Estrategia de prueba para aplicaciones de gobierno electrónico

Test strategy for e-government applications

Felinda Rosabel León Mendoza
Ingeniería en Centro CEGEL,
Universidad de las Ciencias Informáticas.
Km 21⁄2 Autopista La Habana - San Antonio de los Baños, La Habana.
frleon@uci.cu.

Anabel Fé León Mendoza
Subdirección de Tecnología DESOFT,
Empresa de Desarrollo de Software.
Calle 22 No.104 e/ 1era y 3era, Miramar, Playa, La Habana.
anabel.leon@desoft.cu

Fecha de recibido: 3 de julio de 2019

Fecha de aprobado: 4 de julio de 2019

Abstract— El mundo actual atraviesa cambios de una manera vertiginosa, todo cuanto por mucho tiempo se ha realizado de forma manual, en estos momentos se ha automatizado por medio del software. En virtud de este nuevo contexto, la mayoría de las empresas dedicadas al desarrollo de software invierten grandes esfuerzos para obtener productos con mayor calidad, al mismo tiempo aspiran a ser un referente en el mercado mundial. Las herramientas de desarrollo de software, se han vuelto imprescindibles para automatizar la ejecución de las pruebas, medir y controlar los procesos dentro del desarrollo del software, para con ello conseguir un producto de software de mejor calidad.

La integración de productos, es uno de los problemas comunes en el desarrollo de software, pues para hacer funcionar un sistema es necesario integrar cada uno de los componentes, los cuales se desarrollan de manera independiente. En el Centro de Gobierno Electrónico, por sus siglas, CEGEL de la Universidad de las Ciencias Informáticas (UCI) se presentan diversos problemas causados por la deficiente gestión de las pruebas de integración de los productos desarrollados en el centro.

La presente investigación tiene como objetivo dar a conocer la estrategia de pruebas que se utiliza en el Centro de Gobierno Electrónico (CEGEL) de la Universidad de las Ciencias Informáticas (UCI) para las

aplicaciones de gobierno que se desarrollan en dicho centro, permitiendo garantizar la calidad de sus productos.

Keywords— calidad, herramientas, integración, pruebas, software, gobierno digital.

Abstract— Today's world is going through changes in a dizzying way, everything that for a long time has been done manually, now it has been automated through software. By virtue of this new context, most of the companies dedicated to software development invest great efforts to obtain products with higher quality, at the same time they aspire to be a reference in the world market. Software development tools have become essential to automate the execution of tests, measure and control the processes within software development, in order to achieve a better-quality software product.

Product integration is one of the common problems in software development, since to make a system work it is necessary to integrate each of the components, which are developed independently. In the Center for Electronic Government, for its acronym, CEGEL of the University of Computer Sciences (UCI) there are various problems caused by poor management of the integration tests of the products developed at the center.

The present research aims to present the test strategy used in the Electronic Government Center (CEGEL) of the University of Computer Sciences (UCI) for the government applications that are developed in said center, allowing to guarantee the quality of its products.

Keywords— quality, tools, integration, testing, software, digital government.

I. INTRODUCCIÓN

La calidad del software se ha convertido en un eje dentro de las empresas que utilizan procesos de desarrollo de software, en los cuales se invierten grandes esfuerzos para lograr obtener productos de alta calidad, con el fin de convertirse en un referente en el mercado de la producción de software. Actualmente, el desarrollo de aplicaciones se encuentra apoyado por diferentes normas, certificaciones de procesos, prácticas y herramientas que aportan mejoras en el diseño, implementación y desarrollo de aplicaciones (Rodríguez, 2015). Las

herramientas de desarrollo de software, se han vuelto imprescindibles para automatizar la ejecución de las pruebas, tanto para medir, así como controlar los procesos dentro del desarrollo del software, para con ello conseguir un producto de software de mejor calidad.

Cabe preguntarse entonces, ¿qué es la calidad? Por un lado la Real Academia Española la define como: "propiedad o conjunto de propiedades inherente a algo, que permite juzgar su valor, definición que está orientada al mercado" (Española, 2019) por otro lado, la International Standards Organization, ISO en la norma 8402:1994, la define como la "Totalidad de propiedades y características de un producto, proceso o servicio que le confiere su aptitud para satisfacer unas necesidades expresadas o implícitas." En la actualización de la Norma ISO 9000:2000, la definición de calidad indica: "Grado en el que un conjunto de características inherentes cumple con los requisitos". En suma, la definición de calidad enfocada al software, según la IEEE es "el grado con el que un sistema, componente o proceso cumple los requerimientos especificados y las necesidades o expectativas del cliente o usuario" (Standardization, 2016).

El mejoramiento de los procesos de desarrollo de software busca reducir los costos, incrementar la productividad y aumentar la calidad de los productos y servicios resultantes de la ejecución de los procesos. El mejoramiento de los procesos, persigue alinear las organizaciones con referentes de la industria del software, tales como la International Organization for Standardization (ISO) 9001 o el Capability Maturity Model Integration (CMMI), para poder competir en los mercados globalizados donde, en muchos casos, son requeridas certificaciones y/o evaluaciones. Según Crosby, Deming, Juran y Humphrey (S.E.I., 2017) definen el CMMI como una representación simplificada del mundo que contiene los elementos esenciales de los procesos eficaces de una o más disciplinas, y describen un camino evolutivo desde procesos ad hoc e inmaduros a procesos disciplinados y

maduros con calidad y eficiencia mejoradas.

El diccionario Cambridge lo define como un método de 5 etapas para desarrollar y mejorar programas informáticos o procesos de gestión con el fin de cumplir con altos estándares. El Software Engineering Institute (SEI, 2010).

En consideración a la bibliografía consultada se puede definir, que CMMI es un conjunto de modelos elaborados por el Software Engineering Institute (SEI) que permite obtener un diagnóstico preciso de la madurez de los procesos relacionados con las tecnologías de la información de una organización, y describe las tareas a llevar a cabo para mejorar esos procesos. En Cuba, varias organizaciones tratan de mejorar sus procesos de desarrollo en la rama de la industria del software, aplicando técnicas, herramientas y metodologías para mejorar la gestión de los mismos. La Universidad de la Ciencias Informáticas (UCI), uno de los centros líderes para el proceso de informatización de la sociedad cubana no está exenta de esto, y para lograr un mejor funcionamiento de sus procesos productivos está inmersa en el proceso de certificación de CMMI nivel 3.

Este modelo define un conjunto de buenas prácticas de ingeniería del software que facilitan el mejoramiento gradual de los procesos de gestión de proyectos, gestión de procesos, ingeniería y soporte. Dentro del área de ingeniería se encuentra el área de Integración del Producto (que es un área de proceso de ingeniería en el nivel de madurez 3 de CMMI) que establece las expectativas de una organización para desarrollar las estrategias de integración de productos, los procedimientos y un entorno de integración; asegura la compatibilidad entre las interfaces de los componentes del producto; para ensamblar los componentes del producto; y para entregar el producto y sus componentes con calidad.

El Centro de Gobierno Electrónico CEGEL de la UCI durante el proceso de desarrollo de software

no lleva a cabo una correcta aplicación de esta área de proceso pues, al aplicar una encuesta a programadores, jefes de proyectos, administradores de calidad y arquitectos de software de los diferentes proyectos del centro se identificó lo siguiente: El f lujo de actividades que deben desarrollarse cuando un componente necesita integrarse no está debidamente documentado. Esto provoca en cada proyecto del centro que los artefactos existentes relacionados con la integración de los componentes de productos, e interfaces no brindan la información suficiente para que el proceso se desarrolle satisfactoriamente.

Según lo anterior, resulta muy difícil conocer cuáles son todos los servicios que brinda y consume un componente; pues, cuando existen cambios en un componente y en sus servicios es muy complicado determinar el impacto del cambio en el sistema.

Tampoco cuentan con un registro de dependencia, por ello, cuando un componente no funciona correctamente; es habitual que no se conozca en principio si el problema es propio de él, o de algún servicio de los cuales consume.

No se definen las pruebas para determinar el correcto funcionamiento de la integración, ocasionando que la ocurrencia de defectos en las últimas etapas del desarrollo sea habitual. La solución de estos errores eleva los costos del proyecto y aumentan el tiempo de desarrollo porque se hace más complicada su solución.

Los programadores y arquitectos de software no utilizan las herramientas de pruebas establecidas en la UCI para la integración continua, pues, cuando los productos son revisados por el grupo de calidad del centro, el número de no conformidades correspondiente tanto a los casos de pruebas como a opciones que no funcionan es muy elevado. Esto atenta a la planificación de los proyectos, pues si los programadores no tienen tiempo en la resolución de las No Conformidades (NC) se atrasa el

proyecto en ser enviado al laboratorio de pruebas de calidad UCI, adicionalmente siempre se quedan algunas NC sin resolver, por ello el número de iteraciones es siempre de tres y en ocasiones de cuatro.

Como resultado del análisis de la situación antes mencionada se decide:

Desarrollar y ejecutar una estrategia de pruebas para la aplicación de la práctica de integración continua para mejorar la integración de los productos desarrollados en el Centro de Gobierno Electrónico CEGEL hasta contribuir a perfeccionar la compatibilidad entre los componentes de producto e interfaces.

II. MÉTODOS

Para llevar a cabo la investigación de una población de 50 especialistas entre 30 programadores, 20 arquitectos y 10 probadores, se realizó un muestreo por conveniencia, obteniendounamuestrade 12 arquitectos y 10 programadores y 5 probadores. Los métodos científicos utilizados para realizar la investigación se citan a continuación:

Entre los métodos teóricos se utilizó el histórico- lógico para el estudio crítico de los trabajos anteriores y para utilizarlos como punto de referencia y comparación con los resultados alcanzados. El analítico-sintético al descomponer el problema de investigación en elementos por separado y profundizar en el estudio de cada uno de ellos, para luego sintetizarlos en la solución de la propuesta. La inducción para la identificación de la problemática y de las soluciones.

Entre los empíricos se realizó la entrevista para realizar un levantamiento de las características del proceso de Integración del Producto en la UCI. La encuesta para identificar como se realizaba el proceso de Integración del Producto en los proyectos del centro. El método experimental para comprobar la utilidad. de los resultados obtenidos a partir de la estrategia.

Para una correcta elaboración de la estrategia, se determinó el uso de

una metodología ligera, escogiéndose el Proceso Unificado Ágil (AUP, por sus siglas en inglés, Agile Unified Process) dado que cuenta con siete f lujos de trabajos, cuatro: ingenieriles y tres de apoyo Modelado, Implementación, Prueba, Despliegue, Gestión de configuración, Gestión de Proyectos y Ambiente. El f lujo de trabajo de Prueba tiene como objetivo realizar una evaluación objetiva para garantizar la calidad, esto incluye la búsqueda de defectos y validar que el sistema funciona tal como está establecido, tras verificar que se cumplan los requerimientos. Además de promover el desarrollo dirigido por pruebas (Test Driven Development-TDD) y ser la metodología de software definida para los proyectos de desarrollo en la UCI.

III. DESARROLLO DE CONTENIDOS

Para llevar a cabo la estrategia de pruebas se hace necesario definir que es una estrategia, según el diccionario de la Real Academia Española una estrategia es una serie de acciones muy meditadas, encaminadas hacia un fin determinado (Española, 2019). Por otro lado, las pruebas de software son las investigaciones empíricas y técnicas cuyo objetivo es proporcionar información objetiva e independiente sobre la calidad del producto a la parte interesada o stakeholder. Es una actividad más, en el proceso de control de la calidad que puede ser ejecutada en cualquier momento del desarrollo del software (Rodríguez, 2015).

Para la ejecución de la estrategia el grupo de calidad de dicho centro se usó la familia de normas ISO/IEC 25000, normas de referente para la calidad del producto de software cuyo objetivo principal es guiar el desarrollo de los productos de software mediante la especificación de requisitos y evaluación de características de calidad.

En la primera etapa de ejecución de la estrategia, el grupo de calidad estableció para los proyectos del centro implementar el proceso de integración continua, práctica de desarrollo de software que consiste en automatizar el proceso de

integración de código con determinada frecuencia, permitiendo identificar en etapas tempranas del desarrollo de la solución errores de integración entre componentes, códigos inestables y errores de validación. La integración continua, funciona realizando comprobaciones interconectadas entre diferentes herramientas que miden o controlan un proceso dentro del desarrollo de Software. Para determinar las herramientas a emplear se realizó un estudio de varias herramientas, tanto privativas como Open Source (código abierto o libre) entre las más conocidas, algunas de las estudiadas fueron: el CruiseControl (Open Source), AntHill (privativa/ Open Source) y el DamageControl (privativa), Jenkins (Open Source), GitLab(Open Source) (Tello, 2016).

En la segunda etapa de la estrategia se definieron los siguientes procesos a seguir para llevar a cabo la integración continua:

1. Proceso de control de versiones
2. Proceso de automatización de pruebas unitarias
3. Proceso de Construcción de Código Fuente
4. Proceso de Automatización Pruebas Funcionales
5. Proceso de Despliegues
6. Proceso de Integración Continua

El proceso de control de versiones es el proceso de gestionar los cambios que se realizan a una aplicación de software, ya sean modificaciones al código fuente o la configuración de la misma. Dentro de las herramientas estudiadas para el control de versiones están Concurrent Versions System (CVS) (Open Source), Subversion (Open Source), AccuRev(Propietario), GIT Lab(Open Source), Team Foundation Server(Propietario) etc (Salomón, 2015). El proceso de pruebas unitarias es la forma de comprobar el correcto funcionamiento de una unidad de código, para estas pruebas el uso de herramientas agiliza actividades de pruebas atómicas como probar una clase, o un módulo en particular. La mayoría de ellas funcionan como un

framework para la automatización de las pruebas unitarias, y de integración. Contienen clases y métodos que facilitan la tarea de realizar pruebas en el sistema para asegurar la consistencia y funcionalidad. Las herramientas estudiadas fueron JUnit (Open Source), PHPUnit (Open Source), SimpleTest (Propietario) y TestNG (Open Source) (Torrado, 2017).

El proceso de construcción de código fuente permite realizar pruebas estáticas de código, por eso se debe contar con unos lineamientos para la codificación. En la realización de estas pruebas es fundamental contar con una herramienta que facilite el encontrar errores como: código muerto, código quemado, código no documentado, ciclos mal construidos, errores de sintaxis, etc. Las herramientas estudiadas para la construcción de código fueron: PHPLint (Propietario), YASCA (Open Source), PMD (Propietario) y SonarQube (Open Source). Para la elección de la herramienta de pruebas estáticas de código las características más relevantes son el los lenguajes de programación y las plataformas que se utilizan en la compañía y la integración con las demás herramientas con las que se desee trabajar.

El proceso de pruebas funcionales requiere de una inversión en recurso humano y tiempo para el diseño y ejecución de las pruebas, sin contar con el tiempo de capacitación, si las pruebas las realizará un tercero. La automatización de Pruebas de software, se realiza a través de una herramienta diseñada para ese fin. Las herramientas estudiadas fueron Unified functional testing (HP) (Propietario), Selenium By SeleniumHQ (Open Source), QMatic by QVision(Open purce) etc (Salomón, 2015).

El proceso de Integración Continua busca principalmente verificar que cada modificación que se realice al código de una aplicación no genere problemas al momento del despliegue en los ambientes de pruebas o Productivo. La Integración Continua permite al desarrollador tener la totalidad de la aplicación y su

comportamiento ante cualquier cambio. Las herramientas estudiadas fueron (Cruise Control (Open Source), Jenkins/Hudson (Open Source), Continuum (Open Source) y Bamboo (privativa) (Salomón, 2015).

De las herramientas estudiadas para la estrategia de pruebas para las aplicaciones de gobierno electrónico de CEGEL se seleccionó la herramienta GIT Lab, debido a que integra en una sola herramienta varias de las herramientas estudiada, es necesaria para llevar a cabo los diferentes procesos de pruebas para manejar nuestro código, compilarlo, y realizarle diferentes acciones y pruebas al resultado (Martínez, 2018).

El proceso que se siguió para aplicar la integración continua en el centro fue el siguiente:

Los desarrollares envían sus modificaciones al repositorio gestionado por un sistema de control de versiones (Git). • El servidor de integración continua (Git Lab) monitoriza el repositorio buscando cambios y ejecuta automáticamente el build cada vez que se produzca algún cambio. • Una vez finalizado el proceso de build (integración, construcción, pruebas y despliegue) el servidor envía a los responsables un email con el resultado del proceso para obtener feedback de posibles errores que se estén produciendo. • El servidor realiza el paso 2 continuamente. Las ventajas de usar un servidor de integración continua son: • Posibilidad de monitorizar el control de versiones y actuar ante cualquier cambio y aplicación de herramientas de integración continua, en proyectos software.

Además, se identificaron los roles involucrados para el proceso de pruebas como fueron los que se muestran a continuación:

1. Administrador de la configuración: Encargado de la administración de la configuración del servidor de integración.
2. Probador de pruebas automatizadas: Encargado de desarrollar los casos de pruebas.
3. Probador de script: Encargado de ejecutar los scripts cada cierto

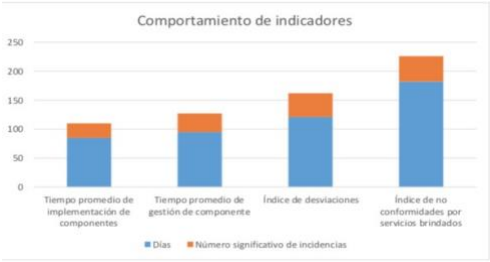
tiempo en el proyecto para la integración continua.

4. Probadores: Son los encargados de realizar las pruebas estáticas a las aplicaciones desarrolladas en el centro.

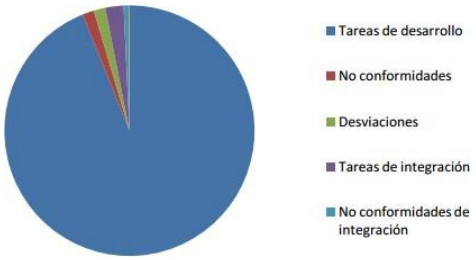
IV. RESULTADOS Y DISCUSIÓN

Como resultado final de la investigación se realizó un experimento para determinar el efecto causado por la aplicación de la estrategia propuesta. El experimento se desarrolló en condiciones naturales. Para perfeccionar la compatibilidad entre los componentes de producto e interfaces se tuvieron en cuenta los siguientes indicadores:

- Tiempo promedio de implementación de componentes: Significa el tiempo promedio de implementación de los servicios de un componente, medido en días.
- Tiempo promedio de gestión de componente: Se refiere al tiempo transcurrido desde que se detecta la necesidad de un nuevo componente hasta que se notifica la disponibilidad del mismo.
- Índice de desviaciones: Es la razón entre la cantidad de desviaciones por integración en un proyecto sobre la cantidad de componentes consumidos.
- Índice de no conformidades por servicios brindados: Es la razón entre el número de no conformidades de integración en los servicios brindados de un componente y el total de servicios brindados.



A continuación, se muestra un gráfico con el comportamiento de las tareas correspondiente a los indicadores mencionado con anterioridad y como se puede observar el número de no conformidades de integración, las desviaciones y las no conformidades disminuyeron, aunque para ello se aumentó el número de tareas en el desarrollo para lograr productos con mayor calidad y estabilidad en sus procesos.



V. CONCLUSIONES

La ejecución del experimento permite presentar el impacto positivo que tiene la aplicación de la práctica de integración continua en los proyectos de desarrollo del centro CEGEL como guía para otras entidades. A través de la integración continua se logró que el equipo de desarrollo se desentendiera del proceso de compilación e integración de los componentes y que pudieran visualizar en que momento fallan las integraciones. Por otra parte, la sincronización de los equipos de trabajo permitió tanto identificar en el momento que los desarrolladores están actualizando componentes dependientes entre sí, así como alertar esta situación para evitar conflictos, propiciando así el reúso de componentes en los proyectos de software. Todos los indicadores mejoraron con la aplicación de la propuesta, destacándose la reducción del tiempo promedio de implementación de componente, el índice de desviaciones y el índice de no conformidades.

VI. REFERENCIAS

- Española, R. A. (marzo de 2019). Diccionario de la Real Academia Española. Recuperado el 25 de marzo de 2019, de https://dle.rae.es/calidad
- Martínez, J. I. (28 de enero de 2018). GitLab, la completa herramienta integrada para desarrollo de software. Recuperado el 18 de marzo de 2019, de https://picodotdev.github. io/blog-bitix/2017/12/gitlab-la-completa-herramienta-
- Rodríguez, N. G. (marzo de 2015). Las Pruebas de Integración como Proceso de la Calidad del Software en el Ámbito de las Telecomunicaciones. Recuperado el 20 de marzo de 2019, de

https://www.sciencedirect.com/science/article/pii/S0123592

- S.E.I., S. (2017). CMMI. Recuperado el 21 de Febrero de 2019, de https://www.tutorialspoint.com/es/cmmi/cmmi_overview.htm
- Salomón, A. y. (enero de 2015). La Integración Continua Aplicada en el Desarrollo de Software en el Ámbito Científico – Técnico. Recuperado el 27 de mayo de 2019, de revistas.uigv.edu.pe/index.php/perspectiva/article/download/253/251
- SEI, S. (2010). CMMI para Desarrollo, en Mejoras de los procesos para el desarrollo de mejores productos y servicios C.M.U.A.S.I. (Vol. Versión 1.3). Instituto. México : Editorial Universitaria Ramón Areces.
- Standardization, I. O. (febrero de 2016). ISO. Recuperado el 26 de marzo de 2019, de https://www.iso.org/home.html
- Tello, P. G. (abril de 2016). Tesina de licenciatura Evaluación de calidad de un producto de software. Recuperado el 27 de Marzo de 2019, de http://sedici.unlp.edu.ar/bitstream/handle/10915/58934/Documento_completo.pdf- PDFA.pdfsequen
- Torrado, J. y. (12 de octubre de 2017). Introducción a GitLab. Recuperado el 18 de marzo de 2019, de https://desarrolloweb.com/articulos/introduccion-gitlab.html