Problem:

We need a model to represent a particular problem which can be called as Domain Model. The next question is "How accurate and perfect the model should be?". To constrcut a model we need tools to construct effective modelling like

Design principles, Design patterns, Design philosophies.

When working in a large organization that deals with a complex domain, there are still problems that are unaddressed by the principles, patterns, and philosophies listed above.

For example, When multiple people were given the task of modelling domain, we get different models completely. The models will be structurally and linguistically different.

The domain experts may have their own language to express certain domain concepts, and the vocabulary used by the domain experts may be completely different from the vocabulary used by the technical experts.

This divide is very common in large organizations, you can notice it in any discussions where people from two different groups (tech vs domain) are trying to define a problem or explore possible solutions to the problem.

Solution to this problem is to adapt Domain Driven Design (DDD).

Approach:

The word Domain used in context of software development refers to business.A domain in software engineering field is business on which application is intended to build.

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.

DDD talks about Ubiquitous language of a domain which means that everyone technical and business people should use the same language, same terms and same names to implement the software.

DD helps in making software a model of a real-world system or process. In using DDD, you are meant to work closely with a domain expert who can explain how the real-world system works.

Methodolgy:

Domain-driven design talks about two kinds of design tools, first one is Strategic design tools and another one is Tactical design tools.

Stractegic Design is approach similar to Object Oriented design which makes to think in terms of objects.

Strategic Design :

Model- A system of abstractions that describes selected aspects of a domain and can be used to solve problems related to that domain.

Ubiquitous Language – Common language used by all team members like technical and non technical people to connect alla activites of the software.

Bounded Context- It refers to boundary conditions of context. It is a description of a boundary and acts as a threshold within which, a particular domain model is defined and applicable.

Tactical Design :

Tactical design talks about implementation details i.e., modeling domain and this occurs during product development phase.

Entity- Entity is a class which has some properties. An entity implements some business logic and could be uniquely identified using an ID. In context of programming, it generally persisted as a row in DB and it consists of value objects.

Value Objects- Value objects reduce complexity by performing complex calculations, isolating heavy computational logic from entities and are immutable.

Services- service is a functionality that exists somewhere between entities and values objects but it is neither related to an entity nor values objects.

Aggregates- An aggregate is a collection of entities and values which come under a single transaction boundary. Aggregates basically, control change and have a root entity called aggregate roots. The root entity governs lifetime of other entities in aggregates.

Factories and Repositories –

Factories and Repositories are used to handle aggregate. Factories help in managing beginning of lifecycle of aggregates whereas Repositories help in managing middle and end of lifecycle of an aggregate. Factories help in creating aggregates whereas Repositories help in persisting aggregates. We should always create a repository per aggregate root but not for all entities.

A picture containing text, diagram, map, font

Description automatically generated