

## Casi todas las pruebas del software

Elena Raja Prado

Quality Assurance & Regulatory Affairs, NTE S.A  
eraja@nte.es

### Resumen

Un breve recorrido por todo el proceso de desarrollo software en donde en cada una de las fases se apuntarán datos, problemas, herramientas, metodologías, recomendaciones o casos de la experiencia real referentes a las pruebas del software.

### 1. EL ANTES

Cierto es que un desarrollo podrá estar marcado por características variables como sus dimensiones, tecnología, sector o criticidad, pero todos ellos comparten una característica común, y es que lo desarrollan personas, las personas cometen errores y las personas cambian de ideas, por eso hay que probar y volver a probar.

Iniciemos un recorrido por todo el proceso seguido en el desarrollo de un software, desde que se concibe como idea hasta su puesta en marcha.

En la mayoría de los casos, un proyecto de desarrollo software se inicia con la aceptación de una oferta; y es ya en este punto, incluso antes de empezar, en donde deberemos tener en consideración las pruebas del software, y por lo tanto en donde en muchos casos nos encontramos, podríamos decir, con uno de los principales problemas: la dirección.

Se considera imprescindible que dirección dé apoyo a los procesos de calidad, es decir, que asuma los costes iniciales y exista convencimiento de su retorno, lo cuál se verá reflejado, aunque de un modo implícito, en la mencionada oferta.

Pero, ¿cómo podemos convencer la primera vez? Será muy probable que nos encontremos en la difícil situación de no poder aportar datos 'internos' que lo demuestren o justifiquen y nos

veremos obligados a recurrir a la 'literatura', a casos de éxito de otras compañías, a estudios, tesis, estadísticas, etcétera.

Probablemente nos aferremos a la 'famosa' estadística del coste de corrección un defecto según la fase en la que se detecta, apostaremos por el valor competitivo a conseguir, nos apoyaremos en la posibilidad de certificaciones para un reconocimiento que además favorecerá nuestra imagen, explicaremos los costes de la no-calidad y argumentaremos lo mejor que podamos el extenso material del que disponemos y que nos ofrece la literatura y en el mejor de los casos la experiencia.

- La mayor parte de defectos se concentran en las fases tempranas del proceso de desarrollo y el costo de corrección aumenta a medida que permanece no detectado.
- El coste relativo de corrección de un error se multiplica x1 en la fase de Requisitos, x3-6 en la fase de Diseño, x10 en la de Código, x50 en la de Pruebas y hasta por varios cientos en la de Explotación. [1]
- La calidad de un producto está influenciada por la calidad del proceso de producción. [5]
- Disponemos de estándares que agrupan las mejores prácticas, y que incluso disponemos de modelos de evaluación y mejora específicos del proceso de pruebas, como por ejemplo: Acercamiento a la gestión de pruebas (Test Management Approach, TMap); Modelo de Mejora del Proceso de test (Test Process Improvement, TPI); Modelo de Madurez del Test (Test Maturity Model, TMM)
- Existen datos sobre el retorno de la inversión (Return Of Investment, ROI) de diferentes compañías internacionales tras la aplicación de modelos y estándares. [14]

- Equilibrio del triángulo Coste-Calidad-Plazos.

En ausencia del compromiso de dirección, las pruebas pueden correr su peor suerte, y esto es, quedar excluidas del presupuesto, planificación de proyecto y su consecuente asignación de recursos.

## 2. LA PLANIFICACIÓN

Si por suerte y gracias a los múltiples argumentos a favor, continuamos con el desarrollo de nuestro producto considerando la importancia de las pruebas, en la siguiente fase de planificación nos plantearemos preguntas como ¿Qué niveles de pruebas aplicaré? ¿Con qué técnicas? ¿Cuándo deberé parar de probar? ¿Automatizaré? ¿Haré uso de alguna herramienta? ¿Criterios de re-test?,... a las que deberemos dar respuesta en el plan de pruebas a definir.

El propósito del plan de pruebas es explicitar el alcance, enfoque, recursos requeridos, calendario, responsables y la gestión de riesgos de un proceso de pruebas.

Para la definición del plan de pruebas, deberemos valorar:

- El alcance de la aplicación
- La complejidad de sus procesos
- Plataforma/s en las que deberemos probar
- Conocimientos y formación de quienes ejecutarán las pruebas.
- Normativas legales aplicables.

Deberemos tener en cuenta que:

- Las pruebas deben estar presentes a lo largo de todo el ciclo de vida del desarrollo (la correspondencia entre las fases de desarrollo y tipos de prueba se conoce como el llamado "modelo en V").
- El coste medio se encuentra entre un 30% y un 50% del coste total del proyecto.
- Siempre hay errores.
- Probar exhaustivamente el software es imposible.
- No es recomendable que el programador pruebe sus propios programas.

- Podemos disponer de herramientas.

Y finalmente, no deberemos olvidar la importancia de actualización del plan de pruebas para reflejar los cambios que se produzcan en los requisitos y/o proceso de desarrollo de nuestro producto.

## 3. LOS REQUISITOS

En paralelo a la planificación iniciamos la importante actividad de especificación de Requisitos [8], en dónde se revisa que cada uno de ellos debe cumplir, entre otras, la característica de ser verificable (que se pueda probar).

No siempre se asocia la revisión de un documento como una prueba en si misma, sin embargo si lo es, y es en esta actividad en donde gana su principal relevancia [11]. La revisión de los requisitos nos permitirá detectar errores en tan temprana fase, evitando así su propagación e incremento del coste.

Además, deberemos de empezar a preocuparnos por asegurar que todos ellos queden cubiertos por las pruebas, entramos en el concepto de trazabilidad.

Es frecuente la queja sobre lo costoso y tedioso que resulta la actividad de realizar la gestión de requisitos y en especial la de mantener la trazabilidad, ¿nos podría ayudar una herramienta?

Pensar ahora, en los cambios de requisitos que tienen lugar en cualquier momento del desarrollo... ¿cómo trataremos las pruebas de estos cambios?

Si nos decantamos por el uso de una herramienta será importante considerar las funcionalidades que ésta nos aporte, donde usualmente se destacan como básicas: la posibilidad de importación/exportación a formatos 'agradables', el control de peticiones de cambios, matrices de trazabilidad e histórico de cambios; se valoran otras como: la posibilidad de definir atributos, las que nos ayudan a la comunicación y organización del proyecto e incluso aquellas funcionalidades que aún formando parte de otras actividades del proceso de desarrollo se proporcionan de un modo

integrado en la propia herramienta de gestión de requisitos.

Es importante realizar una selección de la herramienta apropiada, en el mercado disponemos de un amplio listado, sin embargo las conclusiones de una evaluación realizada sobre cuarenta de ellas resultó en que tan sólo un 20% eran recomendables como posibles candidatas, y donde aproximadamente la mitad de las cuales coincidían con las herramientas líderes en el mercado.

Otro 10% eran herramientas gratuitas, de las cuales cabe decir que están muy alejadas de las comerciales pero que sin embargo podemos valorar en función de las características de nuestro proyecto: número de requisitos, grado de estabilidad, organización del equipo, etc.

#### 4. LA ARQUITECTURA Y EL DISEÑO

Si por fortuna seguimos las mejores prácticas en nuestro proceso de desarrollo, tras las especificaciones pasaremos al diseño de la arquitectura y en el mejor de los casos al diseño detallado ¿alguien dijo que la revisión no era una prueba?

#### 5. LA CODIFICACIÓN

Y llegamos a la fase de codificación, ¿qué pruebas hacemos? Seguro que como mínimo obtenemos una de éstas dos respuestas: revisiones de código y test unitario (pruebas de caja blanca)

Con estas pruebas obtendremos el beneficio de la detección de errores de manera temprana y es importante considerar que con ellas podremos detectar determinados errores que con otra clase de pruebas nos resultaría muy difícil, además de de permitirnos conseguir un código de mayor calidad y que cumpla con los estándares, aspectos que sin duda resultarán en un mejor mantenimiento. Otro aspecto positivo a valorar es que a su vez se generará un intercambio del conocimiento y las técnicas de codificación entre nuestros profesionales.

A la hora de decidir las pruebas a realizar, deberemos considerar los aspectos de nuestro código en cuanto a su complejidad, estabilidad,

estructuración e incluso si se trata de código nuevo, en mantenimiento o heredado.

Resultará muy útil conocer determinadas medidas que nos proporcionan ‘las herramientas de análisis de código estático’ y que nos ayudarán a valorar la calidad de nuestro código, así como el obtener medidas de ‘cobertura’ de ejecución que nos permitan comprobar la efectividad de las pruebas.

#### 6. LAS PRUEBAS

Y finalmente llegamos a la comúnmente llamada fase de pruebas [9], en donde encontramos diferentes niveles y tipos de prueba,...definémoslas:

*Pruebas de Integración:* Se comprueba la compatibilidad y funcionalidad de los interfaces entre las distintas ‘partes’ que componen un sistema, estas ‘partes’ pueden ser módulos, aplicaciones individuales, aplicaciones cliente/servidor, etc. Este tipo de pruebas es especialmente relevante en aplicaciones distribuidas.

*Pruebas de Validación:* Son las pruebas realizadas sobre un software completamente integrado para evaluar el cumplimiento con los requisitos especificados.

*Pruebas de Sistema:* el software ya validado se integra con el resto del sistema donde algunos tipos de pruebas a considerar son:

- Rendimiento: determinan los tiempos de respuesta, el espacio que ocupa el módulo en disco o en memoria, el flujo de datos que genera a través de un canal de comunicaciones, etc.
- Resistencia: determinan hasta donde puede soportar el programa determinadas condiciones extremas.
- Robustez: determinan la capacidad del programa para soportar entradas incorrectas.
- Seguridad: se determinan los niveles de permiso de usuarios, las operaciones de acceso al sistema y acceso a datos.

- Usabilidad: se determina la calidad de la experiencia de un usuario en la forma en la que éste interactúa con el sistema, se considera la facilidad de uso y el grado de satisfacción del usuario.
- Instalación: se determinan las operaciones de arranque y actualización del software.

*Pruebas de Aceptación:* Son las que hará el cliente, se determina que el sistema cumple con lo deseado y se obtiene la conformidad del cliente.

## 7. EL DESPUÉS

Hemos ahorrado tiempo y el cliente está más satisfecho! La versión definitiva del producto tras las pruebas de aceptación final del usuario se entregó antes que en procesos de desarrollo anteriores en los que el ‘aluvión’ de defectos tras la entrega final provocó que el cliente aún teniendo el software en sus manos no lo considerara válido hasta mucho más tarde, además de mostrar su descontento, claro.

Y aún nos quedará poder demostrar con datos los beneficios obtenidos y calcular el retorno de la inversión realizada, ¿pero cómo? con ‘métricas’, pero ese ya es tema suficiente para la realización de otro taller completo y muy recomendable.

## Referencias

- [1] Bohem Barry, Software Engineering Economics. Englenwood Cliffs, NJ:Prentice-Hall, Inc. 1981
- [2] Bohem Basili, Software Management, IEEE Computer, January 2001
- [3] Cem Kaner, James Bach, Bret Pettichord, Lessons Learned in Software Testing: A Context-driven Approach”, Wiley 2002
- [4] Juran’s Quality Control Handbook, McGraw-Hill, 1988, 4<sup>th</sup> Ed
- [5] Knox S.T., Modeling the Cost of Software Quality, Digital Tech. J., vol. 5, no. 4, 1993, pp. 9–16.
- [6] Koomen T., Pol M., Test Process Improvement: a Practical Step-by-Step Guide to Structured Testing, Addison Wesley Longman, 1999
- [7] McCabe, T.J., Structured Testing; A Software testing Methodology Using the Cyclomatic Complexity Metric, , NBS Special Publication 500-99, 1992
- [8] Wiegers Karl, Software Requirements: Practical techniques for gathering and managing requirements throughout the product development cycle. E,Microsoft Press, 2003.
- [9] Wiley John and Sons, The Art of Software Testing, G. Myers, 2004
- [10] ESA PSS-05-0 Issue 2 February 1991, ESA Software Engineering Standards
- [11] IEEE Std. 1028-1997, Standard for Software Reviews
- [12] ISO/IEC 12207, Software Life Cycle Processes
- [13] SPICE – ISO/IEC STD. 15504 , Software Process Improvement and Capability Determination
- [14] [www.klocwork.com/company/downloads/ROI-Executive.pdf](http://www.klocwork.com/company/downloads/ROI-Executive.pdf), The ROI from Software Quality, An Executive Briefing, Khaled El Emam
- [15] [www.esi.es](http://www.esi.es), European Software Institute
- [16] [www.ise.gmu.edu](http://www.ise.gmu.edu) Department of Information and Software Engineering,
- [17] [www.aemes.org](http://www.aemes.org) Asociación española de Métricas de Sistemas Informáticos,
- [18] [www.calidaddelsoftware.com](http://www.calidaddelsoftware.com), Calidad del Software