UNIVERSIDAD
COMPLUTENSE
MADRID

Proyecto de Innovación

Convocatoria 2016/2017

Proyecto 198

Diseño de una metodología para la implantación de herramientas automáticas de corrección de estilo y buenas prácticas en asignaturas de programación.

Responsable del proyecto: Jesús Correas Fernández

Facultad de Informática

Departamento de Sistemas Informáticos y Computación

1. Objetivos propuestos en la presentación del proyecto

El objetivo principal de este proyecto de innovación consiste en estudiar la aplicación, adaptación e implantación de herramientas automáticas de análisis de la calidad de programas en el contexto de la enseñanza de la programación y desarrollo de software. Desde el punto de vista del alumnado, el uso de estas herramientas, además de favorecer buenos hábitos de programación, les introduce en el uso de herramientas cuya utilización es cada vez más habitual en proyectos de desarrollo de software en la industria y, en definitiva, les ayudará a realizar trabajos de mayor calidad. Desde el punto de vista del profesorado, la utilización de este tipo de herramientas fundamentalmente les ayudará en los procesos de corrección de ejercicios, prácticas y exámenes, haciendo que este proceso de corrección sea más ágil, y sobre todo más fiable. Su uso dejará también más tiempo para que el profesor se centre en aspectos más complejos de los programas que herramientas automáticas no pueden evaluar.

Las asignaturas susceptibles de ser consideradas para el uso e implantación de las citadas herramientas son fundamentalmente las siguientes: En los grados de la Facultad de Informática de Universidad Complutense de Madrid (UCM), *Tecnología de la Programación*, *Ingeniería del Software* (ambas asignaturas de segundo curso) y *Métodos Algorítmicos en Resolución de Problemas* (asignatura de 3º del grado en Ingeniería Informática) y sus diferentes variantes en el resto de grados. En los grados de la E.T.S. de Ingenieros en Informática de la Universidad Politécnica de Madrid, *Programación I*, *Progrmación II*, *Programación para Sistemas* y *Algoritmos y Estructuras de Datos*. Todas ellas utilizan el lenguaje de programación Java para la realización de ejercicios, prácticas, proyectos y exámenes prácticos.

Por otro lado, también se estudiará el uso e implantación de herramientas similares que traten el lenguaje C++, lo que permitiría que el proyecto abarque también las asignaturas *Fundamentos de la Programación* y *Estructuras de Datos y Algoritmos*, asignaturas de primer y segundo curso respectivamente comunes a todos los grados de la Facultad de Informática en UCM, y las asignaturas *Programación 1* y *Programación 2* del Grado en Estadística Aplicada de la Facultad de Estudios Estadísticos. De cara a abordar el objetivo fundamental presentado al inicio de esta sección identificamos los siguientes sub-objetivos:

- **Investigación y estudio de las herramientas de análisis de la calidad de programas.** Esto incluye, para cada una de ellas, su instalación en diferentes sistemas operativos, un aprendizaje profundo de su utilización desde el punto de vista del usuario, y la identificación y categorización de las distintas clases de propiedades que cada herramienta es capaz de inferir.
- **Selección de herramientas.** Selección de aquellas herramientas que se estimen más apropiadas, y para cada una de ellas, estudio detallado del tipo de configuraciones de uso que permite y resultados que ofrece.
- **Automatización de uso de las herramientas seleccionadas.** Se implementará un servidor web y/o un *plugin* para la plataforma Eclipse que permita un uso sencillo por parte de los alumnos, y también por parte de profesores no iniciados en el uso de estas herramientas.
- **Introducción a los alumnos en los conceptos de calidad de software y buenas prácticas de programación** y utilización de herramientas automáticas dentro del proceso general de desarrollo de software.
- **Evaluación del impacto de estas herramientas en la formación de los estudiantes** mediante la evaluación de las propiedades categorizadas antes y después de la introducción de estas herramientas en la formación de los alumnos.

2. Objetivos alcanzados

Se han alcanzado los siguientes objetivos:

- **Investigación y estudio de las herramientas de análisis de la calidad de programas.** Se han estudiado las herramientas de análisis de calidad de programas más utilizadas: `Findbugs` (`http://findbugs.sourceforge.net`), `PMD` (`https://pmd.github.io/`), `CheckStyle` (`http://checkstyle.sourceforge.net/`) o `Sonarqube` (`http://www.sonarqube.org/`). Se ha realizado una instalación de pruebas de las tres primeras y se han analizado las distintas clases de propiedades que cada herramienta es capaz de inferir.

- **Selección de herramientas.** Después del estudio realizado, se ha seleccionado la herramienta que se ha considerado más apropiada: PMD (`https://pmd.github.io/`). Esta herramienta contiene un conjunto de comprobaciones muy extenso: 286 reglas solo para el lenguaje Java agrupadas en 27 conjuntos de reglas (*rulesets*). En el apartado 5 se detallan los conjuntos de reglas seleccionados para que los utilicen los alumnos.

- **Automatización de uso de las herramientas seleccionadas.** Durante el desarrollo del proyecto se ha trabajado en la utilización de un servidor web y un *plugin* para el sistema Eclipse, desarrollado en el grupo de investigación al que pertenecen los profesores, que se puede conectar a un servidor de forma que los alumnos envíen sus programas para ser analizados.

  Sin embargo, después de estudiar detalladamente la última versión de la herramienta `PMD`, se ha optado por utilizar su propio *plugin* pues en el contexto de este proyecto ofrece mayor flexibilidad de uso que la proporcionada por el servidor centralizado.

- **Introducción a los alumnos en los conceptos de calidad de software y buenas prácticas de programación.** Se ha seleccionado un grupo de la asignatura *Tecnología de la Programación* de segundo curso (grupo B de los grados en Ingeniería Informática e Ingeniería de Computadores) de la Facultad de Informática y se ha impartido una clase extra de introducción a las herramientas de análisis de programas.

  La asignatura seleccionada utiliza Java como lenguaje de programación orientada a objetos. Por ello, durante el desarrollo del proyecto se ha decidido centrar los esfuerzos en este lenguaje de programación, dejando su aplicación a otros lenguajes como C++ para futuros cursos.

- **Evaluación del impacto de estas herramientas en la formación de los estudiantes.** Por falta de tiempo en la planificación docente del curso, no ha sido posible la evaluación del impacto de la herramienta antes y después de su utilización por parte de los alumnos. Sin embargo, sí se ha podido verificar a través de una encuesta realizada al grupo de clase seleccionado la buena acogida por parte de los alumnos de este tipo de herramientas de análisis de programas. En el apartado 5 de esta memoria se puede encontrar más información sobre los resultados del uso de estas herramientas en el aprendizaje.

En resumen, se puede concluir que se ha alcanzado el objetivo fundamental del proyecto: se ha demostrado la viabilidad de las herramientas de análisis de programas en las asignaturas de programación. Mediante la encuesta realizada a los alumnos se ha podido verificar que el uso de este tipo de herramientas ha permitido a los alumnos mejorar la calidad de los programas desarrollados durante el curso. Además, han mostrado interés en el uso de este tipo de herramientas, tanto en asignaturas de programación como en otras del plan de estudios y en su futura actividad laboral.

Aunque algunos de los objetivos planteados en la solicitud del proyecto no se han llevado a cabo completamente, creemos firmemente que el desarrollo del proyecto ha sido satisfactorio y ha proporcionado a los profesores participantes la experimentación con este tipo de herramientas en un grupo real de alumnos.

3. Metodología empleada en el proyecto

Se ha desarrollado el proyecto realizando las siguientes tareas:

- **Tarea 1.** Investigación y estudio de herramientas de análisis de software: Se ha llevado a cabo una investigación de las herramientas indicadas en la descripción del proyecto. En particular, se han instalado y estudiado las herramientas `Findbugs`, `PMD` y `CheckStyle`.

  Para cada una de ellas, se ha realizado una instalación preliminar en el sistema operativo Linux y se han identificado las distintas clases de características de los programas que cada herramienta es capaz de analizar en el contexto del lenguaje de programación Java.

- **Tarea 2.** Selección de herramientas y estudio técnico de uso avanzado: Para las herramientas seleccionadas se han estudiado aspectos más avanzados de configuración con el objetivo de determinar el grado de automatización que puede ser factible y apropiado (ver Tarea 4).

- **Tarea 3.** Pruebas de uso en corrección de prácticas de alumnos: Se ha realizado una fase de pruebas preliminar aprovechando una entrega de prácticas del segundo cuatrimestre de la asignatura de *Tecnología de la Programación* de segundo curso de los grados de Ingeniería Informática e Ingeniería de Computadores en UCM.

- **Tarea 4.** Formación introductoria sobre los conceptos de calidad de software y uso de estas herramientas a los alumnos de grupos y asignaturas seleccionados.

  - Se ha estudiado la instalación del sistema elegido para evaluar la manera en la que los alumnos pueden utilizar la herramienta de la forma más sencilla posible.
  - Se ha preparado un instalación especial del sistema Eclipse para que los alumnos puedan instalarlo en una sesión extra de laboratorio organizada con este fin. De esta forma se evitan las complicaciones derivadas de la instalación de *plugins* de Eclipse por parte de los alumnos.
  - Se ha impartido una clase especial a los alumnos a los que se ha enseñado a utilizar la herramienta.

- **Tarea 5.** Automatización e implementación: Se ha estudiado la instalación de la herramienta en dos contextos diferentes: un servidor para el análisis automatizado de prácticas de alumnos y un plugin para la plataforma Eclipse que permita un uso sencillo por parte de los alumnos.

- **Tarea 6.** Evaluación: Para poder evaluar el impacto de la utilización de estas herramientas, en esta tarea se van a utilizar las categorías identificadas en la tarea 1 para los trabajos entregados por alumnos. Por las características particulares en las que se ha desarrollado el curso 2016-2017, detalladas en el apartado 5, no se ha realizado una comparación entre las prácticas entregadas en el curso anterior y el curso actual. En su lugar, se ha realizado una encuesta a los alumnos sobre la aplicación de este tipo de herramientas en asignaturas de programación, y en particular su uso en las prácticas que los alumnos han realizado en el segundo cuatrimestre del curso 2016-2017.

- **Tarea 7.** Realización del informe final; interpretación de los resultados obtenidos. Se han recopilado he interpretado los resultados obtenidos en la encuesta realizada a los alumnos.

4. Recursos humanos

Todos los profesores integrantes del proyecto tienen amplia experiencia en la impartición de asignaturas relacionadas con la programación, y en particular con el lenguaje Java. La mayoría de los profesores son además desde hace años profesores responsables en las asignaturas en las que se ha previsto estudiar la implantación de las herramientas de análisis de la calidad de programas. Por otro lado, los principios técnicos y detalles de implementación de estas herramientas utilizan los fundamentos estudiados por los miembros del proyecto en sus actividades de investigación sobre testing formal y análisis estático.

El grupo de trabajo se ha compuesto de tal manera que todos los aspectos del proyecto queden cubiertos por los conocimientos de sus miembros. Más concretamente:

- **Elvira Albert Albiol**: Lleva 16 años impartiendo asignaturas relacionadas con la programación, y en particular con la programación concurrente. Elvira lidera el grupo de investigación COSTA (`http://costa.ls.fi.upm.es/web/`), al que pertenecen todos componentes del proyecto. Entre sus intereses de investigación se encuentran el análisis estático de programas concurrentes y distribuidos y las técnicas de testing formal de programas.

- **Purificación Arenas Sánchez**: Tiene una dilatada experiencia docente de 25 años. Es desde hace años profesora responsable de *Tecnología de la Programación*, asignatura de programación de segundo curso de los grados en Ingeniería Informática, Ingeniería del Software e Ingeniería de Computadores, impartida en el lenguaje Java. Su actividad investigadora ha estado centrada en el análisis estático y testing de programas concurrentes y distribuidos

- **Jesús Correas Fernández**: Tiene 15 años de experiencia docente en las Universidades Complutense y Politécnica de Madrid. Durante varios años ha impartido la asignatura *Tecnología de la programación*. En el curso 2016-2017 ha impartido las asignaturas *Bases de Datos*, *Administración de Bases de Datos*, y el segundo cuatrimestre casi completo de *Tecnología de la programación*. Su investigación está centrada en el análisis estático de programas orientados a objetos concurrentes y distribuidos.

- **Samir Genaim**: Tiene una amplia experiencia impartiendo este tipo de asignaturas en diferentes universidades y en la actualidad imparte también *Tecnología de la programación* en el grupo en inglés y en el Grado de Desarrollo de Videojuegos. Parte de su investigación en los últimos años está centrada en el desarrollo de un entorno web colaborativo y genérico para el uso de herramientas de análisis de programas.

- **Miguel Gómez-Zamalloa Gil**: Lleva 10 años impartiendo asignaturas relacionadas con la programación. Su investigación se ha centrado en el testing formal basado en la ejecución simbólica.

- **Guillermo Román Díez**: Profesor Ayudante Doctor en la Universidad Politécnica de Madrid, impartiendo diversas asignaturas de programación. En particular, en el curso 2016-2017 ha impartido: *Concurrencia*, *Programación II* y *Algoritmos y Estructuras de Datos*, todas ellas utilizando el lenguaje Java. Además, es experto en el desarrollo de aplicaciones en el lenguaje Java y aplicaciones web, debido a su experiencia profesional (6 años) como Analista y Jefe de Proyecto en diversas empresas en el sector TIC.

5. Desarrollo de las actividades

Como se ha indicado en los apartados anteriores, en la primera tarea de este proyecto se han evaluado las características fundamentales y análisis que realizan las herramientas de análisis más utilizadas. Como resultado de esta evaluación, se decidió centrarse en el lenguaje de programación Java, que es el más utilizado en las asignaturas de programación de las distintas titulaciones de Informática y en el que se ha considerado que es más sencilla la introducción de estas herramientas a los alumnos.

Durante el curso académico se han encontrado algunos inconvenientes para el desarrollo del proyecto. En particular, por diversos motivos algunos de los profesores participantes en el proyecto no pudieron impartir docencia en la asignatura que resulta idónea para este proyecto, *Tecnología de la Programación*, de segundo curso de los Grados en Informática. Una vez comenzado el segundo cuatrimestre, el profesor responsable del proyecto se hizo finalmente cargo de uno de los grupos de esta asignatura en los Grados en Ingeniería Informática e Ingeniería de los Computadores, que es el grupo piloto sobre el que se ha realizado el estudio objeto del proyecto. Dada la incertidumbre inicial sobre la docencia, algunas de las tareas tuvieron que ser retrasadas.

Independientemente de estas circunstancias, se estudiaron las tres herramientas principales, `Findbugs`, `PMD` y `CheckStyle`. De ellas se seleccionó `PMD` por ser la que contenía mayor variedad de reglas (análisis de código fuente de programas) con utilidad para los alumnos de programación. Se han estudiado las 286 reglas para el lenguaje Java de la herramienta `PMD` agrupadas en 27 conjuntos de reglas (*rulesets*). Se puede consultar la lista completa en:

`https://pmd.github.io/pmd-5.8.0/pmd-java/rules/index.html`

De esta lista se han seleccionado los conjuntos siguientes de reglas para que los utilicen los alumnos:

- **Basic:** Es el conjunto básico de reglas que contiene una colección de buenas prácticas que todo programador debe seguir.
- **Basic Ecmascript:** Es un conjunto de reglas básicas para el estándar del lenguaje de programación ECMAscript de ECMA (*European Computer Manufacturers Association*). Contiene reglas para la detección de código inalcanzable, asignaciones en argumentos de funciones, etc.
- **Braces:** Conjunto de reglas para el uso y colocación correcta de llaves en bloques de código.
- **Code Size** y **Complexity:** Conjuntos de reglas para detectar problemas relacionados con el tamaño de métodos y clases, número de parámetros o atributos, y complejidad de métodos y clases. Incluye medidas estándar de la industria como la complejidad ciclomática y *Npath complexity* (número de caminos sin ciclos en un método).
- **Controversial:** Conjunto de reglas diversas que no son aceptadas por toda la comunidad de programadores. Se ha incluido porque contiene algunas reglas muy relevantes para los alumnos: detección de métodos con múltiples `return`, clases sin constructor, etc.
- **Design:** Conjunto de reglas de buenas prácticas de diseño de programas, que proponen enfoques alternativos al alumno.
- **Empty Code:** Detección de bloques vacíos de código que son fuente de problemas potenciales: métodos vacíos, bloques `try/catch` vacíos, etc.

- **Optimization:** Sugerencias de mejoras del programa para mejorar la eficiencia del programa.
- **Security Code Guidelines:** Comprobación de las guías de seguridad publicadas por Oracle, propietaria del lenguaje Java (`http://www.oracle.com/technetwork/java/seccodeguide-139067.html#gcg`).
- **String and StringBuffer:** Reglas relacionadas con el uso de variables de cadenas de caracteres y otras clases relacionadas.
- **Unnecessary** y **Unused Code:** Reglas para la comprobación de código no necesario o bien elementos de código no utilizados, como variables locales o métodos privados que no se utilizan en ningún sitio.

Una vez seleccionada la herramienta `PMD` y los conjuntos de reglas adecuados para el proyecto, se ha procedido a su instalación en dos contextos diferentes:

- Se ha instalado en un servidor del grupo de investigación al que pertenecen los componentes del proyecto (accesible en la url `http://costa.ls.fi.upm.es/web/`) sobre el sistema operativo Linux (distribución `Ubuntu 12.04.5 LTS`). Esta instalación se ha realizado con dos propósitos: por una parte, verificar la posibilidad de uso por parte de los alumnos en un contexto distribuido a través de Internet; por otra parte, para comprobar su idoneidad con algunas de las prácticas entregadas por los alumnos durante el desarrollo normal del curso.
- Se ha probado la instalación de un *plugin* de `PMD` para un entorno de desarrollo particular, `Eclipse IDE for Java Developers`. Se han realizado pruebas en distintas versiones de Eclipse hasta determinar la instalación idónea para facilitar su uso por parte de alumnos y profesores (versión `Luna Service Release 2 (4.4.2)`), pues otras versiones producían problemas al desplegarlas en los ordenadores de los laboratorios de la Facultad.

Después de analizar ambas instalaciones, se ha decidido que los alumnos utilicen el *plugin* proporcionado por la herramienta `PMD`, pues su instalación resulta más sencilla en los laboratorios y permite a los alumnos elegir los conjuntos de reglas de forma individualizada.

La instalación en el servidor del grupo se ha utilizado para probar `PMD` con las prácticas ya entregadas por los alumnos de la asignatura *Tecnología de la Programación*. En particular se ha utilizado la práctica 5, que está formada por un programa de unas 5000 líneas de código Java, el 40 % de las cuales están escritas por los alumnos y que utiliza las clases del API de Java para representar estructuras de datos (`java.util.ArrayList`, fundamentalmente) y librerías gráficas (`java.awt, javax.Swing`), y que hace uso extensivo de genéricos. Estas pruebas nos han permitido refinar la selección de los conjuntos de reglas que son más útiles para que revisen los alumnos.

Una vez hecho esto, se ha organizado una clase extra de laboratorio con los alumnos a los que se les ha proporcionado la instalación de Eclipse con el *plugin* de `PMD` y se les ha formado para su utilización con algunos programas de ejemplo. A continuación han utilizado la herramienta con sus propias prácticas. Al final de la sesión de laboratorio se les ha pedido que rellenen un cuestionario que resume su experiencia con la herramienta.

Este cuestionario ha sido rellenado por 11 grupos de prácticas que corresponden a un total de 18 alumnos de la asignatura. De las comprobaciones más relevantes que ha realizado la herramienta `PMD` se ha pedido a los alumnos que indiquen, en un pequeño subconjunto de comprobaciones, si `PMD` ha detectado alguna anomalía en sus programas. Los alumnos han encontrado resultados relevantes en los siguientes casos:

Comprobación realizada	Núm. prácticas
Perhaps 'xxx' could be replaced by a local variable.	2
[Standard/Modified] Cyclomatic complexity $>$ 10.	8
Avoid really long methods.	3
Avoid `if (x != y) ..; else ..;`	7
Avoid using literals in conditional statements.	6
Position literals first in String comparisons for `equalsIgnoreCase`*.*	1
Use `equals()` *to compare strings instead of == or* `!=`*.*	1
Avoid using `if/for/while` *statements without curly braces.*	8
Avoid unnecessary comparisons in boolean expressions.	3
Avoid empty catch blocks.	1

Además de estos casos proporcionados por los profesores, en el cuestionario presentado a los alumnos se han habilitado dos espacios de respuesta abierta con las siguientes preguntas:

1. *¿Has encontrado otros mensajes que te hayan resultado de utilidad? ¿Cuáles?*
2. *¿Has mejorado el código de tu práctica con la información que has obtenido con PMD? ¿Con qué mensajes y en qué clases?*

los alumnos han encontrado de utilidad otras comprobaciones relevantes, además de las anteriores:

Comprobación realizada	Núm. prácticas
Naming conventions (en general).	6
Too many fields.	1
Methods should have only one exit point.	3
Useless parenthesis.	1
Avoid unused private fields.	1
Parameter 'xxx' is not assigned and could be declared final.	1
Constructor calls overridable method.	1

Se puede verificar que determinadas comprobaciones han sido útiles para gran parte de los grupos que han realizado la encuesta. En general, estas son comprobaciones sencillas y fáciles de entender, o bien están determinadas por la naturaleza del programa y el enfoque proporcionado a los profesores en la solución (en el caso de complejidad ciclomática). Aunque otras comprobaciones solo han sido útiles para un número reducido de alumnos, son muy relevantes porque detectan posibles problemas en los programas.

En cualquier caso, es muy relevante para los alumnos comprobar que existen gran cantidad de normas de estilo que son estándares de la comunidad internacional de programadores para evitar errores potenciales que en algunos casos son muy difíciles de detectar. Mediante este proyecto hemos podido comprobar que los alumnos son muy receptivos a este tipo de herramientas, hasta el punto de que en varios cuestionarios han indicado que les ha resultado de gran utilidad (y solo han recibido una única sesión introductoria a la herramienta) y que herramientas de este tipo deberían implantarse en todas las asignaturas de programación.

Dados los buenos resultados obtenidos en una instalación preliminar de la herramienta `PMD`, nuestro objetivo es implantar este tipo de herramientas como parte del desarrollo ordinario de las asignaturas de programación de los próximos cursos.