DOMAIN DRIVEN DESIGN

Profesor:

Robinson Coronado Garcia

Hecho por:

Luis Fernando Rios Zapata

Jaider Fabian Molina Velez

Universidad de Antioquia

Facultad de Ingeniería

Pregrado de Ingeniería de Sistemas

Antioquia – Colombia

Que es domain driven design(DDD)

Domain driven design viene de la expansión y aplicación del concepto de dominio aplicado al desarrollo de software haciendo facil la creacion de aplicaciones complejas relacionando las piezas de software en un modelo en constante evolución centrándose en tres principios:

- centrarse en el dominio central y la lógica conceptual
- basar los diseños complejos en modelos del dominio
- colaboración constante con expertos en el dominio para tener retroalimentación constante del modelo de la aplicación y darle solución a problemas futuros.

También es importante definir algunos conceptos comunes que son útiles al describir y discutir las prácticas de DDD:

- **Contexto**: el escenario en el que aparece una palabra o declaración que determina su significado. Las declaraciones sobre un modelo sólo pueden entenderse en un contexto.
- **Modelo**: un sistema de abstracciones que describe aspectos seleccionados de un dominio y se puede utilizar para resolver problemas relacionados con ese dominio.
- **Lenguaje ubicuo**: 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.
- **Contexto delimitado**: descripción de un límite (generalmente un subsistema o el trabajo de un equipo específico) dentro del cual se define y aplica un modelo en particular.

DDD y SOA

la arquitectura orientada a servicios ha venido ganando terreno, en el pasado reciente para ayudar a los equipos a desarrollar softwares y servicios basados en procesos comerciales en los diferentes procesos comerciales acelerando así el tiempo de comercialización de los productos nuevos generados por estos equipos. domain driven design se convierte en una pieza clave de la arquitectura SOA porque le ayuda a encapsular la lógica empresarial y las reglas en los objetos de dominio. El dominio nos proporciona el lenguaje y el contexto con los que se pueden definir los diferentes contratos de los servicios.

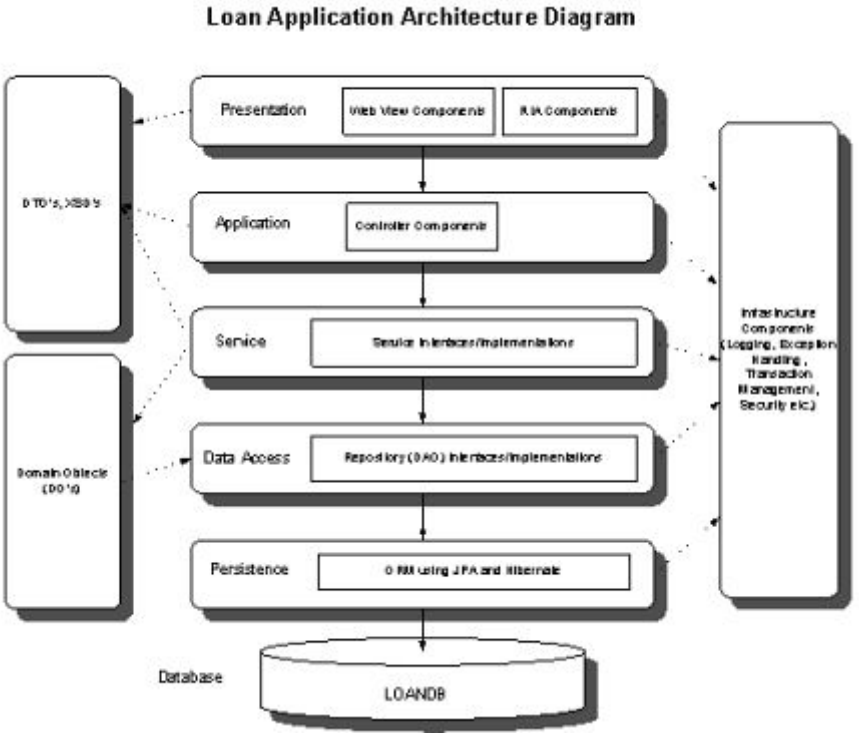
Un esfuerzo de SOA debe incluir el diseño y la implementación del modelo de dominio si aún no existe uno. Si ponemos demasiado énfasis en los servicios SOA e ignoramos la importancia del modelo de dominio, terminaremos con un modelo de dominio anémico y servicios inflados en la arquitectura de la aplicación.

Un escenario utópico es donde el que el esfuerzo de hacer la implementación de DDD de manera interactiva con el desarrollo de la capa de aplicación y de los componentes de SOA al mismo tiempo que estos son los consumidores directos de los diferentes elementos del modelo de dominio. Con una implementación de dominó enriquecida, el diseño de SOA se hará relativamente simple ya que proporciona un shell (proxy) a los objetos de dominio. Pero si nos centramos demasiado en las capa de SOA sin tener en cuenta un modelo

decente en nuestro back-end, los servicios empresariales llamarán un modelo de dominio incompleto, lo que puede dar resultado en una arquitectura SOA frágil

Ejemplo de arquitectura en domain driven design
Una arquitectura de aplicación empresarial típica consta de las siguientes cuatro capas conceptuales:

- **Interfaz de usuario (capa de presentación)**: responsable de presentar información al usuario e interpretar los comandos del usuario.
- **Capa de aplicación**: esta capa coordina la actividad de la aplicación. No contiene ninguna lógica empresarial. No contiene el estado de los objetos comerciales, pero puede contener el estado del progreso de una tarea de aplicación.
- **Capa de dominio**: esta capa contiene información sobre el dominio empresarial. El estado de los objetos comerciales se mantiene aquí. La persistencia de los objetos comerciales y posiblemente su estado se delega a la capa de infraestructura.
- **Capa de infraestructura**: esta capa actúa como una biblioteca de apoyo para todas las demás capas. Proporciona comunicación entre capas, implementa persistencia para objetos comerciales, contiene bibliotecas de soporte para la capa de interfaz de usuario, etc.



Referencias

https://airbrake.io/blog/software-design/domain-driven-design
https://www.infoq.com/articles/ddd-in-practice/
https://books.google.com.co/books?id=X7DpD5g3VP8C&printsec=frontcover&dq=domain+driven+design&hl=es&sa=X&ved=2ahUKEwiNgYfz89jtAhXbElkFHYG-AZgQ6AEwAXoECAUQAg#v=onepage&q&f=false