DOMAIN-DRIVEN DESIGN

POR

ASTRID CAROLINA GOMEZ

LUIS CARLOS MARÍN CAMPOS

ANDRES DARIO HIGUITA PEREZ

MARIO DE JESUS PUENTES SANCHEZ

THE DIVINES

PRESENTADO A:

ROBINSON CORONADO GARCIA

ARQUITECTURA DE SOFTWARE

INGENIERÍA DE SISTEMAS

UNIVERSIDAD DE ANTIOQUIA

2020 - 1

**Domain-driven design**

El diseño orientado a el dominio es una metodología para mapear, entender y resolver un problema dentro de una esfera de dominio; es un enfoque de desarrollo de software que utiliza y se basa en principios e ideas de el análisis y diseño orientado a objetos mediante una profunda conexión entre la implementación y los conceptos del modelo y núcleo del negocio, en este la estructura y el lenguaje del código de software deben coincidir con el dominio empresariales. Se trata de respetar nuestro dominio lo máximo posible ya que el DDD representa las distintas claves, la terminología y los patrones que se utilizan para el desarrollo del software donde el dominio es lo más central e importante de una determinada organización.

Sus principios se basan en:

* Colocar los modelos y reglas de negocio de la organización, en el core de la aplicación
* Basar nuestro dominio complejo, en un modelo de software.
* Se utiliza para tener una mejor perspectiva a nivel de colaboración entre expertos del dominio y los desarrolladores, para concebir un software con los objetivos bien claros.

**Ventajas de usar DDD:**

* Comunicación efectiva entre expertos del dominio y expertos técnicos a través de lenguaje ubicuo.
* Foco en el desarrollo de un área dividida del dominio a través del límite del contexto.
* El software es más cercano al dominio, y por lo tanto es más cercano al cliente.
* Código bien organizado, permitiendo el testing de las distintas partes del dominio de manera aislada.
* La lógica de negocio reside en un solo lugar, y dividida por contextos.
* Mantenibilidad a largo plazo.

**Desventajas**

* Aislar la lógica de negocio con un experto de dominio y el equipo de desarrollo suele llevar mucho esfuerzo a nivel tiempo.
* Necesitamos un experto de dominio
* Una curva de aprendizaje alta, con patrones, procedimientos, etc.
* Este enfoque sólo es sugerido para aplicaciones donde el dominio sea complejo, no es recomendado para simples CRUD’s.

Algunos de los conceptos del Diseño orientado a el dominio son:

* **Dominio:** Es el problema específico que estamos intentando resolver del mundo real. Representa la terminología y los conceptos clave del dominio del problema. Identifica las relaciones entre las entidades incluidas dentro del ámbito del dominio del problema, identifica sus atributos y proporciona una visión estructural del dominio.
  + Core Domain: El diferenciador clave para el negocio del cliente: algo que deben hacer bien y no pueden subcontratar.
  + Subdominio: Es la segregación de un dominio más general en uno más acotado, cohesivo y comprensible.
* **Contexto:** El escenario en el que aparece una palabra o declaración que determina su significado
* **Modelo:** Un sistema de abstracciones que describe aspectos seleccionados de un dominio y se puede usar para resolver problemas relacionados con ese dominio
* **Lenguaje ubicuo (común):** Es un lenguaje estructurado en torno al modelo de dominio y utilizado por todos los miembros del equipo para conectar todas las actividades del equipo con el software.La comunicación efectiva entre los desarrolladores y los expertos del dominio es esencial para el proyecto.
* **Entidades:** Las entidades son objetos del modelo que se caracterizan por tener identidad en el sistema, los atributos que contienen no son su principal característica. Deben poder ser distinguidas de otros objetos aunque tengan los mismos atributos. Tienen que poder ser consideradas iguales a otros objetos aún cuando sus atributos difieren.
* **Value objects:** Representan conceptos que no tienen identidad. Simplemente describen características. Por lo tanto solo nos interesan sus atributos. Los value object representan elementos del modelo que se describen por el que son.
* **Services:** Los servicios representan operaciones, acciones o actividades que no pertenecen conceptualmente a ningún objeto de dominio concreto. Los servicios no tienen ni estado propio ni un significado más allá que la acción que los definen, tienden a ser nombrados como verbos.
* **Arquitectura por capas:** Se debe dividir el sistema en al menos cuatro capas: presentación, aplicación, dominio e infraestructura.
* **Repositorio:** Los métodos para recuperar objetos de dominio deben delegarse en un objeto Repository especializado de modo que las implementaciones de almacenamiento alternativas se puedan intercambiar fácilmente.
* **Factory:** Los métodos para crear objetos de dominio deben delegarse en un objeto Factory especializado, de modo que las implementaciones alternativas se puedan intercambiar fácilmente.
* **Aggregate:** Una colección de objetos que están unidos por una entidad raíz, también conocida como raíz agregada. La raíz agregada garantiza la coherencia de los cambios que se realizan dentro del agregado al prohibir que los objetos externos tengan referencias a sus miembros.

En conclusión el diseño orientado al dominio es una técnica que está estructurada por varias prácticas que pueden ayudar a tomar las decisiones de diseño con el fin de enfocar y acelerar el manejo de dominios complejos durante el desarrollo del software, en donde este debe ser iterativo y debe existir una estrecha relación entre los desarrolladores y los expertos del dominio. Esta técnica fue ideada para el desarrollo de aplicaciones complejas y está orientada a proyectos que usen metodologías ágiles.