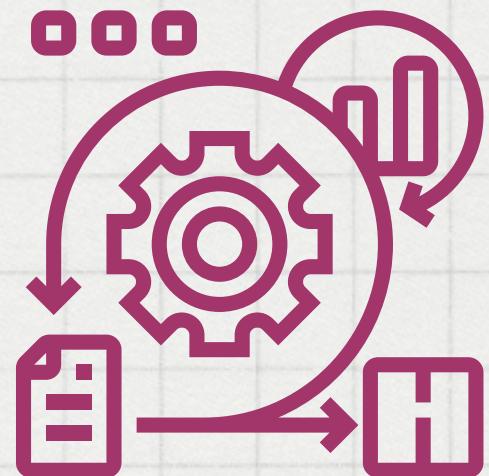


Investigación sobre Feature-Driven Development (FDD)

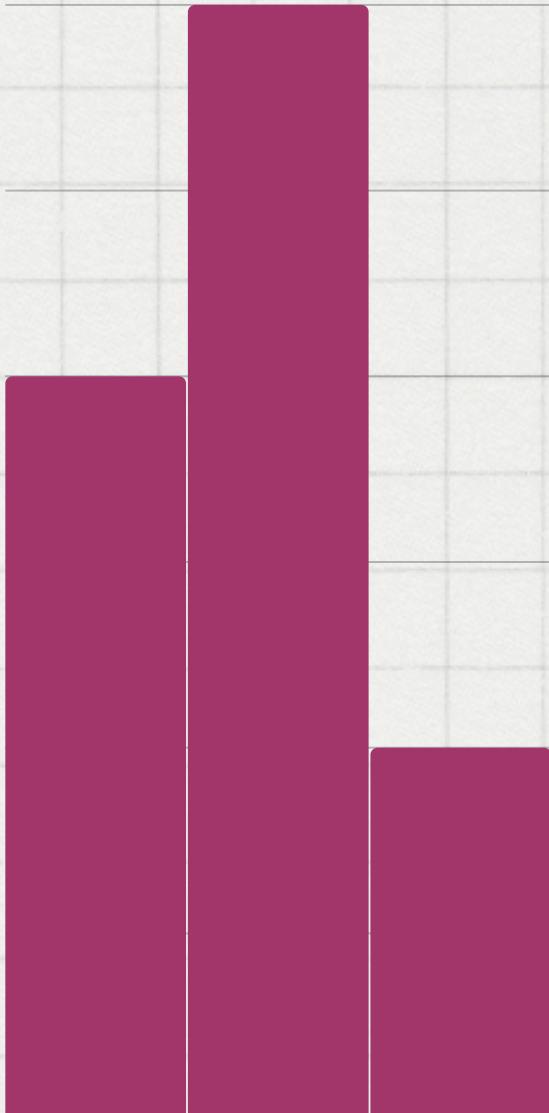
Introducción



El Feature-Driven Development (FDD) es una metodología de desarrollo de software que se enfoca en la creación de funcionalidades específicas y valiosas para el cliente. Se originó a finales de la década de 1990 como parte de un proyecto en una gran corporación financiera en Singapur, y fue formalizado por Jeff De Luca junto con Peter Coad. FDD es una de las primeras metodologías ágiles, aunque a menudo se discute menos que otras como Scrum o Extreme Programming (XP).

Origen y evolución

FDD fue desarrollado para abordar las limitaciones de las metodologías de desarrollo de software más tradicionales, como el modelo en cascada, que a menudo resultaba ineficiente en proyectos grandes y complejos. Jeff De Luca, quien lideró su desarrollo, propuso una metodología que combinaba las ideas de planificación por características (features) y modelado por objetos (objetos empresariales).



Principios fundamentales de FDD

FDD se basa en una serie de principios clave que guían su enfoque:

1. Desarrollo guiado por funcionalidades: En lugar de centrarse en tareas o fases del proyecto, FDD organiza el trabajo en torno a las características que aportan valor al cliente.
2. Modelado en función del dominio: Un componente esencial es el modelado inicial, que ayuda a definir la arquitectura del sistema y a identificar los objetos empresariales clave.
3. Iteraciones cortas: Las iteraciones son rápidas, generalmente de dos semanas, lo que permite una entrega frecuente de características completadas.
4. Colaboración del equipo: FDD promueve la colaboración entre todos los miembros del equipo, incluidos los desarrolladores, diseñadores, testers y clientes.
5. Calidad y seguimiento constantes: El progreso y la calidad del código son monitoreados de manera constante mediante métricas y revisiones.

Proceso de FDD

El proceso de FDD se divide en cinco actividades principales:

1. Desarrollo de un modelo general: Se construye un modelo del dominio en colaboración con los expertos en el dominio y los desarrolladores. Este modelo proporciona una visión general de lo que el sistema debe hacer.
2. Construcción de una lista de características: A partir del modelo general, se identifican y priorizan las características que el sistema debe tener. Cada característica es una pequeña funcionalidad que aporta valor al usuario final.
3. Planificación por características: Las características identificadas se organizan en un plan de desarrollo que describe cuándo y cómo se implementarán.
4. Diseño por características: Para cada característica, se diseña una solución detallada. Este diseño incluye la creación de diagramas, definición de clases y especificación de la interacción entre componentes.
5. Construcción por características: Finalmente, cada característica se desarrolla y prueba individualmente antes de ser integrada en el sistema completo.

Roles en FDD

FDD define roles específicos para asegurar que todos los aspectos del desarrollo sean gestionados eficazmente:

1. Chief Architect: Define la arquitectura general del sistema.
2. Development Manager: Coordina y supervisa el progreso del equipo de desarrollo.
3. Chief Programmer: Responsable de la implementación técnica de las características.
4. Class Owner: Desarrollador responsable de una clase específica o grupo de clases.
5. Domain Expert: Proporciona el conocimiento del negocio y del dominio en el que se va a implementar el sistema.

Comparación con otras metodologías ágiles

FDD comparte varios principios con otras metodologías ágiles, como la iteración rápida y el enfoque en la colaboración. Sin embargo, se diferencia en su enfoque más estructurado y dirigido. A diferencia de Scrum, por ejemplo, FDD tiene un mayor enfoque en la arquitectura y en la identificación temprana de las clases y objetos clave. Además, FDD se centra más en las funcionalidades concretas que en los entregables de alto nivel.

Ventajas y desventajas de FDD

VENTAJAS

1. Orientación al cliente: FDD asegura que cada iteración produzca funcionalidades que tienen valor directo para el cliente.
2. Gestión de proyectos complejos: Es particularmente útil en proyectos grandes y complejos debido a su enfoque estructurado.
3. Mejora continua: La entrega frecuente y la revisión constante permiten ajustes y mejoras continuas.

DESVENTAJAS

1. Curva de aprendizaje: FDD puede ser difícil de implementar para equipos nuevos debido a su énfasis en la planificación detallada y el modelado.
2. Foco en la arquitectura: Algunos equipos pueden considerar que el enfoque en la arquitectura es demasiado rígido o consume demasiado tiempo.
3. Adaptabilidad limitada: Comparado con metodologías como Scrum, FDD puede ser menos adaptable a cambios drásticos en los requisitos.

Casos de uso

FDD ha sido implementado con éxito en diversas industrias, particularmente en aquellos sectores donde los proyectos son grandes y complejos, como en la banca y las telecomunicaciones. Su enfoque en características individuales facilita la gestión de grandes equipos y la entrega continua de valor.

Conclusiones

El Feature-Driven Development es una metodología robusta y estructurada que ofrece un enfoque centrado en el cliente y orientado a la entrega de valor. Aunque puede no ser tan flexible como otras metodologías ágiles, es particularmente valiosa en proyectos de gran escala donde la planificación detallada y la gestión efectiva son críticas. FDD sigue siendo una opción viable para equipos que buscan una alternativa ágil con un enfoque más definido y controlado.



Cuestionario



1. ¿Cuál es el objetivo principal de la metodología Feature-Driven Development (FDD)
2. ¿Cuáles son las cinco actividades principales en el proceso de FDD?
3. ¿Qué rol en FDD es responsable de definir la arquitectura general del sistema?
4. ¿En qué se diferencia FDD de otras metodologías ágiles como Scrum?
5. ¿Cuáles son algunas de las ventajas de utilizar FDD en proyectos grandes y complejos?



iMuchas
thankius !

