Metodología Ágil:
Feature Driven Development

Miguel Albertí Pons
Sofía Almeida Bruno
Pedro Manuel Flores Crespo
María Victoria Granados Pozo
Lidia Martín Chica

Índice

1. Introducción

Feature-Driven Development (FDD), es una metodología que ayuda a crear software mediante un ciclo de vida iterativo incremental que esta orientado a equipos más grandes, con más personas que aquellos donde normalmente se utilizan otras metodologias como SCRUM. FDD ve necesaria la figura del jefe de proyecto y una fase donde se define la arquitectura.
En este documento veramos que procesos hay que seguir para aplicar FDD, sus ventajas y los roles con sus respectivas responsabilidades.

2. Procesos

La metodología FDD consta de cinco fases en su proceso. Son las siguientes:

2.1. Desarrollar el modelo global (*Develop overall model*)

En esta primera etapa los miembros del equipo de desarrollo trabajan juntos para construir un modelo del problema del dominio. Su objetivo es proponer el modelo para el área de dominio. El modelado FDD es una actividad interfuncional, iterativa y colaborativa. Los miembros del equipo (desarrollo, expertos en dominios y programadores principales) trabajan juntos para componer un modelo para el área de dominio y son guiados por un Arquitecto Jefe (*Chief Arquitect*). La idea es tener diferentes equipos proponiendo diferentes modelos y luego, después de ser revisados, seleccionar uno de los modelos propuestos o una combinación de modelos y que se convierta en el modelo para esa área de dominio. A medida que se desarrolle el modelo y el equipo aprenda, se agregarán detalles. Esto ayuda al equipo a tener una visión general clara de todo el proyecto, así como una comunicación sólida. Las tareas llevadas a cabo en esta etapa se resumen en la Tabla 1.

Tarea	Responsables
Formación del *Model Team*	*Project Manager*
Domain Walkthrough	*Modeling Team*
Estudio de los documentos	*Modeling Team*
Construcción de una *Feature List informal*	*Chief Arquitect*, *Chief* y *Programmers*
Desarrollo del *Modelo de Tareas*	*Modeling Team* dividido en pequeños grupos
Desarrollo del *Modelo Global*	*Chief Arquitect* y *Modeling Team*
Registro de alternaticas	*Chief Arquitect*, *Chief* y *Programmers*

Tabla 1: Tareas etapa del desarrollo del modelo global.

2.2. Construir lista de características (*Build Feature List*)

Esta etapa recibe como entrada el modelo de objetos y los requerimientos (*feature list informal*) obtenidos en la etapa anterior. Estos son agrupados según el área de dominio. Cada grupo de se denomina *Major List Sets*. Esta lista a su vez es dividida en otros subconjuntos según la funcionalidad. Posteriormente, cada característica o funcionalidad es priorizada y por último aquellas más complejas son divididas en otras más pequeñas. Esta etapa nos aporta como salida la *Feature List* que es revisada por los usuarios para su aprobación y validación. Las tareas llevadas a cabo en esta etapa se resumen en la Tabla 2.

Tarea	Responsables
Formación del *Feature-List Team*	*Project Manager* y *Development Manager*
Identificación de funcionalidades y formación del *Feature Set*	*Feature-List Team*
Priorización de las funcionalidades	*Feature-List Team*
División de las funcionalidades complejas	*Feature-List Team*

Tabla 2: Tareas etapa de construcción de la lista de características.

2.3. Planificado por características (*Plan by feature*)

El resultado final, de este paso, es un plan de desarrollo, sujeto a revisión y aprobación por parte del gestor de desarrollo y del jefe de la arquitectura, la identificación de la clase y la identificación de los propietarios sobre el conjunto de características. Este paso gira entorno a la planificación: el orden en el que se van a implementar las características, la forma de hacerlo y quién las va implementar, para ello se asignarán a los desarrolladores. Obviamente mientras planificamos consideramos diferentes aspectos como riesgos, complejidad, dependiencias, la carga de trabajo, etc, para evitar que surjan problemas complejos.

En esta etapa se incluye la creación de un plan de alto nivel, en el cual la *features list* es ordenada en base a la prioridad y a la dependencia entre cada *feature*. Además, las clases identificadas en la primera etapa son asignadas a cada programador.

En base a las características de la lista de la anterior etapa, el gestor del proyecto, el desarrollador y el programador jefe establecen hitos y diseñan un cronograma de diseño y construcción.

Finalmente se realiza una verificación del plan teniendo en cuentra la opinión de todos lo miembros del equipo. La salida es un plan de desarrollo incluyendo la fecha de finalización, para cada conjunto de características y cada característica indicar el programador jefe asignado, y las fechas de inicio y fin. Por último para cada clase indicar el correspondiente propietario de la clase.

2.3.1. Diseñando por características (*Design by feature*)

Esta etapa y la siguiente se realizan de forma iterativa, cada iteración debería durar entre unos pocos días y un máximo de dos semanas. Puede haber varios equipos de características trabajando simultáneamente en el diseño e implementación de su propio conjunto de características.

En esta etapa el *chief Programmer*, aprovechando el conocimiento obtenido en las primeras etapas del proceso, selecciona las siguientes características a realizar entre las que tenga en su lista de características asignadas. A continuación, organiza los equipos de características identificando los *class owners* (desarrolladores) que estarán involucrados en el desarrollo de las características seleccionadas y contacta con los expertos de dominio si es necesario. Parte del grupo puede trabajar en el diseño técnico mientras otros trabajan en la infraestructura, escribiendo clases, determinando los métodos y realizando los diagramas de secuencia correspondientes que el *chief Programmer* añadirá al modelo global. Asimismo, cada *class owner* actualizará la descripción de sus clases en base al diagrama de secuencia (añadiendo parámetros, tipos de retorno, mensajes enviados, etc).

Antes de pasar a la siguiente fase todo el equipo revisará y verificará el diseño.

Como salida de esta etapa se debe obtener: el diagrama de secuencia detallado, diagrama de clases actualizado, descripción de clases y métodos, notas del equipo significativas para el diseño.

2.3.2. (*Build by feature*)

3. Comparación con otros métodos

FDD es una mexcla en tre eXtremeProgramming y Scrum, añadiendo técnicas de Domain Driven Design

FDD vs XP

Feature Driven Development da una forma de contrlar la naturaleza iterativa e incremental de los proyectos ágiles, pero sin decir nada sobre como implementar esas características.

Estas características se pueden implementar de diversas formas con distintas técnicas. En un modelo de desarrollo donde se usa FDD para los detalles de la iteración y en donde las características se tratan como tareas a realizar. Usamos métodos ágiles para modelar las tareas que se van a realizar, y usamos XP para implementar los requisitos. Además notemos que en el análisis explícito tiene lugar el diseño o el trabajo de modelado para determinar las características que habría que implementar o dividir en varias tareas. XP utiliza el trabajo en parejas y FDD no, pero se puede incluir. FDD tiene como objetivo controlar proyectos adaptativos, ágiles e iterativos que pueden o no estar basados en XP.

Comunicación

Los métodos ágiles se centran bastante en la comunicación entre los mienbros del equipo y el resto de personas interesadas en el proyecto.

En XP y Scrum la documentación también es importante,

Centro los usuarios

Duración del sprint

Reuniones

Tamaño del proyecto

FDD vs Scrum

Puntos en común: Mejorar la comunicación, enfatizar las cualidades de los componentes del grupo, aumentar la colaboración.

Por otro lado, FDD se centra en las prácticas de ingenierías específicas Scrum no especifica ninguna práctica de ingeniería en particular. Además, mientras que FDD tiene bucles de retroalimentación más largos, Scrum tiene ciclos más cortos.

4. Ventajas

FDD minimiza la complejidad del sistema por lo que es una excelente solución para proyectos grandes y complejos, dando al equipo la capacidad de dividir el problema en

problemas más pequeños que se pueden resolver en menos tiempo.

La comprensión más profunda del sistema de software reduce el riesgo de encontrar problemas durante el desarrollo.

La mayoría de los procesos requieren un flujo de comunicación constante y la realización frecuente de los informes de progreso en casi cualquier nivel del desarrollo del proyecto, permiten al equipo realizar un seguimiento del progreso y los resultados. Además, FDD permite que los miembros del equipo se comuniquen más fácilmente al tiempo que fomenta la creatividad e innovación del equipo.

El hecho de que en FDD se realicen compilaciones regulares garantiza que se puedan identificar fácilmente errores en el código y permite mantener el proyecto actualizado regularmente, identificar cualquier error y que se pueda mostrar al cliente en cualquier momento.

En FDD se maximiza la calidad ya que el concepto de calidad no solo incluye la prueba del código, sino que también incluye estándares de codificación, auditorías de medición y métricas en el código.

5. Roles y responsabilidades

(Dejo esto por aquí porque no se nos escape...)

There are six primary roles on an FDD project: Project Manager, Chief Architect, Development Manager, Chief Programmer, Class Owner, and Domain Expert. An individual will take on one or more roles on a project as you would expect.

FDD also defines a collection of supporting roles, including:

Domain Manager Release Manager Language Guru Build Engineer Toolsmith System Administrator Tester Deployer Technical Writer

6. Bibliografía

[type=inbook,heading=subbibliography,title=Libros]