Aplicación de Scrum en la construcción de un simulador de Redis

Alex F. Armijos[a], Angel Fiallos[a], Mónica Villavicencio, Cristina L. Abad

[a] Facultad de Ingeniería en Electricidad y Computación, Escuela Superior Politécnica del Litoral, Km. 30.5 Vía Perimetral, Guayaquil, Ecuador
afarmijos@espol.edu.ec, anfiallos@fiec.espol.edu.ec, mvillavi@espol.edu.ec, cabad@fiec.espol.edu.ec

Resumen

Este artículo expone un caso de estudio sobre cómo la metodología Scrum puede ser aplicada en proyectos científicos, específicamente un simulador Redis. Los resultados de este estudio, en conjunto con la experiencia profesional de quienes participaron en el proyecto, permitieron establecer diferencias de la aplicabilidad de Scrum en dos tipos de proyectos: desarrollo de aplicaciones para empresas y desarrollo de aplicaciones científicas para la academia. La contribución de este artículo se basa en presentar dichas diferencias. Más casos de estudio son requeridos para confirmar el conjunto de diferencias que fueron identificadas de forma empírica.

Palabras clave: Scrum, metodologías ágiles, redis, simulación

1 Introducción

El presente trabajo expone la experiencia de desarrollar un simulador usando el modelo ágil Scrum. El objetivo del simulador es obtener datos relacionados al rendimiento de los procesos de balanceo de carga y al reemplazamiento de la caché en un sistema de caché distribuida. El software desarrollado está en capacidad de realizar simulaciones aleatorias o de reproducción de rastros en múltiples clientes y servidores.

La necesidad de desarrollar el mencionado aplicativo nace con el fin de afianzar los conocimientos relacionados al manejo de caché distribuida. Los servicios de caché (manejado por Redis: http://redis.io) deben ser probados simulando un alto volumen de transacciones - como si estuviese en un ambiente de producción - para mantener su continuidad y disponibilidad aún en los momentos picos.

El simulador Redis fue un trabajo realizado en el curso de Sistemas Operativos Avanzados dictado en la maestría de ciencias computacionales de la Escuela Superior Politécnica del Litoral. Para la ejecución del proyecto del simulador de Redis, se contó con la participación de dos maestrantes - desarrolladores de software con experiencia profesional (software para el gobierno y para la empresa privada). Ambos maestrantes

habían utilizado en sus trabajos metodologías de desarrollo. Uno de ellos estaba familiarizado solamente con la metodología cascada (tradicional); mientras que el otro tenía experiencia usando tanto cascada y Scrum (ágil).

Por ser un proyecto académico, existía la limitante de ser desarrollado en un corto periodo de tiempo (desde mediados de junio a inicios de agosto del 2015) y con sólo 2 desarrolladores.

En las siguientes secciones explicaremos cómo usamos Scrum para este proyecto. La sección 2 describe el problema a resolver; la 3 explica brevemente las metodologías ágiles - poniendo énfasis en Scrum y su uso en la academia. Seguidamente, la sección 4 detalla la metodología y la 5 presenta los resultados. La sección 6 lista las diferencias identificadas de usar Scrum en proyectos para empresas y para la academia. Finalmente, la sección 7 concluye el artículo.

2. Descripción del Problema

Ante la problemática suscitada por las limitaciones del proyecto (2 desarrolladores, 7 semanas de tiempo), surgió la necesidad de adoptar un marco de trabajo que permita desarrollar el simulador en el menor tiempo posible, de forma organizada y con esfuerzo reducido. Simulador que debía caracterizarse por ser de fácil mantenimiento para posteriores actualizaciones.

Para definir el marco de trabajo a ser usado en el proyecto, se evaluaron en primera instancia las metodologías tradicionales (1). Con respecto a estas metodologías, los dos maestrantes ya habían experimentado en sus trabajos los siguientes problemas:

1. Un alto número de reglas y prácticas que debían seguirse en las etapas del desarrollo; sumadas a toda la documentación que éstas exigen. Los desarrolladores indicaron que – regularmente - ellos necesitaban saltarse actividades para cumplir con la entrega a tiempo del producto solicitado en sus empresas; lo que muchas veces impactaba la calidad del software producido. Poca respuesta al cambio y flexibilidad para aceptar ajustes a los requerimientos; lo cual involucraba la modificación de documentos de análisis, diseño y código fuente necesario para adaptar el cambio.

2. Por todo el tiempo que estos cambios involucraban en sus organizaciones, en muchas ocasiones debieron colocar nuevas fechas en el cronograma; lo que tenía un impacto en el presupuesto y la satisfacción del cliente.

Por lo expuesto, la selección de la metodología fue orientada a la búsqueda de un punto intermedio entre las metodologías de desarrollo tradicionales y la ausencia de

metodología alguna (caso común cuando se dispone de poco tiempo y personas). Por dicha razón fue escogido Scrum como metodología de desarrollo ágil.

Las metodologías ágiles han asumido el cambio como algo inevitable y son más adaptables que predictivas (2). Éstas plantean otra forma de administrar el desarrollo de software, lo que facilita el manejo de nuevos requerimientos a medida que éstos aparecen; en lugar de pretender analizar las necesidades del usuario de manera perfecta.

Por ambos desarrolladores, era muy conocido que no se puede esperar que no se produzcan cambios en los requerimientos porque ambos habían gestionado cambios de proyectos ya en marcha.

3. Metodologías Ágiles

Los actuales métodos de desarrollo ágil, formalizados en el Manifiesto para el Desarrollo de Software Ágil (3), fueron conocidos durante su evolución como métodos ligeros, por contraposición a los métodos tradicionales, pesados o burocráticos. Uno de los principios del manifestó ágil es acoger cambios en los requisitos, no importa cuando éstos se presenten.

Si consideramos que los métodos ágiles asumen el cambio como algo inevitable, su aplicación permitirá el ajuste de requisitos o incorporación de nuevos cuando éstos surjan; en vez de analizar y modelar -de forma tan exhaustiva- los requisitos del cliente recopilados al inicio del proyecto.

Una diferencia obvia entre este tipo de métodos (ágiles) y los tradicionales (o pesados) es que generan mucha menos documentación (4). Los ágiles están centrados en el código, de tal forma que éste se convierte en la principal documentación del proyecto. Sin embargo, esta llamativa característica no es la principal diferencia entre ambos. Lo que esta ausencia de documentación pone de manifiesto son dos diferencias mucho más significativas (1). Primero, los métodos pesados tienden a planificar con gran detalle todo el proceso de desarrollo, pero esto sólo funciona mientras las cosas no cambien. Frente a esta resistencia al cambio, los métodos ágiles al ser más adaptables que predictivos, asumen que éste se va a producir y se preparan para recibirlo. Segundo, los métodos ágiles se centran más en las personas que en el proceso, explicando así la necesidad de tener en cuenta la naturaleza de las personas en vez de ir en contra de la misma.

Los métodos de desarrollo ágiles constituyen -por sí mismos- una respuesta a la necesidad de flexibilidad en los cambios; pero como métodos que son, lo hacen fundamentalmente desde la perspectiva de la gestión de proyectos. De acuerdo a las estadísticas comparativas del Standish Group (5) –ver Figura 1-, el uso de las metodologías ágiles permite que los proyectos tengan mayores posibilidades de implementarse con éxito, en comparación con el uso de metodologías tradicionales.

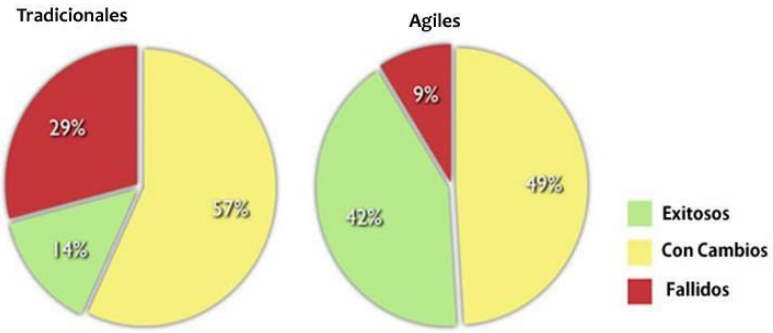

Figura. No.1. Situación de Proyectos de TI ejecutados con metodologías tradicionales y ágiles. Fuente: Manifiesto del CAOS, 2012 [3]

El mismo reporte: "Manifiesto del Caos" (5), establece como factor de éxito que el software debe ser construido en iteraciones cortas, con equipos de trabajo pequeños y focalizados. Además, el reporte indica que el equipo de proyecto debe entregar funcionalidades al usuario en cada iteración y considerar los cambios como normales.

3.1 Scrum

Scrum plantea un proceso de desarrollo de software iterativo y creciente. Aunque inicialmente, Scrum estaba enfocado a la gestión de procesos de desarrollo de software, éste puede también ser utilizado en equipos de mantenimiento de software (6).

Scrum es un modelo que se basa en lograr objetivos de trabajo, integrando las actividades de análisis, diseño y pruebas en períodos muy cortos de revisión, modificación y control con el usuario. Esto facilita el avance de trabajos y la entrega formal de las funcionalidades planificadas para una iteración (conocida como Sprint). Por sus características, Scrum es óptimo para equipos de trabajo de hasta 9 personas, aunque hay casos de éxito con equipos más grandes (7).

Existen tres aspectos fundamentales a diferenciar en Scrum que son: los roles, artefactos y eventos.

Los roles (los que ejecutan las acciones) contemplados en este modelo son los siguientes (9):

- ***Dueño del Producto o Cliente (Product Owner)***.- Conoce y determina las prioridades del proyecto o producto.

- ***Facilitador (Scrum Master).-*** Es la persona que asegura el seguimiento de la metodología, guiando las reuniones y ayudando al equipo ante cualquier problema que aparezca.
- ***Equipo Scrum.-*** Son los encargados de implementar la funcionalidad seleccionada por el dueño del producto.
- ***Usuarios.-*** Son los beneficiarios finales del producto, y son quienes verifican los progresos, aportan con ideas, sugerencias o dan a conocer necesidades.

Los artefactos son:

- ***Lista de objetivos (Product Backlog).-*** Es una lista de objetivos/requisitos priorizada que representa la visión y expectativas del cliente respecto a los objetivos y entregas del producto o proyecto. El dueño del producto o cliente es el responsable de crear y gestionar la lista con la ayuda del Facilitador y del equipo Scrum.
- ***Gráficos de trabajo pendiente (Burndown charts***).- Este gráfico muestra la velocidad a la que el equipo está completando los objetivos/requisitos. Permite extrapolar si el Equipo podrá completar el trabajo en el tiempo estimado.
- ***Lista de tareas de la iteración (Sprint backlog*).-** Se refiere a la lista de tareas que el equipo elabora en la reunión de planificación de la iteración (Sprint planning) como plan para completar los objetivos y requisitos seleccionados para dicha iteración. Esta lista permite ver las tareas donde el equipo está teniendo problemas (Ej.: no avanzan), facilitando la toma de decisiones al respecto.

Los eventos son:

- ***Reunión de Planificación de la iteración (Sprint planning meeting).-*** Se refiere a reuniones cuyo objetivo es planificar la lista de tareas de la iteración a partir de la lista de objetivos. Suelen participar el facilitador, el equipo y el dueño del producto o cliente.
- ***Revisión de la iteración (Sprint Review).-*** Una vez finalizada una lista de tareas de la iteración (lista de sprint), un miembro del equipo muestra los avances al cliente o usuario.
- ***Retrospectiva de la iteración (Sprint Retrospective).-*** Una vez que la iteración ha concluido, el dueño del producto revisa con el equipo de trabajo los objetivos marcados inicialmente en la lista de tareas. Se analizan los aspectos positivos y negativos; y de ser necesario se hacen cambios para la próxima iteración.

3.2 Scrum en la academia

Existen casos de estudios sobre el uso de Scrum como marco de trabajo para el desarrollo de aplicaciones en el ámbito empresarial y académico (9) (10) (11) (12) (13) (14) (15) (16) (17) (18) (19). En el ámbito académico, existen muchos aportes significativos que el marco de trabajo Scrum ha proporcionado al desarrollo de

proyectos realizado en asignaturas. Así lo describen Groena, Guoc, Grogand, Schillera y Osbornee (2014), quienes exponen que Scrum ha sido utilizado con estudiantes por su flexibilidad. Un ejemplo concreto de su utilización es el proyecto HemeLB (21), un sistema de simulación de la circulación sanguínea en el cerebro desarrollado por estudiantes de University College London. Así también, Yuen (21) (de la Universidad de Texas en San Antonio), presenta el desarrollo de un software de aprendizaje (con simuladores para matemáticas, ciencias naturales y computación, juegos e incluso con asistencia para niños con necesidades especiales) que apoya la educación en dicha institución.

Por mencionar otro ejemplo, Jackson y Ellis publicaron un caso de estudio del desarrollo de un software open source para trabajo humanitario colaborativo con el fin de fomentar el mejoramiento de la sociedad a través del uso de la tecnología. Este software fue desarrollado en el curso Computer Science Capstone de la carrera de Ciencias de la Computación (12). Para concluir los ejemplos, debemos anotar que los autores Weissberger, Qureshi, Chowhan, Collins y Gallimore trabajaron con estudiantes de dos universidades (una canadiense y otra finlandesa) para desarrollar Agilfant -una herramienta web para gestionar proyectos realizados con el paradigma ágil (22).

Estas publicaciones demuestran que Scrum puede aplicarse en diferentes entornos, para lo cual se hace necesario adaptarlo. Para el proyecto web Angilfant (22), los miembros del equipo se reunían dos veces por semana y su reunión duraba quince minutos. En el proyecto humanitario (12), las reuniones eran una vez a la semana y demoraban dos horas con cuarenta minutos. Originalmente, Scrum está pensado para grupos de la misma localidad y espacio de trabajo; sin embargo, ha sido aplicado en equipos remotos.

El proyecto del simulador Redis, objeto de discusión de este artículo, también necesitó adaptar Scrum para acoplarse a las limitaciones del entorno académico en el cual fue desarrollado. Las adaptaciones fueron realizadas de acuerdo a nuestras necesidades y recursos disponibles. Cabe anotar que, en la literatura revisada no encontramos un proyecto de orientación científica similar al Simulador Redis; por tanto, éste es otro ejemplo distinto de aplicación de Scrum.

4. Metodología

La Fig. 2 muestra una vista general de cómo Scrum fue usado en el proyecto del simulador Redis.

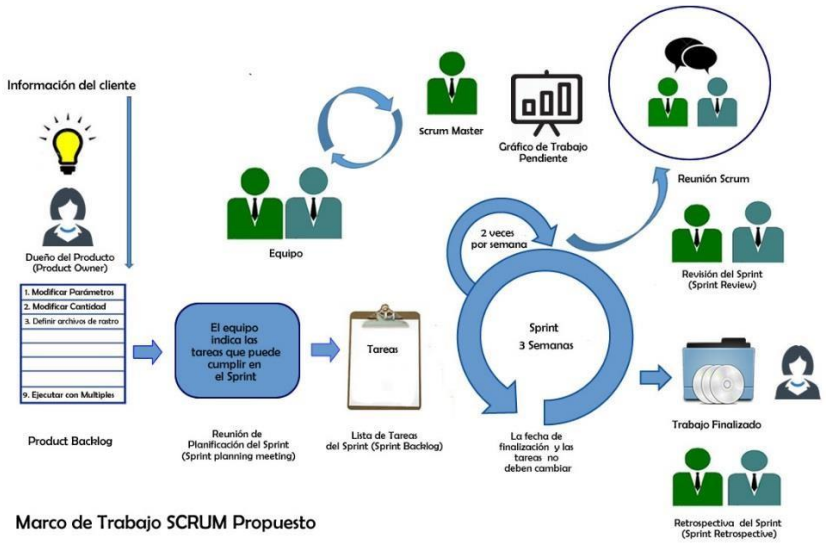

Figura. No.2. Marco de Trabajo Scrum propuesto para el desarrollo del proyecto Simulador Redis – Adaptado de (23).

.

En esta figura podemos notar que el proceso de desarrollo con Scrum parte de la lista de objetivos (Product back log). Esta lista es priorizada por el dueño del producto, rol que fue asumido por el profesor de la materia, quien tiene una amplia experiencia haciendo investigación en sistemas distribuidos. Por cada historia de usuario, los criterios de aceptación fueron definidos.

Una vez realizada la priorización, y con el fin de planificar y decidir el tiempo que tomaría construir los requisitos en cada sprint, los desarrolladores hicieron la estimación de las historias de usuario mediante la técnica del *planning póker*. Uno de los desarrolladores (el que tenía experiencia con Scrum) asumió el rol de Facilitador (Scrum master). A cada historia de usuario se le asignó una complejidad relativa medida en score points, dando un total de 87 score points para todo el simulador. Inicialmente, el equipo planificó tres Sprints en un lapso de 7 semanas; el primer sprint de 34 score points. Sin embargo, durante el desarrollo del simulador, la fecha de entrega del proyecto fue modificada por el dueño del producto. En lugar de 7 semanas, el plazo se redujo a 6. Con la experiencia adquirida en el primer sprint, el equipo decidió terminar las historias de usuario restantes en un segundo y último sprint de 53 score points. Todo el registro de las historias de usuario, la elaboración del Product y Sprint backlog así como otros artefactos que se utilizan en Scrum fueron realizados en Microsoft Excel.

Durante la ejecución de cada iteración (Sprint), el equipo se reunió treinta minutos dos veces por semana (los días en los que tenían que asistir al curso de Sistemas Operativos Avanzados). El objetivo de las reuniones era conocer de primera mano los

obstáculos que se presentaban a lo largo del desarrollo de la iteración y predecir la existencia de algún problema que impida culminar todas las historias planificadas en las iteraciones. El burndown chart fue utilizado para el monitoreo del avance de las historias de usuario y se presentará más adelante, en la sección de resultados.

Cada tres semanas, cuando culminaba el sprint, se realizaba la reunión de retrospectiva. Reunión en la que se identificaban problemas generales de la iteración y se planteaban mejoras. Considerando que las metas de las historias de usuario fueron claramente definidas por el dueño del producto, éste no participaba de estas reuniones, pero era informado de las decisiones que se tomaban.

Una vez culminados los dos Sprints, el simulador fue presentado al dueño del producto, quien lo revisó, realizó las pruebas de aceptación (considerando los criterios de aceptación) y solicitó pequeños cambios. Cambios que fueron rápidamente implementados (2 horas); de tal forma que el producto fuera formalmente entregado y considerado como finalizado.

A continuación, resumimos los cambios realizados a la metodología Scrum en este proyecto:

- Tamaño del equipo: 3 personas.
- Dueño del producto: Profesor del curso (investigador en el área de sistemas distribuidos).
- Facilitador: Estudiante de maestría con experiencia profesional en Scrum. Adicionalmente, se desempeñó como desarrollador.
- Desarrollador: Estudiante de maestría con experiencia profesional usando metodologías tradicionales.
- Reuniones diarias: 2 veces a la semana.
- Reuniones de revisión: Sólo al final, cuando el simulador estuvo terminado.
- Reuniones de retrospectiva: participaban los 2 desarrolladores y era realizada al final de cada sprint.

5. Resultados

Como fue indicado en la sección anterior, el simulador Redis fue desarrollado en 6 semanas. La Tabla 1 muestra el product backlog; es decir las historias de usuarios junto a la complejidad asignada y el sprint en el cual se lo realizó.

Tabla No. 1. Product backlog y los sprints

Product Backlog – Sprints					
No.	**Como**	**Deseo**	**Con el fin de**	**Complejidad**	**Sprint**
1	Tester	Modificar el parámetro de modo de ejecución	Asignar una ejecución manejada por rastros o aleatoria	3	1
2	Tester	Modificar la cantidad de clientes	Simular el rendimiento del servidor Redis con más de 1 cliente	8	1
3	Tester	Modificar los parámetros de generación aleatoria por operación	Asignar una ponderación a cada operación en la ejecución aleatoria	5	1
4	Tester	Definir archivos de rastros de clientes dado por: operación, clave, cantidad de bytes del valor	Asignar la entrada para cada cliente en la ejecución manejada por rastros	5	1
5	Tester	Ejecutar los clientes en paralelo	Simular la medición del rendimiento	13	1
6	Tester	Ver el tiempo promedio por operación	Medir el rendimiento por operación	20	2
7	Tester	Ver el tiempo promedio por tipo de operación	Medir el rendimiento por tipo de operación	8	2
8	Tester	Ver la cantidad de operaciones realizadas por cliente	Observar el volumen de operaciones generado	5	2
9	Tester	Ejecutar el simulador con múltiples servidores	Demostrar la utilidad con más de un servidor	20	2
			Total	87	

En la Figura 3 presentamos el gráfico conocido como *burndown chart*, el cual fue elaborado y actualizado durante las seis semanas que tomó este proyecto. En esta figura se evidencia una desviación inicial en cada sprint con respecto a lo ideal; sin embargo, también se corrobora que el equipo Scrum recuperó los score points pendientes, llegando finalmente a cumplir los límites de tiempo exigidos por el dueño del producto. Como se puede observar, esta recuperación se dio tanto en el primer sprint como en el segundo.

Una ventaja que el marco de trabajo Scrum nos proporciona es que establece puntos claros en el tiempo donde debemos capturar datos de la gestión del proyecto con el fin de medir avances y definir acciones correctivas. En el caso de nuestro proyecto, estas bondades también fueron aprovechadas.

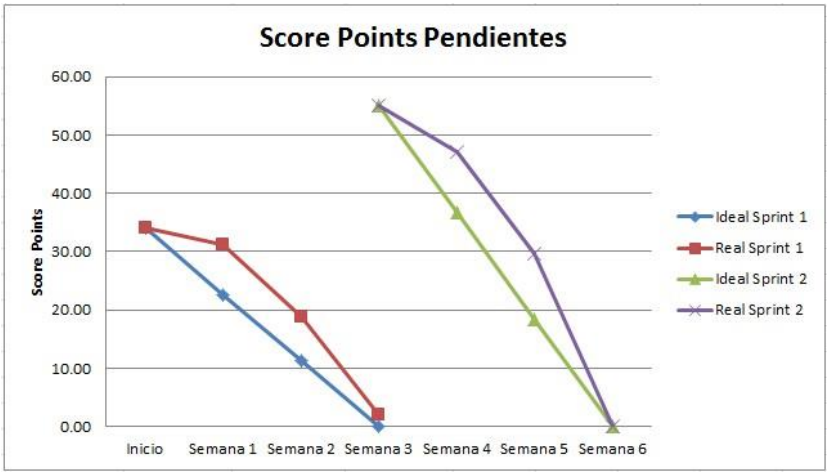

Figura. No.3. Burndown Chart del Proyecto. Abarca Sprint 1 y 2 medidos en score points pendientes de resolver.

Para desarrollar el simulador, fue necesario instalar el sistema operativo Windows 7 en dos equipos, incluyendo los siguientes componentes: Java - J2SE 6 (JDK 6), Eclipse, Apache Server, Redis Server y librerías Redis Client. En la Figura 4 se muestran las capturas de pantalla del simulador Redis obtenidas en cada uno de los sprints.

Simulador al final del Sprint 1	**Simulador al final del Sprint 2**
Iniciando Prueba Clientes ejecutandose en paralelo:3 Clientes ejecutados en total:3 Cliente:nodo3:Op:get duracion:214871071 Resultado:HiloResultadoCliente3:Token:-1 Resultado:HiloResultadoCliente3:Cogi el token:3 Resultado:HiloResultadoCliente3:Se anadio Resultado:2148716 Resultado:HiloResultadoCliente3:Se dejo el token:-1 Cliente:nodo2:Op:get duracion:275177421 Cliente:nodo1:Op:set duracion:275511481 Resultado:HiloResultadoCliente2:Token:-1 Resultado:HiloResultadoCliente2:Cogi el token:2 Resultado:HiloResultadoCliente2:Se anadio Resultado:2751774 Resultado:HiloResultadoCliente2:Se dejo el token:-1 Resultado:HiloResultadoCliente1:Token:2 Resultado:HiloResultadoCliente1:Cogi el token:1 Resultado:HiloResultadoCliente1:Se anadio Resultado:2755114 Resultado:HiloResultadoCliente1:Se dejo el token:-1 Cliente:nodo1:Op:get duracion:122726363 Resultado:HiloResultadoCliente1:Token:-1	Finalizando Prueba Mostrando Resultados Cantidad Tiempos Registrados:300 Tiempo Promedio de Operaciones:122194971,12 Cantidad Tiempos Operacion(0):120 Tiempo promedio por Operacion(get):96666209,1 Cantidad Tiempos Operacion(1):180 Tiempo promedio por Operacion(set):139214145,8 Cantidad Operaciones Procesadas - Cliente (1): 30 Cantidad Operaciones Procesadas - Cliente (2): 30 Cantidad Operaciones Procesadas - Cliente (3): 30 Cantidad Operaciones Procesadas - Cliente (4): 30 Cantidad Operaciones Procesadas - Cliente (5): 30 Cantidad Operaciones Procesadas - Cliente (6): 30 Cantidad Operaciones Procesadas - Cliente (7): 30 Cantidad Operaciones Procesadas - Cliente (8): 30 Cantidad Operaciones Procesadas - Cliente (9): 30 Cantidad Operaciones Procesadas - Cliente (10): 30

Figura. No.4. Capturas del Simulador en ambos sprints

6. Diferencias de aplicar Scrum en los entornos académico y empresarial.

Lo que nos motivó a escribir este artículo, particularmente esta sección, fue el hecho de **no** experimentar –por primera vez- la frustración de enfrentarnos a cambios de última hora requeridos por el cliente (incorporar historias de usuarios adicionales a las solicitadas). Este sentimiento de logro, sumado al hecho de que el proyecto culminó a tiempo y sin mayores complicaciones; nos llevó a analizar los factores que influyeron en el resultado obtenido.

La Tabla 2 resume las diferencias identificadas entre desarrollar aplicaciones en entornos académicos y empresariales. Diferencias que se tornan visibles para profesionales que han trabajado varios años desarrollando sistemas para instituciones públicas o privadas, y que ahora –por estudios de posgrado- tienen el reto de desarrollar aplicaciones de índole científico para la academia.

Tabla No. 2. Características diferenciadoras de aplicar Scrum en entornos académico y empresarial.

Scrum aplicado en nuestro entorno (proyecto científico-académico)	**Scrum en el entorno empresarial**
El producto es de carácter científico.	El producto permite la operación del negocio o agrega valor al mismo.
El PO (Product Owner) es un experto (científico) en el área sobre la que rige el producto.	El PO no es necesariamente un experto en el área de negocio sobre la que rige el producto.
El PO visualiza con mucha mayor claridad el producto que desea obtener.	El PO necesita apoyo de los técnicos para visualizar el producto.
Las historias de usuario están claramente concebidas y pocas consultas son realizadas al PO para afinarlas.	Las historias de usuario necesitan ser afinadas; se requieren varias reuniones con el PO para afinarlas.
Los miembros del equipo trabajan en espacios físicos distintos (casa, biblioteca, etc.)	Los miembros del equipo – generalmente- trabajan en las instalaciones de la organización a la que pertenecen.
Los miembros del equipo tienen horarios distintos, lo que dificulta realizar reuniones de trabajo.	Los miembros del equipo – generalmente- tienen el mismo horario de trabajo, lo que facilita el encuentro entre ellos.
Se torna muy difícil hacer reuniones diarias; en este caso se realizaron 2 reuniones semanales.	Se realizan reuniones diarias para identificar problemas y conocer el estado del proyecto (dónde estamos en la iteración).
Los miembros del equipo disponen de pocas horas al mes para realizar un proyecto (40 horas por miembro del equipo para este proyecto).	Los miembros del equipo –generalmente- trabajan a tiempo completo para la institución que requiere el proyecto (160 horas al mes por miembro del equipo).

Como se puede notar en la Tabla 2, una de las diferencias notables de aplicar Scrum para proyectos científicos reside en el hecho de que el dueño del producto es un experto con mucho conocimiento de la problemática a resolver y del producto que espera recibir. La combinación conocimiento-experiencia del dueño del producto reduce significativamente la necesidad de depurar las historias de usuario y de realizar reuniones constantes. Por la experiencia vivida en este proyecto, consideramos que esta es una ventaja que –ciertamente- tienen los proyectos de estas características. Es decir, proyectos en los que no se presentan cambios, o si se dan, son de muy bajo impacto.

En el campo empresarial, la realidad es diferente. En general, el cliente no sabe exactamente lo que quiere o espera recibir como producto hasta que ve y opera el software entregado. Consideramos que las diferencias descritas en la Tabla 2 pueden ser sujeto de estudio en futuros trabajos de investigación, ya que, son varios factores los que podrían influir en la adaptación de la metodología Scrum para usarla en el desarrollo de software de carácter científico. Por ende, nuevas diferencias pudieran ser observadas con respecto al desarrollo de aplicaciones empresariales.

En el entorno académico/científico, las adaptaciones a la metodología Scrum pudieran ser diferentes a las realizadas por nuestro equipo de trabajo. Consideramos que, las adaptaciones obedecen a las características del proyecto como, por ejemplo: el tamaño del software, los recursos disponibles, el área de investigación, la localización del equipo de trabajo, los modos de comunicación, entre otros factores. Factores que pueden ser motivo de análisis para futuros estudios.

7. Conclusiones y Recomendaciones

El presente artículo presentó adaptaciones que fueron realizadas a la metodología Scrum para desarrollar un simulador Redis como proyecto del curso de la materia Sistemas Operativos Avanzados. El proyecto se completó en dos iteraciones de tres semanas con un equipo de 3 personas. El esfuerzo total fue de 120 horas de trabajo, obteniéndose un producto de software de 1043 líneas de código fuente.

Se puede concluir que la selección de las metodologías ágiles, en particular el uso del modelo Scrum –adaptado a las necesidades de este proyecto, nos permitió desarrollar un producto con flexibilidad y sin mayor afectación a los plazos de entrega. Se recomienda hacer un análisis del producto, de los recursos disponibles y del entorno para hacer adaptaciones de Scrum o de cualquier metodología de desarrollo.

En relación a las diferencias entre desarrollar software científico versus empresarial usando Scrum - mencionadas en la sección 6, podemos recalcar que es recomendable el uso de las metodologías ágiles para la ejecución de proyectos de desarrollo de software de experimentación científica para los cuales:

1. Existan -desde un inicio- especificaciones funcionales claras y precisas
2. Existan equipos de trabajo pequeños que trabajen de manera remota;
3. Tengan una fecha de finalización no negociable (duración de un curso universitario)
4. Exista un dueño del producto con pleno conocimiento de lo que necesita obtener en un tiempo limitado.

Es necesario continuar con la realización de más casos de estudios de la aplicación de Scrum y ampliar el debate que despierta la ejecución de sus disciplinas en entornos académicos y empresariales. Solo la experiencia hace que los marcos de trabajo pongan a prueba su madurez y flexibilicen, de ser necesario, sus restricciones en el tiempo.

8. Referencias

1. Fowler, Martin. Blog de Martin Fowler. [Online] 2005. http://martinfowler.com/articles/newMethodology.html.

2. Fowler, M.: Blog de Martin Fowler. [Online] http://martinfowler.com/bliki/EvolutionarySOA.html (2008).

3. Beck, K.: Manifesto for agile software development. Technical Report (2001).

4. Stahl, T., Voelter, M.: Model-Driven Software Development: Technology, Engineering, Management. Wiley, ISBN 978-0470025703 (2006).

5. Flower, M.: Blog de Martin Fowler. [Online] http://martinfowler.com/articles/newMethodology.html (2005).

6. Group, Standish: Chaos Manifesto. (2012).

7. Schwaber, K.: Scrum Development Process. ISBN 978-3-540-76096-2 (1997).

8. James, M.: Scrum Reference Card. [Online] http://scrumreferencecard.com/ScrumReferenceCard.pdf (2012).

9. Schwaber, Ken. Scrum. [Online] 2009. http://www.controlchaos.com/.

10. Nils Brede, M: .A teamwork model for understanding an agile team: A case study of a Scrum project. Information and Software Technology, ISSN 0950-5849. (2010).

11. Paasivaara, Maria.: Using scrum in a globally distributed project: a case study. Wiley InterScience . (2008).

12. Rover, D: Advantages of agile methodologies for software and product development in a capstone design project. INSPEC Accession Number: 14934586 DOI:10.1109/FIE.2014.7044380 (2014).

13. Jackson, S., Ellis, H.: Supporting HFOSS using scrum in a capstone course. ACM SIGCAS Computers and Society. Vol. 45. Issue 2. Doi: 10.1145/2809957.2809968 (2015).

14. Lisi Romano, B.; Delgado Da Silva, A.: Project Management Using the Scrum Agile Method: A Case Study within a Small Enterprise. Information Technology - New Generations (ITNG), 2015 12th International Conference on. DOI: 10.1109/ITNG.2015.139 (2015).

15. Chandramouli, S.; Kumar, R.: A study on Agile Scrum Engineering practices. IIMS Journal of Management Science. Vol.6 Issue 1. DOI : 10.5958/0976-173X.2015.00005.6 Print ISSN : 0976-030X. Online ISSN : 0976-173X. (2015).

16. Abeer A., Rizwan Jameel Qureshi, M.: The Proposal of Scaling the Roles in Scrum of Scrums for Distributed Large Projects. I.J. Information Technology and Computer Science. DOI: 10.5815/ijitcs.2015.08.10 (2015).

17. Lima Peres, A., Lemos Meira, S.: Towards a framework that promotes integration between the UX design and SCRUM, Aligned to CMMI. Information Systems and Technologies (CISTI), 2015 10th Iberian Conference. DOI: 10.1109/CISTI.2015.7170443 (2015).

18. Sampaio Machado, T., Pinheiro, P., Tamanini, I.: Project management aided by verbal decision analysis approaches: a case study for the selection of the best SCRUM practices. Issue International Transactions in Operational Research International Transactio. DOI: 10.1111/itor.12078 (2015).

19. Paweł, R., Dorota, K.: Implementing Scrum Method in International Teams—A Case Study. Open Journal of Social Sciences. DOI: 10.4236/jss.2015.37043 (2015).

20. Prashant, C.: Analysis of Students' Learning for Efficient Task Assignment in a Distributed Scrum Setting. Maharshi Dayanand University. (2011).

21. Groena D., Guoc X., Grogand J., Schillera U., Osbornee M: Software development practices in academia: a case study comparison. In Proceddings of Cornel University Library (2015).

22. Yuen, T.: Scrumming with Educators: Cross-Departmental Collaboration for a Summer Software Engineering Capstone in. Learning and Teaching in Computing and Engineering (LaTiCE), 2015 International Conference on. pp. 124 – 127 (2015).

23. Weissberger, I., Qureshi, A., Chowhan, A., Collins, E., Gallimore, D.: Incorporating software maintenance in a senior capstone project. International Journal of Cyber Society and Education. Vol. 8 Issue 1. pp. 31-38 (2015).

24. Agile For All: Intro to Agile. http://35qk152ejao6mi5pan29erbr9.wpengine.netdnacdn.com/wp-content/uploads/2013/05/Scrum_Framework.jpg . [Online]

25. Schwaber, K. Agile Project Management with Scrum. s.l. : ISBN-13: 978-0735619937 (2009).