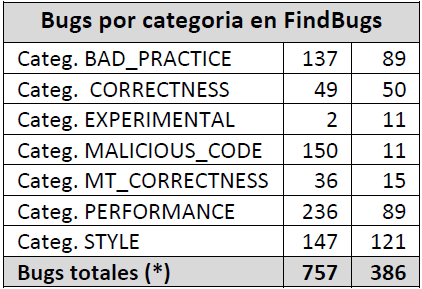
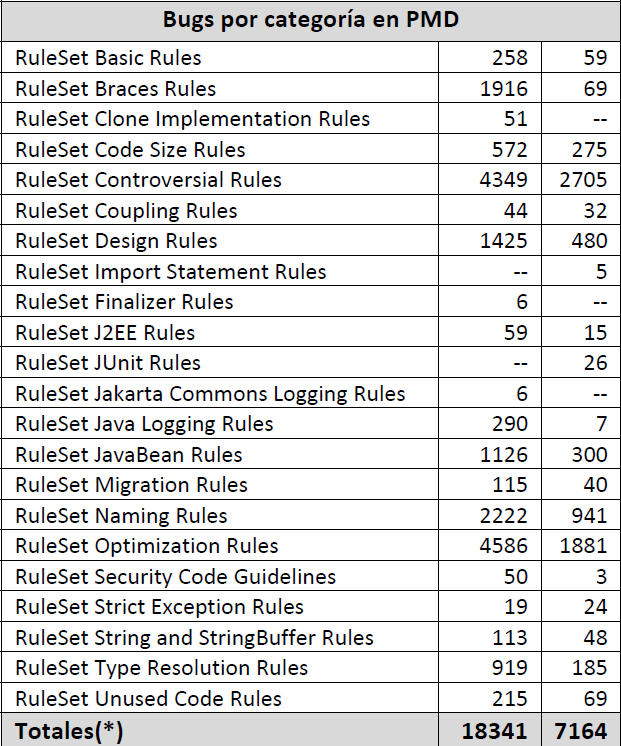
Tarefa 1. Procura de analizadores estáticos de código no mercado e comparativas.

[**Java**](https://es.wikipedia.org/wiki/Java_(lenguaje_de_programaci%C3%B3n))

* [AgileJ StructureViews](https://es.wikipedia.org/w/index.php?title=AgileJ_StructureViews&action=edit&redlink=1)
* [Checkstyle](https://es.wikipedia.org/wiki/Checkstyle)
* [FindBugs](https://es.wikipedia.org/wiki/FindBugs)
* [Hammurapi](https://es.wikipedia.org/w/index.php?title=Hammurapi_(software)&action=edit&redlink=1)
* [PMD](https://es.wikipedia.org/wiki/PMD_(software))
* [Soot](https://es.wikipedia.org/w/index.php?title=Soot&action=edit&redlink=1)
* [Squale](https://es.wikipedia.org/w/index.php?title=Squale&action=edit&redlink=1)
* [Jtest](https://es.wikipedia.org/w/index.php?title=Jtest&action=edit&redlink=1)
* [LDRA Testbed](https://es.wikipedia.org/w/index.php?title=LDRA_Testbed&action=edit&redlink=1)
* [RIPS](https://es.wikipedia.org/w/index.php?title=RIPS&action=edit&redlink=1)
* [SemmleCode](https://es.wikipedia.org/w/index.php?title=SemmleCode&action=edit&redlink=1)
* [SonarJ](https://es.wikipedia.org/w/index.php?title=SonarJ&action=edit&redlink=1)
* [Kalistick](https://es.wikipedia.org/w/index.php?title=Kalistick&action=edit&redlink=1)
* [Visio](https://es.wikipedia.org/wiki/Visio)

**Comparativa**

Cada analizador distingue en su salida las siguientes **categorías de bugs**:

** **

Para elegir una herramienta de análisis estático es necesario tener en cuenta características como la

usabilidad, la eficiencia, la extensibilidad y la técnica de análisis. En muchos casos será necesario

elegir más de un analizador de distinto tipo para poder abarcar más tipos de bugs. De los resultados

extraídos en estas pruebas podemos decir que:

Los dos analizadores son fáciles de usar en cuanto a instalación, configuración, actualización

y uso posterior. Hay plugins para varias plataformas e IDEs. Desde Eclipse ambos añaden una

perspectiva más que permite la ejecución desde un menú de contexto.

PMD resulta más rápido que Findbugs. El consumo de recursos es mayor en FindBugs que en

PMD. FindBugs añade en los ficheros XML atributos de tiempo de ejecución para cada fichero

y proyecto. PMD no dispone de este tipo de información.

En cuanto a extensibilidad PMD resulta más sencillo a la hora de añadir nuevas reglas. Trae

un editor propio que permite copiar, modificar, añadir y eliminar reglas de una forma

intuitiva y sencilla. No pasa lo mismo con FindBugs que dificulta esta tarea por tener que

programar desde cero cada regla que se desee añadir a un proyecto.

La documentación de bugs que permite interpretar los resultados aparece más documentada

en PMD que en FindBugs. PMD incorpora además una funcionalidad para generar

estadísticas resumen.

En los resultados de los dos analizadores, PMD localiza muchos más bugs que FindBugs en los

dos proyectos vistos. Los dos muestran una descripción y localización correcta del error,

aunque referida en algunos casos al número de línea donde comienza la clase y no al número

de línea del bug. Solo en los ficheros XML de PMD se incluyen descripciones de los bugs

detectados.

Los dos analizadores se complementan en cuanto a los tipos de bugs que localizan. La técnica

de análisis es distinta en las dos herramientas, FindBugs está clasificado como Bug Pattern y

PMD como Style‐checker y Bug‐Checker, esta es una de las razones por las que el porcentaje

de bugs comunes no supera el 30% en los dos proyectos vistos.

Con los criterios vistos siempre resultará más eficaz utilizar más de una herramienta para el

análisis de código estático. De esta forma aumentamos el número de bugs detectados.