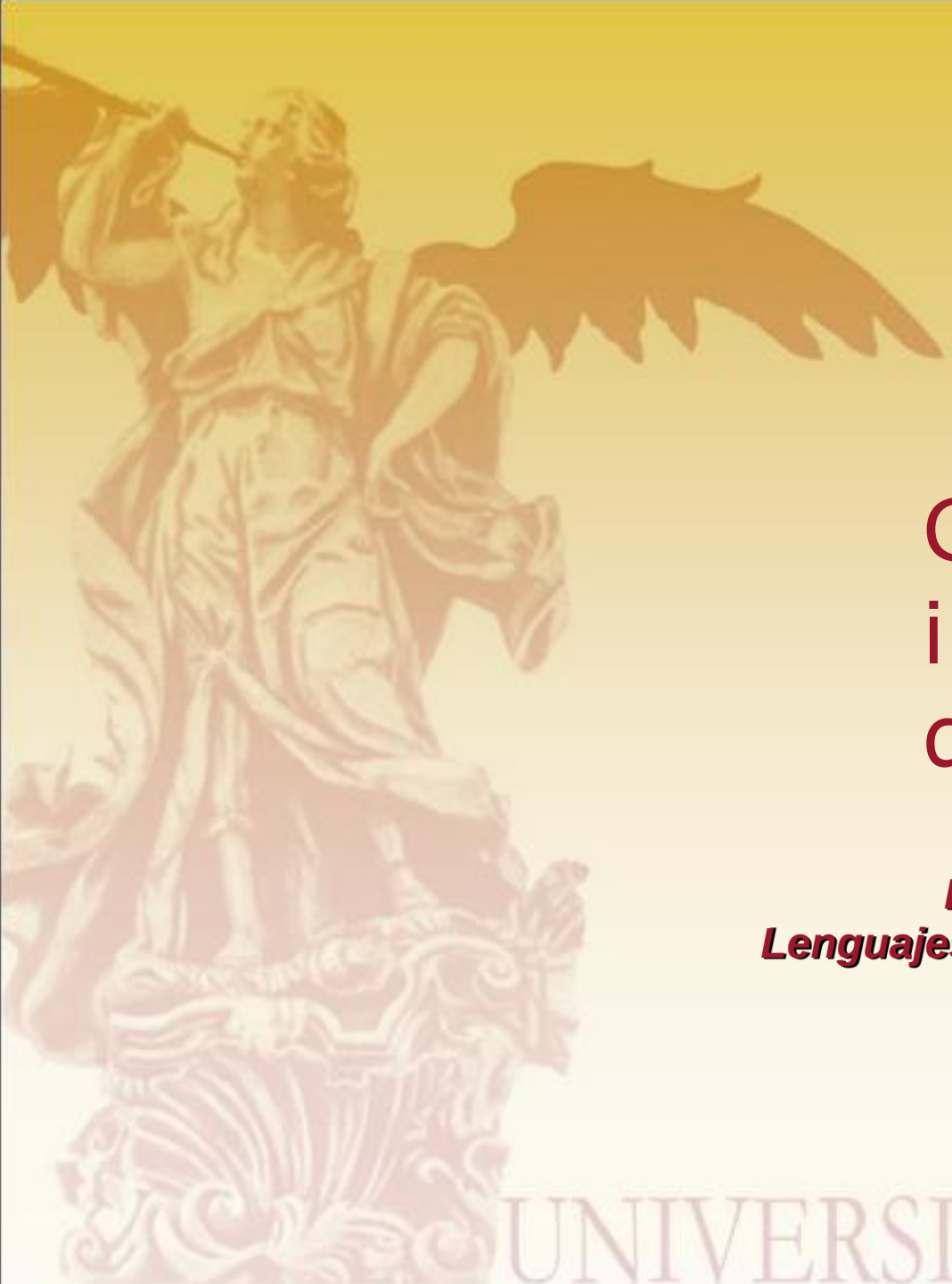


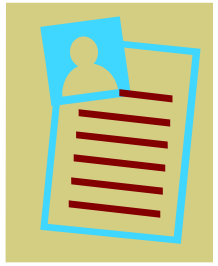
escuela técnica superior
de ingeniería informática

# Gestión de incidencias y depuración

***Departamento de Lenguajes y Sistemas Informáticos***

UNIVERSIDAD D SEVILLA

# Ejemplo de otro dominio



INCIDENCIA

INFORME

DEPURACIÓN, REPARACIÓN

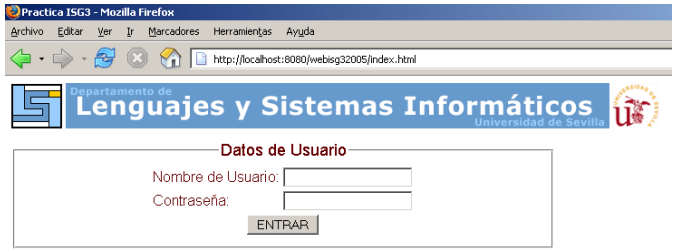
# Índice

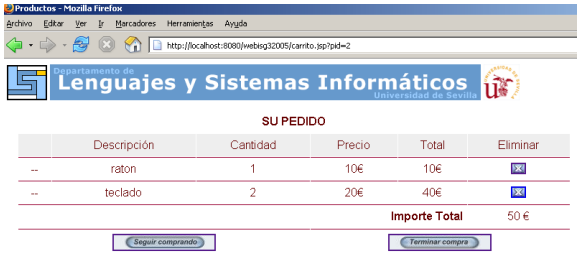
# Aplicación de ejemplo

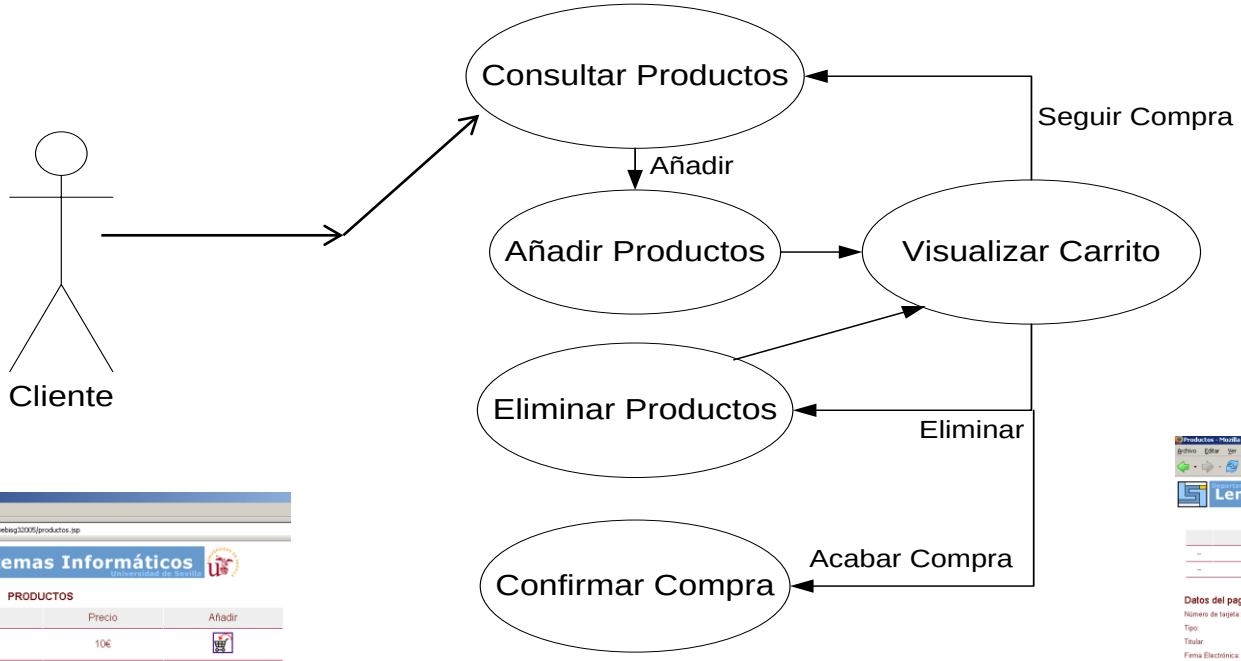
Acciones que debemos permitir:

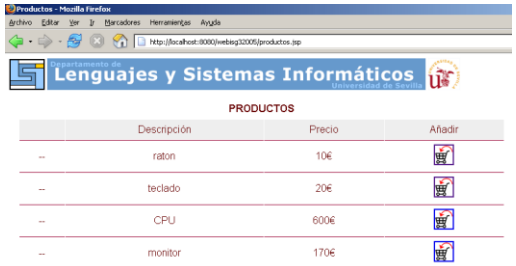
- Identificarse
- Consultar productos
- Añadir productos a un pedido
- Eliminar productos de un pedido
- Confirmar la compra

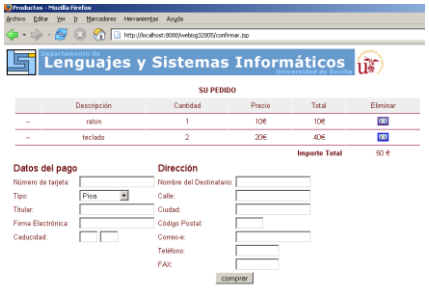
# Aplicación de ejemplo



# ¿Cuál es el problema?

¿Creéis que en USVirtual se encontrarán errores/incidencias durante su tiempo de desarrollo y explotación?
¿Cómo reportar esas incidencias?
¿Cómo abordarlas en caso de que sean fallos?

# Ejemplos de posibles incidencias

- No se conecta con la base de datos
- Lanza una excepción cuándo hay un campo que no se ha rellenado
- Se conectan demasiados usuarios de manera concurrente (automatrícula)
- No se puede instalar la aplicación
- ¡Han cambiado las reglas de negocio!
- …

# ¿Cuál es el problema?

¿Cuántas líneas de código al día produce un ingeniero de nokia de media a lo largo de un año?

Jan Bosch, antiguo miembro de Nokia I+D, Keynote en SPLC 2005

Conjetura: el tiempo que se emplea en depuración de código es más alto de que el que se emplea en la propia codificación a lo largo del ciclo de vida de un sistema software

# ¿Cuál es el problema?

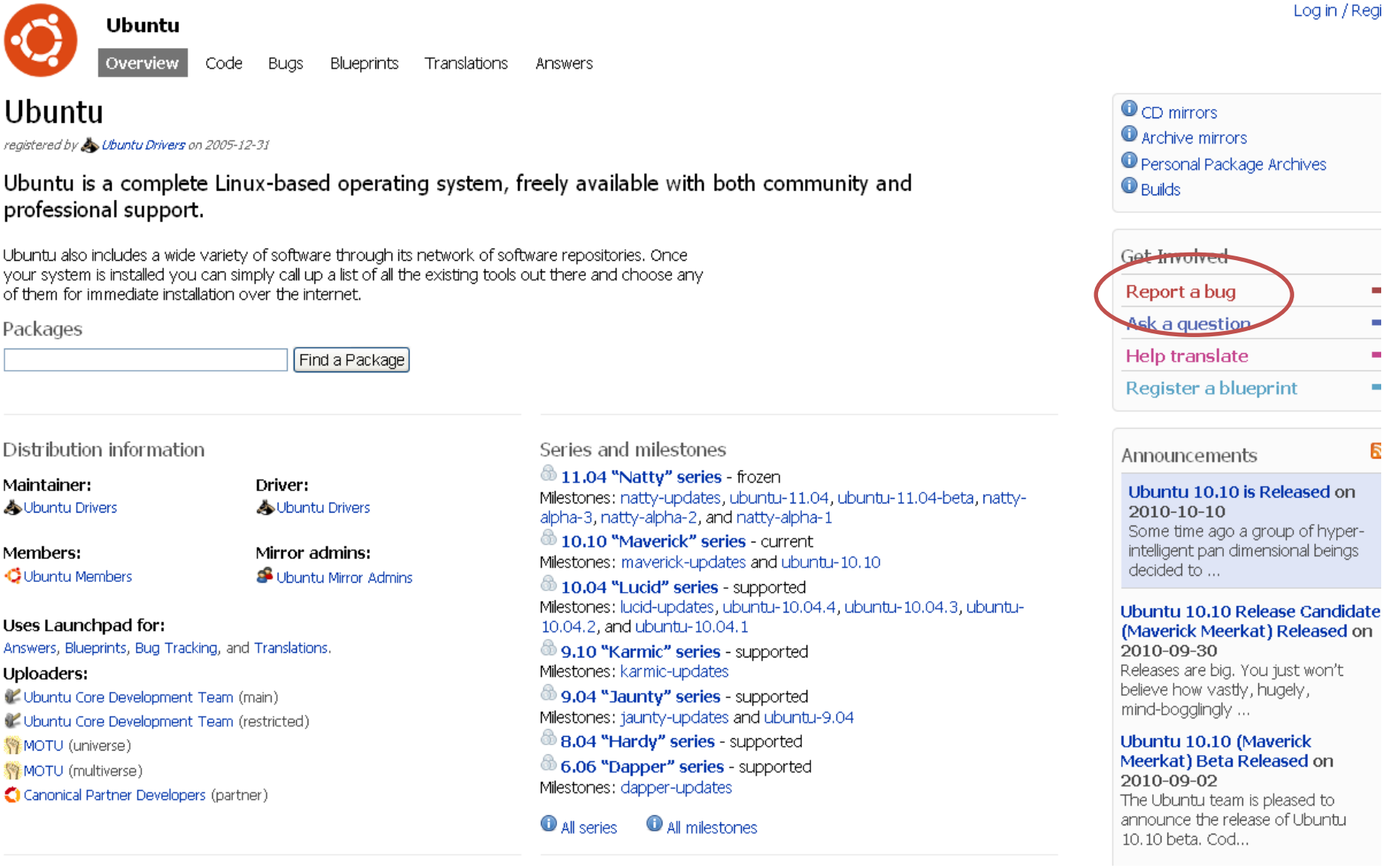
¿Cómo gestionar de manera sistemática los cambios debidos a incidencias en general y a detección de errores en particular en un sistema software?

# Definiciones

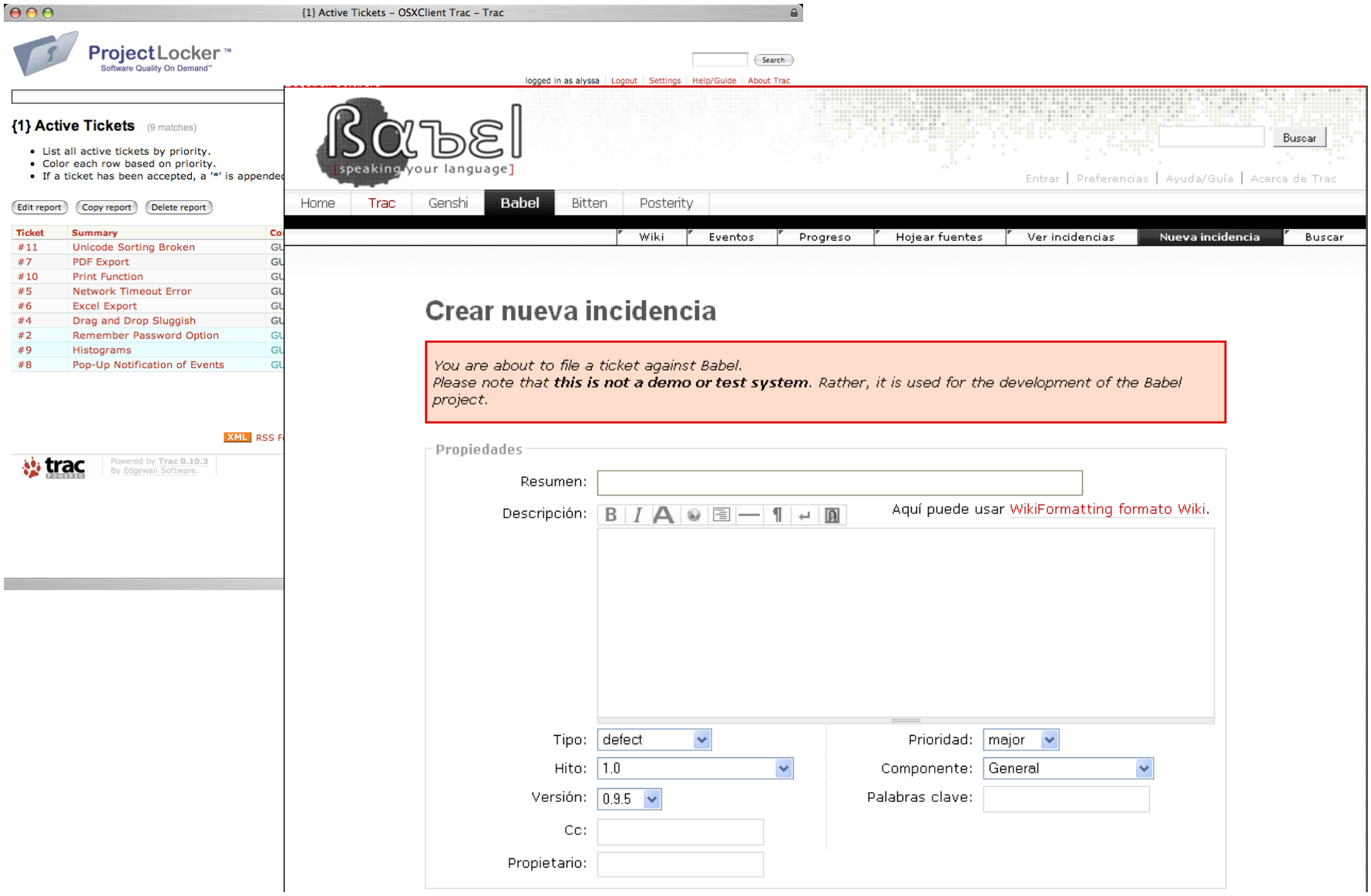
- **Failure** (fallo): La incapacidad de un sistema para realizar su función, es decir,  hay una diferencia entre lo que se espera que haga el sistema y lo que está haciendo el sistema. Se podría definir como “los síntomas”.
- **Fault** (bug): La causa que provoca el fallo. Se podría definir como “la causa”
- **Error**: El estado interno de un programa que contiene un *bug.* Algunos estados no revelan un fallo y algunos estados erroneos pueden no ser la causa del fallo.
- **Debug.** Detectar, localizar y corregir *bugs* en un programa.

# El escenario ideal

# Ejemplos de sistemas con GI

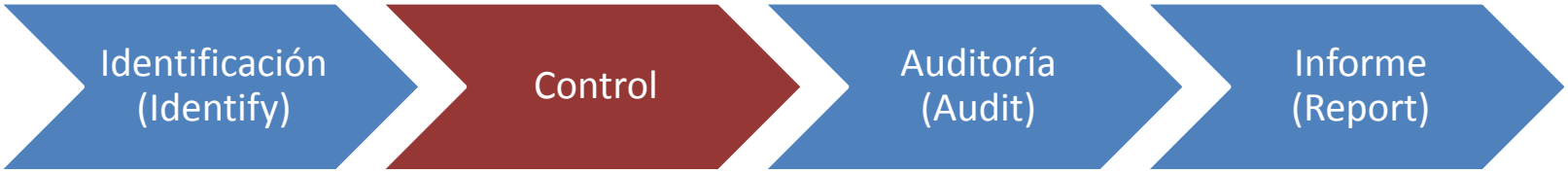


# Ejemplos de sistemas con GI

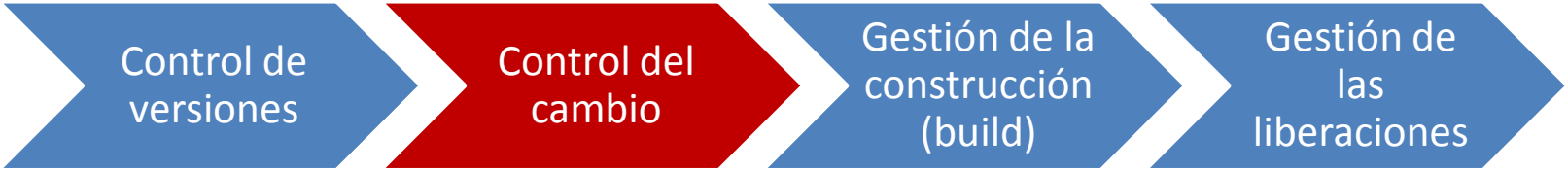


# Índice

# Actividades en SCM




- Identificación permite saber qué cambios han ocurrido
- Control permite decidir cuándo realizar cambios




- Control de versiones: ¿Cuándo y cómo vamos a liberar versiones?
- Control del cambio: ¿Cómo vamos a gestionar los cambios?
- Gestión de la construcción: ¿Cómo vamos a gestionar la construcción de versiones?
- Gestión de las liberaciones: ¿Cómo vamos a gestionar la liberación, instalaciones y configuraciones?

# Proceso de GC y GI

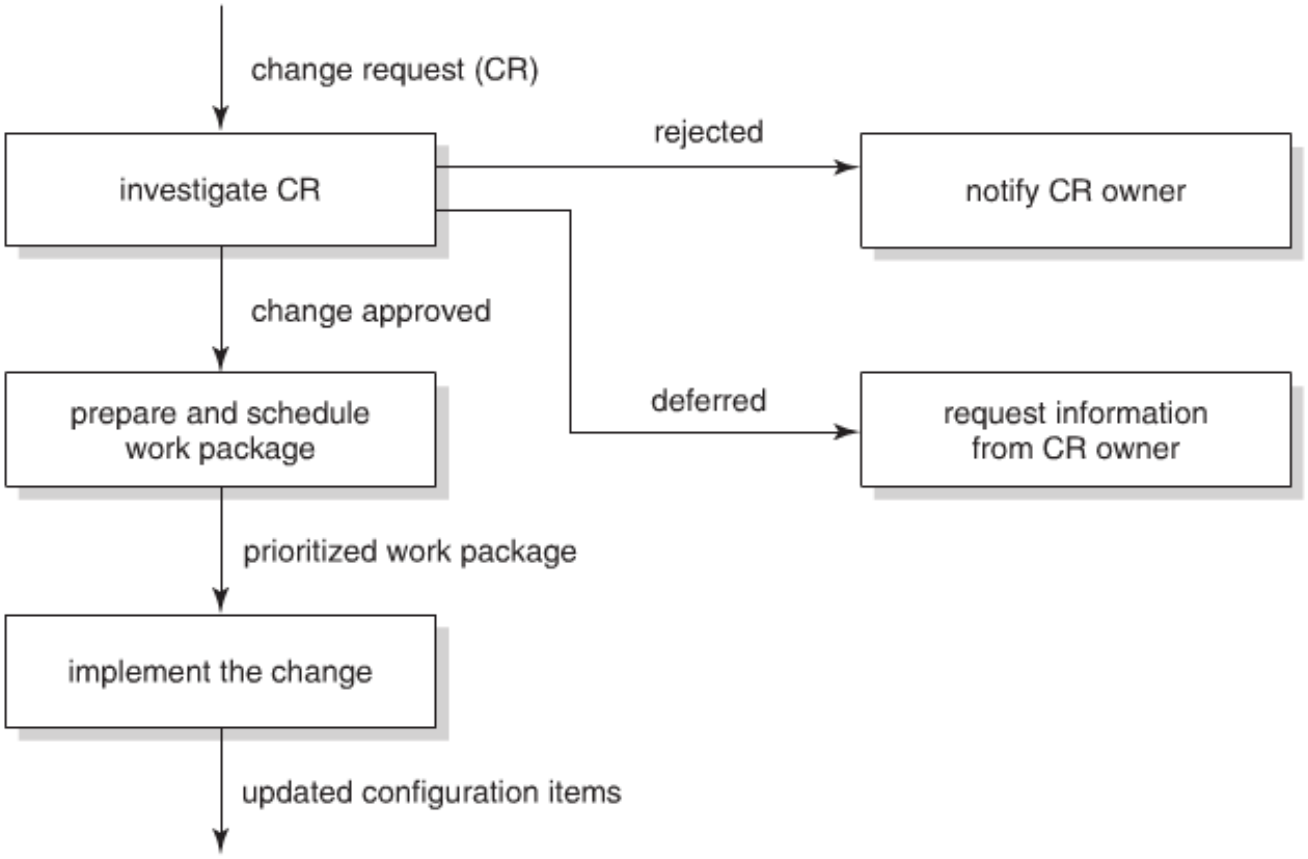
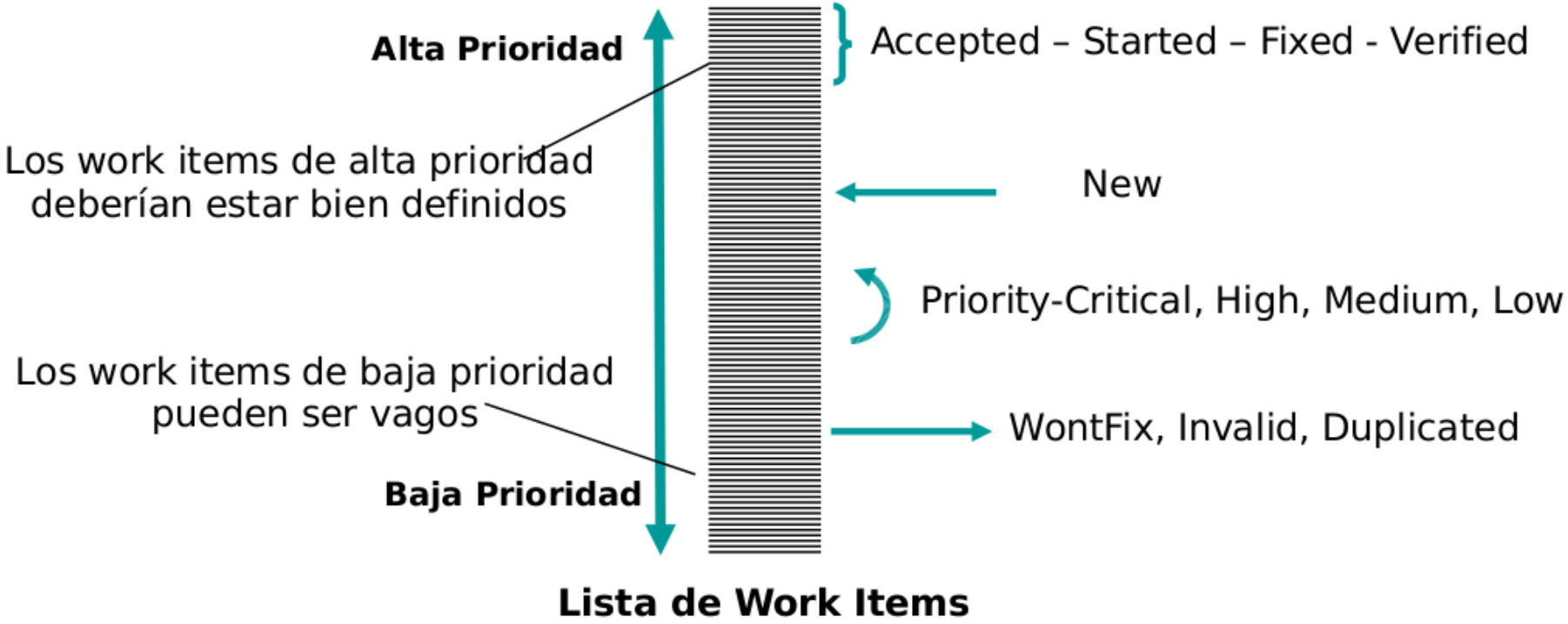



**Figure 4.1** Workflow of a change request

# Estados de un issue/ticket de GC y GI




# Índice

# Proceso general en caso de error

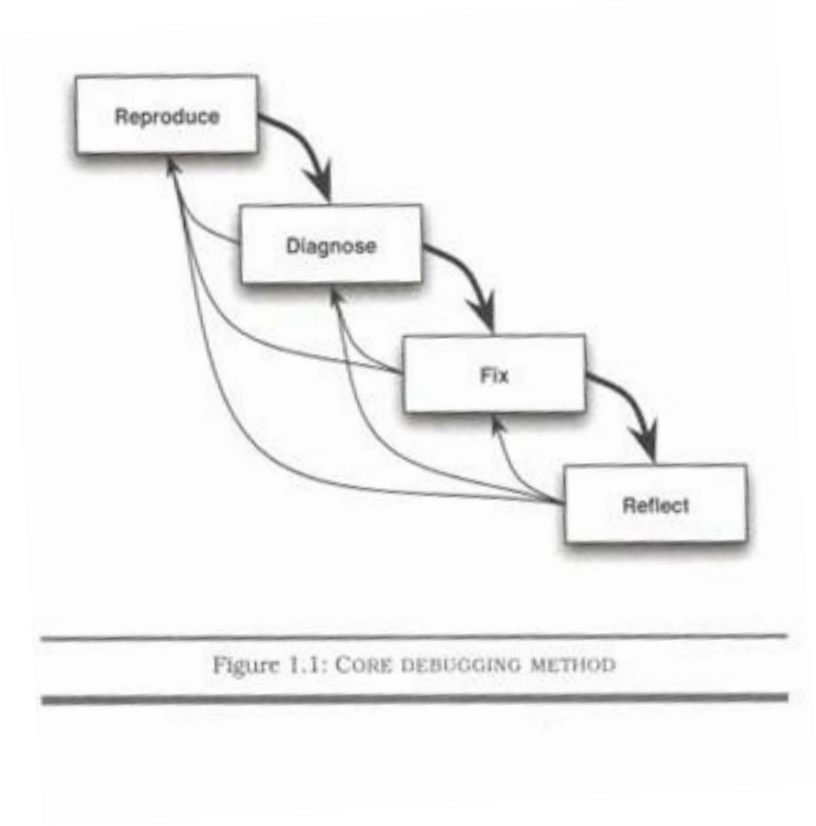



Figure 1.1: CORE DEBUGGING METHOD

- Reproducir: buscar una manera de reproducir el problema de una manera sistemática
- Diagnosticar: Construir hipótesis y validarlas haciendo experimentos
- Reparar: Diseñar e implementar cambios para solucionar el problema
- Analizar: aprender de las lecciones del error

# Índice

## Depuración

- **Informar de la incidencia**
- Reproducir
- Diagnosticar
- Arreglar
- Analizar

# Informar de la incidencia

- Antes de empezar a intentar reproducir debemos estar seguros de entender qué es lo que está pasando:
  - Necesitamos un *bug-tracking system*:

# Bug tracking system

- Nos permite:
  - No olvidar la incidencia
  - Forma estándar de informar sobre *bugs*
  - Para auditar *bugs* (e.g en las entregas)
  - Para priorizar y construir la "*work item list*"
  - Para comunicación en el equipo

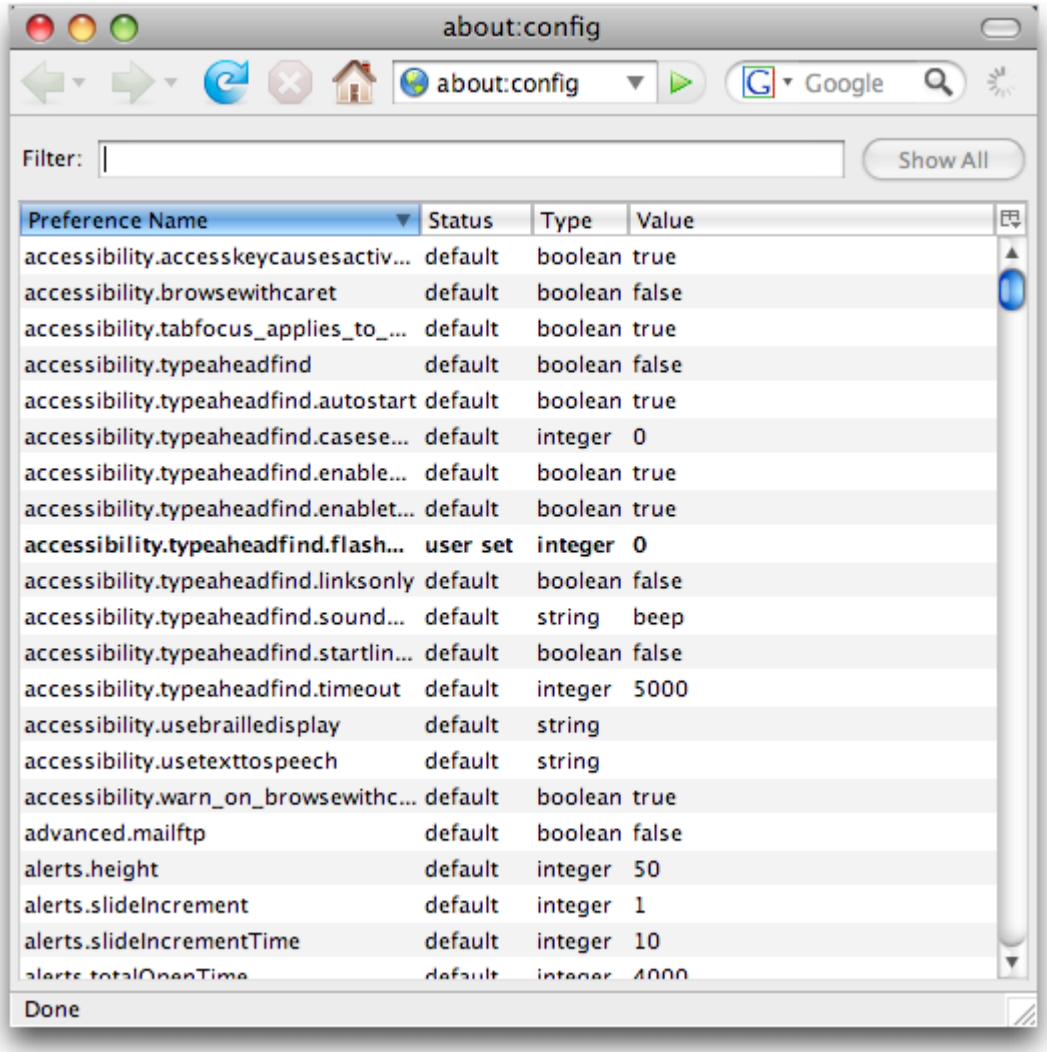
# Cómo informar de una incidencia

- Cuántos más detalles mejor
- Intentar que sea específico, sin ambigüedades, detallado.
- Debe ser único (si ya se reportó otra vez, ignorarlo o mezclarlo)

| Así no | Mejor así |
|---|---|
| • Salta una excepción cuándo añado un producto al carrito | • Al añadir un producto al carrito cuándo tengo 10 elementos dentro, salta la excepción "Overflow Exception" (copia del mensaje de error) |
| • Cuándo entro en la página de confirmar la compra no me aparece el botón de enviar | • En la página "confirmar.jsp" no aparece el botón para confirmar la compra.<br>• entorno: Internet Explorar 8 |

# Cómo informar de una incidencia

- ¿Cuánta información del entorno? Si automática, cuánto más mejor.



# Cómo informar de una incidencia

- Información de los usuarios
  - Ley de Linux:

    *"dado un número suficientemente elevado de ojos, todos los errores se convierten en obvios*"
  - Usar una plantilla sensible
  - Automatizar al máximo
  - Respetar la privacidad
  - *Asumir que de cada error que reporte un usuario, habrá entre 10 y 100 que lo hayan experimentado y no lo reporten [butcher]*

# Cómo informar de un bug



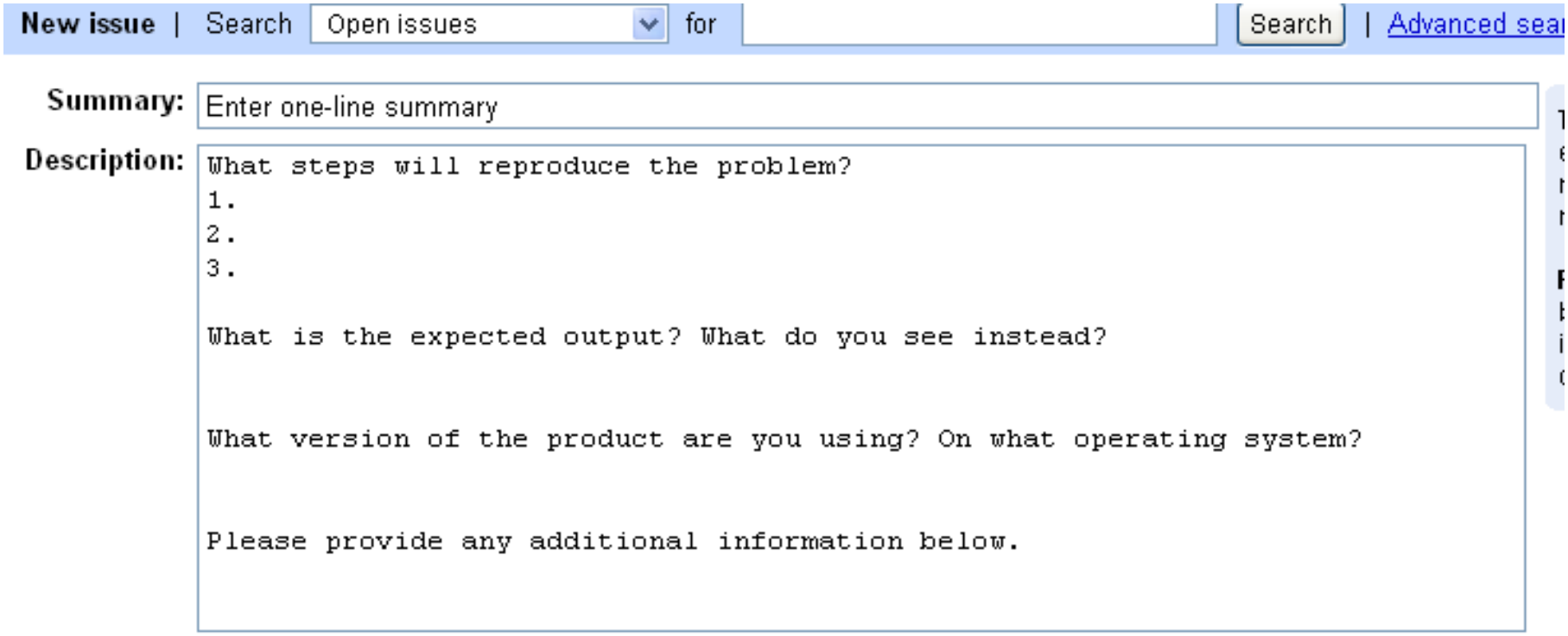
## What Makes a Good Bug Report?

Thomas Zimmermann, *Member*, *IEEE*, Rahul Premraj, Nicolas Bettenburg, *Member*, *IEEE*, Sascha Just, *Member*, *IEEE*, Adrian Schröter, *Member*, *IEEE*, and Cathrin Weiss

**Abstract**—In software development, bug reports provide crucial information to developers. However, these reports widely differ in their quality. We conducted a survey among developers and users of APACHE, ECLIPSE, and MOZILLA to find out what makes a good bug report. The analysis of the 466 responses revealed an information mismatch between what developers need and what users supply. Most developers consider steps to reproduce, stack traces, and test cases as helpful, which are, at the same time, most difficult to provide for users. Such insight is helpful for designing new bug tracking tools that guide users at collecting and providing more helpful information. Our CUEZILLA prototype is such a tool and measures the quality of new bug reports; it also recommends which elements should be added to improve the quality. We trained CUEZILLA on a sample of 289 bug reports, rated by developers as part of the survey. The participants of our survey also provided 175 comments on hurdles in reporting and resolving bugs. Based on these comments, we discuss several recommendations for better bug tracking systems, which should focus on engaging bug reporters, better tool support, and improved handling of bug duplicates.

**Index Terms**—Testing and debugging, distribution, maintenance, and enhancement, human factors, management, measurement.

# Cómo informar de un bug



# Índice

## Depuración

- Informar de la incidencia
- **Reproducir**
- Diagnosticar
- Arreglar
- Analizar

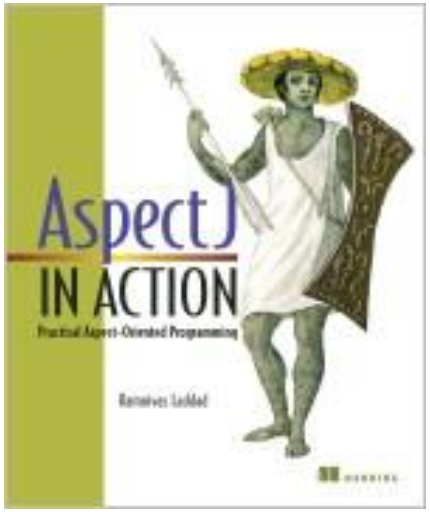
# Primer paso: reproducir

- Dos escenarios: tengo toda la información, no la tengo
- Algunas recomendaciones:
  - Intentar reproducir sin preguntar:
  - Empezar por lo simple
  - Intentar reproducirlo en el mismo entorno (máquinas virtuales)
  - Al menos tres escenarios: desarrollo, preproducción y producción.
  - Usa técnicas de pruebas
  - Si el fallo ha sido detectado usando casos de prueba, reproducir dichos casos de prueba
  - Hacer lo que aparentemente es no determinista, determinista.
  - Usar información de *log* (*log* sí *vs log* no)

# *Logging* sí vs *Logging* no

- ¿Argumentos a favor del logging?
- ¿Argumentos en contra?
  - Ensucia el código impidiendo diferenciar "las hojas de las ramas"
  - Puede tener los mismos problemas que los comentarios: tienen que tener un sincronización con los cambios en el código que según la experiencia, no siempre es así [butcher]
  - Conjetura: "No importa la cantidad de información de *log* que añadas, nunca será la que necesitas"

- Posible solución:



# Índice

## Depuración

- Informar de la incidencia
- Reproducir
- **Diagnosticar**
- Arreglar
- Analizar

# Segundo paso: diagnosticar

- Para diagnosticar software hace falta método y agilidad mental.
- Un experimento debería poder probar algo.
- Un experimento debe introducir un solo cambio.
- Anotar la traza de experimentos
- Pedir una mano al compañero/a
- Ejemplos:

Hipótesis: El error se debe a que nuestro software no soporta la versión del driver jdbc
Experimento: cambiar el driver jdbc

Hipótesis: Hay una variable que no se inicializa o se inicializa a null.
Experimento: inicializar esa variable con un valor distinto de null.

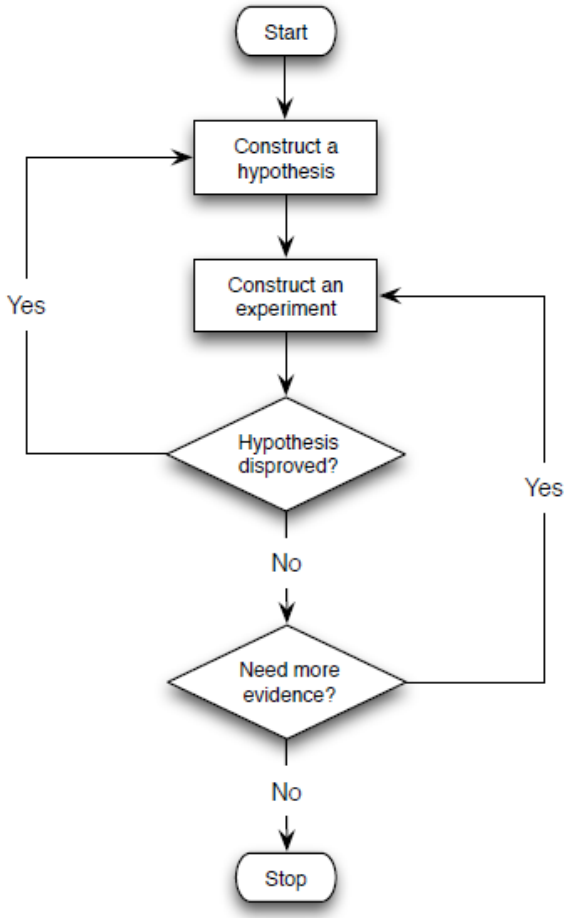



Figure 3.1: A Debugging Method

# Índice

## Depuración

- Informar de la incidencia
- Reproducir
- Diagnosticar
- **Arreglar**
- Analizar

# Tercer paso: arreglar

- Arreglar no significa dejar el sistema en el último estado del último experimento
- Primero hay que volver a empezar en el estado inicial.
- Usemos la función *diff*
- Pasos para arreglar:
  - Pasar las pruebas anteriores
  - Añadir pruebas para este nuevo fallo
  - Ver que la prueba falla
  - Arreglar
  - Demostrar que funciona (pasa las pruebas)
  - Hacer pruebas de regresión

# Tercer paso: arreglar

- Es importante arreglar las causas no los síntomas: si hay un valor de una variable que parece ser incorrecto, no cambiar el valor de la variable, pensar por qué ha llegado ese valor incorrectamente.
- No intentar arreglar y refactorizar al mismo tiempo
- Subir el cambio a “preproducción” o enviar como un “patch” y documentar y gestionar en el “work item list”

# Índice

## Depuración

- Informar de la incidencia
- Reproducir
- Diagnosticar
- Arreglar
- **Analizar**

# Cuarto paso: analizar

- Intentar aprender de lo corregido.

> Tu mejor maestro es tu último error.
> Ralph Nader (1934-?) Activista y abogado de USA

- Las etapas de corrección de errores:
  1. No puede ser
  2. No sucede en mi máquina
  3. No debería pasar
  4. ¿Por qué está pasando?
  5. Ah, ¡ya!
  6. **¿Cómo podía estar funcionando?**

*"The six stages of debugging"*

# Índice

# Resumen

- ¿Qué hemos aprendido?
  - El cambio es inevitable
  - Si no se gestiona bien puede haber muchos problemas
  - Hay que gestionar las incidencias de manera ordenada con estados predeterminados
  - Informar de un error es una tarea crucial
  - Para depurar hace falta método y estrategias
- ¿Qué veremos en las siguientes lecciones?
  - En las prácticas pondremos ejemplos y ejercicios reales de depuración y gestión de incidencias

# Índice

# Bibliografía