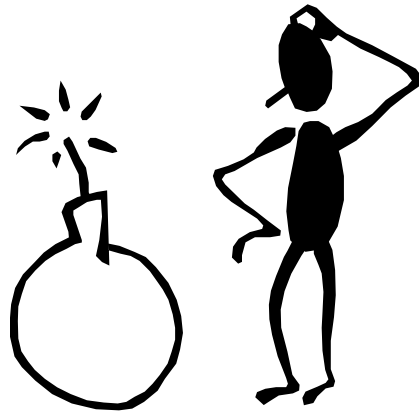


Quality Assurance for WEB developments

Ernesto Kiskurno
Pragma Consultores
QA Factory

ASSE 2001
Buenos Aires, Argentina



1

Objetivo

- Enumerar los **problemas más comunes** de los desarrollos en WEB
- Explicar el **impacto** que estos problemas tienen **en las actividades de aseguramiento de la calidad** en WEB
- Mostrar cómo pueden usarse las **técnicas tradicionales** de aseguramiento de la calidad sobre desarrollos WEB
 - Focalizar fuertemente en las actividades, tipos y proceso de **testing**



2

Audiencia

- Líderes o integrantes de equipos de desarrollo que trabajan sobre plataformas WEB
- Líderes o integrantes de equipos de testing y aseguramiento de la calidad sobre este tipo de ambientes

Temario

- Introducción a la problemática de los desarrollos WEB
- Impacto desde QA
- Qué hacer desde QA?
 - Testing
 - Revisiones
- Resumen

Por qué hay que testear los desarrollos WEB?

- Tener mayor calidad **incrementa la fidelidad** de nuestros clientes
 - Sólo el 2% de los visitantes se transforman en clientes (Boston Consulting Group)
 - El ratio es muy bajo como para empeorarlo con mala calidad
- Siempre **es más barato mantener** un cliente que conseguir uno nuevo (1:6)
- Cuánto perdemos cuando perdemos?
 - Según Forrester Research, por cada cliente que reconocemos como perdido en un website, debemos estimar que estamos perdiendo otros 20 (1:20)



5

Por qué hay que testear los desarrollos WEB?

- Las **malas experiencias** se extienden más rápidamente que las buenas
 - No solo perdemos clientes actuales, también hay pérdida de **clientes potenciales**
- El riesgo en WEB es alto: hay una **exposición inmediata a millones de usuarios**
- La facilidad de cambio es alta, la competencia está a un "click" de distancia
 - BN o Amazon?
 - Yahoo o Hotmail?
 - Clarín.com o LaNación on-line?



6

Por qué hay que testear los desarrollos WEB?

- Un desarrollo WEB, es un desarrollo
 - Pensar en las razones que ya conocemos para software tradicional
- En el área de sistemas
 - Hay ahorro de costos: Las aplicaciones web tienen un mantenimiento continuo, se actualizan y cambian diariamente
 - Hay beneficios intangibles
 - Mejora del proceso de desarrollo; los programas se escriben con menos errores
 - Disminución de tiempos de desarrollo



7

Por qué hay que testear los desarrollos WEB?

- $\text{Importancia}(\$) = \text{costo detección} + \text{frecuencia} * \text{costo corrección} + \text{costo instalación} + \text{costo consecuencias}$
- Consecuencias
 - Pérdida de clientes y consecuencias de negocio
 - Deterioro de relaciones internas (sistemas contra)
 - Atrasos en lo planificado
 - Etc, etc.



8

Por qué hay que testear los desarrollos WEB?

Indicador	Telecomunicaciones	Bancos	WebSites
Cantidad prom. de casos de prueba	29	100	166
Cantidad prom. de problemas	22	40	de 80 a 200
Horas x problema encontrado	6.4	8.4	de 1 a 2.4
Problemas encontrados x caso ejecutado	0.6	0.37	0.6

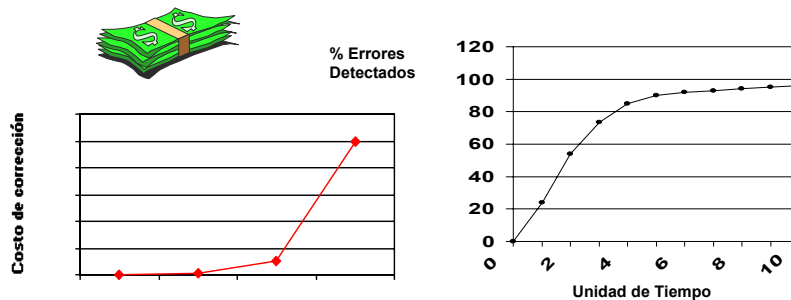
- ¿Es un problema la calidad en WEB?
 - IEEE: 4 horas x problema vs. WEB: 2 horas x problema
 - Según nuestras mediciones, el tiempo entre errores es
 - bancos = 8 vs. web = 2
 - Cuatro veces menos tiempo!!
- La calidad de los desarrollos web es menor!



9

La calidad es siempre una decisión económica

- Siempre es mejor prevenir que curar (recordar la ley del "x10")
- Pero todo tiene su límite ...hay que tener en mente la relación costo/beneficio



10

La calidad es siempre una decisión económica

- Ejemplo
 - Sitio de e-business con 120.000 visitantes y 2400 transacciones por mes a 50\$ por transacción
 - Supongamos que la tasa de conversión es de 2%
 - Supongamos que las actividades de calidad que practico mejoran 0.1% la tasa de conversión. Esto quiere decir que anualmente gano 1440 transacciones a 50\$
 - Tengo un presupuesto anual para calidad de 72.000\$ (por ejemplo esto es equivalente a dos personas con un sueldo de 3000\$ por mes)



11

La calidad es siempre una decisión económica

- Ejemplo (continuación)
 - Costo mínimo promedio de adquisición de un cliente = 38\$ (Forrester Research)
 - Costo de mantenimiento de un cliente = 6,34\$ (1/6)
 - Supongamos que un 10% de nuestros clientes está disconforme: 240 transacciones mensuales
 - Cuánto ahorro manteniendo clientes? $240 \times (38 - 6,34) = 7.598,4\$$ mensuales (o 91.180,8\$ anuales!)
 - Es el ahorro por no tener que reemplazar un cliente que ya tengo por uno nuevo
 - Por ejemplo, es equivalente a 3 personas con un sueldo mensual de 2500\$



12

La calidad es siempre una decisión económica

- Ejemplo (continuación)
 - Según nuestras mediciones, un tester profesional necesita entre 1 y 2.4 horas para encontrar un error. Supongamos 3 hs.
 - En un mes este tester encuentra $160 / 3 = 53$ problemas
 - Si invertimos en calidad 3 personas, estaríamos eliminando 159 problemas por mes
- Cuántos clientes mantendríamos haciendo calidad?
- **Cuánto puede/debe invertir en calidad?**



13

Por qué en WEB es más difícil?

- En testing tradicional se supone que hay varias cosas listas, por ejemplo
 - **Tiempo** para probar
 - **Documentación** a partir de la cual generar casos o entender el producto
 - Una **versión estable** del producto liberada por desarrollo

En testing de aplicaciones web no es posible exigirlos...

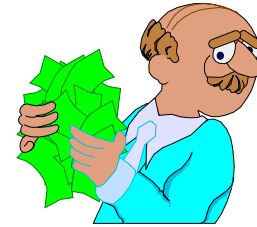


14

Revised on WEB.com 6/16/10

Por qué en WEB es más difícil?

- La volatilidad de los **requerimientos** es mucho mayor
 - Cambian diariamente
 - Concepto de "proyecto viviente"
 - El usuario muchas veces no sabe lo que quiere o subestima el esfuerzo de los cambios que propone
- El nivel de **documentación** es muy bajo
 - No hay tiempo
 - El cambio permanente en la aplicación, hace imposible tener la documentación actualizada
- En vista de esto, resulta muy **difícil conseguir un ambiente [estable]** en el cual probar la aplicación



Por qué en WEB es más difícil?

- Los **equipos de trabajo** son
 - Más jóvenes y menos experimentados
 - A veces distribuidos geográficamente
 - Multidisciplinarios (diseño, multimedia, contenidos)
 - Con menos respaldo metodológico (paradigma relativamente nuevo)



Por qué en WEB es más difícil?

- El **cambio tecnológico** es permanente
 - Esto no permite que el equipo de desarrollo madure los conocimientos
 - Las herramientas de soporte están menos estables
 - En un entorno bancario los desarrollos duran por años. A la larga, los errores se van corrigiendo...**en web todo se resuelve en días**



Qué impacto tiene esto en QA?

- Falta de **tiempo**
 - Cuánto tiempo se asigna a la calidad?
 - En particular, cuánto tiempo hay disponible para el testing?
 - Qué es lo prioritario?
 - Qué tipo de actividades de calidad hacemos?
- Alguien dijo por ahí "*Decidimos a la mañana, programamos a la tarde y probamos a la noche*"



Qué impacto tiene esto en QA?

- Falta de **documentación** y volatilidad de los **requerimientos**
 - Cómo se definen los casos de prueba?
 - Cómo sabemos que estamos construyendo el sistema...
 - correcto?
 - correctamente?
 - Preguntas típicas de un tester:
 - La funcionalidad es correcta?
 - La navegación es la que pidió el cliente?
 - Están todos los contenidos cargados?



Qué impacto tiene esto en QA?

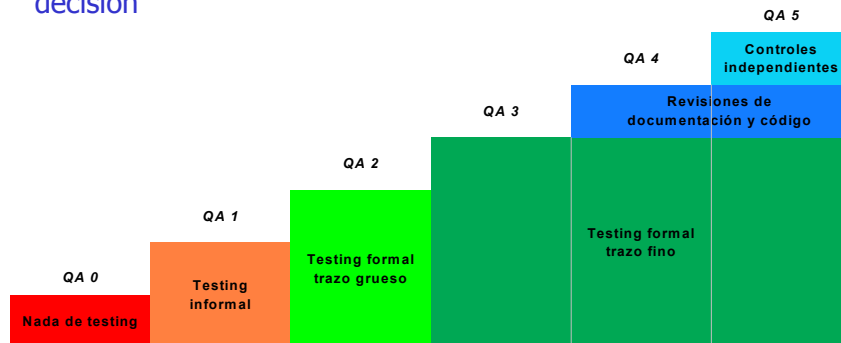
- Otros factores críticos
 - Cómo sabemos si el sistema soportará una **carga** mayor de usuarios?
 - Sabemos cuántos usuarios tendremos?
 - Sabemos el nivel de **seguridad** pretendido?
 - Sabemos en qué **plataformas** debe correr el sistema?
 - Browsers, versiones de sistemas operativos?
 - Cuán cómodo es el sistema para el cliente? (**usabilidad**)
 - Puede realizar una compra sin problemas?
 - Qué **tecnología** usamos?



Niveles de calidad

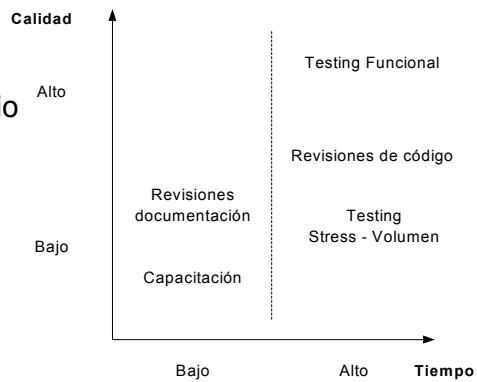
- Qué nivel de calidad necesita nuestro producto?

Depende de la relación costo / beneficio. Otra vez, es nuestra decisión



Qué actividades de QA hacer?

- Mejor relación costo-beneficio
 - Testing manual y automático
 - Revisión de código
 - Revisión de documentación
 - Capacitación
- Peor relación costo-beneficio
 - Inspección de código



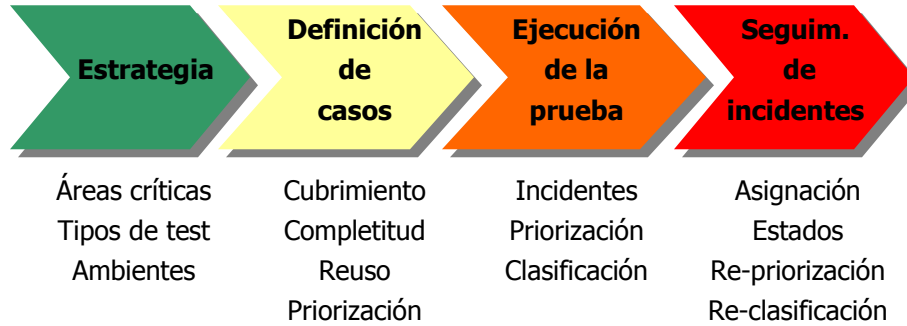
Qué actividades de QA hacer?

- Una posible estrategia sería
 - Comenzar por **Testing Funcional**
- Luego, incorporar otras técnicas
 - Revisiones de documentación
 - Revisiones de código
 - Testing de stress y volumen

Testing - Qué tipos de testing conviene hacer?

- Testing manual
 - Funcional
 - Usabilidad
 - Compatibilidad (entre browsers)
 - Contenidos
- Testing automático
 - Stress y volumen
 - De estructura (por ejemplo links o imágenes)
- Cuidado con la fiebre de la automatización!
 - Verificar el costo / beneficio de la utilización de herramientas de capture & playback
 - El esfuerzo de automatizar tests funcionales debe compensarse con la cantidad de veces que utilizaremos los scripts (se calculan necesarias más de 8 regresiones)

Testing manual - Proceso canónico



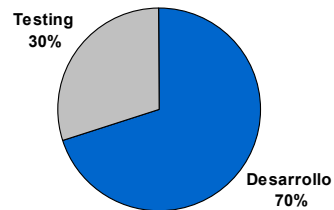
Este proceso es aplicable a múltiples tipos de testing (funcional, usabilidad, compatibilidad, contenidos).



27

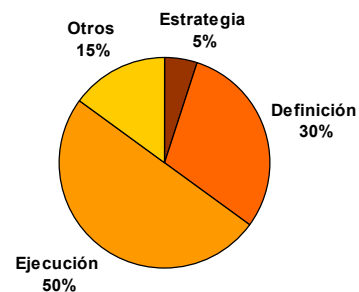
Testing manual - Cuánto tiempo dedicar?

- Según nuestra experiencia ...



No olvidar que el 30% de nada, es nada... La calidad exige una inversión en tiempo y dinero.

Otros incluye tareas de inducción a la aplicación, preparación de ambiente de testing y datos



28

Testing manual - Documentación

- Beneficios de documentar las pruebas
 - Mejora nuestras estimaciones futuras
 - Mejora la curva de aprendizaje de los testers juniors
 - Mejores decisiones
 - Salgo o no salgo a producción?
 - Desarrollo con el proveedor A o con el proveedor B?
 - Hago testing interno o tercerizo?
- Qué documentamos?
 - Los problemas, los casos de prueba, la estrategia de pruebas, las ejecuciones en general: tablero de control y seguimiento



29

Testing manual – Documentación, Tablero de Control

Pragma

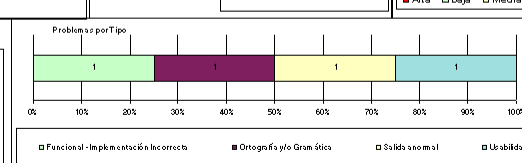
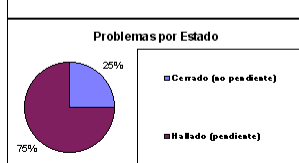
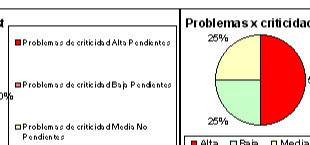
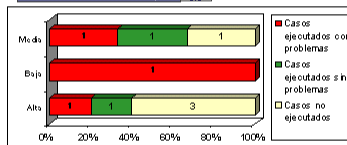
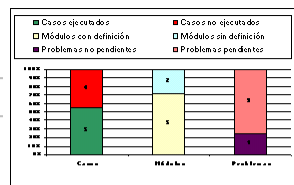
Portal e-ejemplo

Backend 1.0

Resumen Estadístico del Test

Cantidad de módulos definidos	7
Cantidad de casos definidos	3
Promedio de casos por módulo	1.285
Cantidad de problemas detectados	4
Promedio de problemas por módulo	0.571
Problemas detectados x caso ejecut	0.8

Iteración	1	2	3	4	5
Problemas Nuevos	3	1	0	0	0
Problemas Pendientes	2	1	0	0	0
Problemas Corregidos	0	1	0	0	0
Total	3	3	2	0	0



Martes, 07 de Agosto de 2001



30

Testing manual - Algunas claves

- Registrar los **problemas** encontrados
 - Para aprender de ellos
 - Para no volver a cometer los mismos errores
 - Para categorizarlos y priorizarlos antes de su resolución
- **Separar ambientes** y manejar configuraciones
 - Ensayando la implementación



31

Testing manual - Algunas claves

- Definir **casos de prueba durante el análisis** es una forma de documentar el producto
 - Es una forma de adelantarnos al problema
- Técnicas que todavía sirven:
 - Partición de dominio
 - Tablas de decisión



32

Testing manual - Algunas claves ... funcional

- Cuidados especiales durante estas pruebas
 - Validaciones en la entrada de datos
 - Búsquedas, siempre tienen problemas
 - Manejo dinámico de contenidos
 - Ciertos problemas de seguridad (logins y https)
 - Integración Frontend / Backend
- Testing independiente
 - El que desarrolla no prueba
 - Mayor experiencia y concentración
 - Nadie está motivado para encontrar sus propios errores



33

Testing manual - Algunas claves ... funcional

- Checklists, una herramienta muy potente
 - Pueden ser utilizados no sólo por testers sino también por desarrolladores y usuarios
 - Ahorran tiempo de testing
 - Son fáciles de entender y mantener
 - Deben crearse a partir de los problemas encontrados
- Ejemplos
 - Checklist de instalación en producción
 - Checklist de validación de página



34

s ... de Usabilidad

en la forma de uso

funcional introduciendo

ares, principalmente té



s ... de Usabilidad

en la forma de uso

- o **Preparación de pruebas**
 - Preparar pruebas preparadas
 - Encuentro de usuarios para la realización de tareas similares a las que se comprarán (algo) y p...
- o **Focus groups**
 - Para observar cómo se relaciona con la aplicación
 - Ayudan a decidir temas de usabilidad y contenidos



Testing manual - Nuevos nombres, viejas ideas...

- **Exploratory Testing**
 - Dicho en forma simple se trata de definir y ejecutar el testing al mismo tiempo (testing intuitivo)
- **Risk-based Testing**
 - Testing priorizando los componentes más críticos de la solución y los tipos de testing más críticos para la solución
- **Testing de contenidos**
 - Revisión de los contenidos del producto (ortografía y gramática)
- **Testing de compatibilidad**
 - Verificar que la aplicación funciona bajo las distintas plataformas posibles existentes en el mercado (browsers, SOs, etc)



37

Testing - Herramientas útiles

- **Reporte y administración de problemas**
 - Debemos tener una herramienta para reportar incidentes!
 - **Un sitio de tamaño mediano puede llegar a tener entre 150 y 200 incidentes.**
 - No necesitamos tener algo complejo, a veces con algo sencillo y bastante disciplina alcanza.



38

Testing - Herramientas útiles

- Administración de casos de prueba
 - Permiten reusar casos de prueba en
 - otras aplicaciones similares
 - regresiones posteriores sobre el mismo sitio
 - Esto puede disminuir considerablemente los tiempos de definición de casos de prueba
 - En WEB es muy común que se repitan secciones enteras. Por ejemplo foros, chats, etc.

Testing - Herramientas útiles

- De stress
 - Principales cuestiones a resolver en este tipo de testing
 - ¿Cuáles situaciones me interesa analizar?
 - ¿Cuáles operaciones quedan dentro de los alcances de la prueba?
 - ¿Cómo vamos a obtener los datos para realizar TODAS las operaciones deseadas?
 - ¿Cuáles usuarios vamos a utilizar?
 - ¿La prueba simula apropiadamente los distintos escenarios planteados?

Testing - Herramientas útiles

- De stress
 - Etapas del proceso canónico
 - Se identifican las transacciones, procesos del negocio, modelo de uso, etc.
 - Se capturan y se parametrizan las transacciones individuales (*herramienta*)
 - Se crean los distintos escenarios (*herramienta*)
 - Se ejecutan los escenarios de carga y se monitorea la performance (*herramienta*)
 - Se analizan y se interpretan los resultados (*herramienta*)
 - Con este tipo de herramientas la prueba es **repetible sin mayor esfuerzo**
 - El análisis de performance en sitios web es **crítico**
 - No se puede ir a producción y ver que pasa...

 Optimización / Tuning

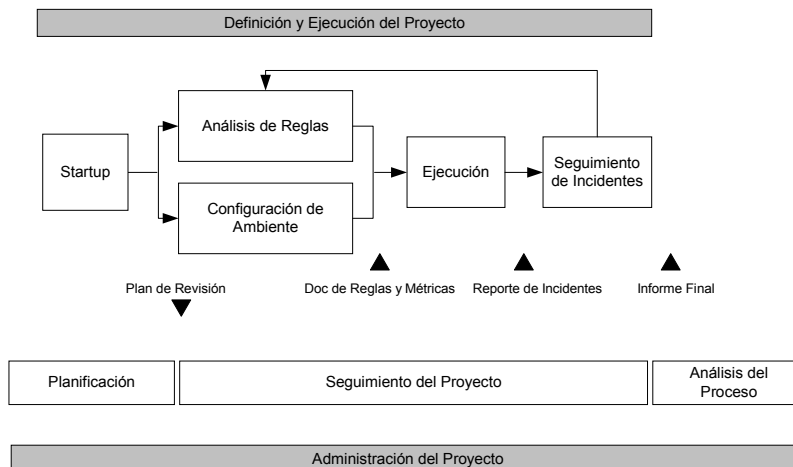
Testing - Herramientas útiles

- Análisis de websites "on-line"
 - Aprovechemos lo que nos da la naturaleza...
 - Hay herramientas gratuitas para chequear estructura
 - Links rotos, accesibilidad de imágenes y/o contenidos
 - Compatibilidad de código html estándar
 - Tiempo de carga de las páginas
 - Ciertos aspectos de compatibilidad entre browsers
 - No resuelven el problema, pero ayudan...
- Algunas urls ...
 - www.control-z.com/pages/swtestlinks.html
 - www.linkbot.com
 - www.netmechanic.com
 - SQA Site Check (www.rational.com)
 - www.mercuryinteractive.com

Revisiones de código

- Por qué hacer revisiones de código?
 - Búsqueda de defectos o errores
 - Obtener métricas de calidad del software
 - Búsqueda de violaciones a estándares de la Empresa
 - Descripción de componentes del código y relaciones; estructuras y características
- Se pueden hacer distintos tipos de revisión
 - Automática, Semi-Automática, Manual

Revisiones de código



Revisiones de código

- Cuales son las tareas típicas?
 - Planificación y StartUp
 - Análisis de Reglas
 - Configuración Ambiente
 - Ejecución
 - Seguimiento Y análisis de incidentes
- Y los entregables típicos?
 - Estrategia de Revisión
 - Plan y Reglas de Revisión
 - Reporte de Incidentes
 - Documentación Código



45

Revisiones de código

- Por qué son útiles en web?
 - Permite encontrar problemas de performance
 - Por ejemplo, queries o código ineficiente
 - Permite detectar no uso de estándares y estilos
 - Por ejemplo, los programadores están usando plantillas de estilo o hacen "hardcode"?



46

Revisiones de documentación

- Son muy valiosas pues permiten detectar problemas antes de la etapa de codificación
- Qué tipo de problemas se encuentran?
 - De requerimientos, de especificación o de diseño
 - Generalmente indefiniciones, inconsistencias, los “agujeros” en la funcionalidad
- Qué podemos revisar? Cualquier entregable en general ...
 - Requerimientos
 - Diseños
 - Casos de prueba
 - Planes de proyecto



47

Revisiones de documentación

- Quienes participarían?
 - Dependiendo del documento a revisar: analistas, desarrolladores, gente de QA, usuarios
- La mecánica...
 - Sesiones individuales de revisión del documento y luego reuniones generales para discutir las observaciones.
- Por ejemplo
 - En WEB es común hacer revisiones de los story boards generados para las discusiones con usuarios
 - También se revisan los contenidos antes de cargarse en cada una de las secciones del sitio



48

Resumen

- La calidad de las aplicaciones influye directamente sobre la percepción del cliente
 - Mantener vs adquirir clientes
 - El tema de los costos
- La calidad es una decisión económica
 - Debemos estimar cuánto dedicarnos a ella
- En web las cosas cambian
 - Hay nuevas reglas para el desarrollo de software
 - Hay que medir para entender mejor

Resumen

- Hacer QA para WEB tiene algunas complicaciones
 - El testing es una de las actividades más útiles en este contexto
 - También podemos hacer otras cosas: revisiones
- Testing
 - No solo funcional, de contenidos, de stress y volumen, de compatibilidad
 - Hay que estar informado:
 - Conceptos nuevos: risk-based testing, por ejemplo
 - Herramientas gratis
 - Técnicas nuevas
- Revisiones
 - Pueden ser una buena opción en algunos casos

Dudas? Preguntas?? Comentarios???

Ernesto Kizskurno
ekizskurno@pragma.com.ar

Pragma Consultores
www.pragma.com.ar

San Martín 575 – 4to Piso
(C1004AAK) Buenos Aires – Argentina
Tel. (+54 11) 4327 – 1999
email: pragma@pragma.com.ar



- Black Box Testing, Boris Beizer
- Exploring requirements - Quality before Design, Gause & Weinberg
- Testing automation snake oil, James Bach
- The web development changed the meaning of testing, Robert Glass
- What is Exploratory Testing?, James Bach
- Getting Web Development Projects Under Control, Wolfgang B. Strigel
- Much Does Site Performance Affect Your Bottom Line?, Eric Sovish

