CORE
Metadata, citation and similar papers at core.ac.uk
Provided by El Servicio de Difusión de la Creación Intelectual

Metodologías para el desarrollo de software en PYMES

Marcelo Alberto Colombani[1], Martín Mauricio Pérez[1], Marcelo Alejandro Falappa[2]

[1] Facultad de Ciencias de la Administración
Universidad Nacional de Entre Ríos
marcelocolombani@hotmail.com, martinmperez@gmail.com

[2] Departamento de Ciencias e Ingeniería de la Computación
Universidad Nacional del Sur
mfalappa@gmail.com

Abstract. En este trabajo se presenta un estudio de los diferentes modelos de proceso de software, desarrollando sus características, bondades y deficiencias, con el fin de detectar qué metodología se encuadra o adapta mejor al escenario de las Pequeñas y Medianas Empresas productoras de software o empresas que, en su estructura, albergan un equipo de desarrollo no demasiado numeroso. Este trabajo pretende establecer guías para que dichas empresas puedan llevar adelante un proceso de desarrollo organizado, confiable y de calidad, en ambientes de requerimientos confusos, incompletos y cambiantes. Como parte del trabajo se realizó una experiencia práctica proponiendo la utilización de la Metodología XP (*eXtreme Programming*) con ciertas adaptaciones.

Keywords: PYMES, requerimientos, proceso de desarrollo de software, modelos de proceso de software, modelos y guías de certificación.

1 Introducción

El objetivo de este trabajo es detectar qué metodología se encuadra o adapta mejor al escenario de las Pequeñas y Medianas Empresas (PYMES) productoras de software o empresas que, en su estructura, albergan un equipo de desarrollo no demasiado numeroso y puedan llevar adelante un proceso de desarrollo organizado, confiable y de calidad, en ambientes de requerimientos confusos, incompletos y cambiantes.

Si bien existen trabajos relacionados a la utilización de metodologías en PYMES, el presente artículo se enfoca en la necesidad de contemplar diferentes aspectos relacionados a los requerimientos, y la posibilidad de que éstos reflejen las necesidades reales de los usuarios para lograr el producto correcto.

El artículo está organizado en varias secciones. La Sección 2 pone en contexto la problemática, la Sección 3 detalla las diferentes metodologías analizadas, desarrollando un ejemplo de la metodología XP y puntualizando cuestiones que se deben tener en cuenta al aplicar dicha metodología. La Sección 4 presenta los resultados obtenidos y la Sección 5 presenta las conclusiones y el trabajo futuro.

2 Situación Problemática

La búsqueda de técnicas que mejoren la calidad y permitan reducir los costos de las soluciones basadas en computadoras ha sido uno de los objetivos más perseguidos desde el inicio de la industria del software. La utilización masiva de las computadoras y, por ende, de los sistemas informáticos en los negocios ha generado una cultura de "sistematización de los procesos". Sumado a esto, las herramientas de desarrollo han permitido la generación de software de una manera mucho más sencilla, rápida y menos costosa, lo que ha sustentado la cultura de la sistematización. Así se logró insertar sistemas software en empresas o negocios pequeños, que de otra manera no hubiesen podido encarar un proyecto de este tipo. A raíz de esto cada vez se requieren más y mejores sistemas y en tiempos considerablemente menores, aunque la construcción de software no es tarea sencilla por muy variados motivos. Normalmente se requiere que el software cubra todas las necesidades de la empresa, y que se desarrolle en un tiempo limitado. Además, existen innumerables metodologías de desarrollo de software que sustentan este proceso de construcción, lo cual complica la decisión de qué metodología utilizar en cada caso. Existen diferentes propuestas ortodoxas [1, 2] que se direccionan especialmente en el control del proceso, definiendo rigurosamente las actividades a desarrollar, los resultados a producir, las notaciones, y las herramientas a utilizar. Como alternativa para mejorar el desarrollo se pueden incluir más actividades, más diagramas [2], pero esto sólo puede llevar a un desarrollo más complejo y a sobrecargar al equipo de trabajo.

Una alternativa a este proceso de desarrollo se basa en centrarse en otros factores, como son el humano o el producto software a construir. Esta idea se conoce como Metodologías Ágiles [3], que dan mayor valor al individuo, mayor participación al cliente, y tienen como objetivo realizar entregas incrementales del software. Estos métodos son utilizados en proyectos en los cuales los requerimientos o expectativas del cliente/usuario son cambiantes o difusos. También suele exigirse un tiempo de entrega reducido con un producto que respete estándares de calidad [2].

En experiencias en desarrollo de software surge una problemática recurrente: la falta de adecuación de los requerimientos establecidos de un software a desarrollar con las expectativas que el cliente/usuario tiene del mismo [4, 2]. Esto tiene su origen en que los métodos tradicionales exigen que los requerimientos sean establecidos rigurosamente, en contraposición con las expectativas del cliente/usuario, que son dinámicas a lo largo del proyecto. De hecho, tales expectativas se modifican a medida que se visualizan las potencialidades del software desarrollado en las entregas parciales del mismo; entregas que, por otro lado, son necesarias para verificar tempranamente el producto de software y renovar el compromiso hacia el proyecto [1, 3, 5]. A medida que el proyecto se desarrolla, es posible que otros interesados (*stakeholders*) intervengan en el mismo, y vean potencialidades en el dominio de aplicación no contempladas en los requerimientos. En este escenario, es habitual que la empresa cliente vea que algunas expectativas no se cumplen con la versión final del producto, por más que éste cumpla plenamente con los requerimientos. Por ende, resulta necesario para la aceptación del producto (y la impresión positiva del equipo de desarrollo) reflejar estas expectativas no cubiertas en los requerimientos de alguna manera, para que se materialicen en la versión final del producto.

Esta problemática ha llevado a la necesidad de utilizar diferentes modelos de proceso de software adaptados y combinados de diferente manera para lograr un proceso óptimo sin perder de vista uno de los objetivos que es lograr un software de calidad en todos los aspectos, y confiable en todas sus dimensiones tanto en disponibilidad, fiabilidad, seguridad y protección. Aún se está en un proceso de maduración, y cada método se ajusta mejor a algún tipo de sistema. Por otro lado, no existen proyectos iguales, con lo cual un proceso que funcionó para desarrollar un sistema puede llegar a encontrar dificultades en otro. El desafío para los próximos años será definir y *consensuar* un modelo estándar que permita llevar adelante el proceso de desarrollo de una manera eficiente y ordenada.

Los modelos son simplificaciones de los procesos de software y, por lo tanto, un modelo de procesos de software es una simplificación o abstracción de un proceso real. Se puede definir a un modelo de procesos del software como una representación abstracta de alto nivel de un proceso de software [2]. Revisando los enfoques propuestos [1, 2, 5, 6] es posible identificar una variedad de modelos de desarrollo de software, como así también modelos similares mencionados de diferente manera, pero realizando un análisis minucioso de su conformación y manera de organizar el proceso, éstos realmente reflejan el mismo funcionamiento.

Cada modelo es una descripción de un proceso de software que se presenta desde una perspectiva particular [2], describiendo una sucesión de fases y una relación entre ellas. Según las fases y el modo en que se produzca esa relación, tenemos diferentes modelos de proceso. Es por ello que un modelo es más adecuado que otro para desarrollar un proyecto dependiendo de un conjunto de características de éste [1, 2]. Análogamente, las versiones entregables de cada etapa varían conforme su tasa de cambio y validación; en el caso que se acepte la dinámica de requerimientos como una suposición presente en el modelo, estos cambios deben documentarse y trazarse adecuadamente, para justificar posibles desviaciones no contempladas en el proyecto.

Los procesos del software que se basan en el enfoque en cascada se siguen utilizando para el desarrollo de software, particularmente cuando éste es parte de grandes proyectos de ingeniería de sistemas. El modelo permite que se produzca la documentación necesaria en cada etapa, haciendo posible un seguimiento del proyecto. Esto provoca una inflexibilidad al dividir el proyecto en etapas, ya que se hacen compromisos en las etapas iniciales, impidiendo los cambios en los requerimientos del cliente una vez que se comienza el desarrollo. El modelo en cascada se debe utilizar cuando los requerimientos se comprendan bien y sea improbable que cambien radicalmente durante el desarrollo del sistema. Es difícil utilizarlo en empresas pequeñas que requieren rápidas respuestas con costos reducidos.

Por otro lado, la esencia de los procesos iterativos es que la especificación se desarrolla junto con el software. Sin embargo, esto crea conflictos con el modelo de obtención de muchas organizaciones, donde la especificación completa del sistema es parte del contrato de desarrollo del mismo. En el enfoque incremental, no existe una especificación completa del sistema hasta que el incremento final se especifica. Esto requiere un nuevo tipo de contrato, que a los clientes grandes como las dependencias del gobierno les puede resultar difícil de incorporar [2].

Se ha desarrollado una variante de este enfoque incremental denominada Metodologías Ágiles. Éstas se basan en el desarrollo y la entrega de incrementos de

funcionalidad muy pequeños, en la participación del cliente en el proceso, en la mejora constante del código y en la programación por parejas. Pero en esta metodología de trabajo también es difícil estimar el costo de un proyecto, dado que el alcance del mismo no está completamente definido al comienzo, y los cambios durante todo el proceso son una característica del modelo.

Todos los métodos tienen limitaciones, y los métodos ágiles son apropiados para algunos tipos de desarrollo de sistemas. Son los más idóneos para el desarrollo de sistemas de negocio pequeños y de tamaño medio, y para el desarrollo de productos para computadoras personales. No parecen ser adecuados para el desarrollo de sistemas de gran escala con equipos de desarrollo ubicados en diferentes lugares y donde pueda haber complejas interacciones con otros sistemas hardware o software. Existe la posibilidad de que, si un proyecto es grande, puede ser dividido en módulos más pequeños, lo suficientemente independientes y con una orientación a servicios se pueden crear equipos ágiles que atiendan el desarrollo de los módulos. No se deben utilizar métodos ágiles para el desarrollo de sistemas críticos en los que es necesario un análisis detallado de todos los requerimientos del sistema para comprender sus implicaciones de seguridad o protección.

En las metodologías ágiles se ve reflejada la manera de llevar adelante los proyectos de software para PYMES. Se aprecia una alta similitud en cuanto a los principios del manifiesto ágil [7], donde se plantea que la mayor prioridad es satisfacer al cliente a través de entregables tempranos y continuos, presentaciones frecuentes de software, requisitos y diseños emergen de equipos que lo organizan por sí mismos, participación activa de los involucrados, entre otros. El inconveniente de las metodologías ágiles puede pasar por la falta de documentación, lo cual podría llegar a afectar el mantenimiento del sistema a futuro, si éste debe ser realizado por una persona o grupo de personas que no estuvieron involucrados en el desarrollo del mismo. Otro problema está relacionado con la estructura general del sistema. La falta de un diseño global del sistema podría llegar a entorpecer las estructuras, generar estructuras de datos no normalizadas y código fuertemente acoplado, si no se logra tener en las primeras etapas una concepción global de lo que se va a realizar.

3 Aplicación de una Metodología en PYMES

A continuación, se presenta una metodología de desarrollo de software y un marco de trabajo pensado para las PYMES. Este tipo de empresas son especialmente sensibles al mal uso de los recursos, por lo que es importante que puedan contar con herramientas que les permitan administrar el proceso productivo y les ayuden a controlar el proceso de desarrollo y en consecuencia mejorar la calidad del software que desarrollan [8]. En particular, este trabajo se enfocó en empresas ubicadas en el área de influencia de la Facultad de Ciencias de la Administración de la Universidad Nacional de Entre Ríos con sede en Concordia en el marco del Proyecto de Investigación y Desarrollo PID-UNER 7049: “Modelos de Certificación para los procesos de Ingeniería de Software en productos desarrollados con Lenguajes de Programación *Open Source*: relevamiento y aplicación en PYMES de la zona de influencia de la UNER Concordia” [9].

La metodología que se expone en el trabajo se basa en el supuesto de que la mayoría de los departamentos o PYMES de la zona dedicados al desarrollo de software, se encuentran formados por un número pequeño de entre uno a quince integrantes, por lo que es muy probable que el proceso de desarrollo de software, desde el análisis de requerimientos hasta la implementación, sea realizado por las mismas personas que tomarán múltiples roles.

El ejemplo se desarrolló utilizando la metodología ágil XP, que fue puesta en práctica en una entidad pública. A través del ejemplo práctico se buscó proveer de una estructura capaz de orientar la aplicación de un proceso de software conocido, ya que la existencia de un modelo es el comienzo en los primeros pasos en la dirección al gerenciamiento y a la mejora del proceso de software.

La metodología XP consiste en una serie de prácticas y reglas que permiten a los programadores desarrollar software de una manera dinámica y ágil, con mínima documentación. En caso de que el cliente necesite documentación formal, se deberá adecuar la metodología para ir documentando a medida que se avanza en la construcción del software. Además, es recomendable para proyectos pequeños donde se involucre un número reducido de personas. Por último, es importante recordar que XP es adecuado en ambientes donde se producen rápidas y frecuentes modificaciones de los requerimientos [10].

El ejemplo se desarrolló describiendo las diferentes fases de XP, ejemplificando las mismas, y mostrando las tareas básicas a realizar en cada una. Como lo propone XP, se utilizaron algunas de las herramientas de software para apoyar el desarrollo del software, como SVN server para el versionado del código y la documentación, Junit para realizar las pruebas unitarias del código Java, y Visual Paradig para documentar los diagramas UML.

3.1 Fase de Exploración

El objetivo de esta fase es entender lo que el sistema debe hacer, para poder estimar los tiempos del proyecto. Esta fase comienza describiendo las reglas de negocio, donde el usuario escribe las historias de usuario y el programador estima los tiempos de las historias [10].

Se realizó una primera reunión con el jefe del área de cada empresa y las personas responsables, lo cual permitió definir el proyecto, ver la dimensión del mismo, las relaciones existentes con otros sistemas, la cantidad de procesos de usuarios involucrados, los usuarios que intervienen en las operatorias, los controles generales que el sistema debe tener, y el tipo de interfaz que el usuario desea, entre otros puntos. En estas reuniones no es necesario que estén presentes todos los integrantes del equipo de desarrollo. Posteriormente se realizaron varias reuniones más a f in de recabar mayor información y centrarse en algunos de los puntos anteriores. En estas reuniones se utilizaron algunas de las técnicas de descubrimiento de requerimientos, como, por ejemplo, entrevistas, escenarios, casos de uso, historias de usuarios, etc.

Como parte de las tareas de los programadores, se redactaron las diferentes historias de usuarios, ya que fue difícil lograr que las redacten los mismos. Por tal motivo, se optó por dialogar con ellos e ir recabando la información y redactar así las diferentes historias de usuarios.

Como resultado de esta etapa se desarrolló un documento que contó con los objetivos del proyecto, las áreas involucradas, un resumen del circuito de trabajo general, sin entrar en detalles de cada una de las tareas, y las historias de usuarios. En proyectos de mayor complejidad se determinó que será necesario agregar un diagrama de casos de uso de los casos identificados, y un diagrama de actividad de los casos de uso más significativos. Otra de las actividades realizadas en las primeras reuniones de los programadores fue la definición de cuestiones técnicas del proyecto.

3.2 Fase de Planificación de Entregas

La siguiente fase fue la de planificación, cuyo objetivo fue definir las historias de usuario o funcionalidades a implementar en la primera iteración y el tiempo de cada una de ellas. Este cálculo se efectuó en base a las estimaciones que realizaran el cliente y los programadores, dándole puntos a cada historia en base a la experiencia de desarrollo en problemas similares. Los usuarios del sistema (clientes) decidieron cuáles historias de usuarios eran vitales para una primera versión, estableciendo una lista de prioridades de las historias.

Esta etapa se realizó en dos días. En general, si la etapa de exploración se realizó correctamente, unos pocos días deben ser suficientes. Como resultado de esta etapa se estableció que la primera iteración se debía realizar en un mes. Se sugiere agregar al documento de proyecto la información que va surgiendo en cada fase con el título "*Número de planificación de entrega*".

3.3 Iteraciones

A partir de la definición realizada en la fase anterior, se comenzaron a producir las funcionalidades asignadas a la primera iteración. La recomendación que surge de la experiencia es la construcción de casos de uso para cada historia de usuario. Cuando el caso de uso poseía cierta complejidad, también se utilizaron diagramas de actividad. Si bien una de las fundamentaciones de XP es la de tener una documentación mínima, la que se utilizó se considera indispensable para una mejor comprensión de cada historia de usuario y para su posterior seguimiento.

Se definieron los modelos de datos necesarios para sustentar los programas realizados para cada una de las historias de usuario asignadas a la iteración. Se diseñaron y programaron las pruebas unitarias de los diferentes módulos. Además, se diseñaron pruebas manuales o test de aceptación a aplicar a las interfaces de usuarios, describiendo los pasos necesarios para completar las acciones y los resultados finales a obtener. Como tarea extra se podrían realizar pruebas de integración con herramientas de automatización.

Por cada historia de usuario se llevó a cabo la programación. Tomando en cuenta la recomendación de XP sobre la programación en parejas, el comienzo de cada historia de usuario se programó en grupos de dos programadores. Una vez que el trabajo estaba avanzado, se continuó programando individualmente. En esta etapa también surgieron casos de usos y problemas propios del proceso de desarrollo, como requerimientos no funcionales de seguridad, tiempo de respuesta, interfaces de

usuarios, etc. que no fueron especificados en la fase de exploración. Estas tareas, que son propias de los programadores, se fueron organizando en las reuniones denominadas "reuniones de parado" (*stand-up meeting*).

Otra recomendación realizada fue la de publicar lo antes posible las nuevas versiones de código, aunque no sean las últimas, siempre que estén libres de errores. Idealmente, todos los días deben existir nuevas versiones publicadas. Todos los desarrolladores necesitan trabajar siempre con la "última versión". Si bien la metodología XP, no plantea la utilización de un servidor de control de versiones, es aconsejable su utilización.

Además, a medida que se avanzó en el desarrollo del sistema, se fueron detectando requerimientos del sistema que no son parte del dominio de la aplicación específica, necesitándose desarrollar funcionalidad de creación de usuarios, menú dinámico del sistema, perfiles de usuarios y auditorías. Es importante subrayar que no siempre los clientes o usuarios expresarán la necesidad de contar con este tipo de funcionalidad, pero es probable que una vez que se pruebe o se intente poner en producción las historias de usuarios programadas en la iteración surjan estas inquietudes. Por este motivo todos estos casos fueron registrados y puestos a consideración para la próxima planificación de iteración.

Otra situación identificada es la adecuación constante de las historias de usuario a medida que se avanza en el desarrollo y se recaba mayor información respecto al funcionamiento, los datos que se procesan y los controles que se realizan sobre los datos. Es frecuente encontrar que ciertos requerimientos o historias de usuario hayan sido planteados de una manera confusa e incompleta, y a medida que se trabaja en el código surgen inconsistencias entre lo programado y lo que el usuario realmente intentó plantear. Una premisa importante de la metodología XP es la idea de mantener una retroalimentación constante con el usuario. Se utilizó la primera iteración para desarrollar un prototipo del sistema, el entorno de trabajo y establecer los formatos de las interfaces.

3.4 Producción

La fase de producción implica trasladar el software generado al cliente. La actividad de producción requiere de pruebas adicionales y revisiones de rendimiento antes de que el sistema sea trasladado al entorno del cliente. Si el sistema es muy grande se podrá comenzar a implementar por módulos de funcionamiento, pero si el problema se acota a un módulo de funcionamiento se deberá esperar a tener gran parte del sistema desarrollado. En el trabajo se optó por la alternativa de continuar el desarrollo y ejecución de nuevas iteraciones sin la implementación definitiva. Sólo se fueron realizando pruebas con los usuarios a fin de verificar la correctitud del producto desarrollado.

3.5 Mantenimiento

La fase de mantenimiento recién se pone en práctica una vez completada una fase de producción, teniendo una versión operativa que el cliente esté utilizando, ya que un

proyecto XP debe mantener el sistema en funcionamiento al mismo tiempo que desarrolla nuevas iteraciones. Es importante recalcar que una vez que se pone en marcha el proyecto se comienza a dar soporte a l os usuarios del sistema, y la velocidad de desarrollo decae ya que parte del esfuerzo se pone en la solución de los inconvenientes.

4. Resultados Obtenidos

Como se mencionó en la Sección 3, la metodología propuesta se aplicó en una entidad pública de la ciudad de Concordia, que cuenta con un equipo de desarrollo integrado por 4 personas. El tiempo insumido por el proyecto fue de 5 meses.

Además del proyecto puntual tomado como ejemplo, la metodología XP se puso en práctica en otros 4 proyectos dentro de la misma institución. En proyectos anteriores se habían utilizado metodologías tradicionales, intentado abarcar la mayor cantidad de situaciones. El inconveniente detectado es el esfuerzo considerable que exigía la confección de la documentación y la gestión del proyecto, lo que generaba una sobrecarga innecesaria.

Teniendo en cuenta la experiencia realizada con la utilización de la metodología XP, se pudo apreciar que es una metodología centrada principalmente en la relación de las personas. A lo largo del trabajo se promovió el trabajo en equipo, el aprendizaje de cada uno los desarrolladores, y siempre se generó un buen clima de trabajo. Siguiendo los pasos propuestos, se hizo hincapié en la satisfacción del usuario/cliente, y esto permitió a los desarrolladores responder a los cambios incluso en las etapas finales del proyecto. Esto permitió dar respuesta a problemas de cambios de requerimientos, ya que los usuarios no tenían una idea firme de lo que el sistema debía hacer, pero sí de su trabajo.

Los programadores no necesariamente tenían una experiencia considerable en la metodología XP, y se podría recalcar que para proyectos con requisitos dinámicos o de alto riesgo, es probable que un equipo pequeño de programadores sea más eficaz que un equipo grande. Resumiendo, XP propone una metodología ágil y c oncreta, aunque requiere de una nueva manera de pensar, ver y hacer las cosas, tanto por parte de los gerentes, como de los desarrolladores y también del cliente. La aplicabilidad de esta metodología, y de las adaptaciones propuestas, a cada proyecto en particular, debería ser analizada en cada caso, teniendo en cuenta el tamaño y tipo de proyecto, sus ventajas y desventajas.

A partir de la experiencia se están elaborando guías de diseño con la intervención de todos los programadores desarrolladores involucrados en los diferentes proyectos a fin de identificar fortalezas y debilidades de la metodología utilizada (XP), así como aquellas actividades no propias de la metodología pero que contribuyen fuertemente a mejorar el trabajo en equipo.

5. Conclusiones y Trabajos Futuros

En base a l a investigación realizada se puede asegurar que los cambios son inevitables en todos los proyectos de software, sean éstos pequeños, medianos o grandes. Los requerimientos del sistema cambian cuando el negocio donde está inserto el sistema responde a las presiones externas.

Las prioridades de gestión cambian y, cuando se dispone de nuevas tecnologías, cambian los diseños y la implementación. Esto significa que el proceso del software no es un proceso único; más bien, las actividades del proceso se repiten regularmente conforme el sistema se rehace en respuesta a peticiones de cambios. Como surge del estudio de las diferentes metodologías, el problema de cada una de ellas siempre gira alrededor de la falta de una correcta definición de los requerimientos, de la gestión de la documentación y de la falta de comprensión del cliente con respecto a l os requerimientos. Todos estos factores llevan a requerimientos confusos, incompletos y cambiantes.

Además de la metodología que se utilice, es importante definir una buena estructura de documentación, a fin de poder reflejar los requerimientos que dieron origen al sistema, los diferentes componentes del sistema software, sus relaciones y sus efectos en las modificaciones del mismo.

Otro aspecto que no debe dejarse de lado, es el aseguramiento de la calidad en el proceso y producto de software; en particular si es un requisito para certificar o validar conforme a es tándares internacionales de calidad [11]. Como equipo (empresa) de desarrollo, resulta adecuado instaurar buenas prácticas en el proceso de desarrollo conforme a g uías de certificación aceptadas: ISO/IEC 9001- 90003 [12, 13], ISO 15504 SPICE [14], ISO/IEC 15288 [4], ISO/IEC/IEEE 12207 y complementarias [15, 16]. Como empresa usuaria o productora de software sería importante verificar la calidad del producto conforme a esquemas de certificaciones ISO/IEC 25000 [17, 18, 19], permitiendo como efecto deseado tener ventajas competitivas y comerciales.

Cómo parte del trabajo se planteó el uso de la metodología XP para el desarrollo de software en las PYMES, dando como resultado un proceso de desarrollo de software confiable, que permite involucrar a todos los participantes, y que logra mantener los requerimientos con cambios previsibles y sin un i mpacto negativo en el resto del sistema ya desarrollado. Esto evita que los mismos se tornen confusos, y permite que se vayan alcanzando a medida que se avanza en el proyecto. En el caso de que algún requerimiento cambie, la metodología propuesta evita que este cambio impacte negativamente en todo el proyecto.

Se realizaron algunas adaptaciones a la metodología, a fin de incorporar documentación necesaria. Por ejemplo, se creó un documento de proyecto y se propuso la utilización de diagramas de casos de uso y actividad. Por otro lado, el abordaje del sistema completo al inicio permitió tener una noción general del sistema y así evitar los problemas mencionados.

Como trabajo futuro será importante continuar el estudio de los Modelos de Proceso de Software y describir pasos metodológicos para llevar adelante un proyecto de software exitoso utilizando alguna combinación de diferentes metodologías tradicionales y ágiles, ya que cada una posee ventajas y desventajas. Así se espera lograr un enfoque híbrido, en el cual los métodos ágiles incorporen algunas técnicas

de planificación y documentación, pero sin perder de vista las iteraciones y los principios del modelo ágil. También, se buscará ampliar el espectro de pruebas a más PYMES de la zona de influencia de la Facultad de Ciencias de la Administración de la Universidad Nacional de Entre Ríos con sede en Concordia.

Agradecimientos

El presente trabajo fue parcialmente financiado por el proyecto de investigación PID UNER 7049 titulado *"Guías para aplicación de Normas de Calidad para los procesos de Ingeniería de Software en productos desarrollados con Lenguajes de Programación Open Source: relevamiento y aplicación en PYMES de la zona de influencia de la UNER Concordia"*.

Referencias

1. Pressman, R. S., Maxim, B.: "Software Engineering: A Practitioner's Approach". McGraw-Hill, 7ma Ed, 2014.
2. Sommerville, I.: "Software Engineering" (10th Ed). Pearson Addison Wesley, 2015.
3. Beedle, M.: "Principles behind the Agile Manifesto". http://www.agilemanifesto.org/principles.html
4. ISO: "ISO/IEC 15288:2008 - Systems and software engineering - System life cycle processes." Mar. 2008.
5. Wells, D.: "Extreme programming: A gentle introduction". Tech. Report, 2009. http://www.extremeprogramming.org
6. Jacobson, I., Booch, G., Rumbaugh, J.: "El proceso unificado de desarrollo de software". Addison-Wesley, primera edición, 1999.
7. Beck, K., Beedle, M., van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., Grenning, J., Highsmith, J., Hunt, A., Jeffries, R., Kern, J., Marick, B., Martin, R. C., Mellor, S., Schwaber, K., Sutherland, J. & Thomas, D.: "Manifesto for Agile Software Development" (2001).
8. Pérez, I., Castelló, R.: "Columna Vertebral del Software, Una Metodología de Desarrollo Para PyMES". ITESM Campus Chihuahua, México.
9. Colombani, M., Pérez, M., Pacifico, C.: "Metodologías para el desarrollo de software en PYMES". Workshop de Investigadores en Ciencias de la Computación WICC (2016).
10. Bona, C., Thiry, M.: "Processo de Software: um Estudo de Caso em XP". CACIC - Congreso Argentino de Ciencias de la Computación. 2003.
11. Schneider, F., Berenbach, B.: "A Literature Survey on International Standards for Systems Requirements Engineering", Procedia Computer Science, Volume 16, 2013, Pages 796-805.
12. ISO: "ISO/IEC 9001:2015 - Quality management systems - Requirements" Sep, 2015.
13. ISO: "ISO/IEC 90003:2014 - Software engineering - Guidelines for the application of ISO 9001:2008 to computer software" Dec, 2014.
14. ISO: "ISO/IEC 15504 Information technology - Process assessment" ISO/IEC, Nov, 2004.
15. ISO: "ISO/IEC/IEEE 12207:08 Systems and software engineering - Software life cycle processes," Mar. 2008.
16. ISO: "ISO/IEC/IEEE 29148: 2011 - Systems and software engineering - Life cycle processes - Requirements engineering," Nov. 2011.

17. ISO: “ISO/IEC 25030 - Software engineering - Software product Quality Requirements and Evaluation (SQuaRE) - Quality requirements,” Jun. 2007.
18. ISO: “ISO/IEC 25040 – Systems and software engineering - Systems and software Quality Requirements and Evaluation (SQuaRE) - Evaluation process.” Feb. 2011.
19. ISO: “ISO/IEC 25010 –Systems and software Quality Requirements and Evaluation (SQuaRE) - System and software quality models,” Mar. 2011.