Un Framework para Evaluación de Metodologías Ágiles

Karla Mendes Calo, Elsa Estevez, Pablo Fillottrani

Laboratorio de I&D en Ingeniería de Software y Sistemas de Información (LISSI)
Departamento de Ciencias e Ingeniería de la Computación
Universidad Nacional del Sur,
Avenida Alem 1253,
(8000) Bahía Blanca, Argentina
{kmca, ece, prf}@cs.uns.edu.ar

Resumen. Las metodologías de desarrollo ágil se basan fundamentalmente en la colaboración con los usuarios de software durante todo el proceso de desarrollo, la facilidad para adaptar el producto a cambios en requerimientos y en la entrega incremental del producto. Basadas en el Manifiesto Ágil, han sido aceptadas y son utilizadas con éxito en proyectos donde los requerimientos detallados son inicialmente desconocidos y se van construyendo durante el proceso de desarrollo a partir de interacciones con los usuarios y de la retro-alimentación obtenida a partir de las mismas. En este trabajo se propone un framework de evaluación para las metodologías ágiles de desarrollo, y se aplica a dos de ellas – Scrum y eXtreme Programming (XP). La definición de este framework cuantitavio es novedosa, especialmente porque permite evaluar en cuánto las metodologías ágiles satisfacen los principios básicos definidos por el Manifiesto Ágil. Su utilización es recomendada al momento de decidir una metodología a adoptar.

Keywords: Manifiesto Ágil, Metodologías Ágiles, SCRUM, XP

1 Introducción

Tradicionalmente, los procesos de desarrollo de software llevan asociado un marcado acento en el control del proceso. Definen actividades, artefactos y documentación a producir, herramientas y notaciones a ser utilizadas, orden de ejecución de las actividades, entre otras definiciones. Si bien existen varios procesos de desarrollo – Proceso Unificado [1], Proceso V [2], etc. , la mayoría de estos procesos se derivan del Modelo de Cascada propuesto por Boehm [3]. Estos procesos, denominados tradicionales, han demostrado ser efectivos en proyectos de gran tamaño, particularmente en lo que respecta a la administración de recursos a utilizar y a la planificación de los tiempos de desarrollo. Sin embargo, el enfoque propuesto por estos métodos no resulta el más adecuado para el desarrollo de proyectos donde los requerimientos del sistema son muy cambiantes, se pide reducir drásticamente los tiempos de desarrollo y al mismo tiempo producir un producto de alta calidad.

Como alternativa a los métodos tradicionales de desarrollo, surgen las Metodologías Ágiles. Manteniendo prácticas esenciales de las metodologías

tradicionales, las metodologías ágiles se centran en otras dimensiones del proyecto, como por ejemplo: la colaboración con los usuarios durante todas las etapas del proceso de desarrollo, y el desarrollo incremental del software con iteraciones muy cortas que entregan una solución a medida. Las prácticas ágiles están especialmente indicadas para productos cuya definición detallada es difícil de obtener desde el comienzo, o que si se definiera, tendría menor valor que si el producto se construye con una retro-alimentación continua durante el proceso de desarrollo.

El objetivo de este trabajo es presentar un marco de evaluación de metodologías ágiles que permite evaluar en qué medida las metodologías cumplen con los valores declarados por el Manifiesto Ágil. El marco de evaluación permite tomar una decisión más informada al momento de seleccionar una de estas metodologías. A modo de ejemplo, el marco se aplica para las metodologías SCRUM y XP.

El resto de este trabajo se estructura de la siguiente manera. La Sección 2 presenta el Manifiesto Ágil y algunas de las metodologías ágiles comúnmente utilizadas. La Sección 3 explica en detalle dos metodologías – SCRUM y XP. A continuación, la Sección 4 presenta y explica el framework de evaluación, mientras que la Sección 5 muestra su aplicación a las metodologías SCRUM y XP. Finalmente, la Sección 6 presenta comparación con trabajos relacionados, conclusiones y trabajo futuro.

2 Manifiesto Ágil y Metodologías Ágiles de Desarrollo

En febrero de 2001, académicos y expertos de la industria del software se reunieron en Utha, Estados Unidos, a fin de discutir los valores y principios que facilitarían desarrollar software más rápidamente y respondiendo a los cambios que surjan a lo largo del proyecto. La idea era ofrecer una alternativa a los procesos de desarrollo tradicionales. Como resultado de esta reunión, se creó The Agile Alliance [4], una organización, sin fines de lucro, dedicada a promover los conceptos relacionados con el desarrollo ágil de software y ayudar a las organizaciones a adoptar dichos conceptos. El resultado de esta reunión fue un documento conocido como el Manifiesto Ágil [5]. El Manifiesto Ágil incluye cuatro postulados y una serie de principios asociados. Sus postulados son:

1) *Valorar al individuo y a las interacciones del equipo de desarrollo por encima del proceso y las herramientas*. Tres premisas sustentan este principio: a) los integrantes del equipo son el factor principal de éxito de un proyecto; b) es más importante construir el equipo de trabajo que construir el entorno; y c) es mejor crear el equipo y que éste configure el entorno en base a sus propias necesidades.
2) *Valorar el desarrollo de software que funcione por sobre una documentación exhaustiva*. El principio se basa en la premisa que los documentos no pueden sustituir ni ofrecer el valor agregado que se logra con la comunicación directa entre las personas a través de la interacción con los prototipos. Se debe reducir al mínimo indispensable el uso de documentación que genera trabajo y que no aporta un valor directo al producto.
3) *Valorar la colaboración con el cliente por sobre la negociación contractual*. En el desarrollo ágil el cliente se integra y colabora con el equipo de trabajo como un

integrante más. El contrato en sí no aporta valor al producto, es sólo un formalismo que establece líneas de responsabilidad entre las partes.

4) *Valorar la respuesta al cambio por sobre el seguimiento de un plan.* La evolución rápida y continua deben ser factores inherentes al proceso de desarrollo. Se debe valorar la capacidad de respuesta ante los cambios por sobre la capacidad de seguimiento y aseguramiento de planes pre-establecidos.

El ciclo de desarrollo que aplican las Metodologías Ágiles es iterativo e incremental. Este modelo permite entregar el software en partes pequeñas y utilizables, conocidas como *incrementos*. Cada iteración se puede considerar como un mini-proyecto en el que las actividades de análisis de requerimiento, diseño, implementación y testing son llevadas a cabo con el fin de producir un subconjunto del sistema final. El proceso se repite varias veces produciendo un nuevo incremento en cada ciclo hasta que se elabora el producto completo. Si bien todas las metodologías ágiles adoptan este ciclo, cada una presenta sus propias características..

Las metodologías ágiles más comúnmente usadas se describen a continuación.

Scrum [6] – Indicada para proyectos con alto ratio de cambio de requerimientos, su principal característica es la definición de *sprints* – cada una de las iteraciones del proceso con una duración máxima de 30 días. El resultado de cada sprint es un incremento ejecutable que se muestra al cliente. Otra característica importante son las reuniones diarias que se llevan a cabo a lo largo del proyecto. Dichas reuniones no requieren más de 15 minutos del equipo de desarrollo y su objetivo son la coordinación e integración del producto a entregar.

Cristal Methodologies [7] – Se trata de un conjunto de metodologías para el desarrollo de software caracterizadas por la valoración a las personas que componen el equipo de trabajo y la reducción al máximo del número de artefactos producidos. Enfatiza en esfuerzos para mejorar las habilidades de los integrantes del equipo y para definir políticas de trabajo en equipo. Las políticas dependerán del tamaño del equipo, estableciéndose una clasificación por colores, por ejemplo Crystal Clear corresponde a equipos con 3 a 8 integrantes y Crystal Orange en equipos con 25 a 50 integrantes.

Dynamic Systems Development Method (DSDM) [8] – Cumple con las características generales de definir un proceso iterativo e incremental. Propone cinco etapas de desarrollo: Estudio de Viabilidad, Estudio del Negocio, Modelado Funcional, Diseño y Construcción, e Implementación. La iteración se produce en las tres últimas etapas, sin embargo prevé retro-alimentación en todas.

Adaptive Software Development (ASD) [9] – Es un proceso iterativo, tolerante a cambios y orientado a los componentes de software. Define tres etapas para el ciclo de vida: a) Especulación – se inicia el proyecto y se planifican las características del software; b) Colaboración – se desarrolla el producto; y c) Aprendizaje – se revisa la calidad del producto y se entrega al cliente. La revisión tiene como objetivo aprender de los errores cometidos y volver a iniciar el ciclo de desarrollo.

Feature-Driven Development (FDD) [10] – Define un proceso iterativo, con iteraciones cortas de dos semanas como máximo. El ciclo de vida consta de cinco pasos: a) Desarrollo de un modelo global, b) Construcción de una lista de funcionalidades, c) Planeación por funcionalidad, d) Diseño por funcionalidad y e) Construcción por funcionalidad.

Programación Extrema (XP) [11] **–** Define un proceso iterativo e incremental con pruebas unitarias continuas y entregas frecuentes. El cliente o un representante del

cliente son integrados al equipo de desarrollo. Recomienda que el desarrollo de las funciones del producto sea realizado por dos personas en el mismo puesto – programación por pares. Antes de incorporar nueva funcionalidad, se deben corregir todos los defectos encontrados. Constantemente, se llevan a cabo pruebas de regresión a fin de detectar los posibles errores.

3 Scrum y XP – Principios, Actividades, Roles y Prácticas

Las próximas dos secciones presentan en detalle los principios, actividades, roles a cubrir en los equipos de trabajo y prácticas recomendadas de Scrum y XP.

3.1 Scrum

La metodología respeta el ciclo de vida evolutivo y la entrega incremental. Al comienzo del proyecto, se identifican los requerimientos funcionales y no funcionales y se conforma una lista de los mismos llamada *product backlog*. El product backlog constituye el artefacto base para medir el avance del proyecto. Las iteraciones, denominadas *sprints* entregan partes del producto llamadas *builds*, que si bien no incluyen toda la funcionalidad del sistema, constituyen ejecutables operativos. Cada iteración comienza con una planificación adaptativa guiada por el cliente y culmina con una demostración del build al cliente. Cada sprint puede durar como máximo 30 días. En cada sprint, el equipo de desarrollo selecciona del product backlog un conjunto de ítems de mayor prioridad que se convierte en el objetivo a desarrollar. La metodología propone las siguientes tres fases:

1) *Fase de Planeamiento* – es subdividida en: a) *Planeación* – se define el equipo del proyecto, herramientas, el sistema de desarrollo y se crea el product backlog con la lista de requerimientos conocidos hasta ese momento, se definen prioridades para los requerimientos y se estima el esfuerzo necesario para llevar a cabo la implementación de los mismos; y b) *Diseño Arquitectónico* – se define la arquitectura del producto que permita implementar los requerimientos definidos.
2) *Fase de Desarrollo* - es la parte ágil, donde el sistema se desarrolla en sprints. Cada sprint incluye las fases tradicionales del desarrollo de software – relevamiento de requerimientos, análisis, diseño, implementación y entrega.
3) *Fase de Finalización* – incluye integración, testing y documentación. Indica la implementación de todos los requerimientos, quedando el product backlog vacío y el sistema listo para entrar en producción.

La metodología propone la creación de equipos de trabajo auto-dirigidos y auto-organizados, aconsejando equipos pequeños que maximizan la comunicación entre sus integrantes. Dentro del equipo de trabajo, se identifican roles com o el S*crum Master* – responsable de asegurar que el proyecto se ejecute en base a las prácticas, valores y reglas de Scrum-; el *Dueño del Producto* – responsable del proyecto, administra, controla y mantiene y publica el product backlog-; los *Miembros del Equipo* – tienen la autoridad para decidir acerca de las acciones a realizar y organizarlas de tal manera de alcanzar los objetivos de cada sprint; y el *Cliente* –

participa de las tareas relacionadas con la lista de requerimientos del producto a desarrollar, aporta ideas, sugerencias y nuevas necesidades-.
Scrum prevee las siguientes prácticas:

- *Reunión de Planeamiento del Sprint* – organizada por el Scrum Master, se divide en dos etapas. En la primera etapa se reúnen los clientes, el dueño del producto y los miembros del equipo para decidir sobre los objetivos y funcionalidad del nuevo sprint. La segunda etapa de la reunión se realiza entre el Scrum Master y el equipo de trabajo y se concentra en cómo el incremento del producto será implementado durante el proceso.
- *Sprint* – es una lista de requerimientos seleccionados para ser implementados en la próxima iteración. Los requerimientos son seleccionados por el equipo de trabajo, en conjunto con el Scrum Master y el propietario del producto en la reunión de planeamiento del sprint. Cuando todos los ítems del sprint se completan, se entrega una nueva iteración del sistema.
- *Reuniones Diarias* – son dirigidas por el Scrum Master. Se organizan básicamente para mantener una revisión constante del avance del proyecto. Los integrantes responden a tres preguntas: 1) ¿Qué se ha logrado completar desde la última reunión?, 2) ¿Qué obstáculos o problemas se han detectado ?, y 3) ¿Qué funciones del backlog planea completar para la próxima reunión?
- *Revisión del Sprint* - el equipo de trabajo y el Scrum Master presentan los resultados del sprint al cliente.
- *Retrospectiva del Sprint* **– s**e realiza al finalizar un product backlog y la revisión del sprint. El equipo de trabajo revisa el cumplimiento de los objetivos marcados al inicio del sprint. Se analizarán y se aplicarán los ajustes y cambios necesarios cuando corresponda, se destacarán los aspectos positivos y se tratará de cambiar aquellos aspectos negativos, para no repetirlos en el siguiente sprint.

3.2 eXtreme Programing (XP)

XP, formulada por Kent Beck, se diferencia del resto de las metodologías por su énfasis en la adaptabilidad. La metodología está diseñada para ofrecer el software que el usuario necesita, en el momento en que lo necesite. El éxito de la metodología se basa en potenciar las relaciones interpersonales, promoviendo el trabajo en equipo, el aprendizaje de los desarrolladores, y un ambiente de trabajo cordial. Los cinco principios básicos de XP incluyen: 1) *Simplicidad* – simplificar el diseño para agilizar el desarrollo y facilitar el mantenimiento mediante la refactorización del código; 2) *Comunicación* – fomenta la comunicación: escrita – como código autodocumentado y pruebas unitarias, recomendando documentar el objetivo de clases y la funcionalidad de métodos; y oral - entre programadores y con el cliente, recomendando que ambas sean constantes y fluidas; 3) *Retro-alimentación* – promueve la retro-alimentación constante del cliente a través de ciclos de entrega cortos y demostraciones de la funcionalidad entregada; 4) *Coraje* – para mantener la simplicidad permitiendo diferir decisiones de diseño, para comunicarse con los demás aún cuando esto permita exponer la propia falta de conocimiento y para recibir la retro-alimentación durante el desarrollo; y 5) *Respeto* – se infunde entre integrantes del equipo – los desarrolladores

no pueden realizar cambios que hagan que las pruebas existentes fallen o que demore el trabajo de compañeros, y hacia el trabajo – los miembros del equipo tienen como principal objetivo lograr un producto de alta calidad con un diseño óptimo.
El proceso de desarrollo consiste de tres etapas:

1) *Interacción con el cliente* – el cliente interactúa permanentemente con el equipo de trabajo. Se elimina así la fase inicial de recolección de requerimientos, y éstos se van incorporando de manera ordenada a lo largo del desarrollo. La metodología propone utilizar la técnica de *Historias de Usuario* mediante la cual el usuario especifica los requerimientos funcionales y no funcionales del producto. Cada historia debe ser lo suficientemente atómica y comprensible, para que los desarrolladores puedan implementar los requerimientos durante una iteración.
2) *Planificación del Proyecto* – el equipo de trabajo estima el esfuerzo requerido para la implementación de las Historias de Usuario. Cada historia debe ser implementada en un período de una a tres semanas. Aquellas historias que requieran más tiempo se sub-dividen tratando de que resulten atómicas y puedan realizarse dentro del plazo máximo.
3) *Diseño y Desarrollo de Pruebas* – la implementación está conducida por las pruebas unitarias. Cada vez que se quiere implementar una función, primero se debe definir el test y luego el código para que lo satisfaga. Una vez que el código cumple el test exitosamente, se amplía y se continúa. A medida que se van implementando las Historias de Usuario, se van integrando los pequeños fragmentos de código. De este modo, se realiza una integración continua, evitando una integración más costosa al finalizar el proyecto. XP fomenta la programación por pares, donde el desarrollo es realizado por una pareja de programadores. Las parejas tienen que ir cambiando de manera periódica, para que el conocimiento sea adquirido por todo el grupo de desarrollo.

Los roles definidos para los integrantes del equipo incluyen el *Programador* – encargado de escribir pruebas unitarias y producir el código-; el *Cliente* – escribe Historias de Usuario y pruebas funcionales. Asigna prioridades a las Historias de Usuario y decide cuáles se implementan en cada iteración-; el *Tester* – es el responsable de las pruebas, ayuda al cliente a escribir las pruebas funcionales, las ejecuta, informa los resultados al resto del equipo y mantiene la herramienta de soporte que se utiliza para llevar a cabo las pruebas.; el *Encargado de Seguimiento* (tracker) - proporciona retro-alimentación al equipo, verifica el grado de acierto de las estimaciones y controla el avance del proyecto. *Entrenador* (coach) – es responsable del proceso en su totalidad, guía al equipo de manera que se respeten las prácticas XP y que el proceso se ejecute correctamente; el *Consultor* – es un miembro externo al equipo con conocimientos específicos en algún tema, necesario para resolver problemas que surjan durante el proyecto; y el *Gestor* (big boss) – es el vínculo entre el cliente y los desarrolladores, ayuda a que el equipo trabaje efectivamente. Su principal tarea es la coordinación.
Entre otras, XP define las siguientes prácticas:

- *Juego de Planificación* – el equipo estima el esfuerzo requerido para la implementación de las Historias de Usuario.
- *Refactorización* – actividad constante de reestructuración de código cuyo objetivo es remover duplicación de código, mejoras de legibilidad y aumentar la flexibilidad para facilitar cambios posteriores.

- *Programación por pares* – el desarrollo lo realiza una pareja de desarrolladores.
- *Integración continua* – el código es integrado una vez que está disponible.
- *Cliente in-situ* - el cliente debe estar presente y disponible todo el tiempo.

4 Framework de Evaluación

El framework de evaluación propuesto mide de qué manera las metodologías ágiles cumplen con los postulados del Manifiesto Ágil descriptos en la Sección 2. A este efecto, el framework define medidas que satisfacen la Teoría Representacional de la Medición [12]. Las medidas se definen usando escala de intervalos [13].

El framework provee mediciones para los cuatro postulados presentados en la Sección 2. Estos postulados (Pi, i=1..4) fueron expresados como la valoración de dos atributos (Pi.1 y Pi.2). La medida de cada postulado se define como la suma de las medidas de los atributos relacionados, como se formula a continuación:

$$m(P_i) = m(P_i.1) + m(P_i.2) \quad i=1..4$$

Por ejemplo, el Postulado 1 (P1) - *Valorar al individuo y a las interacciones del equipo de desarrollo por encima del proceso y las herramientas,* se mide sumando la medida de cómo la metodología valora al individuo y a las interacciones del equipo (P1.1) y la medida de cómo valora al proceso y a las herramientas (P1.2).

El atributo que los principios intentan enfatizar (atributo positivo) se mide en una escala de 0 a 5, y el otro atributo (atributo negativo) en una escala de -5 a cero. De este modo, cada principio podría obtener una medida entre -5 – en el caso que ambos atributos tomen el peor valor (-5 el atributo negativo y 0 el atributo positivo), y 5 – en el caso que ambos atributos tomen el mejor valor (0 el atributo negativo y 5 el atributo positivo). Si se obtiene un valor cero o cercano a cero, significa que la metodología no valora significativamente el atributo positivo por sobre el negativo, en cuyo caso, el principio del Manifiesto Ágil no se satisface plenamente. El framework, los atributos y sus medidas se presentan en la Tabla 1.

Tabla 1. Framework de Evaluación de Metodologías Ágiles

P1	*Valorar al individuo y las interacciones del equipo por sobre el proceso y las herramientas*		
P1.1	*Valorar al individuo y las interacciones*	P1.2	*Valorar el proceso y las herramientas*
valor	descripción	valor	descripción
0	No define roles para individuos	-5	Define actividades, entregables, herramientas de desarrollo y de gestión
1	Clara definición de roles para individuos	-3	Define actividades, entregables y herramientas de desarrollo
2	Clara definición de roles y responsabilidades	-2	Define actividades y entregables
3	Clara definición de roles, responsabilidades y conocimientos técnicos	-1	Define actividades para cada iteración
5	Clara definición de roles, responsabilidades, conocimientos técnicos e interacciones entre miembros del equipo de trabajo	0	Define actividades para el proyecto pero no a nivel de cada iteración

P2	*Valorar el desarrollo de software que funcione por sobre una documentación exhaustiva*		
P2.1	*Valorar desarrollo de software que funcione*	P2.2	*Valorar documentación exhaustiva*
valor	descripción	valor	descripción
0	Generar entregable al finalizar el proyecto	-5	Requiere documentación detallada al comienzo del proyecto
3	Generar entregable con testing satisfactorio al finalizar cada iteración	-2	Requiere solo documentación necesaria al comienzo de cada iteración
5	Generar entregable con testing satisfactorio e integrado con el resto de las funciones al finalizar cada iteración	0	No requiere documentación para comenzar a implementar la funcionalidad incluida en una iteración
P3	*Valorar la colaboración con el cliente pro sobre la negociación contractual*		
P3.1	*Valorar la colaboración con el cliente*	P3.2	*Valorar la negociación contractual*
valor	descripción	valor	descripción
0	Cliente colabora a demanda del equipo	-5	Existe contratación detallada al inicio y no se aceptan cambios
3	Cliente es parte del equipo, responde consultas y planifica las iteraciones	-2	La contratación exige contemplar cambios durante el proyecto
5	Cliente es parte del equipo, responde consultas, planifica iteraciones, y colabora en la escritura de requerimientos y pruebas	0	El contrato por la construcción del producto no aporta valor al producto.
P4	*Valorar la respuesta al cambio por sobre el seguimiento de un plan*		
P4.1	*Valorar la respuesta al cambio*	P4.2	*Valorar el seguimiento de un plan*
valor	descripción	valor	descripción
0	No prevé incorporar cambios durante la ejecución del proyecto	-5	Define un plan detallado al inicio del proyecto
1	Prevé introducir sólo cambios de alta prioridad	-3	Define un plan detallado de iteraciones, no acepta cambios durante una iteración
4	Permite la evolución y el cambio, pero no es recomendable en la iteración en curso	-2	Define un plan detallado para cada iteración, que puede ser modificado
5	Permite introducir cambios en la iteración en curso	0	No define planificación alguna

5 Aplicación del Framework

La aplicación del framework se muestra en la Tabla 2 y se explica debajo.

Postulado P1. Scrum y XP obtienen 5 en el atributo P1.1, ya que ambas metodologías valoran al individuo, definen roles y responsabilidades y reconocen la importancia y promueven la capacitación de los integrantes del equipo. Scrum obtiene -3 en el atributo P1.2 ya que define actividades, entregables y herramientas de desarrollo; mientras que XP obtiene -2 por que solo define actividades y entregables. *Conclusión:* Scrum obtiene 2 puntos y XP 3. Scrum obtiene un menor valor ya que la metodología define herramientas para el desarrollo. XP satisface P1 mejor que Scrum.

Postulado P2. Scrum obtiene 3 puntos y XP 5 en el atributo P1.2. La diferencia radica en que XP también considera la integración parcial del software al finalizar cada iteración. Ambas metodologías se evalúan con un valor -2 para el atributo P2.2 ya que ambas sólo requieren documentación para la iteración planificada. Scrum y XP obtienen un valor positivo para el principio P2, superando XP a Scrum por 1 punto.

Conclusión: XP satisface P2 mejor que Scrum, ya que requiere que los incrementos entregados sean integrados en forma continua con el resto de la funciones.
Postulado P3. Ambas metodologías obtienen el mejor valor en ambos atributos – 5 puntos para P3.1 y 0 punto para P3.2. Ambas consideran al cliente como un miembro del equipo, que colabora desde la planificación de las iteraciones hasta la escritura de requerimientos y pruebas funcionales. Ninguna de ellas utiliza la relación contractual para agregar valor al producto. *Conclusión:* Ambas satisfacen P3 de manera óptima.
Postulado P4. En el atributo P4.1 Scrum obtiene un valor 4 ya que si bien permite introducir cambios, no se recomiendan en el sprint en curso. De ser prioritario el cambio en el sprint en curso, se debe estimar nuevamente el esfuerzo requerido y si es necesario, quitar tareas del sprint ya planificado. XP obtiene el máximo valor debido a que los cambios se pueden incorporar durante la iteración. Debido a que un enfoque similar sucede con la planificación, Scrum obtiene un valor de -3 y XP -2. *Conclusión*: XP obtiene 3 puntos y Scrum 1. XP satisface P4 mejor que Scrum.

Tabla 2. Aplicación del Framework a Scrum y XP

Postulados	P1			P2			P3			P4		
Metodología	P1.1	P1.2	total	P2.1	P2.2	total	P3.1	P3.2	total	P4.1	P4.2	total
Scrum	5	-3	2	4	-2	2	5	0	3	4	-3	1
XP	5	-2	3	5	-2	3	5	0	5	5	-2	3

6 Conclusiones y Trabajo Futuro

Varios trabajos existen en la literatura que comparan metodologías ágiles. En lo que respecta a nuestro conocimiento, todas ellas se basan en comparaciones cualitativas. Abrahamsson et al [14] define una lista de palabras claves y evalúa varias metodologías en base a dicha lista. Las palabras claves incluyen: estado de desarrollo del método, puntos importantes, características especiales, adopción y el grado de soporte de la metodología para las actividades tradicionales del proceso de desarrollo. Iacovelli y Souveyet [15] definen un framework de evaluación en base a cuatro atributos de alto nivel: capacidad de agilidad, uso, aplicabilidad, y proceso y productos. Strode [16] define un framework de comparación que incluye los siguientes atributos: filosofía de la metodología, modelos, técnicas, herramientas, entregables, práctica y el grado de adaptabilidad a una situación. Visconti y Cook [17] analizan de qué manera XP y Scrum satisfacen los principios del manifiesto ágil. Luego de concluir que ninguna de ellas satisface totalmente los principios proponen una metodología combinando aspectos de ambas. Todos estos frameworks existentes son cualitativos. Si bien casi todos, de alguna manera incluyen un análisis sobre la forma en la cual las metodologías cumplen con el Manifiesto Ágil, lo hacen de manera cualitativa. La principal contribución de este trabajo es la definición de un framework de evaluación cuantitativo para evaluar de qué manera las metodologías ágiles cumplen con los postulados del Manifiesto Ágil. En base a la evaluación se puede concluir que XP satisface los postulados ágiles mejor que Scrum.

Trabajos a futuro incluyen extender el framework para medir el cumplimiento de los principios del Manifiesto Ágil, aplicar el framework a otras metodologías ágiles, e incluir en el framework atributos organizacionales para adopción de las metodologías.

Referencias

1. Jacobson, I., et al, *The Unified Software Development Process*, Addison-Wesley (1999).
2. IABG, *The V-Model*, http://www.v-modell.iabg.de/.
3. Boehm, B., *Software Engineering*, IEEE Transactions on Computers, 1226-1241 (1976).
4. Agile Alliance, http://www.agilealliance.org/.
5. Beck, K., et.al, *Manifesto for Agile Software Development*, http://agilemanifesto.org/.
6. *Scrum Alliance*, http://www.scrumaliance.org
7. Cockburn, A, *Crystal Clear a Human-powered Methodology for Small Teams*, (2004).
8. Stapleton J. *"DSDM Dynamic Systems Development Method: The Method in Practice"*. Addison-Wesley, (1997).
9. Highsmith J., Orr K. *Adaptive Software Development. A Collaborative Approach to Managing Complex Systems*, Dorset House (2000).
10. *Feature Driven Development*, http://www.featuredrivendevelopment.com
11. Beck, K. *Extreme Programming Explained. Embrace Change*, Pearson Education, 1999.
12. Berka, K., *Measurement: Its Concepts, Theories and Problems (Boston Studies in the Philosophy of Science),* Kluwer, Vol. 72 (1982).
13. Fenton, N, Pfleeger, S.L., Software Metrics: A Rigorous and Practical Approach, 2nd Edition, PWS Publishing Co, EEUU, ISBN 0534954251 (1998).
14. Abrahansson, P., Salo, O., Ronkainen, J.,Warsta, J., *Agile Software Development Methods, Review and Analysis*, VTT Publications 478 (2002).
15. Iacovelli, A., Souveyet, C., *Framework for Agile Methods Classification*, Workshop on Model Driven Informat. Systems Eng.: Enterprise, User and System Models (2008).
16. Strode, D.E., *The Agile Methods: An Analytical Comparison of Five Agile Methods and an Investigation of Their Target Environment*, MSc Thesis in Information Systems, Massey University, Palmerstin North, Nueva Zelanda (2005).
17. Visconti, M., Cook, C., *An Ideal Process Model for Agile Methods*, LNCS, ISBN 978-3-540-21421-2, Vol 3009, pp.439-441 (2004).