

**ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA**

GRADO EN INGENIERÍA DEL SOFTWARE

Curso académico 2017/2018

Trabajo de Fin de Grado

**LOGANALYZER: Análisis de logs**

Autor: Carlos Vázquez Losada

Tutor: Micael Gallego Carrillo

**Resumen**

En este proyecto nos centraremos en extender la funcionalidad de un proyecto de partida, utilizando medios visuales y algoritmos de Machine Learning, para ser capaces de anali-zar y evaluar un test basándonos en los logs que éste ha mostrado.

Debido a que hoy en día los servicios en Internet son cada vez más complejos, su desarrol-lo se convierte en un proceso más difícil, y son necesarias herramientas que nos permitan analizar y reparar los bugs de la forma más rápida posible.

LOGANALYZER permite visualizar, de forma muy eficiente, los logs asociados a cada test, y cada ejecución almacenada en ElasticSearch. ElasticSearch es una base de datos que funciona a través de índices, y la cual se encuentra altamente optimizada, resolviendo consultas de una manera increíblemente rápida. Esta aplicación permite también realizar comparativas entre ejecuciones de un mismo test, a fin de determinar qué cosas han cam-biado desde la última modificación del código (útil para encontrar fallos y resolverlos).

Este proyecto tiene como punto de partida el ya expuesto y realizado un par de años atrás por Silvia Moreno ***CloudLogMiner***, que consultaba los logs almacenados en una base de datos de índices como ElasticSearch y los mostraba simulando la funcionalidad de la apli-cación ***OLV (OtroLogViewer)***: estaba compuesta por un árbol de directorios y una tabla en la que se mostraban los logs, sobre los cuales se podían aplicar filtros.

**1. Introducción y Motivación**

Hoy en día, Internet está repleto de servicios y aplicaciones que nos hacen la vida más fácil prácticamente en todos los ámbitos de la cotidianidad. Las empresas compiten día a día por ofrecer un servicio mejor que el que pueden ofrecer sus émulas a los potenciales usuarios y clientes.

Ante esta situación, y debido a la creciente e increíblemente rápida evolución del sector tecnológico (lo que implica una evolución en muchos ámbitos, incluyendo el software, objetivo de este TFG), resulta imprescindible adaptarse a estos cambios y avances pione-ros, ya que mejoran la eficiencia de nuestro código y sus algoritmos, proveyendo servicios más competitivos y con una respuesta más rápida frente a los servicios con tecnologías tradicionales y desactualizadas.

La evolución en nuestro software implica una necesaria e inevitable modificación en el código asociado, con lo que suelen aparecer una gran cantidad de errores, algunos de ellos imperceptibles, y que salen a relucir cuando hemos avanzado en el desarrollo de nuestro proyecto, convirtiendo una labor de minutos en una de horas, e, incluso, días. Además, a este coste temporal hemos de sumarle el coste económico que supone el esfuerzo necesa-rio para solventar estos errores.

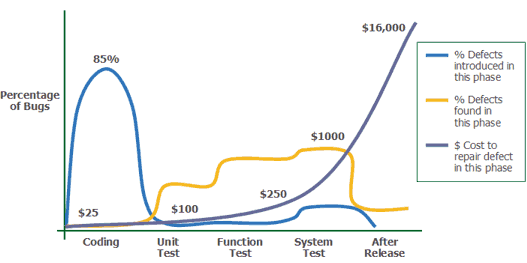


Fig. 1. Evolución económica de la no resolución de cierta cantidad de bugs.

En 1996, Capers Jones [1] hacía explícito, tal y como se muestra en Fig. 1, un aumento en el coste de solucionar ciertos bugs no resueltos durante el proceso de desarrollo del software, haciéndose altamente costosos una vez éste ha sido lanzado al mercado.

Como C. Jones mencionó, incluso programando tests, hay un pequeño porcentaje de erro-res que se escapan a nuestros ojos, por lo que resulta de gran utilidad que desarrolladores y administradores cuenten con herramientas que examinen y revisen la información gene-rada por estos tests en busca de fallos o anomalías.

Este proyecto busca suplir esa necesidad, analizando y comparando distintas ejecuciones de un mismo test y señalando las diferencias, utilizando una interfaz amigable y disponi-ble para cualquier resolución.

<https://www.scientificamerican.com/article/pogue-5-most-embarrassing-software-bugs-in-history/>

# Bibliografía

|  |  |
| --- | --- |
| [1] | C. Jones, Applied Software Measurement, New York: McGraw-Hill, 1996. |