Enseñanza de la ingeniería de software por procesos instrumentados

Rubby Casallas
Universidad de los Andes, Depto.de Ingeniería de Sistemas y Computación
Bogotá, Colombia
rcasalla@uniandes.edu.co

Jaime I. Davila
Universidad de los Andes, Depto.de Ingeniería de Sistemas y Computación
Bogotá, Colombia
jadavila@uniandes.edu.co

Juan P. Quiroga
Universidad de los Andes, Depto.de Ingeniería de Sistemas y Computación
Bogotá, Colombia
jquiroga@uniandes.edu.co

Abstract

In this paper we present a Software Engineering course that combines in an integrated and balanced way: processes, methodology and support tools to develop a software product. We are called our approach "Instrumented Process" to differentiate it from process like TSP™ where the phases and deliverables are defined but neither the methodology to be used during the activities and nor the support tools. In our course, the students have to develop a project in several cycles and in a team group of 5 or 6. They are instructed in the use of OO methodologies and trained in the use of tools like version control, testing, planning, etc.. Our experience have been successful with respect to the development of the project and also, with respect to the understanding of concepts and practices here taught because students use those in later projects.

Keywords: Software Engineering Education, Software Process, Team Work, Software Project Management

Resumen

En este artículo presentamos un curso de ingeniería de software que combina de una manera balanceada y práctica: procesos, metodología y herramientas de apoyo al desarrollo de un producto de software. Hemos llamado nuestro enfoque "proceso instrumentado" para diferenciarlo de procesos como TSP™ en donde se definen las fases y los entregables del desarrollo, pero no la metodología que ha de ser usada durante las actividades ni las herramientas de soporte. En nuestro curso, los estudiantes deben realizar un proyecto en varios ciclos y en grupos de 5 ó 6 personas. Ellos son instruidos en el uso de un proceso predefinido, entrenados en el uso de las metodologías OO y en las herramientas para apoyar tareas como el manejo de versiones, pruebas, planes, etc. Nuestra experiencia ha sido exitosa en cuanto al desarrollo del proyecto y en cuanto a que los estudiantes asimilan de tal manera los conceptos y prácticas aquí vistos que los usan en cursos posteriores y en sus trabajos de grado.

Palabras claves: Enseñanza en Ingeniería de Software, Procesos de software, trabajo en grupo, Administración de Proyectos

1 Introducción

Enseñar un curso de ingeniería de software en un programa de pregrado es una tarea retadora que ha sido enfocada de diversas maneras. Las opciones van desde el clásico curso donde se imparten aisladamente conceptos de administración de proyectos y metodologías, hasta una visión más dinámica en donde el estudiante se enfrenta a desarrollar, por primera vez, un proyecto grande con características similares a los proyectos en el mundo real. El primer enfoque tiene como desventaja principal que, al estar los conceptos aislados, es difícil para el estudiante tener una visón pragmática de cuándo usarlos y cómo integrarlos durante el desarrollo. El segundo, en muchos casos termina siendo un proyecto gigantesco que se convierte en largas noches de trabajo tratando de lograr que algo funcione y generando una gran frustración en los estudiantes cuando no se consigue satisfacer las expectativas.

Una alternativa más reciente son los cursos de ingeniería de software enfocados al proceso de desarrollo. Típicamente esta clase de cursos se apoyan en procesos como PSP [1] [2] o TSP [3]. Aunque se han reportando experiencias exitosas [4][5], este enfoque tiene como desventaja fundamental la poco clara integración entre la parte metodológica, el uso de herramientas y el proceso. El riesgo principal que se tiene es que, para el estudiante, el curso se convierta en "el proceso por el proceso" y que el proceso no se perciba como una herramienta más para desarrollar un producto. En muchas ocasiones las actividades se convierte en completar documentos no necesariamente adecuados para el tipo de proyecto. Además, si se tiene en cuenta que estos cursos de introducción a la ingeniería de software son tomados más o menos después del segundo año de carrera, es difícil que el estudiante ya domine metodologías y herramientas y se pueda concentrar sólo en la utilización de un proceso.

En este artículo, estamos documentando una alternativa a estas opciones. Nuestro enfoque logra una integración entre proceso, metodologías, y herramientas y por eso lo hemos llamado un proceso instrumentado. El proceso de base es inspirado en TSP, particularmente las ideas de roles y la de la organización del desarrollo en ciclos en donde incrementalmente se va construyendo el producto y mejorando el proceso. En cada una de las fases y actividades hemos integrado todos los aspectos metodológicos necesarios para desarrollar las tareas así como todas las herramientas de apoyo para las mismas. El curso está organizado alrededor de un proyecto y, además, va acompañado de clases teóricas y talleres. Impartimos conceptos teóricos relacionados con procesos y metodologías, y realizamos talleres de proceso y de uso de herramientas. El instructor trabaja muy de cerca con los grupos de estudiantes, logrando un seguimiento y una retroalimentación útil que minimiza el riesgo de no terminar a tiempo o no terminar exitosamente el proyecto.

Llevamos más de dos años aplicando este enfoque y hemos podido comprobar los buenos resultados del mismo de varias maneras. Primero, por el éxito de los grupos en términos del buen desempeño del equipo de trabajo, de la satisfacción de los requerimientos y de la calidad de los productos desarrollados. Segundo, hemos podido comprobar que hay una verdadera comprensión de los conceptos, un desarrollo de habilidades y el afianzamiento de una disciplina para desarrollar software con un alto compromiso de calidad. Esto lo hemos podido ver en cursos posteriores y en proyectos de fin de carrera donde los estudiantes usan las prácticas aquí asimiladas.

El artículo está organizado de la siguiente manera: En la sección 2 presentamos el contexto, los objetivos y una descripción global del curso. Luego, en la sección 3, hacemos una descripción detallada del proceso instrumentado. En la sección 4 discutimos acerca de la relación entre los grupos de trabajo y el instructor. En la sección 5 presentamos un análisis de la experiencia pedagógica y terminamos el artículo con algunas conclusiones y reflexiones sobre el trabajo futuro.

2 Contexto y descripción global del curso

En 1999 el departamento de sistemas de la Universidad de los Andes adelantó un proceso de cambio curricular del programa de pregrado en Ingeniería de Sistemas y Computación. El cambio curricular buscaba como objetivos fundamentales modernizar el contenido del currículo de tal forma que el estudiante pudiera: "adquirir conocimientos, generar habilidades y destrezas, desarrollar y afianzar actitudes y valores que le permitan vivir y practicar la profesión de forma competente"[6].

Una preocupación fue la de incluir aspectos transversales a varios cursos como el compromiso con la calidad de las aplicaciones desarrolladas, la responsabilidad y la participación en el trabajo en grupo, los aspectos metodológicos y el uso de herramientas. Dentro de la implementación del cambio, se redefinieron los cursos básicos de programación para incluir lenguaje de programación orientados objetos y se incluyó dentro de los cursos algunos aspectos de PSP[1] como la planeación, el control de tiempos y el manejo de defectos. Dentro de los cursos básicos también se definió un curso de especificación y modelaje y un curso de diseño de algoritmos. Luego de esta formación básica los estudiantes deben tomar el curso introductorio a la ingeniería de software. El mensaje principal que queremos

transmitir al estudiante con este curso es que tanto el proceso como las metodologías y las herramientas computacionales son sólo herramientas de apoyo y ayuda para lograr el objetivo principal que es producir software que satisfaga los requerimientos y que sea de buena calidad. Esperamos que al finalizar el curso el estudiante sea capaz de:

- Seguir un proceso de desarrollo de software
- Definir roles y asignar responsabilidades en el desarrollo de un proyecto
- Definir objetivos de calidad del proceso y del producto de manera cuantitativa
- Mejorar el proceso de estimación de tamaño y esfuerzo para el desarrollo de software
- Mejorar el proceso de planeación y seguimiento de un proyecto
- Coordinar mejor las distintas actividades administrativas y de desarrollo de un proyecto
- Entender mejor las actividades de ciclo de vida de un proyecto y estar en capacidad de realizarlas (análisis, diseño, implementación, pruebas)

El curso gira alrededor del desarrollo de una aplicación de aproximadamente unas 5000 líneas de código escritas en java. El proyecto se realiza durante 16 semanas en las que se espera que cada alumno trabaje 3 horas adicionales a las 3 horas de clase. Cada grupo de trabajo es conformado por 5 o 6 estudiantes, cada uno de ellos actúa como desarrollador y desempeña un rol administrativo particular similar a aquellos definidos en TSP [3].

La estrategia global de desarrollo consiste en definir el número de ciclos que se realizarán (2 ó 3) y en qué consistirá cada uno de ellos. La definición de esta estrategia es realizada conjuntamente con el instructor quien discute con el grupo la arquitectura global del producto y ayuda a definir los ciclos de tal forma que sea posible en el primer ciclo empezar el desarrollo del producto mientras se estudian conceptos teóricos de proceso, metodología y uso de herramientas. Por esta razón, casi siempre el primer ciclo es el de mayor duración (alrededor de 8 semanas), y en él se implementa la funcionalidad básica del producto.

Es importante hacer notar que la descripción del proyecto que se entrega a los estudiantes es relativamente clara. Con base en esta definición, los estudiantes deben hacer casos de uso usando UML [7 y una especificación de los requerimientos. Los aspectos de levantamiento de requerimientos con clientes y/o usuarios finales está por fuera del alcance de este curso. Sin embargo, como de todas maneras puede haber ambigüedades o dudas sobre lo solicitado, creamos un foro de preguntas y respuestas que es administrado por uno de los instructores y allí se resuelven públicamente las dudas.

Los ciclos 2 y 3 son más cortos y varían entre 2 y 6 semanas. Desde el punto de vista de funcionalidad, en estos ciclos se incluyen aspectos como inclusión de funciones más complejas, o la interfaz gráfica, o soporte para internacionalización, localización o seguridad para la aplicación. Desde el punto de vista del proceso, es en estos ciclos donde los estudiantes tienen la oportunidad de mejorar su propio proceso, ajustándolo de acuerdo con su desempeño y, también, pueden hacer un uso mejor de las herramientas computacionales dado que ya han adquirido experiencia manejándolas.

En la próxima sección describimos en más detalle las actividades y su relación con las metodologías y las herramientas.

3 Descripción del proceso instrumentado

Para el desarrollo del proyecto los estudiantes utilizan un proceso instrumentado que se adapta y mejora durante los ciclos. El proceso es el conjunto de actividades y el orden en el que deben desarrollarse; la instrumentación se refiere a que, para cada actividad, se define de manera precisa la metodología que debe ser usada, los entregables que van a producirse con sus respectivos formatos y las herramientas de apoyo. La figura 1 ilustra la organización global del proceso instrumentado:

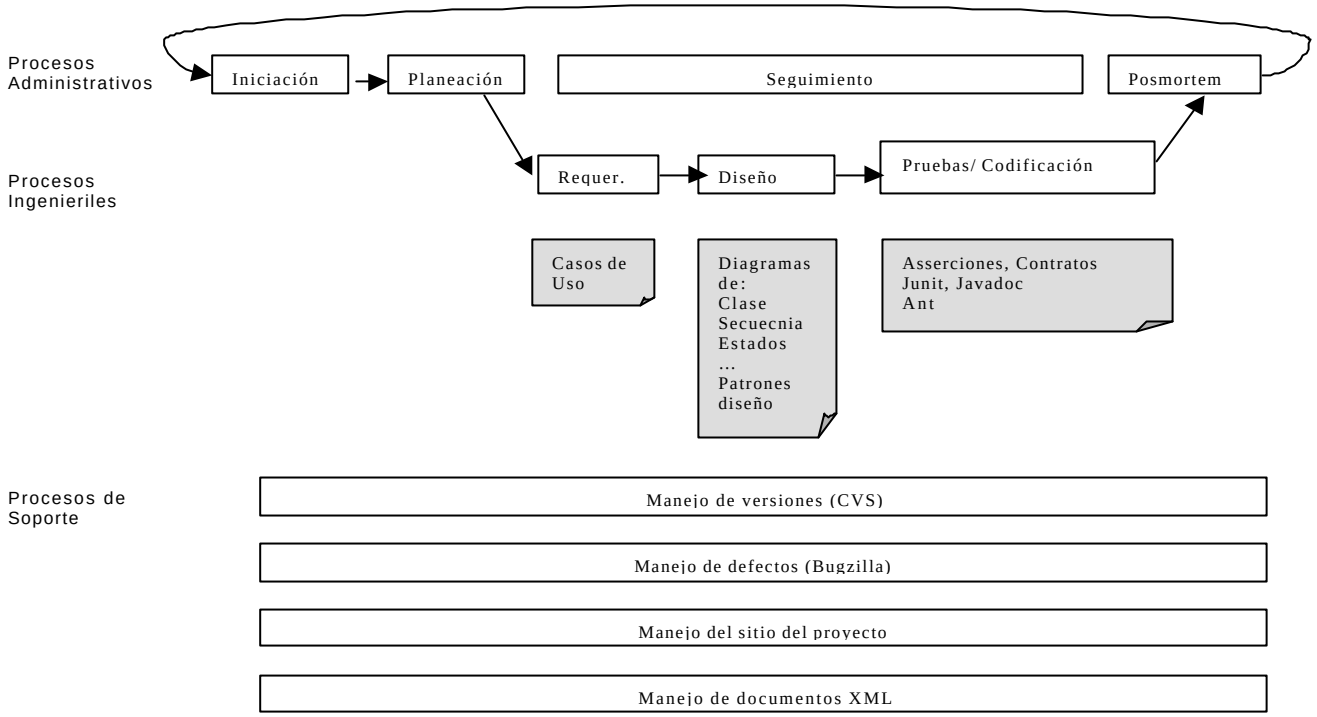

Figura 1

En el resto de la sección describimos los elementos en la figura 1.

3.1 Procesos de administración de proyectos

Por procesos de administración de proyectos hacemos referencia a la organización del grupo de trabajo, la planeación del proyecto, el seguimiento y la evaluación o postmortem. Estas actividades deben ser realizadas en cada ciclo de desarrollo y el grupo de trabajo debe asimilar que se está en un proceso de mejoramiento continuo, esto es, al finalizar cada ciclo se redefinen las metas de trabajo y se adapta el proceso y/o la organización del grupo para corregir desviaciones que puedan poner en riesgo el éxito del proyecto. A continuación describimos las actividades.

3.1.1 Iniciación

En la fase de Iniciación se conforman los grupos, se asignan los roles a los participantes y se definen las metas individuales y grupales que van a ser evaluadas al final del ciclo. Adicionalmente, se identifican los riesgos del proyecto, se hace un diseño y una estimación preliminar del tamaño del producto. Con base en este diseño preliminar y con la ayuda del instructor, el grupo define cuántos ciclos de desarrollo se van a realizar y cuál será, a grandes rasgos, el contenido de cada uno de ellos.

Como resultado de esta actividad inicial se produce un entregable que contiene la información del grupo, la estrategia de desarrollo y los riesgos del proyecto.

3.1.2 Planeación y Seguimiento

En la actividad de planeación se estima el esfuerzo de cada una de las actividades por realizarse durante el proyecto. Para el primer ciclo, el instructor da una métrica de esfuerzo y una estimación sobre el porcentaje de tiempo requerido para cada actividad. Con base en estas métricas, el grupo puede realizar la planeación del primer ciclo y para los ciclos posteriores, el grupo usa la experiencia adquirida para recalibrar sus métricas y mejorar la planeación.

A lo largo de todo el proyecto se realiza la actividad de seguimiento en donde se actualiza la información referente al esfuerzo real invertido a medida que el proyecto transcurre. Se prepara un informe semanal que registra el seguimiento de los riesgos y el desempeño del grupo en general. En la sección 4.2 extendemos este aspecto del seguimiento del proyecto porque nos parece importante resaltar cómo interviene el instructor en esta actividad.

Tanto para la planeación como para el seguimiento se utilizan hojas de cálculo para registrar la información.

3.1.3 Postmortem

En el postmortem se obtienen las conclusiones respecto al ciclo. La actividad principal es la evaluación del cumplimiento de los objetivos, de la conveniencia de la estrategia de desarrollo, del adecuado manejo de los riesgos y del correcto desempeño de los participantes. Desde el punto de vista de aprendizaje del estudiante, esta es tal vez la actividad más importante ya que les permite, de manera constructiva, criticar el desempeño y hacer una evaluación de cada uno de los miembros del grupo. Además, gracias a esta evaluación los estudiantes pueden mejorar el proceso que están usando.

3.2 Procesos de ingeniería del proyecto

Los procesos de ingeniería del producto comprenden las actividades de construcción propiamente dicha: Definición de requerimientos, diseño, codificación y pruebas. A continuación describimos estas fases, haciendo énfasis en los aspectos metodológicos.

3.2.1 Definición de Requerimientos

La metodología de desarrollo que se usa para realizar los proyectos es orientada por objetos. Durante el análisis de requerimientos y el diseño, se usan los diagramas de UML [7] y herramientas como ArgoUML [8], dia[9] o Rose [10] para graficarlos. Como actividad inicial para entender el problema que se quiere resolver se desarrollan casos de uso y luego se documentan los requerimientos funcionales y no funcionales y se establecen prioridades sobre los mismos. Paralelamente a la elaboración del documento de requerimientos se desarrolla el plan de pruebas de sistema para la aplicación.

Como mencionamos antes, para resolver ambigüedades y preguntas, se define un foro electrónico en donde uno de los instructores es responsable de contestar preguntas y resolver dudas acerca de lo que se espera del producto. A través de este foro se logra uniformizar la manera en que los estudiantes entienden los requerimientos del proyecto, evitando que estos desarrollen cosas diferentes.

3.2.2 Diseño

En esta fase se utiliza como lenguaje de diseño los diagramas de clase, de secuencia y de estados de UML. Durante las sesiones teóricas que se llevan en paralelo con el desarrollo del proyecto, se introduce tempranamente algunos patrones de diseño básicos [11], el patrón de MVC [12], Observador [13], Fachada [11], etc.. En casos en que la interacción entre el usuario y la aplicación es muy importante, se usan los diagramas de secuencia para expresar el control de diálogo.

Cabe resaltar que al tema de diseño se le presta bastante atención durante el curso. El instructor realiza varias clases teóricas y talleres para discutir más detenidamente patrones de diseño y/o alternativas arquitectónicas. Estos conceptos son aplicados por los estudiantes en la elaboración del diseño del producto y reciben retroalimentación por parte del instructor quien verifica si el diseño elaborado satisface los requerimientos o no.

3.2.3 Codificación/Pruebas

Hay tres aspectos fundamentales que queremos resaltar aquí: la programación por contrato [14], la documentación del código y la elaboración de pruebas unitarias. La programación por contrato es en realidad parte de lo que llamamos diseño detallado. La definición de los contratos (aserciones, pre y post condiciones) quedan inmersos dentro del código y gracias a herramientas como javadoc [15] o doxygen [16] son generados con el resto de la documentación interna del código.

Inspirados en una de las premisas de eXtreme Programming XP[17] [28] las pruebas unitarias se desarrollan antes de hacer la programación y una vez que se han definido los contratos. Para esto, los estudiantes usan JUnit [18] [28] que es un ambiente de desarrollo de pruebas que permite la programación de pruebas para código java y la ejecución sistemática de las mismas.

3.3 Herramientas adicionales

Además de las herramientas ya mencionadas, existen otras herramientas que ayudan a dar soporte al desarrollo del proyecto en su totalidad. Estas herramientas son los formatos de documentación, el sitio del proyecto, el manejador de versiones y el manejador de defectos.

3.3.1 Formatos de documentación en XML

En cuanto a los documentos que los estudiantes deben producir como entregables de las actividades, hemos trabajado en dos aspectos. Primero, en la definición de cuál debe ser el contenido de los documentos de acuerdo con la metodología usada en la actividad y, segundo, en la automatización de la generación de documentos en diversos formatos. Para la definición de contenido, hemos usado XML [19] y hemos definido un DTD para cada tipo de documento. Para la generación de la documentación, hemos definido hojas de estilo en XSL [20] que permiten la transformación de tales documentos al formato DocBook [21]. La descripción completa de tales documentos se encuentra en [22]. Una de las ventajas de transformar a DocBook es que se pueden generar los documentos y los manuales en formatos como HTML, PS, PDF y RTF.

3.3.2 Sitio web *de los grupos*

Para facilitar el seguimiento del proyecto y la visibilidad del trabajo tanto entre los miembros del grupo y el instructor, cada grupo administra un sitio web del proyecto.

A partir de los documentos XML que describen los entregables de cada una de las fases, así como los integrantes y roles de cada uno de los grupos, generamos automáticamente, de nuevo usando transformaciones XSL, un *esqueleto* del sitio *web* que los grupos irán actualizando a lo largo del proyecto con los documentos entregables de cada fase. Los enlaces a los entregables están predefinidos y organizados de acuerdo con el proceso que los estudiantes van a usar para el desarrollo. Un ejemplo de uno de estos sitios puede ser visitado en [23].

3.3.3 Manejador de versiones

Toda la documentación del proyecto, el código fuente y las pruebas están bajo control de versiones. El manejador que usamos es CVS [24] y cada uno de los grupos tiene un depósito CVS en el que almacenan las versiones de los documentos entregables en formato XML. Es responsabilidad de uno de los integrantes del grupo hacer la administración del depósito y velar por su consistencia.

Adicionalmente al manejador de versiones, otra herramienta útil, especialmente en las actividades de codificación y pruebas, es Ant [25] [28]. Esta es una herramienta que permite la automatización de diversas actividades desarrolladas dentro del proyecto, a través de reglas y comandos similares a los de los *Makefile*. Ant es utilizado para automatizar las labores de actualización del sitio *web*, del depósito CVS, de ejecución de pruebas, compilación y generación de documentación.

3.3.4 Manejador de defectos

Bugzilla [26] es una herramienta *web* que permite administrar los errores detectados en los diversos grupos y seguir los diferentes estados por los que pasan éstos, hasta ser resueltos. Se utiliza como herramienta de retroalimentación con respecto a los diversos documentos entregables y con respecto a los errores encontrados en las entregas.

Con el uso de esta herramienta se puede garantizar que los pasos no se saltan. Esto es, por ejemplo, que no se empieza la programación antes de haber terminado el diseño.

4 Papel del Instructor en el Proyecto

El instructor juega un rol fundamental en este curso, no sólo porque es responsable de impartir los conceptos teóricos y de realizar talleres sobre procesos, metodologías y herramientas, sino porque también es parte integral de los grupos de trabajo. Su papel dentro de los grupos de trabajo es el de aseguramiento de calidad ya que es responsable de verificar y validar los entregables a medida que el grupo avanza en el desarrollo y, también, de administrar los defectos encontrados con la ayuda del manejador de defectos Bugzilla. En las secciones siguientes describimos con un poco más de detalle estas responsabilidades.

4.1 Consideraciones sobre los talleres.

Para la utilización de las herramientas, así como para algunas de las actividades realizadas durante el desarrollo del proyecto, el instructor realiza talleres de apoyo en los que se utiliza cerca de la cuarta parte de las horas dedicadas a la clase. El objetivo de los talleres es ayudar a los estudiantes a general destreza en el uso de las herramientas de tal forma que puedan ser usadas de manera productiva durante el proyecto. Los principales temas de talleres son:

1. Documentación estructurada usando XML.
2. Manejo de versiones utilizando CVS y WinCvs [27].
3. Taller de planeación.

4. Prácticas de inspección de código.
5. Principios de diseño, utilizando patrón MVC.
6. Automatización de tareas usando Ant.
7. Pruebas unitarias con JUnit.

4.2 Seguimiento del proyecto

El sitio del proyecto corresponde al lugar donde cada uno de los entregables del proyecto son colocados en forma de páginas html, para que puedan ser consultadas por el instructor del curso y los integrantes del grupo. El acceso se realiza a través de un navegador web, de forma que se pueda observar la información en cualquier momento y en cualquier sitio que se encuentren.

La forma en que se accede a los documentos, de forma que no se requiere tenerlos físicos sino de manera electrónica, hace que los estudiantes pueden tener una retroalimentación más rápida, que no requiere la presencia del instructor para discutir al respecto. Esto permite indicar problemas que se van encontrando en documentos críticos como el análisis de requerimientos y el diseño, evitando tener problemas graves en la calidad general del producto que ocasione que finalmente no se cumpla con las especificaciones requeridas.

A medida que los estudiantes avanzan durante el proyecto, este sitio constituye una fuente importante para el seguimiento del proceso y la evaluación de cada uno de los entregables. Esto se hace al observar en la fecha que se solicitan los documentos, si efectivamente estos documentos han sido colocados o no. Si se colocan, se puede realizar la retroalimentación respectiva; si no, se pueden tomar medidas de manera que el documento se coloque en el sitio en el menor tiempo posible, evitando que estos sean olvidados en el camino, para lograr que el proceso se desarrolle en su totalidad.

El instructor es responsable de las actividades de control de defectos, revisiones de los entregables y revisión de los reportes semanales.

4.2.1 Control de defectos de los entregables.

Un aspecto importante, que hace que tanto el proceso como el producto en sí sea de alta calidad, es la detección temprana de defectos y el seguimiento que se hace sobre su corrección hasta que realmente sean eliminados de los entregables.

Para facilitar la gestión y control de los defectos, se utiliza la herramienta Bugzilla. Mediante esta herramienta los estudiantes conocen, arreglan los defectos y al final reportan que el defecto ha sido efectivamente reparado. La herramienta facilita que el instructor y el grupo sepan cuántos defectos hay sobre cada uno de los entregables y le ayuda al instructor a decidir si recibe o no un ciclo con base en los defectos aún abiertos sin resolver. Esto hace que los estudiantes se comprometan a corregir los problemas de una manera rápida, permitiendo tener siempre presente la calidad total del producto.

4.2.2 Revisiones e inspecciones

Las revisiones e inspecciones de los entregables del proceso son un aspecto importante para la calidad del producto, que es observado no solamente como un elemento más en el plan de temas del curso, sino que se solicita a los estudiantes realizar estas actividades.

Las inspecciones constituyen una manera importante para que el grupo se dé cuenta de los problemas que tengan en los documentos y en el código. Mediante lo anterior se puede observar defectos recurrentes y ambigüedades en documentos que normalmente son redactados de manera informal, en este caso para el instructor, que hace las veces de cliente.

4.2.3 Reportes semanales

Otra de las formas de hacer seguimiento a los proyectos es a través de un informe muy sencillo en el cual cada uno de los integrantes del grupo indica qué problemas ha tenido y qué actividad ha realizado durante la semana.

Normalmente es un espacio bastante informal, donde se pueden observar algunas estrategias tomadas por los grupos debido a problemas internos y otros debidos al desconocimiento de las herramientas que deben utilizar durante el desarrollo.

Adicionalmente, con este documento ellos describen cuáles son las actividades para realizar la siguiente semana, y cuáles quedan pendientes de la semana anterior. Esto, de alguna forma, es un indicio de qué tan atrasados o qué tan

adelantados van con respecto a las metas esperadas.

5 Experiencia Pedagógica

El curso ha sido impartido durante 5 semestres con un promedio de 7 grupos de estudiantes por semestre. Los resultados obtenidos, en términos de la eficiencia en el trabajo en grupo y de la utilización del proceso instrumentado para producir un producto de buena calidad, han sido muy exitosos. Otro de los logros interesantes con este curso es que ha habido una comprensión por parte de los estudiantes de los conceptos fundamentales impartidos. Esta asimilación de conceptos y generación de habilidades se refleja en cursos posteriores donde los estudiantes deben desarrollar un proyecto, y, a pesar de que el énfasis no es ingeniería de software, los estudiantes usan tanto el proceso como las herramientas vistas para apoyar su desarrollo.

A continuación revisamos algunas de las metas inicialmente planteadas.

5.1 El trabajo en grupo

La manera como se realiza la organización del grupo de desarrollo es totalmente inspirada en las ideas de TSP. El que existan roles y tareas bien definidas para cada uno de los integrantes contribuye enormemente a que todos participen. Además, la visibilidad de los compromisos, la planeación individual y los reportes semanales ayudan a que todos contribuyan y también a evitar que se presenten cargas de trabajo no balanceadas.

Todos los elementos mencionados son fundamentales para que el trabajo en grupo se realice en forma responsable. En caso de problemas en el grupo, estos se pueden detectar a tiempo y, en general, el mismo grupo sugiere mecanismos de corrección en los informes de postmortem.

5.2 Calidad del Proceso

Como ya hemos mencionado, queremos transmitir al estudiante que el proceso es una herramienta más para lograr un producto de calidad. En este sentido, un proceso de software es de calidad si contribuye en el desempeño eficiente y eficaz del grupo para producir el producto que satisfaga los requerimientos dentro de los plazos previstos.

Con respecto a esta definición de proceso, nuestro enfoque ha dado muy buenos resultados. Lo primero que hay que resaltar es que, al incluir en el proceso la metodología y las herramientas, se logra una visión integrada de todos los aspectos del desarrollo y se facilita la elaboración de las tareas. Además, el que el desarrollo sea en ciclos permite que los estudiantes puedan corregir cualquier falla del proceso o adaptarlo a su caso particular. Por último, el que haya un énfasis en el seguimiento del proyecto y en especial en la verificación y validación de los productos intermedios disminuye el riesgo de fracaso en el desarrollo.

El postmortem, por su lado, representa la forma de evaluación más eficaz del proceso, ya que indica los resultados desde el comienzo del ciclo hasta su finalización. Los datos obtenidos indican la mayoría de las veces fallas en la planeación, pero que a medida que se avanzaba en los ciclos se mejora, de forma que los porcentajes de error en los tiempos planeados de las actividades se encuentran en un rango de 0 a 10%, que es un nivel razonable.

5.3 Calidad del producto

Aquí definimos calidad del producto con respecto a dos aspectos: satisfacción de los requerimientos funcionales y no funcionales y baja densidad de defectos por líneas de código. Las estrategias que usamos para satisfacer estos puntos son, primero que todo, una buena definición de requerimientos y, después, la detección temprana de defectos y la ejecución efectiva y eficiente de pruebas.

Gracias al papel del instructor en la revisión de los entregables y la retroalimentación de los mismos, se logra una detección temprana de defectos. Además, estos defectos se administran a través de la herramienta Bugzilla que permite, entre otras cosas, tener una visibilidad clara del estado del proyecto en cualquier momento.

Con respecto a las pruebas eficientes y eficaces, se realizan dos tareas básicas: la elaboración temprana del plan de pruebas y la programación de las pruebas usando la herramienta JUnit. La programación de las pruebas y el uso de un manejador de versiones es muy importante para asegurar la calidad del producto ya que permite realizar pruebas de regresión, sincronizar el trabajo y evitar inconsistencias.

6 Conclusiones y Trabajos Futuros

Para convencer a los estudiantes sobre la idea de que un proceso de software es una herramienta más en el desarrollo de una aplicación, es importante que los estudiantes tengan éxito en la realización del proyecto. Este éxito significa

que son capaces de construir un producto de buena calidad (entendida como requerimientos satisfechos, detección de errores temprana y pruebas efectivas) dentro de las restricciones de tiempo y recursos dadas.

Creemos que nuestro enfoque de proceso instrumentado es una opción viable y muy satisfactoria para introducir los aspectos fundamentales de la ingeniería de software en un currículo de pregrado. Este enfoque nos permite integrar de manera balanceada proceso, metodología y herramientas. Diversos aspectos tanto administrativos como de ingeniería de producto y de soporte a las actividades son cubiertos globalmente durante el desarrollo del proyecto.

El papel del instructor es fundamental durante este curso ya que él no sólo se ocupa de los aspectos teóricos del curso sino que participa activamente en el desarrollo del proyecto sirviendo de asegurador de la calidad. Los estudiantes aprenden a seguir un proceso y también a ser críticos con respecto a la utilidad de éste para lograr sus metas. El que se desarrollen varios ciclos les permite ajustar su proceso y adaptarlo a sus necesidades.

Actualmente estamos trabajando diferentes aspectos con miras a mejorar este curso y a ofrecer también material de apoyo a otros cursos. Dentro de nuestros trabajos en curso más importantes queremos destacar:

- Proceso Flexible: De nuestra experiencia en este curso, y en general en nuestros proyectos de desarrollo, hemos podido comprobar que hay unos procesos, metodologías y herramientas más adecuados que otros según el tipo de proyecto, la duración del proyecto, el número de integrantes, la tecnología que se va a utilizar, los usuarios del producto, etc. El objetivo es, entonces, poder adaptar el proceso a las características específicas de un proyecto de manera que sea más eficiente su administración y la construcción del producto.
- Integración de herramientas: Queremos poder integrar las herramientas utilizadas en el proyecto a través de los documentos que se manejan. Para esto, queremos mejorar la forma como se definen las estructuras de los documentos en XML, como se editan, transforman y como se interpretan en cada una de las herramientas.

Por último, en la medida en que nuestro sistema de métricas mejore, podremos hacer estudios más concienzudos y comparativos sobre el uso de estos proceso entre los estudiantes.

7 Agradecimientos

Queremos agradecer al Profesor Martín Soto por su contribución en la definición de este curso en sus etapas iniciales.

8 Referencias

[1] Watts S. Humphrey, Introduction to the Personal Software Process, 1997. Addison Wesley Longman, Inc..

[2] Watts S. Humphrey, A discipline for software engineering, 1995, Addison Wesley Longman, Inc..

[3] Watts S. Humphrey, Introduction to the team software process, 2000, Addison Wesley Longman, Inc., 0-201-47719-X.

[4] P. Runeson. Experiences from Teaching PSP for Freshmen. 14th Conference on Software Engineering Education and Training. 2001. Charlotte, North Carolina. USA

[5] D. Groth and E. Robertson. It's All about Process: Project Oriented Teaching of Software Engineering. 14th Conference on Software Engineering Education and Training. 2001. Charlotte, North Carolina. USA

[6] Cambio Curricular.Programa de Ingeniería de Sistemas y Computación. Universidad de los Andes. Bogotá. Colombia. 1999 (Documento Interno)

[7] Ivar Jacobson, Grady Booch, y Rumbaugh James, The Unified Modeling Language Reference Manual, 1999, Addison Wesley Longman, Inc., 0-201-30998-X.

[8] Alejandro Ramirez, Philippe Vanpeperstraete, Andreas Rueckert, Kunle Odutola, Jeremy Bennett. ArgoUML User Manual: A tutorial and reference description of ArgoUML. http://argouml.tigris.org/documentation/defaulthtml/manual/, 2002

[9] George Harry. Dia Tutorial. http://www.lysator.liu.se/~alla/dia/, 2000.

[10] http://www.rational.com/

[11] Craig Larman, Applying UML and patterns: an introduction to object oriented analysis and design , 1998, Prentice Hall, 0-13-7488807.

[12] Frank Buschmann, R. Rohnert, H. Sommerland, y P. Stal, Pattern oriented software architecture: A system of patterns, 1996, John Wiley && Sons, 0-47-1958697.

[13] Erich Gamma, Richard Helm, Ralph Johnson, y John Vlissides, Design Patterns: Elements of Reusable Object-Oriented Software, Addison-Wesley, 1995, 0-201-63361-2.

[14] Computer, IEEE, 25, 10, October 1992, Bertrand Meyer, “Applying "Design by Contract" 40-51.

[15] James Gosling, Bill Joy,Guy Steele. The Java Language Specification. Chapter 18: Documentation Comments.http://java.sun.com/docs/books/jls/first_edition/html/18.doc.html, 1996

[16] (http://www.stack.nl/~dimitri/doxygen/manual.html), Dimitri van Heesch, Doxygen Manual.

[17] Kent Beck, Extreme Programming Explained. Addison Wesley, 1999. ISBN 0201616416.

[18] (http://junit.sourceforge.net/doc/cookstour/cookstour.htm), Erich Gamma y Kent Beck, JUnit: A cook's tour, Java Report.

[19](http://www.xml.com/pub/a/98/10/guide0.html), A Technical Introduction to XML, Norman Walsh, 1998.

[20] XSL Transformations (XSLT) Version 1.0. W3C Recommendation 16 November 1999. http://www.w3.org/TR/xslt

[21] Norman Walsh, DocBook: The Definitive Guide, O'Reilly & Associates .

[22](http://abadon.uniandes.edu.co/ingsw/documentos/tutorial/tutorial.html), Camilo Camacho, Documentos de Ingeniería de software: Uso y Descripción, 2002, Proyecto Dirigido.Universidad de los Andes .

[23] http://xue.uniandes.edu.co/~is252c07.

[24] Karl Foegel, Open Source Development with CVS, Coriolis, Inc..

[25] Stephane Bailliez et al. Apache Ant User Manual. 2002.. http://jakarta.apache.org/ant/manual/index.html

[26] Barn2002 (http://www.pjrc.com/bugzilla/docs/html/index.html), Matthew Barnson, The Bugzilla Guide.

[27] Dan Harper. WinCvs Version 1.1: Users Guide. 1999.

http://www.wincvs.org/winhtml/wincvs11.htm

[28] Rick Hightower y Nicholas Lesiecki, Java Tools for Extreme Programming: Mastering Open Source Tools, including Ant, JUnit and Cactus, John Wiley and Sons, 2001, 0-471-20708-X.