Implementación práctica del *agilismo* en proyecto de Ingeniería de Software

Santiago J. Levy[1], J. Ramiro Romero Dapozo[1], Ariel Pasini[2]

[1] Estudiante, Cátedra de Ingeniería de Software – Universidad Nacional de La Plata
Facultad de Informática
50 y 120 La Plata, Buenos Aires, Argentina.
santiagojlevy@gmail.com, ramiroromero94@gmail.com

[2] Profesor, Ingeniería de Software – Instituto de Investigación en Informática LIDI (III-LIDI) - Facultad de Informática – Universidad Nacional de La Plata
50 y 120 La Plata, Buenos Aires, Argentina.
apasini@lidi.info.unlp.edu.ar

Resumen Se presenta una experiencia pedagógica orientada al desarrollo de un proyecto software utilizando una versión adaptada de la metodología Scrum, implementada en la asignatura Ingeniería de Software de la carrera Ingeniería en Computación de la Facultad de Informática de la Universidad Nacional de La Plata. Se describen las funcionalidades de la aplicación Web desarrollada, la adaptación propuesta sobre Scrum, el conjunto de herramientas utilizadas para la aplicación de la metodología y el resultado de la experiencia desde el punto de vista de los alumnos.

Keywords: Scrum, Ágil, Ingeniería de Software, Aprendizaje

1. Introducción

En la primera conferencia de Ingeniería de Software realizada por la OTAN (Organización del Tratado del Atlántico Norte) en 1968 se presentó lo que se conoce como crisis del software. Esta crisis se refiere a las problemáticas más comunes a las que se enfrentan los desarrollos de software. Su aparición dio origen a la Ingeniería de software como disciplina [1].

Desde la identificación de esta crisis, se vienen realizando estudios sobre procesos en el área de la Ingeniería de Software con diferentes planificaciones. Estos tienen el propósito de mejorar la eficiencia en los tiempos de desarrollo y producción de los proyectos de software. Los estudios realizados, por ejemplo, por Standish Group y eGovernment, revelaron numerosas conclusiones respecto a la progresión con que deben abarcarse los proyectos relacionados con la tecnología informática para poder brindar software funcional en tiempos reducidos [2] [3].

Surgen así lo que se conoce como metodologías ágiles: modelos de procesos de desarrollo de software enfocados en la participación colaborativa y la producción acelerada de elementos entregables al cliente. Estas metodologías contribuyen enormemente a una organización más eficiente a la hora de administrar proyectos de desarrollo de software [4].

Este trabajo describe una experiencia pedagógica orientada al desarrollo de un proyecto software utilizando una versión adaptada de la metodología Scrum, implementada en la asignatura Ingeniería de Software de la carrera Ingeniería en Computación (IS-IC) de la Facultad de Informática de la Universidad Nacional de la Plata en 2015.

La forma de Scrum utilizada en esta experiencia será denominada *Scrum IS-IC*.

En la sección 2 se explica el concepto de metodología ágil, haciendo particular énfasis en el cuadro de trabajo Scrum. En la sección 3 se detalla como se dio la experiencia del proyecto de aplicación. Finalmente en la sección 4 se sintetizan las conclusiones obtenidas del proyecto en cuestión.

2. Metodologías ágiles

La organización del desarrollo de proyectos de software presenta problemas particulares derivados de sus paradigmas esenciales. Algunos ejemplos son la incertidumbre sobre cómo se implementará una funcionalidad, los detalles específicos sobre algún requerimiento que quizás en el futuro sea quitado del proyecto, la organización de qué miembros del equipo participarán en el desarrollo de una u otra funcionalidad, etc. Esta serie de contratiempos conllevan un amplio espectro de riesgos, además de la generación de desperdicio de tiempo de desarrollo y de todo tipo de recursos [5].

Las metodologías ágiles buscan proponer un esquema alternativo, donde se minimice la producción de desperdicio en el proceso de desarrollo. Estos avances en la manera de desarrollar software se reflejan en los doce principios del Manifiesto Ágil, cuyos conceptos claves son el valorar [6]:

- **Individuos e interacciones** sobre herramientas y procesos
- **Software funcionando** sobre documentación extensiva
- **Colaboración con el cliente** sobre negociación contractual
- **Respuesta ante el cambio** sobre seguimiento de un plan

El manifiesto ágil no provee pasos concretos. Las organizaciones optan por incorporar directamente métodos más específicos dentro del movimiento ágil, como Crystal Clear, XP (*Extreme Programming)*, FDD (*Feature Driven Development*), Scrum, entre otros [5].

En el cuadro 1 se puede ver una comparación a grandes rasgos entre las metodologías ágiles y las metodologías tradicionales. Las metodologías tradicionales tienden a mantener una estructura más rígida a lo largo del desarrollo, mientras que las metodologías ágiles mantienen una menor carga burocrática y se caracterizan por su flexibilidad ante cambios que puedan ir surgiendo a lo largo del proceso.

2.1. Marco de trabajo Scrum

Scrum es un marco de trabajo ágil y simple para la colaboración en equipo sobre proyectos de software complejos [5].

Cuadro 1: Comparación entre metodologías tradicionales y metodologías ágiles [7]

Metodologías tradicionales	Metodologías ágiles
Predictivos	Adaptativos
Orientado a procesos	Orientado a personas
Proceso rígido	Proceso flexible
Se concibe como un proyecto	Un proyecto es subdividido en varios proyectos más pequeños
Poca comunicación con el cliente	Comunicación constante con el cliente
Entrega de software al finalizar el desarrollo	Entregas constantes de software
Documentación extensa	Poca documentación

Scrum ha resultado ser útil y eficaz en la mayoría de las empresas reales que lo adoptaron, donde el 87 % de las que empezaron a trabajar dentro del marco de trabajo Scrum confirma haber tenido mejoras respecto a sus formas de organización previas [8].

El resto de esta sección se dedica a explicar el funcionamiento a grandes rasgos de Scrum según sus creadores, Ken Schwaber y Jeff Sutherland [9].

El marco de trabajo Scrum está compuesto por los equipos Scrum, roles, eventos, artefactos y reglas que definen relaciones entre estos.

El equipo de Scrum (*Scrum Team*) se compone de un dueño de producto (*Product Owner*), el equipo de desarrollo (*Development Team*) y un *Scrum Master*. Los Equipos Scrum son autoorganizados y multifuncionales.

Los Equipos Scrum entregan productos de forma iterativa e incremental, obteniendo *feedback* entre cada incremento.

Roles de Scrum Los roles definidos por Scrum son:

Dueño de producto (*Product Owner*): Es quien debe encargarse de la gestión de la *product backlog*, expresando con claridad sus elementos y ordenándolos para priorizar los que él considere más relevantes.
El *product owner* debe ser una única persona y puede delegar estas tareas al equipo de desarrollo.

Equipo de desarrollo (*Development Team*): Está formado por los profesionales que deben entregar un incremento de producto terminado, que potencialmente se pueda poner en producción, al final de cada *sprint*.

***Scrum Master*:** Es el responsable de asegurar que Scrum es entendido y adoptado. Los *Scrum masters* hacen eso asegurándose de que el equipo Scrum trabaje ajustándose a la teoría, prácticas y reglas de Scrum.

Eventos de Scrum En Scrum existen eventos predefinidos con el fin de crear regularidad y minimizar la necesidad de reuniones no definidas en Scrum. Ellos son:

El *Sprint*: Es un bloque de tiempo de un mes o menos durante el cual se crea un incremento de producto terminado y utilizable. Los *sprints* consisten y contienen la reunión de planificación del Sprint (*sprint planning meeting*), los Scrums diarios (*daily Scrums*), el trabajo de desarrollo, la revisión del Sprint (*sprint review*) y la retrospectiva del Sprint (*sprint retrospective*).

Reunión de planificación de *sprint* (*Sprint Planning Meeting*): El trabajo a realizar durante el *sprint* se planifica de forma conjunta por el equipo Scrum en este evento.

Scrum Diario (*Daily Scrum*): Es una reunión con una duración de quince minutos para que el equipo de desarrollo sincronice sus actividades y cree un plan para las siguientes 24 horas.

Revisión del Sprint (*Sprint Review*): Al final del *sprint* se realiza una revisión de *sprint* para inspeccionar el incremento y adaptar la lista de producto si fuese necesario. Durante la revisión de *sprint*, el equipo Scrum y cualquier cambio a la lista de producto durante el *sprint*.

Retrospectiva de Sprint (Sprint Retrospective): Es una oportunidad para el equipo Scrum de evaluarse a a sí mismo y crear un plan de mejoras para el siguiente *sprint*.

La retrospectiva de *sprint* se hace después de la revisión de *sprint* y antes de la siguiente reunión de planificación de *sprint*.

Artefactos de Scrum Scrum define los siguientes artefactos:

Lista de producto (*Product Backlog*): La lista de producto es una lista ordenada de todo lo que podría ser necesario en el producto, y debe ser la única fuente de requisitos para cualquier cambio a realizarse en el producto.

Lista de pendientes del Sprint (*Sprint Backlog*): La lista de pendientes del *sprint* es el conjunto de elementos de la lista de producto seleccionados para el Sprint.

A medida que se requiere nuevo trabajo, el equipo de desarrollo lo añade a la lista de pendientes del *sprint*.

Incremento: El incremento es una versión del producto resultante de lo trabajado en un sprint. El incremento debe estar en condiciones de utilizarse sin importar si el dueño de producto decide liberarlo o no.

En la figura 1 se puede ver un esquema que resume de manera gráfica los eventos que se dan en un *sprint* y los artefactos involucrados.

3. Experiencia de uso de Scrum en IS-IC

En el segundo semestre del ciclo lectivo 2015, en la asignatura de Ingeniería de Software de la carrera Ingeniería en Computación de la Facultad de Informática

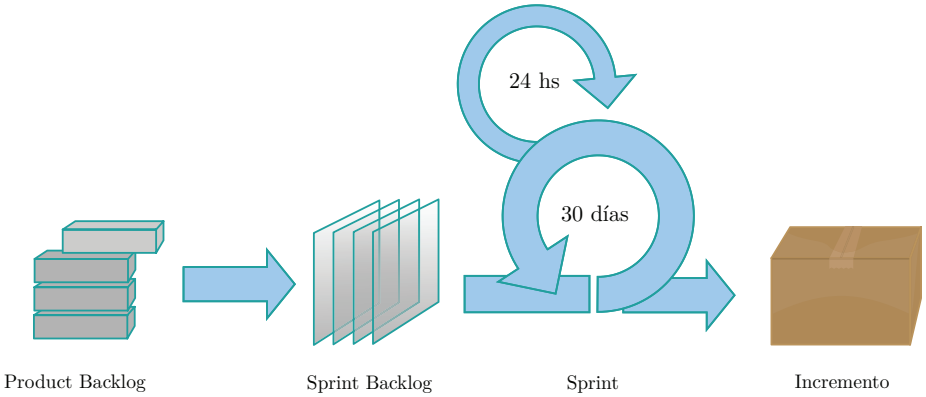

Figura 1: Esquema representativo de la progresión de un *sprint*.

de la Universidad Nacional de La Plata, se les propuso a los alumnos como proyecto el desarrollo de una aplicación Web utilizando el marco de trabajo Scrum. Debido a las modificaciones realizadas sobre lo definido originalmente por Scrum, se referenciará a esta variación de Scrum como *Scrum IS-IC*.

La aplicación Web en cuestión se llama CouchInn. CouchInn es un sitio Web cuya finalidad principal es ayudar a que viajeros en territorio argentino encuentren hospedaje de forma temporal (sin costo o bien compartiendo gastos cotidianos) en residencias o espacios que pongan a disposición otros usuarios del sitio voluntariamente.

Para esto los usuarios al registrarse pueden publicar sus espacios bajo alguna de las categorías provistas por el sistema, informando la cantidad de huéspedes que pueden albergar y la ubicación del hospedaje. Finalmente pueden además agregar algunas fotos y una breve descripción del lugar en cuestión.

Otros usuarios podrán solicitar reservas indicando cantidad de viajeros que conforman el grupo, un intervalo de tiempo en el cual planean visitar la localidad del hospedaje. El dueño del hospedaje podrá entonces aceptar o rechazar las reservas en cuestión.

Tanto los usuarios como los hospedajes tienen un sistema de reputación y comentarios que permite a los usuarios exponer críticas de todo tipo que pudieran llegar a servir para futuros usuarios del sitio.

La cátedra planteó a los alumnos que CouchInn originalmente funcionaba con el formato de blog, y que los creadores de la idea (interpretados por los auxiliares docentes) decidieron convertirlo en un sitio Web para ampliar sus funcionalidades.

3.1. Elicitación de requerimientos

Teniendo en cuenta que la metodología Scrum propone una comunicación constante con el cliente, en esta actividad los auxiliares docentes tomaron el rol

de clientes y participaron como tales en las entrevistas a cargo de los grupos de alumnos.

También se realizaron cuestionarios dirigidos a los usuarios que utilizaban CouchInn cuando éste aún era un blog.

A partir de lo adquirido de las entrevistas se elaboró un documento de especificación de requerimientos (*Software Requirements Specification*) conforme al estándar Std 830-1998 de la IEEE [10].

Finalmente, en base a los documentos realizados se elaboró la lista de producto a utilizar en el desarrollo.

3.2. Uso de *Scrum IS-IC*

Las diferencias entre lo que propone Scrum y lo realizado con Scrum-IS-IC son:

- Las retrospectivas de *sprint* se realizaron implícitamente durante las revisiones del producto, es decir, no fueron definidas explícitamente.
- Los *daily Scrums* se realizaron dos veces por semanas en vez de diariamente, aunque el equipo de desarrolladores se mantuvo en constante contacto a través de la herramienta de comunicación para equipos Slack.
- Los elementos de la lista de producto fueron, en este caso, lo que se denominan historias de usuario. Las historias usuarios son un elemento de la metodología XP. Una historia de usuario es una descripción escrita de alguna funcionalidad que debe tener el producto y tiene asignada un valor numérico denominado *story point* que representa el esfuerzo estimado para su implementación [11].

El desarrollo del producto se llevo a cabo a lo largo de tres *sprints* de dos semanas cada uno.

Al final de cada *sprint*, se llevaron a cabo revisiones del producto utilizando demos que incluían las funcionalidades desarrolladas hasta el momento. El *product owner* era responsable de aceptar o rechazar las historias de usuario, después de verificarlas.

Antes de cada *sprint*, se hicieron las correspondientes *sprint planning meetings*, en las cuales se hizo lo que se conoce como *planning poker*, una técnica para estimar grupalmente el valor de los *story points* de las historias de usuario a través de un sistema de votación y debate en el que participa todo el equipo, incluyendo el *product owner*. [12]

Los roles ocupados, de acuerdo a lo definido por Scrum, fueron:

Product Owner: Ayudantes de cátedra que fueron asignados a cada grupo.

Scrum Master: Cada integrante del equipo de desarrollo ocupó este rol durante un *sprint*.

Development Team: Los alumnos que componen el equipo, incluyendo al que haya sido asignado *Scrum master*.

3.3. Herramientas de apoyo

Se utilizaron las siguientes herramientas para asistir al proceso de desarrollo:

- Pivotal Tracker: Una aplicación Web que permite gestionar y administrar el *product backlog* y el *sprint backlog* de forma que todos los miembros del equipo de desarrollo y el *product owner* puedan verlas y, en el caso del *product owner*, aceptar o rechazar las historias al final del *sprint*. Pivotal Tracker permite además indicar el estado y los *story points* correspondientes a cada *user story* [13].
- Slack: Una aplicación Web que permite a equipos de desarrollo (o equipos en general) comunicarse en un único medio para una mejor organización. El sistema tiene incorporada la administración de grupos contenidos en el equipo [14].
- GitHub: Repositorio Web para el control de versionado. Se utilizó Git para trabajar sobre el código en fuente y la documentación en conjunto [15]. Los docentes pudieron utilizar esta herramienta para evaluar el progreso del desarrollo durante la cursada.

En la figura 2 se muestra una recreación de la organización provista por Pivotal Tracker para las historias de usuario del proyecto CouchInn. El sistema permite llevar un seguimiento automático sobre la cantidad de *story points* abarcados por sprint (*velocity*).

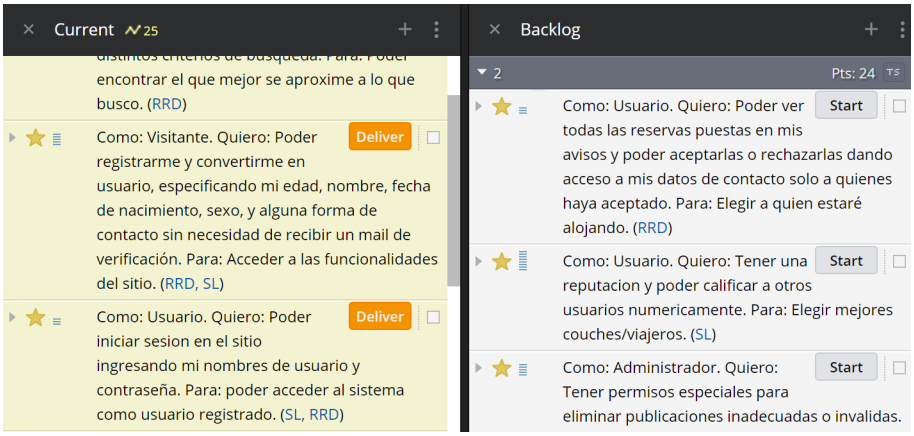

Figura 2: Captura de pantalla de Pivotal Tracker.

3.4. Tecnologías de desarrollo

Para el desarrollo del sitio Web se utilizó el lenguaje de programación Ruby con el *framework* Ruby on Rails. Éste *framework* permite empezar a desarrollar aplicaciones Web en menor tiempo que otros *frameworks* [16].

Ruby trabaja con librerías denominadas «gemas». Existe una amplia variedad de gemas específicamente para el desarrollo de aplicaciones Web con Ruby on Rails. Esto permitió al equipo reusar Software ya desarrollado para algunas funcionalidades que están frecuentemente en aplicaciones de este tipo; como el mecanismo de *log in* para los usuarios.

Para el diseño visual de la aplicación se utilizó Bootstrap. Bootstrap es un *framework* que provee plantillas CSS, extensiones JavaScript y elementos visuales para estilizar la interfaz gráfica de la aplicación Web.

Para el manejo de los datos – usuarios, publicaciones, comentarios, etc. – se utilizó PostgreSQL como sistema de gestión de base de datos.

4. Conclusión

La implementación de Scrum adaptada a la materia Ingeniería de Software permitió al equipo experimentar con una forma de trabajo innovadora que probó ser útil y eficaz en la práctica. Al ser una metodología ágil organizada en tiempos definidos y enfocada en las entregas rápidas de cambios incrementales – en contraposición a las formas de trabajo convencionales donde se realizan cambios una vez finalizado el desarrollo del producto –, los resultados de Scrum pueden verse en un plazo menor, resultando en una mejor organización del proyecto. La metodología aplicada al desarrollo CouchInn, permitió una correcta planificación y procesos de desarrollo obteniendo el producto final en el plazo pautado.

Los conceptos claves de Scrum están definidos de forma simple, lo cual permite entender rápido el cuadro de trabajo y poder empezar a aplicarlo con un grado de capacitación al respecto relativamente breve.

Existen varias herramientas informáticas en el mercado cuya utilización puede hacer que los desarrollos basados en Scrum resulten mucho más prácticos.

Referencias

1. K. A. Saleh, Software engineering. Estados Unidos: Fort Lauderdale, Florida, U.S.A.: J Ross Pub, 2009.
2. «Standish Newsroom - CHAOS 2009». [Online]. Disponible: `https://web.archive.org/web/20100410161800/http://www1.standishgroup.com/newsroom/chaos_2009.php`. Accedido el 2 de junio, 2016.
3. «EGovernment for Development - Success And Failure Rates of eGovernnment Projects in Developing/Transitional Countries». [Online]. Disponible: `http://www.egov4dev.org/success/sfrates.shtml`. Accedido: 2 de junio, 2016.
4. J. Shore, S. Warden, and Chromatic, The Art of Agile Development. Estados Unidos: O'Reilly Media, Inc, USA, 2007.
5. M. James, «An Empirical Framework for Learning (Not a Methodology)» 2013. [Online]. Disponible: `http://scrummethodology.com/`. Accedido: 22 de mayo, 2016.
6. K. Beck; M. Beedle; A. Bennekum; A. Cockburn; W. Cunningham; M. Fowler; J Grenning; J. Highsmith; A. Hunt; R. Jeffries; J. Kern; B. Marick; R. C. Marin, S. Mellor, K. Schwaber y J. Sutherland, «Agile Manifesto» 2001. [Online]. Disponible: `http://www.agilemanifesto.org/`. Accedido: 21 de mayo, 2016.

7. A. N. Cadavid, «Revisión de metodologías ágiles para el desarrollo de software» Prospectiva, vol. 11, no. 2, p. 30, Sep. 2013.
8. The Scrum Alliance, «The 2015 State of Scrum report». [Online]. Disponible: http://www.agile42.com/en/blog/2015/07/31/2015-state-scrum-report/. Accedido: 22 de mayo, 2016.
9. Ken Schwaber y Jeff Sutherland, «The Scrum Guide». [Online] Disponible: http://www.scrumguides.org/docs/scrumguide/v1/Scrum-Guide-US.pdf. Accedido: 21 de mayo, 2016.
10. IEEE Recommended Practice for Software Requirements Specifications, 1998. IEEE Std. 830-1998.
11. M. Cohn y K. Beck, User stories applied: For agile software development, 12th ed. Boston: Addison-Wesley Educational Publishers, 2004.
12. M. Cohn, Agile Estimating and Planning, 1ra ed. Prentice Hall, 2005.
13. Pivotal Software, «Quick start». 2016. [Online]. Disponible: https://www.pivotaltracker.com/help/articles/quick_start/. Accedido: 28 de mayo, 2016.
14. Slack, «Slack: Be less busy». [Online]. Disponible: https://slack.com/. Accedido: 2 de junio, 2016.
15. F. J. Lopez-Pellicer, R. Béjar, M. A. Latre, J. Nogueras-Iso, and F. J. Zarazaga-Soria, «GitHub como herramienta docente», Actas de las XXI Jornadas sobre la Enseñanza Universitaria de la Informática, págs. 66–73, julio 2015.
16. V. Viswanathan, «Rapid Web Application Development: A Ruby on Rails Tutorial», IEEE Software, vol. 25, no. 6, págs. 98–106, nov. 2008.