



*Business Consulting Services*

# **Testeo y aseguramiento de la calidad**

Procesos clave para mantenerse competitivo

[www.ermesconsulting.com](http://www.ermesconsulting.com)

## Testeo y aseguramiento de la calidad

Procesos clave para mantenerse competitivo.

**T**odos los logros empiezan con un primer paso. La industria del desarrollo del software nació siendo quizás una actividad complementaria a la fabricación de productos electrónicos, y era considerada una actividad “artesanal”, donde la calidad tomaba la identidad del programador o del reducido equipo de programadores. Los equipos realmente grandes eran una utopía en esos días, y hablar de un proceso de fabricación signado por estándares era solo un argumento de ciencia ficción. Sin ir mas lejos, las incipientes *software factories* de los '80 (Microsoft, Apple, Lotus, Digital Research) contaban con equipos de 10 o 15 “artesanos” para el desarrollo de productos como el *Lotus 123*, *Word*, *D-Base*, *MS-DOS*<sup>1</sup>. Eran bien conocidas las batallas por los “artesanos” que libraron estas compañías, llegando inclusive a hacer todo lo imaginable para tentar a profesionales de aquella época, que habían participado en proyectos “pioneros”, como el del laboratorio de Xerox en California, donde se intentaba por primera vez hacer un sistema operativo con el paradigma de “oficina virtual” y utilizando un dispositivo apuntador (fuente de inspiración para Microsoft y Apple en sus respectivos sistemas de escritorio).

La industria de la fabricación de software *ha evolucionado de una manera asombrosa los últimos años*. Hoy, los equipos de desarrollo de algunos proyectos, han crecido de unos pocos, a cientos de programadores y de una sola ubicación física, a trabajar coordinados con personas ubicadas a miles de kilómetros. Está claro que *hoy es impensable trabajar como se hacía en los primeros días*. De la misma manera, la industria sigue evolucionando de manera vertiginosa, y a pesar de que las grandes necesitaron implementar políticas, metodologías y hasta la formali-

---

<sup>1</sup> Todas son marcas registradas de sus respectivas compañías.

zación de muchos de los procesos de fabricación (por una cuestión básica de supervivencia), **en muchas factorías parece persistir la idea del software artesanal.**

Sin embargo, y como efecto consecuente de la “globalización”, muchas de nuestras compañías vieron invadidas sus arenas, por empresas extranjeras con un proceso de fabricación ya maduro y con políticas y definiciones bien firmes desde la comercialización hasta el soporte de post-venta; hecho que obligó a muchos a madurar desde la óptica artesanal a la industrial.

La buena noticia es que los procesos de fabricación de software, siguen en constante revisión en todo el mundo, y en ese análisis siempre existe un elemento de gran importancia: **la CALIDAD.** Todos hablamos de ella, pero ... ¿Qué es la calidad? ¿Cuán importante es? ¿Qué es asegurar la calidad? ¿Cuál es su relación con los *tests* en el proceso de fabricación? Si *tests* y *calidad* tienen una estrecha relación ... ¿Qué, cuándo y cuáles *tests* son recomendables?

## ¿Qué es la Calidad?

De acuerdo a las normas ISO 8402-1986, “CALIDAD es la totalidad de aspectos y características de un producto o servicio que se sustentan en su capacidad de cumplir las necesidades especificadas o implícitas.”

*Philip B. Crosby*, autor del famoso libro: “Quality is Free” (1979), nos enseña:

- La definición de Calidad: *cumplimiento con los requerimientos.*
- Sistema para alcanzar la Calidad: *prevención, no cura.*
- Medida del éxito: *el costo de la calidad.*
- La meta del proceso de calidad: *Sin defectos – hágalo bien la primera vez-.*

Calidad es la medida de algo. Calidad es una **métrica.** La “cosa” que mide es la **Excelencia.** En nuestro caso ¿*Cuánta excelencia posee este sistema?*

Las características intrínsecas que podríamos medir en cuanto a calidad, pueden ser las siguientes:

- **Facilidad de mantenimiento.** *¿Puedo corregirlo?*
- **Flexibilidad.** *¿Puedo cambiarlo?*
- **Facilidad de prueba.** *¿Puedo probarlo?*
- **Corrección.** *¿Hace lo que se necesita que haga?*

La *Administración de la Calidad*, es un área de conocimiento de todo proyecto, que incluye tres fases bien identificadas: la **Planificación**, el **Aseguramiento** y el **Control**.

El proceso de **Planificación de la Calidad** en todo proyecto, toma como entradas las *Políticas de Calidad de la compañía*, el informe de *Alcance del proyecto*, la *descripción del sistema* (producto/s), *normas, regulaciones y estándares* que deben aplicarse, para producir como salidas un **Plan de Dirección de Calidad**, definiciones operativas y check-lists utilizadas por los procesos de *Aseguramiento y Control de Calidad*.

Para ello se utilizan como herramientas el **Análisis de costo-beneficio**, **diagramas de flujo**, los **diagramas de causa-efecto** (*Ishikawa*), o el **Benchmarking**. Esta fase es la mas importante en cuanto a calidad se refiere, porque **se identifican posibles cuellos de botella, duplicación del trabajo o problemas de seguridad**, y se ofrecen soluciones dentro del *Plan de Calidad*.

El Control de la Calidad comprende el seguimiento de los resultados específicos del proyecto para determinar si cumplen con las normas de calidad, e identificar la forma de eliminar los resultados insatisfactorios.

En cuanto al Aseguramiento de la calidad (QA), “... es el conjunto de actividades sistemáticas, desarrolladas dentro del Sistema de Calidad, para garantizar que el proyecto va a satisfacer las Normas de Calidad.” [PMBOK-2000]

Por lo tanto SQA o *Software Quality Assurance*, es el proceso de asegurar la calidad, aplicado al software, que debe realizarse a lo largo de todos los procesos de fabricación: desde el análisis de requerimientos hasta la puesta en producción.

## ¿Cuál es la diferencia entre SQA y la actividad de testing?

De acuerdo a las definiciones del Instituto de Estándares Americanos (ANSI) y el Instituto de Ingenieros Electrónicos (IEEE), dos instituciones que han regido los estándares en varias industrias, incluyendo la del software, las diferencias se resumen en que:

- *TESTING* significa “control de calidad”.
- El Control de calidad mide la calidad de un producto o servicio.
- El Aseguramiento de la calidad mide la calidad de los procesos utilizados para la creación de un producto o servicio.

El *testing* es una herramienta de control para examinar la calidad del software. En varias compañías con una metodología de desarrollo ya madura, los *testers* son responsables por el aseguramiento de la calidad del software. Son contadas las que tienen un staff dedicado a QA. La razón por la que esto sucede es sim-

ple: los métodos tradicionales de aseguramiento de la calidad son demasiado caros para agregar valor a un producto.

En un reporte publicado por la consultora *Software Productivity Research* hace algunos años, se mide la *performance* de los 4 métodos de eliminación de errores utilizados más comúnmente en la industria: **Diseño Formal e Inspección de Código, Aseguramiento formal de la calidad, y Testing formal.** El resultado se muestra en el infograma de la figura 1.

### Efectividad en la eliminación de errores

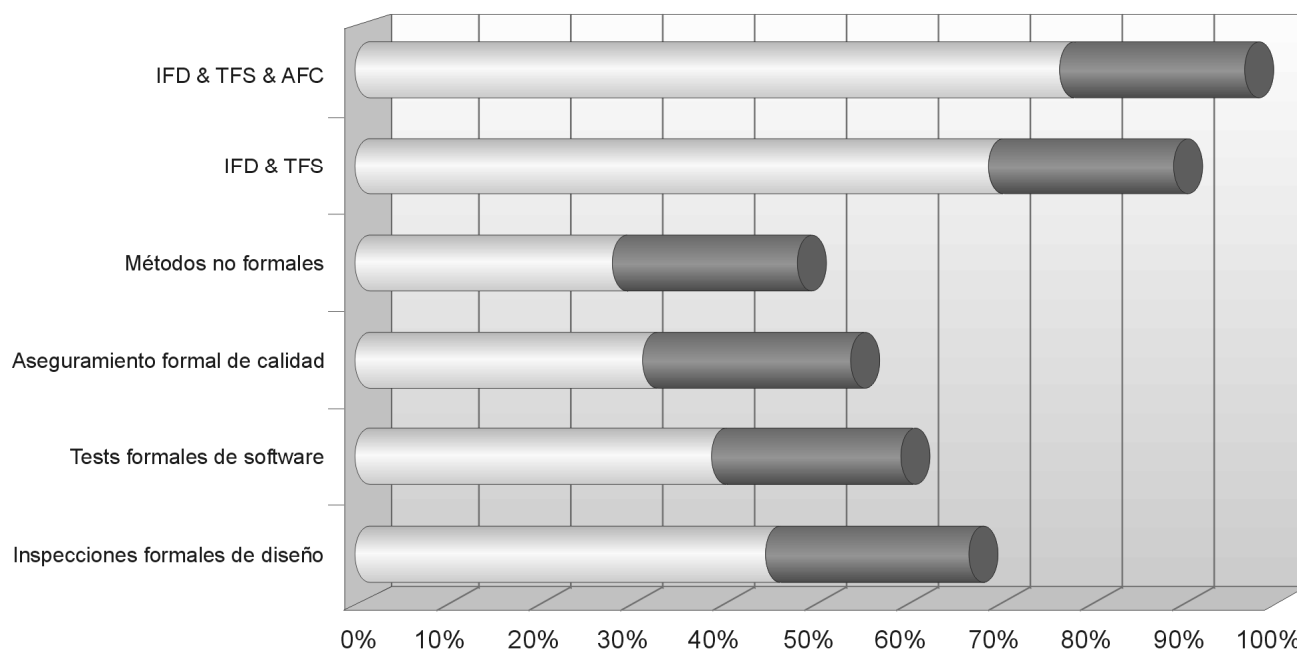


Figura 1: Efectividad de los distintos métodos en la eliminación de errores.

El por qué el Aseguramiento formal de la calidad por sí solo, no es tan efectivo como debiera, se debe a que el QA tradicional se basa en una serie de preceptos que no encajan en la realidad del software comercial de hoy.

### El *testing* en la actualidad.

En una de mis primeras experiencias como desarrollador, y en una etapa madura de cierto módulo que estaba probando, quien era mi jefe, en alusión a mi trabajo, me preguntó: “¿ya está probado? ¿está listo para ir a producción?”, a lo que respondí “Si, está listo.”. El añadió “Bien. ¿y qué se estuvo probando?”. Mi respuesta fue simple: “Y ...todo, ya está probado todo.”.

La experiencia me ha enseñado a nunca más responder de esa manera tan imprecisa. En la actualidad, no siempre se cuenta con un jefe tan permisivo para aceptar una respuesta tan vaga, y menos si uno tiene una tarea tan importante como la de detectar los defectos en un módulo que tiene una misión significativa para uno o muchos usuarios.

Está claro que esta respuesta para un *tester* en otra industria, sería inaceptable. Con solo aplicar la misma anécdota al sector de la Ingeniería Aeronáutica sería un desastre. Imagino que el ingeniero en jefe, formula la misma pregunta al diseñador del tablero de comandos de un Boeing 737, al concluir la etapa de pruebas, y acepta una respuesta del mismo tipo. Sabiendo eso, ¿qué voluntario se atrevería a volarlo cargado de un centenar de personas? Entonces, ¿por qué la industria del software debiera ser diferente?

Hoy, sin embargo, esperaríamos una respuesta con un grado mayor de detalle. Por ejemplo:

*“Ya se efectuaron el 80% de las pruebas planificadas. Las más importantes ya concluyeron, de acuerdo con nuestro previo **análisis de riesgos**. La **tasa de errores** y la **severidad** de los que se encontraron, están dentro del rango esperado, y aproximadamente un 75% de los errores fueron corregidos. Han pasado dos semanas hasta que encontramos un **problema de prioridad alta**. Actualmente no hay inconvenientes importantes abiertos, todos han sido solucionados. Los de prioridad media fueron sometidos a tests de regresión y han sido aprobados hace una semana. Ahora el equipo está efectuando pruebas adicionales en tres módulos nuevos con los que esperamos terminar para el final de la semana. Por lo demás, el sistema está estable.*

*Por otro lado, se detectó en las pruebas de carga y stress que el sistema falla cuando está al 90% de su capacidad. Los ingenieros creen entender el problema pero de acuerdo a las estimaciones que efectuaron, les tomaría cerca de dos semanas para implementar una corrección. Las proyecciones dan que el sistema debería estar por lo general al 60% de su capacidad, pero si la carga se eleva al 90% el sistema fallará. Nuestra recomendación es que sigamos de acuerdo al cronograma actual, con la premisa de que tenemos esa exposición a este riesgo, hasta tanto los ingenieros implementen la solución.”*

Lo más increíble es que este tipo de respuestas no abundan en la industria. Partiendo de la base que las tareas de pruebas no son reconocidas formalmente en muchas compañías (donde se supone erróneamente que el programador es el responsable por la totalidad del sano funcionamiento del módulo que desarrolla), existe muy poca educación formal en métodos de pruebas y como consecuencia, muy pocos proyectos implementan un sistema formal de pruebas con un responsable visible.

## Sin especificación no hay *test* posible

Otro problema de la industria es que nadie puede hacer un test sin una especificación. En la industria del software, la palabra **test** es menos entendida que la palabra **calidad**.

El IEEE define Testing como “el proceso de analizar un módulo para detectar diferencias entre las condiciones existentes y las exigidas (esto es, bugs o errores) y evaluar las características del mismo.”. Tampoco es posible correr un test automático si no hay un estándar para la respuesta esperada. Un test automatizado no puede hacer juicios subjetivos sobre la validez de las respuestas, debe tener un estándar sobre las respuestas esperadas para determinar si pasa o falla.

## La solución: mejorar el proceso de calidad.

La solución que propone *Marnie L. Hutchenson* en su libro “*The Most Important Test Method (MIT)*” es la redefinición de algunos conceptos genéricos para su adaptación a la industria.

Así, lo que *Crosby* definió de una manera, quedaría plasmado de esta otra mas pragmática y adaptada a la industria:

- **Definición de calidad:** *satisfacción del cliente.*
- **El sistema para conseguir calidad:** *el refinamiento constante.*
- **La medida de calidad:** *la rentabilidad.*
- **La meta del proceso de calidad:** *un hito a la vez.*

Y con este replanteo de las bases, la fórmula para crear un producto excelente se basa en las siguientes premisas:

1. *Sea el primero en el mercado con el producto.*
2. *Responda con el precio correcto.*
3. *Brinde las características correctas el en producto. Las requeridas y un poco mas, realmente harán clientes mas satisfechos.*
4. *Mantenga los errores inaceptables al mínimo absoluto. Asegúrese que los errores son menos caros e irritantes que los de su competidor.*

Históricamente, a medida que el mercado madure, la importancia de ser el primero disminuirá y la confiabilidad se volverá mas importante. Las fuerzas del mercado eliminarán a los competidores que no provean la suficiente **calidad**.

## Pautas para un buen test de calidad

Para diseñar y ejecutar un Plan de Tests con probabilidades de éxito, deberíamos hacernos las siguientes preguntas:

1. **¿Qué creemos conocer sobre el proyecto?** Los conocimientos correctos los encontraremos si escribimos las presunciones sobre el proyecto, es decir, lo que se supone que el software debería hacer.
2. **¿Qué tan grande es el esfuerzo total en las pruebas? ¿Cuántos testers necesitaremos?** Esto se basa no en lo que planeamos probar, sino en la identificación de las posibles pruebas.
3. **Si pudiéramos probar todo ¿qué deberíamos probar?** Aquí habría que ordenar los *tests* por prioridades y filtrar los menos importantes. Esta priorización tendrá como protagonista un adecuado análisis de los riesgos.
4. **¿Cuánto tiempo tomarán las pruebas?** Una vez que los tests han sido identificados, hay que estimar los tiempos. Esto incluye las restricciones de recursos, tiempos o costos que tenemos para las pruebas.
5. **Comenzar a construir los scripts de las pruebas.**
6. **Conducir las pruebas.** Utilizar indicadores de progreso para controlar el proceso.
7. **¿Qué tan exitoso está siendo el proceso de testing? ¿Tuvo una cobertura adecuada? ¿Fue adecuado el esfuerzo implicado?** Este es un buen momento para documentar las lecciones aprendidas acerca de la experiencia.

Aquí se utilizarán métricas para contestar estas preguntas y mejorar los esfuerzos para pruebas posteriores, o en otros proyectos. Es recomendable documentar los resultados pues este debería ser el punto de comienzo para el plan del próximo proyecto. Si el esfuerzo fue conducido de una manera metódica y reproducible, tiene chances de ser duplicado y mejorado la próxima vez y de esta manera, **contribuir a la maduración de una metodología para un desarrollo competitivo.**

# Glosario

***Diagramas de Causa-efecto o de Ishikawa:*** Diagramas con forma de cola de pescado utilizados para analizar las causas y consecuencias de los defectos en el diseño de procesos, productos o servicios..

***Benchmarking:*** Consiste en comparar las prácticas de los proyectos reales o la concepción de otros proyectos con el fin de generar ideas y definir una norma con la cual medir la realización del proyecto en cuestión.

# Bibliografía

*“Quality – Management for Projects and Programs” - (1979) Juran, Joseph*

*“Project Management Body of Knowledge” – (2004) Project Management Institute*

*“Object Oriented Software Testing” - Siegel*

*“Object Oriented Software Engineering” – Ivar Jacobson*

*“The Most Important Tests Method” - Marnie L. Hutcheson*

## En la web:

<http://www.qacity.com>

*Recursos para testers ocupados.*

[comp.software.testing](http://comp.software.testing)

*Foro de noticias sobre la actividad de pruebas de software.*

<http://satc.gsfc.nasa.gov/assure/agbsec3.txt>

*Manual de políticas y procedimientos, capítulo de aseguramiento de la calidad para el desarrollo de software en la NASA.*

**Autor:** Ernesto Sebastián Melgin

**Correo electrónico:** sebastian.melgin@ermesconsulting.com

*Investigador, docente, analista de sistemas y especialista en dirección de proyectos egresado de la Universidad Tecnológica Nacional, con 18 años de experiencia en la industria del software. Participó como diseñador, analista, desarrollador y Project Manager en proyectos de software en diversas industrias, en nuestro país y en el extranjero. Actualmente se desempeña como consultor*

*de la firma Melgin & Asociados y está preparando su certificación como PMP del Project Management Institute).*