Enfoque Ágil XP Aplicada a la Educación de desarrollo de Software

Arbeláez, Cesar, Astudillo, Fabián y Polindara, Ana María
caaber2@gmial.com, anie_polindara@hotmail.com, fara0307@hotmail.com
Fundación Universitaria de Popayán

Resumen—Este artículo explora la necesidad que hay en el aula de clases por enseñar a los estudiantes como programar sin necesidad de técnicas expuestas en la normalidad, si no con nuevas propuestas que llevan al estudiante aprender de una manera más didáctica, con un diálogo más personalizado con el docente y de esta manera hacer que se sienta cómodo, además ayuda al estudiante hacerlo más competente y por ende encuentra eficiencia en los procesos afines al desarrollo de software con esta metodología implementada.

Índice de Términos—XP, programación, UML, java.

Abstract—This article explores the need that exists in the class room by teach to the students how to program without need techniques exposed normally, but with new proposals that take to the student to learn in a way more didactic, with a dialogue more customized with the teacher and thus to make that feel comfortable, also it helps the student to be more competent and hence find efficiency in the processes related with software development with this methodology implemented.

Index of Terms—XP, Programming, UML, Java.

I. INTRODUCCIÓN

Una de las dificultades que se encuentran en el desarrollo de software es la enfatización que tienen los docentes al producto que se obtendrá y no se centra en las técnicas y métodos que propongan soluciones de calidad.

Tomando en consideración lo antes expuesto, al tratar de introducir la enseñanza de un lenguaje de Programación, como lo es Java, debemos tomar en cuenta que existe una gran diversidad en las habilidades, velocidad en el aprendizaje y aptitudes de los estudiantes, es por ello que la atención sobre las dificultades que enfrenta una persona no puede ser tratada fácilmente en una clase en la que hay muchos estudiantes, de tal forma que los docentes deben buscar diversas vías alternativas que ayuden a los estudiantes en su aprendizaje[1][2].

Una de las dificultades que surgen en el ámbito de la educación en el aula de clases de un curso de programación y en especial en la Fundación Universitaria de Popayán; es la escasa experiencia y predisposición de los alumnos a trabajar o a estudiar para aprender [3] en gran parte de los cursos de programación en las universidades, la mayoría de estudiantes se preparan para el momento, es por eso que no es sólo la comunidad software la que está preocupada, sino también la comunidad académica, ya que su metodología es demasiado gruesa y resistente, y ha comenzado a trabajar en el desarrollo de técnicas para aumentar la participación de los estudiantes, en lugar de utilizar planes de estudio que solo hacen preocupar al estudiante en cómo ganar una materia, pero realmente con esta metodología no los están preparando para los desafíos de la vida cotidiana, ya que es allá donde aquellos conocimientos son realmente aplicados.

Muchos métodos han sido utilizados por los docentes en la enseñanza de la programación, como lo son [1]:

- La elección del primer lenguaje de programación;
- La elección del libro de texto del curso;
- La elección del ambiente de programación;
- Variaciones en los métodos de enseñanza;
- Variaciones en las asignaciones.

La programación extrema o eXtreme Programming (XP) es un enfoque de la ingeniería de software formulado por Kent Beck, autor del primer libro sobre la materia, Extreme Programming Explained: Embrace Change (1999) [10]. Es el más destacado de los procesos ágiles de desarrollo de software.

Es una metodología ágil centrada en potenciar las relaciones interpersonales como clave para el éxito en desarrollo de software, promoviendo el trabajo en equipo, preocupándose por el aprendizaje de los desarrolladores, y propiciando un buen clima de trabajo.

XP se basa en la realimentación continua entre el cliente y el equipo de desarrollo, comunicación fluida entre todos los participantes, simplicidad en las soluciones implementadas y coraje para enfrentar los cambios.

XP se define como especialmente adecuada para proyectos con requisitos imprecisos y muy cambiantes, y donde existe un alto riesgo técnico.

[11] Teniendo en cuenta que XP se basa en lo antes mencionado, es ahí donde puede ser rentable para darle soluciones efectivas y eficaces a las metodologías de diseño para la educación; ya que las tradicionales no han sido útiles hasta el momento.

En este escenario surge la pregunta: ¿Cómo aplicar metodología ágil XP (Programación Extrema) para metodologías de educación común en programación? Es precisamente en este escenario donde nuestro trabajo propone una metodología de enseñanza utilizando XP (Extreme Programming), para estudiantes de programación de la FUP. Para realizarlo, primero realizamos un diagnóstico de las capacidades de programación de los estudiantes, posteriormente proponemos una metodología ágil XP para la enseñanza de la programación. Esta metodología es aplicada en los estudiantes de la Fundación Universitaria de Popayán.

En este trabajo exploramos el uso de la metodología ágil XP en el desarrollo de las actividades pedagógicas facilitando el aprendizaje con el propósito de formalizar un marco de trabajo cuyas fases y actividades permitan definir ciclos de vida que aseguren metodológicamente su calidad. Vamos a utilizar este marco para sugerir prácticas que pueden hacer la enseñanza más ágil. Por ende, se pretende impulsar una nueva metodología en la Fundación Universitaria de Popayán para los estudiantes de Ingeniería de Sistemas, en especial en el área de Programación en Java, usando la programación extrema como herramienta en los procesos de desarrollo de software, ya que reduce el tiempo de análisis, desarrollo y ejecución. De esta forma se mejora la calidad del software realizado por los estudiantes en el aula de clases.

II. ESTADO DEL ARTE

A. Metodología Tradicional

La ACM y la IEEE-CS han propuesto en conjunto un documento denominado *Software Engineering 2004* (SE2004), el cual provee recomendaciones para la educación de pre-grado en ingeniería de software. Este documento, el *Software Engineering Education Knowledge* (SEEK) [12], incluye una lista de tópicos que todos los graduados deben conocer, así como una serie de lineamientos para implementar currículos y un conjunto de cursos propuestos. Los conocimientos y habilidades que un ingeniero de software debería tener al graduarse pueden resumirse en el siguiente listado:

1. Demostrar dominio de un conjunto necesario de conocimientos y habilidades para comenzar su práctica profesional como ingeniero de software.
2. Trabajar tanto individualmente como en equipo para desarrollar y generar software ejecutable.
3. Reconciliar objetivos en conflicto, encontrar compromisos aceptables dentro de limitaciones de costo, tiempo, conocimiento, sistemas pre-existentes y organizaciones.
4. Diseñar soluciones apropiadas en uno o más dominios de aplicación, usando enfoques de ingeniería que integren aspectos éticos, sociales, legales y económicos.
5. Demostrar entendimiento y poder aplicar teorías, modelos y técnicas actuales que provean una base para identificación de problemas y análisis, diseño de software, desarrollo, implementación y verificación.
6. Negociar, trabajar efectivamente, asumir liderazgo cuando sea necesario, y comunicarse bien con los distintos interesados en un ambiente de desarrollo de software típico.
7. Aprender nuevos modelos, técnicas y tecnologías a medida que ellas aparecen y apreciar la necesidad de un desarrollo profesional continuo.

Teniendo en cuenta lo antes mencionado se muestra en la tabla No. 1, la metodología tradicional usada normalmente para enseñar Programación a los estudiantes de Ingeniería de Sistemas:

Tabla 1. Metodología Tradicional (Como se enseña la Programación)
1. Objetos y clases, parámetros.
2. Comprendiendo definición de las clases: definición de las clases, campos,

constructores, métodos.
3. Interacción con los objetos: objetos creando objetos, constructores múltiples.
4. Agrupando objetos en colecciones de tamaños flexibles (ArrayList), colecciones de tamaño fijo (Array).
5. Usando librerías de clases
6. Pruebas y Depuración
7. Diseñando clases: cohesión, duplicación de código, reutilización de código
8. Estructuras con herencia
9. Polimorfismo
10. Clases abstractas e interfaces.
11. Métodos estáticos - el método principal.

B. Programación Extrema XP

Las metodologías ágiles de desarrollo de software se centran en el factor humano, así como también en el producto de software, el cual se basa en pequeñas versiones, dándole más valor al individuo, al trabajo colaborativo y al desarrollo de software en conjunto con el cliente, quien puede monitorear constantemente el producto de software a través de su desarrollo [12]. La programación extrema como tal, es la metodología ágil de desarrollo de software más popular actualmente, ya que busca cambiar el concepto tradicional de desarrollo de software, en el que se centra la atención en establecer todos los requerimientos del software al inicio del proyecto de forma que el mismo no requiera de modificaciones posteriores, por un nuevo enfoque en el que se busca la flexibilidad del software a modificaciones cuando se consideren necesarias y que las mismas no afecten el proyecto. Se muestra a continuación la Tabla No. 2, que contiene las reglas en las cuales se basa la programación extrema [12] y cómo podrían ser aplicadas en nuestra investigación.

Tabla 2. Reglas de XP y su aplicación en la propuesta

1. Equipo completo: Se formarán equipos completos por salones de clases, incluyendo al docente, quien en este caso desempeña las funciones del cliente.
2. Planificación: En este caso el docente es quien planificará los requerimientos y en qué orden se trabajarán y revisará continuamente.
3. Test del cliente: En este caso el docente propone las pruebas para validar si las mini versiones están funcionando.
4. Versiones Pequeñas: Debido al área de aplicación de la propuesta, se trabajarán versiones pequeñas y útiles de software.
5. Diseño simple: Se les solicitará desarrollar las mini versiones de la forma más sencilla posible.
6. Pareja de programadores: Se formarán grupos de trabajo de dos estudiantes por estación de trabajo y se rotarán una vez por cada sesión de laboratorio, de tal forma que el código sea conocido por todos los integrantes del proyecto.
7. Mejora del diseño: La rotación de pares por sesiones de clase, permite que se mejore el diseño del software.
8. Integración continua: Integrar continuamente las mini versiones al proyecto.
9. El código es de todos: Todos los pares serán rotados y trabajarán en las mini versiones de los demás grupos y el código será propiedad de todos los colaboradores del proyecto.
10. Normas de codificación: Mantener un estilo común de desarrollo (por definir).
11. Metáforas: Cada parte del programa será definida por nombres para que sean entendibles por todos los integrantes del equipo.
12. Ritmo sostenible: Se trabajará en todas las sesiones de clases, sin interrupciones.

A continuación se presentan algunos de los trabajos relacionados que han permitido guiarnos hacia nuestra investigación.

La compañía de software Role Model Software ha implementado un modelo alternativo para formar desarrolladores de software denominado "Software Studio" [5] que recuerda en su estructura al taller de oficios medieval, en donde desarrolladores novicios aprenden de maestros desarrolladores. El maestro resuelve los problemas verbalizando en voz alta las estrategias de resolución expertas, mientras los aprendices observan, aprenden y asumen tareas menores, dejando al maestro liberado para asumir las tareas más difíciles. Progresivamente, los aprendices van siendo expuestos a problemas más difíciles, aprendiendo de esta manera junto con sus pares los diversos aspectos de su oficio. Este modelo favorece un aprendizaje acelerado de destrezas expertas a partir del maestro, que mejora el aprendizaje proveniente de aprender de los propios errores. Las destrezas que adquieren los aprendices se basan en los valores, principios y prácticas de XP, las que ofrecen un contexto natural de aprendizaje de todos los aspectos nombrados anteriormente.

Nuestro artículo toma algunas estrategias de este estudio, pero sin entregar a los estudiantes tareas menores y luego exponerlos a problemas más difíciles, en este caso estamos enfocados en seguir una metodología paso a paso con puntos específicos que hacen que el estudiante se interese más por aprender y desarrollar software.

Fred Brooks en su famoso artículo *"No Silver Bullet"* [6] que corresponde en el lenguaje expresado como la complejidad accidental de un software:

- ✓ **Comunicación**: Un ingeniero de software debe dedicar gran parte de su tiempo comunicándose con diversas tipos de interlocutores, debiendo ser capaz de adaptar el grado de abstracción y de tecnicismos en sus conversaciones de acuerdo al lenguaje que domine su contraparte, y además debe ser capaz de escuchar e interpretar correctamente las necesidades expresadas por ellos, necesidades que muchas veces impactarán el diseño e implementación del sistema de software a desarrollar.
- ✓ **Habilidad de trabajar en equipo:** Toda ingeniería de software es ejecutada en equipos, sin embargo esto es algo sobre lo cual los alumnos no reciben entrenamiento. Los ingenieros de software por lo general se destacan por características individuales, es por eso que XP (Programación a Pares) lleva a la persona a formarse en equipo, esto hace que las personas que se acostumbran a realizar acciones individuales ("heroicas") no solo sean capaces de generar los productos (documentos, diseños, módulos) de lo que es responsable sino también poder encontrar un lugar dentro de un equipo de trabajo.

Instituto Técnico de Lund, en Suecia [7] donde se realizaron dos ramos, uno a comienzos de la carrera, justo después de los ramos de programación básica, en el que se introduce el trabajo en equipo mediante XP, y otro curso posterior de profundización, orientado a la formación de *coaches*, cuya parte práctica es justamente dirigir a los equipos del primer curso. En dichos ramos, no sólo se enseñan las prácticas de XP "canónicas" (definidas originalmente por Kent Beck), sino muchas otras prácticas que han sido propuestas por la comunidad de desarrolladores que ha adoptado XP, y que complementan las propuestas por Beck [8].

Preconceptos de los alumnos versus XP

1. *A los alumnos no les agrada el trabajo adicional implicado por las metodologías tradicionales:* Los alumnos suelen ser reticentes a procesos que exigen demasiado esfuerzo en producir artefactos intermedios tales como documentación, especificaciones y diseños detallados, etc., por lo que el énfasis de XP en producir código funcional por sobre generar documentación lo hace más atractivo a los alumnos [9].
2. *Los alumnos tienden a auto-presionarse para lograr la mayor cantidad de funcionalidades, a pesar de que no se les exija:* En el caso del Instituto Técnico de Lund, los alumnos se rigen por la regla del "tiempo fijo", es decir que los alumnos sólo tienen que disponer del tiempo pre-determinado definido para realizar los proyectos del curso, pudiendo este tiempo ser destinado tanto a producir código como a aprender de errores cometidos, en lo que constituye una excelente interpretación de la práctica "40 horas a la semana".

3. *Los alumnos son optimistas sobre el funcionamiento de su código:* Este fenómeno, indica que los alumnos creen que su código funcionará correctamente sin la necesidad de programar pruebas. Esto tiene dos explicaciones: creer que la revisión interactiva resultante de la depuración es suficiente para validar los casos relevantes, y a la ignorancia acerca de los problemas que la evaluación del código puede provocar.
4. *Los alumnos prefieren trabajar en pasos grandes:* La noción de trabajar en pasos pequeños resulta contra intuitiva para muchos, debido a que se cree que si se tiene diseñada una solución completa, lo anterior es una pérdida de tiempo. Este preconcepto se basa en la idea de que el diseño está 100% correcto (lo que nunca es así), y que una vez definido un diseño, éste no variará.
5. *Incomodidad ante cambios de requerimientos:* Los alumnos suelen actuar con incomodidad ante los cambios de requerimientos naturales en todo proyecto de software, muchas veces pensando que esto implicará más trabajo, ignorando por ende el modelo de gestión ágil de alcance variable.

En [9]" los autores proponen desarrollar software por medio de la programación, usando la metodología ágil XP y de esta manera haya un aprendizaje; rápido, eficaz y con un conocimiento aplicado en la vida cotidiana.

Este trabajo apoya nuestro estudio, ya que demuestra la necesidad que hay en los estudiantes por una metodología diferente, más didáctica que comprometa al estudiante a producir con eficiencia y por sobre todo a concientizar la necesidad de trabajar en equipo.

III. OBJETIVO GENERAL

Proponer una metodología de enseñanza utilizando XP (Extreme Programming), para estudiantes de programación de la FUP.

IV. OBJETIVOS ESPECÍFICOS

Realizar un diagnóstico de las capacidades de programación de los estudiantes.

Implementar la metodología ágil XP vs Metodología tradicional con los estudiantes de la Fundación Universitaria de Popayán.

V. PROPUESTA

En la propuesta se desarrolló un estudio que contenía la siguiente información:

Primero se hizo un diagnóstico, luego se presenta la metodología usada y por último una encuesta de satisfacción.

Diagnóstico

Para realizar el diagnóstico se realizó la aplicación de una encuesta, diseñada con un cuestionario de 37 preguntas cerradas con selección de única respuesta, la cual fue aplicada en formato físico y se evaluaban los conceptos básicos de programación java, bases de datos y UML, las cuales contenían preguntas básicas acerca del conocimiento adquirido de semestres anteriores.

Se realizó a los estudiantes de la Fundación Universitaria de Popayán, del programa de Ingeniería de Sistemas VIII semestre diurno y nocturno. La encuesta produjo un total de 20 alumnos encuestados siendo esta una muestra representativa de estos estudiantes.

Fórmula Estadística

Para calcular los resultados obtenidos en cada una de las encuestas se usó la siguiente fórmula:

$$v = (x*y)/z$$

x: *Total de respuestas acertadas*
y: *Porcentaje*
z: *Total de preguntas (cada una de las encuestas)*

- ***Total de respuestas*** sale de la suma total de cada una de las respuestas acertadas.
- ***Porcentaje*** siempre será el 100% en cada encuesta.
- ***Total de preguntas*** sale de multiplicar el total de ellas por el total de las encuestas realizadas.

a) Encuesta Bases de Datos:

Respuestas acertadas	50,4166667
Respuestas erróneas	49,5833333

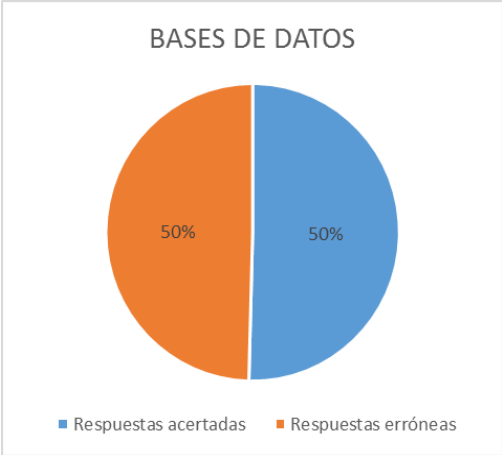

Ilustración 1: Resultados Obtenidos Bases de Datos

De acuerdo a los resultados arrojados que otorga esta grafica donde se evalúa conocimientos de bases de datos, se logra observar que todos los indicadores evaluados se encuentran en un 50% tanto para preguntas acertadas como las erróneas.

b) Encuesta Programación Java:

Respuestas acertadas	59,25
Respuestas erróneas	41

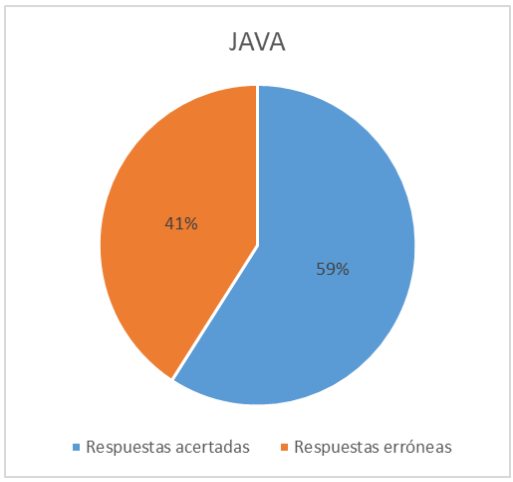

Ilustración 2: Resultados Obtenidos Java

En relación a la encuesta de programacion java se presentó un nivel de acertacion del 59% con respecto a las preguntas erroneas evaluadas con un 41%.

c) Encuesta UML:

Respuestas acertadas		**56**
Respuestas erróneas		44

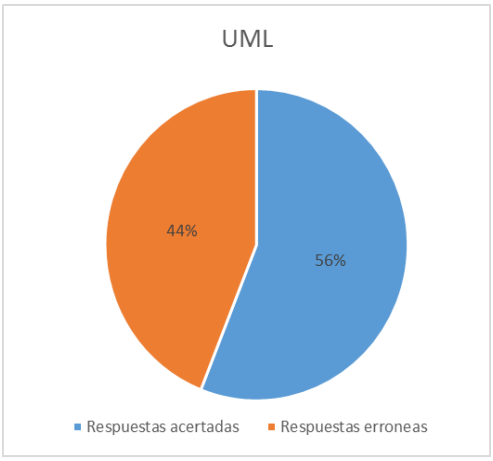

Ilustración 3: Resultados Obtenidos UML

Se observa que el 56% de los participantes contestaron la encuesta de evaluacion de conocimientos basicos UML con un nivel medio de acertacion.

d) Encuesta General:

Total Respuestas	**100%**	**Total Preguntas**
414	100	740
327	100	740

En general acertadas	55,94594595
En general erróneas	44,18918919

Ilustración 4: Resultados Obtenidos Encusta General

En general los indicadores de preguntas acertadas de la metodología aplicada se encuentran sobre el 56%, se observa que el conocimiento que los estudiantes han adquirido en sus estudios universitarios presenta bajo nivel de aceptación.

Hipótesis de la investigación

Las hipótesis que se busca poner a prueba en esta investigación son:

- **Es posible la reproducción efectiva de un ambiente de desarrollo ágil (XP) en las clases de ingeniería de sistemas de la FUP:** Se puede ofrecer a los alumnos de ingeniería de sistemas de la FUP una experiencia representativa de las metodologías ágiles - en particular Extreme Programming – dentro del marco, recursos y tiempo de un semestre de la carrera.
- **La exposición de los alumnos a un desarrollo auténtico en un ambiente ágil genera buenos aprendizajes sobre las metodologías ágiles:** Gracias a la naturaleza de los ambientes de desarrollo ágil, que están orientados a potenciar el aprendizaje continuo en sus integrantes para así lograr la mayor productividad, al reproducirlos fielmente en el contexto de un curso de Ingeniería de Sistemas se obtienen buenos resultados en los aprendizajes de los alumnos.

A continuación se presenta la propuesta elaborada a partir de las hipótesis anteriores.

Tabla No 3

Hipótesis	Propuesta
Es posible la reproducción efectiva de un ambiente de desarrollo ágil (XP) en las clases de ingeniería de sistemas de la FUP.	Un diseño instruccional que configura una clase de Ingeniería de Sistemas en un ambiente de trabajo representativo de XP
La exposición de los alumnos a un desarrollo auténtico en un ambiente ágil genera buenos aprendizajes sobre las metodologías ágiles	Una conceptualización de como XP organiza en ambiente de desarrollo de software para lograr armonizar los diversos componentes de éste y lograr así una construcción continua de conocimiento y valor. Un modelo evaluativo basado en la conceptualización anterior, que servirá para medir los aprendizajes de los alumnos a los que se les aplica el modelo instruccional nombrado anteriormente.

Imagen 1

Sistema de evaluación

La evaluación que se propone para llevar a cabo la metodología XP está enfocada para ser aplicada durante el semestre de clases, tiene 3 componentes y está divido por 3 roles con su respectiva planificación que podría mejorar la calidad de la enseñanza en los estudiantes de la FUP.

Componentes:

Evaluación de resultado (Ponderación: 30%):

Como todo curso de desarrollo software, no debe omitirse la meta de lograr un software satisfactorio en los tiempos y plazos estipulados. Se basa en la calidad del producto de software desarrollado, y considera en ella la opinión del propio cliente.

Evaluación de proceso (Ponderación: 40%):

Esta evaluación considera los siguientes aspectos:

- Cumplimiento de tareas: puntualidad y cumplimiento requisitos estipulados por el profesor
- Co-evaluación: Evaluación recibida por el alumno de sus pares, de acuerdo a su dedicación, responsabilidad y capacidad de trabajo en equipo.

Evaluación experimental (Ponderación: 30%):

En forma grupal e individual los alumnos deberán reflexionar y evaluar la metodología aprendida, considerando:

- Aplicación: el grado de uso y la dificultad relativa experimentada en cada práctica.
- Utilidad: qué tan útiles fueron las prácticas enseñadas.

Para aprobar el curso, el alumno debe lograr una calificación superior o igual a 4.0 en los tres componentes especificados. En caso de que el alumno no logre el 4 en alguno de éstos tópicos, deberá realizar un trabajo adicional.

Roles:

Apreciando la ilustración 6 podemos observar los roles que se plantea para llevar a cabo la metodología XP, en donde participan los estudiantes , el profesor que a su vez sería el cliente el cual tiene el conocimiento del problema y es quien explica, orienta el uso de las prácticas, principios y valores de la metodología.

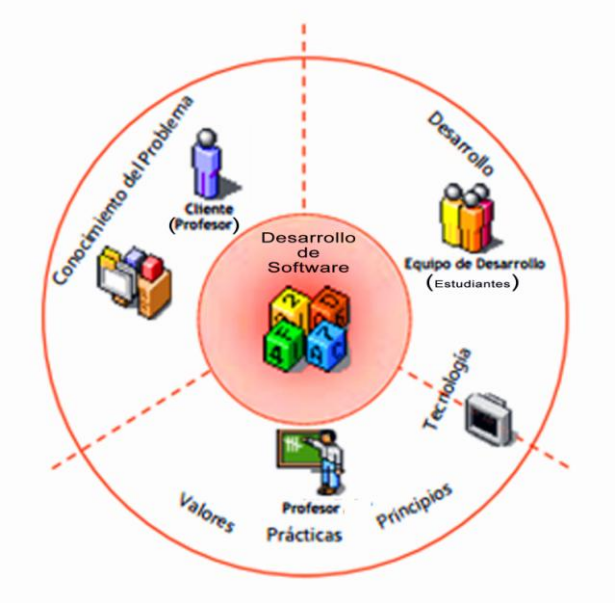

Ilustración 5: Roles

Planificación del curso:

El formato del curso es de sesiones de 4 horas de trabajo presencial a la semana y 8 horas de trabajo de investigación autónoma. Se dispone de un profesor y los estudiantes previamente matriculados académicamente y el trabajo se dividirá en grupos de 2 personas.

Tal como se puede apreciar en la Tabla No 8 el curso está estructurado en dos grandes secciones, las que serán descritas a continuación.

Fase Teórica (3 a 4 sesiones)

En esta fase todo el curso colabora para explorar en detalle en qué consiste XP. La primera sesión es dictada por el profesor, quien presenta los fundamentos que dieron origen a las metodologías ágiles y a XP en particular, haciendo una breve revisión del estado del arte de la Ingeniería de software y enfatizando las razones que motivan el surgimiento de esta nueva mirada al desarrollo de software.

El involucramiento temprano de los alumnos se logra por los siguientes medios:

- Deben discutir en grupo y luego presentar sus conocimientos previos sobre XP
- Se realiza un "extreme hour" o una simulación para poder entender de manera más simple cómo funciona el modelo de gestión de XP.
- Los alumnos realizan debates sobre prácticas de XP en las sesiones posteriores, de preferencia escogiendo ellos mismos los temas presentados

Tabla No 4

Semana		Actividades	
		En horas de clase	Fuera de clase
Teórico	0	Introducción al curso	Informe sobre conocimientos previos
	1	Presentaciones iniciales	Informes de opinión y dudas sugeridas a partir de lo expuesto en clase
	2		
Practico	3	Desarrollo del proyecto usando XP	Generación de ensayo de XP y su aplicación Investigaciones con soluciones específicas con respecto al desarrollo o a la metodología. Generación de informe del desarrollo software y la aplicación XP en el sistema de información.
	4		
	5		
	6		
	7		
	8		
	9		
	10		
	11		
	12		
	13		
	14		
Posterior		Presentación final – grupal	Generación de informe sobre la aplicación de XP en el proyecto y su proyección futura

Fase Práctica

En esta fase se comienza a practicar la versión "miniaturizada" de XP, que restringe el desarrollo a la duración de la sesión (4 horas). La sesión inicial parte con un "Planning game", en donde se define el alcance de la iteración.

A partir de entonces, cada sesión tiene una estructura de manera similar, tal como se presenta en la Tabla 5:

Momento	Descripción
Preparación del ambiente de trabajo.	Consiste en convertir la sala de clases del curso en un ambiente propicio de trabajo. En particular: Instalación de equipos (los necesarios para dictar la clase) con conectividad a red. Disposición de puestos de trabajo: se

	cambia la disposición de mesas y sillas para facilitar la programación de a pares y la rotación dentro del equipo. Los alumnos disponen en el tablero de la sala los las historias de usuario que deben completar
Stand-up meeting	Tal como lo indica la práctica de XP, el equipo se reúne de pie y se pone al día acerca de cuál es el estado de avance. Luego se coordinan y distribuyen las metas de la sesión.
Desarrollo	Este es el momento del trabajo de desarrollo en sí. La dinámica del trabajo sucede tal como en XP: cada desarrollador se ha anotado para el desarrollo de alguna tarea específica, para lo cual solicita la ayuda de algún compañero para trabajar de a pares. En caso de dudas sobre la historia de usuario o dudas metodológicas deben consultarle al profesor. También se le van presentando al profesor las historias de usuario para su aprobación de acuerdo a las pruebas funcionales anteriormente definidas con él en el planning game. Respetando el principio de adaptabilidad al cambio, el profesor puede en cualquier momento cambiar historias de usuario. Con respecto a los descansos, durante la fase teórica, es el profesor quien define un receso en la mitad de la sesión, en cambio en esta fase se motiva a los alumnos que tomen un descanso (razonable) cada vez que se han obtenido logros, o como una manera de despejarse en el caso de llevar demasiado tiempo empantanados en un problema.
Reflexión final y asignación de tareas para la semana	Al ir finalizando el tiempo de la sesión, el profesor motiva el cierre del trabajo, con el objeto de que el curso en conjunto pueda evaluar lo aprendido durante la sesión. Esto se realiza en un diálogo abierto moderado por el profesor. Fruto de la reflexión, el profesor en conjunto con cada equipo define las tareas de investigación para la semana. En el caso de los alumnos que faltaron a la sesión, se definen las tareas de desarrollo con las que deberán devolver las horas de trabajo que sus compañeros ya aportaron.
Sesión y vuelta a la normalidad de la sala	El ambiente de trabajo configurado en la sala es retirado: las mesas, sillas y los computadores son devueltos a su posición normal, los tableros totalmente limpios y la información es guardada por los alumnos para poderla utilizar la clase siguiente.

Imagen 2

Encuesta de satisfacción de la Hipótesis

Se presentan los resultados obtenidos de acuerdo a la hipótesis que se propuso en nuestra investigación con los estudiantes de la Fundación Universitaria de Popayán, VIII Semestre de Ingeniería de Sistemas.

Tabla No 6	
Excelente	98
Aceptable	57
Malas	5
TOTAL PREGUNTAS	160

Ilustración 5: Resultados Obtenidos Encuesta Satisfacción Hipótesis

De las 160 preguntas en total, el 98 de aquellas preguntas respondidas por los estudiantes de la Fundación Universitaria de Popayán aprueban la encuesta y la hipótesis de la propuesta.

VI. CONCLUSIÓN

Tal como se indicó al inicio de esta investigación, este artículo fue pensado para reflejar de la manera más fiel posible un ambiente de enseñanza en la programación mucho más ágil, y en particular, de XP. La observación informal de las encuestas del artículo mostró que los alumnos agradecían la experiencia vivida, y que se obtenían resultados bastante positivos de los alumnos que participaban de él. Todo lo anterior dio pie a la propuesta de esta investigación, que está descrita en el capítulo V.

REFERENCIAS

[1] J. Kuljis, "Orienting the Teaching of an Introductory Object-Oriented Programming to Meet the Learning Objective," presented at the 26th Int. Conf. Information Techonlogy Interfaces ITI 2004, Cavtat, Croatia, June 7-10, 2004.

[2] Aubin, Verónica Blautzik, Leonardo Dejan, Gustavo Mejoras en el Proceso de enseñanza-aprendizaje de programación utilizando metodologías de la industria del software. No.1. 2011. pp.1.

[3] Gabriela Salazar Bermúdez, Desafíos del Curso de Ingeniería de Software Revista Educación en Ingeniería ISSN 1900-8260 Enero a Junio de 2012, Vol. 7, N°. 13, pp. 32-43 • © 2012 ACOFI.

[4] Talbott, N., Auer, K., "Apprenticeship in a Software Studio: An Old and New Model for Education", Educator's Symposium, OOPSLA 2000, Minneapolis, Minnesota, USA, October 15-19, 2000. pp. 49-51.Brooks, Fred. "No silver bullet: Essence and Accidents of Software Engineering", Computer, Vol. 20, Nº 4 (April 1987): 10-19.

[5] Hedin, G., Bendix, L., Magnusson, B., "Teaching Software Development using Extreme Programming" en el "Book of Scandinavian Pedagogy of Programming Network" en http://www.spop.dk/chap-v3/SPOP-Lund-051104.pdf, pp. 586-593, 2003

[6] Hedin, G., Bendix, L., Magnusson, B., "Introducing Software Engineering by means of Extreme Programming" Proceedings of the 25th International Conference on Software Engineering, pp. 586- 593, 2003.

[7] Williams, L., Upchurch, R. "Extreme Programming For Software Engineering Education?" 31st Annual Frontiers in Education Conference, 2001.

[8] Alfonso and Botia, An Iterative and Agile Process Model for Teaching Software Engineering, Proceedings of the 18th Conference on Software Engineering Education & Training, (CSEET'05), Abril 2005. pp. 9-16

[9] Kent Beck Planning Extreme programming http://www.onlinetechbooks.com/software-engineering books/Addison-Wesley-Planning-Extreme Programming.pdf, October 12, 2000, pp.160.

[10] Patricio Letelier y Mª Carmen Penadés, Métodologías ágiles para el desarrollo de software: eXtreme Programming (XP), Universidad Politécnica de Valencia (UPV). No.1. 12 Nov 2003. pp.7.

[11] P. L. Ph.D. and M. C. P. Ph.D., "Metodologías ágiles para el desarrollo de software: eXtreme Programming (XP)." 12 de Noviembre de 2003 pp. 1-59.

[12] S. Xinogalos, et al., "Re-designing an OOP course based on BlueJ," presented at the Seventh International Conference on Advanced Learning Technologies (ICALT'07), 2007.

Artículo

Enfoque Ágil XP Aplicada a la Educación de desarrollo de Software

Trabajo de Grado para obtener el Título de Ingeniería de Sistemas

Estudiantes:

Ing. Ana María Polindara
Ing. Cesar Augusto Arbeláez
Ing. Fabián Astudillo Rebolledo

Asesores de Trabajo de Grado:

Ing. José Armando Ordoñez
Ing. Luis Freddy Muñoz

Seminario: Seminario de Investigación

Programa: Ingeniería de Sistemas

Popayán 2014