Area de Beca: CT - Tecnologías

Título del Trabajo: MÉTRICAS ORIENTADAS A OBJETOS Y DETECCIÓN DE DEFECTOS DE SOFTWARE

Autores: CHIAPELLO, JORGE - GREINER, CRISTINA - DAPOZO, GLADYS

E-mail de Contacto: jorgechiapello@gmail.com **Teléfono:**

Tipo de Beca: UNNE Pregrado **Resolución Nº:** 974/13 **Período:** 03/03/2014 - 02/03/2015

Proyecto Acreditado: F010-2013 "Métodos y Herramientas para la calidad del software" - SECYT-UNNE 2014-2017.

Lugar de Trabajo: Facultad de Cs. Exactas y Naturales y Agrimensura

Palabras Claves: calidad de software, gestión cuantitativa de proyectos, prueba de software

Resumen:

En el desarrollo de software, la etapa de prueba se centra en encontrar defectos en el producto a fin de otorgarle confiabilidad. Se conoce que más de la mitad de los errores pasan desapercibidos incluso luego de la entrega al usuario. Una contribución a la solución de esta situación es encontrar formas de realizar la prueba dirigida a las secciones de código más propensas a errores. Para ello resultan útiles los predictores. Un predictor es una métrica señalada de forma temprana, y que tiene una fuerte correlación con algún resultado posterior. Permite anticipar las secciones más propensas a defectos o a producir errores en el futuro. Por otra parte, la tendencia de la industria hacia la adopción de la programación orientada a objetos (OO) no ha disminuido en los últimos años debido a la promoción de características deseables en el software, como la reutilización de código, encapsulación, abstracción y modularidad, entre otras. Paralelamente al desarrollo de aplicaciones OO crece la necesidad de métricas que permitan medir los atributos del software que indican su calidad. Se calcula que el ahorro de costo de mantenimiento es del 42% mediante el uso de métricas OO. Con el objetivo de contribuir a la calidad del software a través de la medición y prueba, se realizó un estudio del estado del arte a fin de obtener una visión actualizada sobre las métricas OO y sus posibles relaciones con la detección de errores en la prueba. Adicionalmente, se profundizó el estudio sobre las herramientas utilizadas en el marco de la gestión cuantitativa de proyectos de software, a fin de avanzar en el diseño y desarrollo de una aplicación software que permita detectar clases propensas a errores en función del análisis de los valores de las métricas aplicadas al código fuente. Con este propósito, se llevó a cabo una Revisión Sistemática de Literatura (RSL), según la metodología propuesta por Kitchenham. El protocolo de búsqueda incluyó los términos más relevantes. Se realizó una clasificación de los resultados con los siguientes criterios: Herramienta propuesta, Contexto en el cual se aplica, Aspecto de la gestión o del ciclo de vida del software a la que se orienta, País, Año. Se obtuvo un panorama actualizado acerca de la gestión cuantitativa de proyectos, y las herramientas de apoyo utilizadas. Posteriormente se indagó sobre metodologías para determinar el grado de complejidad en aplicaciones OO mediante el cálculo de métricas OO. Como repositorio de código fuente se estudió GitHub, una plataforma para alojar proyectos utilizando control de versiones. Permite acceder tanto al código de un proyecto, como a los errores y defectos reportados, y sus soluciones. Para determinar la relación entre las métricas OO que evidencian la complejidad del software con los errores detectados en la prueba, se propone la elaboración de una aplicación que permita automatizar este proceso, mediante el registro de las mediciones y de los errores detectados. Se realizó el análisis y diseño de la misma, considerando la arquitectura MVC (Modelo, Vista, Controlador). Se implementaron los módulos Vista y Controlador. Actualmente se trabaja en el desarrollo del Modelo. Los datos requeridos se tomarán del repositorio GitHub. Utilizando el código de las aplicaciones open source, se calcularán las métricas complejidad y se relacionarán con los errores informados en la prueba. Esta relación establecerá características de las clases propensas a fallo, lo que permitirá anticipar un mejor control, haciendo más eficiente la etapa de mantenimiento del software.