Domain Driven Design (DDD)

Saraí Castillo Hernández

Los sistemas de software, al igual que la mayoría de las cosas, tienen una actividad o interés de su usario específico. Vagamente podemos mencionar algunos contextos o ambientes como la salud, la educación o la economía, sin embargo, cada uno de ellos se desglosa cada vez más en pequeños ambientes, abarcando cada vez, una porción más pequeña del contexto inicial, una porción más específica. Estos contextos o ambientes son conocidos como dominio. Algunos dominios involucran el mundo físico (lo tangible), mientras que otros son intangibles (como un programa de contabilidad donde su dominio es el dinero y las finanzas).

El diseño dirigido por el dominio más allá de ser una tecnología o una metodología proporciona prácticas y terminología para tomar decisiones de diseño, las cuales se centran y aceleran proyectos de software de dominios complejos.

Es una técnica de modelado que da paso a una descomposición organizada de dominios de problemas complejos. Es un enfoque para desarrollar software para necesidades complejas, mediante la conexión profunda de la implementación con un modelo en constante evolución de los conceptos fundamentales del negocio.

Se basa tres premisas, la primera de ellas consiente en colocar el enfoque principal del proyecto en el dominio central y la lógica del dominio, en vez de colocarlo en las capas comúnmente usadas en el software. La segunda de ellas menciona que se debe basar los diseños complejos en un modelo. Finalmente, la tercera de ellas expresa la importancia de tener una colaboración creativa entre expertos técnicos y de dominio para acercarse cada vez más al núcleo conceptual del problema de manera iterativa.

Esta serie de prácticas y terminología tiene sus fundamentos en una serie de conceptos, cuya finalidad es comprender y modelar el dominio de forma efectiva. Entre ellos podemos encontrar:

1. Lenguaje ubicuo: En ocasiones, en un contexto podemos encontrar diversas palabras, que a simple vista, parecieran sinónimos o son palabras muy similares. Investigando más e identificando el comportamiento de las diferentes palabras, podemos encontrar el valor que realmente tiene cada una y a lo que esta se refiere. Por ejemplo, en ocasiones podemos encontrar que la palabra “banco” se refiere a un objeto donde nos podemos sentar, pero en otras ocasiones se refiere al lugar donde se maneja el dinero. La definición de cada una de estas palabras la da el dominio. De igual manera, en dado que el desarrollo de software maneja diversos equipos, cada uno puede tener una manera diferente de comprender los conceptos, es por esto que se necesita tener un lenguaje compartido entre el equipo de desarrollo y los expertos de dominio.
2. Contextos delimitados: Al igual que en las arquitecturas de microservicios, la granularidad es un aspecto importante al hablar de DDD, ya que es necesatio que el dominio se divida en diferentes contextos delimitados, donde cada contexto represente una parte coherente y acotada del negocio.
3. Modelado del dominio: El modelo de dominio es más que un simple diagrama, en este se expresa la idea que el diagrama está tratando de cubrir. Va más allá del conocimiento de experto en el dominio, es una rigurosa y selectiva abstracción de conocimiento. Es la base de todo el sistema, una completa representación de la realidad para un propósito en específico.

Este enfoque es utilizado en conjunto con diversos patrones arquitectónicos para tener una estructura ideal para un sistema de software. El uso de DDD tiene ciertos beneficios, de los cuales se pueden destacar:

* Comprensión profunda del dominio: Al invertir tiempo en el entendimiento del dominio, el equipo de desarrollo adquiere un conocimiento sólido de los procesos y reglas del negocio. Esto facilita la toma de decisiones y la resolución de problemas complejos.
* Flexibilidad y mantenibilidad: Al construir un modelo de dominio bien estructurado, se logra una arquitectura flexible y modular. Los cambios y evoluciones en el negocio se pueden manejar de manera más eficiente, minimizando el impacto en otras partes del sistema.
* Comunicación efectiva: El uso de un lenguaje ubicuo y de un modelo de dominio compartido mejora la comunicación entre los miembros del equipo de desarrollo y los expertos del dominio. Esto reduce la brecha entre ambos grupos y evita malentendidos.
* Separación en la lógica de negocio: Al separar la implementación técnica de los detalles del dominio, el Diseño Dirigido por el Dominio permite que el equipo de desarrollo se concentre en la lógica de negocio. Esto promueve la construcción de software más orientado a los requerimientos del negocio y menos a las preocupaciones técnicas.

En conclusión, se puede decir que el Diseño Dirigido por el Dominio es un enfoque valioso para el desarrollo de software. Al comprender profundamente el dominio del problema y construir un modelo de dominio expresivo, los equipos de desarrollo pueden crear aplicaciones más adaptables, mantenibles y centradas en la lógica de negocio. Si bien la implementación del DDD puede requerir un esfuerzo y costo adicional, sus beneficios a largo plazo compensan ampliamente esta inversión. En un panorama cada vez más complejo y cambiante del desarrollo de software, el Diseño Dirigido por el Dominio se posiciona como una metodología relevante y efectiva para construir software de calidad que satisfaga las necesidades del negocio.

Referencias:

*What is Domain-Driven Design?* (2015, 25 enero). dddcommunity.org. <https://www.dddcommunity.org/learning-ddd/what_is_ddd/>

Evans, E., & Evans, E. J. (2004). *Domain-driven Design: Tackling Complexity in the Heart of Software*. Addison-Wesley Professional.