Desarrollo de proyectos finales mejorando la relación Universidad-Industria de Software

Marcela Daniele, Marcelo Uva, Ariel Arsaute y Franco Brusatti

Departamento de Computación
Facultad de Ciencias Exactas, Físico-Químicas y Naturales
Universidad Nacional de Río Cuarto,
Río Cuarto, Argentina.
{marcela,uva,aarsaute,fbrusatti}@dc.exa.unrc.edu.ar

Abstract

Durante la última década, la relación entre la Universidad y la industria de software se ha vuelto cada vez más estrecha. Pequeñas, medianas y grandes empresas se acercan continuamente a las universidades en busca de recursos humanos calificados. Los dinámicos avances tecnológicos y su aplicación a numerosos ámbitos del mercado, han generado la necesidad de adoptar nuevas herramientas, nuevas metodologías de desarrollo de software y nuevas modalidades de trabajo, como outsourcing y freelance, entre otros. Estos permanentes cambios, generan un trabajo constante de adaptación de los métodos de enseñanza y aprendizaje utilizados, principalmente en asignaturas de Ingeniería de Software, procurando generar graduados altamente capacitados y aprestos a insertarse de manera inmediata en el mercado laboral. En este trabajo se expone una experiencia desarrollada durante los dos últimos años en las asignaturas de Ingeniería de Software que favorece la necesaria proximidad entre la formación académica brindada a los graduados y las necesidades del mercado laboral. Esta propuesta se basa principalmente en el desarrollo de un proyecto integrador donde los estudiantes deben analizar, seleccionar y aplicar un conjunto de herramientas y tecnologías, fuertemente utilizadas en la industria de desarrollo de software, a un proyecto de software concreto.

1. Introducción

El constante crecimiento y expansión de la informática en los más diversos ámbitos, deja a la luz la imperiosa necesidad de disponer de recursos humanos calificados, con una sólida formación para adaptarse rápidamente a los cambios tecnológicos, en las modalidades de trabajo, en nuevas metodologías y lenguajes tanto de codificación como de modelado. La industria de desarrollo de software demanda profesionales capaces de afrontar estos desafíos. La relación entre la Universidad y las empresas dedicadas a desarrollar software es cada vez más estrecha. Pequeñas, medianas y grandes empresas se acercan continuamente a las universidades en busca de analistas, programadores e ingenieros de software que puedan enfrentarse y adaptarse de manera rápida y sólida a la demanda del mercado laboral. Paralelamente, dicho crecimiento ha propiciado en la industria de software, la generación y adopción de nuevas metodologías de desarrollo y conjuntamente, nuevas modalidades de trabajo, como desarrollos outsourcing y freelance. La selección de herramientas a utilizar ha evolucionado fuertemente hacia Open Source Software (OSS) y estándares abiertos. Para que una empresa resulte competitiva necesita adaptarse a los cambios tecnológicos. Las herramientas permiten automatizar y mejorar las actividades de administración, gestión, planificación, implementación, diseño, automatización de pruebas, seguimiento y control de las tareas que hacen a la producción del software. Estos cambios en las tecnologías, metodologías y herramientas requieren de una constante adaptación en las prácticas de enseñanza y de aprendizaje en las carreras de computación. Todo ello ha establecido una relación de necesidad entre la Universidad y la industria de software en ambos sentidos.

En el marco de los Proyectos de Investigación e Innovación para el Mejoramiento de la Enseñanza de Grado (PIIMEG)[1-6], desarrollados desde la Secretaría Académica y la Secretaría de Ciencia y Técnica de la

Universidad Nacional de Río Cuarto (UNRC), se han llevado adelante interesantes propuestas que permitieron detectar y analizar diversos problemas en las carreras de computación, fundamentalmente en asignaturas del área de ingeniería de software, y en base a ello proponer y ejecutar acciones concretas a fin de aportar soluciones a estas problemáticas observadas. Principalmente, estos trabajos se basaron en asignaturas de tercer año de las carreras de Analista en Computación, Profesorado y Licenciatura en Ciencias de la Computación.

El equipo de docentes que trabaja en estos proyectos apuesta fuertemente a mejorar las prácticas docentes en la enseñanza y el aprendizaje de los diferentes temas vinculados al área de la ingeniería de software. Con esta premisa, es una condición de que cada año se revise los contenidos y las metodologías de enseñanza aplicadas en las asignaturas, a fin de detectar y reforzar las fortalezas y logros, como así también observar con mucha atención las debilidades y sus causas, de manera que surjan propuestas superadoras que permitan mejorar las metodologías de enseñanza empleadas y medir los resultados obtenidos luego de su aplicación.

En este trabajo se presenta una experiencia de dictado de las asignaturas del área de ingeniería de software, llevada a cabo durante los años 2014 y 2015. Esta propuesta consiste en el desarrollo de un proyecto integrador de software, donde se profundiza la formación de los estudiantes, futuros profesionales, en sus capacidades de analizar, seleccionar y adoptar nuevas tecnologías aplicadas en la actualidad en la industria del software. Este proyecto integra los conocimientos adquiridos por el estudiante en cuanto a sus capacidades expresadas para la resolución de problemas genéricos, buenas prácticas de diseño y de programación, con un fuerte énfasis en la aplicación de un proceso de desarrollo ágil y en la selección y utilización de herramientas modernas, con un poderoso impacto en la inserción laboral del futuro graduado.

Para la organización de este trabajo en la sección 2 se exponen los fundamentos de la propuesta, en la sección 3, se explicita presenta la metodología de desarrollo aplicada en el proyecto integrador de software desarrollado. La siguiente sección describe una la experiencia presentada en este trabajo, además de la planificación, y las tecnologías y herramientas utilizadas durante 2014 y 2015. En la sección 5 se muestra una evaluación de los resultados obtenidos, y por último se exponen las conclusiones.

2. Fundamentos

El surgimiento de la Ingeniería de Software determinó, que un producto de software debe considerarse como el desarrollo de un producto complejo, con un proceso de construcción basado en un trabajo ingenieril, apoyado por metodologías, técnicas, teorías y herramientas, y una secuencia de actividades a seguir para completar el ciclo de vida de desarrollo de un software. Con una adecuada gestión, se debe obtener un producto de calidad, planificando, organizando, supervisando y controlando la evolución del proyecto durante todo su ciclo de vida. Además, es muy importante seleccionar los indicadores adecuados para la medición del proyecto y lograr estimaciones confiables en cuanto a costos, duración y recursos, permitiendo una correcta planificación temporal para cada una de las tareas. También, se deben evaluar riesgos, planificar y controlar adecuadamente los cambios y su evolución. Y considerar los factores principales como la complejidad, el tamaño y el grado de incertidumbre.

Todos estos conceptos expuestos, y también otros asociados, relacionados y transversales a los mismos, se brindan a los estudiantes de las carreras de computación. Esto permite deducir, que los estudiantes adquieren los conocimientos adecuados y suficientes, obteniendo las habilidades requeridas para lograr desarrollar un producto de software con las características de calidad exigidas para cada caso en particular.

Desde hace más de una década, en el marco de los proyectos PIIMEG [1-6], los autores de este trabajo, han realizado acciones tendientes a dar solución a diversas problemáticas identificadas en el dictado de asignaturas de las carreras de computación. A continuación se presenta un breve resumen de las temáticas abordadas:

- En el año 2004, se propuso la definición y uso de plantillas genéricas para la descripción de Casos de Uso. El objetivo fue proporcionarle al estudiante una forma clara y concreta de describir las funcionalidades de un sistema a desarrollar mediante soluciones genéricas a problemas similares o recurrentes. Y al siguiente año, continuando en la misma temática, se evolucionó a la definición de plantillas genéricas para la descripción de las etapas de análisis y diseño de un proyecto de software. [1]

- Desde el año 2006, se propuso trabajar en temáticas vinculadas a la gestión de proyectos de software y la aplicación de herramientas que favorecieran su automatización, enfocando principalmente en mejorar las prácticas vinculadas a las actividades de un gestor de proyectos de software. Se elaboró un proyecto de articulación de contenidos con todas las asignaturas de tercer año de las carreras de computación, permitiendo al estudiante establecer una relación directa entre los contenidos estudiados en cada una de las asignaturas [2,3].

- A partir del año 2010, este equipo se focalizó en la investigación y análisis de las causas que se producen y provocan que un importante número de estudiantes no logren cumplir con la planificación temporal establecida para concluir sus proyectos finales de carrera. Y a partir de estos estudios y las conclusiones obtenidas, proponer acciones correctivas a fin de mejorar la prácticas para favorecer la finalización de los trabajos y por consiguiente, el egreso de los estudiantes [4-7].

Los autores de este trabajo tienen a cargo el dictado de los cursos de Análisis y Diseño de Sistemas e Ingeniería de Software, durante el tercer año de las carreras: Licenciatura y Profesorado en Ciencias de la Computación y Analista en Computación.

La planificación y ejecución de procesos de enseñanza y de aprendizaje para cursos de Ingeniería de Software, plantean un gran desafío a los docentes universitarios involucrados. La necesidad de una actualización dinámica de los contenidos tiene siempre como finalidad la formación integral del estudiante, no solo desde lo académico, sino con una fuerte capacitación en las tecnologías utilizadas en la industria.

3. Proceso de desarrollo aplicado al proyecto integrador

En el marco de las asignaturas de Ingeniería de Software, se presenta una visión general sobre las diferentes metodologías utilizadas en el de desarrollo de software [8,9]. Algunos de los ciclos de vida del software estudiados son: el lineal o secuencial, el método del Análisis Estructurado de Yourdon[10], desarrollo basado en componentes, diseño por contratos [11] de Bertrand Meyer, entre otros. Particularmente, se pone énfasis en el Proceso Unificado[12], como método de desarrollo orientado a objetos tradicional y en SCRUM [13], como metodología de desarrollo ágil.

En este trabajo se presenta una modificación al proyecto de articulación de contenidos mencionado en la sección 2 mediante la incorporación de un conjunto de herramientas que la industria de desarrollo de software utiliza intensamente en la actualidad. Dichas tecnologías están íntimamente ligadas a la aplicación de metodologías ágiles [14], especialmente con SCRUM, metodología seleccionada por el equipo docente para el desarrollo de este proyecto integrador.

SCRUM es una metodología ágil que posibilita trabajar en ambientes muy cambiantes, permitiendo replanteamientos continuos. Por lado, reduce el tiempo de producción y de comercialización del producto, aporta un gran beneficio o valor agregado al cliente, minimiza los riesgos de desperdiciar esfuerzo/tiempo en la construcción de artefactos que no serán utilizados o que no son fundamentales para el cliente. Facilita también la comunicación entre todos los integrantes del proyecto.

La documentación producida dentro de un proyecto SCRUM es relativa al usuario, dueño, producto y equipo. SCRUM es un marco de trabajo basado sobre la premisa de que el equipo de desarrollo conocerá la mejor manera de resolver el problema que se le presenta. La reunión de planificación de cada conjunto de requerimientos a producir se describe en términos del resultado deseado, en lugar de un conjunto de criterios de ingreso, definiciones y tareas. SCRUM se basa en una auto-organización, con un equipo multifuncional y sin líder (dentro del equipo). El equipo es apoyado por dos individuos quienes ocupan los roles de SCRUM Master y ProductOwner. El SCRUM Master es una especie de entrenador para el equipo, su función es ayudar a los miembros del mismo a utilizar el marco que ofrece la metodología para conseguir un alto nivel de productividad. Mientras que el ProductOwner representa los negocios, clientes o usuarios. Éste, guía al equipo hacia la construcción del producto esperado.

Los proyectos SCRUM avanzan en orden a la definición de los sprints, que son las iteraciones que poseen una duración de entre dos y cuatro semanas.
En el inicio de cada sprint, los miembros del equipo se comprometen a producir un cierto número de características que se enumeran en el artefacto conocido como ProductBacklog del proyecto. Al final de cada sprint, cada funcionalidad (nombrada como UserStory) debe estar codificada, probada e integrada a una versión demo del sprint anterior. Luego se realiza una revisión, y por último se presenta la nueva funcionalidad frente al ProductOwner y las otras partes interesadas que proporcionarán información requerida para el siguiente sprint. Las iteraciones han de continuar hasta obtener el producto deseado. Como se puede observar de lo expuesto, SCRUM establece una forma de trabajo en donde todo el equipo se auto-regula y en donde no hay una documentación establecida a priori. Cada equipo utilizará los elementos que le sean necesarios para poder llevar adelante el conjunto de UserStories con el cual se ha comprometido. Algunos equipos podrán utilizar diagramas UML [15], otros utilizarán diagramas de flujos de datos, etc. En la figura nro. 1 se esquematiza todo el proceso.

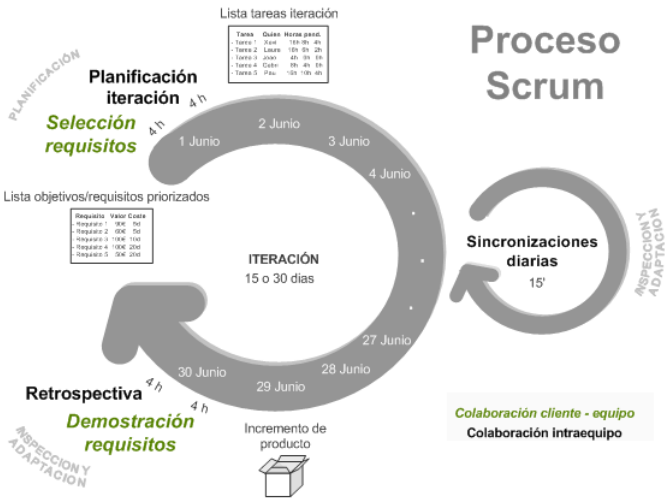

Figura1. Proceso de Desarrollo SCRUM

4. Descripción de la propuesta

La propuesta presentada en este trabajo consiste en el desarrollo de un proyecto integrador donde, además de integrar los contenidos teórico-prácticos estudiados en las asignaturas de Ingeniería de Software, se incorpore un conjunto de tecnologías y herramientas fuertemente utilizadas actualmente en la industria del desarrollo de software, permitiendo establecer un vínculo más estrecho entre el futuro egresado y los requerimientos del mercado laboral. Se describe a continuación los objetivos planteados, así como los detalles de modalidad de trabajo, organización y planificación para ejecutar el desarrollo de esta propuesta.

4.1. Objetivos

A continuación se enumeran los objetivos de esta propuesta:

- Integrar en un proyecto de desarrollo de software los contenidos trabajados en la carrera, y en particular, aquellos pertenecientes a las asignaturas de Ingeniería de Software.

- Aplicar un proceso de desarrollo ágil como SCRUM, aprovechando de esta manera los beneficios que conlleva la misma y a sabiendas de que es una de la más utilizadas en la actualidad por las empresas de desarrollo de software.

- Simular un ambiente de trabajo con características similares a las de un trabajo real. Para ello, se establecen objetivos del equipo, y es el mismo el que se auto-gestiona. Pero, si bien los objetivos son grupales, en el marco del proyecto integrador, se realiza un seguimiento y valoración particular de cada uno de los estudiantes, su integración con el resto, su compromiso, su forma de trabajo, su capacidad de superación, su conocimiento de los temas, etc.

- Desarrollar en los estudiantes habilidades para adoptar y obtener beneficios de un conjunto de herramientas utilizadas en la industria del software. El objetivo no es sólo que aprendan a usar una herramienta, sino que sean capaces de seleccionar las herramientas adecuadas en futuros proyectos.

- Favorecer la inserción laboral de los futuros graduados, brindándoles una formación profesional y técnica más cerca de lo que el mercado laboral demanda.

4.2. Organización y planificación

El proyecto integrador es planificado para ejecutarse durante el primer cuatrimestre del año. Con el objeto de brindar un acercamiento mayor a las herramientas utilizadas en el mercado actual, el equipo de docentes ha seleccionado a un docente con dedicación part-time para la coordinación general del proyecto integrador, motivada en su vinculación con la industria de desarrollo de software. Este docente es fundador de una empresa de desarrollo de software local y está habituado a formar parte de equipos de trabajo con desarrolladores a nivel internacional en proyectos bajo la modalidad outsorcing y freelance.

Uno de los objetivos del proyecto integrador es la aplicación de una metodología ágil, es por ello que el proyecto está guiado por SCRUM, y en cada una de las fases del ciclo de vida, se aplica una herramienta específica. No es objetivo de este proyecto agregar complejidad en demasía ni sobrecargar tiempos de desarrollo. Se focaliza en que cada grupo de estudiantes pueda completar el trabajo, investigando, analizando y usando las tecnologías propuestas.

Se formaron equipos de tres estudiantes, más el SCRUM Master [13], rol ocupado por un docente de la asignatura. El proyecto desarrollado en el año 2014 consistió en la construcción de un sistema web donde un usuario puede identificarse y publicar avisos de compra y venta de vehículos. En el año 2015, el proyecto

desarrollado fue la construcción de una versión del juego *cuatro en línea*. El diseño del problema se planteó de una manera simplificada evitando el crecimiento del proyecto de manera descontrolada. El equipo docente estableció un ProductBacklog inicial y en la medida de que cada grupo avanzaba se incluían nuevas funcionalidades. Todos los grupos asistían a un encuentro semanal de dos horas donde se introducían las herramientas y se establecían los objetivos según la planificación correspondiente. Durante uno de los encuentros, los ayudantes alumno, con los que cuentan las asignaturas involucradas en este trabajo, elaboraron un taller sobre una de las herramientas seleccionadas. Esto permitió realizar aportes a su formación docente, y al mismo tiempo brindó una mirada diferente al aprovechamiento de estas tecnologías. En el resto de los encuentros, los docentes realizaron el seguimiento de cada grupo de estudiantes.

A continuación se presenta una planificación detallada en semanas de la evolución del desarrollo del proyecto integrador.

- **Primera y segunda semana**.
  Presentación general del taller e introducción de Git. Durante este encuentro se realiza una presentación de los objetivos a cumplir durante el desarrollo del proyecto. Se expone el problema a solucionar y se conforman los grupos de trabajo y se realiza un taller introductorio a los sistemas de control de versiones GIT en particular.

- **Tercera semana**.
  Taller de SCRUM. Definición del SRS (Software Requeriment Specification). PivotalTracker - Creación del Product Backlog.
  Se realiza una revisión sobre metodologías ágiles, se da una introducción a la herramienta PivotalTracker. Se establecen las pautas para que cada grupo defina su propio SRS, el cual será acordado con el equipo docente.

- **Cuarta semana**.
  Taller de Java y sus entornos de desarrollo (IDES) Eclipse. Presentación de ActiveJDBC y Postgressql.

- **Quinta semana**.
  Taller de Maven. Se presentan principales características de la herramienta. Se define un proyecto modelo para que a partir de éste, los grupos puedan acceder al mismo y comprender su estructura y funcionalidades.

- **Desde la Sexta semana en adelante**.
  Diseño / Implementación. Se realiza el diseño del problema planteado y se comienza con la etapa de implementación.

- **Octava semana.**
  Taller de JUnit - definición de la Test Suite.

- **Décima semana.**
  Checkpoint (Test Suite running). Se establece este punto como un hito en donde se testea que el proyecto cumpla con la test suit definida anteriormente.

- **Un décima semana.**
  Taller Spark - Sinatra e incorporación de la capa web. Se presenta el framework web Spark y se dan las pautas para que cada grupo incorpore una vista web.

- **En la semana catorce se realiza la presentación final de cada proyecto terminado.**
  Cada grupo expone el trabajo realizado y realiza una demo de su funcionamiento. Además los estudiantes exponen individualmente su experiencia en lo que respecta a la selección y evaluación de herramientas, y se presta especial importancia al trabajo en equipo.

4.3. Tecnologías y herramientas utilizadas

La industria del software se encuentra en continua expansión, requiriendo la constante incorporación de nuevas tecnologías para generar nuevas tecnologías. Sólo añadiendo nuevas herramientas de soporte para estos procesos es que se logra producir sistemas para millones de usuarios, como por ejemplo las redes sociales.

Las principales herramientas utilizadas en este proyecto integrador fueron las siguientes:

- Github[16]: plataforma de desarrollo colaborativo de software para alojar proyectos utilizando el sistema de control de versiones Git. El código se almacena de forma pública, aunque también se puede hacer de forma privada, creando una cuenta de pago.
  Github no sólo ofrece alojamiento del código sino muchas más posibilidades asociadas a los repositorios como son, forks, issues, pullrequests, diffs, etc.

- Git[17]: Sistema de control de versiones distribuido de código abierto, originalmente escrito por Linus Trovalds, creador de Linux (sistema operativo de código abierto), diseñado para manejar proyectos muy grandes con velocidad y eficiencia, pero igual de apropiado para repositorios pequeños; es especialmente popular con la comunidad open source, sirviendo como plataforma de desarrollo para proyectos como el Kernel Linux, Ruby onRails, WINE o X.org.

  Git encuadra en la categoría de herramientas de manejo de código fuente distribuido, similar por ejemplo a Mercurial o Bazaar. Cada directorio de trabajo de Git es un repositorio completo con historial y capacidades totales de tracking de revisiones, independiente de acceso de red o un servidor central. Aun así, Git es extremadamente rápido y eficiente con el espacio

- Java 8[18]: Lenguaje de programación de propósito general, concurrente, orientado a objetos, originalmente desarrollado por Sun Microsystems, adquirida por Oracle, para aplicaciones software independiente de la plataforma.

- PostgreSQL[19]: Sistema de gestión de base de datos objeto-relacional de código abierto. Cuenta con muchos años de desarrollo activo y una arquitectura probada que se ha ganado una sólida reputación de fiabilidad e integridad de datos. Se ejecuta en los principales sistemas operativos que existen en la actualidad.

  Es totalmente compatible con ACID, tiene soporte completo para claves foráneas, uniones, vistas, disparadores y procedimientos almacenados (en varios lenguajes). Incluye la mayoría de los tipos de datos del SQL 2008. También soporta almacenamiento de objetos binarios grandes, como imágenes, sonidos o vídeo. Cuenta con interfaces nativas de programación para C / C + +, Java,. Net, Perl, Python, Ruby, Tcl, ODBC, entre otros, y la documentación que actualmente existe es realmente excepcional.

  Una base de datos de clase empresarial, PostgreSQL cuenta con características avanzadas tales como Multi-Version Control de concurrencia (MVCC), puntos en tiempo de recuperación, tablespaces, replicación asincrónica, transacciones anidadas (savepoints), respaldos online/hot, un sofisticado queryplanner/optimizer. Es altamente escalable, tanto en la enorme cantidad de datos que puede manejar y en el número de usuarios concurrentes que puede administrar

- ActiveJDBC[20]: ORM (Object Relational Mapping) para desarrollo ágil. Es una implementación en el lenguaje Java del patrón arquitectural Active Record. El patrón ActiveRecord establece que cada objeto contiene los datos de un registro de una tabla o vista, encapsulando la lógica necesaria para acceder a la base de datos. De esta manera el acceso a datos se presenta de manera uniforme a través de la aplicación. ActiveJDBC permite inferir los parámetros del esquema de la base de datos desde la propia. Por otro lado no requiere configuración de archivos, sólo requiere el seguir una serie de convenciones. No requiere que el estudiante aprenda otro lenguaje de consultas, con SQL es suficiente. No requiere de manejadores de persistencia, entre otras características.

- JUnit[21]: JUnit es un framework que permite automatizar la etapa de la prueba dentro de un proceso de desarrollo en el cual se ha seleccionado Java como lenguaje de programación. La herramienta brinda la posibilidad de realizar pruebas de regresión, de esta forma, el analista programador puede comprobar la consistencia de los cambios incorporados al código. JUnit permite realizar pruebas unitarias, las que comprueban la funcionalidad del módulo de forma independiente, y pruebas de integración que verifican la integración del módulo al resto del sistema.

- Spark[22]: framework inspirado en Sinatra para crear aplicaciones web con Java 8. Sinatra es un framework Open Source que permite desarrollar un producto de software funcional con una mínima cantidad de código fuente Java. Es una buena alternativa ante otros frameworks tales como Rails, Padrino o Merb. Entre las características que posee Sinatra es posible mencionar:

  - Rutas: Permite manejar las peticiones HTTP (GET y POST) definiéndolas de manera secuencial. Esto permite definir para cada URL, en definitiva, la lógica de negocio.

- Request y Response: proporciona un completo control sobre las cabeceras y funcionalidades de HTTP, posibilitando el envío y la recepción de objetos.

- Filtros: Los filtros se definen como una serie de características que se pueden incorporar antes o después de la ejecución de una solicitud.

- Sinatra[23]: Sinatra es un framework para desarrollo de aplicaciones web de software libre y de código abierto con una DSL (Domain Specific Language) escrito en Ruby on Rail. Sinatra permite desarrollar un producto de software funcional con una mínima cantidad de código fuente. Es una buena alternativa ante otros frameworks tales como Rails, Padrino o Merb.

- PivotalTracker[24]: herramienta para la gestión de proyectos ágiles. Pivotal Tracker es una herramienta web para gestión de proyectos ágiles que se ajusta perfectamente a las metodologías tales como Scrum. Si bien no es una herramienta de acceso gratuito para proyectos privados, si lo es para proyectos públicos. La herramienta reside en 'la nube' por lo que no es necesario instalar ningún software, todas las funcionalidades son accesibles desde el browser. Su diseño establece la gestión de requerimientos mediante pilas de historias de usuario, también brinda una serie de métricas para medir el producto que se está desarrollando. Pivotal Tracker lleva un registro histórico de las acciones llevadas a cabo para cumplir con cada uno de los requerimientos. Esto es de gran importancia al momento de requerir niveles de calidad, como por ejemplo ISO que exige trazabilidad del trabajo realizado.

- SCRUM [13]: metodología ágil para desarrollar proyectos de software. Durante el desarrollo del proyecto se realiza una revisión constante de la metodología de desarrollo expuestas en la teoría y adaptándolas al trabajo del proyecto. En particular se pone foco en el proceso de Scrum.

- Maven[25]: herramienta de software para la gestión y construcción de proyectos Java creada por Jason van Zyl, de Sonatype, en 2002. Es similar en funcionalidad a Apache Ant (y en menor medida a PEAR de PHP y CPAN de Perl), con un modelo de configuración de construcción más simple, basado en un formato XML.

- GanttProject[26]: herramienta para crear una completa planificación de un proyecto de forma visual. Todo queda bajo control en GanttProject, desde los recursos necesarios en forma de personal, los días festivos, hasta dividir el proyecto en un árbol de tareas y asignar a cada uno los recursos oportunos.

5. Evaluación de la propuesta

Para la evaluación de la propuesta, se tienen en cuenta los resultados obtenidos en cada uno de los proyectos desarrollados por los estudiantes, los aportes de los docentes involucrados, y también, se generó una encuesta a fin de reflejar el trabajo individual y grupal, la adopción de las nuevas tecnologías y la aplicación de del proceso de desarrollo ágil, SCRUM. La encuesta fue completada por 45 estudiantes que cursaron las asignaturas de Ingeniería de Software durante los años 2014 y 2015. A continuación se muestra el modelo de la encuesta realizada.

Encuesta de finalización de cursado de las asignaturas de ingeniería de software en 3er. Año de las carreras de computación.

1. ¿Cuáles fueron las materias que cursaste en este cuatrimestre ?

2. ¿Cuáles regularizaste?

3. ¿Te pareció interesante la propuesta del taller? Si/No/Parcialmente. Justifique en cada caso.

4. ¿Cuántas horas semanales promedio le dedicaste al taller?

5. ¿Se comprendió el objetivo del taller al momento de presentarlo? Si/No/Parcialmente.

6. ¿Habías realizado algún proyecto de materia utilizando alguna metodología de desarrollo? Si(Cuál)/ No.

7. ¿Todos los integrantes del grupo trabajaron de igual manera? Si/No/ En ocasiones.

8. ¿Qué porcentaje del total aportaste al proyecto?

9. En general, como calificas la experiencia de trabajo grupal. Excelente/Buena/Regular/Mala.

10. ¿Conocías de las herramientas antes de comenzar con el taller?: Puntuar del 10 al 1,

a) Lenguaje de programación Java
b) IDEs tales como Eclipse
c) Netbeans
d) Maven
e) Spark web framework
f) Git
g) ActiveJDBC
h) Otras herramientas usadas:

11. Describe muy brevemente para qué utilizaste cada una de las herramientas anteriores.

a) Lenguaje de programación Java
b) IDEs tales como Eclipse
c) Netbeans
d) Maven
e) Spark web framework
f) Git
g) ActiveJDBC
h) Otras herramientas usadas:

12. Del 10 al 1 califica la utilidad de cada una de las herramientas utilizadas respecto a tu experiencia en este cuatrimestre.

13 ¿Cuáles fueron las herramientas que te resultaron más fáciles de adoptar y cuáles fueron las más difíciles?

14. Con respecto a las otras asignaturas de 3ero. ¿Pudiste relacionarlas con el taller ?

a - Base de datos: Si/No/Parcialmente.

b - Diseño de Algoritmos: Si/No/Parcialmente.

15. ¿Tuvieron problemas con el tiempo y las entregas parciales?, ¿Cuáles fueron los motivos?

16. ¿Te sentís capaz de desarrollar un proyecto similar pero a mayor escala?.

Luego de realizar y analizar las encuestas, y evaluando el desempeño de los grupos es posible destacar que todos los grupos se mostraron muy entusiasmados e interesados con la propuesta. Algunos grupos lograron superar ampliamente los objetivos planteados (85\%), mientras que al resto le resultó más costoso lograr adaptarse a las nuevas herramientas.

La modalidad del taller, exige tiempo de investigación fuera del horario de clases. Un par de grupos tuvieron un desempeño sobresaliente, obligando al cuerpo docente a proponer nuevas funcionalidades y nuevas herramientas.

La mayoría de los estudiantes desconocían herramientas como Git, JUnit, PivotalTracker, ActiveJDBC. Si bien muchas de ellas son citadas en las asignaturas de Ingeniería de Software, fue dentro de este proyecto el primer contacto con las mismas. Un gran porcentaje de los grupos expresa que se siente capaz de escalar esta experiencia a un proyecto de dimensiones mayores, aplicando las mismas tecnologías.

El trabajo en equipo fue altamente positivo. Se pudo observar el desempeño de cada uno de los integrantes de cada equipo, ayudado por una de las herramientas utilizadas (Git). Si bien, los docentes realizaron un seguimiento particular de cada estudiante, se le otorgó al grupo autonomía para trabajar, de acuerdo a lo establecido por el proceso de desarrollo elegido para este proyecto, SCRUM. En la mayoría de los casos, ellos mismos se dividieron las tareas, por ejemplo cada uno se dedicaba a investigar una herramienta particular para luego mostrarla al resto del grupo. Git fue una de las herramientas más resistidas en un comienzo. Esto, en parte se debe a que esta herramienta establece una nueva modalidad de trabajo, en particular para trabajos distribuidos. En un comienzo provocaba problemas en muchos grupos, hasta que pudieron aprenderla y adoptarla, y finalmente sacar provecho de los beneficios del versionado establecido por Git.

Con respecto a la metodología de desarrollo utilizada en esta experiencia, todos los grupos pudieron seguirla sin inconvenientes, comprendieron los roles dentro del proceso, sus obligaciones como parte de un equipo de desarrollo y lograron producir el software acordado.

6. Conclusiones

La propuesta realizada en este trabajo, se fundamenta en el desarrollo de un proyecto integrador de ingeniería de software, sobre un dominio particular, con estudiantes de tercer año de las carreras de computación. Fundamentalmente, la propuesta está centrada en integrar todos los conocimientos adquiridos por el estudiante, y con el fuerte propósito de profundizar la formación del mismo en cuanto a su capacidad para analizar y seleccionar tecnologías, y en particular herramientas altamente utilizadas en la actualidad en la industria de software.

Esta experiencia favorece a una capacitación más completa e integral del estudiante, en virtud de que éste, pueda integrar fuertemente los conceptos teóricos con los prácticos, visualizar el comportamiento y reacción del mercado ante las nuevas tecnologías y reflexionar acerca

de su impacto. Los objetivos planteados en la propuesta se consiguen en su totalidad, y en muchos casos, superan las expectativas que definió el equipo docente. Ello en el sentido, de que algunos estudiantes presentan tanto interés y dedicación, que alcanzan a superar los objetivos y proponen nuevos desafíos que sirven tanto para mejorar sus proyectos, como para una superación de esta propuesta. Con esta metodología, se consigue que el estudiante se convierta en un actor más activo de su proceso de enseñanza y aprendizaje a partir de las tareas de investigación que debe realizar, para determinar ventajas y desventajas de las herramientas y técnicas seleccionadas para su proyecto. Luego, el estudiante es capaz de analizar y seleccionar las herramientas que más se adecúen al proceso de automatización bajo su desarrollo.

Por otro lado, es muy importante destacar que la simulación del desarrollo de un proyecto de software real, con un ambiente de trabajo en equipo y colaborativo, brinda la posibilidad de que los estudiantes se auto-gestionen, distribuyan sus tareas y tiempos, analicen riesgos, y que puedan ser capaces de aplicar acciones correctivas para controlar las desviaciones en la planificación original. Siendo los docentes, los encargados de realizar un seguimiento muy cercano de cada grupo, controlando permanentemente que se adecúen a la planificación prevista y evitando que el proyecto adquiera demasiada complejidad o que crezca en tamaño de forma descontrolada, ya que no es objetivo de este trabajo aumentar su carga horaria en perjuicio de su rendimiento académico general en la carrera.

Con esta propuesta se favorece la inserción laboral de los futuros graduados, brindándoles una formación profesional y técnica más cerca de lo que el mercado laboral demanda.

En este año, se está trabajando en recabar información de los estudiantes, muchos de ellos actualmente egresados, que en 2014 y 2015 desarrollaron el proyecto integrador de las asignaturas de ingeniería de software bajo esta propuesta, y que actualmente trabajan en la industria del software, con el fin de obtener información de experiencias y realizar mediciones respecto a cuanto los favoreció la aplicación de esta metodología. Además, se han generado contactos con las empresas empleadoras donde trabajan los recientes graduados, a fin de obtener su opinión, expectativas y sugerencias, siempre apuntando a mejorar las propuestas y prácticas de enseñanza y la formación de los actuales estudiantes.

7. Referencias

[1] Proyectos de Innovación e Investigación para el Mejoramiento de la Enseñanza de Grado (PIIMEG), Secretarías de Ciencia y Técnica, y Académica, Universidad Nacional de Río Cuarto. De [2] a [6].

[2] M. Daniele. D. Romero. Definición y uso de Plantillas Genéricas para la descripción de Casos de Uso. RR Nº 302/04. 2004.

[3] M. Daniele, D. Romero. Evolución de Plantillas Genéricas para la descripción de Casos de Uso a Plantillas Genéricas para Análisis y Diseño. RR Nº 109/05. 2005.

[4] M. Daniele. D. Romero. La enseñanza de gestión de proyectos de software y la aplicación de herramientas que favorezcan su automatización. RR Nº 499/06. (01/08/2006, 31/07/2008).

[5] M. Daniele. F. Zorzan. Estimación y Planificación de Proyectos de Software versus duración de proyectos finales en la carrera Analista en Computación. RR Nº 171/11. (2011, 2012).

[6] M. Daniele. F. Zorzan. Causas que producen que los estudiantes de Computación retrasen la culminación de su Trabajo Final. RR Nº 923/12. (2013,2014).

[7] Fabio Zorzan, Mariana Frutos, Ariel Arsaute, Marcela Daniele, Paola Martellotto, Marcelo Uva, Carlos Luna “Delayed Completion of Final Project of the Career Computer Analyst: Seeking its Causes”. XX Congreso Iberoamericano de Educación Superior (CIESC 2012), en el Marco de la XXXVIII Conferencia Latinoamericana en Informática – CLEI 2012 - Octubre 1 al 5 de 2012 - Medellín, Colombia. ISBN 978-1-4673-0792-5.

[8] Roger S Pressman. Libro: Software Engineering: A Practitioner's Approach. 8th Edition.McGraw-Hill Education. 2014.

[9] An Integrated Approach to Software Engineering. Pankaj Jalote. Springer 2006.

[10] E. Yourdon. Libro: Análisis estructurado moderno. Prentice-Hall Hispanoamericana, 1993.

[11] Meyer Bertrand. Object Oriented Software Construction. Prentice Hall. 1997.

[12] Ivar Jacobson, Grady Booch, James Rumbaugh. Libro: El proceso unificado de desarrollo de software. The Addison-Wesley, 2000

[13] SCRUM in Action: Agile Software Project Management and Development. Andrew Pham, Phuong Van Pham. Course Technology Ptr. 2011.

[14] Highsmith Jim. Agile Software Development Ecosystems. Addison-Wesley 2002. ISBN:0201760136

[15] Grady Booch. Libro: The Unified Modeling Language User Guide. The Addison-Wesley, 2005.

[16] GitHub. (2016). Build software better, together. [online] Disponible en: https://github.com [Accessed 5 Aug. 2016].

[17] Git-scm.com. (2016). Git. [online] Disponible en: https://git-scm.com/ [Accessed 5 Aug. 2016].

[18] Oracle.com. (2016). Oracle | Integrated Cloud Applications and Platform Services. [online] Disponible en: http://www.oracle.com [Accessed 5 Aug. 2016].

[19] Postgresql.org. (2016). PostgreSQL: The world's most advanced open source database. [online] Disponible en: https://www.postgresql.org [Accessed 5 Aug. 2016].

[20] Javalite.io. (2016). JavaLite, ActiveJDBC -. [online] Disponible en: http://javalite.io/activejdbc [Accessed 5 Aug. 2016].

[21] Junit.org. (2016). JUnit. [online] Disponible en: http://junit.org [Accessed 5 Aug. 2016].

[22]Sparkjava.com. (2016). Spark Framework - A tiny Java web framework. [online] Disponible en: http://sparkjava.com [Accessed 5 Aug. 2016].

[23] Sinatrarb.com. (2016). Sinatra. [online] Disponible en: http://www.sinatrarb.com [Accessed 5 Aug. 2016].

[24] Pivotal Tracker. (2016). Pivotal Tracker | Agile Project Management. [online] Disponible en: https://www.pivotaltracker.com [Accessed 5 Aug. 2016].

[25] Porter, B., Zyl, J. and Lamy, O. (2016). Maven – Welcome to Apache Maven. [online] Maven.apache.org. Disponible en: https://maven.apache.org [Accessed 5 Aug. 2016].

[26] Ganttproject.biz. (2016). GanttProject: free desktop project management app. [online] Disponible en: https://www.ganttproject.biz/ [Accessed 5 Aug. 2016].