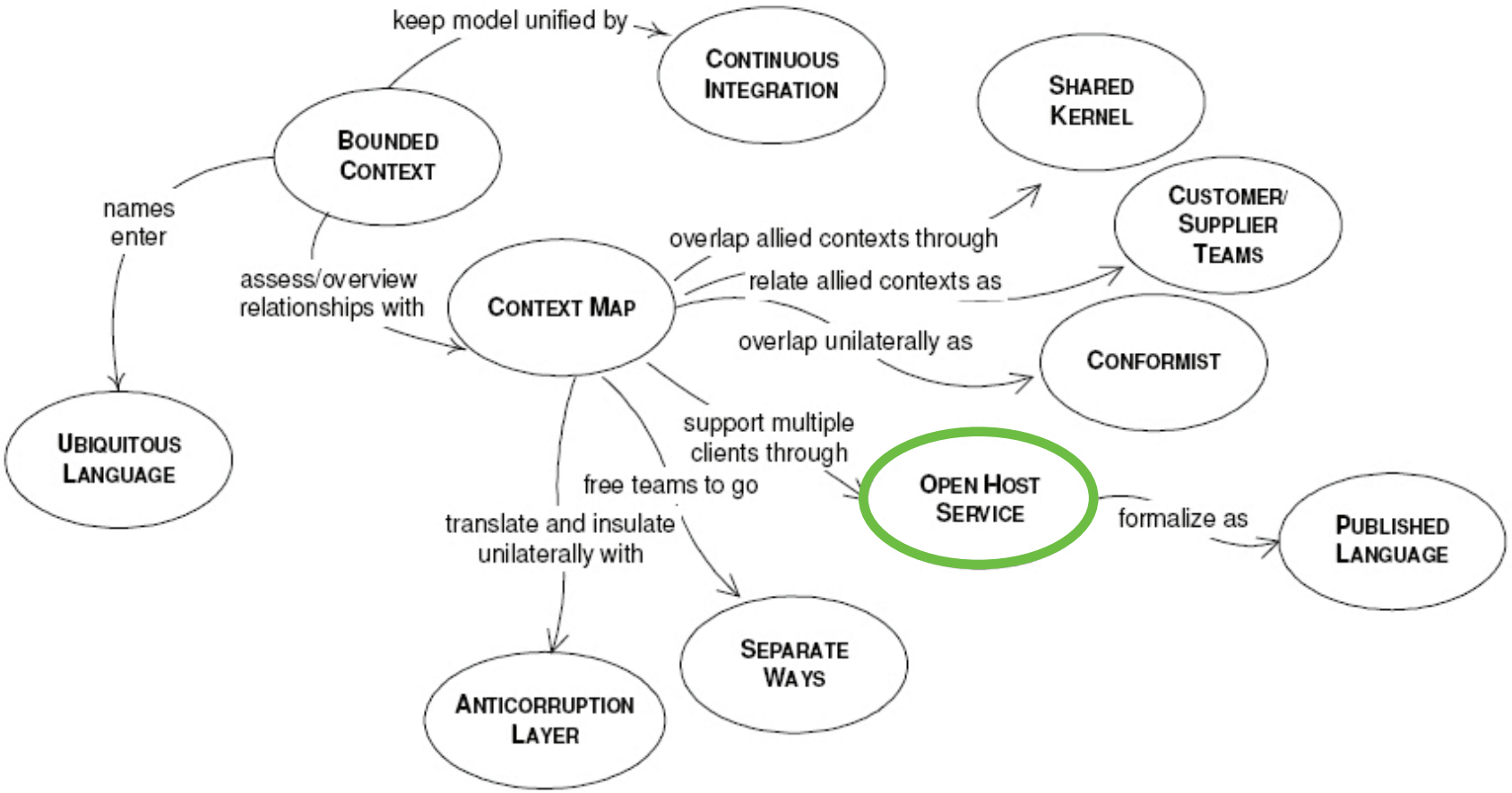
THE MEANING

Strategic Design



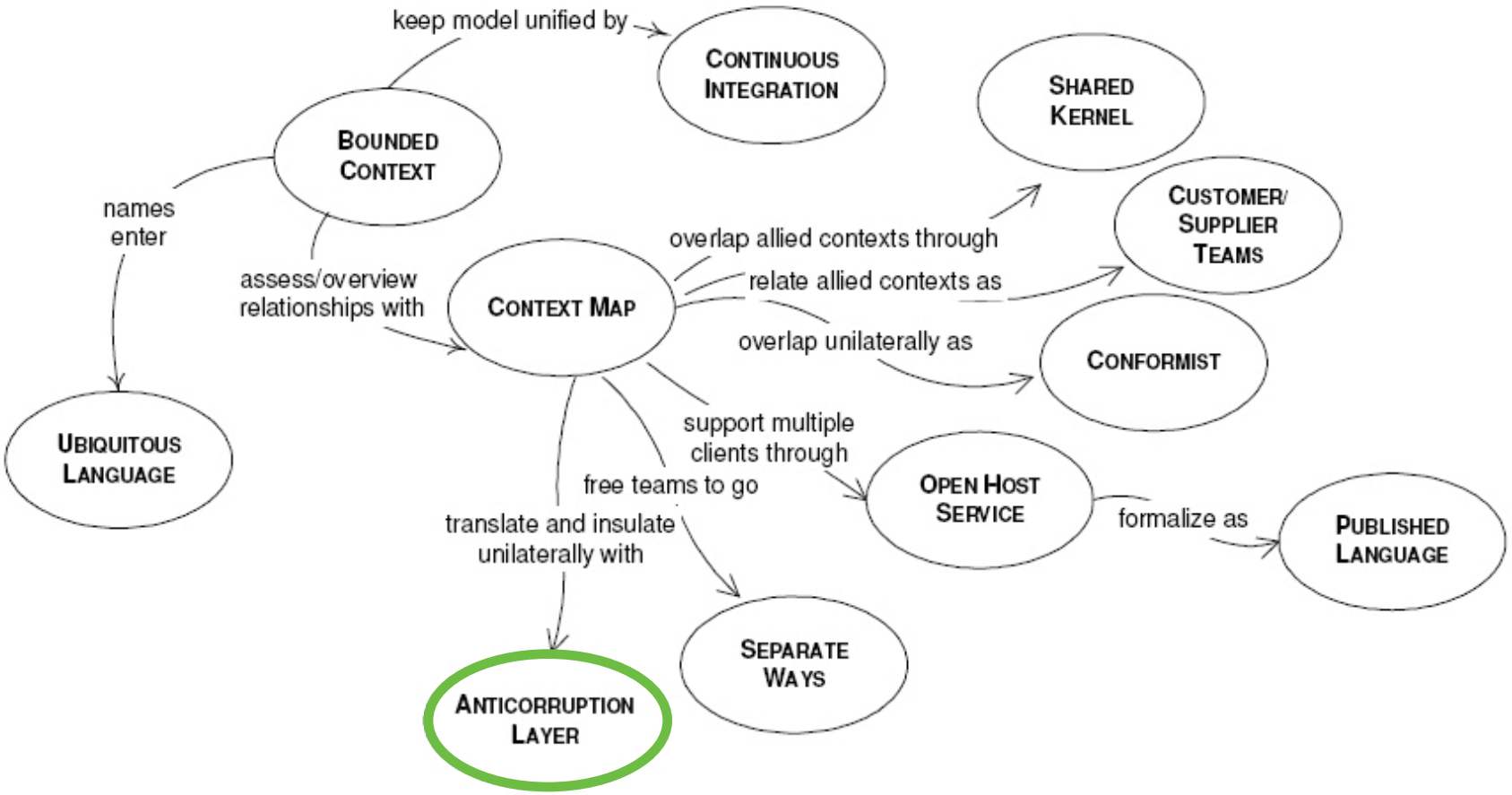
THE MEANING

Strategic Design



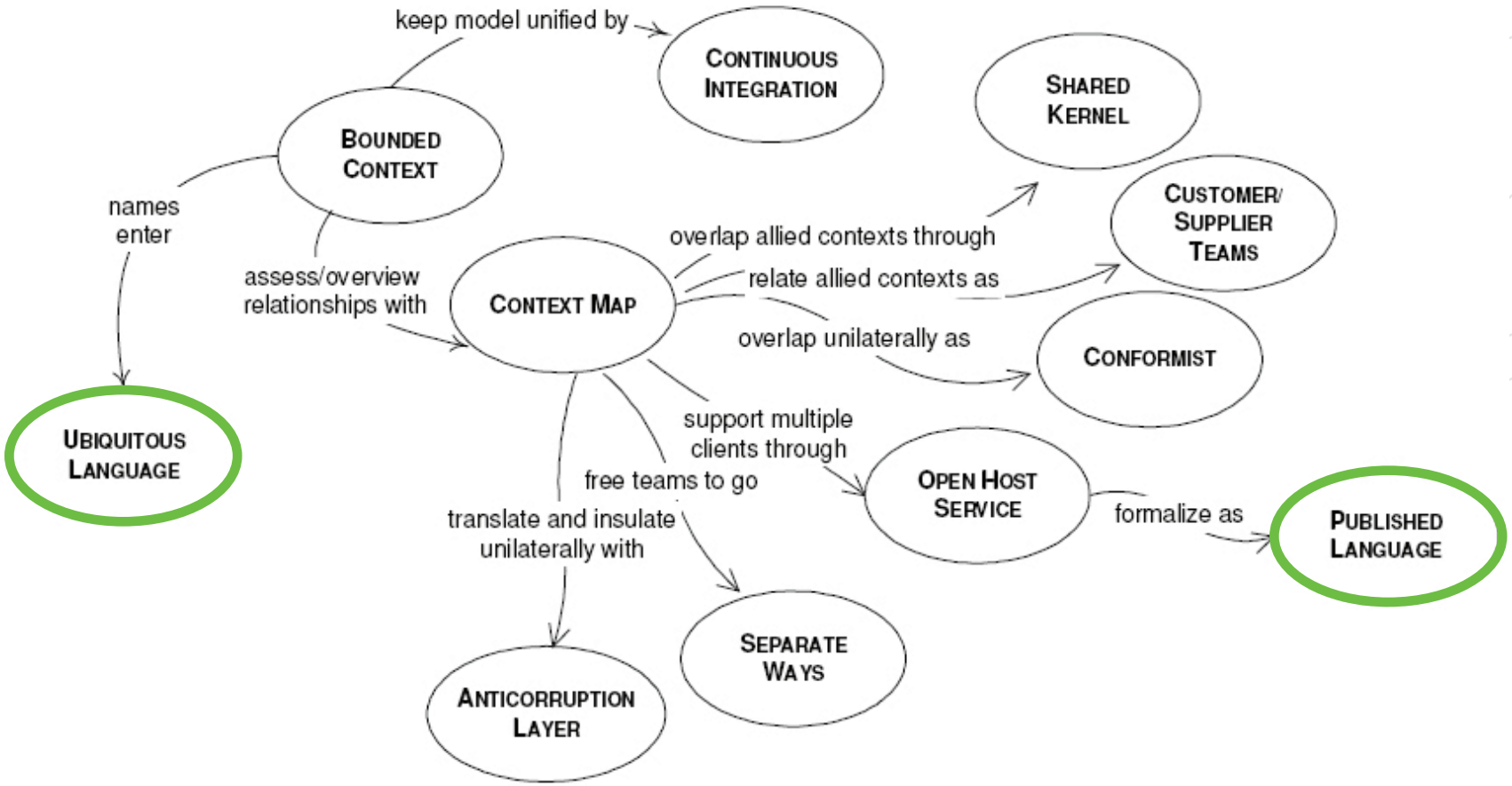
THE MEANING

Strategic Design



THE MEANING

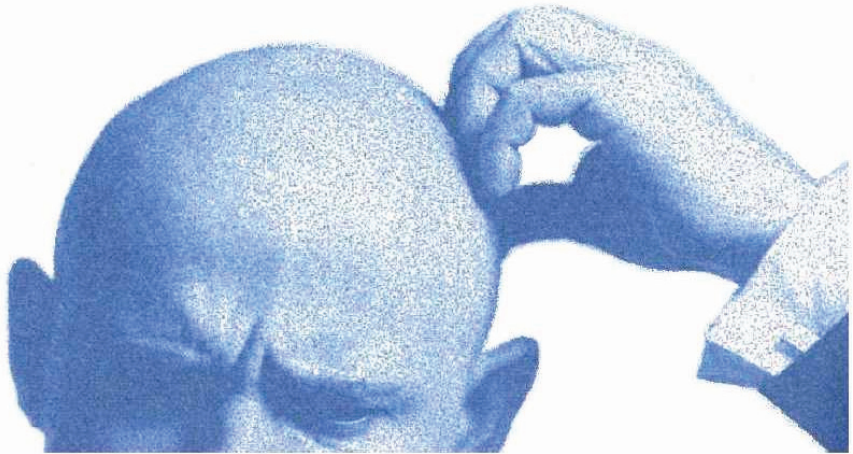
Strategic Design



THE MEANING

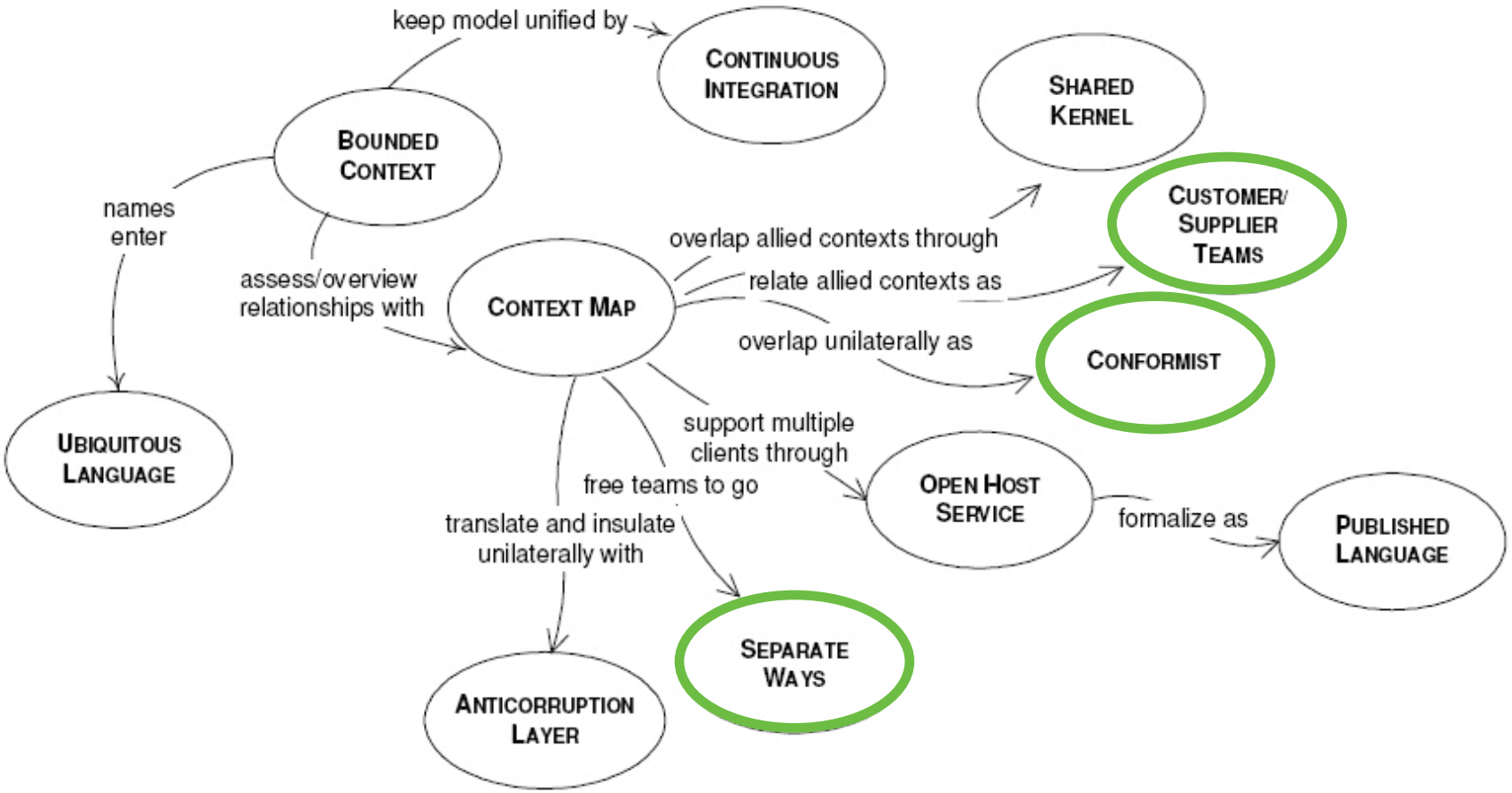
Wait a Minute!

Have you considered making a connection between "Published Language" and DSLs? I forgot about this aspect, but using DDD is a fantastic way to have an architecture that can transform into a DSL via this route.



THE MEANING

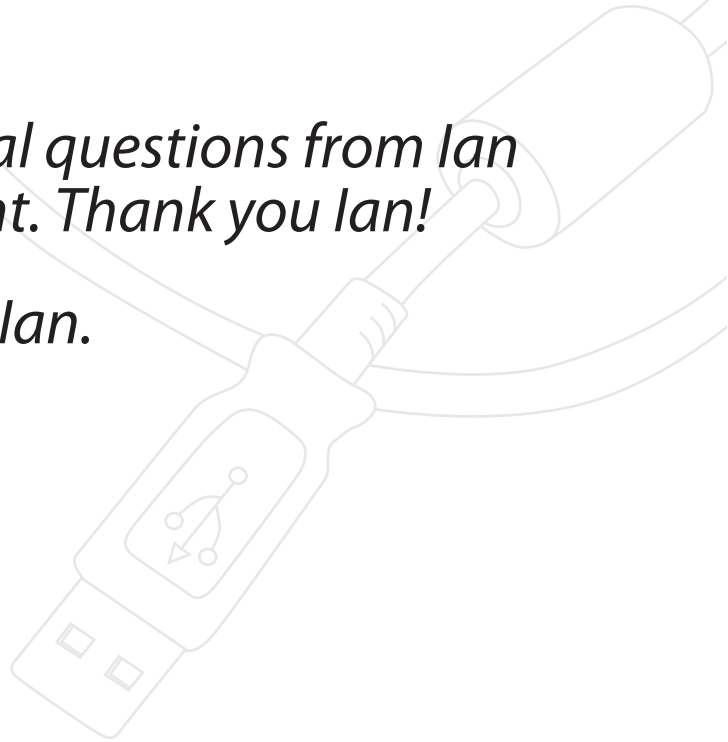
Strategic Design



WAIT A MINUTE!

** The "Wait a minute!" questions are real questions from Ian Roughley after review this document. Thank you Ian!*

*** No, that man is not Ian.*



Obrigado!

Felipe Rodrigues de Almeida

felipe@fratech.net

blog.fratech.net