

Las Revisiones de Código, renovadas y más vigentes que nunca

DESDE LOS 70, LOS ESTRICITOS EXÁMENES REALIZADOS SOBRE EL CÓDIGO FUENTE HAN IDO MUTANDO EN LAS PRESTACIONES Y RESULTADOS. EN UN CONTEXTO DE ALTA TERCEORIZACIÓN COMO EL ACTUAL ESTA HERRAMIENTA DE CONTROL DE CALIDAD PERMITE MEDIR LAS ACCIONES EN EL LARGO PLAZO, DETECTAR DEFECTOS Y PLANTEAR MEJORAS. INTEGRACIÓN DE PROVEEDORES, DETECCIÓN DE ERRORES EN ETAPAS INCIPIENTES Y AHORROS CONSIDERABLES SON SUS PRINCIPALES VENTAJAS.

Las pirámides de Egipto -construidas hace más de 4000 años- tienen parámetros que las acercan casi a la perfección en materia de construcción, teniendo un error máximo en la alineación N-S, E-O de 6 minutos de arco y distando la base de algunas de ellas de ser un cuadrado perfecto en menos de 18 cm. El secreto fueron los métodos de inspección empleados durante su construcción.

Hoy en día no intentamos alinear gigantes bloques de piedra, sino que siendo protagonistas de la era digital, la solución a casi todos nuestros problemas incluye un software.

A medida que la Tecnología de la Información avanza, el desarrollo del software se torna una tarea cada vez más compleja, que en muchos casos ha sobrepasado a las empresas que lo construyen y mantienen. Por este motivo, al igual que en las pirámides de Egipto, es fundamental inspeccionar cada paso del proceso de construcción para obtener un software de alta calidad.

Definiciones iniciales

Una revisión de código consiste en el análisis estático del código fuente de una aplicación. Esto significa que no se revisa el flujo de la ejecución de los programas, sino la adherencia a estándares o buenas prácticas utilizados en la codificación.

Definidas inicialmente por Fagan, a principio de los 70 para IBM, estos estrictos exámenes realizados sobre el código fuente emulaban en aquellos tiempos los procesos de inspección realizados sobre los componentes de hardware. Por entonces, se buscaba optimizar el tiempo de acceso al compilador – un recurso caro en esos tiempos – para que cuando el código llegara a compilación, fuera lo más correcto posible. Nos separan muchos años desde aquellas inspecciones de software iniciales. Hoy el compilador ya no es un recurso caro sino casi invisible. Pero nuevas formas de revisión han cobrado renovado valor en el contexto actual de desarrollo y mantenimiento de software -fuertemente

tercerizado-, en el cual no basta solamente con probar el software, sino que se hace crucial garantizar ciertos atributos que prevengan su posible deterioro. ¿Podemos estar tranquilos de que tras tres años de mantenimiento tercerizado nuestras aplicaciones siguen siendo correctas respecto de las mejores prácticas de construcción? Ese es el desafío al que se enfrentan nuestras organizaciones hoy. Y la revisión de código actual acerca una respuesta.

Modernizando el proceso

El proceso de revisión se basa en la definición de un conjunto de reglas, las cuales se verifican sobre el código a revisar, para finalmente obtener una lista de potenciales defectos.

Dependiendo del tamaño de la aplicación esto puede llegar a ser una tarea muy costosa, por lo que es conveniente contar con un proceso optimizado para realizar esta tarea de forma eficiente.

Para optimizar el proceso de revisión de código, es fundamental tener definido un método que permita realizarlo de forma sistemática y que no tenga una fuerte dependencia de la persona que realiza la revisión.

Esto se puede lograr contando con un proceso dividido en etapas con objetivos concretos y apoyándose en la utilización de herramientas de búsqueda automática de reglas.

Una revisión de código se divide principalmente en 4 etapas:



El proceso comienza con la planificación, durante la cual se define el alcance de la revisión, es decir los componentes de software que serán comprobados, y se programan las fechas del resto del proceso.

A continuación se realiza la definición de las reglas a analizar. Esta es la etapa que tiene mayor incidencia en el resultado del proceso, y consiste en la identificación y selección de buenas prácticas y estándares que se desean asegurar en el código construido por la organización. Esto, en muchas ocasiones no está definido de antemano, por lo que es posible que requiera un trabajo de análisis puntual.

Una vez definidas las reglas se puede comenzar con la ejecución. Esta etapa consiste en aplicar las pautas al código fuente estudiado, para lo cual se pueden emplear algunas herramientas automáticas que facilitan la identificación del código que no cumple con la norma. No obstante, luego se requiere un análisis manual, tanto para descartar falsos positivos (excep-

ciones a las reglas), como para clasificar los resultados en función de su tipo y criticidad.

Finalmente la revisión termina con un reporte de resultados que incluye los potenciales defectos encontrados en la etapa anterior y un detalle con las conclusiones y recomendaciones para mejorar la calidad del software revisado.

Los beneficios clásicos de usar revisiones

La ganancia más directa que se obtiene mediante la realización de revisiones de código es la integración y la educación implícita de los desarrolladores, nivelando la calidad del código que construyen todos los integrantes del equipo u organización. Esto, a su vez, reduce significativamente los tiempos de preparación y capacitación de los nuevos desarrolladores.

Por otra parte, las revisiones permiten garantizar el nivel de calidad del software entregado por proveedores, asegurando que estos cumplan con los estándares y prácticas definidas al momento de solicitar el desarrollo.

Como consecuencia de la revisión se obtiene un conjunto de resultados, los cuales brindan un diagnóstico del código revisado. Dentro de los hallazgos típicos se pueden encontrar:

- Errores potenciales.
- Incumplimiento a estándares de programación.
- Malas prácticas de programación difundidas en la organización.
- Posibles fraudes.

Contar con esta información y poder tomar acciones al respecto, permite asegurar el nivel de calidad mínimo del software desarrollado, lo cual consiste en el principal beneficio que brindan las revisiones de código.

Impacto de los defectos del software en el costo

Son conocidas las frases: un defecto no detectado, que pasa a la próxima fase, puede costar de dos a diez veces más para descubrirlo y corregirlo o el costo de reparación de un error en el producto aumenta en un orden de magnitud cada vez que se avanza una etapa en su elaboración.

Esto da una sensación de que las revisiones de código podrían reducir el costo de la construcción de un software, pero no queda claro como se traduce esto en dinero. Para dar una idea más clara se puede utilizar un ejemplo práctico teniendo en cuenta datos de casos reales.

Es posible que en 5 minutos se pueda corregir un error detectado durante el desarrollo. Mientras que corregir el mismo error detectado durante el testing funcional puede llegar a demandar 30 minutos (contando el tiempo requerido para reproducirlo y repararlo). En el peor de los casos, si el error es detectado por un usuario final, corregirlo puede demorar hasta una hora (sumando el tiempo que se requiere del usuario, del funcional y del desarrollador).



POR GABRIEL NAIMAN
PRAGMA CONSULTORES.

testimonios

Suponiendo que se produce una cantidad fija de errores (sin importar en que fase se corrijan) se puede llegar al siguiente cálculo de costo de los errores:

Fase	Costo (Minutos)	Errores	Total (Minutos)
Construcción	5	50	250
Test funcional	30	100	3000
Productivo	60	20	1200
TOTAL			4450

Fase	Costo (Minutos)	Errores	Total (Minutos)
Construcción	5	0	0
Test funcional	30	150	4500
Productivo	60	20	1200
TOTAL			5750

El resultado de este ejemplo es un ahorro de 1250 minutos en el costo del software.

No obstante, el ejemplo anterior es un caso muy simplificado, ya que un error cuando pasa a la siguiente etapa es muy probable que produzca al menos dos errores. En el siguiente ejemplo se puede ver este impacto, partiendo desde el mismo inicio.

Fase	Costo (Minutos)	Errores	Total (Minutos)
Construcción	5	0	0
Test funcional	30	200	6000
Productivo	60	20	1200
TOTAL			7200

El resultado de este ejemplo es un ahorro de 2750 minutos en el costo del software.

Si bien estos casos son simplificaciones de la realidad, muestran como la detección de un defecto en las primeras fases del proyecto reduce significativamente el costo del mismo. Sin duda, realizar estas revisiones tiene un costo para el equipo de desarrollo y por ende para la organización. Sin embargo, a partir de los resultados expuestos se podría inferir que este costo es menor que el de corregir los errores en fases posteriores.

Revisión y tercerización de software: una combinación ganadora

En el contexto actual, la mayoría de las corporaciones han tercerizado gran parte de sus actividades de mantenimiento de software, así como sus proyectos de transformación e integración de sistemas. Algunos de los problemas que suelen repetirse en diferentes empresas, incluso de diferentes industrias, son:

- El proveedor no necesariamente tiene preocupación por mantener y adherir a estándares.
- Los desarrollos llave en mano no se integran fácilmente a los manten-

imientos, por corresponder a proveedores diferentes.

- Un cambio de proveedor implica cambios de formas de trabajo.
- ¿Cómo eliminar la cuestión personal de una discusión respecto de la aprobación de entregables o hitos?

En este sentido, una revisión semiautomática puede ayudar, permitiendo definir acuerdos de nivel de servicio que sean fácilmente verificables. De esta forma, se quita toda subjetividad respecto de la calidad del software y se reduce la dependencia con los proveedores.

Cuándo y cómo

A la hora de incluir las revisiones en el proceso de desarrollo hay que decidir cuál es el objetivo buscado. Siendo un poco simplistas se puede decir que hay dos posibilidades: asegurar la calidad del código antes de ser liberado o hacer un diagnóstico de una aplicación productiva.

En el primero de los casos, es importante identificar cual es el momento dentro del proceso de construcción de la aplicación donde tiene menor costo realizar la revisión. Normalmente el momento más apropiado es antes de realizar las pruebas funcionales, de esta forma el desarrollador tiene que pasar el código generado por la revisión para luego poder liberarlo.

Como en todo proceso, cada etapa que se agrega genera un retraso, por este motivo es fundamental en estos casos realizar revisiones muy optimizadas. De esta forma, en este tipo de revisiones es necesario contar con una herramienta que automatice al máximo la detección de defectos y que el revisor verifique manualmente la menor cantidad posible de resultados.

Otra consideración importante a tener en cuenta es que para hacer más eficiente el proceso de revisión es conveniente agrupar la mayor cantidad de código junto, por lo que es necesario realizar las revisiones en bandas horarias predefinidas.

Para las revisiones de diagnóstico no es tan crítico el tiempo de ejecución, ya que no frena la liberación de la aplicación, por lo que en estos casos la revisión manual suele ser bastante más exhaustiva que en el caso anterior, así como los informes de resultados cuentan con mayor análisis sobre los defectos encontrados y recomendaciones mucho más específicas. Este tipo de revisiones suelen realizarse cuando la aplicación no funciona adecuadamente o cuando se desean implementar nuevos estándares y se necesita realizar un análisis de impacto.

Ponerlo en práctica

La tercerización de tareas de desarrollo es una necesidad a la que todas las organizaciones actuales suscriben, por lo que la independencia de los proveedores y el control de calidad sobre estos es algo en lo hay que ponerse a trabajar cuanto antes.

En este contexto las revisiones de código son una herramienta fundamental, que sumado a otras actividades como el testing funcional, permiten medir de forma cuantitativa la calidad del software, de forma tal que se puedan definir acuerdos de servicio precisos y verificables.

No obstante, las revisiones tienen un costo, el cual puede ser elevado si no se realizan de forma eficiente y sistemática, por lo que al igual que para el testing funcional, deben realizarse con el adecuado acompañamiento de profesionales expertos en proceso y metodología asociados.



Marcelo Dalceggio
(Gerente Corporativo de Calidad, Seguridad Informática & Software Factory de Cencosud)

¿Qué valor agregaron las revisiones a las actividades de QA que ya realizaban?

La revisión de código agregó mucho valor ya que no solo permitió demostrar cualitativamente sino cuantitativamente el impacto de los errores en el negocio. En muchas actividades de aseguramiento de calidad que ejecutamos regularmente se nos hace difícil demostrar con un número en dinero cuál es el beneficio (sobre todo con aquellas revisiones de pares que se realizan al comienzo del ciclo de vida de un proyecto). En el caso

de estas inspecciones, los defectos que se encuentran están directamente relacionados con el incumplimiento a un estándar o con una falla que podría ocurrir en el entorno productivo (algunas ya conocidas porque han ocurrido y otras porque tenemos registro del esfuerzo de corrección que han demandado incidentes similares en el pasado). La revisión además nos ha permitido detectar en forma mas económica los defectos recurrentes, acortar los ciclos de testing (sobre todo aquellos de performance) y mejorar la mantenibilidad del código. Por otro lado, generó mayor concentración en la actividad de programación y nos evitó la propagación de defectos producto de la técnica de construcción del "copy & paste". Finalmente nos facilitó tener un diagnóstico rápido del estado de salud del código que heredamos de otras compañías o de desarrollos encargados a terceros.

¿Qué recomendaría a un gerente que apunta a introducir revisiones en su cadena de QA?

Personalmente recomendaría varios puntos:

- 1) Asignar a los desarrolladores de mayor skill para participar del proceso de definición de las "reglas de inspección" (éste es el activo mas importante).
- 2) Buscar que el proceso de inspección pueda ser automatizado lo máximo posible (esto permite ganar en volumen y poder dedicar el esfuerzo disponible de los revisores a los incidentes mas complejos o a nuevas situaciones).
- 3) Asegurar que el proceso de revisión arroje la menor cantidad de "falsos positivos" para darle mayor credibilidad al proceso.
- 4) Medir y tomar decisiones con los resultados (si da lo mismo tener código saludable que no tenerlo, el proceso pierde fuerza)
- 5) Colocar este proceso dentro de un círculo de mejora continua (todos los días se presentan nuevos hallazgos). No sirve quedarse con la foto inicial.
- 6) Ejecutarlo oportunamente (en nuestro caso, antes del unit test). Cuanto más temprano, mayor ahorro.
- 7) Finalmente, y para que todo funcione armoniosamente, recalcar que lo que se está revisando es software y no a los profesionales (de manera que nadie lo sienta como algo "personal").

En nuestra compañía, las inspecciones están implementadas como un servicio tercerizado. Luego de más de un año de trabajo conjunto estamos plenamente satisfechos con los beneficios del servicio como del equipo de Pragma que lo soporta.

"A fines del 2008 comenzamos a incorporar al Proceso de Testing actividades de revisión de código. Esto se sumó a los controles de calidad de software que realizamos desde hace varios años, siempre ejecutadas por un tercero especializado. Esta práctica de Aseguramiento de la Calidad del Software nos permite detectar posibles defectos de manera temprana y reducir los costos de duplicar el trabajo y de la corrección. A diferencia de las pruebas de control funcionales o pruebas de estrés, la revisión de código tiene como beneficio la posibilidad de realizarlo en forma remota y sin infraestructura compleja ya que no requiere de ambientes de testing donde ejecutarse: el equipo profesional simplemente debe poder acceder al código fuente. Como estrategia elegimos para el análisis un módulo de una de las aplicaciones core de la empresa. A continuación acordamos -entre el área de Arquitectura de software de Nextel, el equipo de Testing de Pragma y el proveedor encargado del desarrollo- cuales serían los puntos primordiales de la revisión y así definir un conjunto acotado de reglas. Esto tiene por objetivo focalizarse sobre un reducido grupo de problemas para que el equipo de desarrollo pueda incorporar este aprendizaje. Luego, una vez aprendido ese segmento sí, agregar un nuevo conjunto de reglas a verificar en las sucesivas revisiones. En este sentido es fundamental que el equipo de desarrollo tenga una participación activa en este proceso, pues la eficiencia con la que logre incorporar los resultados de cada revisión a las nuevas etapas de producción de código, será lo que realmente convierta esta práctica en efectiva".



Marcelo Girotti
(Supervisor Área Gestión de Proyectos, de Nextel Communications Argentina S.A.)