**Pruebas Continuas**

AUTOR, OSCAR TORAL RUIZ

PRAXIS, **TESTING**

MÉXICO, CIUDAD DE MÉXICO.

**Introducción**

Una de las características más importantes del desarrollo ágil es la entrega continua y regular de valor al cliente. Para esto existen varias herramientas para hacer la, las cuales nos permitirán tener siempre un aplicativo compilable.

Una experiencia que ayuda aminorar el riesgo por los cambios de las actualizaciones hechas durante un sprint, es la ejecución de las pruebas automatizadas que tengan las diversas etapas del ciclo de vida del producto así que serán las pruebas unitarias las que se ejecuten con un periodo más frecuente y de manera completa.

**Palabras Clave:** Integración Continua, Pruebas Unitarias, Pruebas de Automatización, Pruebas Continuas,

**Resumen**

Lo descrito antes, implica más esfuerzo y tiempo, además de las etapas de diseño y construcción o codificación, el equipo de ingeniería debe tener tiempo suficiente para ejecutar estas pruebas. La respuesta es realizar el mismo proceso de continuidad que se utilizó en la codificación del software, pero para aplicarlo a las pruebas. A esta técnica se le conoce como pruebas continuas y consiste en adecuar al ciclo de vida, etapas donde se ejecuten las pruebas de los diferentes niveles tales como: (unitarias, integración y sistema)

El objetivo es tener una retroalimentación rápida de la calidad del aplicativo y dársela al equipo de ingeniería.

Esta no es una actividad fácil de realizar, debido a que se debe cambiar la forma en la que se lleva el proceso.

Es necesaria la responsabilidad de todos los miembros del equipo para construir y mantener las pruebas del sprint, además se deben seleccionar las herramientas que se ocuparan e introducirlas al proceso.

Primero se debe diseñar una prueba por cada una de las piezas de software. La prueba debe verificar que el aplicativo haga lo que tiene que hacer en base a los requerimientos del cliente final, entonces, deberá validar que, al ejecutar la prueba, esta regrese un resultado de acuerdo al diseño, esto es lo que se conoce como prueba unitaria automatizada.

Estos son algunos principios que se deben tomar en cuenta para el diseño de pruebas:

* Debe reducirse el riesgo en las pruebas de software, no introducirlo.
* Realizar una sencilla ejecución en las pruebas de software.
* Fácil mantenimiento en las pruebas de software.

Para ejecutar prácticas comunes en el buen diseño de pruebas son:

* Cada prueba puede revisar solo una cosa a la vez, es una mala práctica tener más de una verificación en un solo script de prueba.
* Tener una línea base de pruebas definida, no debe haber dos o más pruebas que fallen por el mismo motivo.
* Las pruebas deben ser independientes a la base de datos, de otros tests, de ui, archivos, etc.

Con estos indicios es más fácil realizar un proyecto de pruebas de software.

Ya listas las pruebas de software es necesario concentrarlas y aislarlas ya sea por arquitectura o capa de integración, con el fin de tener claro qué partes son las que se verán afectadas.

Ya aisladas las pruebas se forman “suites”, el cual, su objetivo es ser otro elemento de la organización del proyecto, así al tener las pruebas segmentadas podemos ejecutarlas de manera selectiva.

Lo siguiente es la elección de una herramienta que se encargue de ejecutar los tests y mostrar los resultados.

Una vez teniendo todas las herramientas necesarias, es fundamental integrarlas de modo que no afecten nuestro ciclo de desarrollo.

¿Inversión o gasto?

En este tipo de pruebas es bien sabido que se requiere un esfuerzo adicional y requieren un alto grado de conocimiento de la herramienta que se utilizará, pero es una inversión que muestra su verdadero valor una vez que los proyectos van creciendo y se vuelven más complejos.

Para la uno de los elementos es clave, definir el momento y la frecuencia con la cual se ejecutaran los distintos tipos de prueba.

**Otras pruebas**

Las pruebas unitarias son las más baratas y efectivas, pero estas no son las únicas que se pueden ejecutar dentro de un ciclo continuo hay otras que también deben ser consideradas.

Por ejemplo, realizar una prueba de integración únicamente con tests unitarios es muy complicado ya que por su origen estas son independientes, es en estas situaciones donde se debe tener en cuenta otras estrategias basadas en pruebas de caja negra.

Los otros tipos de prueba (funcionales y NO Funcionales) generalmente son costosas en tiempo y recursos de cómputo, por lo que se recomienda realizarlar un modelo independiente al de integración continua.

### **Conclusión**

El valor agregado de la práctica de pruebas continuas en la integración continua, depende de cómo se haya implementado el proceso y de adecuado a la organización, la prueba continua genera una rápida retroalimentación del aplicativo

.

Es necesario un conocimiento especializado. Así como un skill técnico para poder desarrollar los scripts de pruebas y resolver todos los retos a los que nos enfrentaremos dentro de la ejecución de las pruebas

La integración continua es una tendencia que está siendo implementada en muchas organizaciones y que ejecutarla bien, requiere de un seguimiento constante para validar que la práctica se está llevando acabo de la mejor manera.

**Referencias**

<https://software.microfocus.com/es-es/products/automated-continuous-testing-integration-delivery-tools/overview>

<http://www.silicon.es/75-las-organizaciones-ve-las-pruebas-continuas-software-aspecto-critico-negocio-2352854>

<https://sg.com.mx/revista/46/pruebas-continuas>