ENSAYO

DOMAIN- DRIVEN DESIGN

Autores:

Carlos Enrique Morales Rodríguez Id:1077420979
Alcides Antonio Zapata Méndez Id: 8055453
Leonel Alfonso Zapata Méndez Id: 1038100562
Dilsa Doriela Barrientos Ciro Id: 1037237118

Curso:

Arquitectura de Software

Docente:

Robinson Coronado García

Universidad de Antioquia

Facultad de Ingeniería

Ingeniería de Sistemas

2020

DOMAIN- DRIVEN DESIGN
(DISEÑO IMPULSADO POR EL DOMINIO)

La arquitectura de software es una disciplina que estudia la manera de garantizar ciertas cualidades del software como la disponibilidad, la continuidad, la mantenibilidad, la escalabilidad, la reusabilidad, la observabilidad, etc. Dicho esto, es importante entender que no es posible diseñar un software que goce del máximo nivel en todas las cualidades. O, dicho de otro modo, ciertas cualidades son opuestas con otras. De ahí, se puede inferir que es difícil encontrar diseño perfecto en el software, lo que nos obliga a entender en profundidad las necesidades del negocio para aplicar la arquitectura que garantice los requerimientos necesarios en nuestro contexto, teniendo muy presente que otras tendrán que ser sacrificadas.

Domain Driven Design (desde ahora DDD) es un enfoque de desarrollo de software utilizado por Eric Evans en su libro *"Domain*-Driven Design — Tackling Complexity in the Heart of Software*, 2004",* que representa distintas claves, terminologías y patrones utilizados para desarrollar software, donde el dominio es lo más central e importante de una determinada organización. Es una aproximación holística al diseño de software que va más allá de los aspectos técnicos, centrándose en aspectos culturales y organizacionales de una empresa que busca entregar valor, dicho enfoque (DDD) nos ayuda, en primer lugar, a definir exhaustivamente el dominio; nos permite también conocer de modo extenso el problema de negocio, de manera que podamos dividirlo en subdominios con el objetivo de maximizar el desacoplamiento entre los subdominios y lograr así soluciones modulares, que premian la adaptabilidad de cada subdominio y por lo tanto, la adaptabilidad de la solución en general.

Domain Driven Design abarca desde el diseño de todo un mapa de sistemas de una corporación hasta el diseño de un único sistema. Este enfoque se divide en dos grandes bloques: Bib Picture, que es todo lo referente a cómo separar los procesos de negocio, analizar cómo estos procesos se relacionan entre sí y cómo especificar los procesos de negocio y el Code Architecture, el cual hace referencia a cómo el DDD propone la estructura del código y la comunicación entre los componentes del mismo.

Para poner en funcionamiento los bloques anteriores en DDD existe algo llamado Strategic Design el cual propone: Separar durante el análisis, el espacio del problema del espacio de la solución, esto es necesario para un correcto entendimiento del problema y por lo tanto un mejor diseño de la solución. Cuanto más solapamiento haya entre la solución y el problema, mejor. El DDD se centra en la búsqueda de los siguientes aspectos:

- Buscar que los expertos de negocio y técnicos trabajen juntos en entender el dominio (Domain), es decir el problema.
- Generar la separación del dominio, de los aspectos tecnológicos, lo que permite una mejor comprensión del dominio.
- Aporta herramientas para dividir el dominio en parcelas más pequeñas, lo que lleva a soluciones modulares.

El dominio o Domain es, en términos de DDD, el problema de negocio para el cual está diseñada la solución. El primer ejercicio que propone DDD es entender de la mano de los expertos el Domain, el problema de negocio, para ello se pueden realizar dinámicas inclusivas como Event Storming, siendo muy importante que se hable en exclusiva del problema, sin referir a la tecnología, a la solución, en síntesis Domain trata de el todo, es decir de todos los procesos de la empresa (Big picture) y está formado por subdominios.

Otro concepto importante en el Domain Driven Design es el Ubiquitous Language, Eric Evans lo define como "el lenguaje que especifica el dominio y es importante por que a partir de ese lenguaje es que nos permite o nos ayuda a dividir el dominio en subdominios", hay que tener en cuenta que el lenguaje es regulador del pensamiento, entonces cuando se piensa en un concepto, es primordial tener claro como estamos definiendo dicho concepto y ese es un aspecto importante entre IT y el negocio; el Ubiquitous Language es un lenguaje consensuado y empleado por los expertos de negocio para definir y tratar un subdominio en concreto, hay que tener en cuenta que distintos Ubiquitous Languages significa distintos subdominios, y estos evolucionan a medida que evoluciona el conocimiento sobre el modelo.

El Bounded context es otro concepto que utiliza Domain Driven Design, este ya no pertenece al problema sino a la solución y es una herramienta básica para dividir el Domain en subdomains. Se identifican en la base del Ubiquitous Language, pues son las fronteras que al ser cruzadas cambian el entendimiento de los conceptos. Los beneficios que ofrece dividir el dominio en subdominios mediante Bounded context son los siguientes:

- Mejora la especialización de los equipos y el alineamiento entre el negocio e IT.
- La división del problema lleva a una solución con más cohesión y reduce la dependencia entre equipos.
- Como hay un desacoplamiento fuerte entre los subdominios, pueden ser desarrollados con tecnologías distintas.

En conclusión, Domain Driven Design es una técnica que implica dos transformaciones, del negocio al modelo y del modelo al software. El modelo juega un papel central para asegurar la correspondencia entre el software y el negocio al que debe representar. El modelo de dominio es el punto de partida importante al emprender un proyecto de diseño dirigido por dominio. El diseño basado en dominios se trata de resolver los problemas de una organización, por lo que el modelo de dominio se trata de comprender e interpretar los aspectos importantes de un problema dado.

Llegar a un modelo de dominio sólido para un problema dado es un enfoque iterativo de colaboración entre los expertos comerciales y el equipo técnico. Debe actuar como una fuente de verdad para el proyecto y es un documento vivo a medida que avanza el proyecto.