Aplicación de herramientas CASE a la enseñanza de Ingeniería de Software: Gestión de la Configuración de Software y Testing Funcional

Marcela Daniele, Marcelo Uva, Paola Martelloto y Gonzalo Picco

Universidad Nacional de Río Cuarto
Facultad de Ciencias Exactas, Físico-Químicas y Naturales
Departamento de Computación, Ruta 36 Km 601 CP X5804BYA, Río Cuarto, Córdoba, Argentina.

{marcela, uva, pmartelloto}@dc.exa.unrc.edu.ar
gonzalopicco@gmail.com

Resumen

La Ingeniería de Software (IS) es la disciplina o área de la informática que ofrece métodos y técnicas para el desarrollo y mantenimiento de software de calidad. A pesar de todos los esfuerzos realizados, en las sucesivas etapas del desarrollo y en las actividades de gestión de proyectos de software, ocurren problemas como: carencia de fiabilidad, necesidad de mantenimiento permanente, etc. Las dinámicas de los procesos de desarrollo de software exigen una actualización constante de técnicas y tecnologías. Esta tarea es favorecida por la utilización de herramientas CASE que asisten a todos los involucrados en la construcción del software.

En el siguiente trabajo se presenta una propuesta de enseñanza de algunos tópicos fundamentales de la IS como son: Gestión de la configuración del Software (GCS) y Testing. El trabajo muestra los resultados obtenidos con la ejecución de dos Proyectos-Taller, en los temas mencionados. Se propone un aporte a la enseñanza de la IS, capacitando a los alumnos en el uso de técnicas y herramientas. Se busca generar en los estudiantes los mecanismos adecuados para profundizar la comprensión de los temas de GCS y Testing.

Palabras claves: Testing Funcional, Gestión de la Configuración de Software, Herramientas CASE, Enseñanza de Ingeniería de Software.

1. Introducción

La IS es la disciplina o área de la informática que ofrece métodos y técnicas para el desarrollo y mantenimiento de software de calidad. Dentro de la IS, las metodologías son las encargadas de caracterizar las fases principales del desarrollo, administrando el progreso y mantenimiento, en el sentido de que permitan estimar recursos, definir puntos de control y monitorear el avance del proyecto, entre otras actividades.

El mercado de desarrollo de software actual exige profesionales capaces de adaptarse rápidamente a los cambios. Grandes y pequeños sistemas de información necesitan un mantenimiento sostenido para satisfacer la evolución de las necesidades del cliente y del propio mercado. Las dinámicas de los procesos de desarrollo de software exigen una actualización de técnicas y tecnologías diaria. Esto trae como consecuencia, la utilización de herramientas CASE *Computer Aided Software Engineering* que asistan a todos aquellos involucrados en la construcción de proyectos de software, posibilitando una mejor gestión y administración de los recursos.

Tanto usuarios como los ingenieros de software reconocen la necesidad de enfocar de manera más disciplinada el desarrollo de software [1,2]. A pesar de todos los esfuerzos realizados, en las sucesivas etapas del desarrollo y en las actividades de gestión de proyectos de software, ocurren problemas como: carencia de fiabilidad, necesidad de mantenimiento permanente, retrasos en las

entregas y costos superiores a los presupuestados [2,3]. En 1996, la firma internacional Standish Group Internacional, Inc realizó lo que se conoce como "The Chaos Report" [4] reportando que sólo un 16 % de los proyectos de software son exitosos, es decir, finalizan dentro de plazos, costos y requerimientos acordados, otro 53 % sobrepasa costos y plazos y cumple parcialmente los requerimientos y el resto no llega a término. Los sucesivos reportes (1996, 1998, 2000, 2003 y 2004) muestran que estas cifras siguen tomando importancia en los proyectos de software. En CEIS (Centro Experimental de Ingeniería de Software) definen lo que llaman "peores prácticas y prácticas ausentes del desarrollo de software", entre ellas:no existe medición del proceso ni registro de datos históricos, estimaciones imprecisas de plazos y costos, mal uso de herramientas automatizadas, crecimiento excesivo de los requerimientos para un producto de software, escaso o deficiente monitoreo en el progreso del proceso de desarrollo, ausencia de la gestión de riesgos y del testing, entre otras.

Se considera que uno de los compromisos fundamentales de todas aquellas instituciones formadoras de recursos humanos en el área de Informática, es el de generar nuevas metodologías, estrategias y procesos de enseñanza-aprendizaje en pos de lograr revertir las problemáticas anteriores.
En el siguiente trabajo se presenta una experiencia llevada a cabo en la cátedra de Ingeniería de Software perteneciente a las carreras de Analista, Profesorado y Licenciatura en Ciencias de la Computación de la Universidad Nacional de Río Cuarto, en el año 2009. El trabajo consistió en el dictado de dos Proyectos-Taller de aplicación, paralelos al dictado de la asignatura. Los objetivos principales de estos talleres fueron: profundizar aquellos temas de IS cuya aplicación práctica a través de herramientas CASE sean determinantes para una mejor comprensión por parte de los alumnos.
Los talleres fueron realizados en forma grupal, con la intención de favorecer el trabajo en equipo. En ellos es importante la obtención de una buena comunicación y coordinación entre todos los miembros. Los talleres consistieron en la aplicación de herramientas automáticas CASE para la realización de las tareas de Gestión de la Configuración del Software (GCS) y Testing. Las herramientas utilizadas fueron Subversion [18] y JUnit [17]. Ambos proyectos-taller fueron diseñados cuidadosamente por el cuerpo docente para simular problemáticas presentes en proyectos de desarrollo reales.
Este trabajo propone un aporte a la enseñanza de la ingeniería de software capacitando a los alumnos en el uso de técnicas y herramientas en proyectos de desarrollo de software. Se persigue generar en los estudiantes los canales adecuados para una mejor compresión de los tópicos de GCS y Testing.

En la sección 2 se fundamenta esta propuesta y se plantean los objetivos. La sección 3 se expone la metodología de trabajo. La sección 4 presenta las etapas de implementación de los proyectos-taller. La sección 5 esboza las formas de evaluación y algunos resultados de la propuesta, y por último, las conclusiones.

2. Fundamentación y propuesta

La planificación y ejecución de procesos de enseñanza-aprendizaje para cursos de IS, plantean un gran desafío a los docentes universitarios involucrados. La necesidad de una actualización dinámica de los contenidos no deben provocar el descuido de conceptos básicos vinculados a los principios fundamentales del desarrollo de sistemas de software.
El cuerpo docente, autor de este artículo, tiene a cargo el dictado de los cursos 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, de la Universidad Nacional de Río Cuarto.
Los conceptos básicos de IS, son introducidos en el curso de Análisis y Diseño de Sistemas.

En este se presentan diferentes metodologías de desarrollo [6, 7]. Para el curso de Ingeniería de Software, el principal propósito es que el alumno tome conocimiento de los conceptos más avanzados de IS; desde la planificación y gestión del proyecto hasta técnicas de testing o prueba.

Durante los años 2006 [10,11] y 2007 [12,13] , se abordó otro importante desafío a cubrir en la enseñanza de ingeniería de software, la utilización de herramientas que permitan automatizar la gestión de proyectos de software[14]. En este marco, se logró fomentar el uso de una metodología de desarrollo de software en proyectos reales. Promoviendo de esta manera la división de roles operativos y gerenciales entre los integrantes de un grupo de desarrollo de software, capacitándolos para la evaluación y selección de diferentes herramientas aplicables a las actividades de gestión de software.
En el año 2008 [15], se presentó un framework genérico que implementa de manera distribuida los problemas de inserción, eliminación, modificación y búsqueda de elementos, incluyendo tecnologías distribuidas a través del uso de servicios web.
Logrando impulsar un pensamiento crítico en el estudiante, en que la utilización de patrones de diseño [8, 9] ayuda a la comunicación con los demás ingenieros, a la estandarización, a la flexibilidad y a la robustez en la arquitectura del software. El alumno aprende a usar patrones de diseño de una forma disciplinada para la resolución de problemas y es motivado para construir nuevas propuestas genéricas que den solución de otros tipos de problemas.
Durante el año 2009, en la asignatura de Análisis y Diseño se planteó el desafío de utilizar técnicas de Ingeniería Inversa [5] con el fin de generar en el estudiante la habilidad de diseñar proyectos de software implantados, reusar componentes de software y extender las funcionalidades de un producto de software ya implementado.
Durante el segundo cuatrimestre del 2009 se planteó un nuevo desafío, el de utilizar herramientas CASE para la GCS y Testing funcional de manera automática.
Los objetivos buscados con esta nueva propuesta fueron:

- Profundizar los conocimientos en GCS y Testing.
- Capacitar al alumno en el uso de herramientas CASE vinculadas a GCS y Testing.
- Identificar los posibles cambios dentro de un proyecto.
- Controlar y garantizar que los cambios sean implementados de manera correcta.
- Fomentar e incentivar la reutilización de componentes de software.
- Promover la división de roles en un equipo de desarrollo de software.
- Mostrar la necesidad de poseer documentación detallada de cada uno de los artefactos construidos en el ciclo de vida del software, cuando se pretende extender y/o modificar un producto de software.
- Ejecutar testing funcional a módulos de software pertenecientes a sistemas reales.

3. Contexto y Metodología de trabajo

La metodología de enseñanza descripta en este trabajo se aplica a un curso de Ingeniería de Software de tercer año de las carreras de computación de la Universidad Nacional de Río Cuarto. El principal objetivo de este curso es que el alumno tome conocimiento de conceptos básicos de IS, que van desde la planificación y gestión de proyectos hasta técnicas de prueba,. incluyendo otros métodos de desarrollo de software complementarios a los vistos por el alumno. Al mismo tiempo se cubren conceptos transversales a las etapas de desarrollo como el gerenciamiento de la configuración de software.
Esta metodología incorpora la realización de proyectos-taller, abordando específicamente el desafío de la utilización de herramientas que asisten en las actividades de gestión de proyectos de software [1,2] y ayuden al mismo

tiempo la compresión profunda de los temas abordados. El principal propósito es conseguir que los alumnos vivencien situaciones muy cercanas a la realidad y, de esta manera, disminuir la brecha entre la teoría universitaria y la realidad profesional.
El dictado de la asignatura IS se divide en clases teóricas, clases prácticas y proyectos-taller.
En cada teoría se dictan los contenidos generales de cada unidad temática y se desarrollan ejemplos concretos de casos particulares, que ayudan a los alumnos a reflexionar sobre la aplicación del conocimiento teórico en un caso práctico específico.
Para las clases prácticas, se divide al total de alumnos en comisiones de unos 20 cada una alumnos aproximadamente.
Se cuenta con una abundante batería de ejercicios prácticos para cada unidad temática. Los problemas planteados son ordenados de manera creciente en complejidad. En cada clase práctica, el docente y los alumnos realizan la tarea de confrontar las diferentes soluciones planteadas para cada problema y se debaten las ventajas y desventajas de cada solución.

El temario de la asignatura de Ingeniería de Sofware abarca un gran volumen de temas como los son: Gestión de Proyectos, Proceso de Software y Métricas de Proyecto, Planificación, Métodos de Testing o Prueba del Software, Métricas Orientados a Objetos, Calidad del Software, Gestión de la Configuración del Software, Arquitectura de Software y Patrones, entre otros.

4. Proyectos – Taller

Los dos proyectos-taller realizados durante el año 2009 en la asignatura IS fueron confeccionados con el objetivo de profundizar el conocimiento de dos tópicos muy importantes como lo son "Gestión de la Configuración de Software" y "Automatización del Testing".

La gestión de la configuración del software es uno de los procesos clave para toda organización dedicada a la Ingeniería del Software, ya que posibilita una mejor organización del desarrollo y mantenimiento, facilitando el resto de procesos de producción.
Durante el proceso de construcción de un software, los cambios son inevitables. Los cambios provocan confusión e incertidumbre, sobre todo cuando no se han analizado o pronosticado correctamente. Es importante considerar ciertas modificaciones que pueden ocurrirle al software dentro de todo el proceso de ingeniería. El arte de coordinar el desarrollo de software para minimizar la confusión, se denomina gestión de la configuración. La gestión es el arte de identificar, organizar y controlar las modificaciones que sufre el software, la finalidad es maximizar la productividad minimizando la fallas.
La gestión de configuración del software realiza un conjunto de actividades desarrolladas para gestionar y registrar los cambios a lo largo del ciclo de vida del software. La GCS es una actividad de garantía de calidad del software que se aplica en todas las fases del proceso de IS. La herramienta seleccionada para la ejecución de este proyecto-taller ha sido Subversion svn. Subversion[18] es un software de sistema de control de versiones. Es software libre bajo una licencia de tipo Apache/BSD.

En Subversion, los archivos versionados no tienen cada uno un número de revisión independiente. Subversion puede acceder al repositorio a través de redes, lo que le permite ser utilizado por personas que se encuentran en distintos lugares físicos. Provee la capacidad de que varias personas puedan modificar y administrar el mismo conjunto de datos desde sus respectivas ubicaciones, esto estimula la colaboración. Todo esto permite un progreso más enérgico del proyecto.

Por otro lado, el segundo proyecto-taller propuesto "Automatización del Testing", se basa en el hecho de que probar código nunca tuvo tanta importancia en el ciclo de desarrollo

de una aplicación hasta hace algunos años, donde se ha desatado una revolución en los procesos de desarrollo, apareciendo nuevas y ágiles forma de construcción, donde ejecutar pruebas. La herramienta seleccionada para la realización del segundo proyecto-taller ha sido JUnit.

JUnit [17] es una herramienta simple, Open Source que se ha convertido en el estándar para probar clases java de forma unitaria.
Es fácil de usar y configurar por lo que prácticamente no existe curva de aprendizaje, salvo el cambio de mentalidad que debe producir en las mentes de los desarrolladores al usar técnicas que faciliten el uso de unidades de prueba.
JUnit esta diseñado para reportar fallos o éxitos sobre código sin necesidad de interpretar el reporte. JUnit ejecuta casos de prueba (Test Cases) contra un conjunto de objetos y provee formas de inicializar y configurar cada Caso de Test.

Los objetivos específicos de estos proyectos-talleres fueron los siguientes:

- Lograr que los alumnos pudieran interrelacionarse con sus pares.
- Realizar un uso intenso de herramientas para la automatización de las actividades de gestión de proyectos de software.
- Valorar las ventajas que obtiene la gerencia del proyecto con la utilización de herramientas.
- Favorecer la investigación, la interacción y la coordinación para el trabajo en equipo de los futuros profesionales.

4.1 Implementación los Proyectos -Taller

A continuación se plantean las actividades y pautas de trabajo presentadas a los alumnos para la realización de los proyectos-taller.

Planteo del proyecto-taller de GCS
Actividades

a. Investigar y analizar el funcionamiento de la herramienta Subversión. En la página de la asignatura hay una serie de documentos que pueden ser de utilidad. También es posible encontrar información en
http://subversion.tigris.org.

b. Los grupos de trabajo ya formados deberán implementar una calculadora de polinomios que permita la realización de operaciones matemáticas básicas entre polinomios.
Para cada grupo, el cuerpo docente ha creado un repositorio utilizando la herramienta SVN en donde se encontrarán las clases base de la aplicación.

El la figura nro. 1 se muestra un diagrama de clases de diseño en donde se presenta el diseño del la calculadora de polinomios.

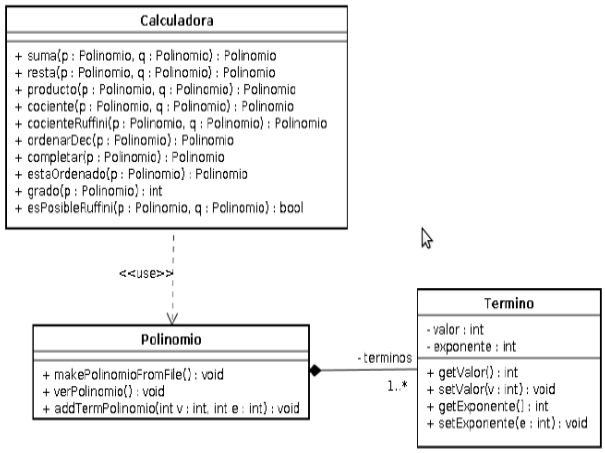

Figura nro. 1 – Diseño de la Calculadora de Polinomios.

c. Internamente, cada grupo de desarrollo (6 alumnos) deberá sub-dividirse en dos equipos de trabajo: Equipo I (rojo) y Equipo II(azul) . A cada uno de ellos se les asignará la tarea de implementación de un determinado conjunto de funcionalidades del sistema. Las actividades de cada equipo están establecidas por el diagrama de Gantt de la figura nro. 2.

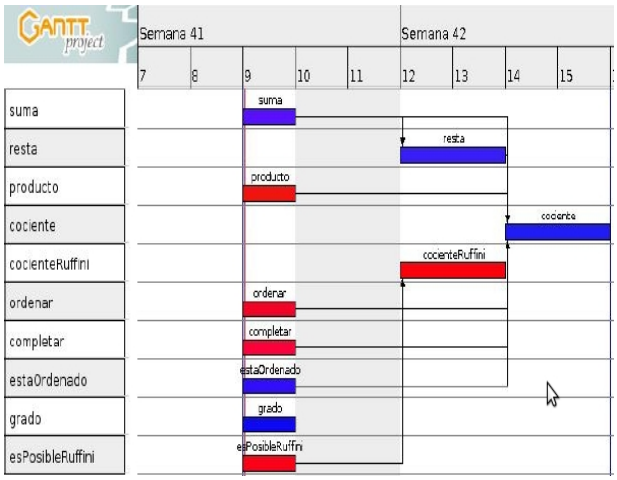

Figura nro. 2 – Diagrama de Gantt

La dinámica de trabajo será la siguiente:

Cada equipo deberá realizar una copia local del repositorio SVN en su máquina de trabajo. Luego de realizar la implementación de cada método que le ha sido asignado deberá actualizar la copia del sistema en el repositorio teniendo en cuenta los posibles conflictos y soluciones entre las distintas versiones de la aplicación.
Para la realización de estas actividades deberán utilizarse los comandos provistos por la herramienta SVN.

Cada equipo deberá respetar la estructura de los métodos presentados en el diagrama de diseño.

Finalmente, todo el equipo de desarrollo deberá realizar un presentación del trabajo realizado con la calculadora de polinomios funcionando.

Planteo del proyecto-taller de Testing Funcional

Actividades a realizar:

a. Cada grupo deberá investigar acerca del funcionamiento de las herramientas JUnit y Dbunit. Además deberá presentar un resumen(impreso y digital) en donde se explique el funcionamiento de las mismas. Incluir al menos un ejemplo de uso de cada herramienta. Explicar la relación de JUnit con los patrones de diseño Command y Composite.

b. Cada grupo deberá descargar e instalar las herramientas JUnit y DBunit. Eclipse ya cuenta con un plugins para JUnit. DBUnit está disponible en muchos repositorios en la Web
http://www.java2s.com/Code/JarDownload/dbunit.jar.zip.
JUnit puede ser también instalada como una aplicación independiente http://www.junit.org.

c. Seleccione al menos 4 métodos del código fuente asignado[16] para ser testeados utilizando JUnit. Escriba los casos de prueba y ejecútelos con la herramienta.

d. Seleccione al menos 3 métodos del código fuente del proyecto asignado[16] para ser testeados utilizando DBunit. Escriba los casos de prueba y ejecútelos con la herramienta. Dentro de los casos de prueba posibles deberá incluirse el test de duplicación de clave.

e .Los casos de prueba seleccionados deberán seguir algunos de los criterios vistos en la teoría, justificando la elección del mismo en cada caso.

f. Finalmente el grupo deberá realizar una presentación oral en donde explique el trabajo realizado, junto con un informe impreso.

5. Evaluación de la Propuesta

La evaluación de la propuesta es realizada por el grupo de docentes que dicta el curso, que también recibe aportes de alumnos, docentes de otras asignaturas, asesores pedagógicos y directivos.
Los instrumentos de recolección de información empleados son fundamentalmente: Entrevistas individuales a

alumnos, auxiliares y docentes participantes. Reuniones semanales de discusión entre los docentes de la asignatura, y mensuales con docentes de asignaturas muy relacionadas. Análisis del puntaje obtenido por los alumnos en los exámenes parciales, particularmente en los temas abordados en los proyectos-taller. Evaluación de los informes de cada una de las entregas presentadas por cada grupo y análisis comparativo entre ellas. Elaboración de tablas y/o gráficos con los resultados obtenidos.

De todos los instrumentos mencionados hemos extraído los siguientes resultados:

Se dividió el total de alumnos que cursaron la asignatura en grupos de 6 integrantes. Luego cada grupo fue subdividido en grupos de 3 alumnos para que cada uno implementara un conjunto de funcionalidades diferentes teniendo en cuenta las dependencias entre las mismas.

Se observó un gran entusiasmo por parte de todo los grupos. Para el primer proyecto-taller, el de GCS, los alumnos trabajaron simultáneamente sobre un conjunto de clases de código base proporcionadas por el cuerpo docente, donde se habían definido los perfiles de los métodos a implementar. Intencionalmente, las funcionalidades implementadas por cada subgrupo de alumnos estaban muy relacionadas con las que debían implementar otros grupos, por lo que se logró uno de los objetivos planteados en cuanto a la simulación de situaciones reales. Muchos de los alumnos, que ya habían tenido algunas experiencias profesionales en el desarrollo de pequeños sistemas, podían vislumbrar algunos problemas muy comunes cuando se trabaja en ambientes distribuidos. Se logró obtener una muy buena comunicación entre los grupos.

Por otro lado los alumnos vieron la necesidad de generar un código limpio, correctamente comentado, para evitar que sus compañeros cometieran algún error en la interpretación de lo que ya estaba implementado.

El aprendizaje de la herramienta fue muy rápido. El docente simplemente presentó una descripción muy breve de la misma y dio las pautas para que cada grupo realizara un pequeño trabajo de investigación. En este punto de ejecución del proyecto-taller, se buscó incentivar en el alumno la necesidad de conocer nuevas herramientas CASE para ordenar y simplificar el desarrollo de tareas en la construcción de software. Este objetivo fue alcanzado ampliamente.

Vinculando este trabajo con los exámenes finales presentados hasta el momento por estos alumnos, se ha observado que los mismos pueden establecer un conexión entre la teoría y la práctica ya que utilizan continuamente ejemplos de lo realizado en los talleres.

Para el caso del segundo proyecto-taller propuesto, "Testing Funcional", se impartió un código fuente real [16], sobre el cual los alumnos debieron diseñar y ejecutar los casos de test basándose en algunos de los métodos de testing vistos en las teorías. Este taller fue organizado en grupos de 3 alumnos. Todos lograron cumplir con las espectativas planteadas en un comienzo. De todos los grupos uno sólo tuvo algunos inconvenientes en cumplir con los plazos de entrega, esto fue debido a la coincidencia con exámenes parciales de la carrera. A este grupo se le extendió brevemente el plazo establecido para la entrega final pudiendo completarlo sin problemas. Por las características de la herramienta JUnit, los métodos de testing seleccionados, fueron testing de caja negra, "valor límite", "clases de equivalencias" y "tablas de decisión".

Durante la ejecución de este taller se observó en una primera instancia una cierta incertidumbre por parte de los alumnos, ya que no podían vislumbrar las ventajas de utilizar esta herramienta. JUnit exige la instanciación de un grupo de clases Java para creación de clases de test. Esta incertidumbre se debió a que las pruebas eran realizadas en ejemplos muy simples, los alumnos debían crear una serie de clases para que, por ejemplo, se verificara que una suma de dos números diera el resultado correcto. La situación fue cambiando al descubrir que la herramienta les proporcionaba un medio para controlar las pruebas de regresión, necesarias cuando una parte del código ha sido modificado y se desea

verificar que el nuevo código cumple con los requerimientos anteriores, sin alterar la funcionalidad del sistema después de una modificación. Se detectó un gran entusiasmo en los alumnos cuando lograban identificar algún error en el código asignado, incentivándolos a buscar nuevas falencias.
Durante los exámenes finales se han observado una mejora en la compresión de las técnicas prueba de caja negra.

6. Conclusiones

Los esfuerzos realizados para mejorar la enseñanza y el aprendizaje de los temas de Ingeniería de Software, son bien recibidos si se realizan con una planificación clara y una inserción adecuada en la currícula de estos cursos.
La propuesta metodológica presentada en este trabajo, refuerzan en el accionar de los alumnos la aplicación de herramientas que permiten asistir en la gestión de proyectos de software, mejorando considerablemente la toma de decisiones gerenciales. Además, permite la adquisición de conductas profesionales efectivas y ordenadas, hacia el uso y la búsqueda de elementos que colaboren con la obtención de software de alta calidad.
Se logró obtener una muy buena comunicación y coordinacion entre los integrantes de los grupos. Por otro se puso en evidencia clara la necesidad de realizar y mantener las tareas de GCS para un buen aprovechamiento de los recursos disponibles.
Los alumnos pudieron comprender y utilizar herramientas propuestas sin grandes inconvenientes. Se logró fomentar la necesidad de utilización de herramientas CASE para optimizar los tareas de desarrollo en la construcción de software.
Vinculando este trabajo con los exámenes finales presentados hasta el momento, se puede concluir una profundización efectiva de los conocimientos de GCS y Testing.
Por otro lado un algunos de los estudiantes han comenzado la utilización de herramientas tales como SVN en sus proyectos de fin de carrera.

Bibliografía

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

[2] Carlo Ghezzi, Mehdi Jazayeri, Dino Mandrioli. Fundamentals of Software Engineering. Prentice Hall, 1991.

[3] Roger Pressman-Software Engineering. A Practitioner's Approach. Mc-GrawHill-5ta E 2001

[4] Inc Standish Group Internacional. The chaos report, 1996. Available at: http://www.standishgroup.com/..

[5] Briand, L. C.; Labiche, Y. and Miao, Y. (2003). Towards the reverse engineering of UMLsequence diagrams. 10th Conference on Reverse Engineering, Victoria (Canada), pp. 57-66.

[6] Booch G., Rumbaugh J., Jacobson I., The Unified Modeling Language. 2.0. Addison Wesley, Second Edition. 2005.

[7] Ivar Jacobson, Grady Booch, James Rumbaugh. El Proceso Unificado de Desarrollo de Software. Addison-Wesley, 2000.

[8] Erich Gamma, Richard Helm, Ralph Johnson, John Vlissedes. Design Patterns, Elements of Reusable Object/Oriented Software- Addison-Wesley, 1995.

[9] Buschmann F., Meunier R., Rohnert H., Sommerlad P.. Stal M.. Pattern-Oriented Software Architecture, Volume 1: A System of Patterns. 2002.

[10] Daniele Marcela, Florio Nicolás, Romero Daniel. Definición y uso de Plantillas Genéricas para la descripción de Casos de Uso. Proyecto de Innovación e Investigación para el Mejoramiento de la Enseñanza, Sec. Ciencia y Técnica, Sec. Académica, Universidad Nacional de Río Cuarto. RR Nº

302/04. 2004.

[11] Daniele Marcela, Romero Daniel. Evolución de Plantillas Genéricas para la descripción de Casos de Uso a Plantillas Genéricas para Análisis y Diseño. Proyecto de Innovación e Investigación para el Mejoramiento de la Enseñanza, Secretaría de Ciencia y Técnica y la Secretaría Académica de la Universidad Nacional de Río Cuarto. RR Nº 109/05. 2005.

[12]. Marcela Daniele, Daniel Romero. "Evolución de Plantillas Genéricas para la descripción de Casos de Uso a Plantillas Genéricas para Análisis y Diseño". VII Workshop de Investigadores en Ciencias de la Computación (WICC 2005), Universidad Nacional de Río Cuarto, Argentina, Mayo 2005. ISBN: 950-665-337-2.

[13]. Daniele Marcela, Romero Daniel, Martellotto Paola, Frutos Mariana. La enseñanza de gestión de proyectos de software y la aplicación de herramientas que favorezcan su automatización.(PIIMEG 2006). Del 01/08/2006 al 31/07/2008. RR Nº 499/06, Secretarías Académica y de Ciencia y Técnica de la Universidad Nacional de Río Cuarto.

[14]. Marcela Daniele, Paola Martellotto, Daniel Romero. "Fomentando el uso de herramientas
en las actividades de Gestión de Proyectos de Software". 10º Congreso Iberoamericano EDUTEC 2007, Universidad de Murcia, España, y Universidad Tecnológica Nacional. Buenos Aires. 23, 24 y 25 de octubre de 2007.

[15]. Marcela Daniele, Daniel Romero. "Extendiendo las plantillas genéricas para la definición de casos de uso con un framework genérico distribuido", III Congreso de Tecnología en Educación y Educación en Tecnología (TE&ET'08), Bahía Blanca, Argentina, 12-13/06/08.

[16]. http://sourceforge.net/projects/plarpebu/.

[17] JUnit. http://www.junit.org

[18] Subversion. http://subversion.tigris.org