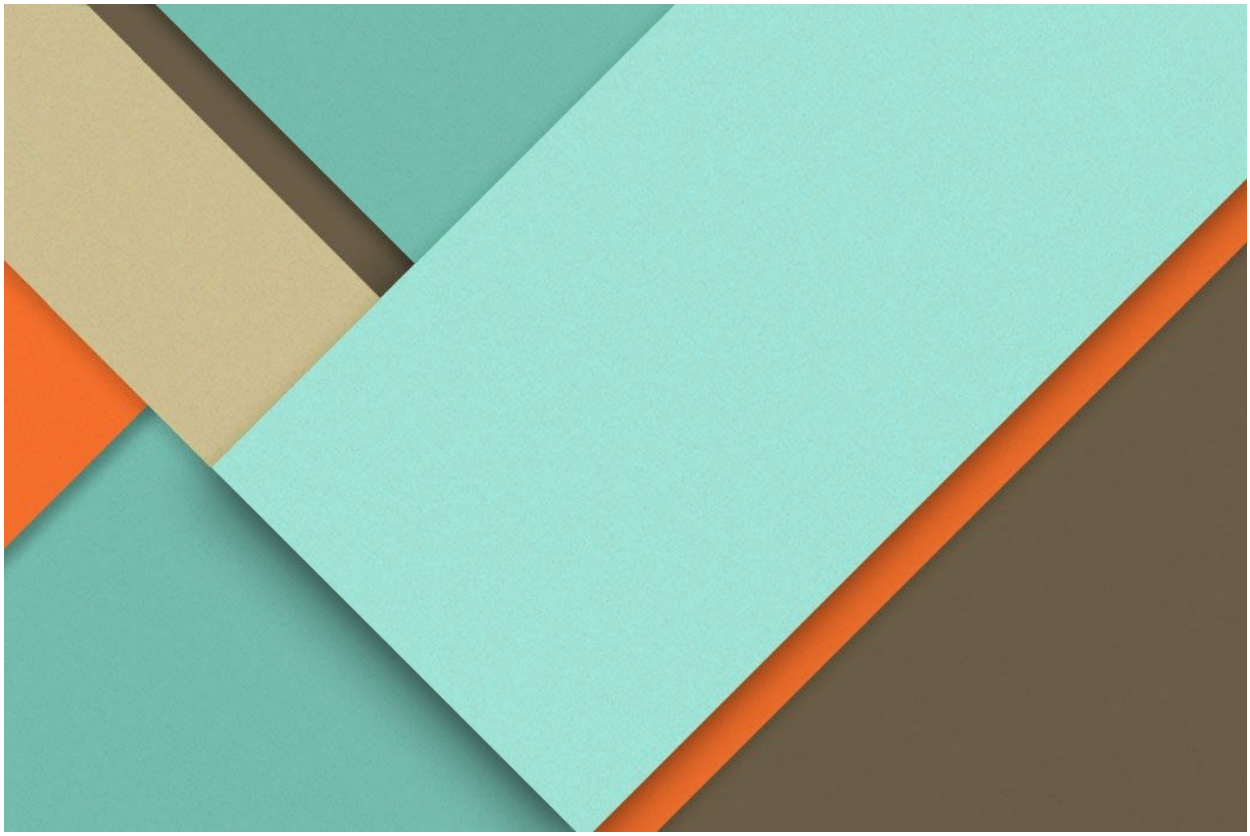

Feature Driven Development

08/01/2021

Alejandro Manuel Hernández Recio

Juan Pablo García Sánchez

Daniel Matilla Bastero

José María Gómez García

Mónica Cordovilla Gámez

Resumen

Feature-Driven Development (FDD) es una metodología ágil centrada en el cliente de una forma iterativa e incremental, con el único objetivo de desarrollar un software usable y eficiente (Lynn, n.d).

Qué es Feature Driven Development

Feature Driven Development (FDD) es una metodología ágil iterativa que presta especial cuidado a la funcionalidad. Fue usado por primera vez en 1997 por Jeff De Luca y Peter Coad durante 15 meses por 50 personas para un proyecto para un banco de Singapur. Tras el éxito de este, volvieron a poner a prueba esta metodología en 18 meses y 250 personas (W. Ambler, n.d.). Para que su equipo de desarrollo se adaptara mejor a las necesidades del cliente diseñó este modelo de desarrollo basado en los siguientes 5 pasos (Lucidchart Content Team, n.d.):

1. Desarrollar un modelo superficial
2. Crear lista de funcionalidades
3. Planificar por funcionalidad
4. Diseñar por funcionalidad
5. Construir por funcionalidad

Está dirigido a proyectos grandes con equipos numerosos (Garzás, 2012). Se desarrolla es iteraciones de hasta 2 semanas en las que entregan un producto mínimo viable y se centra en el seguimiento continuo del proyecto y en seguir las “buenas prácticas” del software (Valesca, 2012).

Al tener un seguimiento continuo pueden corregirse errores rápidamente y el cliente puede recibir información del estado del proyecto en cualquier punto del proyecto. Muchos desarrolladores tienen preferencia por esta metodología gracias a esto, ya que evita que el desarrollo se convierta en algo caótico y tener que rehacer grandes partes ya desarrolladas (Lynn, n.d.).

Fundamentos

Se debe tener en cuenta una serie de principios o fundamentos para implementar correctamente Feature Driven Development, y son los siguientes:

1. **Modelado de objetos del dominio**: Crear diagramas de clases para describir objetos en un dominio y las relaciones entre estos.
2. **Desarrollo por función**: Las funciones deben dividirse en otras más pequeñas y manejables, implementables en un periodo de dos semanas.
3. **Equipo de características**: Cada persona es responsable del rendimiento y calidad de cada clase, pero una característica puede involucrar más de una clase, por lo que todos los miembros del equipo de características contribuyen a las decisiones de diseño e implementación.
4. **Inspecciones**: Los equipos de FDD realizan inspecciones para detectar defectos y garantizar la mejor calidad del producto.
5. **Programa de compilación regular**: Con esta práctica se asegura que el equipo siempre tenga un sistema actualizado que pueda enseñarse al cliente
6. **Informes de progreso**: los gerentes del proyecto deben proporcionar estos informes de manera frecuente con el trabajo completado.

(Lucidchart Content Team, n.d.) (Valesca, 2012)

Pasos de diseño

Esta metodología se basa en los pasos de diseño ilustrados en la figura 1 y descritos más ampliamente en este apartado.

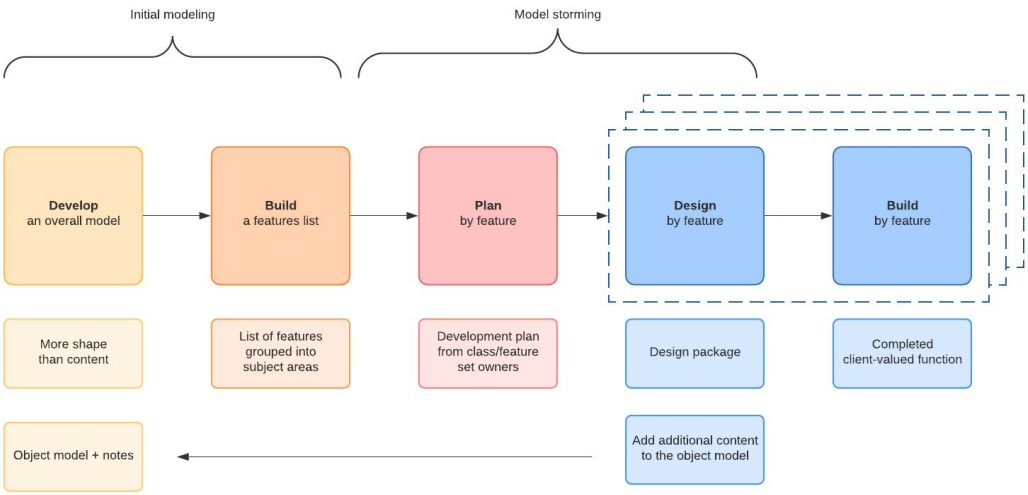

Fig. 1: Descripción general de los pasos de diseño a seguir. (Lucidchart Content Team, n.d.)

Desarrollar un modelo superficial

Se define el esquema que abarcará el dominio del modelo. Es decir, el problema que se pretende solucionar con el desarrollo del software. El equipo de desarrollo trabaja codo con codo con el jefe arquitecto para definir el alcance y contexto del sistema. Una vez que los equipos proponen sus propios modelos, uno de ellos o la unión de todos se elige y se convierte en el modelo creado para ese área de dominio. Como resultado, el equipo obtiene así una imagen clara del proyecto entero. (Lucidchart Content Team, n.d.), (Lynn, n.d)

Crear lista de funcionalidades

Tras desarrollar el modelo superficial, se identifican la lista de funcionalidades descomponiendo el dominio en pequeñas funciones. Si una función no puede ser desarrollada en un plazo de dos semanas, debería ser disgregada en funciones más pequeñas y manejables. (Lucidchart Content Team, n.d.), (Lynn, n.d),

Planificar por funcionalidad

Entre todos los miembros del equipo se analiza la complejidad de las características y se planean tareas para conseguir implementarlas. Teniendo en cuenta la complejidad de cada una, se asigna un orden en el que se van a realizar y también los miembros del equipo que deben trabajar en ellas.

En esta etapa también se identifican los dueños de las clases que son desarrolladores individuales a los que se les asigna una clase. Ellos son los responsables de todos los principios conceptuales de esa clase y en el caso de producirse cambios en varias clases se necesitará la colaboración de estos dueños de clases. (Lynn, n.d.)

Diseñar por funcionalidad

El desarrollador principal escogerá las funciones que se diseñan y construyen y les asigna una prioridad. Seguidamente, a cada equipo le adjudicará un rol temporal y quienes compondrán estos equipos (Lynn, n.d.) (Lucidchart Content Team, n.d.). Cada desarrollador también deberá prestar atención al desarrollo y mantenimiento de las funcionalidades que le han sido asignadas (Rychlý & Tichá, 2008, 198).

Entre las tareas que tienen que realizar estos equipos se encuentran revisión de diseño, codificación, pruebas unitarias, integración y revisión de código (Valesca, 2012).

Construir por funcionalidad

Todos los items necesarios para lograr el diseño de las características se implementan. Se diseña la IU y se hace y testea un prototipo de las características. Si éstas pasan los tests y se aprueban, se pueden pasar a los clientes. (Lucidchart Content Team, n.d.)

Cualquier característica que sobrepase las dos semanas de longitud se descompone en características más pequeñas que cumplan con la regla de las dos semanas. (Lynn, n.d)

Roles

El equipo que forma parte activa de la ejecución de la metodología FDD está jerarquizado en los siguientes roles:

- **Project manager**: Encargado de supervisar el desarrollo y transcurso del proyecto.
- **Chief architect**: diseña y modela el sistema.
- **Development manager**: Supervisa diariamente el trabajo del equipo de desarrollo y resuelve conflictos surgidos en el mismo.
- **Chief programmer**: Ayuda en el diseño. Analiza los requerimientos y selecciona las funcionalidades a desarrollar de la última fase del FDD. Lidera equipos de desarrollo reducidos.
- **Class owner**: Pertenece al equipo de desarrollo liderado por el Chief programmer. Diseña, codifica, prueba y documenta las funcionalidades a desarrollar. Participa en la decisión de qué clase será incluida en la lista de funcionalidades de la próxima iteración.
- **Domain expert**: Puede ser un usuario, un cliente, analista o una mezcla de estos. Entiende el problema que quiere resolver el Cliente. El equipo confía en el criterio del Domain expert para seleccionar qué valor es el más importante para el Cliente y dar prioridad.

(Lucidchart Content Team, n.d.) (Valesca, 2012)

Ventajas y desventajas frente a SCRUM

Ventajas

- Varios equipos de desarrollo trabajan a la vez, lo que al final reduce el tiempo
- La documentación está bien estructurada. Los desarrolladores se aseguran de ello.
- Funciona muy bien para equipos o proyectos grandes.

- Se tiene en cuenta la experiencia de cada desarrollador a la hora de asignar las tareas.

Desventajas

- Es difícil de usar en equipos o proyectos pequeños.
- El programador jefe cumple demasiados roles, por lo que casi todo depende de él.
- Al tener varios roles, aumentamos también la posibilidad de errores humanos.
- La comunicación es formal y documentada, mientras que en SCRUM se usa la comunicación verbal.

(Tirumala et al., 2016, #)

Conclusión

Feature Driven Development es una buena opción cuando se trata de un equipo lo bastante grande como para no poder ejecutar el método Scrum de una forma efectiva. Esta metodología ágil es adecuada para proyectos a largo plazo que cambian que agregan características continuamente en iteraciones regulares.

FDD funciona en equipos grandes y pequeños ya que está diseñado para enfocarse siempre en lo que el cliente necesita (Lucidchart Content Team, n.d.).

Bibliografía

Garzás, J. (2012, Septiembre 4). *Un resumen de la metodología ágil FDD*. javiergarzas.com. https://www.javiergarzas.com/2012/09/metodologia-gil-fdd-1.html

Globalluxsoft. (2017, Octubre 18). *5 Popular Software Development Models with their Pros and Cons*. medium. https://medium.com/globalluxsoft/5-popular-software-development-models-with-their-pros-and-cons-12a486b569dc

Lucidchart Content Team. (n.d.). *Why (and How) You Should Use Feature-Driven Development*. Lucidchart. https://www.lucidchart.com/blog/why-use-feature-driven-development

Lynn, R. (n.d.). *What is FDD in Agile?* planview. https://www.planview.com/resources/articles/fdd-agile/

Rychlý, M., & Tichá, P. (2008). A Tool for Supporting Feature-Driven Development (B. Meyer, J.R. Nawrocki, & B. Walter, Eds.). *Springer*, 196-207. https://link.springer.com/content/pdf/10.1007%2F978-3-540-85279-7_16.pdf

Tirumala, S. S., Ali, S., & Anjan Babu, G. (2016, Diciembre). A Hybrid Agile model using SCRUM and Feature Driven Development. *International Journal of Computer Applications*. https://www.researchgate.net/profile/Sreenivas_Sremath_Tirumala/publication/311775064_A_Hybrid_Agile_model_using_SCRUM_and_Feature_Driven_Development/links/588b1800a6fdcc225a341c72/A-Hybrid-Agile-model-using-SCRUM-and-Feature-Driven-Development.pdf

Valesca. (2012, Junio 12). *Metodología FDD*. Metodología FDD – Feature Driven Development / Desarrollo Basado en Funciones. http://metodologiafdd.blogspot.com/

W. Ambler, S. (n.d.). *Feature Driven Development (FDD) and Agile Modeling*. Agile Modeling (AM). http://agilemodeling.com/essays/fdd.htm