Pruebas del Software: Prácticas, Procesos y Herramientas durante el Desarrollo

Claudio de la Riva
Universidad de Oviedo
claudio@uniovi.es

TECH BUSINESS WEEK

Santiago de Compostela, 5 de Noviembre de 2007

“El ojo humano tiene una capacidad casi infinita para no ver lo que no quiere ver”

G. Weinberg

(The Psychology of Computer Programming, 1971)

“El ojo humano tiene una capacidad casi infinita para no ver lo que no quiere ver”

G. Weinberg

(The Psychology of Computer Programming, 1971)

“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 […]. The industry can't survive with this level of quality”

W. Humphrey

(http://www.cs.usask.ca/grads/jpp960/490/BombSquad.html)

“El ojo humano tiene una capacidad casi infinita para no ver lo que no quiere ver”

G. Weinberg

(The Psychology of Computer Programming, 1971)

“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 […]. The industry can't survive with this level of quality”

W. Humphrey

(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)

Agenda

- "Economía" de la calidad
- Visión rápida del estado del arte
- El factor humano
- Automatización y herramientas
- La prueba del software como disciplina de ingeniería

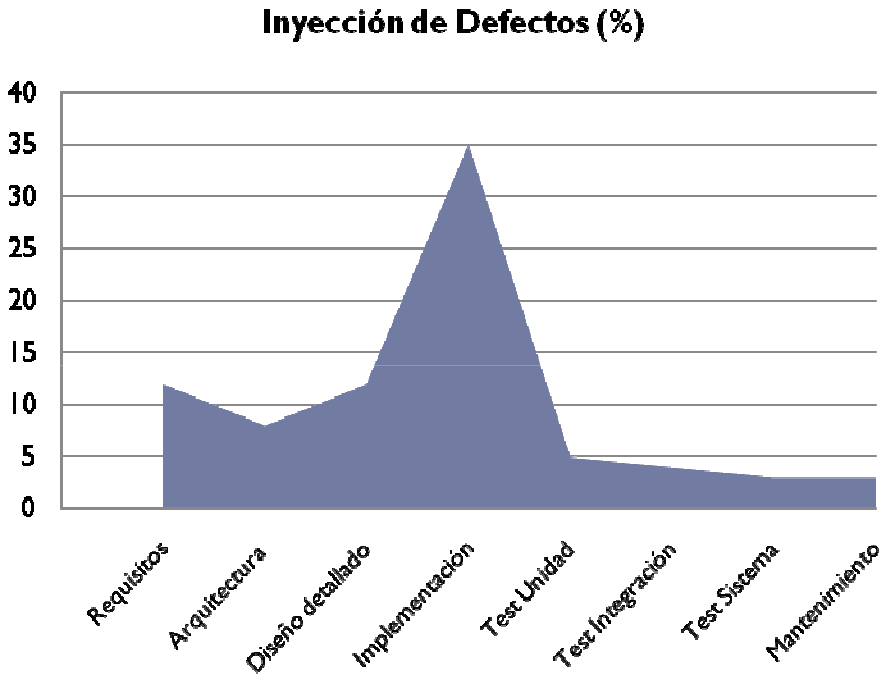
Calidad del Software

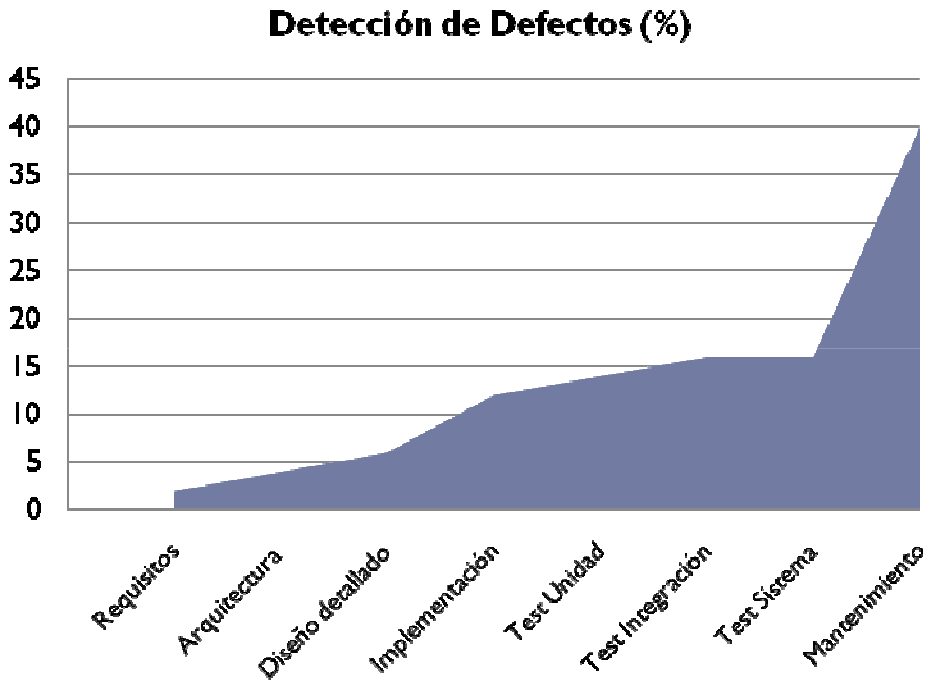
[Chaos Report – Standish Group, 2004]

"Only 28% of software projects succeed these days, down from 34% a year or two ago. Outright failures [projects cancelled before completion] are up from 15% to 18%. The remaining 51% of software projects are seriously late, over budget and lacking features previously expected."

¿La "crisis del software" no ha llegado aún a su punto de inflexión?

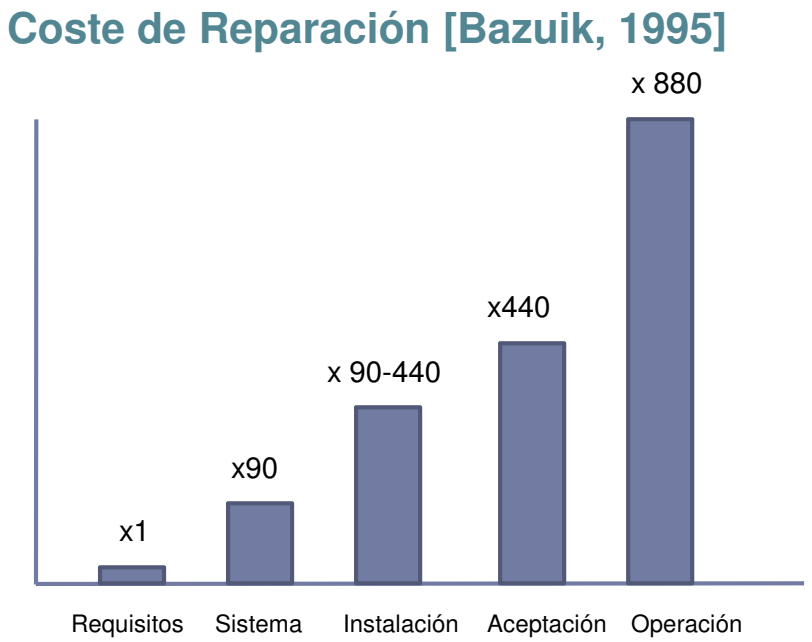
Origen y detección de defectos





Fuente: [Howe and Sassenburg, 2004]

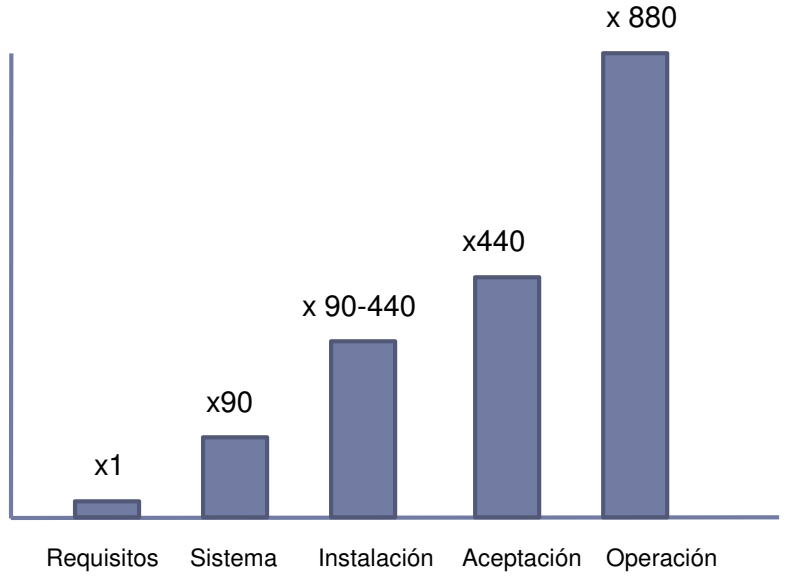
Costes


Coste de Reparación [Bazuik, 1995]
x 880
x440
x 90-440
x90
x1
Requisitos
Sistema
Instalación
Aceptación
Operación

Costes

- Recursos empleados en las pruebas
  - 30% al 90% [Beizer, 1990]
  - 50% al 75% [Hailpern, 2002]
  - 30% al 50% [Hartman, 2002]

Coste de Reparación [Bazuik, 1995]

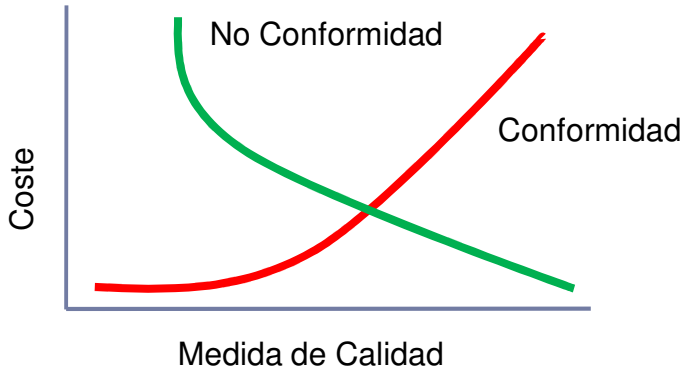


- Coste infraestructura inadecuada [NIST, 2002]
  - Transporte y manufactura: $1,840 billion
  - Servicios financieros: $3,342 billion

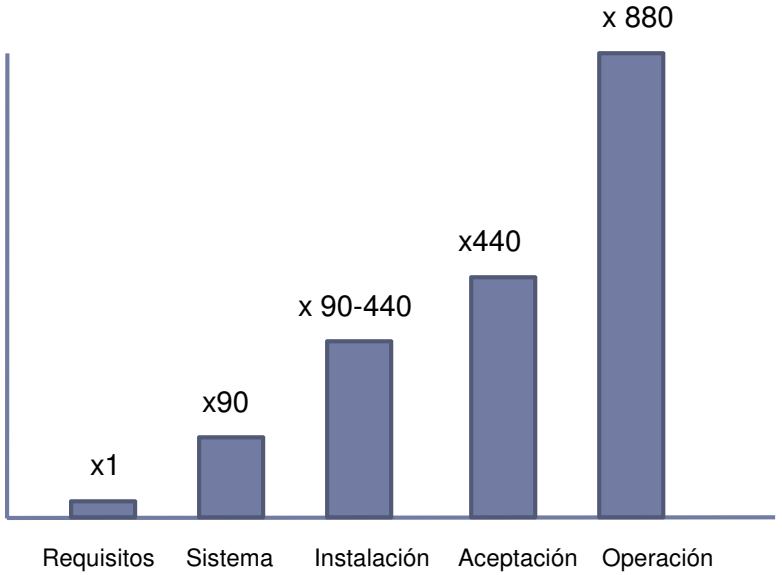
Costes

- Recursos empleados en las pruebas
  - 30% al 90% [Beizer, 1990]
  - 50% al 75% [Hailpern, 2002]
  - 30% al 50% [Hartman, 2002]

Coste de la Calidad del Software [Herb, 1998]



Coste de Reparación [Bazuik, 1995]



- Coste infraestructura inadecuada [NIST, 2002]
  - Transporte y manufactura: $1,840 billion
  - Servicios financieros: $3,342 billion

¿Qué es la Prueba del Software?

- Una de las prácticas más utilizadas dentro de las actividades de QA
- Objetivos generales
  - Descubrir fallos
  - Proporcionar e incrementar una medida de confianza
    - Fiabilidad y Rendimiento
    - Ausencia de determinados fallos
    - …
- Definición (SWEBOK)
  - *Software testing consists of **dynamic** verification of the behavior of a program on a **finite** set of test cases, suitably **selected** from the usually infinite executions domain, against the specified **expected** behavior*

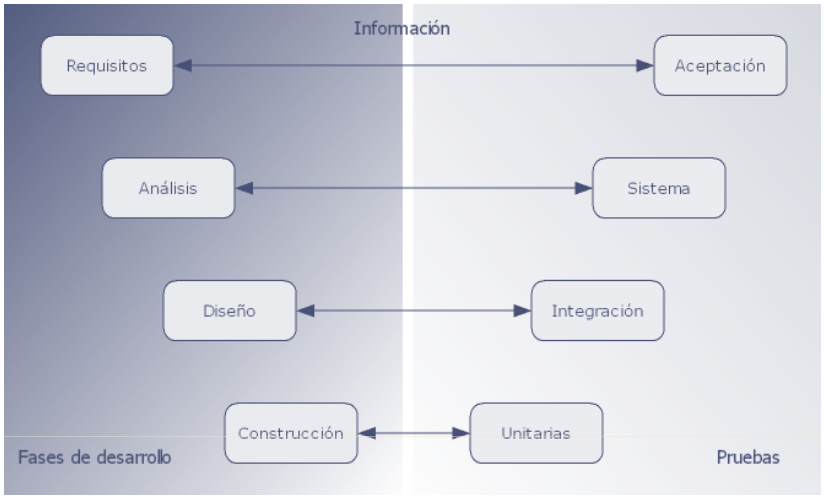
¿Qué es la Prueba del Software?

- Una de las prácticas más utilizadas dentro de las actividades de QA
- Objetivos generales
  - Descubrir fallos

Muestrear el software con el objetivo de emitir un veredicto

  - …
- Definición (SWEBOK)
  - *Software testing consists of **dynamic** verification of the behavior of a program on a **finite** set of test cases, suitably **selected** from the usually infinite executions domain, against the specified **expected** behavior*

¿Qué es la Prueba del Software? –cont.



- Diferentes niveles
  - Unitarias
  - Integración
  - Sistema
  - Aceptación
  - Regresión
- Diferentes técnicas
  - Funcionales
  - Estructurales
  - Estáticas
  - Dinámicas
  - ….
- Diferentes objetivos
  - Conformidad y Funcionales
  - Rendimiento (sobrecarga, volumen, rendimiento, …)
  - Seguridad
  - Usabilidad
  - Instalación, operación y comunicaciones
  - ….
- Diferentes enfoques
  - Exploratory Testing
  - Test-Driven Development
  - Model-Based Testing
  - Manuales
  - Automáticas
  - …

¿Existe un plan?

- 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 (logs)

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.

¿Existe un plan?

- 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 (logs)
- 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.

¿Existe un plan? – cont.

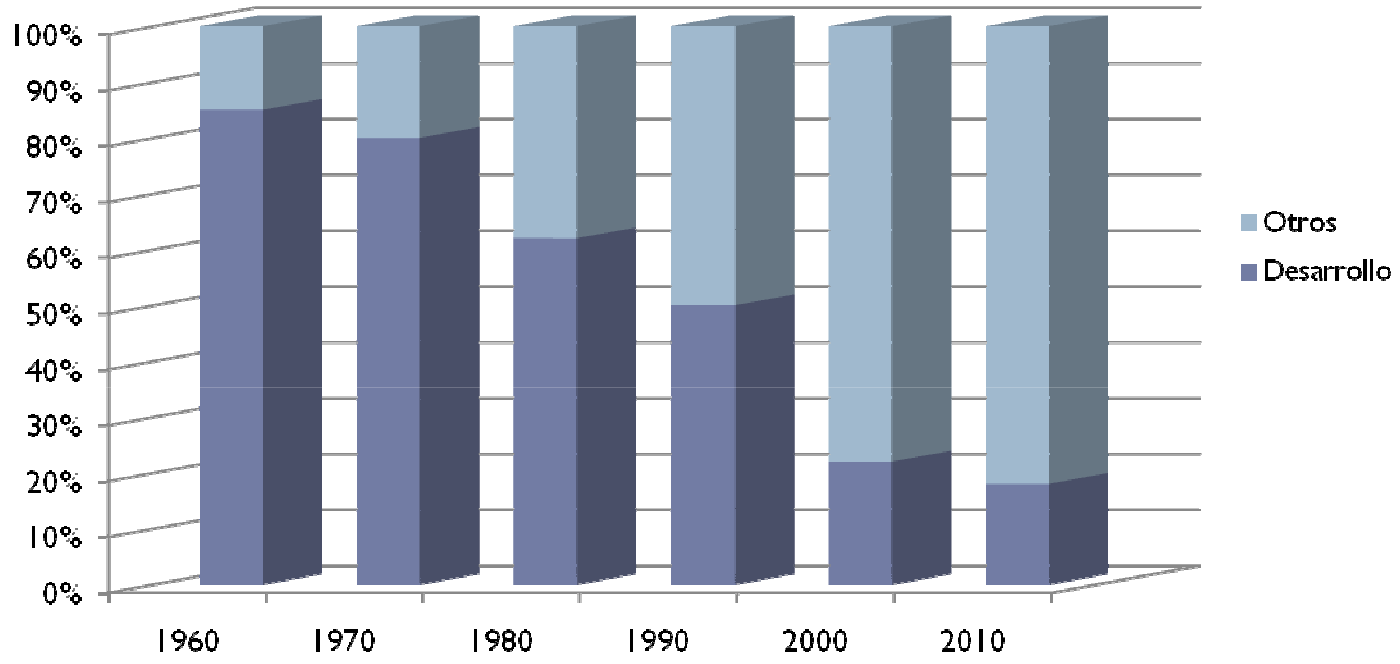
- Más sencillo, hojas de cálculo o similar:
  - Especificación de casos de prueba
  - Seguimiento de ejecución de los casos de prueba
- Especificación de casos de prueba
  - Entradas
  - Salidas deseadas
- Seguimiento
  - Estructura jerárquica de casos
  - Descripción e identificación
  - Estado (pass/fail)
  - Otras (fecha, autor, comentarios)

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)

El factor humano

Fuente: [Jones, 2006]



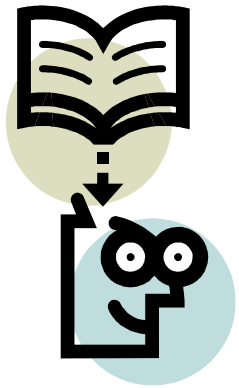
"*Tester skill is at the centre of testing*" [Bolton, 2006]

El factor humano – cont.

- Habilidades específicas:
  - No solamente "comprobar que funciona"
  - Especial atención a los detalles
  - Pensamiento crítico
  - Adaptación a situaciones cambiantes
  - Posibles conflictos con equipo de desarrollo
  - Capacidades comunicativas
  - Conocimiento funcional (no siempre)
- Estructura:
  - Rotación con equipos de desarrollo
  - Equipo independiente
  - Mixtas

Automatización y Herramientas

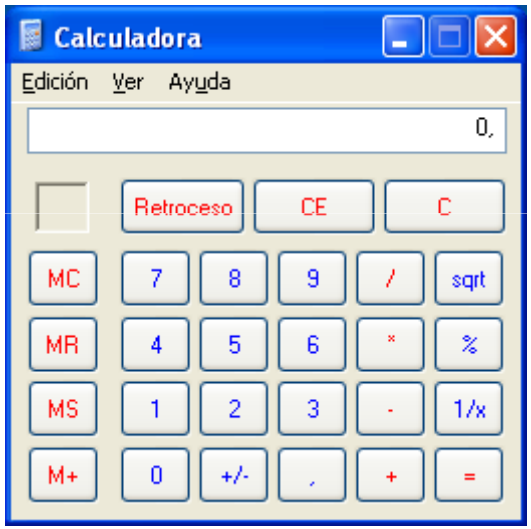
Manual



Veredicto:
Pass/Fail

Casos de Prueba

Resultados

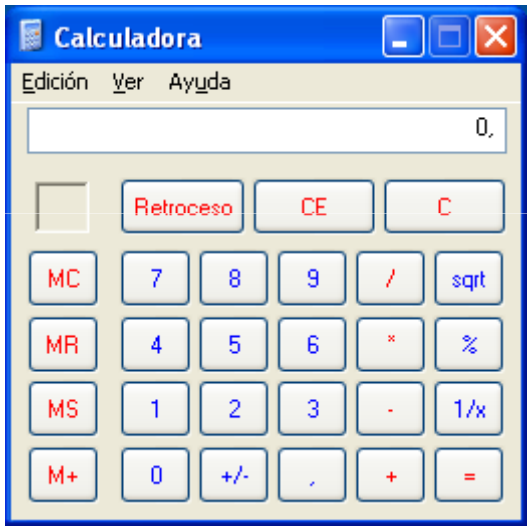


Automatización y Herramientas

Automático

Casos de Prueba

Resultados

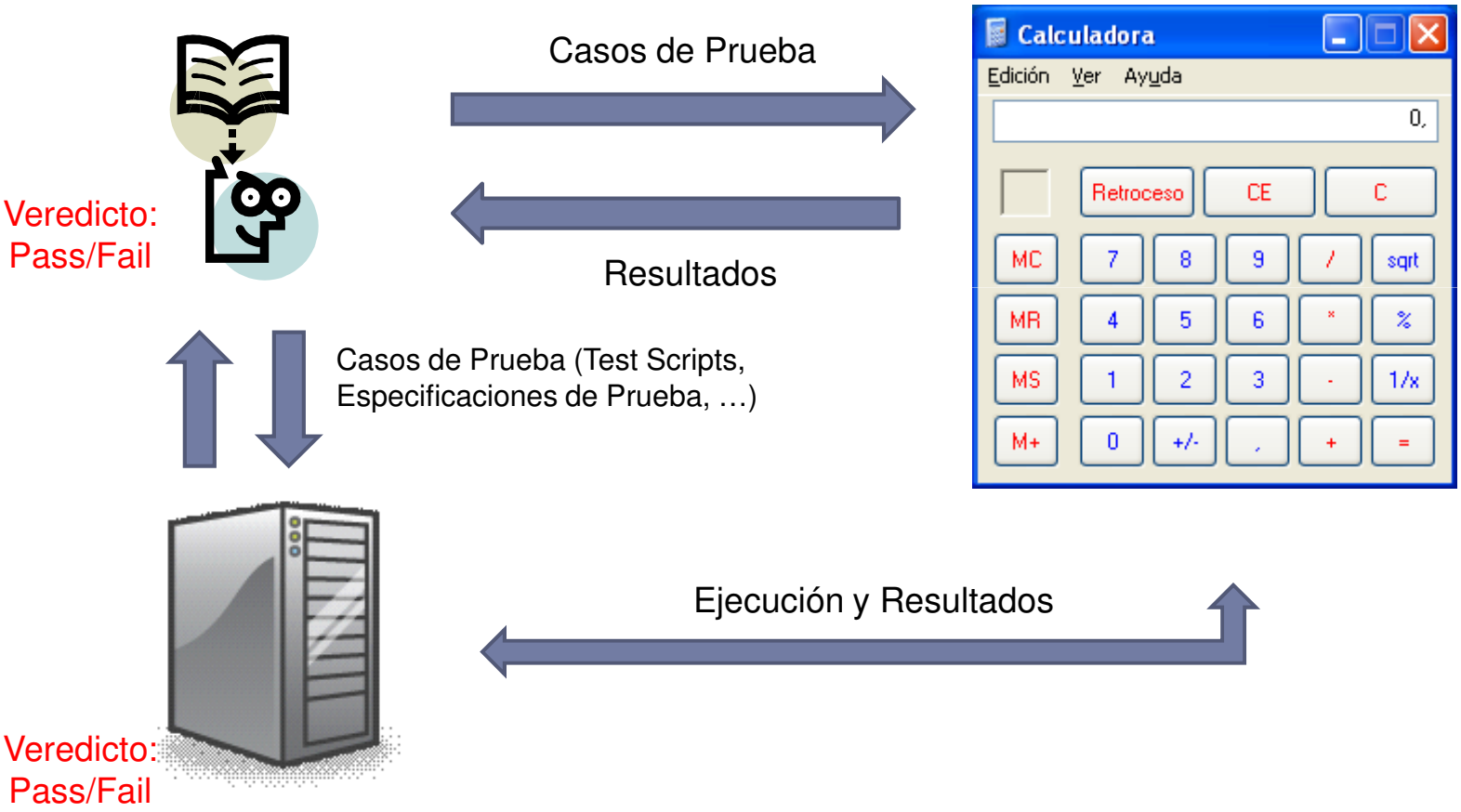


Veredicto:
Pass/Fail

¿Sueño o Realidad?

Automatización y Herramientas

Realidad: Fuerte componente manual



Automatización y Herramientas – cont.

- ¿Qué se puede (parcialmente) automatizar?
  - Pruebas de carga y rendimiento
  - GUI y algunas funcionales
  - Pruebas unitarias
  - Seguimiento de defectos
  - Cobertura de código
- Beneficios
  - Eliminación de errores "humanos" (componente manual)
  - Reducción del tiempo de ejecución de casos
  - Reproducibilidad (pruebas de regresión)
  - Mayor cobertura
  - Documentación automática

Automatización y Herramientas – cont.

- Pero también tiene costes:
  - Hardware, software
  - Formación
  - Puesta en marcha inicial
  - Análisis de resultados
  - …
- En resumen:
  - Planificar coste y recursos del proyecto de automatización
  - Lo importante es un buen diseño de los casos de prueba
  - La automatización es complementaria
  - Las herramientas no son la panacea, pero ayudan

Herramientas - Ejemplos

- Captura y Reproducción (Test Scripts)

Captura

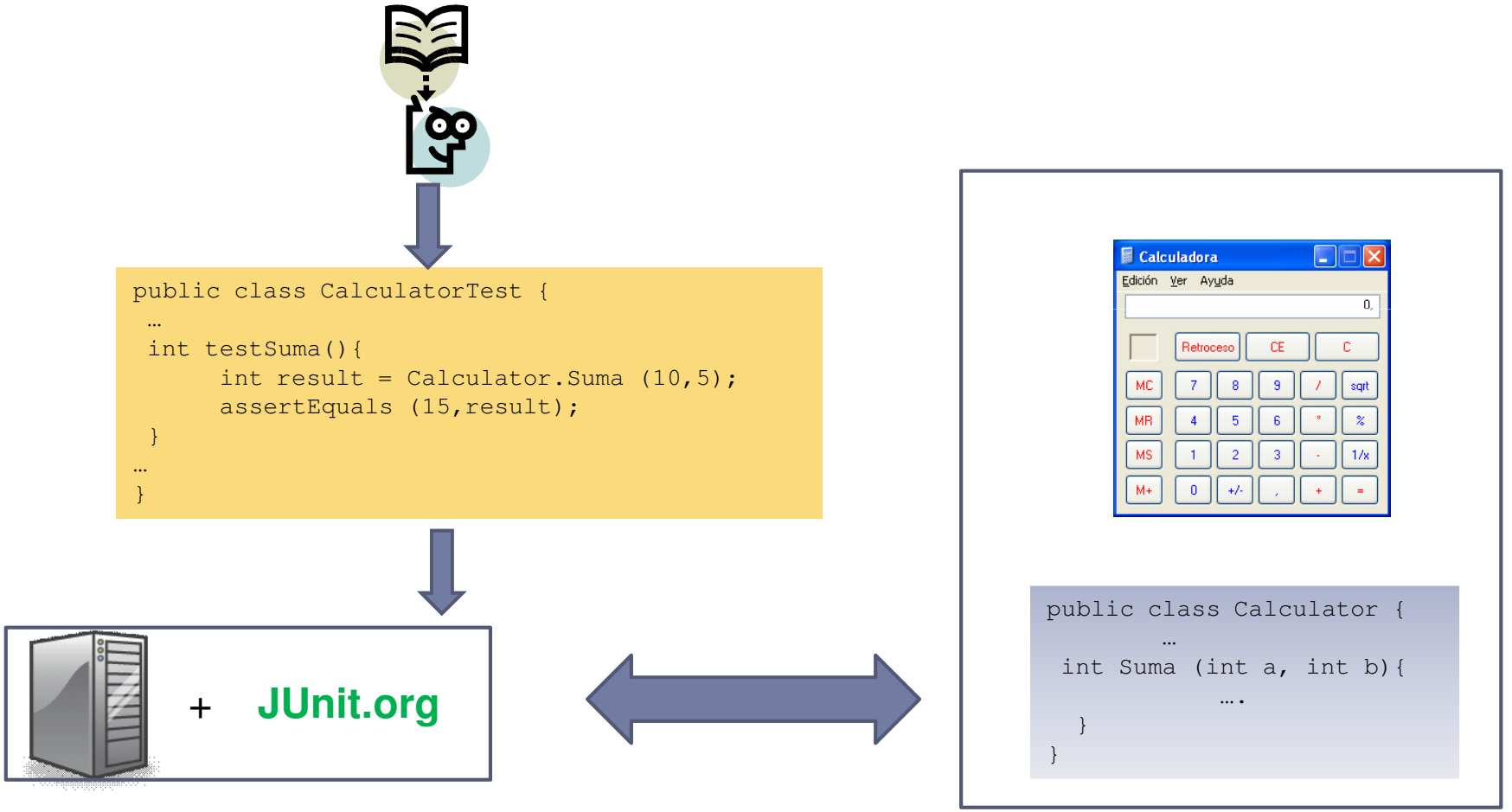
```
COMIENZO
     Comprobar Display = 0;
     Pulsar Tecla = 10
     Comprobar Display = 10;
     Pulsar Tecla = "+"
     Pulsar Tecla = 5
     Comprobar Display = 5;
     Pulsar Tecla = "="
     Comprobar Display = 15

FIN
```

Reproducción

Herramientas – cont.

- Framework para pruebas unitarias (JUnit)



Herramientas – cont.

- Extensiones JUnit (XUnit)
  - Entorno .NET (NUnit)
  - Interfaz Web (HTTPUnit)
  - Interfaz de Usuario (JFCUnit)
  - Bases de Datos (DBUnit)
- Cobertura de código
  - Coverlipse
- Carga
  - OpenSTA
- Seguimiento de defectos/errores
  - Bugzilla
- Eclipse TPTP (Test and Performance Tool Platform)
  - Análisis de Código Estático
  - JUnit, Manual Test, GUI Tests, URL Tests
  - Instrumentación Estática y Dinámica
  - Monitorización y Análisis de logs (Run-time testing)
- Otras muchas comerciales

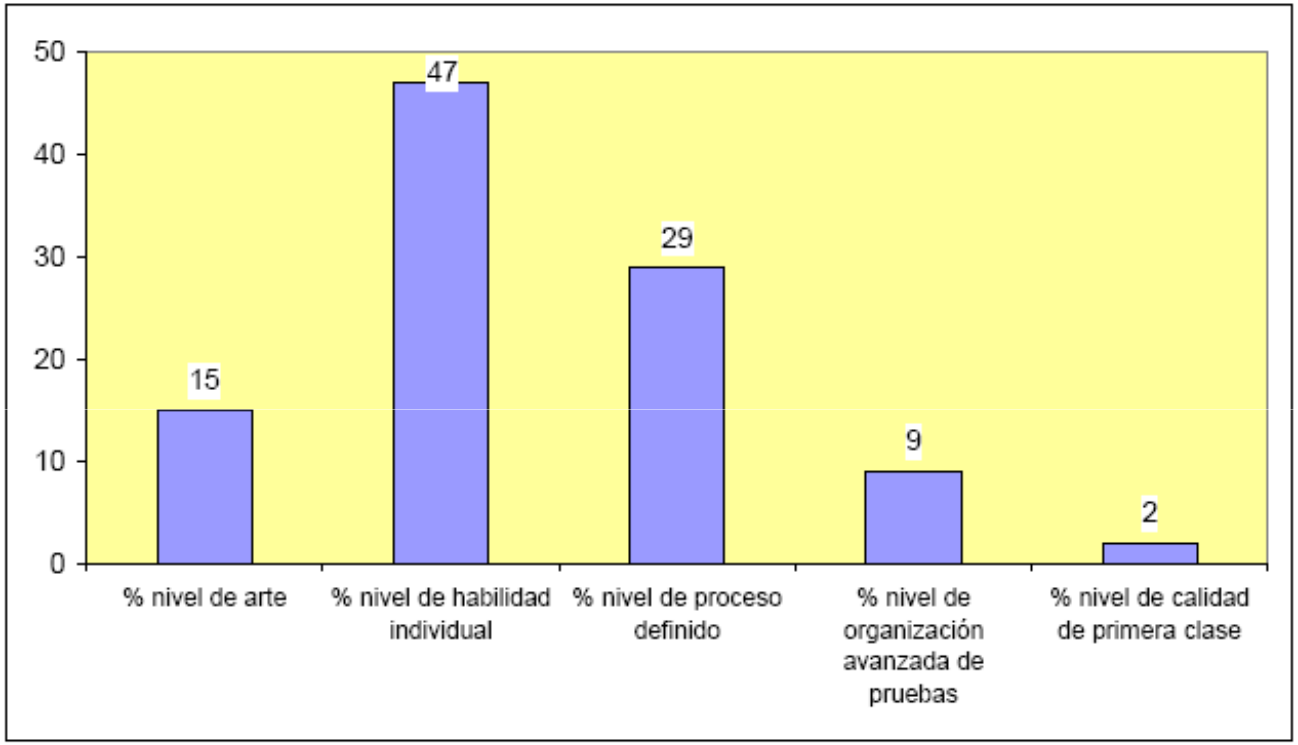
Madurez de las Pruebas del Software

- ¿(In)Madurez de las Pruebas del Software?
- Cuerpo de conocimiento:
  - Varias técnicas
  - Varias prácticas
  - Varias herramientas
  - Algunos modelos de procesos
- ¿Cuál es el "estado de la práctica"?
  - Hechos probados (Nivel de madurez **ALTO**)
  - ….
  - ….
  - Conjeturas o creencias (Nivel de madurez **BAJO**)

Madurez de las Pruebas del Software - cont

- De la investigación a la práctica:
  - Pruebas de cobertura
    - Investigación: Método de generación de casos de prueba a partir de un criterio de cobertura (ramas, condición, MC/DC, …)
    - Práctica: Monitorización del software para cumplir una cobertura (sentencias, ramas, ..)
  - Pruebas funcionales
    - Investigación: Búsqueda de métodos (automáticos) para derivar casos de prueba a partir de especificaciones
    - Práctica: Diseño de casos basados (principalmente) en la intuición
  - Nivel
    - Investigación: Fundamentalmente pruebas unitarias
    - Práctica: Más interés en pruebas de sistema, aceptación,…
- [Juristo, 2004]
  - "Más de la mitad de las técnicas existentes están basadas en impresiones o percepciones y no tienen una base formalizada ni experimentada"

¿Y en España?



Fuente: [Fernández, 2005]

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

Queda bastante por hacer

Fuente: [Bolton, 2006]

Muchas gracias

Claudio de la Riva
Universidad de Oviedo
claudio@uniovi.es