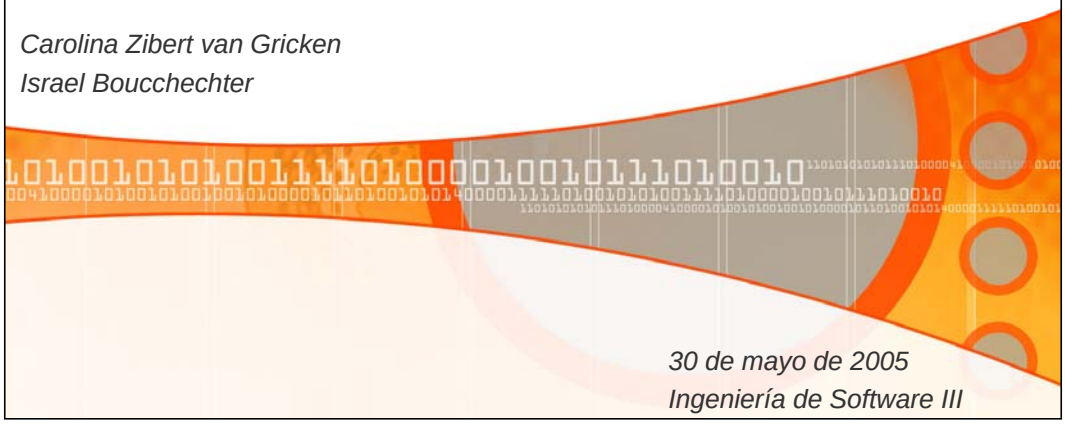
Pruebas de Confiabilidad

Carolina Zibert van Gricken
Israel Boucchechter



30 de mayo de 2005
Ingeniería de Software III

¿QUÉ ES CONFIABILIDAD?

La confiabilidad del software se refiere a la precisión con la que una aplicación proporciona, sin errores, los servicios que se establecieron en las especificaciones originales. El diseño para favorecer la confiabilidad, además de referirse al tiempo de funcionamiento de la aplicación antes de que se produzca algún error, está relacionado también con la consecución de resultados correctos y con el control de la detección de errores y de la recuperación para evitar que se produzcan errores.

Se producen errores en la aplicación por distintos motivos:

- Comprobación inadecuada
- Problemas relacionados con cambios en la administración
- Falta de control y análisis continuados
- Errores en las operaciones
- Código poco consistente
- Ausencia de procesos de diseño de software de calidad
- Interacción con aplicaciones o servicios externos
- Condiciones de funcionamiento distintas (cambios en el nivel de uso, sobrecargas máximas)
- Sucesos inusuales (errores de seguridad, desbordamientos en la difusión)
- Errores de hardware (discos, controladores, dispositivos de red, servidores, fuentes de alimentación, memoria, CPU).
- Problemas de entorno (red eléctrica, refrigeración, incendios, inundaciones, polvo, catástrofes naturales)

PROCEDIMIENTOS RECOMENDADOS PARA LA CONFIABILIDAD

El objetivo de conseguir un software de calidad abarca todo el ciclo vital de desarrollo del programa. Se recomiendan los siguientes procedimientos para crear aplicaciones confiables:

1. Pensar en la confiabilidad:

   Las aplicaciones confiables han de ser compatibles con operaciones confiables y necesitan también procesos de implementación confiables. Céntrese en el modo en que se proporciona el servicio y busque posibles problemas allí donde las alternativas de diseño o de procedimiento permitan reducir las causas de error.

2. Invertir en personal:

   El personal de operaciones y los programadores deben conocer a fondo las prácticas de administración del ciclo de vida y de la arquitectura, poniendo especial atención en la prevención de los errores más comunes.

   Cree una referencia cultural de equipo en la que la confiabilidad sea un aspecto crítico. Proporcione educación sobre los procedimientos de la compañía, las herramientas de programación, las tecnologías de aplicación y los conceptos de confiabilidad.

3. Eliminar los puntos con errores desde el diseño de aplicaciones:

   Un sistema confiable es más fácil de mejorar que un sistema no confiable (con eventos de error ocultos y distribuidos por todo el programa), que es muy costoso cambiar.

4. Utilizar un sistema operativo consistente
5. Proporcionar supervisión de confiabilidad continua

   Todas las aplicaciones críticas para una misión deben proporcionar datos de supervisión. El análisis de los datos recopilados desempeña un papel importante en la observación del estado, de los problemas actuales y de las tendencias de largo alcance de las aplicaciones críticas.

6. Invertir en procesos de diseño de software de calidad:
   - *Utilizar una metodología de ciclo vital de desarrollo*
   - *Uso de revisiones de código y estándares de codificación*
   - *Desarrollo de procedimientos de recuperación*
   - *Uso de procedimientos de control de cambios probados*
7. Utilizar pruebas inteligentes

   Los procesos de prueba de control de calidad deberán proporcionar una respuesta a tres cuestiones importantes:

   ¿Están correctamente implementadas en la aplicación las funciones descritas en las especificaciones?

   ¿Satisface la aplicación las situaciones de usuario previstas sin producir errores?

   ¿Se ajusta el perfil de confiabilidad de la aplicación a los requisitos originales o los supera?

   Cuando el nivel de calidad y confiabilidad no sea aceptable, deberá corregirse el software hasta que se alcance el nivel deseado.

8. Implementar cambios con cautela
9. Prestar atención al presupuesto

¿Qué son Pruebas de Confiabilidad?

Consiste en probar una aplicación para descubrir y eliminar errores antes de que se implemente el sistema.

Como existen infinidad de combinaciones distintas a lo largo de una aplicación, no es muy probable que se encuentren todos los errores de una aplicación compleja.

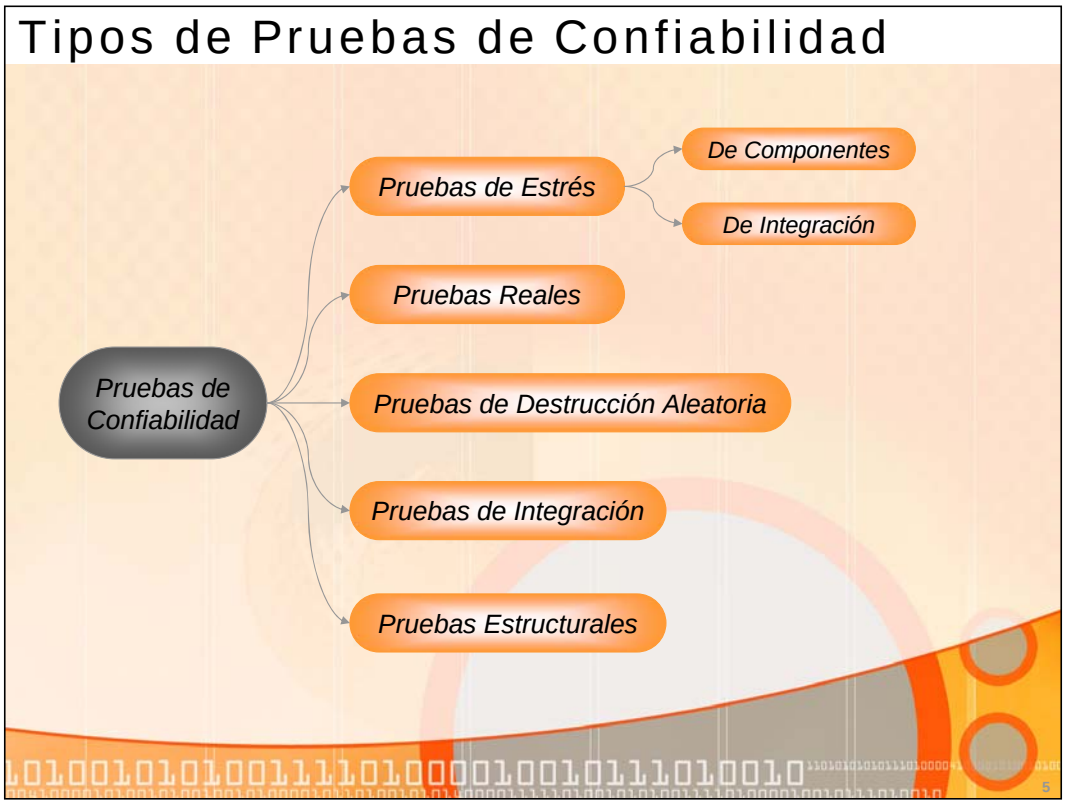
Pero puede probar las situaciones más probables bajo condiciones normales de uso.

Con tiempo suficiente se pueden realizar pruebas más complicadas para detectar defectos menos evidentes.

¿QUÉ SON PRUEBAS DE CONFIABILIDAD?

La comprobación de la confiabilidad consiste en probar una aplicación para descubrir y eliminar errores antes de que se implemente el sistema. Puesto que hay infinidad de combinaciones distintas de recorridos alternativos a lo largo de una aplicación, no es muy probable que encuentre todos los errores posibles de una aplicación compleja. No obstante, puede probar las situaciones más probables bajo condiciones normales de uso y confirmar que la aplicación proporciona el servicio previsto. Si dispone de tiempo suficiente, puede realizar pruebas más complicadas para detectar defectos menos evidentes.

Tipos de Pruebas de Confiabilidad



Pruebas de Estrés

Consisten en la simulación de grandes cargas de trabajo para observar de qué forma se comporta la aplicación ante situaciones de uso intenso

De Componentes

La idea es forzar cada componente de forma aislada más de lo que la aplicación podría experimentar en condiciones normales.

Por ejemplo: usar un bucle de 1 a 10.000.000 lo más rápidamente posible y observar si hay problemas evidentes.

De Integración

Están relacionadas con las interacciones con otras estructuras de datos, procesos y servicios tanto de los componentes internos y externos de la aplicación.

Es necesario conocer los recorridos codificados y las situaciones a las que se enfrenta el usuario y que se identifiquen todas las maneras en las que el usuario se mueve por la aplicación.

Pruebas de estrés

Las pruebas de estrés consisten en la simulación de grandes cargas de trabajo para observar de qué forma se comporta la aplicación ante situaciones de uso intenso.

Pruebas de estrés de componentes

Con las pruebas de estrés de los componentes, se aíslan los servicios y componentes que conforman el sistema, se infieren los métodos de navegación, de funcionamiento y de interfaz de estos servicios y componentes y se crea un cliente de prueba que llame a dichos métodos.

Para aquellos métodos que tienen acceso a un servidor de base de datos o a cualquier otro componente, puede crear un cliente que proporcione datos simulados en el formato previsto.

El equipo de prueba inserta datos simulados una y otra vez mientras observa los resultados.

La idea es forzar cada componente de forma aislada más de lo que la aplicación podría experimentar en condiciones normales. Utilice, por ejemplo, un bucle de 1 – 10.000.000 lo más rápidamente posible y observe si hay problemas evidentes. La comprobación de cada DLL ayuda a identificar errores más importantes con el componente.

Pruebas de estrés de integración

Después de forzar cada componente individual, deberá someter a una situación de estrés a toda la aplicación con todos sus componentes y servicios.

Las pruebas de estrés de integración están íntimamente relacionadas con las interacciones con otras estructuras de datos, procesos y servicios tanto de los componentes internos como de otros servicios externos de la aplicación.

Las pruebas de integración comienzan con una comprobación básica del funcionamiento. Es necesario que conozca los recorridos codificados y las situaciones a las que se enfrentan los usuarios, que comprenda lo que intentan hacer estos y que identifique todas las maneras en las que el usuario se mueve por la aplicación.

Las secuencias de comandos de prueba deberán probar la aplicación de acuerdo con el uso previsto.

Por ejemplo, si la aplicación muestra una página Web que un 99% de los clientes simplemente visitará y en la que sólo un 1% comprará realmente, tiene sentido proporcionar secuencias de comandos de prueba que fuercen la búsqueda y las distintas funciones de exploración. Por supuesto, la cesta de la compra también debe comprobarse, pero el uso previsto sugiere que la mayoría de las pruebas deberían centrarse en las funciones de búsqueda.

Intente prolongar siempre la duración de las pruebas, tanto como se lo permitan el calendario y el presupuesto. En lugar de realizar pruebas durante unos cuantos días o una semana, prolongue el período de pruebas a un mes, un trimestre o un año y observe cómo funciona la aplicación durante un período de tiempo más largo.

Pruebas reales

El software que es confiable de forma aislada en un entorno de prueba protegido puede no serlo en la implementación real. Mientras que las pruebas aisladas son útiles en los primeros procesos de prueba de confiabilidad, un entorno de prueba real garantiza que las aplicaciones simultáneas no interfieren entre sí. En dichas pruebas es frecuente detectar interacciones con otras aplicaciones que producen errores imprevistos.

Deberá asegurarse de que la nueva aplicación puede ejecutarse en el espacio del servidor, con la configuración final, y con plena experiencia en el perfil de eventos del cliente previsto. El plan de pruebas debe incluir la ejecución de la nueva aplicación en el entorno de destino final o en un entorno lo más cercano posible al entorno de destino. Esto último puede realizarse normalmente con una replicación parcial o compartiendo precavidamente el entorno final.

Pruebas de destrucción aleatorias

Una de las formas más sencillas de probar la confiabilidad es utilizar datos de entrada aleatorios. Este tipo de pruebas intenta por todos los medios bloquear la aplicación o que ésta produzca errores; para ello, se proporcionan datos ilógicos y falsos. Los datos de entrada pueden ser eventos del mouse (ratón) o del teclado, secuencias de mensajes del programa, páginas Web, cachés de datos o cualquier otra condición de entrada que pueda introducirse en la aplicación. Deberá utilizar pruebas de destrucción aleatorias para comprobar las rutas de errores importantes y poner de manifiesto errores de programación del software. Este tipo de pruebas mejora la calidad del código ya que da lugar a errores que permiten examinar el control de los errores devueltos.

Las pruebas aleatorias pasan por alto de forma intencionada cualquier especificación del comportamiento del programa. Si se interrumpe la aplicación, no se ha superado la prueba. Si no se interrumpe la aplicación, la prueba se ha superado. La cuestión aquí es que las pruebas aleatorias pueden tener un alto nivel de automatización porque nada tienen que ver con el modo en que se supone que funciona la aplicación subyacente.

Necesitará algún tipo de instrumento de prueba para realizar eventos de prueba caóticos, fuera de toda lógica y de gran estrés en la interfaz de la aplicación.

Pruebas de Integración

- *Su objetivo es:*
  - *Identificar errores introducidos por la combinación de programas probados unitariamente.*
  - *Verificar que las especificaciones de diseño sean alcanzadas.*

- *Los componentes no están implementados en el ambiente operativo.*
- *La fase de integración requiere mayor planificación y un conjunto de datos de prueba.*
- *Los sistemas grandes requieren varios pasos para realizar la integración.*
- *Existen tres tipos básicos de pruebas:*

1 Todo de una vez 2 Top-Down 3 Down-Top

Pruebas de Integración

Su objetivo es identificar errores introducidos por la combinación de programas probados unitariamente. Además Verificar que las especificaciones de diseño sean alcanzadas.

Componentes individuales son combinados con otros componentes para asegurar que la comunicación, enlaces y los datos compartidos ocurran apropiadamente.

No son verdaderamente pruebas de sistema porque los componentes no están implementados en el ambiente operativo. La fase de integración requiere mayor planificación y un conjunto de datos de prueba. Los sistemas grandes requieren varios pasos para realizar la integración.

Existen tres tipos básicos de pruebas:

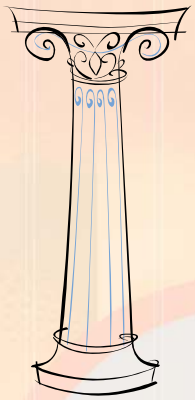
Todo de una vez: provee una solución útil para realizar la integración de problemas simples.

Down-Top: Se empieza con los módulos de nivel inferior, y se verifica que los módulos de nivel inferior llaman a los de nivel superior de manera correcta, con los parámetros correctos.

Top-Down: se empieza con los módulos de nivel superior, y se verifica que los módulos de nivel superior llaman a los de nivel inferior de manera correcta, con los parámetros correctos.

Pruebas Estructurales

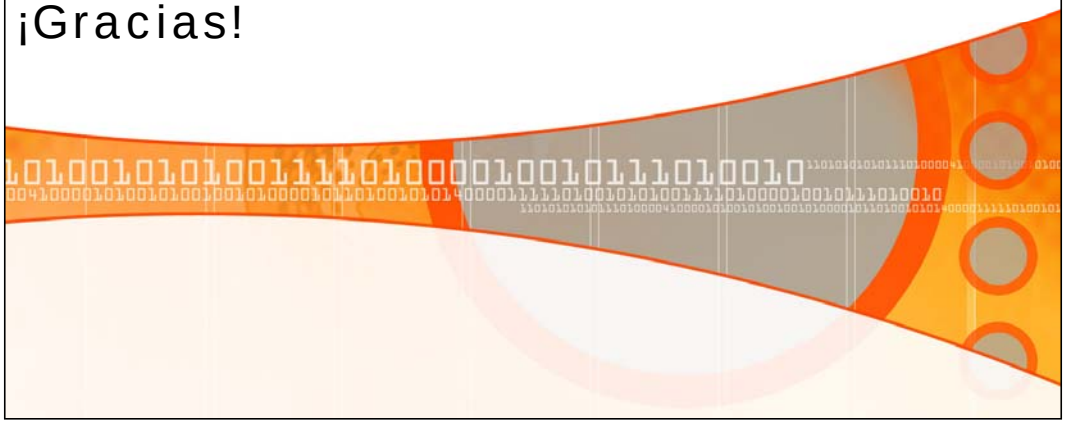
Son también conocidas como "pruebas de caja blanca" o "pruebas basadas en código", donde se enfocan en probar cada una de las estructuras de código, para que su comportamiento sea el esperado.



Son las pruebas donde se conoce la estructura interna del componente a probar, y se efectúa una prueba sobre dicha estructura.

En el caso de una aplicación web también se revisa la estructura interna de los links y otros elementos.

¡Gracias!



Referencias

- ***Probar la confiabilidad***
  http://office.microsoft.com/en-us/templates/CT011153611033.aspx
- ***Introducción a la confiabilidad***
  http://msdn.microsoft.com/library/SPA/vsent7/html/vxconreliabilityoverview.asp?frame=true
- ***Procedimientos recomendados para la confiabilidad***
  http://msdn.microsoft.com/library/SPA/vsent7/html/vxconbestpracticesforreliability.asp?frame=true
- ***Structural Testing Course***
  http://www.westfallteam.com/Software_structural_testing.htm

- *ABAD L., Jorge H.* ***Tipos de Pruebas de Software.*** *Abril 2005.*
  http://jorgeabad.e2uhosting.com/files/TIPOS-DE-PRUEBAS.pdf
- *DYCK, Simon y David SLOANE.* ***Software Testing****. Seng 621 Winter 1999.*
  http://sern.ucalgary.ca/~sdyck/courses/seng621/webdoc.html
- *RANKINE, Ian.* ***Introduction to Software Testing****. QES Inc.*
  http://www.qestest.com/principl.htm
- *WHITTAKER, James A.* ***What Is Software Testing? And Why Is It So Hard?*** *Florida Institute of Technology. IEEE SOFTWARE January / February 2000*
  http://www.computer.org/software/so2000/pdf/s1070.pdf
- *RIVEST, Raymond.* ***What is black box/white box testing?*** *June 15 2004*
  http://www.faqs.org/faqs/software-eng/testing-faq/section-13.html