PRÁCTICAS ÁGILES PARA EL DESARROLLO DE SOFTWARE EN SEMILLEROS DE INVESTIGACIÓN

JULIANA TAMAYO CIFUENTES

UNIVERSIDAD PONTIFICA BOLIVARIANA

INGENIERÍA

FACULTAD DE INGENIERÍA EN TIC

INGENIERÍA DE SISTEMAS E INFORMÁTICA

MEDELLÍN

2013

PRÁCTICAS ÁGILES PARA EL DESARROLLO DE SOFTWARE EN SEMILLEROS DE INVESTIGACIÓN

JULIANA TAMAYO CIFUENTES

UNIVERSIDAD PONTIFICIA BOLIVARIANA

INGENIERÍA

FACULTAD DE INGENIERÍA EN TIC

INGENIERÍA DE SISTEMAS E INFORMÁTICA

MEDELLÍN

2013

PRÁCTICAS ÁGILES PARA EL DESARROLLO DE SOFTWARE EN SEMILLEROS DE INVESTIGACIÓN

JULIANA TAMAYO CIFUENTES

Trabajo de grado para optar al título de Ingeniera de Sistemas e Informática

Asesor

OSCAR EDUARDO SÁNCHEZ GARCÍA

Ingeniero de Sistemas y Computación

UNIVERSIDAD PONTIFICIA BOLIVARIANA

INGENIERÍA

FACULTAD DE INGENIERÍA EN TIC

INGENIERÍA DE SISTEMAS E INFORMÁTICA

MEDELLÍN

2013

Este trabajo está dedicado a mi padre, Héctor, quien con sus consejos me ha apoyado y animado en todo el proceso de mi formación profesional.

AGRADECIMIENTOS

Agradezco a mis padres por el apoyo constante que me han brindado durante toda mi vida, a mi novio por el impulso y la exigencia, a mis compañeros, a mis profesores y a la universidad, que me acogió por más de cinco años para poder adquirir el conocimiento necesario en mi formación profesional.

CONTENIDO

LISTA DE FIGURAS

Pág.

LISTA DE TABLAS

GLOSARIO

ANDROID: Es un sistema operativo open-source basado en Linux, desarrollado principalmente para dispositivos móviles.

APLICACIÓN MÓVIL: Programa o software instalable en dispositivos móviles por lo general a través de una tienda de aplicaciones.

BLACKBERRY OS: Sistema operativo móvil creado por BlackBerry para sus dispositivos.

DESARROLLO GUIADO POR PRUEBAS: Mejor conocido por sus iniciales en inglés como TDD (Test Driven Development), es una metodología utilizada para realizar las pruebas unitarias antes de comenzar a codificar.

GIDATI: Grupo de Investigación, Desarrollo y Aplicación en Telecomunicaciones e Informática.

GICU: Grupo de Investigación Comunicación Urbana.

GUI: Sigla de la frase en inglés Graphical User Interface, que traduce en español Interfaz gráfica de usuario, esta interfaz es una vista gráfica que permite al usuario interactuar con las diferentes funcionalidades del software.

INGENIERÍA DE SOFTWARE: Proceso por el cual un individuo o un grupo organiza y gestiona la creación de un sistema de software.

INTEGRATION HELL: Vocablo ingles que traduce integración del infierno, que explica el caos producido al integrar todos los cambios del código fuente al final del proyecto de software.

IOS: Sistema operativo móvil propiedad de la empresa Apple que es usado en dispositivos de la misma empresa como iPod, iPhone, iPad y Apple TV.

KAIZEN: Vocablo japonés que traduce mejoramiento continuo, es la base de la metodologías Lean y Kanban.

KANBAN: Herramienta de administración visual utilizada por metodologías como scrum y Lean.

MANIFIESTO ÁGIL: Conjunto de los principios de las metodologías ágiles una iniciativa entre los principales responsables de los procesos ágiles.

METODOLOGÍA DE DESARROLLO: Marco de trabajo usado para estructurar, planificar y desarrollar software.

MUDA: Vocablo japonés que traduce desperdicio.

MURA: Vocablo japonés que traduce irregularidad.

MURI: Vocablo japonés que traduce imposibilidad.

PHONEGAP: Es una plataforma de desarrollo de código libre, usada para la elaboración de aplicaciones para dispositivos móviles por medio de herramientas genéricas como JavaScript, HTML5 y CSS3.

ROI: Return On Investment, el retorno sobre la inversión es un cálculo que realizan las empresas para medir los beneficios que se obtienen de un proyecto.

RUP: Siglas de Rational Unified Process en español Proceso Racional Unificado, es un proceso iterativo e incremental que provee una guía para el desarrollo de un proyecto de software con metodologías adaptables al contexto y a la organización.

SCRUM: Es un marco de trabajo que provee estructura y reglas a equipos de trabajo que desarrollan un producto, dividiendo el trabajo en pequeñas entregas para que el equipo pueda detectar fallas, retroalimentar y realizar los cambios que sean necesarios para entregar un producto de alta calidad al cliente.

SDK: Software Development Kit**,** conjunto de herramientas de desarrollo de software.

SEMILLERO DE INVESTIGACIÓN: Agrupación voluntaria que permite a los estudiantes adquirir competencias investigativas a través de actividades particulares donde ellos mismos son los responsables de su propio aprendizaje.

TRELLO: Herramienta de apoyo en la administración de tareas mediante tableros tipo Kanban.

RESUMEN

Este proyecto propone un conjunto de prácticas ágiles que pueden ser aplicadas en los semilleros de investigación y en grupos universitarios de desarrollo de software, con el fin de que los estudiantes a lo largo de su proceso de aprendizaje adquieran la cultura de la estructuración, la planificación, la ejecución y el seguimiento de los proyectos de ingeniería de software.

Las técnicas descritas en este trabajo son tomadas de grandes exponentes de las metodologías ágiles, Scrum, XP, ASD y Lean; con el propósito de realizar una prueba piloto y validar la estrategia en los semilleros de investigación Internet of Things y Comunicación Digital de la Universidad Pontificia Bolivariana sede Medellín.

INTRODUCCIÓN

En el 2012 los estudiantes de comunicación social de la Universidad Pontificia Bolivariana realizaron una investigación en varias universidades de Colombia titulado *"Usos de los dispositivos móviles entre la comunidad universitaria",* con el propósito de identificar el grado de alfabetización digital y reconocer los usos de dispositivos móviles en el ámbito académico, además de indagar sobre las expectativas que estudiantes y docentes de las universidades hicieron sobre aplicaciones de uso en el ámbito educativo.

Con los resultados de esta investigación, los grupos GICU y GIDATI, en un marco de cooperación, iniciaron el desarrollo de un aplicativo de software para dispositivos móviles que concrete un uso académico de estos dispositivos. La propuesta consiste en integrar en un mismo aplicativo las herramientas necesarias para potenciar las relaciones de la comunidad universitaria y optimizar el acceso a la gran cantidad de información que se encuentra en repositorios de múltiples sistemas de información de la Universidad Pontificia Bolivariana.

Este proyecto de software, denominado UPB Móvil, se dividió en varias fases, la primera denominada UNotes, una agenda colaborativa para la gestión de actividades académicas. Esta agenda fue desarrollada por los integrantes del semillero de investigación Internet of Things del grupo de investigación GIDATI, conformado por estudiantes de diversos semestres de la facultad de Ingeniería en TIC, bajo la orientación del docente Oscar Eduardo Sánchez García adscrito a este grupo de investigación.

Es entonces, cuando se decide abordar una segunda fase del proyecto, la cual consiste en la construcción de una red social institucional que permitirá la creación de una comunidad académica virtual, donde los miembros de la universidad puedan ser parte de interacciones sociales y acceder a información sobre las áreas de su interés, específicamente en temáticas relacionadas con las facultades, dependencias administrativas, investigación y/o extensión.

Debido a la magnitud y el reto académico que implica este proyecto, los integrantes de los Semilleros de investigación mostraron una preocupación debido a que hasta el momento, los desarrollos habían sido prototipos creados sin una documentación formal, y peor aún sin la rigurosidad metodológica que un desarrollo de software requiere. Este ejercicio de reconocimiento, plantea la necesidad de aplicar modelos de ingeniería de software que permita gestionar procesos de desarrollo y lograr que este proyecto sea escalable y sostenible en el tiempo.

Se reconoce la necesidad de una propuesta metodológica que soporte este proyecto y los demás procesos de software que el semillero Internet of Things realiza, dado su énfasis en desarrollo de aplicaciones software para dispositivos móviles. Se inicia entonces, con el rastreo de las metodologías que se pueden aplicar a los proyectos de software para aplicaciones móviles, y en este sentido Anthony I. Wasserman [1]

manifiesta que las técnicas tradicionales de ingeniería de software pueden ser aplicadas al desarrollo de aplicaciones móviles, sin embargo, a su vez, menciona la pertinencia del uso de Scrum y de otras técnicas de las metodologías ágiles para este tipo de proyectos.

El poco reconocimiento que tienen los miembros del semillero sobre estas metodologías ágiles demanda la tarea de indagar un marco para las prácticas ágiles, las cuales fomentan una cultura de adaptabilidad al cambio de los requerimientos y de retroalimentación continua en el desarrollo de un proyecto de software, además, ponen a las personas por encima de los procesos, algo completamente diferente de las metodologías tradicionales; estos valores despertaron el interés de los estudiantes y se ve la oportunidad de realizar una indagación de las prácticas ágiles que se pueden aplicar en los semilleros de investigación.

Este documento está organizado en tres capítulos, en el primero se explica el origen y la importancia de las prácticas ágiles y la relevancia que estas le prestan al factor humano. En el segundo capítulo se propone una guía compuesta por algunas de las técnicas estudiadas en el primer capítulo. Finalmente, la aplicación de las técnicas seleccionadas, serán puestas a prueba por los integrantes del semillero de investigación Internet of Things de la facultad de Ingeniería en TIC y por los estudiantes que hacen parte de los semilleros de investigación del grupo GICU de la Facultad de Comunicación Social.

CAPÍTULO 1

Este primer capítulo identifica algunos de los procesos de desarrollo y las metodologías más representativas de la ingeniería de software; se estudian los componentes centrales de cada estrategia para extraer las prácticas ágiles que puedan ser aplicadas a proyectos desarrollados por miembros de los semilleros de investigación y/o grupos de desarrolladores en formación.

1. MODELOS DE PROCESOS DE SOFTWARE

Actualmente las empresas buscan mejorar sus procesos para lograr una mayor calidad en sus productos en el menor tiempo posible, disminuyendo los costos y minimizando errores en la manufactura de sus productos con el fin de tener satisfechos a los clientes. Las empresas de desarrollo de software no son la excepción, cada día se esfuerzan por ser más ágiles en todos sus procesos con el fin de obtener software de alta calidad que cumplan con las necesidades de los clientes.

Para que un producto de software, ya sea una aplicación web o aplicación móvil, conserve los criterios de calidad, además de satisfacer las expectativas del cliente, debe contar con las siguientes características [2]:

- Funcionalidad: El producto debe satisfacer los requerimientos del cliente.
- Confiabilidad: El sistema debe ser creado de manera que evite tener fallos, y en el momento que se presenten poder recuperarse.
- Usabilidad: Un sistema de software debe tener una interfaz de usuario adecuada para los clientes.
- Eficiencia: El producto debe ser capaz de utilizar todos sus recursos y no desperdiciarlos.

El problema de la mejora de los procesos en la ingeniería de software es que la base del desarrollo son los requerimientos especificados por el cliente al principio del proyecto, lo que hace que el resultado sea impredecible y que muchas veces funcionalmente no cumplen su cometido, se presentan años de retrasos en la entrega de los sistemas, los costos sobrepasan lo presupuestado y que los sistemas tengan un desempeño muy bajo.

Estos malos resultados no se deben a la inexperiencia o falta de conocimiento de los desarrolladores, es solo falta de técnicas o habilidades de ingeniería de software necesarias para un proyecto a gran escala.

La industria del software ha respondido a esto aplicando ingeniería en la creación del software, creando diferentes metodologías, modelos y prácticas, las cuales estructuran las actividades requeridas para desarrollar sistemas de software incrementando la calidad del producto.

En la década de 1970 Winston Royce creó el denominado modelo en cascada, dicho modelo sirvió como base para la formulación del análisis estructurado, el cual fue uno de los precursores en este camino hacia la aplicación de prácticas estandarizadas dentro de la ingeniería de software.

Con el tiempo, en el modelo en cascada se fueron evidenciando ciertas desventajas ya que los resultados del producto solo se podían ver al final del proceso, lo que exigía

que el usuario tuviera paciencia. Con esto se dio paso a la creación de otros modelos como lo son el modelo en espiral y el modelo evolutivo donde el objetivo es construir una versión inicial que se expone al cliente y se refina con sus comentarios y sugerencias.

Estos modelos dieron paso a la creación de diferentes metodologías denominadas metodologías tradicionales, de las cuales se han desplegado procesos como es el caso de Rational Unified Process (Proceso Racional Unificado) el cual incorpora las etapas del ciclo del modelo en espiral.

Mejor conocido como RUP, es uno de los procesos con más influencia en la industria del software desarrollado y sostenido por Rational® Software de IBM, como se explicó previamente el ciclo de vida RUP se basa en el modelo en espiral que organiza las tareas en fases e iteraciones [3].

RUP divide el proceso en cuatro fases, dentro de las cuales se realizan varias iteraciones en número variable según el proyecto y en las que se hace un mayor o menor hincapié en las distintas actividades. Se centra en la funcionalidad que el sistema debe poseer para satisfacer las necesidades de un usuario (persona, sistema externo, dispositivo) que interactúa con él y maneja los casos de uso como el hilo conductor que orienta las actividades de desarrollo. Mediante un proceso continuo de pruebas y retroalimentación, garantizando el cumplimiento de ciertos estándares de calidad

Otro de los procesos tradicionales es CMMI-DEV (Capability Maturity Model Integration Development) conocido en español como CMMI para Desarrollo; los principios del modelo de CMMI fueron creados por Walter Shewhart en la década de 1930 y en los años 80 estos principios fueron refinados por W. Edwards Deming, Phillip Crosby y Joseph Juran, pero fue en el año 1989 cuando Watts Humphrey comenzó a utilizar estos conceptos en IBM y en SEI (Software Engineering Institute) [4].

CMMI es un modelo de mejora de los procesos de desarrollo de software que consta de buenas prácticas que tratan las actividades de desarrollo aplicadas a productos y servicios, aborda las prácticas que cubren el ciclo de vida del producto desde la concepción hasta la entrega y el mantenimiento.

El SEI también es el encargado de PSP (Personal Software Process) un conjunto de métodos creados por Watts Humphrey en 1995 para el control y medida de la estimación de los proyectos de software [5]. El proceso PSP ayuda a los desarrolladores a medir y a analizar su desempeño.

En este tipo de metodologías se considera la importancia de la documentación del sistema, lo cual permite entender, extender y darle mantenimiento al software durante todo el tiempo de su vida útil [6]. Además, estas metodologías proporcionan un orden y una estructura bien definida para el desarrollo del software. Sin embargo, para que estas metodologías funcionen adecuadamente, se requiere de un alto grado de disciplina por parte de todos los miembros del equipo de desarrollo, lo cual es aún más importante en nuestra cultura latina.

A principio de los 90 comienza a aparecer un nuevo tipo de metodología, la metodología ágil, la cual presenta respuestas rápidas y efectivas al cambio; tienen un plan de proyecto flexible y muestran simplicidad en el desarrollo.

Las metodologías ágiles son procesos iterativos, es decir, se divide el trabajo en partes pequeñas progresivas que avanzan a través de la colaboración con el cliente, ya que por medio de reuniones continuas se verifica el proceso para ajustarlo de manera conveniente.

Estos procesos de mejora continua promueven la colaboración entre los miembros del equipo, incluido el cliente, quien pasa a ser un integrante más del equipo de desarrollo.

A continuación se presentaran las metodologías ágiles más destacadas de la ingeniería de software.

1.1. Programación extrema

Uno de los grandes representantes de las practicas ágiles es XP (eXtreme Programming), la programación extrema es una disciplina de desarrollo de software basada en los valores de la simplicidad, comunicación, retroalimentación, coraje y respeto [7] creada por Kent Beck en el año 1996 cuando fue convocado a trabajar en la aplicación de nóminas que estaba desarrollando la compañía Chrysler Corporation.

Es allí donde se da cuenta de las carencias de los métodos tradicionales, la importancia de incluir al cliente como otro miembro del equipo de desarrollo, y los beneficios de potenciar el trabajo en equipo.

En la programación extrema cada persona que esté involucrada en el proyecto es un miembro importante del equipo, hasta el cliente debe trabajar con el equipo durante el desarrollo [8].

Existen cuatro fases en el ciclo de vida de XP:

- Fase de planificación de la entrega: En esta fase los clientes crean la historias de usuarios, los desarrolladores toman estos requerimientos y los convierten en un grupo de iteraciones, es decir en un plan de entregas (releases).
- Fase de diseño: En esta fase de diseño se enfatiza en la filosofía de XP, la simplicidad en el diseño.
- Fase de codificación: En XP antes de comenzar con la codificación los desarrolladores deben realizar las pruebas unitarias, se crea el código fuente y se procede a la refactorización.
- Fase de pruebas: Con ayuda de las pruebas unitarias realizadas antes de la codificación se prueba constantemente y después se realiza las pruebas de aceptación del cliente.

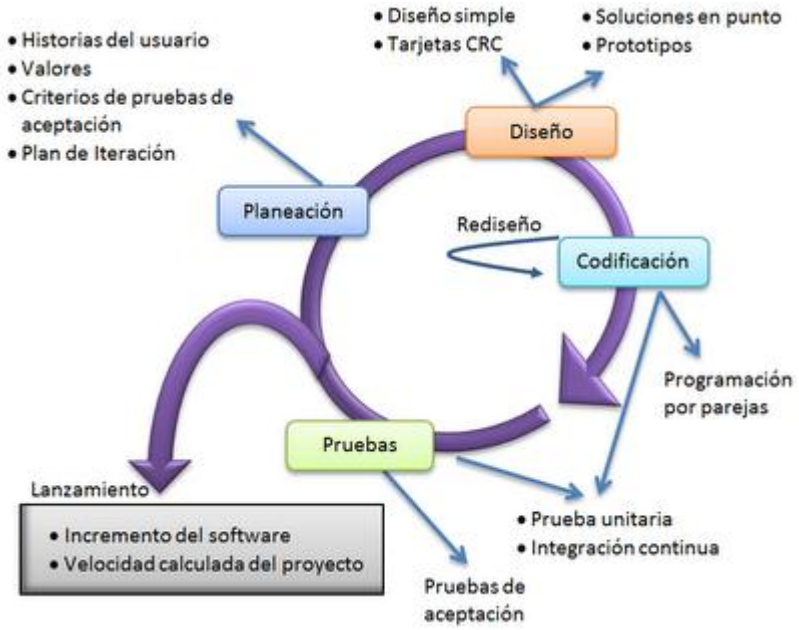

Fig. 1. Fases programación extrema. Tomado de [9]

Los principios básicos de la programación extrema son los siguientes:

- Juego de planeamiento: En esta fase el cliente es de gran importancia ya que es el encargado de definir los requerimientos del sistema por medio de documentos llamados historias de usuario, donde el cliente plasmará sus deseos y señalará las prioridades del sistema.

  Con las historias de usuario el equipo de desarrollo determina el alcance del sistema, identifica problemas, propone soluciones y señala aquellos requisitos a los cuales se les deben dar más importancia por su dificultad o por su punto crítico.

- Pruebas unitarias: Cada vez que se implemente una parte de código, el desarrollador tiene que escribir una prueba, uno de los métodos más utilizados es el desarrollo guiado por pruebas conocido por sus siglas en ingles TDD (Test Driven Development), donde se realizan primero las pruebas luego se codifica y después se realiza la refactorización del código escrito.

- Pequeños releases e Integración continua: la integración continua permite a los desarrolladores integrar sus cambios varias veces al día para crear pequeñas versiones. Cada versión debe de ser tan pequeña como sea posible, conteniendo los requisitos de negocios más importantes y tiene que tener sentido como un todo.

  Este proceso evita que cuando se finalice el proyecto, al momento de que todos los desarrolladores deban integrar el código fuente, se produzca el llamado integration hell, el proceso tedioso de realizar merge y subir cambios en la

herramienta de control de versiones. Es necesario que al momento de integrar el código compile perfectamente.

- Metáfora: son breves descripciones de un trabajo de un sistema en lugar de los tradicionales diagramas y modelos.
- Diseño simple: se basa en la filosofía de que el mayor valor de negocio es entregado por el programa más sencillo que cumpla los requerimientos.
- Refactoring: sirve para minimizar el código duplicado y prepara al sistema para que en un futuro acepte nuevos cambios, la refactorización es mantener la sencillez, la claridad, la cantidad mínima de funcionalidades en el código.
- Programación en parejas: En XP se propone que dos personas escriban su código en una misma máquina, las parejas van cambiando en determinado tiempo para que el conocimiento se difunda en todo el equipo, esto para prevenir errores y para extender el uso de estándares de codificación.
- Propiedad Colectiva del código: ningún miembro del equipo es propietario del código, todo el que quiera modificar el código puede hacerlo.
- Semanas de 40 horas: En XP se espera que los miembros del equipo no deben trabajar horas extras, es necesario que los desarrolladores estén descansados y animados para codificar.
- Estándares de codificación: Incentiva a los desarrolladores a escribir todo el código de la misma manera, para que parezca que todo el sistema lo codifico la misma persona.

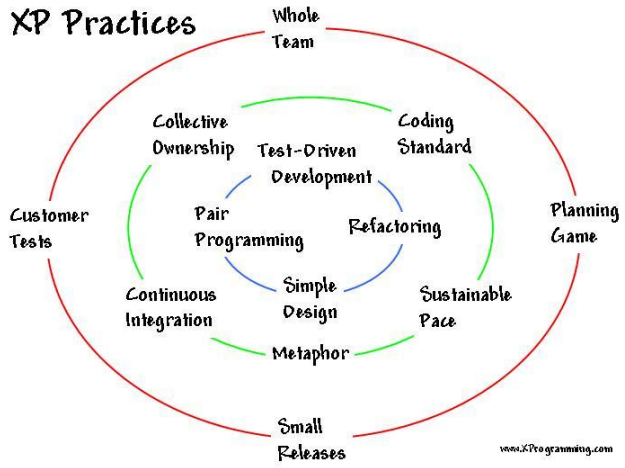

Fig. 2. Prácticas programación extrema. Tomado de [5]

1.2. Scrum

Otro exponente de las prácticas ágiles es Scrum, una metodología ágil de desarrollo de proyectos basada en el estudio realizado por Ikujijo Nonaka y Hirotaka Takeuchi en 1986 sobre las mejores prácticas de desarrollo de productos tecnológicos de importantes empresas japonesas. Pero fue en el año 1995 cuando Ken Schwaber y Jeff Sutherland lo presentaron formalmente como un proceso para el desarrollo de software [10].

El objetivo de Scrum es la de mejorar la realimentación del desarrollo para corregir errores en las fases tempranas. Scrum es ideal para proyectos con cambios persistentes o requerimientos emergentes tales como proyectos web o productos para un nuevo mercado [11].

Al igual que en XP, en Scrum se realizan pequeñas versiones en intervalos de tiempo llamados Sprint, estos ciclos por lo general son de un mes de duración. Antes de cada sprint se prioriza el trabajo y se seleccionan las tareas que se deben entregar al final del sprint.

Scrum es una metodología muy simple, de carácter adaptable y tiene una estructura incremental basada en iteraciones y revisiones.

El equipo de Scrum son equipos auto organizados y multifuncionales que consiste de tres roles esenciales

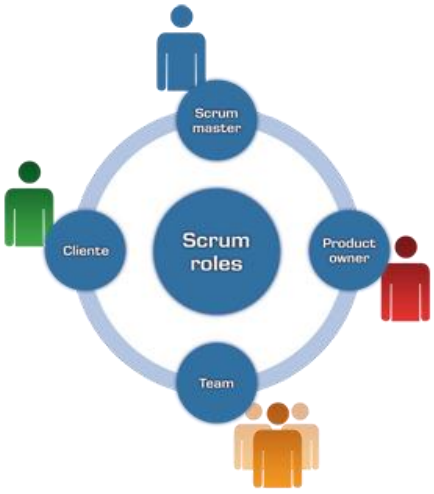

Fig. 3. Roles Scrum. Tomado de [12]

- Dueño del producto (Product Owner): Es la persona encargada de crear y administrar la lista del producto (product Backlog), además se asegura que el equipo de desarrollo entienda bien los elementos que contiene la lista.

- Facilitador (Scrum Master): El scrum master se asegura de que el proceso Scrum se cumpla como debe ser. Entrena al equipo de desarrollo en ser auto organizado y multifuncional.

- Equipo de desarrollo (Development Team): consiste en todas las personas responsables de la entrega del producto, el equipo de desarrollo tiene el poder de organizar y gestionar su propio trabajo, los equipos de desarrollo tienen que tener la siguientes características:

    - Son auto organizado, nadie le indica al equipo como deben realizar su trabajo.

    - Son multifuncionales, todos los miembros del equipo ejecutan las tareas necesarias para sacar adelante la entrega del producto.

    - Scrum no tiene títulos para los miembros del equipo de desarrollo todos son desarrolladores.

    - No existen sub-equipos en el equipo de desarrollo.

Componentes

- Lista de producto (Product Backlog): Es una lista que contiene todos los requerimientos necesarios del sistema, el encargado de gestionarla es el dueño del producto quien velara porque siempre se esté actualizando con las características y funcionalidades que se presenten para que el producto sea el más adecuado.

  La lista del producto se organiza por la prioridad que se le asigna a cada requerimiento, agregando mayor detalle y claridad a los elementos con mayor valor en la lista.

- Gráfico de trabajo (Burndown Chart): Es una gráfica que muestra la cantidad de elementos pendientes de la lista de producto.

- Lista de tareas del sprint (Sprint Backlog): Es un conjunto de elementos de la lista de producto seleccionados para el sprint

Reuniones

- Reunión de planificación del Sprint (Sprint Planning): Es una reunión de ocho horas donde se crea el plan de trabajo del sprint, por medio de la lista de productos se obtienen las actividades a realizar.

- Scrum diario (Daily stand up Scrum): Es una reunión de quince minutos que se celebra todos los días, para que cada miembro del equipo de desarrollo explique el avance de sus actividades, cada persona responde las siguientes preguntas:
    - ¿Qué he hecho desde la última reunión de sincronización?
    - ¿Pude hacer todo lo que tenía planeado? ¿Cuál fue el problema?
    - ¿Qué voy a hacer a partir de este momento?
    - ¿Qué impedimentos tengo o voy a tener para cumplir mis compromisos en esta iteración y en el proyecto?

- Revisión del sprint (Sprint review): Al final de un sprint se realiza la revisión del Sprint, donde el dueño del producto revisa que los productos entregados cumplan con los requerimientos definidos en la reunión de planificación. Además los miembros del equipo de desarrollo hablan sobre los problemas que se presentaron durante el sprint.
- Retrospectiva del sprint (Sprint retrospective): Es una reunión que se realiza después de la revisión del sprint, para que los integrantes del equipo scrum puedan crear un plan de mejoras

Durante el sprint se utilizan técnicas visuales para gestionar las tareas, se usa un tablero o un muro llamado Scrum Taskboard donde se lista el estado de los requerimientos, el objetivo es que quede claro el trabajo a realizar; también se utiliza el Burndown Chart un gráfico que muestra los avances y el trabajo pendiente.

Con la implementación de Scrum se incrementa la creatividad y facilita al equipo de trabajo a responder a la retroalimentación, provee un conjunto de simples reglas que crean una estructura para que los equipos se concentren en resolver los retos y dificultades.

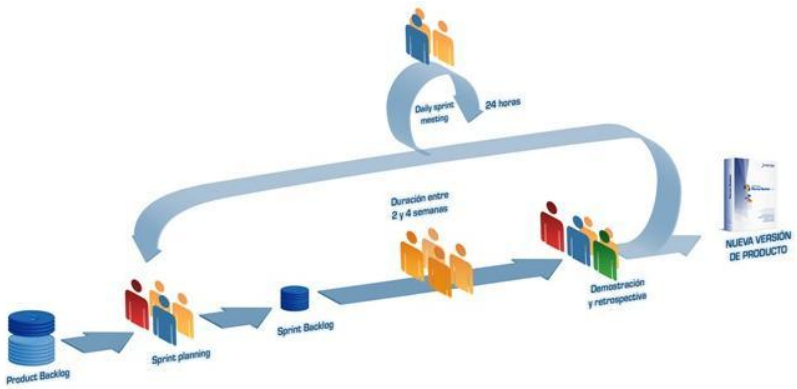

Fig. 4. Ciclo Scrum. Tomado de [12]

1.3. Desarrollo de Software Adaptativo

Conocido por sus siglas en inglés como ASD (Adaptive Software Development), es una técnica creada por Jim Highsmith y Sam Bayer en el año 1998 para el desarrollo de software complejo, tomando los conceptos de los sistemas adaptativos de inteligencia artificial.

Este método es iterativo e incremental favoreciendo el desarrollo del producto por medio de prototipos, esta metodología se enfoca en la colaboración humana y en la organización del equipo.

Las principales características del ASD son [13]:

- Proceso iterativo.
- Orientado a los componentes de software más que a las tareas en las que se va a alcanzar dicho objetivo.
- Tolerante a los cambios.
- Guiado por los riesgos
- La revisión de los componentes sirve para aprender de los errores y volver a iniciar el ciclo de desarrollo

Esta metodología se divide en tres fases la especulación, colaboración y aprendizaje.

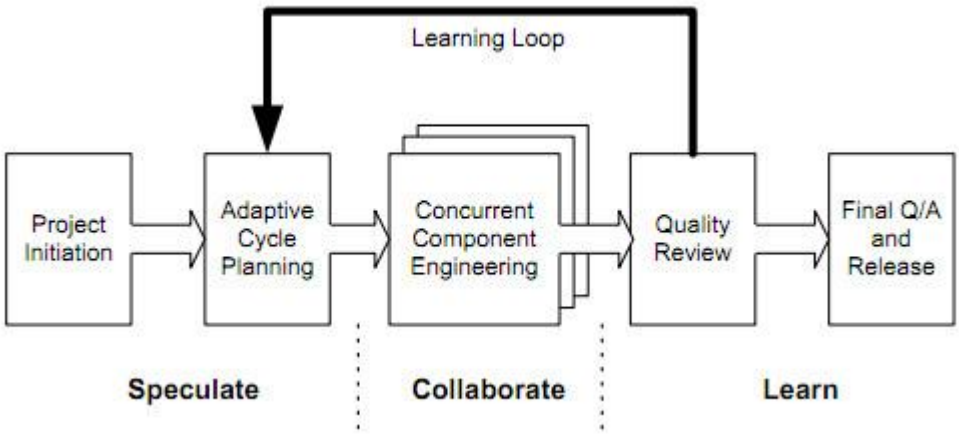

Fig. 5. Ciclo ASD. Tomado de [14]

La vida de ASD comienza con la fase de especulación, es aquí donde se planifican de manera tentativa los incrementos en el desarrollo del software, en cada uno de estos ciclos o iteraciones se tiene la opción de desviar el desarrollo, es decir, se puede cambiar el orden de entrega de las funcionalidades para no retrasar el proyecto.

La segunda es la fase de colaboración, cuando los miembros del equipo de desarrollo colaboran y se comunican entre sí para sacar adelante la entrega planificada en la fase de especulación.

La tercera fase es la de aprendizaje, en esta etapa se revisa la calidad de la entrega y con esto se pretende entender mejor los procesos y herramientas utilizados en todo el ciclo de desarrollo y se analizan cuatro factores que ayudaran a absorber este conocimiento:

• Calidad del resultado desde la perspectiva del cliente.

• Calidad del resultado desde la perspectiva técnica.

• El funcionamiento del equipo de desarrollo y las prácticas que este utiliza.

• El status del proyecto.

1.4. Lean

Antes de iniciar un proyecto las empresas analizan la viabilidad de este para saber qué beneficios se pueden obtener, esto es conocido como el retorno de la inversión o por sus siglas en ingles ROI (Return of Investmen), el objetivo de la metodología Lean es que se cumpla el ROI analizado al principio del proyecto.

Lean es un modelo de gestión utilizado en el sistema de producción de Toyota, cuyo fin es entregar el máximo valor para los clientes y evitar los desperdicios de recursos. Es una filosofía de trabajo basada en el culto japonés del Kaizen (mejora continua) en donde el único fin es la perfección en el producto que se entrega al cliente.

El concepto del Kaizen consiste en mejoras continuas que aunque sean pequeños cambios, influyen en la perfección no solo de los productos si no de las personas, “Hoy mejor que ayer, mañana mejor que hoy” [15].

Aunque este modelo empezó en la industria textil, se adaptó a la industria de los automóviles y ahora se adapta también al desarrollo de software, ya que no es una técnica de trabajo, es una manera de pensar y de actuar de una organización.

El objetivo principal de Lean está concentrado en el costo y el ROI de un proyecto, y a obtener la calidad de los productos desde el origen, esto se ve reflejado en sus siete principios [16].

- Eliminar desperdicio

- Ampliar el aprendizaje
- Decidir lo más tarde posible
- Reaccionar tan rápido como sea posible
- Potenciar el equipo
- Crear la integridad
- Ver todo el conjunto
- Eliminar desperdicio: La palabra desperdicio se describe mucho más profundo en japonés dando paso a tres términos:
    - ✓ Muri: Evitar la sobrecarga en los procesos.
    - ✓ Muda: Implica la eliminación de actividades que no dan valor desde la perspectiva del cliente
    - ✓ Mura: Significa desigualdad, esto quiere decir falta de regularidad en los procesos.
- Amplificar el aprendizaje: En Lean también se apoya la virtud del aprendizaje, por medio de reuniones cortas durante todo el proceso de desarrollo se retroalimenta junto con el cliente los puntos a favor y en contra que se encontraron durante el ciclo. Con estos pasos el cliente entiende cuáles son sus verdaderas necesidades y los desarrolladores aprenden a satisfacer estas necesidades.
- Decidir lo más tarde posible: Esta premisa advierte sobre la necesidad de tener toda la información disponible para tomar decisiones y tener un mejor enfoque con varias opciones al momento de diseñar; para así proporcionar un sistema flexible evitando que se necesite realizar cambios costosos después de finalizar el proyecto
- Entregar lo más rápido posible: Iteraciones de desarrollo cortas, al igual que en XP y en Scrum, Lean apoya la entrega rápida para obtener una retroalimentación rápida y evitar los errores.
- Dar poder al equipo: A los miembros del equipo de desarrollo se les da la oportunidad de tomar decisiones, determinar el tiempo que les conlleva una actividad asignada y con esto se logra motivarlos ya que saben que la empresa confía en ellos, además pueden tener acceso al cliente para comprender mejor las necesidades de este.
- Embeber la integridad: Siguiendo el lema de XP por la sencillez Lean apoya la refactorización para evitar código repetitivo y deficiente. Además se utiliza la herramienta test-driven-development para que el desarrollador realice la prueba unitaria antes de codificar, con la ayuda de las iteraciones cortas previene los defectos que se podrían encontrar al final del proyecto.

- Ver el todo: Este se refiere a optimizar a todo el sistema, no concentrarse en solo las partes principales de este, sin olvidar las pequeñas funcionalidades lo que da como resultado una visión global del proyecto.

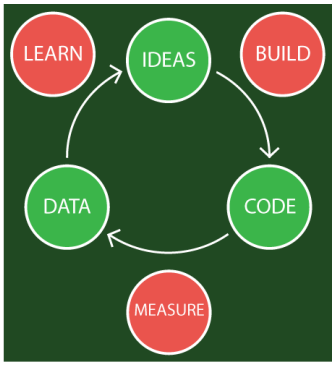

Fig. 6. Fases Lean. Tomado de [17]

1.5. MANIFIESTO ÁGIL

Al estudiar estas metodologías se puede observar que tienen muchas características en común, es por esto que en el año 2001 los creadores de las principales metodologías ágiles decidieron unificar estos aspectos, declarándolos como los principios del manifiesto ágil.

Estos son los doce principios del manifiesto ágil[1]:

- Nuestra mayor prioridad es satisfacer al cliente mediante la entrega temprana y continua de software con valor.
- Aceptamos que los requisitos cambien, incluso en etapas tardías del desarrollo. Los procesos Ágiles aprovechan el cambio para proporcionar ventaja competitiva al cliente.
- Entregamos software funcional frecuentemente, entre dos semanas y dos meses, con preferencia al periodo de tiempo más corto posible.
- Los responsables de negocio y los desarrolladores trabajamos juntos de forma cotidiana durante todo el proyecto.
- Los proyectos se desarrollan en torno a individuos motivados. Hay que darles el entorno y el apoyo que necesitan, y confiarles la ejecución del trabajo.

[1] Los doce principios del manifiesto ágil [27]

- El método más eficiente y efectivo de comunicar información al equipo de desarrollo y entre sus miembros es la conversación cara a cara.
- El software funcionando es la medida principal de progreso.
- Los procesos Ágiles promueven el desarrollo sostenible. Los promotores, desarrolladores y usuarios debemos ser capaces de mantener un ritmo constante de forma indefinida.
- La atención continua a la excelencia técnica y al buen diseño mejora la Agilidad.
- La simplicidad, o el arte de maximizar la cantidad de trabajo no realizado, es esencial.
- Las mejores arquitecturas, requisitos y diseños emergen de equipos auto-organizados.
- A intervalos regulares el equipo reflexiona sobre cómo ser más efectivo para a continuación ajustar y perfeccionar su comportamiento en consecuencia.

De estos doce principios se extraen cuatro características básicas que distinguen a las metodologías ágiles:

- ✓ Individuos e interacciones sobre procesos y herramientas
- ✓ Software funcionando sobre documentación comprensiva
- ✓ Colaboración del cliente sobre negociación de contrato
- ✓ Responder al cambio sobre seguir un plan

Según Kent Beck, Jeff Sutherland, Jim Highsmith y los demás miembros que conformaron el manifiesto ágil, estas cuatro características son necesarias para definir una metodología ágil, ligera y sin excesos de normatividad; valores que marcan la diferencia con los métodos tradicionales.

CAPÍTULO 2

El capítulo dos propone un conjunto de prácticas para el desarrollo de software basada en los valores expuestos en el manifiesto ágil. Esta propuesta sirve como guía metodológica que facilitará la ejecución de proyectos desarrollados por miembros de los semilleros de investigación y/o grupos de desarrolladores en formación y obtengan una experiencia con las prácticas ágiles en desarrollo de software. Está propuesta se orienta desde un marco de trabajo basado en el aprendizaje, la colaboración, aceptar el cambio, la intervención de diferentes actores en las etapas de desarrollo y la comunicación de todos los miembros del equipo.

2. PROPUESTA METODOLÓGICA

2.1. Criterios

“Individuos e Interacciones por sobre procesos y herramientas” con esta gran premisa del manifiesto ágil se deja claro que el factor humano es la clave principal en este tipo de metodologías, esto lo explican James Shore y Shane Warden [8] quienes manifiestan que el desarrollo ágil es un arte humano.

Las personas son el recurso más importante en un proyecto de ingeniería de software, sus conocimientos y experiencias son un gran aporte y el hecho de no saberlas motivar y ahuyentar su entusiasmo puede ser perjudicial para el resultado final del proyecto.

Es por esto que las metodologías ágiles han causado gran revuelo, el hecho de enfatizar en la motivación y en el bienestar de las personas por sobre la perfección de la documentación y los procesos hacen que los escépticos encuentren grandes falencias en este tipo de metodologías.

Al momento de implementar una metodología ágil en una empresa se debe tener en cuenta que cada individuo tiene actitudes y aptitudes diferentes frente a un mismo problema, para la aptitud o la carencia de esta, todas las metodologías ágiles manejan el concepto de retrospectiva o retroalimentación, un concepto que ayuda a que los miembros del equipo de desarrollo mejoren cada vez que terminen una iteración y puedan observar sus errores y conservar sus cualidades.

Para evitar que los miembros del equipo tengan una mala actitud frente al cambio de metodología se propone inculcar estas técnicas ágiles desde la formación académica, por ahora esta guía se dirige a los estudiantes que forman parte de semilleros de investigación, ya que fueron ellos mismos los que decidieron tomar parte en este tipo de actividades extracurriculares.

Después de que los estudiantes cambien su actitud frente al esquema que presentan las metodologías ágiles, se debe comenzar a enseñar el ambiente de trabajo en equipo, los integrantes deben confiar entre sí y ser responsables de que sus tareas se cumplan a tiempo y sin errores, estos equipos son denominados equipos auto organizados.

Las metodologías ágiles buscan que los integrantes de los equipos auto organizados cuenten con los siguientes valores [18]:

- ✓ Confianza.
- ✓ Respeto.
- ✓ Escucha.
- ✓ Colaboración.

- ✓ Lealtad.
- ✓ Disciplina.
- ✓ Altruismo.

Reglas de los equipos auto organizados:

- ✓ Recibir las críticas de manera constructiva.
- ✓ Tener ganas de aprender.
- ✓ No buscar culpables.
- ✓ No existen jefes, ellos son solo un miembro más del equipo de desarrollo.
- ✓ No buscar el crédito
- ✓ No hacer sentir mal a los demás.
- ✓ Nadie es superior a nadie.
- ✓ No opacar a los demás, se debe buscar el bien común.

Hacer que todas estas características funcionen en un grupo con un gran número de integrantes es una difícil tarea, es por esto que las metodologías ágiles sugieren que el número de personas que deben conformar el equipo de desarrollo no debe sobrepasar los siete integrantes.

Según el estudio expuesto por Vikash Lalsing en su artículo [19] a menudo la eficiencia del equipo depende de la interacción de sus integrantes, al aumentar la cantidad se requiere más interacción y es más complicado manejarlos, mientras más personas haya en un equipo más difícil será la comunicación entre ellos, esto afecta la eficiencia y la productividad de cada uno.

Aunque en la última década se han creado muchas metodologías ágiles que se basan en la teoría empírica, muy pocas empresas han podido implementarlas de manera adecuada, en primer lugar porque es un cambio en el paradigma de la empresa y segundo porque estas metodologías implican cambios profundos en las estrategias y comportamiento interno del grupo de trabajo.

Uno de los mayores impedimentos de la implementación de este tipo de metodologías, consiste en la resistencia al cambio, y la mala actitud de los integrantes del grupo de desarrollo.

Debido a esto se propone listar un conjunto de principios ágiles, que se puedan aplicar en los semilleros de investigación y de desarrollo de software, para que los estudiantes

puedan llevar a cabo una experiencia similar a la que se presenta en las empresas al momento de desarrollar un proyecto a gran escala, y así poder mejorar su desempeño en un ambiente de técnicas ágiles.

El objetivo de combinar estas metodologías ágiles es el de permitir que los estudiantes obtengan la experiencia necesaria con las técnicas más importantes implementadas en cada una de estas, igualmente, facilitar su entendimiento, hasta volverlas parte natural de los procesos de los participantes de cada uno de los grupos.

La principal característica de este conjunto de prácticas es el énfasis que tiene en la formación y preparación de los estudiantes, orientándolos a un marco de trabajo basado en el aprendizaje, la colaboración, la buena actitud al cambio, la intervención de diferentes procesos en las etapas de desarrollo y la comunicación de todos los miembros del equipo.

2.2. Objetivos de la propuesta

Los objetivos de emplear este método de trabajo en los semilleros de investigación y en los grupos de desarrollo son los siguientes:

- ✓ Incentivar a los estudiantes a aplicar las diferentes metodologías que existen en la ingeniería de software, ya que, aunque en el ámbito universitario no se aprecia el uso de estos métodos, son un gran aporte para las empresas de desarrollo de software que carecen de esta cultura.
- ✓ Mostrar a los estudiantes las ventajas que conlleva el uso de una metodología en el desarrollo de proyectos de software.
- ✓ Ayudar a los integrantes de los semilleros a planificar, diseñar y desarrollar sistemas robustos
- ✓ Promover los valores inculcados por las metodologías ágiles, como lo son la comunicación con todo el equipo de trabajo, la colaboración, la confianza, el respeto y la lealtad.
- ✓ Preparar a los participantes desde una temprana etapa de su formación académica para los ambientes y diferentes escenarios que se podrán encontrar en el ámbito laboral.

2.3. Marco de trabajo

Siguiendo la filosofía de los procesos ágiles se estructura una metodología iterativa e incremental basada en la colaboración y el aprendizaje, propiedades de ADS, de Lean y que además contará con los mismos roles de Scrum. Este marco de trabajo consta de dos fases que combinan técnicas de estas metodologías ágiles.

- Fase Planificación (Especulación)
- Fase Implementación (Colaboración- Aprendizaje)

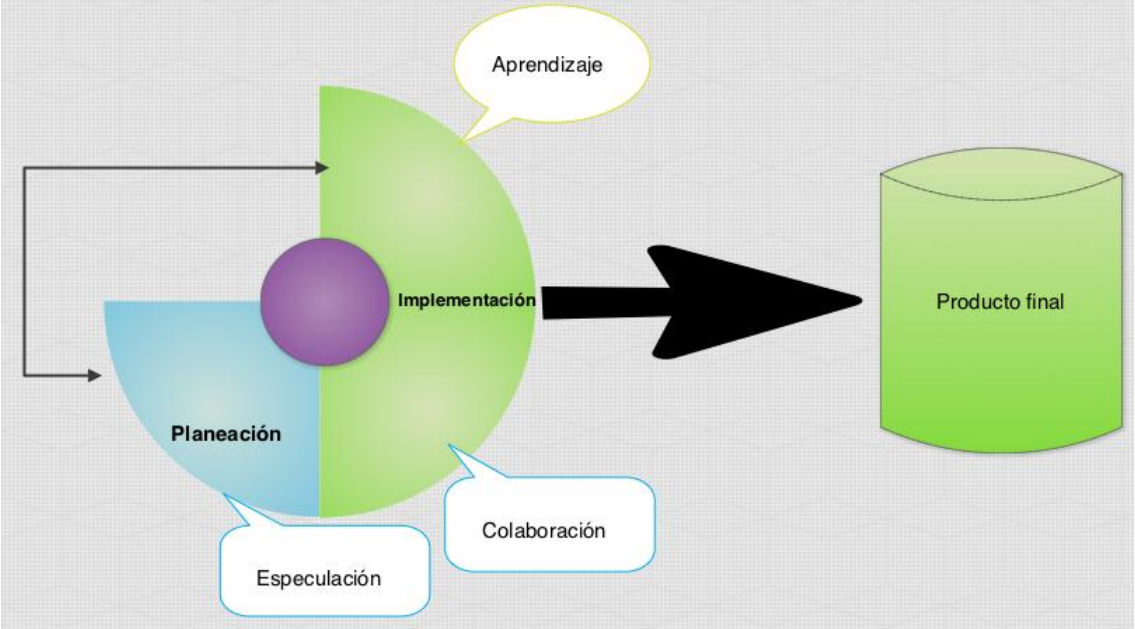

Fig. 7. Fases Metodología[2]

[2] Construcción propia del autor.

En cada una de estas fases se tendrán diversas actividades, las cuales a su vez tendrán cuantas iteraciones sean necesarias para lograr las metas definidas al inicio de cada actividad.

Fases	Actividades	Prácticas	Artefactos (Entregables)
Planeación	Especulación	Historias de usuario	Lista Requisitos (Product backlog)
		Pruebas de aceptación	
		Realizar cronograma	Cronograma
		Diseño de la solución	Diagramas UML
			Prototipos GUI
			Estándares de Codificación
			Frameworks
Implementación	Colaboración y Aprendizaje	Codificación	Plan de pruebas Ejecutables
		Pruebas unitarias	
		Refactoring	
		Integración Continua	

Tabla 1. Fases Metodología[3]

[3] Construcción propia del autor.

2.3.1 Roles

Los roles planteados en esta propuesta son los mismos que se emplean en Scrum ya que en esta metodología todo el equipo de desarrollo es el encargado de entregar el software, no hay distinciones ni títulos entre los integrantes todos trabajan con el mismo empeño para lograr el objetivo de la iteración, gracias a este enfoque se le brinda experiencia a todos los estudiantes en todas las áreas de la creación de un producto de software.

1. Propietario del producto: El propietario del producto puede ser una o varias personas que representan al cliente y a los usuarios finales del sistema de software, ellos son los encargados de realizar las historias de usuario y responsables de que todos las funcionalidades propuestas se cumplan.

2. Facilitador: Es una persona, integrante del equipo de desarrollo, que se asegura de que el proceso se utiliza como es debido y evita que existan obstáculos que le impidan al equipo de desarrollo realizar las actividades propuestas al principio de la iteración. Además se encargará de transmitir el conocimiento adecuado para que los estudiantes puedan cumplir con las actividades asignadas.

3. Equipo: Son todos los integrantes del equipo de desarrollo, los cuales tendrán los mismos roles, en cuanto al análisis, desarrollo y pruebas, para que puedan experimentar ambos lados del desarrollo de software.

Fig. 8. Roles [4]

[4] Construcción propia del autor.

2.3.2. Fases

2.3.2.1. Fase de planificación

Esta es la fase inicial del proceso donde los miembros del semillero, con la colaboración del cliente obtienen el alcance del proyecto, además se recolecta los requerimientos del sistema por medio de las historias de usuario que se obtienen de las reuniones periódicas sostenidas con el cliente.

Cuando los integrantes del equipo comprenden el alcance del proyecto, comienzan con la clasificación de los requerimientos, dando prioridad a las funcionalidades primordiales llevándolas a la lista de requerimientos funcionales y no funcionales

Con la lista de requerimientos comienza la planeación de las entregas, tomando los requerimientos con prioridad alta y se les asigna una fecha de entrega, planeando así el cronograma de la primera iteración.

Tomando el concepto de Scrum del tablero de tareas (Scrum Taskboard), con la ayuda de los tableros virtuales los estudiantes organizarán las actividades de cada iteración y tendrán un mejor control de cada actividad a realizar.

A medida que se define el sistema, por medio de la especulación, se comienza a diseñar un bosquejo de la arquitectura del primer prototipo para analizar las restricciones y riesgos que se puedan presentar a lo largo del proyecto.

En esta fase también se seleccionarán las herramientas de desarrollo que mejor se adapten a las necesidades definidas por el cliente, y siguiendo la técnica expuesta por XP de la programación por parejas, se definen los equipos de trabajo para el intercambio de conocimientos.

Los estudiantes que no tengan conocimiento sobre algunas de estas actividades o herramientas, serán capacitados por los compañeros que ya tengan este tipo experiencias o que deciden investigar sobre el tema para luego exponer el conocimiento adquirido. Claro está que este tiempo adicional de enseñanza aprendizaje se le debe agregar al tiempo del proyecto.

Adicionalmente, se define los estándares de codificación, tanto para la codificación en base de datos, herramienta de desarrollo y demás utilitarios que intervengan en la construcción del sistema, con la finalidad de que todos los miembros del equipo puedan entender el código fuente cuando requieran realizar modificaciones en funcionalidades que no hayan sido realizadas por ellos.

Listado de productos de la primera fase:

- Lista de requerimientos funcionales y no funcionales: Este documento es mantenido por el facilitador, y se obtiene por medio de las Historias de usuario, es como contrato entre el Cliente y el Equipo de Desarrollo respecto a lo que se va a construir.

- Cronograma de reuniones y de entregas de versiones: Los cronogramas también serán gestionados por el facilitador, y basados en las fechas estipuladas se administrará el tiempo de entrega de las versiones.

- Herramientas de desarrollo: Dependiendo de la arquitectura seleccionada para el proyecto, se elegirán las herramientas que mejor se adapten a la arquitectura propuesta.

- Estándares de codificación: Basados en las herramientas de desarrollo, los miembros del equipo establecerán los estándares de codificación que serán utilizados en todo el proceso de codificación.

2.3.2.2. Fase de implementación:

En la segunda fase del proceso, el equipo de desarrollo define la arquitectura del software teniendo en cuenta los requerimientos tomados en la primera fase. Se evoca la característica de XP para conservar la simplicidad en el diseño evitando la duplicidad en la lógica y minimizando la cantidad de métodos para satisfacer las necesidades del cliente.

Se debe diseñar el sistema de forma que sea fácil de utilizar por el usuario, sea flexible, con gran escalabilidad y se adapte a los cambios que se puedan presentar en cualquier etapa de la vida del software.

Teniendo una imagen completa del sistema, se comienza con el diseño de las interfaces de usuario (GUI) buscando una estrategia que satisfaga las necesidades puntuales del cliente y que a su vez facilite su mantenimiento a través del tiempo.

También se realiza el diseño del modelo estructural de base de datos que permita soportar la lógica de sistema y negocio que se ha propuesto hasta esta etapa.

En esta fase se comienza la codificación utilizando los estándares de codificación definidos en la fase de planeación, es en esta fase donde entra el concepto de refactorización, otra de las características básicas de las metodologías ágiles.

Después de implementar una nueva característica al sistema, se busca la manera de hacerlo lo más sencillo posible pero sin perder la funcionalidad, a este proceso se le llama refactorización; este, aunque implica un esfuerzo e inversión mayor en la etapa de desarrollo, permite minimizar el código duplicado, adoptar mejores prácticas,

implementar nuevas tecnologías y preparar el sistema para recibir de la forma más transparente posible los cambios que sean necesarios a través de su tiempo de vida.

En esta fase se realiza como tarea adicional al proceso de desarrollo, la primera aplicación de pruebas, las cuales son ejecutadas por los desarrolladores con el fin de probar y certificar las modificaciones y las nuevas funcionalidades inyectadas en la etapa de desarrollo actual.

Una vez se certifican las modificaciones y nuevas funcionalidades implementadas, el equipo de desarrollo procede a realizar la integración de las modificaciones y cambios realizados para la implementación de las funcionalidades, haciendo entrega de cada sección de desarrollo para su posterior revisión y certificación.

Por último, con base al entregable realizado durante la integración, se realizan las pruebas finales de las funcionalidades agregadas, con las cuales se busca que las modificaciones generadas cumplan y satisfagan todos los requerimientos que se pensaban resolver durante esta etapa.

Adicionalmente, es posible que durante la ejecución de estas etapas se identifiquen nuevos requerimientos o cambios en las necesidades expresadas inicialmente por el cliente, para las cuales se inicia nuevamente el proceso desde la etapa de planeación.

Listado de productos segunda fase:

- Plan de pruebas: los casos de prueba contienen la especificación de cómo será validado el sistema. Estos son realizados basándose en los casos de uso y planteando todos los escenarios posibles que éstos pueden contener.

- Ejecutables: Este artefacto incluye el ejecutable de código fuente y los scripts de despliegue, necesarios para la instalación del producto en el ambiente del cliente.

CAPITULO 3

En este capítulo se describe el proceso que llevaron a cabo estudiantes y docentes de los semilleros de investigación para el desarrollo del primer módulo Agenda denominado UNotes del proyecto “UPB Móvil”. Este proceso es el primer piloto de las prácticas ágiles propuestas en este trabajo y fue realizado por los grupos GICU adscrito a la facultad de Comunicación Social y grupo GIDATI adscrito a la facultad de Ingeniería en TIC de la Universidad Pontificia Bolivariana sede Medellín.

3. APLICACIÓN EN SEMILLERO DE INVESTIGACIÓN

Con este proyecto se buscó identificar las prácticas ágiles que mejor se adapten a la formación de los estudiantes de semilleros de investigación y de grupos de desarrollo de software. Indagar sobre las técnicas de las metodologías ágiles más representativas de la ingeniería de software, que incluyeran prácticas para la gestión de diversas expectativas y el alto porcentaje de incertidumbre asociada a proyectos de investigación formativa.

Estos criterios fueron decisivos para indagar en procesos ágiles definidos y documentados como programación extrema XP, SCRUM, ASD, Lean entre otros, que nos permitiera encontrar pautas importantes para fomentar la cultura del trabajo colaborativo, promover el aprendizaje autónomo y aplicar estrategias para el trabajo interdisciplinario.

El proyecto UPB Móvil comenzó como una investigación formativa de los estudiantes de comunicación social, acerca del uso de los dispositivos móviles en la universidad Pontificia Bolivariana; como resultado de este proceso se identificó la oportunidad de aprovechar este levantamiento de escenarios vinculado con las aplicaciones móviles, y así intentar capitalizar en desarrollo de prototipos las necesidades y expectativas que fueron manifestadas por los estudiantes y docentes.

La características asociadas con el proyecto UPB Móvil, desarrollo a largo plazo, complejidad creciente y la posibilidad de potenciar en un solo aplicativo los diferentes sistemas que existen en la universidad, el semillero de Comunicación digital y el Semillero Internet of Things asumen el reto de dejar un aporte a la universidad con el desarrollo de los prototipos de software necesarios para concretar el proyecto, y en este sentido, se busca implementar un estrategia que permita formalizar la documentación y el proceso de desarrollo.

Los Integrantes del semillero Internet of Things, deciden rastrear y aplicar para el desarrollo de un primer prototipo las prácticas ágiles estudiadas en sus cursos de Ingeniería de software del programa de Ingeniería de Sistemas e Informática de la Facultad de Ingeniería en TIC UPB e iniciar bajo este marco metodológico el desarrollo del proyecto UPB Móvil.

A continuación se describen los componentes de la estrategia aplicada para el desarrollo del aplicativo UNotes en el marco del proyecto UPB Móvil.

3.1. Roles:

Propietario del producto: El propietario del producto fue representado por estudiantes del semillero de Comunicación Digital adscrito al grupo GICU de la facultad de Comunicación Social y Periodismo UPB, cuyos miembros fueron los encargados de definir, especificar y validar los requerimientos del proyecto.

Facilitador: La persona encargada de suministrar a los estudiantes las herramientas necesarias para el desarrollo del proyecto y establecer las dinámicas de trabajo del equipo así como la interlocución entre semilleros en este proyecto fue el Ingeniero Oscar Eduardo Sánchez.

Equipo: El equipo de desarrollo fue conformado por estudiantes de Ingeniería de Sistemas e Informática de segundo semestre en adelante y con grados de experiencia diversos en las tecnologías del proyecto, fueron integrantes del semillero Internet of Things desempeñando tareas de análisis, desarrollo y pruebas.

3.2. Fase de Planeación

Los estudiantes del semillero Internet of Things iniciaron reuniones periódicas con los integrantes del semillero de Comunicación Digital, esto con el propósito de reconocer el alcance del proyecto UPB Móvil mediante técnicas de levantamiento, especificación y estimación de requisitos apoyadas en historias de usuarios. A medida que realizaban la tarea de definir la lista del requerimiento con el propietario del producto, fueron realizando talleres de formación en herramientas de desarrollo de aplicaciones para dispositivos móviles.

Algunos de los integrantes del Semillero IoT con un mayor grado de experticia en el manejo de herramientas, expusieron sobre los ambientes de desarrollo y formas de licenciamiento a los demás integrantes del semillero, para identificar cual era la que mejor se adaptaba al proyecto. Se realizaron talleres con una primera herramienta gratuita para el desarrollo de aplicaciones de dispositivos móviles mediante tecnologías Web llamada Phonegap. Este framework permite la construcción de Apps multiplataforma, es decir, el prototipo a desarrollar podría ser utilizado en sistemas operativos como iOS, Android, BlackBerry OS, Windows Phone 7 o Symbian.

Los estudiantes también investigaron sobre los tableros tipo Kanban para la gestión de las actividades del equipo. Estos herramientas son tablas de control de tareas ampliamente utilizadas en metodologías como Lean y Scrum. La herramienta elegida por su usabilidad y licenciamiento fue Trello, que básicamente es una aplicación web que permite almacenar lista de tareas en un tablero virtual, esto permitió que los integrantes del semillero realizaran el control de tareas asignadas en las reuniones presenciales mediante una herramienta libre y de acceso virtual.

Después de concretar los requisitos con el semillero de comunicación social, los integrantes de Internet of Things, clasificaron las funcionalidades de mayor prioridad y diseñaron las interfaces de usuario que fueron aprobadas por comunicación social.

Luego de la especulación con los ambientes de desarrollo y técnicas para la gestión de las historias de usuario, se procedió a concretar los requisitos con el Semillero de Comunicación social. El documento de especificación de requisitos (Product Backlog) fue el insumo para que integrantes del semillero Internet of Things, clasificaran funcionalidades, priorizaran tareas y estimaran esfuerzo, esto acompañado de una estrategia de validación de requerimientos denominada prototipado interfaces graficas de usuario.

3.3. Fase de Implementación

El proyecto UPB Móvil será considerado como el APP institucional, es necesario que la propuesta gráfica del sistema tenga colores, tipologías e imágenes alineadas con la imagen corporativa de la universidad. Esto obligó a que los integrantes del equipo buscaran apoyo en la dependencia de comunicaciones de la UPB para el diseño gráfico del aplicativo.

Fig. 9. Interfaz de usuario Unotes

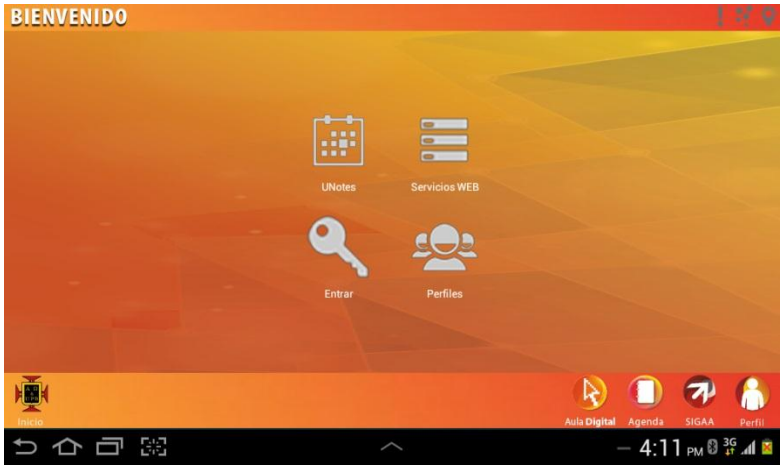

Fig. 10. Interfaz de usuario principal

En cuanto al diseño del sistema se realizó el modelado de datos y la definición de diagramas de procesos y bloques para apoyar la definición de la arquitectura.

Los estudiantes también procedieron a definir la arquitectura del primer módulo arquitectura Cliente servidor tres capas:

- Front End: Capa encargada de especificar los componentes del sistema que se ejecutarán del lado del cliente (Dispositivo Móvil), teniendo como punto departida las restricciones de procesamiento, almacenamiento, tamaños de pantalla y carga de estos dispositivos.

- Services Layer (Capa de Servicios): Capa encargada de la gestión de los datos y transporte del cliente al servidor y viceversa. Esta capa incluye un conjunto de funcionalidades expuestas en la nube como, autenticación, registro, notificaciones entre otros.

- Data Acces Layer (Capa de Acceso a datos): Es la capa encargada del almacenamiento de la información que circula por el sistema. Esta capa incluye un sistema de gestión de bases de Datos (SGBD) que soporte el modelo relacional para la organización de información.

Finalmente, el equipo procedió en la fase de implementación a realizar la codificación, pruebas e integración en ciclos de desarrollo de 30 días, lo cual permitió la construcción de versiones incrementales del producto que a su vez fueron validados por los estudiantes del Semillero de Comunicación Digital en cada entrega.

El prototipo desarrollado fue el pilotaje de un conjunto de prácticas ágiles en el marco de la propuesta expuesta en este trabajo. Este pilotaje fue propuesto con el propósito

de encontrar y/o afirmar las técnicas trabajadas y someterlas a prueba para posteriores revisiones y ajustes de la guía metodológica aplicable a semilleros de investigación y grupos de desarrolladores en formación.

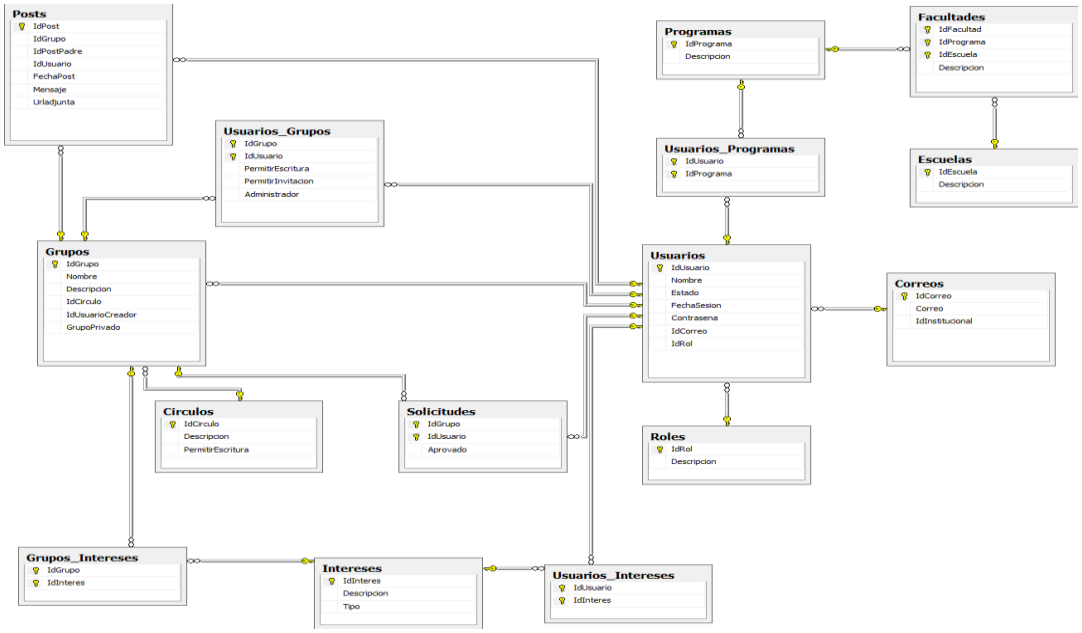

Fig. 11. Modelo de datos

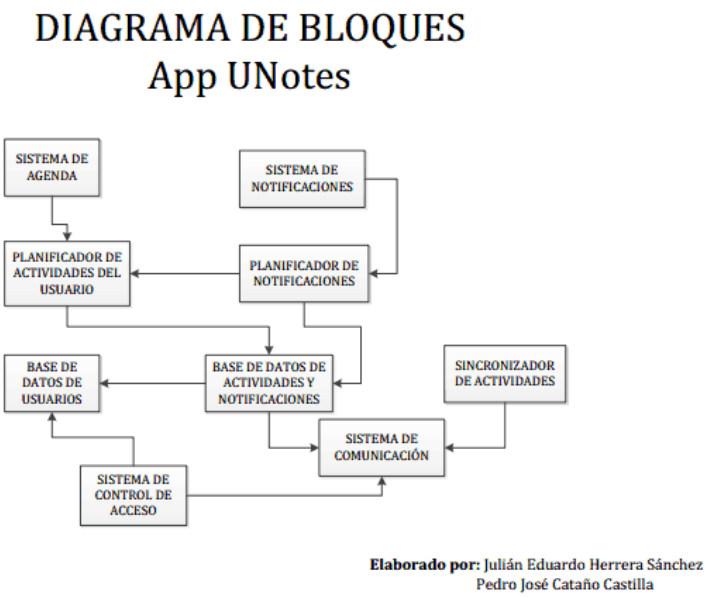

Fig. 12. Diagrama de bloques

CONCLUSIONES

Existe una variedad de metodologías ágiles que permiten gestionar, administrar y planear desarrollos de software, sin embargo las características de los miembros en algunos equipos de trabajo y las condiciones de operación de los mismos requieren adaptar procesos que permitan articular prácticas provenientes del diversas fuentes de las cuales se pueden adaptar técnicas hibridas que logren ajustarse a las diferentes necesidades que presentan los proyectos de software.

Con este ejercicio se buscó generar mecanismos alternativos para la selección de estrategias metodológicas mediante la caracterización del grupo de desarrolladores. Se propone un diseño metodológico flexible sin desconocer la rigurosidad asociada a la elección de la metodología a adoptarse en un proyecto, y finalmente una propuesta de aplicación práctica de un conjunto de actividades a implementar en busca de potenciar las características de cada una y articularlas para el logro de mejores resultados.

También, se quiso proveer a los estudiantes integrantes de los grupos de investigación de un conjunto de técnicas probadas en diferentes metodologías de desarrollo de software, las cuales puedan ser aplicadas en proyectos de investigación formativa y en los retos laborales que se puedan encontrar en su desempeño profesional.

Por las experiencias vividas en esta investigación y validadas en el ejercicio profesional, se concluye que la importancia concedida al tipo de metodología que se aplique a un proyecto de software, se traslada a la actitud y la disposición de los integrantes del equipo de desarrollo. Los factores humanos son las variables más influyentes para un resultado exitoso.

Finalmente se hace énfasis en la importancia del aprendizaje continuo, la colaboración y la comunicación, capacidades y competencias que deben adquirir los miembros de un equipo de desarrollo para sacar adelante un proyecto de software.

BIBLIOGRAFÍA

[1] A. I. Wasserman, «Software Engineering Issues for Mobile application Development,» [En línea]. Available: http://www.cmu.edu/silicon-valley/wmse/wasserman-foser2010.pd. [Último acceso: 13 Noviembre 2012].

[2] P. Mejía Alvarez, «Ingenieria de software,» Septiembre 2003. [En línea]. Available: http://www.ctic.uni.edu.pe/files/insoft01.pdf. [Último acceso: 18 Mayo 2013].

[3] W. S. Humphrey, PSP A self-Improvement Process for Software Engineers, Addison-Wesley, 2005.

[4] E. M. H. Jiménez y S. D. J. Orantes, «Metodologías híbridas para desarrollo de software: una opción factible para México,» *Revista Digital Universitaria,* nº http://www.revista.unam.mx/vol.13/num1/art16/art16.pdf, 2012.

[5] R. Jeffries, 2013. [En línea]. Available: http://xprogramming.com/what-is-extreme-programming.

[6] J. Shore y S. Warden, The art of agile development, O'Reilly Media, 2007.

[7] H. Jaspe, «Ingeniería de Software,» 10 Agosto 2012. [En línea]. Available: http://hjaspe.blogspot.com/2012/08/ingenieria-del-software-la-ingenieria.html. [Último acceso: 6 Julio 2013].

[8] K. Schwaber y J. Sutherland, «Scrum,» [En línea]. Available: https://www.scrum.org/Resources/What-is-Scrum. [Último acceso: 19 Enero 2013].

[9] M. Cohn, Succeeding with Agile, Software Development Using Scrum, Addison-Wesley Pearson Education.

[10] «SOFTENG,» [En línea]. Available: http://www.softeng.es/es-es/empresa/metodologias-de-trabajo/metodologia-scrum/proceso-roles-de-scrum.html. [Último acceso: 20 Abril 2013].

[11] «Ingeniería de Software,» 2013. [En línea]. Available: http://ingenieriadesoftware.mex.tl/61154_ASD.html. [Último acceso: 15 Junio 2013].

[12] M. Behr , «Desarrollo de Software Adaptable,» [En línea]. Available: http://www.concierto.com/articulos/archivos/2007/02/desarrollo-de-software-adaptable/. [Último acceso: 6 Julio 2013].

[13] [En línea]. Available: http://tesis.uson.mx/digital/tesis/docs/12021/Capitulo3.pdf.

[14] «Lean Entreprise Institute,» [En línea]. Available: http://www.lean.org/whatslean/. [Último acceso: 4 Mayo 2013].

[15] «Agilezen,» [En línea]. Available: http://www.agilezen.com/uses/lean-startup/. [Último acceso: 18 Mayo 2013].

[16] M. Alaimo y P. Tortollera, «Introducción a Scrum,» de *Agile Open Medellín*, Medellín, 2013.

[17] V. Lalsing, S. Kishnah y S. Pudaruth, «People factors in agile software development and project management,» *International Journal of Software Engineering & Applications,* vol. III, pp. 117-137, 2012.

[18] « iOS Dev Center,» [En línea]. Available: https://developer.apple.com/devcenter/ios/index.action. [Último acceso: 18 Noviembre 2012].

[19] «IBM developerworks,» [En línea]. Available: https://www.ibm.com/developerworks/mobile/library/mo-rapid-development/index.html. [Último acceso: 20 Noviembre 2012].

[20] «Android Developers,» [En línea]. Available: http://developer.android.com/index.html. [Último acceso: 18 Noviembre 2012].

[21] «All about agile,» [En línea]. Available: http://www.allaboutagile.com/7-key-principles-of-lean-software-development-2/. [Último acceso: 19 Mayo 2013].

[22] P. Blanco, «Metodología de desarrollo ágil para sistemas móviles Introducción al desarrollo con Android y el iPhone,» [En línea]. Available: http://www.adamwesterski.com/wp-content/files/docsCursos/Agile_doc_TemasAnv.pdf. [Último acceso: 18 Noviembre 2012].

[23] F. «Desarrollo Adaptativo de Software,» 13 Junio 2011. [En línea]. Available: http://desarrolloadaptativodesoftware.blogspot.com/2011/06/desarrollo-adaptativo-de-software-das.html. [Último acceso: 11 Mayo 2013].

[24] «Microsoft,» [En línea]. Available: http://msdn.microsoft.com/en-us/library/vstudio/hh533841.aspx. [Último acceso: 18 Mayo 2013].

[25] «Manifiesto Agil,» 2001. [En línea]. Available: http://agilemanifesto.org/iso/es/principles.html. [Último acceso: 19 Mayo 2013].

[26] H. Kniberg, Scrum y XP desde las trincheras, Lulu, 2007.

ANEXO 1

INTERFACES GRÁFICAS DE USUARIOS

Fig. 13. Interfaz de usuario Agenda

Fig. 14. Interfaz de usuario Loging

ANEXO 2

ARTÍCULO PUBLICABLE