Capítulo 11. Conclusiones y trabajo futuro

En esta tesis ha realizado un entorno de desarrollo Web que proporciona herramientas para la mejora de la calidad del código de los desarrolladores.

Para conseguir este objetivo se han analizado distintos sistemas tanto académicos como comerciales (Capítulo 3) revisando sus aportaciones a la mejora de la calidad del código. En dicha revisión, hay un gran número de aplicaciones académicas que se centran solamente en la descripción de los conceptos fundamentales de programación y además en el caso de entornos, están pensados también para un curso inicial. En la parte comercial las herramientas disponen de un gran número de opciones que les proporcionan gran potencia; pero está supeditada a la experiencia de los desarrolladores. En el caso concreto del tratamiento de errores, hay disponibles gran cantidad de herramientas que realizan análisis estático y que proporcionan una información mucho más profunda que el compilador sobre los errores del código fuente; sin embargo, la información que proporcionan prácticamente se limita al mensaje de error, si el desarrollador es la primera vez que trata este error tendrá difícil analizar sus causas y por tanto, corregir este error.

Además, se han establecido los requisitos generales que debería cumplir un sistema para la mejora de la calidad del código (Capítulo 4) y se han diseñado una serie de prototipos para refinar estos requisitos de forma iterativa (Capítulo 5 al Capítulo 9). Obteniendo un sistema que:

- Está centrado en el entorno de desarrollo que permite trabajar de forma colaborativa con proyectos de tamaño medio y grande.
- Se basa en técnicas de procesadores de lenguaje para realizar la captura y el análisis de errores sobre el código fuente de los proyectos que se están desarrollando.
- Proporciona una interfaz Web que permite la fácil integración con otros sistemas Web como sistemas de aprendizaje en el campo académico o sistemas Web de gestión de proyectos software; subsanando carencias actuales de estos sistemas: permitiendo el desarrollo de proyectos on-line; y un seguimiento continuo y personalizado de los desarrolladores mediante los datos de los errores capturados y analizados automáticamente.

Por último, se ha realizado un estudio empírico sobre la frecuencia de errores de los estudiantes de Ingeniería Informática en los distintos cursos de la titulación.

11.1 Sistema diseñado: SICODE

A continuación describimos las características más destacadas del sistema SICODE.

11.1.1 Entorno Web de desarrollo que integra los distintos subsistemas

El entorno de desarrollo (IDEWeb) es el centro del sistema. El entorno integra las distintas herramientas necesarias para realizar el desarrollo del software. Dispone de una arquitectura Web con lo que el desarrollador no necesita nada más que un navegador Web para poder programar, todo el procesamiento necesario y el almacenamiento de los archivos del proyecto se realiza en el servidor.

El entorno permite escribir y modificar código fuente, realizar la compilación y el análisis directamente. Muestra los errores resultantes de la compilación y facilita la corrección de los mismos al conectar directamente con la base de conocimientos. Además, recoge la información del Sistema de Análisis de Programas (PBA) y la muestra al usuario a través de avisos facilitando la repetición de errores a la hora de escribir el código fuente.

Para realizar la edición sobre el navegador Web se utiliza un campo de texto adicional. Para superar la limitación que supone esto, se ha desarrollado un editor que se implementa como un applet con características equiparables a un editor de un IDE. De esta forma obtenemos facilidades en la edición pero pudiendo usarlo desde el navegador Web.

11.1.2 Sistema de colaboración en el desarrollo de aplicaciones

El Sistema para la colaboración en el desarrollo de aplicaciones (COLLDEV) permite la comunicación y la colaboración entre varios desarrolladores y supervisores. Para ello permite la creación de grupos de trabajo que tienen asignados uno o más proyectos y la comunicación entre desarrolladores mediante mensajes asíncronos de distintos tipos.

Motiva a los desarrolladores a la toma conjunta de decisiones sobre el proyecto mediante sistemas de encuestas. Estas decisiones son tanto a nivel de código fuente, para corregir un error de programación; como a nivel de diseño para reparar errores conceptuales o hacer una refactorización que mejore el diseño para hacerlo más claro y mantenible.

11.1.3 Historia de trabajo de Proyectos compartidos

El sistema de colaboración también permite compartir los archivos del proyecto. Por un lado este sistema es imprescindible para construir un proyecto de mediana o gran entidad; además permite que todos los desarrolladores puedan leer el código fuente de cualquier clase del proyecto con lo que facilita la revisión de código y la corrección de errores de forma compartida.

A parte de compartir el proyecto, también se van guardando las sucesivas versiones de este para hacer posible el seguimiento del desarrollo del proyecto en etapas intermedias y no sólo en su etapa final; lo que permite explorar cuándo se realizaron determinadas modificaciones, cuándo se corrigieron determinados errores o cuándo se introdujeron errores nuevos.

11.1.4 Sistema de análisis de errores en programas

Sistema de análisis de errores de programas (PBA), se basa en varios procesadores de lenguaje que realizan un análisis estático sobre el código fuente de los proyectos en desarrollo. El sistema permite configurar de forma flexible la secuencia y las opciones de ejecución de estas herramientas mediante unidades de compilación escritas en XML y que se validan mediante XML-Schema.

El sistema crea una historia de compilación con el histórico de errores resultado de la captura de todos los análisis a lo largo del desarrollo de la aplicación. Esta historia de

compilación permite al sistema analizar no sólo la información de la compilación actual, sino también la información de anteriores compilaciones y de múltiples herramientas.

El sistema permite configurar los análisis para adaptarlos a lo que requiere cada usuario, descartando errores poco significativos o combinando los errores recogidos por varias herramientas. Además, permite realizar el análisis para un usuario individual o un grupo de usuarios. Y permite obtener información sobre errores frecuentes y la evolución de los errores en el desarrollo del proyecto. Por otra parte, presenta esta información al usuario de forma tanto numérica como gráfica. Esta información sirve además para seleccionar los avisos que le van a aparecer al usuario en la interfaz del entorno de desarrollo (IDEWeb).

11.1.5 Base de conocimientos colaborativa

Base de conocimientos que aporta información extra sobre los mensajes de error, a través de ejemplos y proporcionando información sobre las causas y posibles soluciones. Esto ayuda a los desarrolladores de dos formas: a entender y corregir el error y a aprender a como evitarlo para que no se vuelva a producir la próxima vez que escriban código.

Esta base de conocimientos está basada en un sistema Wiki que permite añadir contenido desde el propio navegador, sin herramientas especiales. Además, en el sistema los usuarios no crean las páginas desde cero sino que el propio sistema introduce la información básica y a partir de ella los usuarios introducen sus experiencias concretas.

La concepción del Wiki permite que todo el mundo introduzca contenido en el sistema y supone un cambio en la concepción respecto a muchos sistemas en los que únicamente un experto o un grupo de expertos reducido puede aportar conocimiento al sistema y todos los demás únicamente pueden consultar. En este nuevo planteamiento todos pueden aportar, con lo que la base de conocimientos se enriquece más rápidamente, y aunque pueda haber ciertos errores en el contenido estos pueden ser reparados rápidamente.

11.2 Clasificación de usuarios

Se ha llevado a cabo un estudio de los errores que suelen cometer los desarrolladores en el proceso de aprendizaje de la programación (Capítulo 10). El estudio ha sido amplio tanto por la cantidad de proyectos analizados (por ejemplo para segundo curso se han analizado 337 proyectos), como por el número de cursos (se han analizado proyectos de cuatro cursos más proyectos final de carrera); además no se han analizado clases aisladas o determinadas partes sino que se ha considerado el proyecto completo.

Se ha utilizado el sistema propuesto en esta tesis: SICODE a través del subsistema PBA para obtener las tablas de los diez o doce errores que aparecen más frecuentemente en cada uno de los cursos. Nos hemos centrado en los avisos que pueden incidir sobre la corrección del código fuente, su claridad, su facilidad de lectura y comprensión y la facilidad de mantenimiento.

Por último, se han caracterizado distintos tipos de errores dependiendo de los cursos en los que aparecen y su evolución: errores que aparecen en todos los cursos, errores que son exclusivos de un curso (en todos los cursos hay este tipo de errores) y errores que evolucionan con los cursos. Esto nos permitirá construir las bases para un modelo de usuario y que el sistema proporcione información adaptada dependiendo del tipo en el que se encuadre.

11.3 Principales aportaciones del trabajo

En esta tesis se ha conseguido implementar un sistema que cumple los requisitos establecidos en el Capítulo 4. De todas las características que se detallan en ese capítulo destacaremos cuatro.

11.3.1 Creación de un entorno de desarrollo integrado con una interfaz Web

Los sistemas que utilizan la Web para el aprendizaje de la programación raramente permiten al usuario interactuar con los programas, nuestro sistema no sólo permitirá interactuar al desarrollador para escribir y modificar código fuente sino que permitirá compilar y corregir errores.

Los entornos de desarrollo comerciales son locales, la única posibilidad de compartir información entre varios usuarios es la de conectarse a un repositorio donde se compartan los archivos del proyecto; pero los entornos no integran posibilidades de comunicación. El sistema SICODE permite compartir los archivos del proyecto, incluye un sistema de intercambio de mensajes y toma de decisiones, guarda un registro con todos los errores de compilación del sistema y además permite intercambiar conocimientos entre los desarrolladores mediante la base de conocimientos colaborativa.

Existen sistemas de gestión de proyectos por ejemplo SourceForge que están basados en la Web y a través de ella proporcionan múltiples servicios a los desarrolladores del proyecto; sin embargo no llegan tan lejos como SICODE, ya que la escritura de código siempre se realiza en un sistema local.

SICODE a través de su entorno Web permite una independencia total de la plataforma y evita la instalación y configuración del sistema integrando a la vez las herramientas esenciales para el desarrollo de la aplicación. Esta interfaz Web permite además al sistema integrarse con otros sistemas Web como sistemas de e-learning o de gestión de proyectos.

11.3.2 Diseño de una historia de compilación

Para asegurar la corrección de los programas en los procesos de desarrollo tradicionales se utilizan varias técnicas:

- Arreglar los errores que proporciona el compilador, muchas veces esto es insuficiente ya que la principal tarea del compilador es generar código objeto y no detectar posibles problemas en el código.
- Realizar un conjunto de pruebas lo más amplio posible y corregir los problemas encontrados, este proceso lleva bastante tiempo ya que tenemos que crear casos de prueba y ejecutarlos.
- Una técnica que siempre se recomienda es la inspección de código; pero de nuevo necesitamos bastante tiempo para llevarla a cabo.

El sistema SICODE permite compilar y hacer un análisis exhaustivo del código fuente del proyecto que está en desarrollo, para ello se basa en herramientas que realizan este análisis estático, se combinan sus resultados y se almacenan en un registro de errores que denominamos "Historia de compilación". De esta forma, el sistema no sólo proporciona información sobre la última compilación, sino que produce avisos en función de esta historia de compilación en la que se pueden buscar errores frecuentes y una evolución de los errores para proporcionar al desarrollador las herramientas para evitar nuevos errores cuando esté escribiendo más código.

11.3.3 Utilización del análisis activo de errores para obtener una mejora en el código fuente

Los entornos de desarrollo actuales no están pensados para que los desarrolladores mejoren su estilo de programación y la calidad del código que escriben. Sin embargo, constantemente están apareciendo nuevas versiones de herramientas, nuevas técnicas y nuevos métodos de trabajo que proporcionan ventajas sobre los anteriores y que es imprescindible no sólo conocer, sino aplicar de la forma más adecuada. Los desarrolladores tienen que establecer procesos de formación continua, que son paralelos al proceso de desarrollo, para permanecer al día.

Por otra parte, algunas aplicaciones tienen una serie de requisitos de aceptación en los que incluyen pruebas de aceptación para comprobar que las aplicaciones tienen la funcionalidad requerida; pero también para comprobar que el diseño y el código fuente siguen unas pautas previamente marcadas. Por otra parte, muchos equipos de desarrollo establecen una guía de estilo para la escritura de código donde se incluyen convenios y buenas prácticas en el diseño y la codificación de las aplicaciones. Sin embargo, el cumplimiento de estas normas es muchas veces difícil de llevar a cabo y de comprobar.

En nuestro sistema se propone integrar el desarrollo con el aprendizaje, basándonos en la comprobación automática del código que está escribiendo el desarrollador. La forma de realizar esto conlleva el trabajo conjunto de varios elementos del sistema:

- Análisis exhaustivo del código fuente mediante el subsistema PBA, permite detectar que el código fuente no cumple determinada norma de la guía de estilo, o no se está empleando correctamente una técnica recogida en el grupo de desarrollo o no se cumple un requisito recogido en las pruebas de aceptación.
- A partir de las comprobaciones del subsistema PBA se pueden generar mensajes de aviso y mostrárselos al desarrollador a través del entorno Web (IDEWeb). Estos avisos indicarán errores concretos en el código; pero también facetas más generales del código (estilo, nuevas técnicas) que debería mejorar el programador.
- Los mensajes de aviso aparecerán mientras el desarrollador está escribiendo código. El sistema es activo y no espera a la fase de compilación para notificar los problemas.
- A través de los avisos el usuario puede consultar la base de conocimientos colaborativa que recoge la experiencia de otros desarrolladores; a partir de esta experiencia el desarrollador puede "aprender" a hacer las cosas mejor.

De esta forma, el sistema va desde la notificación de un error concreto en el código fuente a la información más general sobre el estilo de programación y técnicas de desarrollo que podrán servir para que futuros desarrollos tengan una mayor calidad.

11.3.4 Diseño de un modelo que permite realimentarse con la experiencia de los desarrolladores

La base de conocimientos donde se guarda información sobre la experiencia de los desarrolladores para entender y solucionar errores, no sigue un modelo estático; sino que está diseñada para que los desarrolladores sigan introduciendo más información de forma colaborativa mientras trabajan en las aplicaciones. De esta forma la base de conocimientos cada vez se hace más completa y refinada gracias a las aportaciones de los usuarios.

11.3.5 Realización de un estudio de los errores de programación y su evolución a lo largo de los distintos cursos académicos

Como ya se ha comentado anteriormente en esta tesis se realiza un amplio estudio que ha permitido conocer los tipos de errores que aparecen en cada uno de los cuatro cursos más el proyecto fin de carrera de la titulación de Ingeniería en Informática. Pero sobre todo se ha estudiado la evolución de esos errores a lo largo de la carrera, obteniendo una relación entre los errores y la experiencia del estudiante. Hasta ahora los estudios publicados se centraban en una práctica concreta o, los más amplios, en una asignatura; pero no se había trabajado sobre varias asignaturas para estudiar la evolución de los usuarios.

11.4 Futuras líneas de investigación

El trabajo realizado en esta tesis abre nuevas líneas de investigación relacionadas con mejoras en el sistema o nuevas aplicaciones del mismo.

11.4.1 Reducción de la granularidad en la comprobación y generación de avisos de ayuda al desarrollador

Se pretende seguir trabajando en un sistema activo que permita ir guiando al usuario en la mejora del estilo y la calidad del código. Para ello pretendemos trabajar en dos líneas:

- Realizar las comprobaciones del código fuente y emitir los avisos derivados de estas en tiempo real.
- Añadir a los avisos un mecanismo que permita automatizar la corrección de errores.

Esto evitará que errores cometidos permanezcan mucho tiempo sin corregirse ya que el sistema podrá avisar y hacer propuestas de corrección para que el usuario las utilice inmediatamente.

11.4.2 Descentralización del sistema

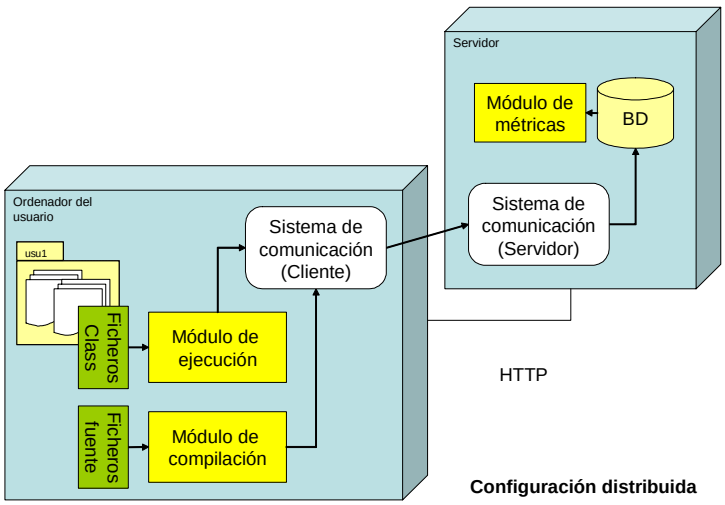

Figura 71. Arquitectura distribuida para el entorno de desarrollo y el sistema de análisis

La arquitectura planteada para el sistema ha sido una arquitectura Web centralizada en la que todo el procesamiento se realizaba en el servidor y los usuarios tienen una interfaz única a través del navegador. El planteamiento de esta línea de investigación consiste en

mantener la arquitectura Web; pero descentralizando el entorno de desarrollo y el sistema de análisis.

En este planteamiento el usuario utilizará un entorno de desarrollo instalado localmente; puede utilizar un entorno comercial en el que dispondrá de facilidades en la escritura del código. Los módulos de análisis se integrarán como plug-ins en el entorno para poder hacer su uso transparente al desarrollador. La base de datos y el módulo de generación de estadísticas permanecerán en un servidor centralizado y la comunicación se realizará mediante servicios Web sobre el protocolo HTTP.

Para realizar esto hay que abordar la creación de un vocabulario XML que permita expresar de forma genérica la información de análisis de errores y otro para poder expresar la información de las estadísticas en forma de avisos. De esta forma la información se transmitirá en dos direcciones: del cliente a la base de datos centralizada y del módulo de estadísticas en el servidor a cada cliente para presentar al desarrollador los avisos en su entorno.

11.4.3 Aplicación de técnicas de minería de datos para mejorar el análisis

El subsistema PBA ya dispone de ficheros de configuración que permiten realizar análisis filtrando determinados datos. Se podrían ampliar estos ficheros para utilizarlos en técnicas de minería de datos que se aplicarían sobre la historia de compilación; para obtener un mayor conocimiento sobre la calidad del código y dónde se debería mejorar.

11.4.4 Potenciación del análisis dinámico

En los prototipos implementados del sistema SICODE se realizaba fundamentalmente un análisis estático del código fuente para encontrar errores. Sin embargo, ya se dejaba abierta la posibilidad de ejecutar el código generado para realizar análisis dinámicos que complementen a los primeros (apartado 7.4.4).

Por tanto sería interesante, potenciar este módulo integrándolo con herramientas de pruebas como JUnit y que el sistema proporcionase ayuda al usuario para la creación de un conjunto de pruebas adecuado.

11.4.5 Potenciar el uso de la historia de trabajo

La historia de trabajo es una herramienta que puede proporcionar mucha información. Actualmente el sistema almacena la información de las versiones, proporciona una forma de navegación y permite realizar comparaciones. Sin embargo, este mecanismo tiene un gran potencial y permitiría hacer cosas como simulaciones de qué pasaría si en esta versión un fragmento se hubiese programado de otra manera.

11.4.6 Mejora de la adaptabilidad del sistema a los usuarios

El trabajo realizado en esta tesis pone las bases hacia la construcción de un modelo de usuario consistente. El sistema debería implementar este modelo de usuario para conseguir una adaptabilidad a cada usuario que interactúe con el sistema.

11.4.7 Aplicación del sistema para automatizar la comprobación de requisitos de aceptación

Actualmente los clientes de productos software no se conforman con que el producto cumpla una determinada funcionalidad, lo cual se puede comprobar con pruebas de aceptación. Además, el código fuente del producto debería cumplir unos niveles mínimos

de calidad que faciliten su modificación para la corrección de problemas que se encuentren en un futuro y para la incorporación de nuevas funcionalidades. Actualmente esta segunda comprobación se realiza normalmente mediante inspecciones de código manuales basadas en una guía de estilo, esto tiene dos problemas: es difícil la revisión de todo el código y se invierte mucho tiempo en realizar esta tarea.

El sistema SICODE ya realiza automáticamente muchas de las comprobaciones que puede contener una guía de estilo convencional, por otra parte habría que añadir una forma de añadir reglas de comprobación de otras características propias de cada producto.