

Las Pruebas del Software



Javier Tuya

Universidad de Oviedo

Dpto. de Informática

Grupo de Investigación en Ingeniería del Software


<http://www.di.uniovi.es/~tuya/>



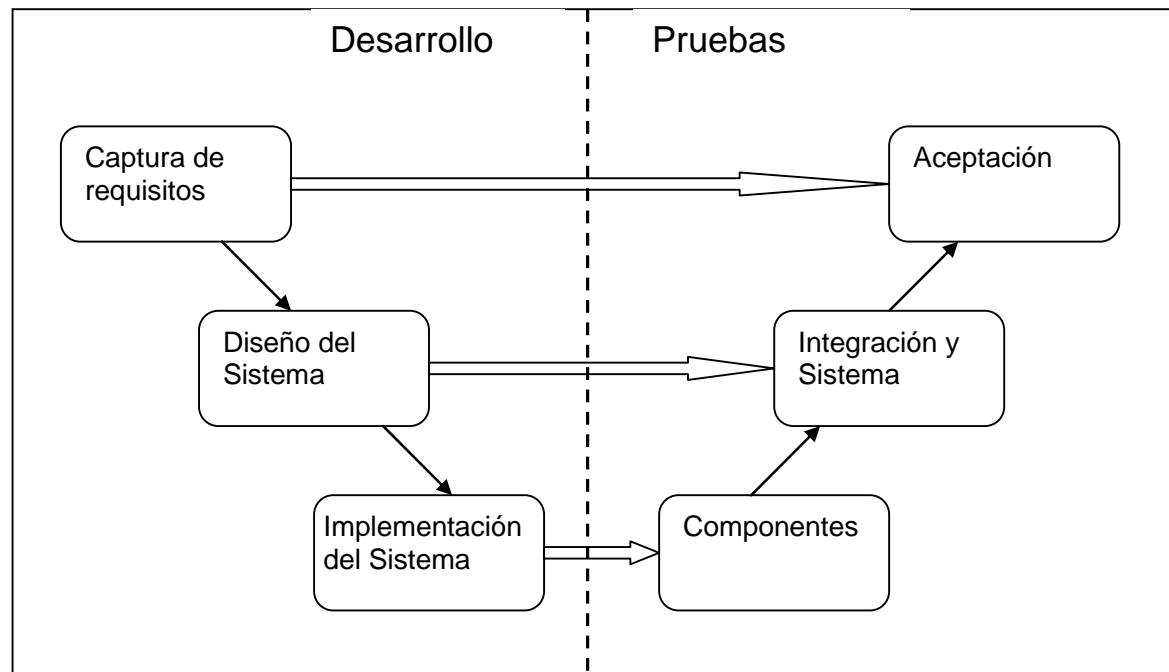
giis.uniovi.es

Universidad de Cádiz, (9 de Marzo de 2007)



- 
- “All in all, coders introduce bugs at the rate of 4.2 defects per hour of programming. If you crack the whip and force people to move more quickly, things get even worse”
 - W. Humphrey (note), <http://www.cs.usask.ca/grads/jpp960/490/BombSquad.html>
 - 20 por Día / 100 por Semana / 400 por Mes / 5000 por Año
 - “5000 Defect Project (not atypical for IBM)”
 - P. Gibson, Testing Challenges for IBM, UK Test 2005 Keynote, <http://www.uktest.org.uk>

El proceso: Modelo en V



Otros conceptos:

- Pruebas Unitarias (CSI)
- Estrategias de Integración
 - Ascendente. Conductores (test drivers, test harness)
 - Descendente. Resguardos (Stubs, mocks)
- Pruebas de regresión (regression testing)
- Pruebas de humo (smoke testing)

□ Mapeo a Métrica V3

- Componentes: Proceso CSI (Construcción del sistema)
- Integración y Sistema: Proceso CSI (Construcción del Sistema)
- Aceptación: Proceso IAS (Implantación y aceptación del Sistema)

Pruebas de sistema: diferentes objetivos



- ❑ Funcionales: se realizan las funciones especificadas
- ❑ Pruebas relacionadas con el rendimiento del sistema:
 - Rendimiento (tiempos de respuesta adecuados)
 - Volumen (funcionamiento con grandes volúmenes de datos)
 - Sobrecarga (funcionamiento en el umbral límite de los recursos)
- ❑ Disponibilidad de datos (cuando se produce una recuperación ante fallos)
- ❑ Facilidad de uso (usabilidad)
- ❑ Operación e instalación (operaciones de re arranque, actualización de software)
- ❑ Entorno (interacciones con otros sistemas) y comunicaciones
- ❑ Seguridad (control de acceso e intrusiones, ej. Inyección código SQL, XSS). Aunque esté al final de la lista no es el menos importante!!!

El Plan de Pruebas



- IEEE Standard for Software Test Documentation. IEEE Std 829-1983.
 - especificación del diseño de las pruebas,
 - casos de prueba,
 - procedimientos de prueba
 - informes
 - registros de prueba.
 - Problema: Complejo, mucha documentación
- a) Test plan identifier;
 - b) Introduction;
 - c) Test items;
 - d) Features to be tested;
 - e) Features not to be tested;
 - f) Approach;
 - g) Item pass/fail criteria;
 - h) Suspension criteria and resumption requirements;
 - i) Test deliverables;
 - j) Testing tasks;
 - k) Environmental needs;
 - l) Responsibilities;
 - m) Staffing and training needs;
 - n) Schedule;
 - o) Risks and contingencies;
 - p) Approvals.

Enfoque minimalista



- Hoja de cálculo para seguimiento de pruebas
 - Estructura jerárquica hasta llegar a los casos de prueba individuales
 - Identificación, descripción
 - estado (indicando si se ha superado la prueba o si se ha encontrado un fallo)
 - información adicional (fecha, quién la realizó y comentarios)
- La misma u otra con similar estructura como especificación de los casos:
 - Entradas
 - salidas esperadas.

Personal



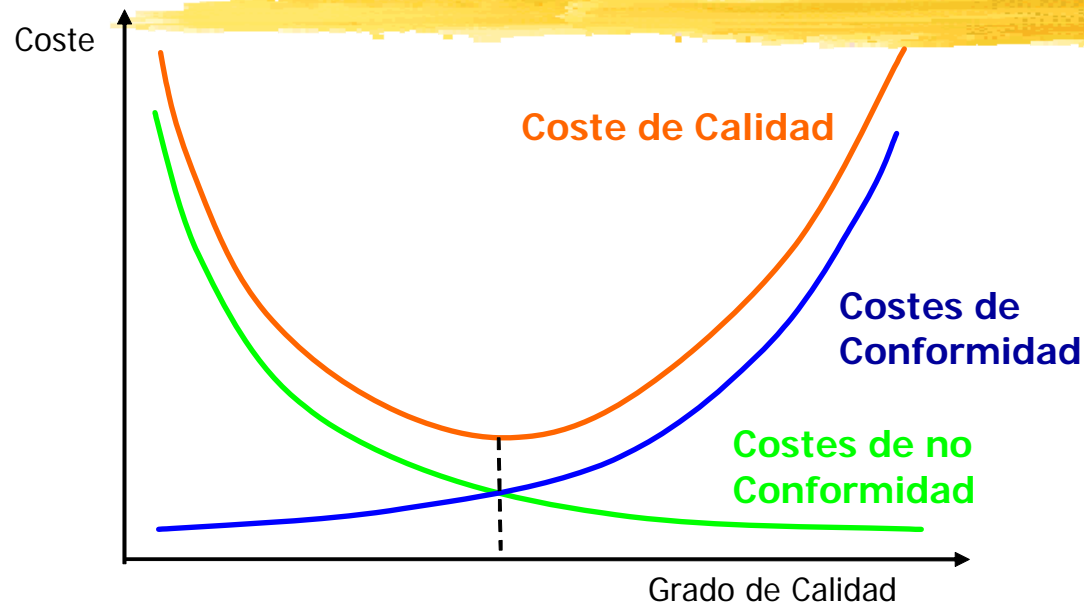
- Habilidades específicas del ingeniero de pruebas
 - Asumir el reto de encontrar fallos, más allá de “comprobar que funciona”
 - Concentrar la atención en los detalles
 - Curiosidad e intuición: explorar situaciones donde se pueden encontrar errores.
 - Capacidad de actuar bajo presión
 - Actuar cuando la presión del plazo es mayor
 - Conflicto de intereses con el equipo de desarrollo
 - Habilidades comunicativas
 - Comunicar de forma clara y precisa los defectos encontrados
 - Discutirlos con los equipos de desarrollo
 - Conocimiento funcional (depende de la tarea)
- Organización de equipos de pruebas. Máxima independencia
 - Asignación temporal o rotación entre equipo de desarrollo y prueba
 - Formación de equipos especializados
 - Internos (mejor conocimiento funcional)
 - Externos (más especializados, menor conocimiento funcional)
 - Mixtas

Planificación



- Principio básico:
 - Asignar recursos suficientes tanto a la preparación de pruebas como a su ejecución.
- Relación de tamaños equipos de desarrollo respecto de pruebas
 - Cifras típicas: desde 4 a 1 hasta 3 a 2
- Ventana temporal para las pruebas
 - Siempre considerar varios ciclos
 - Mínimo dos
 - Mejor más (el segundo ciclo puede incluir más pruebas para detectar más errores)

Costes



□ Invertir en pruebas (y en general en calidad) es rentable

■ (Krasner)

- En manufactura: 5-25%
- En software: 10-70%

■ En Empresas de Automoción:

- 4% Excelencia
- 4-8% Buenos
- >10% No deseables

□ $C = C_{\text{conformidad}} + C_{\text{noconformidad}}$

■ $C_{\text{conformidad}} = C_{\text{prevención}} + C_{\text{evaluación}}$

■ $C_{\text{noconformidad}} = C_{\text{internos}} + C_{\text{externos}}$

□ Costes de inactividad en cliente (N. Donfrio, 2002):

- Cadena Suministro: 300K\$/hora
- ERP/e-comm: 480K\$/hora

Algunas cifras



- ❑ Total de recursos empleados en pruebas:
 - 30% a 90% [Beizer 1990]
 - 50% a 75% [Hailpern & Santhanam, 2002]
 - 30% a 50% [Hartman, 2002]
- ❑ Mercado de herramientas: \$2,6 billion
 - Coste por infraestructura inadecuada:
 - Transporte y manufactura: \$1,840 billion
 - Servicios financiero: \$3,342 billion
 - Costes de reparación en función del instante en el ciclo de vida [Baziuk 1995]
 - Requisitos: x 1
 - Pruebas de sistema: x 90
 - Pruebas de instalación: x 90-440
 - Pruebas de aceptación: x 440
 - Operación y mantenimiento: x 880
 - Fuente general: The economic impact of inadequate infrastructure for software testing. NIST Report – May 2002

Modelos de Madurez. TMMI

- ❑ TMMI (Test Maturity Model) (<http://www.tmmifoundation.org/>), específico para pruebas (complemento a CMM, CMMI)
- ❑ Cinco niveles:
 1. (Inicial): Comprobar que el sistema **funciona sin fallos graves**. No suficiente asignación de recursos, herramientas ni especialización.
 2. (Definición): Verificar que **se satisfacen los requisitos**. Proceso separado, planes y estrategia. Comienzan relativamente tarde en el proyecto. **Políticas/Objetivos. Planificación, Técnicas/Métodos. Entorno.**
 3. (Integración): Las pruebas se enfocan hacia los **fallos en el sistema**. Procesos de pruebas integrados en el modelo en V. Planificación en etapas tempranas. Organización de pruebas y formación. Organización. Entrenamiento. **Organización. Entrenamiento. Ciclo de vida e integración. Control y Monitorización.**
 4. (Gestión y medición): Pruebas percibidas como **evaluación durante todo el ciclo de vida**. Proceso medible. Revisiones forman parte de este proceso. Las pruebas se organizan y reutilizan en pruebas de regresión. **Revisiones. Medición. Evaluación Calidad.**
 5. (Optimización): **Prevenir defectos**. Control sistemático de costes y efectividad. Enfoque mejora continua. **Prevención. Control. Optimización.**

Herramientas



- JUnit (<http://www.junit.org/>)
 - Las pruebas se escriben en Java
 - Clases derivadas de TestCase
 - Comparación salidas (assertXXX)
 - Automatización prueba de regresión
 - Pruebas autodocumentadas (en el código)
 - Independiente e Integrado en Eclipse
- NUnit (entorno .NET) (<http://www.nunit.org/>)
 - Equivalente a Junit, diversos lenguajes
 - Menor nivel de integración

Herramientas



- Extensiones Junit (Xunit)
 - Interfaz de usuario swing: JFCUnit (<http://jfcunit.sourceforge.net/>)
 - Identificar e interactuar con objetos gráficos
 - Interfaz Web: HttpUnit (<http://httpunit.sourceforge.net/>) y JWebUnit (<http://jwebunit.sourceforge.net/>)
 - Identificar e interactuar con objetos de la página
 - Manejo de request/response
 - Datos: DBUnit (<http://dbunit.sourceforge.net/>)
 - Conexión con la base de datos
 - Diversos formatos para cargar datos (XML, base de datos, CSV, etc.)
 - Métodos assertXXX para comparar las estructuras de tablas
 - Cargas masivas de datos: DBMonster (<http://dbmonster.kernelpanic.pl/>)

Otras herramientas



- ❑ Análisis de cobertura de código: Clover (<http://www.cenqua.com>)
 - Integración con Eclipse
 - Cobertura de líneas y decisiones. Estadísticas
- ❑ Carga y Stress: OpenSTA (<http://www.opensta.org/>).
 - Registro de una sesión interactiva
 - Lenguaje de programación, modificar script
 - Ejecutar, simulando usuarios y controlando la carga
 - Estadísticas
- ❑ Seguimiento de defectos: Bugzilla (<http://www.bugzilla.org/>).
- ❑ Muchas otras comerciales.
- ❑ Links de interés:
 - "Software testing tools FAQ" (<http://www.testingfaqs.org/>)
 - "Software QA Testing and Test Tool Resources" (<http://www.aptest.com/resources.html>).

Ejemplo Cobertura Clover

The screenshot shows the Eclipse IDE interface. The top part displays the Package Explorer on the left with a tree view of test classes. The main editor shows the source code for `Recibos.java` and `Test11ReclamacionImpagados.java`. The bottom part of the IDE shows the Clover View, which provides a detailed coverage report.

```
db.ejecutaSql(query);
return nuevoLote;
}
/**
 * Marca un lote como cerrado.
 * @param lote id del lote
 * @throws DataException
 */
public void cierraLote(int lote) throws DataException {
    db.ejecutaSql("UPDATE Lote SET estado='"+LOTE_GENERADO
}
/**
 * Obtiene la lista de socios a los que se debe girar una
```

Clover View Coverage Report:

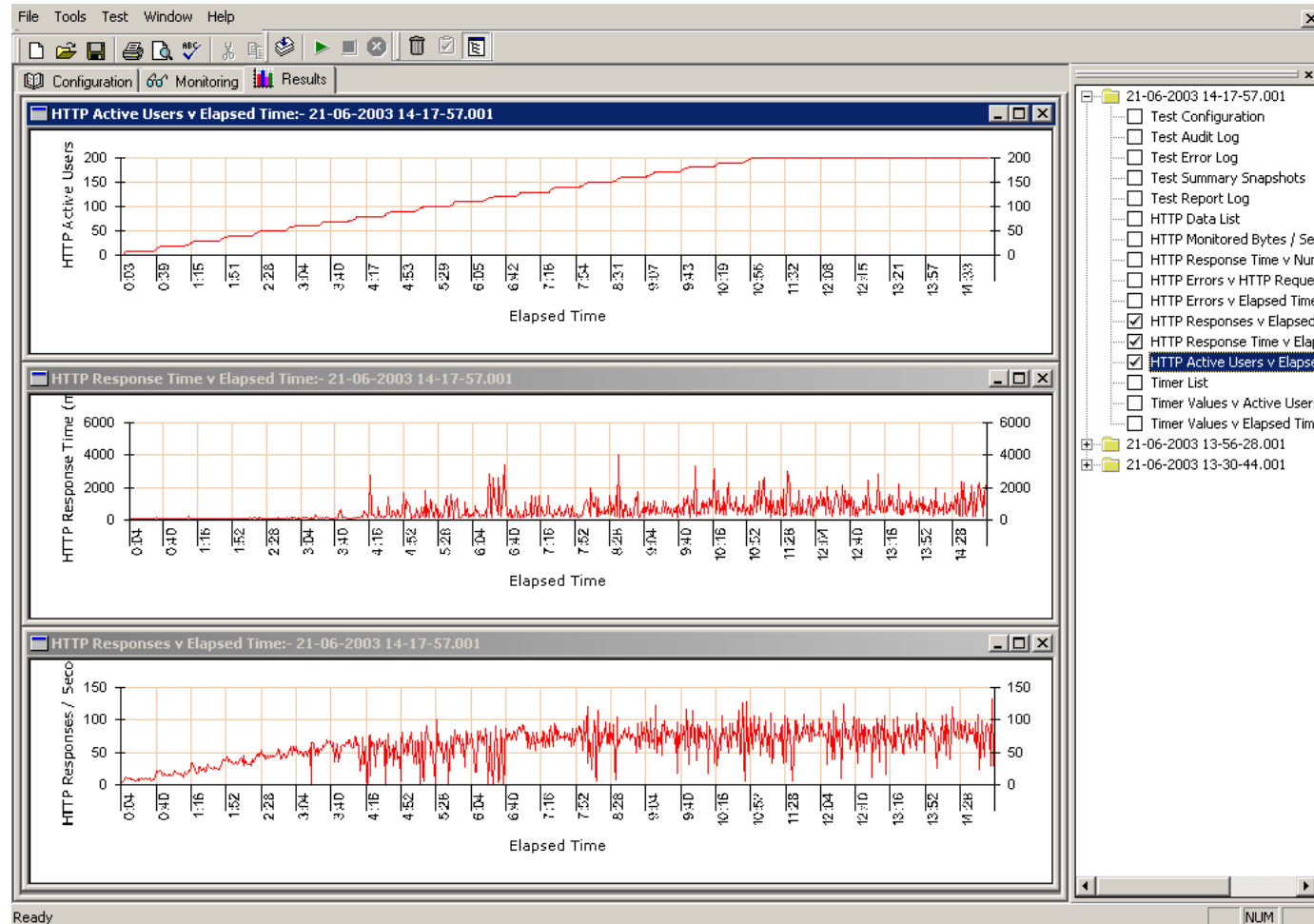
Package	Coverage
fact	42%
demos	0%
evaltest	75,2%
fact.interfaz	57%
fact.negocio	88,9%

Category	Count	Percentage	Visual
Methods	13 / 16	81,2%	<div style="width: 81.2%; background-color: green;"></div>
Statements	91 / 101	90,1%	<div style="width: 90.1%; background-color: green;"></div>
Conditionals	16 / 18	88,9%	<div style="width: 88.9%; background-color: green;"></div>
TOTAL		88,9%	<div style="width: 88.9%; background-color: green;"></div>

Metrics:

Lines of Code:	337	Classes:	2
NC Lines of Code:	176	Files:	2
Methods:	16	Packages:	0

Ejemplo salida OpenSTA



Automatizar o no automatizar?



- ❑ Las herramientas no son la panacea, pero ayudan
- ❑ Siempre se requiere un buen diseño
- ❑ La automatización es complementaria, facilita la repetibilidad
- ❑ Algunos problemas, p.e. herramientas capture/playback, difícil mantenimiento
- ❑ Casos típicos: pruebas de humo y regresión, pruebas de configuraciones, aplicaciones web, extreme programming
- ❑ Prever el coste y recursos del proyecto de automatización
- ❑ Subconjunto mínimo: Junit/DBUnit, carga/stress, seguimiento defectos

Investigar en pruebas



- La práctica industrial usa un pequeño conjunto de técnicas y herramientas
 - Podemos mejorar éstas?
 - Y que sean aplicables?
 - Y además eficientes y efectivas?
 - Y además obtener resultados científicos?
- Algunas líneas:
 - Test case adequacy criteria (criterios de “suficiencia”)
 - Test case selection criteria (generación de casos)
 - Test case prioritization (priorización de casos)
 - Herramientas
 - Validación experimental. Mutation systems

Red para la Promoción y Mejora de las Pruebas en Ingeniería del Software (RePRIS):

<http://in2test.lsi.uniovi.es/repris/>



RePRIS - Objetivos de la Red - Windows Internet Explorer

http://in2test.lsi.uniovi.es/repris/

Google

Google

RePRIS - Objetivos de la Red

RePRIS Red para la promoción y mejora de las Pruebas en Ingeniería del Software

[Objetivos](#) | [Participantes](#) | [Actividades](#) | [Enlaces](#) | [Taller PRIS 2006](#)

Objetivos de la Red

La Red para la promoción y mejora de las Pruebas en Ingeniería del Software (RePRIS) tiene como objetivo coordinar los esfuerzos y conocimientos en pruebas del software entre grupos investigadores y el sector empresarial para difundir el conocimiento y promover la mejora de la práctica de la prueba del software en la industria, de las líneas de investigación y de la formación tanto a nivel universitario como no universitario.

- Dar a conocer y promover la investigación en pruebas de software.
- Promover el uso de buenas prácticas de pruebas y de las herramientas adecuadas en la industria del software.
- Analizar y promover la docencia en las pruebas del software.
- Incrementar el nivel competitivo de la investigación.
- Contribuir a mejorar la calidad del software desarrollado.

Acción Especial TIN2005-24792-E financiada por el Plan Nacional de I+D+I, [Ministerio de Educación y Ciencia](#), cofinanciado con Fondos FEDER.

Intranet local 100%

Grupo de la Universidad de Oviedo – Campus de GIJON

- <http://www.di.uniovi.es/~tuya/testing/>
- Criterios de suficiencia para aplicaciones con bases de datos: :
 - Using a SQL Coverage Measurement for Testing Database Applications, FSE 2004
 - A practical guide to SQL white-box testing, SIGPLAN Not. 41(4) 2006.
- Evaluación de la calidad de los casos de prueba de bases de datos:
 - Mutantes para SQL: Mutating Database Queries, Information and Software Technology, 49(4) 398-417, 2007
 - Automatización: SQLMutation: <http://in2test.lsi.uniovi.es/sqlmutation/>
- Pruebas de sistemas Internet y XML:
 - Verificación: Automatic Generation of Assumptions for Modular Verification of Software Specifications, Journal of Systems and Software, 79(9) 2006
 - Composiciones de Servicios Web: Generating test cases specifications for compositions of web services, WS-MaTe, 2006).
 - Consultas de repositorios XML: Testing XPath Queries using Model Checking, STV 2006, A Partition-based Approach for XPath Testing, ICSEA 2006.
- Generación automática de casos de prueba con técnicas metaheurísticas
 - Búsqueda Tabú: A tabu search algorithm for structural software testing , Computers and Operations Research, Special issue SB SW Eng, in press, 2007.