LATIN AMERICAN CONGRESS ON REQUIREMENTS ENGINEERING & SOFTWARE TESTING
LACREST MEDELLÍN 2012. July 13-14, Medellín, Colombia.

Redmine and Subversion integration in the construction of a test platform for software projects

Integración de Redmine y Subversion en la construcción de una plataforma para la prueba de proyectos de software

Guillermo Esteban Murillo Goussen
CITIC, Universidad de Costa Rica
Alajuela, Costa Rica
guillermo.murillogoussen@ucr.ac.cr

Gabriela Salazar Bermúdez
CITIC, Universidad de Costa Rica
San José, Costa Rica
gabriela.salazar@ucr.ac.cr

ABSTRACT
During the last decade, universities, state institutions and representatives of industry have interacted to contribute to improving the quality of software produced in Costa Rica. This year was established at the University of Costa Rica, the Research Center on Information Technology and Communication (CITIC). Within its structure is proposed to constitute research units, and one of them is the Software Quality Assurance Laboratory (LACSOFT) that is in process of creation and for which there are tools under development to support the management of administrative processes. This article describes the results of an investigation of two free software tools (Redmine and Subversion) that, integrated, could be incorporated into LACSOFT, to facilitate the management of their projects software verification and validation.

Resumen
Durante la última década, las universidades, las instituciones del estado y los representantes de la industria han interactuado para contribuir al mejoramiento de la calidad del software producido en Costa Rica [1]. Este año se creó en la Universidad de Costa Rica (UCR) el Centro de Investigaciones en Tecnologías de la Información y Comunicación (CITIC). Dentro de su estructura se propuso constituir Unidades de Investigación, y una de ellas es el Laboratorio de Aseguramiento de la Calidad del Software (LACSOFT) que está en proceso de creación y para la cual se trabaja en la definición de herramientas que permita dar soporte a la gestión de sus procesos administrativos [2]. Este artículo describe los resultados de una investigación de dos herramientas de software libre (*Redmine y Subversion*) que integradas, pudieran ser incorporadas en LACSFOT para facilitar la administración de sus proyectos de verificación y validación de software.

Categories and Subject Descriptors
D.2.4 Software/Program Verification: Validation, D.2.9 Management: Software quality assurance (SQA), K.3.2 Computer and Information Science Education: Computer science education

General Terms
Management, Design, Experimentation, Standardization, Verification.

Keywords
Redmine, Subversion, Project Managment, Software Testing

Palabras clave
Redmine, *Subversion*, Administración de proyectos, Pruebas de Software.

1. INTRODUCCIÓN
Hoy día poseer una administración efectiva de la información es un elemento crítico para el éxito de cualquier proyecto. La creciente dependencia de la información y los sistemas que la proporcionan han convertido a los sistemas de información en activos muy valiosos dentro de las empresas. Sin embargo, se ha vuelto común en el desarrollo de software, proyectos que no concluyen a tiempo, que exceden su presupuesto, que se abandonan, o que concluyen y no se utilizan por no satisfacer las expectativas del usuario.

Destinar esfuerzos en pro del mejoramiento de la calidad del software producido, ha sido en la última década, común denominador entre la industria y la academia. Se espera que uno de estos puntos de confluencia sea LACSOFT, laboratorio creado con el propósito de: desarrollar y extender la cultura del aseguramiento de la calidad del software, así como promover,

facilitar y estimular el uso de metodologías de alcance mundial o regional para mejorar la competitividad de las empresas. Para lograrlo, LACSOFT trata temas de investigación básica y aplicada relacionados con la mejora y la certificación de la calidad de los procesos, productos y proyectos que realiza una organización TIC, considerando las pruebas funcionales del software como el primer servicio que brindará.

Por otro lado, LACSOFT como institución universitaria tiene la responsabilidad de asegurar un ámbito adecuado, i.e. un laboratorio bien equipado, para la práctica profesional necesaria en carreras tanto de grado como de posgrado, de manera que los egresados contribuyan de forma inmediata al crecimiento productivo, económico y social basados en nuevas tecnologías desarrolladas con calidad.

Uno de los componentes claves de todo proceso de desarrollo de software, son las actividades de Aseguramiento de la Calidad del Software (ACS) y entre ellas las actividades de verificación y validación (V&V) del software que se realizan durante las diferentes fases que del ciclo de vida de los sistemas. Los objetivos principales del proceso de V&V son: verificar la calidad de los productos obtenidos en cada fase del ciclo de vida y validar que el producto final cumpla con los requerimientos del software establecidos.

Buscando garantizar la calidad del software es que han surgido distintas herramientas y aplicaciones, utilizables a lo largo del ciclo de vida del proyecto. *Redmine* es una de ellas y fue seleccionada para esta investigación debido a sus buenas referencias y sobre todo a la experiencia positiva que ha tenido la Universidad de Costa Rica al utilizarla.

En la siguiente sección se describen brevemente las características que hicieron de *Redmine*, herramienta candidata para la investigación. Posteriormente se mencionan algunas de las ventajas de utilizar *Subversion* como manejador de versiones de un proyecto. Posteriormente se explica cómo se configuró *Redmine* para funcionar como plataforma de pruebas de software. Se especifican también los complementos instalados para incrementar la funcionalidad de *Redmine* y cómo se logró su integración con *Subversion*. Adicionalmente se describe la experiencia al poner a prueba la plataforma, con estudiantes del curso de Ingeniería de Software, en un proceso de validación de requerimientos funcionales, simulando un ambiente real de trabajo. Finalmente, se exponen los problemas encontrados y las conclusiones obtenidas.

2. MARCO TEÓRICO

2.1. Redmine

Redmine es una herramienta web dedicada a la administración de proyectos y al seguimiento de errores, desarrollada con el *framework Ruby on Rails*. Para su instalación se requiere poseer en el servidor web: *Ruby*, un gestor de bases de datos (*MySQL*, *PostgreSQL*), *Rack* y *Ruby on Rails*. Es una herramienta multiplataforma, a nivel de cliente sólo requiere una conexión a la red y un navegador web (cualquiera) para comenzar a utilizarla. *Redmine* es gratuito, puede descargarse desde el sitio oficial www.redmine.org (el último *release* estable, versión 1.2.1, fue lanzado el 11 de julio del 2011) y se distribuye bajo la licencia GNU *General Public License* [3].

La principal característica que convirtió a *Redmine* en una herramienta atractiva para la presente investigación es su enorme flexibilidad. Si bien es cierto, la aplicación fue pensada para el ámbito empresarial, su facilidad de configuración permite utilizarla en la administración de cualquier proyecto, independientemente del área en la que se desarrolle. Además, la coordinación de tareas, comunicación entre los participantes y la integración con sistemas de versiones, terminan por inclinar la balanza al comparársele con otras herramientas para la gestión de proyectos [4].

Específicamente para la administración de proyectos, resultó beneficioso para esta investigación, el hecho de que *Redmine* permite que cada proyecto posea una configuración propia. Es posible activar o desactivar los módulos para noticias, peticiones, control del tiempo, documentos, y además agregarles campos personalizados (de variedad de formatos: texto, lista, numérico, fecha) que le confieren una identidad propia a cada proyecto. Además un mismo usuario puede ocupar un rol distinto en cada proyecto.

La funcionalidad con la que más se identifica a *Redmine* es la forma en que se le da seguimiento a las tareas y actividades. Dentro de un proyecto, la asignación y visualización de las responsabilidades se logra a través de "peticiones". Estas "peticiones" son asignadas a un miembro del proyecto y en su definición contienen: fechas de inicio y fin de la tarea, tipo de petición, porcentaje de realización, prioridad y descripción. Los miembros del proyecto reciben notificaciones acerca de las actividades a las que han sido vinculados, tanto en su página personal de *Redmine* como por correo electrónico. Además, si así se desea, estas peticiones pueden observarse a través del calendario del proyecto o mediante un diagrama de *Gantt*.

A continuación se resumen algunas de las funcionalidades que posee *Redmine* y que lo convierten en una herramienta útil para la administración de proyectos [4]:

- Integración con repositorios: permite establecer un canal de comunicación directo con un repositorio para el control de versiones de cada proyecto. Entre los tipos de repositorios compatibles están: *Subversion*, *Git*, *CVS*.
- Disposición de una página personal: construye una página específica para cada usuario donde se le muestra la información sobre los proyectos en los que participa. Se muestran las tareas que ha asignado, cuáles le han sido asignadas y la actividad reciente del proyecto.
- Utilización de filtros: permite realizar consultas sobre las peticiones existentes, tanto por campos predeterminados como por aquellos creados por el usuario.

- Administración de archivos y documentos: se cuenta con una sección donde se pueden adjuntar los documentos relacionados al proyecto.
- Extensión de sus características: contiene en su sitio web una larga lista de *plug-ins* para ampliar las funcionalidades de la herramienta. Las extensiones van desde la generación de gráficos, hasta crear salas de *chat*. De todas ellas se ofrece una descripción breve, las versiones de *Redmine* para las que es compatible y algunos detalles sobre su instalación.

2.2 Subversion

Subversion es una herramienta de software libre utilizada para implementar un sistema de control de versiones. A través de ella es posible realizar de forma automática tareas como: guardar, descargar, y mezclar las distintas versiones que a través del desarrollo de un proyecto, se generan de un mismo archivo. *Subversion* se distribuye bajo la licencia APACHE/BSD (*Berkeley Software Distribution*), lo cual permite la inclusión de su código tanto en software libre como comercial.

La primera versión de la herramienta fue lanzada en octubre del año 2000 y desde octubre del presente año se encuentra disponible para la descarga gratuita la versión 1.7.1, último release estable. Fue programada enteramente en C y es multiplataforma, compatible con una gran cantidad de sistemas operativos: *Windows / MacOSX / AIX / Centos Linux / Debian Linux / Fedora Linux / FreeBSD / HP-UX / NetBSD / OpenBSD / Red Hat Linux / Solaris / SUSE Linux / Ubuntu Linux*) [5].

Para la administración de los archivos versionados, *Subversion* maneja los conceptos de "repositorio" y "copia de trabajo". Por repositorio se entiende aquel espacio, común y compartido por el equipo desarrollador, en el que se conservan todas y cada una de las versiones de los archivos producidos durante el ciclo de vida del proyecto, mientras que la copia de trabajo es aquel espacio perteneciente al entorno local de cada desarrollador, sobre el cual posee dominio total y que necesita periódicamente sincronizarse con el repositorio. Esta sincronización se traduce en las tres operaciones básicas de la herramienta:

- Importar (*Checkout*): a través de este comando se descarga del repositorio una copia exacta de sus contenidos, tanto de los archivos como de su estructura. Es posible descargar la última versión o versiones más antiguas.
- Actualizar (*Update*): esta operación permite descargar únicamente aquellas modificaciones realizadas al repositorio, ocurridas desde la última sincronización con la copia de trabajo.
- Confirmar (*Commit*): esta operación ocurre en vía inversa a las dos anteriores, ya que permite actualizar el contenido del repositorio con los cambios realizados en la copia de trabajo. Esta operación se ejecutará correctamente si no hay conflicto con alguno de los archivos del repositorio, es decir, no habrá sincronización si alguien más, ha modificado el mismo elemento en forma paralela.

Para una mejor organización de las versiones, *Subversion* permite la creación de *Tags* y *Branches* dentro del repositorio. La creación de un *Tag*, técnicamente congela una versión, alejándola del "tronco" (*Trunk*) de desarrollo principal del proyecto. Por otro lado, la creación de *Branches* permite la evolución paralela de varias versiones, por ejemplo si se está trabajando al mismo tiempo en varias maneras de solucionar un mismo problema. *Subversion* le permite a los desarrolladores crear *Branches* y así tener entornos disjuntos para el desarrollo paralelo. Cuando estas versiones necesiten fusionarse, igualmente *Subversion* provee asistencia para su integración [5].

A continuación se listan una serie de características que hacen de *Subversion* elegible y ventajoso en comparación con otros sistemas y estrategias no automatizadas para el manejo de versiones [6]:

- Los directorios también se encuentran versionados, no sólo los archivos.
- El estado general del repositorio posee asignado un único número de versión, y no cada archivo.
- Permite adjuntar metadatos a los archivos o directorios versionados.
- Las operaciones sobre el repositorio son atómicas, lo que significa que ninguna tiene efecto parcialmente. Ej: un *commit* concluye hasta que todo él haya ocurrido exitosamente.
- Posee una bitácora en la que se registra un mensaje por cada *commit* y no por todos los archivos afectados por el *commit*, evitando así la redundancia.
- Posee *merge tracking* el cual es una asistencia automática que maneja el flujo en los cambios de líneas de código, lo cual es muy útil para la resolución de conflictos al mezclar archivos.
- Permite el bloqueo selectivo de archivos.
- Permite conocer los cambios realizados en el repositorio, cuándo, y por quién.
- Le permite al usuario crear, copiar y borrar las carpetas y archivos en la copia de trabajo (o del repositorio) con la misma flexibilidad con la que se ejecutan estas acciones en el ambiente del sistema operativo.

En la sección a continuación, se detalla sobre el procedimiento de pruebas definido para la presente investigación.

3. METODOLOGÍA

El objetivo principal de esta investigación es la construcción de una plataforma que apoye y sistematice las actividades de verificación y validación de software. Para ello se decidió aprovechar las conocidas funcionalidades de herramientas como *Redmine* y *Subversion* e integrarlas en una única plataforma.

El "híbrido" de estas dos herramientas debiera poder participar en un ambiente de pruebas, donde un equipo de evaluadores realiza pruebas sobre los entregables de otro equipo de trabajo. Así mismo, la herramienta debería garantizar la total separación entre la información a la que tiene acceso el equipo revisado y la que ve el equipo revisor. Para satisfacer estos objetivos, se inició la configuración de *Redmine* con la definición de una estructura de proyectos y sub-proyectos, y la especificación de los permisos y privilegios para cada rol de usuario.

Conscientes de lo subjetivo que puede ser el registro de las no conformidades durante la ejecución de las pruebas de software, se necesita de alguna manera estandarizar la forma en la que la plataforma recopila la información concerniente a los resultados de las pruebas. Para esto, se tomó como modelo el estándar de la IEEE para Revisiones y Auditorías del Software (IEEEStd. 1012-1987) y se agregaron nuevos campos y nuevos tipos de peticiones en *Redmine*.

Otro objetivo de la presente investigación es proveerles a los equipos de trabajo acceso a un repositorio de control de versiones para los entregables de las pruebas. Para ello se integrará *Redmine* con *Subversion*, de tal forma que cada equipo pueda no solo versionar sus documentos sino también acceder a los entregables a evaluar, sin abandonar el contexto de ejecución. Dicha integración debe conferirle total independencia a la plataforma, de manera que no sea necesaria la utilización de otras herramientas, para la creación o para la alteración del estado de los repositorios. Para ello se investigó y probó varios *plug-ins* de *Redmine.*

Finalmente, se puso a prueba la plataforma desarrollada, en un ambiente controlado, para depurar la herramienta y para identificar aciertos y desaciertos en la creación del procedimiento de pruebas de pruebas "ideal", luego de enfrentarlo con actores reales, situaciones y problemas "reales". Esto se logró a través de la interacción con la docencia, incorporando el proceso de validación de requerimientos funcionales dentro del curso de pregrado Ingeniería de Software I y su correspondiente laboratorio de la Escuela de Ciencias de la Computación e Informática de la UCR.

En el siguiente apartado se describe detalladamente el proceso a través del cual se configuró *Redmine* para convertirla en una herramienta apta para las actividades de validación y verificación de software.

4. CONFIGURACIÓN DE REDMINE

Al comenzar la presente investigación, se contaba con la versión 1.1.0 de *Redmine* ya instalada. Se dispuso entonces a configurar la plataforma y ajustarla al proceso de pruebas. Primero fue necesario personalizar la herramienta, creando no sólo la estructura de proyectos, sino también nuevos roles, campos y peticiones. El paso siguiente fue, hacer efectiva la integración entre *Redmine* y *Subversion*, creando repositorios y accediendo a ellos desde la plataforma. Finalmente debía comprobarse la efectividad de los enlaces *Redmine* para direccionar revisiones y archivos específicos en un repositorio asociado a algún proyecto.

4.1 Estructura de proyectos

El proceso de personalización de *Redmine*, comenzó con la creación de la estructura de proyectos necesaria para que la plataforma resultara apta tanto para el desarrollo, como para la administración de las pruebas al software desarrollado.

La estructura se planeó en forma jerárquica: un proyecto "padre" y dentro de él tres sub-proyectos "hijos". Estos sub-proyectos servirían de apoyo para el desarrollo de software de cada equipo. A su vez, por cada proyecto "hijo" se creó un "sub-proyecto" dedicado a pruebas. En la Figura #1 se puede observar el diagrama con la estructura de proyectos definitiva.

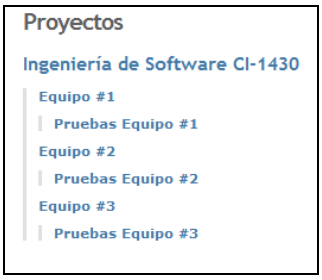

Figura 1. Estructura de proyectos definida para la plataforma.

Seguidamente se procedió a configurar cada proyecto de acuerdo a su finalidad. Para cada tipo de proyecto, se definieron campos específicos, tipos peticiones y roles de usuario.

4.2 Personalización de proyectos

A los proyectos de desarrollo se les agregaron los siguientes campos personalizados:

- Objetivo: de tipo texto largo, en él se especifica el objetivo del proyecto.
- Alcance: de tipo texto largo, en él se define el alcance del proyecto.
- Iteración: de tipo numérico, en él se especifica la iteración del ciclo de vida en la que se encuentra el proyecto.
- Versión: de tipo versión, en él se indica la versión del archivo o documento.
- Tipo de entregable: de tipo lista, de él se elige el tipo de entregable sobre el que se está trabajando. Los entregables pueden ser: conceptualización del proyecto, plan del proyecto, cronograma, especificación de requisitos, casos de uso, casos de prueba, prototipo, modelos de diseño, código, manual de usuario y manual de instalación.

En la Figura #2 se observa el módulo de *Redmine* para la generación de nuevos campos personalizados, entre ellos peticiones, proyectos y usuarios.

Figura 2. Módulo para la creación de nuevos campos.

Para los proyectos de prueba se agregaron los siguientes campos:

- Objetivo: de tipo texto largo, en él se especifica el objetivo del proyecto.
- Alcance: de tipo texto largo, en él se define el alcance del proyecto.
- Error de documentación: de tipo lista, en él se elige el tipo de error de documentación identificado. Estos errores pueden ser: de correspondencia con otra documentación, de formato, de idioma, de ortografía, de redacción, técnico y otros.
- Error de aplicación: de tipo lista, en él se elige el tipo de error de aplicación encontrado. Este error puede ser de: correspondencia con lo documentado, de interfaz, de excepciones, de funcionalidad, de idioma, de opciones que no funcionan, de ortografía, de redacción y de validación),
- Tipo de entregable: de tipo lista, en él se elige el tipo de entregable sobre el que se está trabajando. Los entregables pueden ser: conceptualización del proyecto, plan del proyecto, cronograma, especificación de requisitos, casos de uso, casos de prueba, prototipo, modelos de diseño, código, manual de usuario y manual de instalación.

También fue necesario definir en los sub-proyectos de prueba una serie de peticiones nuevas. Específicamente:

- Liberar entregable: petición utilizada para informarle al Coordinador sobre la disponibilidad de un entregable para su revisión.
- Asignar entregable: petición utilizada para asignarle el entregable a un Revisor.
- No conformidad documentación: petición utilizada a manera de registro, sobre el hallazgo de un error de documentación.
- No conformidad aplicación: petición utilizada a manera de registro, sobre el hallazgo de un error de aplicación.
- Liberar resultado: petición mediante la cual se informa sobre la conclusión y resultados del proceso de pruebas.

4.3 Perfiles

Según la definición de los actores del proceso de pruebas, sus capacidades y limitaciones, así fueron definidos los perfiles para los usuarios de *Redmine:*

- Coordinador: aquel que dirige ambos procesos (desarrollo y pruebas), está al tanto de todas las notificaciones del sistema. Posee todos los permisos.
- Jefe de proyecto: persona de mayor responsabilidad dentro del grupo desarrollador. Dentro de sus labores más importantes están: asignar tareas a los desarrolladores y vigilar por el cumplimiento de la ruta crítica del proyecto. Posee todos los permisos a excepción de aquellos de carácter administrativo: administrar el repositorio, administrar categorías de peticiones, borrar peticiones, mover peticiones, y crear proyecto, modificar proyecto, seleccionar módulos, crear sub-proyectos y administrar versiones.
- Desarrollador: integrantes del grupo desarrollador. Sus responsabilidades son más que variadas pues se encargan de diseñar, documentar, implementar y probar, a lo largo del ciclo de vida del proyecto de software. Posee habilitados todos los permisos menos aquellos relacionados a proyectos (crear proyecto, modificar proyecto, seleccionar módulos, crear sub-proyectos y administrar versiones), administrar categorías de peticiones, administrar relación con otras peticiones, modificar notas, mover peticiones, borrar peticiones, administrar consultas públicas, grabar consultas, borrar seguidores, administrar noticias y administrar repositorio.
- Revisor: miembros de otro equipo desarrollador, elegidos por el Coordinador del proceso, para que funjan como revisores del material del equipo revisado. El rol revisor puede ver peticiones, añadir peticiones, modificar peticiones, administrar relación con otras peticiones, gestionar sub-tareas, borrar peticiones, hojear repositorio y ver los cambios del repositorio.
- Revisado: son los miembros dueños de su sub-proyecto de pruebas. Los usuarios con este perfil, pueden ver peticiones, añadir peticiones, modificar peticiones, hojear repositorio, ver cambios en repositorio, acceso de escritura al repositorio y actualizar el repositorio.

En las siguientes secciones se explica cómo fueron cubiertas las necesidades del proceso de pruebas, mediante la instalación de extensiones y el aprovechamiento de las características propias de *Redmine*.

4.4 Integración con *Subversion*

Una de las propiedades de *Redmine*, fundamental para lograr la consecución del objetivo principal de esta investigación, es su capacidad de integración con repositorios de versiones, específicamente *Subversion*. La herramienta cuenta con un módulo llamado "Repositorio" el cual debe ser habilitado primero y posteriormente configurado. Dicha configuración consiste en:

1. En la pantalla de "Configuración de Proyecto" habilitar el módulo "Repositorio", seleccionando la caja de chequeo correspondiente.

2. En el módulo "Repositorio" elegir el tipo de repositorio de versiones con el que se asociará (*Git*, *Mercury*, *Subversion*, entre otros).
3. Especificar el URL del repositorio.
4. En caso de haber definido usuario y contraseña para el repositorio, se deben ingresar los mismos en los campos correspondientes.

Para este caso en particular debió realizarse una configuración adicional: la habilitación del acceso *WebDAV* (extensión del protocolo HTTP), para poder mover, copiar y modificar el contenido de los repositorios, ya que fueron creados en un servidor *Apache* de acceso únicamente por la red local de la institución.

Una vez realizada la configuración del módulo "Repositorio", se comprobó que a través de *Redmine* era posible visualizar el historial de revisiones y descargar los archivos del repositorio *Subversion* (Figura #3), sin embargo, no se permitía modificación alguna sobre el mismo. En otras palabras, las operaciones disponibles eran solamente de lectura y parte de los objetivos de la investigación era lograr el manejo total de los procesos de desarrollo y de pruebas desde *Redmine*, sin necesitar de una segunda herramienta para realizar los *commits* hacia el repositorio. Por esta razón se procedió a buscar, instalar y configurar una extensión (*plug-in)* que satisficiera este requerimiento.

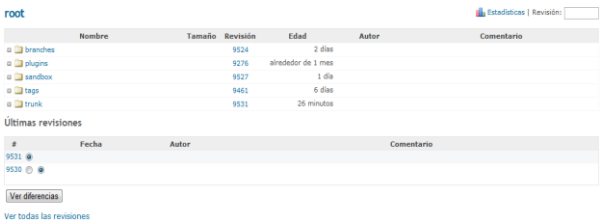

Figura 3. Módulo "Repositorio" de *Redmine*.

Se instaló el *plug-in* "SCM *extensions"*, con el cual se habilitan las operaciones de escritura al repositorio desde *Redmine*. El *plug-in* hace disponibles tres nuevas operaciones en el módulo "Repositorio", a saber: subir archivo, crear carpeta y borrar carpeta. Además, se agrega al módulo de configuración de roles (Figura #4), un nuevo permiso sobre actualización del repositorio, es decir, permite definir qué usuarios pueden modificar el repositorio y cuáles únicamente consultarlo.

Como detalle importante debe aclararse que si durante la configuración del repositorio se especificó un usuario y una contraseña, los cambios realizados al estado del repositorio quedarán a nombre de dicho usuario, por lo tanto, para que la autoría de estas modificaciones vaya acorde con el usuario identificado en la plataforma, los campos de usuario y contraseña deben dejarse en blanco.

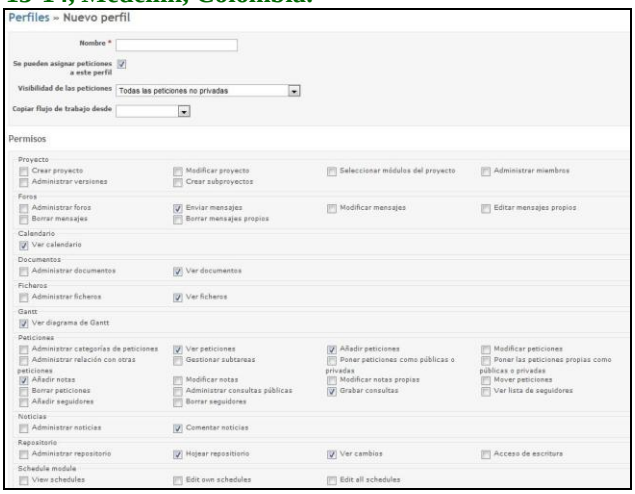

Figura 4. Pantalla para definir privilegios.

4.5 Creación automática de repositorios desde *Redmine*

Una vez lograda la actualización de los repositorios desde la plataforma, se consideró la posibilidad de que éstos fuesen creados también desde *Redmine*, logrando así una independencia total al no requerir de otra herramienta o de otra persona para crear los repositorios en el servidor. Se descubrió que el *plug-in* "SCM *Creator*" permitía dicha funcionalidad. Luego de su instalación, debe configurarse la ruta del directorio donde serán creados y otorgarle permisos de lectura, ejecución y escritura al usuario dueño de dicho directorio (código 0755). Cuando el *plug-in* es instalado correctamente, se puede ver en el módulo "Repositorio", junto al campo para digitar el URL, un botón (Figura #5) a través del cual se crea el repositorio automáticamente. [8].

Figura 5. Botón para crear automáticamente el repositorio.

4.6 Links entre proyectos

Para concluir la configuración de la plataforma, es necesario comprobar la factibilidad de utilizar los enlaces de *Redmine* para direccionar objetos en el repositorio de otro proyecto, es decir, de un proyecto distinto a aquel desde el cual se realizaba la petición. Estos *links* le permitirían al integrante del "grupo revisor" acceder directamente al archivo en el repositorio *Subversion* del "grupo revisado", sin tener que acceder primero a un proyecto ajeno, ni desperdiciar tiempo buscando el archivo a evaluar entre las carpetas del repositorio.

Sin embargo se descubrió que ninguno de los enlaces *Redmine*, de la versión instalada (1.1.0), satisfacía la funcionalidad

anteriormente descrita. Estos "*links* entre proyectos" fueron agregados a la herramienta a partir de la versión 1.2.0, bajo el nombre de "*cross-project links*". Por lo tanto, se procedió a actualizar la plataforma a la versión 1.2.1 de *Redmine.*

Luego de la actualización del *Redmine* se comprobó la efectividad de los links entre proyectos, los cuales deben seguir el siguiente formato: idProyecto:source:rutaAlArchivo, "idProyecto" corresponde al identificador del proyecto donde se encuentra el repositorio y "rutaAlArchivo" se refiere a la ubicación específica del archivo que se desea referenciar. Estos enlaces pueden utilizarse para direccionar tanto a archivos específicos, como a toda una revisión del repositorio.

5. EJECUCIÓN DEL PROCEDIMIENTO

Una vez finalizada la configuración de la herramienta para soportar el procedimiento, se puso a prueba utilizando el curso de Ingeniería de Software y su laboratorio. La experiencia se realizó durante el desarrollo del proyecto práctico del curso.

Se intentó simular un ambiente real de trabajo, donde los estudiantes pudieran aplicar revisiones técnicas entre grupos revisores y grupos revisados, permitiéndole a cada uno de los grupos (cuatro en total) evaluar la calidad del software producido por otro grupo de compañeros.

Se dedicaron una serie de laboratorios para capacitar en la utilización de la plataforma e instruir a los estudiantes sobre sus beneficios. Se les facilitó una lista clasificada de errores para que en el momento de ingresar a la plataforma, una petición del tipo "No Conformidad", pudieran clasificar y registrar el tipo de error encontrado. También se les insistió en describir en forma clara el error y de ser necesario adjuntar una imagen del mismo. Se les solicitó que no crearan ninguna clase de jerarquía en el momento de crear peticiones del tipo "No Conformidad", sino que todas debían ser hijas (sub-tareas) de la petición original de tipo "Asignar Entregable".

6. PROBLEMAS ENCONTRADOS

El primer problema experimentado fue el hecho de que la versión instalada de *Redmine* no disponía de la funcionalidad conocida como "*cross-project links*". Estos enlaces representaban una mejora a los enlaces que ya proveía *Redmine* y permiten referenciar archivos en un repositorio, revisiones enteras o incluso documentos adjuntos pertenecientes a un proyecto distinto. Sin embargo, la actualización mencionada en la sección anterior, requirió instalar también la versión 2.3.11 de *Ruby on Rails* lo cual generó una gran inestabilidad en el *Mongrel* (servidor web que viene con *Redmine)* y cierta incompatibilidad con los *plug-ins* instalados. Dicha inestabilidad provocaba que el tránsito entre una pantalla y otra de la interfaz web se viera interrumpido y muy frecuentemente concluyera con una pantalla en blanco. Incluso si se "refrescaba" la página, la acción ejecutada antes del bloqueo, se repetía, produciendo la duplicación de peticiones o de archivos "subidos" al repositorio. Para solucionar esto, debió utilizarse el *Webrick* (servidor Web que viene con *Ruby)* como servidor alterno, el cual brindó mayor estabilidad a la plataforma.

La mayor dificultad en términos de configuración a la que hubo que hacer frente, fue la política que posee *Redmine*, de "compartirlo todo o no compartir nada". Esto quedó en evidencia al probar el funcionamiento de los *links* entre proyectos (*cross-project links*), pues los enlaces resultaron funcionales sí y sólo sí, el usuario que los definía era miembro de ambos proyectos. Esta limitación afectó la definición original de los roles en el proceso de pruebas, pues se deseaba una total separación entre los equipos. La solución óptima fue que los revisores no tuvieran acceso más que a la descarga directa del entregable en el repositorio del equipo revisado, sin embargo, para utilizar los links debieron hacerse también miembros del proyecto de pruebas. La solución fue crear un perfil bastante limitado para estos revisores, de manera que sólo tuvieran acceso de lectura al repositorio y acceso denegado a todos los demás módulos del proyecto revisado.

Si bien los estudiantes recibieron toda la inducción necesaria para la utilización de la plataforma, el ambiente del curso no era el óptimo para que se alcanzaran los mejores resultados. La inexperiencia de los participantes en cuanto al desarrollo de pruebas de software y la inclinación por proveer sus propias soluciones cuando se encontraban frente a un problema, atentó contra la uniformidad de los resultados. Para solucionar esto se elaboró un manual de usuario y se realizó una nueva sesión de capacitación sobre el uso de la herramienta, donde se atendieron las dudas de los estudiantes y se explicó paso a paso el procedimiento de pruebas.

Además por la irregular asistencia de los estudiantes a los laboratorios, quedó evidenciada en la definición de roles, una fuerte dependencia sobre el líder de proyecto, ya que sólo él podía modificar o borrar las peticiones realizadas, lo cual restaba fluidez al trabajo de los desarrolladores. La solución fue flexibilizar la definición de los roles de usuario, otorgando a los desarrolladores mayores permisos sobre las peticiones.

7. CONCLUSIONES Y TRABAJO FUTURO

Como herramienta para la administración de proyectos, *Redmine* satisface todas las necesidades en cuanto a seguimiento de tareas y comunicación entre los participantes. Esto sumado a su integración con *Subversion* para administrar repositorios de versiones, terminan por convertirla en una plataforma más que funcional para el desarrollo de proyectos de software.

La gran flexibilidad y facilidad para la personalización que posee *Redmine*, representaron características claves durante la investigación, ya que extienden el campo de acción de la herramienta, permitiéndole ajustarse a proyectos de la más diversa naturaleza. La posibilidad de definir tipos concretos de peticiones y agregar nuevos campos a los formularios, le

conceden al usuario la capacidad de utilizar *Redmine* en proyectos de formato no tradicional.

Incluso cuando parece que la herramienta no puede satisfacer necesidades específicas, el sitio oficial de *Redmine* posee una gran cantidad de extensiones (*plug-ins)* que le ayudan en gran manera al usuario a solventar sus problemas. Esta investigación no hubiera podido llegar a buen término sin la posibilidad de instalar los *plug-ins* (SCM *extensions* y SCM *creator*) que enriquecieron de gran forma las operaciones sobre los repositorios de *Subversion*. Crear repositorios y realizar operaciones de escritura sobre ellos sin tener que abrir otra aplicación, le confirieron a la plataforma cierta autonomía que en un inicio no se creía posible.

Los laboratorios con los estudiantes representaron una gran fuente de retroalimentación sobre el procedimiento de pruebas definido. También, la utilización de la plataforma por usuarios ajenos al desarrollo de la misma, resultó en una variedad de comentarios y opiniones sobre su facilidad de uso, las ventajas y desventajas de su introducción en las revisiones técnicas formales y por supuesto el señalamiento de carencias o errores en el diseño de los formularios desde el punto de vista de usuario final.

Entre las dificultades encontradas que atentaron contra el rendimiento de la plataforma y su deseada configuración estuvieron: el comportamiento inconstante de los enlaces entre proyectos, la inestabilidad de la plataforma posterior a su actualización y la escasa capacitación que recibieron los estudiantes sobre la utilización de la plataforma, especialmente el modo de registrar las no conformidades.

Aún así, por encima de todas estas dificultades se concluye que sí es posible configurar *Redmine* para su utilización como plataforma de un proceso de pruebas de software bajo una modalidad de equipos. Su completa integración con *Subversion*, permite el "intercambio" de entregables para su respectiva evaluación y su naturaleza web, permite el registro, envío y acceso de los resultados de las pruebas, rápida y fácilmente.

Se espera que el éxito de la presente investigación no culmine con los resultados ya obtenidos, sino que impulse otros trabajos en los que se experimente la integración de Redmine con otros repositorios como Git, o Mercurial.

Actualmente se está trabajando en la inclusión de nuevos campos para la obtención de métricas de pruebas y la generación de reportes de incidencias.

Posterior a esta experiencia, se concluye que en proyectos sucesivos sobre integración de herramientas, es vital que desde un inicio se elijan aquellas que demuestren una complementariedad natural, ya sea por la existencia de módulos dedicados a compartir su información a través de servicios web o la exportación de datos. Esto permitiría que la integración suceda de manera rápida y transparente. Otra característica importante es la afinidad temática de las herramientas.

Finalmente como proyecto a futuro siempre en el ámbito de las pruebas de software, se considera la integración de NUnit y Selenium para una agilización de las actividades de validación y verificación en aplicaciones web.

8. REFERENCIAS BIBLIOGRÁFICAS

[1] Cámara Costarricense de Tecnología de Información y Comunicación. Sobre nosotros. 2010. Disponible en: http://www.camtic.org/ES/camtic/sobre_nosotros/. Consultado en 15 de marzo del 2011.

[2] Consejo Nacional para la Calidad. Calidad Costa Rica. 2006. Disponible en: http://www.calidadcostarica.go.cr/Eventos.html. Consultado el 18 de marzo del 2011.

[3] Desconocido. Sitio oficial de *Redmine*. 2011. Disponible en : http://www.redmine.org/. Consultado el 15 de marzo del 2011.

[4] Centro de excelencia de software libre de Castilla La Mancha. Análisis de aplicación: *Redmine*. 2010. Disponible en: http://www.bilib.es/recursos/analisis-de-aplicaciones/analisis/doc/analisis-de-aplicacion-redmine. Consultado el 20 de marzo del 2011.

[5] Desconocido. Sitio oficial de *Subversion*. 2011. Disponible en: http://subversion.apache.org/. Consultado el 4 de abril del 2011.

[6] Gómez Eyles, Sebastián. Buenas prácticas de gestión de versiones con *Subversion*. 2010. Disponible en: http://blogs.tecsisa.com/articulos-tecnicos/buenas-practicas-de-gestion-de-versiones-con-subversion/. Consultado el 5 de abril del 2011.

[7] Collins-Sussman Ben, W. Fitzpatrick Brian, Pilato Michael. Control de versiones con *Subversion*. 2010. Disponible en: http://svnbook.spears.at/nightly/es/svn-book.html. Consultado el 5 de abril del 2011.

[8] Lesyuk, Andriy. *Redmine SCM Creator*. 2011. Disponible en: http://projects.andriylesyuk.com/projects/scm-creator. Consultado el 28 de abril del 2011.