SISFIUX: adaptación de Feature-driven Development para el desarrollo de un sistema financiero para una universidad

César Ricardo Alducin Ruiz, Jorge Octavio Ocharán Hernández, Lizbeth A. Hernández González

Facultad de Estadística e Informática, Universidad Veracruzana
Av. Xalapa Esq. Manuel Ávila Camacho s/n CP. 91020

zs12015283@estudiantes.uv.mx, jocharan@uv.mx, lizhernandez@uv.mx

Resumen. El papel de las tecnologías de información en la economía actual ha crecido en importancia. Así, las organizaciones desarrolladoras de software contemporáneas se enfrentan a ambientes de negocio altamente cambiantes y a un incremento tanto en la complejidad de las tecnologías utilizadas para el desarrollo de software de calidad así como en la resolución de los puntos de vista de los diferentes interesados en el sistema. Por lo anterior, es deseable que los métodos de desarrollo de software sean adaptables para poder magnificar los beneficios para la organización a la que van dirigidas dichos sistemas. El presente trabajo describe la adopción y adaptación del enfoque de desarrollo basado en características tomando un sistema financiero como caso de estudio. La construcción de un modelo de dominio, la creación de la lista de características, la planeación del proyecto, así como el diseño y la construcción del sistema utilizando este método ágil son descritas y discutidas. Se concluye con una serie de lecciones aprendidas a partir de esta experiencia, las cuales buscan contribuir en la adopción exitosa de este tipo de métodos en proyectos futuros.

Palabras clave. Enfoque de desarrollo basado en características, Feature-Driven Development, FDD, desarrollo ágil de software.

1 Introducción

En desarrollo de software actual se encuentra ante diferentes retos. Los ambientes de negocio son altamente cambiantes, la complejidad del software y las tecnologías que se utilizan para desarrollarlo han aumentado tanto en escala como en alcance y se tienen considerar los puntos de vista y necesidades, a veces en conflicto, de un mayor número de interesados (*stakeholders*) en el software [1]. Por otro lado, las metodologías de desarrollo de software guiadas por planes (*plan-driven development methodologies*) dificultan la

respuesta a los cambios en el entorno de negocio pues estas metodologías requieren de un conjunto completo de requerimientos de software que se anticipa de manera temprana y se trata de reducir el costo al eliminar el cambio en lugar de administrarlo durante el desarrollo y beneficiarse de él [2]. Ante tal situación, 17 profesionales destacados de la industria del desarrollo de software se reunieron en 2001 para discutir la situación y publicaron lo que se conocería como el "Manifiesto por el desarrollo ágil de software" [3]. En el manifiesto se definieron los valores que sustentan a los entonces nacientes métodos ágiles de desarrollo de software. Jim Highsmith define a la agilidad como la habilidad de crear y responder al cambio para obtener beneficios en un ambiente de negocios turbulento [4], haciendo así énfasis en el alcance de los objetivos de negocio de una organización.

A la fecha, existen diferentes métodos de desarrollo de software que se consideran ágiles de acuerdo a la literatura, destacando entre ellos *eXtreme Programming* (XP) [5], *Scrum* [6], *Dynamic Systems Development Method* (DSDM) [7], [8], *Adaptive Software Development* (ASD) [9], *Lean Software Development* [10], *Crystal Methods* [11] y *Feature-driven Development* (FDD) [12], [13]. Los métodos ágiles de desarrollo de software han sido adoptados incluso en grandes empresas desarrolladoras de software como Google, IBM, Microsoft, Nokia o Yahoo! [14]–[17] y ha sido objeto de estudio de diferentes investigadores en los últimos años [18]. Para el caso de México, además de la evidencia anecdótica, existen pocos trabajos que den cuenta de la adopción de los métodos ágiles para el desarrollo de software por parte de la industria y la academia [19] [20] [21] y para el caso particular de FDD aún es menor [22].

En el caso de estudio que aquí se presenta, se adoptó FDD para el desarrollo de un sistema financiero para una universidad. Para este proyecto, fue necesario adaptar algunos de los procesos que proponen los autores de este método de desarrollo ágil debido a la misma naturaleza del proyecto. Además de la adaptación de los procesos de FDD, se presenta el proceso de planeación para el desarrollo del Sistema Financiero para la Universidad de Xalapa (SISFIUX), el cual permitió asignar a cada una de las etapas el tiempo necesario para desarrollar la arquitectura de la aplicación, la lista de requerimientos, el modelo del dominio y el diagrama de clases, entre otros artefactos necesarios.

2 Contexto

La Universidad de Xalapa (UX) es una institución de educación superior privada. A nivel administrativo, la UX se encuentra dividida en diferentes departamentos los cuales son: rectoría, finanzas, administración escolar, jefaturas de escuelas, ingresos y contabilidad. Dentro del departamento de ingresos, se realizan diferentes procesos y actividades tales como la administración de egresos e ingresos, las proyecciones financieras de ingresos y la generación de reportes. Cabe destacar que dichas actividades no se encuentra bien definidas y son semi-asistidas por computadora (mediante el uso de hojas de cálculo, principalmente) lo que origina que el personal invierta una cantidad de tiempo y esfuerzo considerable, presentándose además errores de doble captura haciendo que la información sea

inconsistente, no se tenga en el momento y formato adecuado y se dificulte la toma de decisiones por parte de los interesados (Rector, Vicerrector financiero y de planeación y Jefaturas de Ingresos, Contabilidad, Egresos y Presupuestos). Por lo anterior, el desarrollo de un sistema financiero que apoye en la realización de estos procesos y actividades de manera automatizada al generar las proyecciones financieras de la universidad era deseable. Los beneficios que se podrían obtener de este sistema son el manejo de egresos, la generación de reportes financieros y la proyección presupuestal de la universidad de una manera más eficiente lo que repercutirá en un mejor control del ingreso que se obtiene, el contar la información necesaria para la toma de decisiones de manera oportuna y en el formato adecuado.

3 *Enfoque de desarrollo basado en características*

El enfoque de desarrollo basado en características o FDD por sus siglas en Inglés, es un método ágil y adaptativo para el desarrollo de software creado por Peter Coad y Jeff De Luca a partir de su experiencia en el desarrollo de un sistema bancario en Singapur en el que el dominio del problema era muy complejo [12], [13] y posteriormente formalizado por Steve Palmer [13]. FDD es un método ágil para el desarrollo de software el cual propone una serie de procesos para el desarrollo en los que se enfatizan las iteraciones cortas, la calidad en cada uno de sus pasos, la entrega frecuente de resultados tangibles y un seguimiento preciso y significativo del proyecto.

FDD propone cinco procesos secuenciales con criterios de entrada, tareas y criterios de salida bien definidos los cuales guían a los interesados en el proyecto (*stakeholders*) desde el modelado hasta la liberación del sistema. FDD incluye seis roles principales y otros roles de soporte, artefactos, objetivos y prácticas que necesita el sistema para ser construido y liberado (modelado de objetos de dominio, desarrollo por características, propiedad individual del código por clases, equipos por características, inspecciones, construcciones continuas, administración de la configuración y visibilidad/reporte de resultados) [23].

3.1 Modelo de procesos del enfoque de desarrollo basado en características

FDD propone un modelo de procesos minimalista en el que únicamente en cinco procesos se diseña y construye el sistema. Los procesos son: Desarrollar un modelo general, Construir una lista de características, Planear por característica, Diseñar por características y Construir por característica [23]. Cada uno de los procesos son representados en una descripción de una página utilizando el patrón Criterio de entrada-Tareas-Verificación-Criterio de salida.

Proceso 1: desarrollar de un modelo general

Se desarrolla un modelo general del dominio de la aplicación, el cual es realizado por los expertos del dominio y el arquitecto en jefe. En él, se busca establecer el alcance del sistema y contextualizar los requerimientos del cliente. FDD se apoya para este proceso de los denominados recorridos por el dominio o "*Walkthrough*", el cual es descripción del dominio a alto nivel por parte de los expertos [23] y por el estudio de documentación relevante del dominio.

El resultado final que se obtiene de este proceso es un diagrama de clases con los tipos de objetos más significativos en el dominio del problema y las relaciones entre ellos complementado con un conjunto de diagramas de secuencia de alto nivel que muestran a los objetos interactuando unos con otros para cumplir sus responsabilidades [13]. La técnica que se sugiere para el desarrollo de este modelo general es "Modelando en color" [12] la cual usa cuatro arquetipos con diferentes colores para dividir las categorías de clases de la siguiente manera:

- Amarillo: Roles, usualmente una persona o una organización.
- Azul: Descripciones tipo catálogo.
- Verde: Una persona, lugar o cosa.
- Rosado: Un momento en el tiempo o un intervalo de tiempo asociado con un proceso del negocio.

Proceso 2: construir una lista de características

En este proceso se descompone la funcionalidad del dominio, a partir del modelo y diagramas creados en el Proceso 1, en un conjunto de características priorizadas. Una "característica" se define como una pieza de funcionalidad del sistema pequeña, entregable y de valor para el cliente que puede ser implementada en no más de dos semanas [13]. La división del dominio es desarrollada por el equipo de lista de características (expertos del dominio, programadores en jefe y arquitecto en jefe). Para la construcción de la lista de características se tiene que identificar las áreas del dominio, las actividades de negocio y los pasos de las actividades del negocio o características. Cada actividad del negocio puede estar constituida por un número variable de características. FDD recomienda un formato para la descripción de una característica el cual facilita su mapeo entre objetos y métodos [12].

Proceso 3: plan por características

La planeación de características incluye la creación de un plan de alto nivel en donde se asignan estas a los programadores en jefe, de acuerdo a su complejidad y dependencia entre ellas [23]. El equipo de planeación, el cual es integrado por el administrador del proyecto, el programador en jefe y los programadores, planean el orden en el cual se desarrollaran las características de acuerdo a dependencias, riegos, complejidad, balanceo de trabajo y prioridades del cliente. Posteriormente, las actividades son asignadas a equipos por características, liderados por un programador en jefe y cada clase en especifico es

asignada a un desarrollador el cual a partir de ese momento será el responsable de dicha clase durante todo el desarrollo (*class owner*) [4].

Proceso 4: diseño por característica

A partir de la planeación por características, el programador en jefe y su equipo de característica produce los diagramas de secuencia para las características asignadas. Posteriormente el programador en jefe refina el modelo de objetos (dominio) con base en el contenido de los diagramas de secuencia al mismo tiempo que el equipo por características escribe las clases y prólogos de los métodos. Finalmente, se realiza una verificación del diseño a través de una inspección del diseño por parte de equipo de característica y el resto del equipo de desarrollo [4]. Hay que destacar que tanto este proceso como el Proceso 5 son iterativos.

Proceso 5: construcción de características

El equipo de características toma el paquete de diseño creado en el proceso anterior y realiza las siguientes actividades [4]: implementación de las clases y métodos, inspección del código producido, realización de pruebas de unidad y promoción de las clases para la construcción de una versión del sistema (*software build*). Al igual que el proceso anterior, se trata de un proceso iterativo.

4 Caso de estudio: SISFIUX

Se desarrollaron cada una de los procesos propuestos por FDD en el proyecto SISFIUX. En las siguientes subsecciones se habla a detalle de cada uno de los procesos, las tareas realizadas, los artefactos construidos así como las adaptaciones que se hicieron a este método de desarrollo ágil de software.

4.1 Modelo del dominio general

De acuerdo con FDD el primer proceso está dedicado al modelo de dominio, por lo que fue necesario realizar una reunión con los expertos del dominio y el arquitecto en jefe. Los procesos que se llevan a cabo en el departamento de ingresos de la Universidad de Xalapa que necesitan ser automatizados y que se toman como base para la elaboración del modelo de dominio general son: Proyecciones financieras de ingresos, Reportes de ingresos y Reportes de Vales (ver figura 1).

Criterios de entrada

Los participantes para este proceso fueron son los siguientes:

- Expertos del dominio: Jefa del departamento de ingresos de la Universidad.
- Usuarios: Cajeras del departamentos de ingresos de la Universidad.

- Programador en jefe: Ingeniero en sistemas, encargado de adaptar FDD al desarrollo de SISFIUX.
- Dueño de las clases: Igual que el anterior.
- Desarrollador: Igual que el anterior.

Tareas

Formar el equipo de modelado: Para este caso de estudio se conformó de los expertos del dominio, programador jefe y desarrolladores y como un elemento externo el administrador del proyecto.

Recorrido del dominio: El experto del dominio explica el área en general que fue modelada al programador en jefe.

Estudiar los documentos: Los documentos que se estudiaron y que como se comento en la sección de contexto, son las hojas de calculo que ocupan los usuarios para realizar los reportes y proyecciones.

Desarrollar el modelo del dominio: El modelo inicial del dominio se desarrollo con la ayuda del experto del dominio y el arquitecto en jefe utilizando la técnica de modelado a color "UML-color". Esta técnica permitió identificar los elementos del modelo de una manera mas entendible para el experto del dominio ya que esta técnica permite identificar roles, descripciones, momentos e intervalos y lugares o cosas[12].

Refinar el modelo del dominio: Se refinó el modelo de dominio considerando los comentarios del experto del dominio.

Escribir las notas del modelo: Se escribieron las notas correspondientes para refinar el modelo de dominio.

Verificación: Se realizó de manera conjunta entre experto del dominio y el equipo de modelado para verificar el modelo de dominio construido.

Criterios de salida: El resultado de este proceso es el modelo de dominio en forma de: Diagrama de clases, Métodos y atributos identificados y Diagramas de secuencia.

4.2 Lista de características

Una vez que se ha construido el modelo de dominio general, la siguiente fase es la de construir una lista de características, considerando el modelo de dominio se identifica que el sistema debe contar con los módulos principales tales como: proyecciones financieras, reportes de ingresos y reporte de vales. Para ello se construye la lista de características considerando áreas, subáreas y características.

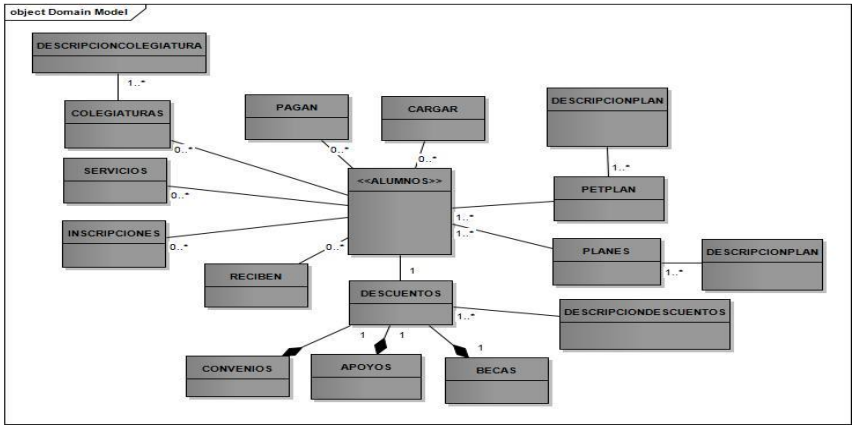

Fig. 1. Modelo del dominio parcial.

Criterios de entrada: Se seleccionaron los expertos del dominio, el programador en jefe y el arquitecto en jefe. En este caso fueron las mismas personas del Proceso 1.

Tareas:
Formar el equipo de características: el equipo de características se integro por el experto del dominio y el programador en jefe.
Construir la lista de características: se construye la lista de características bajo el siguiente formato establecido por parte del autor de FDD.

Ejemplo de una característica del sistema financiero:
"Calcular el total de pagos Semestre completo por modalidad"

Verificación: La verificación de la lista de características se llevó a cabo por parte del experto del dominio el cual también realizo una priorización de las características considerando aquellas que le son de mayor importancia:
Interna*:* verificación interna por parte del experto del dominio, quien apoyó al programador en jefe para refinar la lista de características y hacer los comentarios oportunos, esto permitió que la lista de características tenga validez, sea consistente, este completa, sea realista acorde con las necesidades del experto del dominio y los interesados y verificable.
Externa: verificación externa por parte del programador en jefe, para este caso de estudio ya no se tuvo que realizar una modificación al modelo de dominio.

Criterios de salida: el resultado del proceso es una lista de características por área, subáreas y actividades del negocio (ver figura 2).

ID	Características
1	Calcular el total de vales por subcuenta
2	Calcular el total de ingreso por folio de cobro
3	Calcular el total de vales por cuenta contable
4	Calcular el total de saldos de vales
5	Calcular el total de vales por unidad presupuestal
6	Calcular el total de vales pagados
7	Calcular el total de detalle de vales pagados
8	Calcular el total de vales de un usuario "X"
9	Calcular el total de abonos a los vales de un usuario "X"
10	Calcular el total de ingresos acumulados
11	Calcular el ingreso de las formas de pago por semana
12	Calcular el ingreso por forma de pago y por usuario
13	Calcular el flujo de efectivo diario
14	Calcular el total de la forma de pago efectivo
15	Calcular el total de la forma de pago tarjeta

Fig. 2. Lista parcial de características.

4.2.1 Plan por características

Una vez construida la lista de características se estableció un plan por características que consiste en calendarizar la forma en la que serán construidas estas. Como parte de las adaptaciones que se realizaron al proceso de FDD se adopto SCRUM como marco de trabajo para la administración del proyecto mas adelante se detalla como se llevo acabo esta adaptación.

Criterios de entrada: Lista de características construida en el proceso 2.

Tareas

Formar el equipo de planeación: Se integro por el programador en jefe y el administrador del proyecto.

***Determinar la secuencia de desarrollo*:** El equipo de planeación consideró los siguientes elementos para determinar la secuencia en que serán construidas las características.

- Priorización del experto del dominio
- Dependencia entre características y clases
- Balanceo de carga de trabajo de los dueños de las clases
- La complejidad de las características

Asignar un conjunto de características a los programadores en jefe*:* El equipo de planeación asigna a los programadores en jefe las actividades del negocio basándose en la secuencia de desarrollo, dependencia entre características y la complejidad de las mismas. Hay que considerar que los programadores en jefe también son dueños de las clases y que posteriormente estos asignaran a los programadores.

Asignación de las clases a los desarrolladores*:* El equipo de planeación asignó las clases que se identificaron para ello se realizó un formato que se adapto a la metodología y que se muestra parcialmente en la tabla 1.

Cada característica no debe de llevar más de 2 semanas realizarla si eso pasara entonces hay que dividir la característica en otras más pequeñas. FDD propone que se refine el modelo de dominio general aún y cuando ya se encuentre en la fase de planeación. Por lo cual si el programador en jefe decide regresar al proceso uno esta en libertad de hacerlo. En la tabla 2 se observar el plan por características.

Verificación: El equipo de planeación verifico que los formatos reportados no presenten inconsistencias y sea claros para los miembros del equipo que llevó a cabo el proceso FDD.

Criterios de salida: Los criterios de salida que se tuvieron después de realizar este proceso FDD al proyecto, fue los formatos que me permiten saber cuales son las actividades del negocio, a quien fueron asignadas las clases, así como la calendarización de las mismas. Como parte de las adaptaciones que se realizaron a esta metodología en este proceso en particular es el uso de Scrum considerando el *product backlog* y los *sprints*, quedando de la siguiente manera: en el *product backlog* se integro el conjunto de características divididas por *sprints* y cada sprint tuvo una duración de 3 semanas calendario.

Tabla 1. Extracto de asignación de clases a los desarrolladores

ID	Descripción	Programadores en Jefe	Dueños de las clases
1	Calcular el total de vales por subcuenta	I.S.C.César Ricardo	I.S.C.César Ricardo
2	Calcular el total de ingreso por folio de cobro	I.S.C.César Ricardo	I.S.C.César Ricardo
3	Calcular el total de vales por cuenta contable	I.S.C.César Ricardo	I.S.C.César Ricardo
4	Calcular el total de saldos de vales	I.S.C.César Ricardo	I.S.C.César Ricardo
5	Calcular el total de vales por unidad presupuestal	I.S.C.César Ricardo	I.S.C.César Ricardo
6	Calcular el total de vlaes pagados	I.S.C.César Ricardo	I.S.C.César Ricardo
7	Calcular el total de detalle de vales pagados	I.S.C.César Ricardo	I.S.C.César Ricardo
8	Calcular el total de vales de un usuario "X"	I.S.C.César Ricardo	I.S.C.César Ricardo
9	Calcular el total de abonos a los vales de un usuario "X"	I.S.C.César Ricardo	I.S.C.César Ricardo
10	Calcular el total de ingresos acumulados	I.S.C.César Ricardo	I.S.C.César Ricardo
11	Calcular el ingreso de las formas de pago por semana	I.S.C.César Ricardo	I.S.C.César Ricardo
12	Calcular el ingreso por forma de pago y por usuario	I.S.C.César Ricardo	I.S.C.César Ricardo
13	Calcular el flujo de efectivo diario	I.S.C.César Ricardo	I.S.C.César Ricardo
14	Calcular el total de la forma de pago efectivo	I.S.C.César Ricardo	I.S.C.César Ricardo
15	Calcular el total de la forma de pago tarjeta	I.S.C.César Ricardo	I.S.C.César Ricardo

Tabla2. Extracto del plan parcial por Características

ID	Descripción	Diseño		Inspección del Diseño	
		Plan	Actual	Plan	Actual
1	Calcular el total de vales por subcuenta	16/09/2013	01/03/2014	01/04/2014	15/04/2014
2	Calcular el total de ingreso por folio de cobro	16/09/2013	01/03/2014	01/04/2014	15/04/2014
3	Calcular el total de vales por cuenta contable	16/09/2013	01/03/2014	01/04/2014	15/04/2014
4	Calcular el total de saldos de vales	16/09/2013	01/03/2014	01/04/2014	15/04/2014
5	Calcular el total de vales por unidad presupuestal	16/09/2013	01/03/2014	01/04/2014	15/04/2014
6	Calcular el total de vales pagados	16/09/2013	01/03/2014	01/04/2014	15/04/2014
7	Calcular el total de detalle de vales pagados	16/09/2013	01/03/2014	01/04/2014	15/04/2014
8	Calcular el total de vales de un usuario "X"	16/09/2013	01/03/2014	01/04/2014	15/04/2014
9	Calcular el total de abonos a los vales de un usuario "X"	16/09/2013	01/03/2014	01/04/2014	15/04/2014
10	Calcular el total de ingresos acumulados	16/09/2013	01/03/2014	01/04/2014	15/04/2014
11	Calcular el ingreso de las formas de pago por semana	16/09/2013	01/03/2014	01/04/2014	15/04/2014
12	Calcular el ingreso por forma de pago y por usuario	16/09/2013	01/03/2014	01/04/2014	15/04/2014
13	Calcular el flujo de efectivo diario	16/09/2013	01/03/2014	01/04/2014	15/04/2014
14	Calcular el total de la forma de pago efectivo	16/09/2013	01/03/2014	01/04/2014	15/04/2014
15	Calcular el total de la forma de pago tarjeta	16/09/2013	01/03/2014	01/04/2014	15/04/2014

4.3 Diseño por características

Una vez construido el plan por características se realizo un diseño por características el cual consistió en que el programador en jefe seleccionara un conjunto de características para formar un paquete de trabajo, para ellos se crearon los paquetes de la siguiente manera: Proyecciones financieras, Reportes de egresos y Reporte de ingresos.

Criterios de entrada: El proceso de planeación se completo correctamente en el proceso anterior.

Tareas

Formar el equipo de características: El programador en jefe identifico el conjunto de características que deberán ser desarrolladas y que tienen en común clases para conformar un diseño de paquete.

Revisión del modelo de dominio: El experto del dominio revisó el modelo del domino para identificar las clases que serán construidas, esta tarea solo es para que el experto este enterado de cuales son las características que serán diseñadas.

Estudiar los documentos de referencia: **los** documentos que se estudiaron y que como se comentó en la sección de contexto, son las hojas de calculo que ocupan los usuarios para realizar los reportes y proyecciones.

Desarrollar los diagramas de secuencia: *se* desarrollan los diagramas de secuencia por diseño de paquete para la construcción de una o de un conjunto de características.

Refinar el modelo de dominio: no se tuvo que refinar el modelo del dominio, en este proceso cuatro.

Escribir los prólogos de las clases y métodos: los desarrolladores escribieron los prólogos de las clases y métodos utilizando una herramienta para la construcción de software (IDE Netbeans).

Verificación: La decisión de inspeccionar el diseño de las características es del programador en jefe, para este caso de estudio se realizo esta tarea para evitar incongruencias en la lista y planeación de características.

Criterios de salida: El resultado de este proceso es el diseño por paquetes de las características que se van a construir.

4.4 Construcción por características

Una vez diseñadas e identificadas las características en el modelo de dominio se comenzó con la programación de las clases que fueron asignadas. Los programadores desarrollan las características con las tecnologías que se decidieron implementar, el programador aquí encuentra una relación entre las clases y características, ya que una clase se conforma de una o más características.

Criterios de entrada: El diseño por características ha sido completado e inspeccionado por parte del programador en jefe.

Tareas
Implementación de clases y métodos: Se realizó la implementación utilizando tecnología Java, por ser orientado a objetos y el manejo de clases.
Inspección del código: Inspección de código por parte del programador en jefe.
Pruebas de unidad: Se realizaron las pruebas de unidad utilizando JUnit, por cada característica construida se realizó su respectiva prueba de unidad.
Promoción de la característica: Una vez que se construía una característica, se verificaba y se validaba con el cliente, la clase se promovía para formar parte de la siguiente versión del sistema.
Verificación
Pruebas de unidad: Se utilizo JUnit para esta tarea y un plan de pruebas de unidad.
Pruebas de integración: Se utilizo un plan de pruebas de integración.
Pruebas de aceptación de usuario: Se utilizo un plan de pruebas de aceptación.
Criterios de salida: El resultado de este proceso fue: Clases y métodos exitosos con código inspeccionado, características promovidas por el experto del dominio y Características evaluadas por el cliente y usuarios finales.

5 Lecciones aprendidas

El aplicar esta metodología ágil en un contexto en el cual no se ha aplicado algún proceso de la Ingeniería de Software de manera formal, implica establecer procesos para la definición de las actividades que se realizan. Guiar a un equipo de desarrollo para la construcción de un software a través de un proceso definido como FDD permite establecer una forma de trabajo en el cual los resultados se ven tangibles en un corto plazo. Seguir una metodología ágil implica generar los artefactos y actividades propuestas por esta de manera sistemática, además de lograr una implementación que apoye el desarrollo de un software que cumpla con los requerimientos del cliente y de los estándares definidos.

6 Conclusiones y trabajo futuro

En el presente artículo se describe la adaptación de FDD para el desarrollo del SISFIUIX. En la adaptación de esta metodología ágil, se tuvieron que realizar adaptaciones para lograr producir un software que cumpliera con los requerimientos del cliente. Para lograr la adaptación de FDD se tuvo que conformar un equipo multidisciplinario, lo que permitió contar con un conjunto de opiniones que permitieron un mejor desarrollo del software.

El programador en jefe, los dueños de las clases y los programadores tuvieron que estudiar los documentos necesarios para un mejor entendimiento del sistema a desarrollar, conocer los procesos y ayudar a sistematizar estos al personal del departamento de ingresos de la Universidad es una de las lecciones que deja la adaptación de FDD. Las adaptaciones a esta metodología para complementar cada una de las actividades que propone FDD, permitió generar nuevos artefactos complementarios a los FDD. El presente proyecto al ser de naturaleza transaccional se adaptó a la metodología porque precisamente el origen de FDD proviene de un proyecto financiero.

SISFIUX permite la elaboración de proyecciones financieras para conocer los ingresos que se percibirán por conceptos de pago de servicios en los distintos niveles educativos de la UX. Gracias a la implementación de FDD para el desarrollo de SISFIUX se logró poner énfasis en las etapas de diseño y construcción, ya que son las que principalmente promueve esta metodología ágil. SISFIUX permite además de la generación automática de los procesos antes mencionados, contar con un sistema que está desarrollado aplicando las técnicas y herramientas de la Ingeniería de Software.

Al ser un sistema financiero, las pruebas jugaron un papel muy importante ya que nos permiten saber que el software que se está desarrollando cumple tanto con los requerimientos del cliente. Es por ello que se realizaron pruebas de unidad en cada construcción de la lista de características utilizando pruebas de unidad, pruebas de integración y pruebas de aceptación de usuario.

Referencias

[1] S. Nerur, A. Cannon, V. Balijepally, and P. Bond, Towards an understanding of the conceptual underpinnings of agile development methodologies, in Agile Software Development, Springer, pp. 15–29, (2010).

[2] J. Highsmith, A. Cockburn, C. Consortium, and A. Cockburn, Agile software development: the business of innovation. Computer (Long. Beach. Calif)., vol. 34, no. 9, pp. 120–127, Sep.(2001).

[3] K. Beck, M. Beedle, A. van Bennekum, A. Cockburn, W. Cunningham, M. Fowler, J. Grenning, J. Highsmith, A. Hunt, R. Jeffries, J. Kern, B. Marick, R. C. Martin, S. Mallor, K. Shwaber, and J. Sutherland. The Agile Manifesto (2001).

[4] J. Highsmith, Agile software development ecosystems. Addison-Wesley Professional, 2002.

[5] K. Beck, Extreme Programming Explained: Embrace Change. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., (2000).

[6] K. Schwaber and M. Beedle, Agile software development with scrum. Pearson Education International, (2002).

[7] J. Stapleton, DSDM, Dynamic Systems Development Method: The Method in Practice. Addison-Wesley, (1997).

[8] J. Stapleton, DSDM: Business focused development. Pearson Education, (2003).

[9] J. A. Highsmith, Adaptive Software Development: A Collaborative Approach to Managing Complex Systems. Dorset House Pub., (2000).

[10] M. Poppendieck and T. Poppendieck, Lean Software Development: An Agile Toolkit. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., (2003).

[11] A. Cockburn, Crystal clear: a human-powered methodology for small teams. Pearson Education, (2004).

[12] B. P. Coad, E. Lefebvre, and J. De Luca, Java Modeling in Color with UML: Enterprise Components and Process. (1999).

[13] S. R. Palmer and M. Felsing, A practical guide to feature-driven development. Pearson Education, (2001).
[14] J. Sutherland and K. Schwaber, The scrum papers: nut, bolts, and origins of an Agile framework. 2001. [Online]. Available: http://jeffsutherland.com/ScrumPapers.pdf. [Accessed: 02-Jun-(2014)].
[15] E. Woodward, S. Surdek, and M. Ganis, A practical guide to distributed Scrum. Pearson Education, (2010).
[16] M.-W. Chung and B. Drummond. Agile at Yahoo! From the Trenches. In Agile Conference, 2009. AGILE '09., 2009, pp. 113–118, (2009).
[17] M. Laanti, O. Salo, and P. Abrahamsson, Agile Methods Rapidly Replacing Traditional Methods at Nokia: A Survey of Opinions on Agile Transform5ation. Inf. Softw. Technol., vol. 53, no. 3, pp. 276–290, (2011).
[18] T. Dingsøyr, S. Nerur, V. Balijepally, and N. B. Moe. A Decade of Agile Methodologies: Towards Explaining Agile Software Development. J. Syst. Softw., vol. 85, no. 6, pp. 1213–1221, Jun. (2012).
[19] J. A. Perez-Torres and M. Mejia. Software Development Using Agile Methodologies: An Airline Case. In Proceedings of the Sixth Mexican International Conference on Computer Science, pp. 129–135, (2005).
[20] C. Enríquez-Ramírez and P. Gómez-Gil. Análisis empírico sobre la adopción de las metodologías ágiles en los equipos de desarrollo de software en empresas mexicanas.In Tópicos selectos de tecnologías de la información y las comunicaciones. Memorias del Congreso Nacional y Congreso Internacional de Informática y Computación 2012, (2012).
[21] Aceves-Ortega H. M.: Agile Methods in Mexico. 2014. [Online]. Available: http://www.scrumalliance.org/community/articles/2014/february/success-story-agile-methods-in-mexico.
[22] L. C. A. Gutiérrez, E. S. C. Castro, and M. R. Hurtado. A Mexican Experience Redesigning a Software Development Process Using XP, FDD and RUP.
[23] P. Abrahamsson, O. Salo, J. Ronkainen, and J. Warsta, Agile software development methods: Review and analysis, (2002).