

**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

**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.

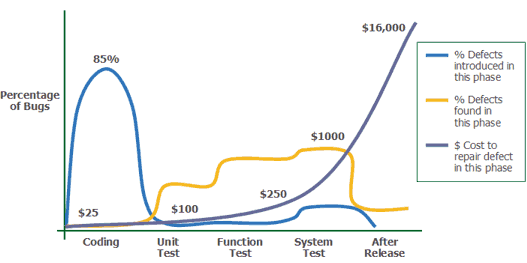


Fig. 1. Evolución económica de la no resolución de cierta cantidad de bugs.

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.

Prácticamente a diario aparecen errores en el software que, en mayor o menor medida, se traducen en pérdidas económicas para la empresa afectada (ya sea por imposibilidad de que los clientes puedan realizar transacciones económicas, la pérdida de información con-fidencial de los usuarios, pérdida de clientes…). Revistas como Scientific American [2] hicieron públicas algunas de las pérdidas económicas por fallos en el software más gran-des de la historia:

* **AT&T publica su servicio de larga distancia**. Durante 9 horas, el software que controlaba los conmutadores de relevo de larga distancia de esta compañía había sido actualizado con errores al lanzar este nuevo servicio, por lo que, en enero de 1990, cualquier cliente que no fuese de la compañía telefónica AT&T podía realizar una de estas llamadas de larga distancia. AT&T terminó perdiendo más de 60 millones de euros en cargos ese día.
* **El MCO (Mars Climate Orbiter) se desintegra en el espacio**. En diciembre de 1998, el software en las máquinas en tierra que controlaban el sistema de propul-sión de la sonda utilizaba las unidades incorrectas (libras por segundo en lugar de Newton por segundo), produciendo que, la sonda espacial robóta de 65 millones de euros ardiera en la atmósfera superior de Marte al impactar en el ángulo incor-recto.
* **Apple Maps direcciona a ningún lugar**. En 2012, tratando de rivalizar con la famosa Google Maps Application, Apple intentó sustituir al aclamado Maps susti-tuyéndolo por un nuevo mapa incorporado en los nuevos iPhone y creado por ellos mismos. El problema fue que todos los lagos, estaciones, puentes y atracciones turísticas se encontraban mal posicionados o ausentes del mapa.

TestLogAnalyzer permite almacenar, consultar y comparar ejecuciones de tests finaliza-dos, ya hayan concluido como erróneos o exitosos, para buscar cambios entre ellos. Lo anterior unido al criterio del usuario, es la herramienta perfecta para encontrar diferencias que haya llevado a una ejecución a fallar con respecto a anteriores que finalizaron correc-tamente y viceversa.

**2. Objetivos**

El objetivo principal de este proyecto será realizar un nuevo componente para ElasTest, plataforma para labores de test en complejos sistemas de software distribuidos y de gran tamaño.

TestLogAnalyzer cubre la necesidad de analizar los logs provenientes de los distintos subsistemas de una aplicación. Así, ElasTest permitirá, una vez este proyecto haya sido integrado, crear proyectos, compuestos por un número concreto de subsistemas, y, a cada subsistema, se le podrá agregar sus correspondientes logs. Después, estos logs pueden ser visualizados y comparados entre sí, apoyándose en un algoritmo de diferenciación.

Este algoritmo es el núcleo del proyecto y parte del creado por Neil Fraser, Google Diff Match Patch, con algunas modificaciones, para realizar esta comparativa semántica:

Proyecto 1

Proyecto 2

Proyecto 3

**. . .**

**. . .**

Proyecto 4

Proyecto N

**Proyectos**

[2] David Pogue, 5 Most Embarrassing Software Bugs in History, Scientific American, 2014 (<https://www.scientificamerican.com/article/pogue-5-most-embarrassing-software-bugs-in-history/>) Consultado 30 de Marzo de 2018

# Bibliografía

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