Propuesta Uso de Metodologías Formales Combinadas con Metodologías Agiles para el Desarrollo de Software

[1]F. J. Lomas, [2]G. Raura y [2]M. Campaña
[1]*Kruger Corporation* S.A.
[2]*Departamento de Ciencias de la Computación, Escuela Politécnica del Ejército, Sangolquí, Ecuador*
xlomas@kruger.com.ec, georaura@gmail.com, mcampana@espe.edu.ec

RESUMEN: Una ventaja competitiva dentro de la industria del software es la velocidad con la cual se puede innovar y crear productos para distribuirlos al mercado. El presente artículo propone la combinación de metodologías formales, como el Proceso Unificado de Desarrollo, y Metodologías Agiles de Desarrollo de Software, como la Programación Extrema y sus técnicas, como métodos para mejorar la velocidad de construcción del software sin dejar de lado la calidad del mismo. Esto se consigue usando los artefactos de las metodologías formales en la etapa de toma de análisis del problema y toma de requerimientos, para en el resto de fases del desarrollo proceder a usar los artefactos y técnicas de las metodologías agiles. Para llevarlo a cabo, se modificó el documento de caso de uso incluyendo información grafica detallada en forma de un flujo de actividades para describir y delimitar el proceso contenido en el caso de uso. Así mismo, se generan casos de evaluación para ser verificados con pruebas unitarias extendidas para cubrir el alcance de cada caso. Los resultados obtenidos muestran una mejora en la velocidad de comprensión de los requerimientos por parte del equipo técnico. Como consecuencia de esto, el tiempo que toma la verificación y corrección de la funcionalidad es reducido entre el 15% y 35%, además de que los defectos del producto descubiertos cuando ya es liberado al mercado se reducen entre un 10% y un 20%.

Palabras clave: Proceso unificado de Desarrollo, Metodologías Agiles, Programación Extrema

ABSTRACT: Inside the software industry a competitive advantage is the speed on the innovation and creation process oriented to put products on the market. The present article propose the combination of formal software development methodologies, like the Unified Process, and agile methodologies, like Extreme Programming and it's techniques, as a method to speed up the software develop time without leaving the quality behind, this can be done using the formal methodologies artifacts in the requirements gathering and analysis phase, for the next phases we proceed to use the artifacts and techniques of the agile methodologies. To accomplish the objective the use case document was modified to include graphic information like a flow chart that represents the different activities of the process, the flow chart helps on the description and delimitation of the process. Also the test cases are generated to verify the use cases with extended unit tests to cover the use case scope. The results show a improve in the way and speed on understanding of the requirements inside the technical team, an secondary result is the reduced time to verify and fix the functional errors, the improvement is measured in the range of 15% and 35%, also the released to market product defects where reduced in the range of 10% and 20%.

Keywords: Agile Methodologies, Extreme Programming, The Rational Unified Process.

1. INTRODUCCIÓN

En la creciente industria del desarrollo de software, cuyas ventas se han incrementado en un 22.8% [1], que se enfrenta a un mundo completamente conectado y que brinda la suficiente información para que cualquier persona pueda plasmar sus ideas en programas de computadora simples o complejos sistemas de software, la innovación y velocidad al crear nuevos productos es la mejor ventaja competitiva que se puede tener [2]. La construcción de software a pesar de ser un tópico reciente ha evolucionado con gran velocidad en los últimos años, hoy se conocen ya metodologías que tienen varios años siendo usadas como el Proceso Unificado de Desarrollo que propone estructurar dicho proceso en varias fases, así como disciplinas que ejecutan las diferentes tareas basados en artefactos que pueden ser documentos o productos de software propios o de terceros. Sin embargo este tipo de metodologías que son más estructuradas y maduras requieren también de un mayor esfuerzo para ser aplicadas correctamente, este esfuerzo se ve traducido en un incremento los recursos que se usan en el proyecto como tiempo y dinero.

La calidad del software producido es uno de los factores que ha empujado el surgimiento de las distintas metodologías de desarrollo de software, dado que el software es incluido cada vez más en industrias que requieren de alta fiabilidad como la medicina y el automovilismo. Sin embargo el proceso de verificación de calidad debe ser riguroso, ya que el costo de cada falla encontrada esta dado por el costo de el trabajo que toma repararlo [3], por lo tanto menos fallas se reflejan en un menor costo. Al tener la necesidad de producir software de calidad en un tiempo corto se nos presenta el reto de lograr este objetivo con las herramientas maduras que son las metodologías formales de desarrollo de software, y además se tiene la posibilidad de usar nuevas herramientas como las técnicas de la programación extrema [4], que han probado ser una alternativa válida y sencilla de usar [5], se propone combinar a conveniencia el uso de los artefactos de las metodologías formales con el uso de las técnicas de las metodologías agiles para acortar el tiempo empleado para sacar el producto al mercado. En general el propósito es generar un ahorro en tiempo y recursos al ejecutar un proyecto de desarrollo de software combinando los artefactos y las técnicas de las metodologías de desarrollo de software formales con las metodologías de desarrollo agiles.

El resto del artículo ha sido organizado como sigue: La sección 2 muestra los artefactos del Proceso Unificado a ser modificados y las técnicas de programación extrema a ser usados. La sección 3 detalla las modificaciones y forma de empleo de las técnicas de la sección 2. En la sección 4 se muestran los resultados obtenidos. En la sección 5, se analizan algunos trabajos relacionados. Finalmente, en la sección 6, se presentan las conclusiones y líneas de trabajo futuro sobre la base de los resultados obtenidos.

2. MATERIALES, ARTEFACTOS Y TÉCNICAS A SER USADOS Y/O MODIFICADOS

Caso de Uso: El documento de caso de uso tiene por objetivo organizar la información de uno o más requerimientos para que quien construye el sistema pueda guiarse estructuradamente, además sirve como un documento que debe ser presentado a los interesados con fines de validación de los requerimientos. Al ser este artefacto de comunicación entre el equipo de desarrollo y los interesados en el desarrollo se entiende la importancia del mismo.

Caso de Prueba: Los casos de prueba están derivados de los casos de uso, en resumen un caso de prueba es un plan estructurado de cómo se debe verificar la funcionalidad construida. En un caso de prueba se debe tomar los subprocesos del proceso principal automatizado para ser verificados.

Pruebas Unitarias: Las pruebas unitarias son una técnica de la Programación Extrema, que propone generar código que prueba la funcionalidad de ciertas piezas del sistema en manera aislada, los resultados se basan en la comparación del resultado obtenido con el resultado esperado.

3. DISEÑO E IMPLEMENTACIÓN

Este es un ejemplo del caso de uso modificado:
Especificación CU01: Evaluación y selección de Aspirantes

Descripción: La presente especificación tiene por objetivo mostrar la propuesta de automatización del proceso de la Evaluación y selección de aspirantes de la Carrera de Ingeniería en Informática en Sistemas en un nivel macro.
Precondiciones: Que existan aspirantes a la Carrera de Ingeniería en Informática y Sistemas. Que exista un modelo de competencias definido para la Carrera, incluyendo las evaluaciones correspondientes.
Post condiciones: Existirán evaluaciones realizadas por los Aspirantes de la Carrera y calificadas en forma automática. Existirán informes sobre los Aspirantes que cubran los requerimientos mínimos definidos en el modelo de competencias de la Carrera.

I. **Actores**

i. Actores Principales

TABLA 1: Listado de actores que participan en el caso de uso, se detalla cómo se lo conoce dentro del caso de uso y su rol dentro del proceso que se está describiendo en el documento de caso de uso.

Actor	*Descripción*
Aspirante	*Persona nacional o extranjera que ha culminado sus estudios secundarios y que cumple con los requisitos legales de la República del Ecuador establecidos por los organismos competentes para acceder a la Educación de Tercer Nivel.*
Administrador de la Solución	*Persona encargada de ingresar los datos básicos necesarios y vigilar el funcionamiento de la Solución de Evaluación y Selección de Aspirantes para que pueda cumplir su objetivo.*
Solución de Evaluación y Selección de Aspirantes (E-Recruit)	*Sistema informático que ayuda con el procesamiento de los datos de los Aspirantes a la Carrera en el proceso de selección y evaluación, este automatiza el proceso de recolección de datos, calificación de evaluaciones basado en el modelo de competencias de la Carrera y generación de informes del resultado del proceso.*

ii. Actores Secundarios: *N/A*

II. Flujo de Eventos

i. Flujo Base

a. *El Administrador del Sistema ingresa los datos del modelo de competencias de la Carrera y define los niveles mínimos.*
b. *El Administrador del Sistema ingresa las posiciones o cupos disponibles para el siguiente periodo.*
c. *El Administrador define las evaluaciones y su correspondencia con el modelo de competencias ingresado para la Carrera.*

d. El Aspirante accede al sistema e ingresa sus datos básicos.
e. El aspirante procede a rendir las evaluaciones.
f. El sistema procede a calificar las evaluaciones terminadas.
g. El Administrador invoca la generación de informes y notificación de resultados.

ii. Flujo Alternativo: *N/A.*

III. Diagramas

i. Diagramas de Caso de Uso

Este diagrama muestra qué actividades se realizan dentro del caso de uso y su relación entre los actores, también muestra dependencia entre las actividades.

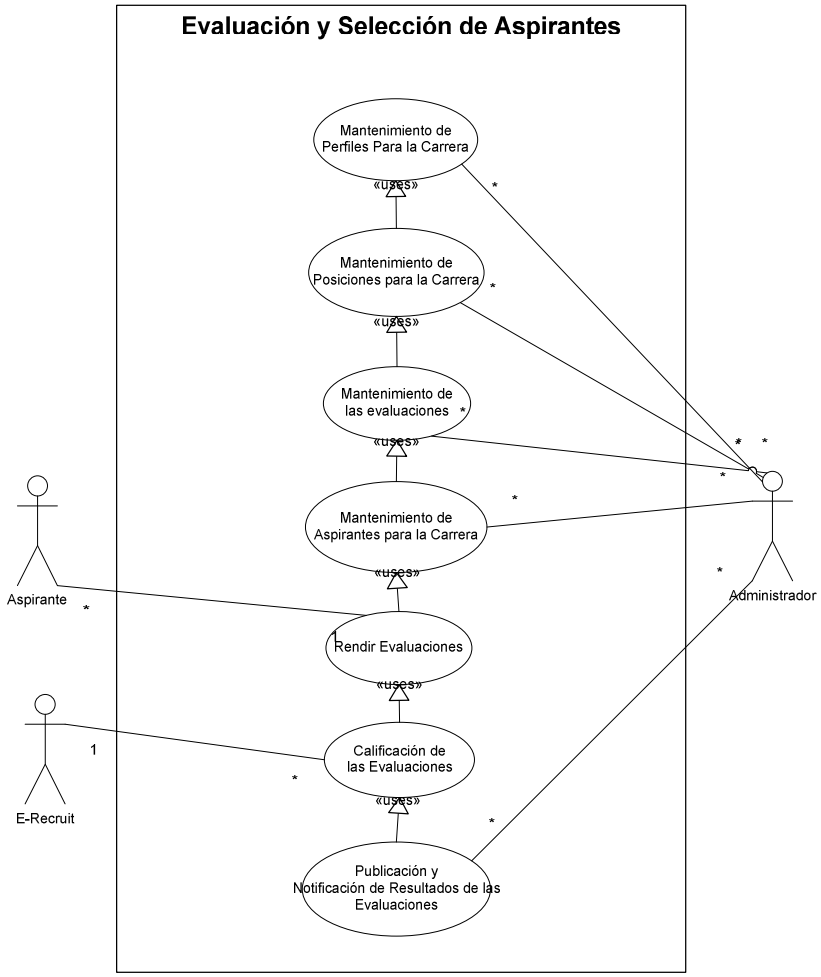

Figura 1: diagrama de caso de uso de evaluación y selección de aspirantes

ii. Diagramas de Flujo

El diagrama de flujo muestra cómo se dan las actividades dentro del proceso, delimitando qué actor está involucrado con qué actividad y a su vez clarifica como debe fluir la información.

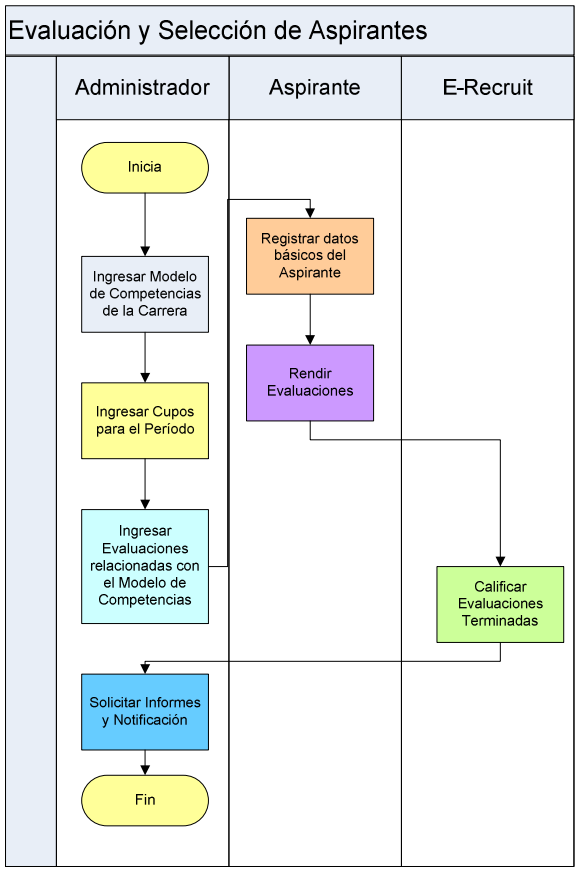

Figura 2: Diagrama de Flujo del Caso de Uso de Evaluación y Selección de Aspirantes

IV. Interfaz Gráfica de Usuario

Este proceso tendrá una Interfaz Gráfica de Usuario basada en un explorador de Internet, se definirán pantallas de acuerdo a la necesidad para satisfacer los requerimientos del usuario.

V. Relaciones

i. Interfaces con otros sistemas: *La presente solución no plantea ninguna interfaz con otro sistema.*

VI. Inclusiones: *N/A.*

VII. Exclusiones: *N/A.*

VIII. Suposiciones y Dependencias

i. Suposiciones: *Se cuenta con toda la información del modelo de competencias para la Carrera.*

IX. Dudas: *N/A.*

X. Observaciones

La solución procesará el grupo de evaluaciones basada en los parámetros ingresados en el modelo de competencias y la definición de perfiles, si estos datos no están verificados pueden ocasionar contratiempos dentro del proceso.

XI. Requerimientos Especiales: *N/A.*

Nótese en la sección de *Diagramas* en la Fig. 2 se encuentra un gráfico que normalmente no es incluido en un caso de uso, esta figura refleja el flujo de eventos y de información que se describe en la sección *Flujo de Eventos*, este diagrama puede ser generado en cualquier herramienta, tomando en cuenta que mientras

más descriptivo es el mismo mejor serán los resultados obtenidos a posterior. A continuación un caso de prueba de ejemplo:

CP03: Revisión y publicación de resultados

Descripción: *En este caso de prueba se pretende verificar las notificaciones a los aspirantes que han sido considerados idóneos para cursar la carrera.*

Precondiciones: *Evaluaciones ejecutadas*

Escenario (ver Tabla 2)

Como se puede notar en el cuadro que muestra el escenario, el caso de prueba se compone de varias verificaciones en pasos individuales, análogamente las pruebas unitarias verifican funcionalidad de un componente a la vez comparando los resultados obtenidos con resultados esperados teóricamente.

En este caso la propuesta requiere que se generen una prueba de unidad por cada paso del caso de prueba, para posteriormente en una prueba unitaria más grande juntar todas las pruebas unitarias que representan al caso de prueba. Para ejecutar las pruebas se pueden usar otras técnicas de la programación extrema como es la integración continua, o a su vez cuando el software pase a control de calidad estas pruebas pueden ser ejecutadas por las herramientas que se disponga en el ambiente de trabajo. Un ejemplo de las pruebas que se deben generar se lo puede encontrar en el blog *"Algo de .NET"* [6].

También es posible usar la técnica de integración continua, la cual a más de combinar y construir el código del sistema en cuestión puede ejecutar las pruebas unitarias de forma automática con los parámetros que sean necesarios, con esto se consigue que se pruebe la calidad del software incluso antes de que llegue al equipo de control de calidad.

TABLA 2: Aquí se describe el escenario que debe ser probado, que tiene que ser concordante con el caso de uso que pretende cubrir, describiendo claramente los pasos de cómo se debe ejecutar la prueba.

CP02: Revisión y publicación de resultados (CU08)

Paso del Caso de Uso	***Descripción del Paso***	***Resultado Esperado***	***Resultado Obtenido***	***Completo/ Fallo***	***Ambiente***	***Número de Log***
1. El administrador solicita el informe de aspirantes idóneos	*El administrador solicita el reporte de aspirantes idóneos*	*Presencia en el menú de opciones el reporte de aspirantes idóneos*				
2. La solución genera el reporte	*La solución genera el reporte de aspirantes idóneos*	*Reporte generado listo para revisión*				
3. El administrador pública la lista	*Si se considera completa la lista, se puede publicar y generar notificaciones*	*Pantalla que permite publicar la lista final de aspirantes aceptados*				
4. Se generan notificaciones para los aspirantes aceptados	*Se notifica a los aspirantes que han sido aceptados*	*Envío de mails a los aspirantes aceptados, se publica el reporte*				

4. EVALUACION DE RESULTADOS

Para poder medir la efectividad de los artefactos y de los métodos aplicados, se consideró ejecutar dos comparaciones, la primera se ejecuta dentro de un mismo proyecto, y la segunda se compara con otro proyecto de similares proporciones. En el primer proyecto se decidió dividir en 2 equipos, los que usan los artefactos modificados y las técnicas de pruebas, y quienes usan los artefactos y proceso normal, el trabajo fue repartido equitativamente a los dos equipos en función del esfuerzo estimado requerido para que la medición sea más certera. El siguiente cuadro muestra los datos más significativos obtenidos:

TABLA 3: Comparación de los resultados obtenidos entre los equipos A y B que participan en el experimento.

	Equipo A (Artefactos Modificados)	**Equipo B (Desarrollo Tradicional)**	**Variación Porcentual**
Número de Casos de Uso	10	10	0%
Tiempo estimado (Horas Hombre)	640	640	0%
Tiempo real (Horas Hombre)	660	743	11%
Número de Errores Reportados por QA	305	437	30%
Número de Errores Reportados por Usuario Final	65	83	22%

En la Fig. 3, se puede apreciar una comparación de las horas hombre empleadas para ejecutar los trabajos por parte de cada uno de los equipos dentro del proyecto, se puede ver un menor tiempo empleado por el equipo A.

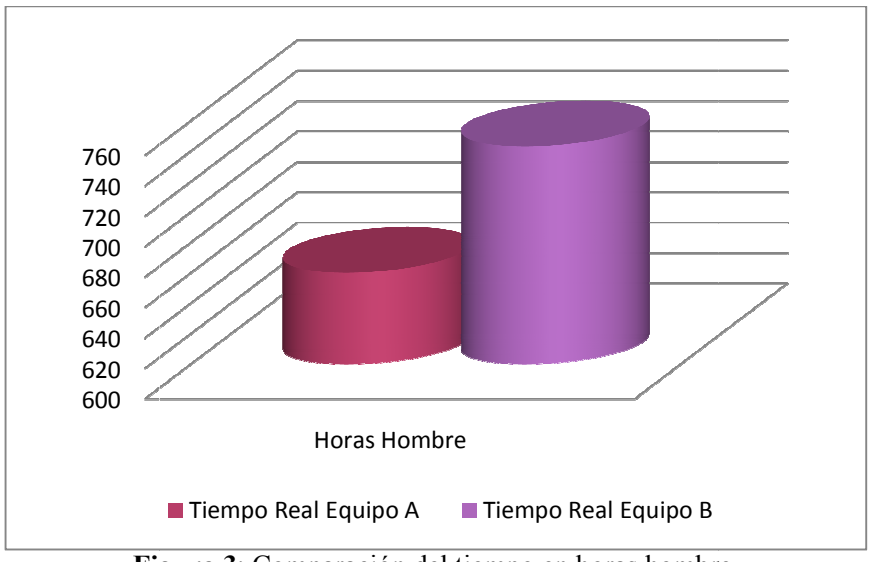

Figura 3: Comparación del tiempo en horas hombre.

En la Fig. 4, se comparan el número de errores reportados por QA y por el usuario final, se nota nuevamente que quienes usan las técnicas propuestas tienen una mejora en sus resultados.

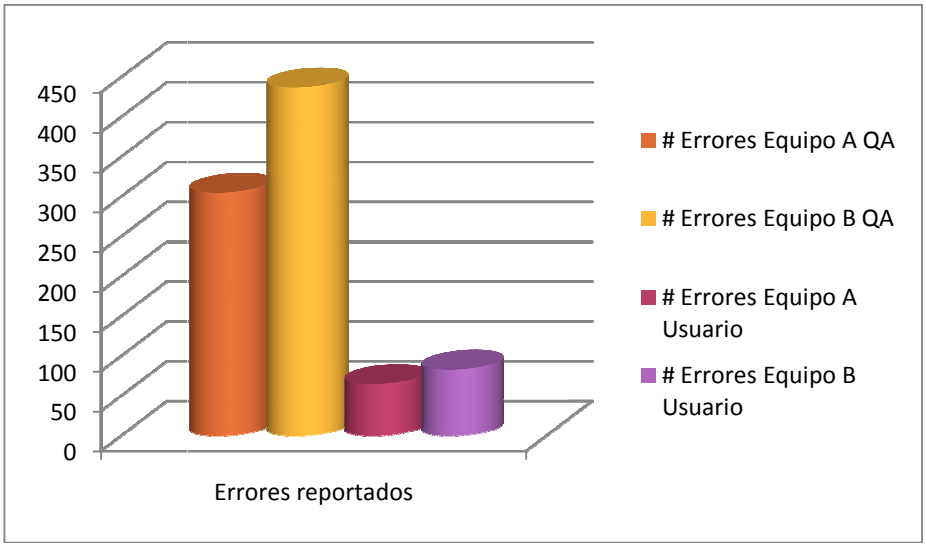

Figura 4: Número de errores reportados para los equipos A y B.

Como se puede observar el tiempo de desarrollo mejoró en un 11%, el número de errores reportados por Control de Calidad fue un 30% menor, y los errores reportados por el usuario final son un 22% menores. En la segunda comparación se tomaron dos proyectos, uno previamente realizado, y uno que se realizo con los artefactos realizados y las técnicas propuestas. Se usaron parámetros como puntos de función para comparar su tamaño así como recursos empleados en el proyecto, basados en esos factores se compararon proyectos de similares tamaños, los datos recogidos son los siguientes:

TABLA 4: Comparación de los resultados obtenidos entre los proyectos A y B, siendo el proyecto A el que usa artefactos modificados y técnicas agiles y el proyecto B desarrollo tradicional.

	Proyecto A (Artefactos Modificados)	**Proyecto B (Desarrollo Tradicional)**	**Variación Porcentual**
Número de Puntos de Función	481	489	2%
Tiempo estimado (Horas Hombre)	2886	2934	2%
Tiempo real (Horas Hombre)	3095	3589	14%
Número de Errores Reportados por QA	367	475	23%
Número de Errores Reportados por Usuario Final	175	231	24%

En la Fig. 5 se muestra una comparación del número de horas hombre que se emplearon en los proyectos A y B, el proyecto A que usa las técnicas propuestas usa menor cantidad de horas hombre.

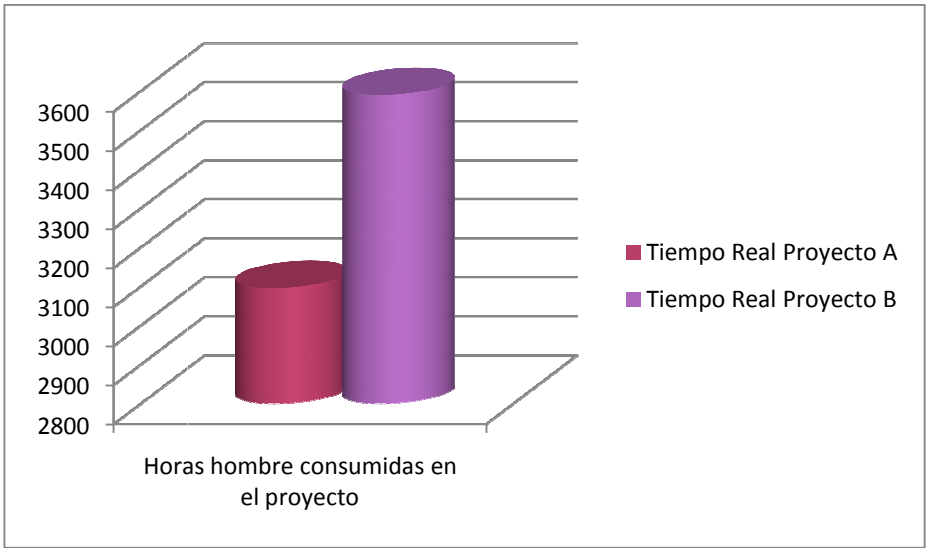

Figura 5: Horas hombre por cada proyecto.

La Fig. 6 muestra la comparación de número de errores reportados para los proyectos A y B, nuevamente se nota una mejora en el proyecto A que usa las técnicas propuestas incluso si se totalizan los errores.

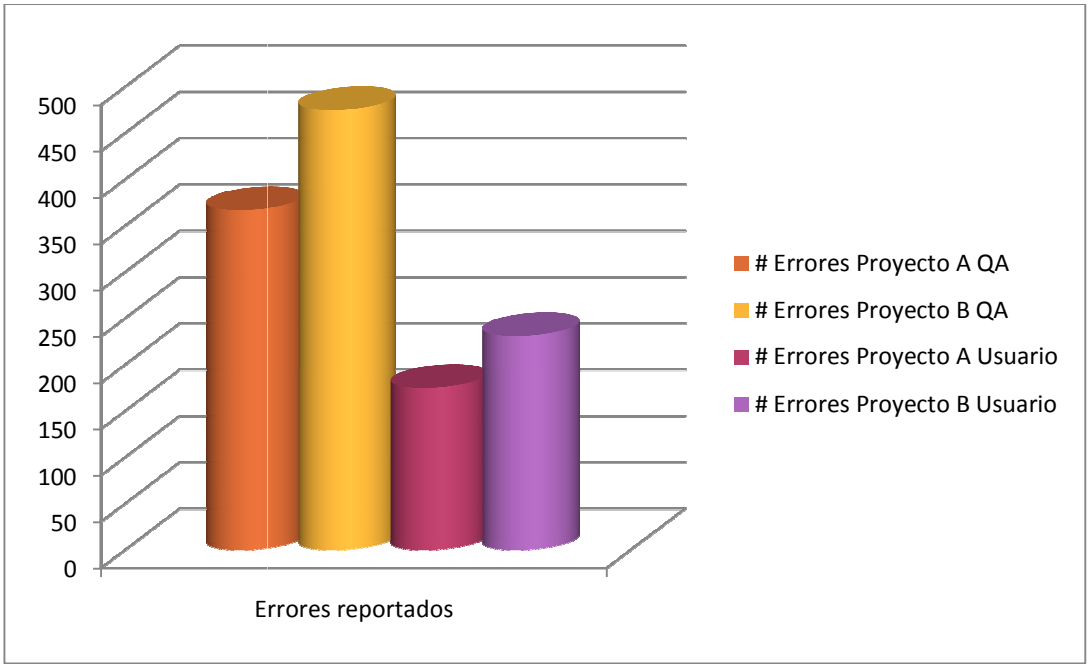

Figura 6: Comparación de errores reportados entre los proyectos A y B

Tomando en cuenta que hay una diferencia del tamaño del proyecto de aproximadamente 2%, los resultados pueden ser interpretados como que necesitan ser ajustados a la misma variación porcentual, el tiempo de desarrollo mejoró un 12%, el número de errores reportados por control de calidad fue menor en un 21%, y el número de errores reportados por el usuario final fue menor en un 23%.

5. DISCUSIÓN

Como se muestran en los cuadros y gráficos anteriores, la modificación de los artefactos y la combinación de técnicas agiles con metodologías formales puede traer beneficios inmediatos en la reducción de uso de recursos durante la vida del proyecto, el medio es mejorar los artefactos de forma que sean más claros y entendibles tanto para el equipo técnico como para los interesados, esto nos asegura que lo que se va a desarrollar es más aproximado a lo que desean los interesados. Un problema que puede presentarse es el alcance y extensión de los casos de uso, según quien analice y diseñe pueden existir un numero alto de casos de uso, esto puede ser contraproducente porque requerimos más recursos para generar las pruebas, así como para ejecutarlas, y claro esto también suma en número de errores, tiempo de manejo de los errores, tiempo de corrección y tiempo de verificación, por lo que es recomendable tener en cuenta si se desea usar las modificaciones y técnicas sugeridas en este articulo se debe también evaluar la granularidad de los casos de uso.

Las limitaciones que existirían por las herramientas de pruebas a ser usadas se pueden resolver con uso de herramientas de código abierto como NUnit y JUnit, siempre es recomendable usar herramientas integradas con el entorno de desarrollo, la falta integración de estas herramientas en el entorno de desarrollo puede suponer un problema de bajo impacto. Otro posible problema es la necesidad de entrenamiento para el equipo técnico con el fin de aprovechar al máximo las técnicas propuestas, por esto es recomendable escoger herramientas de pruebas que son comunes en el mercado además de que en lo posible estas deberían estar orientadas a cualquier tipo de usuario técnico, no solo al personal de control de calidad.

6. TRABAJOS RELACIONADOS

No existe suficiente documentación sobre trabajos relacionados, dado que la modificación de artefactos de metodologías y la personalización de una metodología, como la del proceso unificado de desarrollo, generalmente son consideradas como un trabajo comercial, sin embargo existen publicaciones de libros como *"Building J2EE applications with the rational unified process"* de Peter Eeles, Kelli Houston y Wojtek Kozaczynski, en las cuales se trata el tema de la personalización de la metodología de desarrollo de software para adaptarlo a las necesidades del equipo de desarrollo. Un libro que puede servir también de guía sobre cómo aplicar correctamente el proceso unificado de desarrollo es *"The rational unified process made easy: a practitioner's guide to the RUP"* de Per Kroll y Philippe Kruchten.

Con respecto a la aplicación de las pruebas unitarias con casos de pruebas se tiene más difusión sobre su uso y beneficio. La aplicación puede ser revisada en la referencia [7], donde se muestra una técnica muy parecida a la propuesta en el presente documento, además en la referencia [8] se puede revisar como implementar las pruebas unitarias con un alto detalle. También se habla de la efectividad de estas técnicas, en algunos casos se ha hecho referencia a que los resultados son concluyentes y además un poco exagerados [9].

7. CONCLUSIONES Y TRABAJO FUTURO

Con los datos recopilados se puede concluir que existe una mejora en el tiempo de desarrollo de un proyecto de software modificando los artefactos que se usan y aplicando técnicas no convencionales al desarrollo de software tradicional. Se consiguió una mejora de entendimiento entre los interesados y quienes desarrollan el software volviendo el documento de caso de uso más visual, agregándole un diagrama de flujo que resuma los requerimientos del proceso a ser automatizado. El menor número de errores reportado por el área de control de calidad es una consecuencia directa de ejecutar las pruebas unitarias que reflejan los casos de pruebas como política para que las secciones del proyecto sean pasadas a revisión, desembocando en un menor tiempo de revisión del software así como menor tiempo de reparación de errores de software y otros beneficios indirectos como menor probabilidad de regresión de errores. El menor número de errores reportados también es influenciado por la modificación de los casos de uso, al comprender mejor la funcionalidad quien construye puede ser más efectivo el desarrollador al construir la solución. Si bien los resultados oscilan entre el 10% y 25% de mejora estos números pueden mejorar en una medida moderada confirme a que quienes usan estos artefactos y estas técnicas adquieren mayor experiencia en el uso de los mismos, sin embargo estos resultados también pueden ser afectados por un sinnúmero de otras variables de ambiente que afectan comúnmente a los proyectos de desarrollo de software, es por esto que se deben aplicar estas técnicas recomendadas paso a paso en los ambientes de desarrollo finales, evaluando y ajustando a las necesidades propias los artefactos y las técnicas.

Como trabajo futuro se investigará la generación de código automática, incluyendo pruebas de unidad, basada en los artefactos de documentación creada. También se puede proponer un estudio sobre la aplicación de las técnicas y modificación de los artefactos propuestos sobre proyectos de largo aliento, es decir de duración de más de 6 meses.

Referencias Bibliográficas

[1.] The Entretainment Software Association. The Entretainment Software Association. [En línea] The Entretainment Software Association, 01 de 01 de 2009. [Citado el: 07 de 12 de 2009.] http://www.theesa.com/facts/index.asp.

[2.] The Free Library. Software company a roaring success; INDUSTRY: Innovative product helps H&S firm increase sales by ten-fold. *The Free Library.* [En línea] 13 de 11 de 2009. [Citado el: 7 de 12 de 2009.]

[3.] W. Ward *Calculating the real cost of software defects.*. 1, s.l. : Hewlett-Packard Journal, 1991, Vol. 1.

[4.] Wells, Don. Extreme Programming Rules. *Extreme Programming* . [En línea] 1 de 1 de 1999. [Citado el: 7 de 12 de 2009.] http://www.extremeprogramming.org/rules.html.

[5.] Extreme Programming Research. Extreme Programming Research. [En línea] 22 de 06 de 2002. [Citado el: 7 de 12 de 2009.] http://c2.com/xp/ExtremeProgrammingResearch.html.

[6.] Lomas, Francisco. Ejemplo de Pruebas Unitarias. *Algo de .NET.* [En línea] 01 de 01 de 2009. [Citado el: 8 de 1 de 2010.] http://fcolomas.blogspot.com.

[7.] Exforsys. Unit Testing: Why? What? & How? *Free Training.* [En línea] Exforsys, 01 de 01 de 2009. [Citado el: 14 de 12 de 2009.] http://www.exforsys.com/tutorials/testing/unit-testing.html.

[8.] Laurie Williams, Dright Ho, Ben Smith and Sarah Heckman. Unit Testing in Jazz Using JUnit. *North Carolina State University.* [En línea] North Carolina State University, 28 de 08 de 2008. [Online:] http://agile.csc.ncsu.edu/SEMaterials/tutorials/junit/junit_tutorial_jazz.html.

[9.] Proffitt, Jacob. TDD Proven Effective! Or is it? *TheRuntime.com.* TheRuntime.com, [Online:] http://theruntime.com/blogs/jacob/archive/2008/01/22/tdd-proven-effective-or-is-it.aspx.