

# Fundamentos de la Prueba de Software

## CONCEPTOS, JUSTIFICACIÓN Y ALCANCE



**Luis Vinicio León**

**Carrillo** es profesor-investigador del Departamento de Electrónica, Sistemas e Informática del ITESO, y director general de e-Quality S.A. de C.V., empresa especializada en prueba de software. Luis Vinicio es doctorando por la Universidad Técnica de Clausthal, Alemania; su trabajo predoctoral giró alrededor a la aplicación de los lenguajes formales en la Ingeniería de Software. Es coautor de un marco tecnológico que hoy permite a e-Quality desarrollar empresas de prueba de software. Su tesis doctoral está enfocada en la aplicación de métodos y lenguajes formales para hacer más eficiente y efectiva la prueba de software. Luis Vinicio es co-fundador del Capítulo Guadalajara de la AMCI y su Secretario actual.

**E**n el número anterior hablamos sobre el contexto de la prueba de software. En esta ocasión, nos concentraremos en definir algunas definiciones relacionadas con la prueba de software, así como su justificación y alcance.

Las descripciones de algunos conceptos que expondré en el presente, si bien son generalizables, estarán enunciados desde la perspectiva de la técnica denominada “prueba de caja negra”, consistente en ejecutar el sistema a probar revisando los requerimientos, con la consigna de detectar insatisfacción de los mismos. La razón es que facilita la exposición sin introducir complejidad innecesaria.

### Definiciones

Complementando nuestro primer acercamiento a una definición, que tuvimos en el número anterior, podemos argüir que la prueba de software:

*Es un proceso en el que se revisa el sistema a probar (el SUT) bajo condiciones definidas explícitamente, y se le aplica (eventualmente con apoyo de software especializado de tipo CAST) un conjunto de estímulos (los casos de prueba) diseñados de manera sistemática utilizando técnicas apropiadas, con el objetivo de detectar niveles inadecuados de calidad. Este proceso debe llevarse a cabo disciplinadamente, y respaldarse en métricas bien definidas. Todas estas actividades y sus resultados son documentados, en especial las fallas detectadas [1].*

Precisemos cada uno de los conceptos de esta definición. Intuitivamente, un proceso puede verse como una secuencia de actividades, cada una de las cuales genera productos, tiene insumos asociados, e involucra gente (roles) y otros recursos (v.gr. hardware y software).

Un primer bosquejo del proceso de prueba de caja negra sería el siguiente, que refinaremos en números subsiguientes:

1. Establecer alcances, entregables y criterios de éxito
2. Estimar el esfuerzo de prueba
3. Planear el proyecto

4. Reproducir el contexto del SUT
5. Hacer
  - a) Diseñar casos de prueba
  - b) Aplicar casos de prueba
  - c) Reportar métricas y dar seguimiento
  - d) Reportar análisis de resultados
 Mientras no (criterio de terminación)
6. Hacer el cierre del proyecto

El **SUT** (*System Under Testing*) se refiere en general al elemento a probar. Por otro lado, las herramientas que utiliza el *tester* para llevar a cabo las actividades anteriores son cobijados bajo el acrónimo **CAST** (*Computer Aided Software Testing*).

En cuanto a los casos de prueba, es deseable que presenten las siguientes características:

- Ortogonalidad. No tener casos que incluyan segmentos de otros
- Efectividad. Que detecten fallas.
- Ejemplaridad. Que “con poco se pruebe mucho”.
- Claridad. Que evidencien fallas de manera clara.

Con calidad nos referimos a [2]:

- El grado en que el producto satisface los requerimientos funcionales y no-funcionales explícitamente establecidos.
- El nivel al que se siguieron los estándares de desarrollo explícitos y documentados.
- Que el producto muestre las características implícitas que se espera de todo software desarrollado profesionalmente.

Una equivocación es una acción incorrecta cometida por un humano (v.gr. no presionar [shift]), que ocasiona que se genere una falta (sin el [shift], queda “<” en vez de “>” en el código fuente); al ser ejecutada, la falta se evidencia en forma de lo que llamamos una falla. El error es la magnitud de la diferencia entre el resultado esperado y el obtenido [3].

### Justificación

Los principales objetivos que se buscan con la prueba de software suelen ser:

- Conocer el nivel de calidad de productos intermedios, para actuar a tiempo (v.gr. rehacer un componente); esto

facilita una administración realista del time to market del producto en cuestión.

- No pagar por un producto de software sino hasta que alcance el nivel de calidad pactado; esto eleva el nivel de certidumbre en el comprador de software, y minimiza riesgos.
- Disminuir la penosa y costosa labor de soporte a usuarios insatisfechos, consecuencia de liberar un producto inmaduro. Esto puede mejorar la imagen de la organización desarrolladora (y la credibilidad en ella).
- Reducir costos de mantenimiento (la fase más costosa del desarrollo de software), mediante el diagnóstico oportuno de los componentes del sistema (v.gr. seguimiento a estándares, legibilidad del código, integración adecuada de los componentes, rendimiento apropiado, nivel y calidad del reuso, calidad de la documentación, etc.).
- Obtener información concreta acerca de fallas, que pueda usarse como apoyo en la mejora de procesos, y en la de los desarrolladores (v.gr. capacitación en áreas de oportunidad).

Entre más pronto se apliquen mecanismos de prueba en el proceso de desarrollo, más fácilmente podrá evitarse que el proyecto se salga del tiempo y presupuesto planeado, pues se podrán detectar más problemas originados en las fases tempranas del proceso, que son los que mayor impacto tienen.

### Alcance

La prueba de software tiene limitantes, tanto teóricos como prácticos. Desde el punto de vista teórico, la prueba es un problema que llamamos no-decidible; esto implica, grosso modo, que no podemos escribir un programa que pruebe los programas sin intervención humana. Sin embargo, como

mencionábamos anteriormente, la prueba sí es automatizable en muchos aspectos.

Desde el punto de vista práctico, la cantidad de posibilidades para probar exhaustivamente un sistema es sencillamente inmanejable; es necesario entonces utilizar técnicas adecuadas para maximizar la cantidad de fallas importantes encontradas con los recursos asignados. Cada método que se utilice para detectar defectos deja un residuo de defectos más sutiles contra los cuales ese método es ineficaz (la llamada “Paradoja del Pesticida”).

La prueba de software implica pues, la aplicación de técnicas y herramientas apropiadas en el marco de un proceso bien definido, determinado por el tipo de proyectos de desarrollo de software que se abordan. En el siguiente número veremos con mayor detalle las principales técnicas de prueba de software. ©

- Luis Vinicio León

### Referencias

1. [www.e-quallity.net](http://www.e-quallity.net) pestaña Definiciones - Conceptos
2. Pressman, R.: Ingeniería de Software. Un Enfoque práctico. McGraw-Hill; 1993
3. Kit, E.: Software Testing in the Real World. ACM Press, 1995

**Transformamos la Calidad en TI**

Servicios de Consultoría, Capacitación y Evaluación, con experiencia y reconocimiento a nivel mundial en la implantación de prácticas innovadoras de Administración de Proyectos e Ingeniería de Software para la Industria de Tecnologías de Información.

[www.avantare.com](http://www.avantare.com)  
Tel. +52(55) 55-44-33-21

Procesos  
Liderazgo  
Tecnología  
Serite  
Calidad

**Avantare**

CMM®  
PMBOK®  
ISO9000  
CMMI™