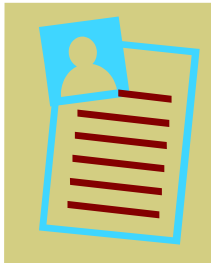
escuela técnica superior
de ingeniería informática

Gestión de incidencias y depuración en el ciclo de integración continua

Departamento de
Lenguajes y Sistemas Informáticos

EGC

Ejemplo de otro dominio



INCIDENCIA

INFORME

DEPURACIÓN, REPARACIÓN

Índice

¿Creéis que en Decide 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

- Problemas con la subred en el docker
- Problemas con las versiones de paquetes
- 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
- Problemas con las sesiones, los usuarios y el tally
- Han cambiado las reglas de negocio
- …

El tiempo que empleamos

¿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)

Incapacidad del sistema
Para realizar una función:
"los síntomas"

Fault (bug)

Aquello que provoca el
Fallo: "las causas"

Error

Estado interno de un sistema que contiene un "bug".:
"La enfermedad"

Debug. Detectar, localizar y corregir *bugs* en un programa. *"proceso de curación"*

Índice

Proceso de GC y GI

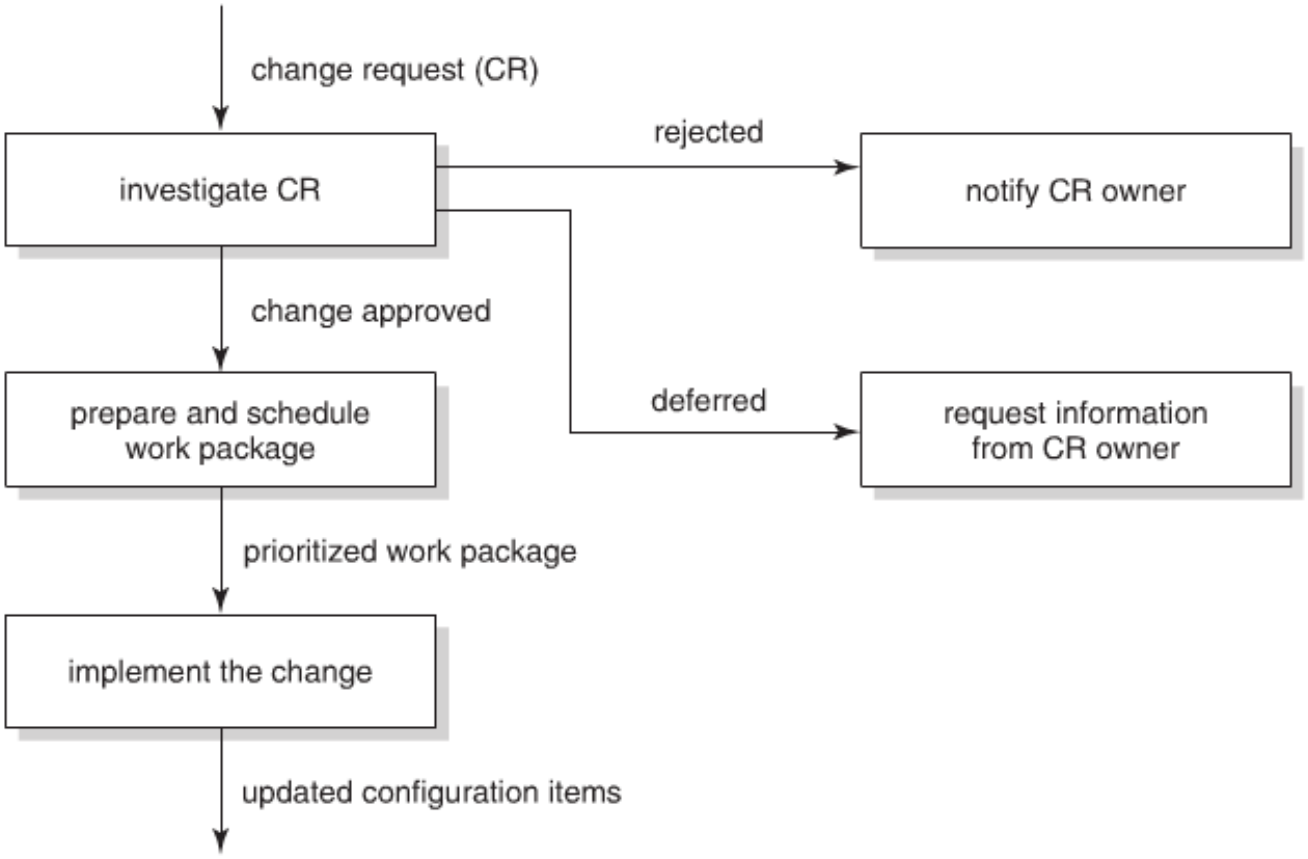
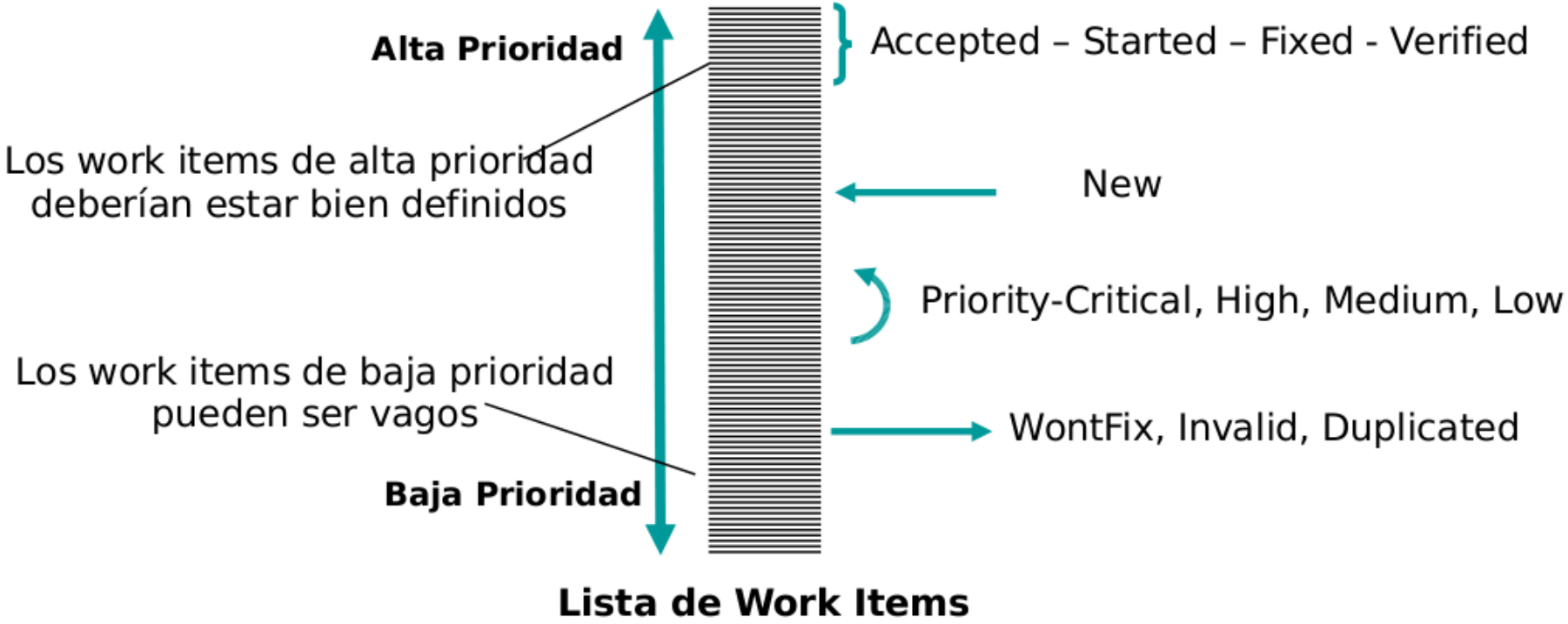
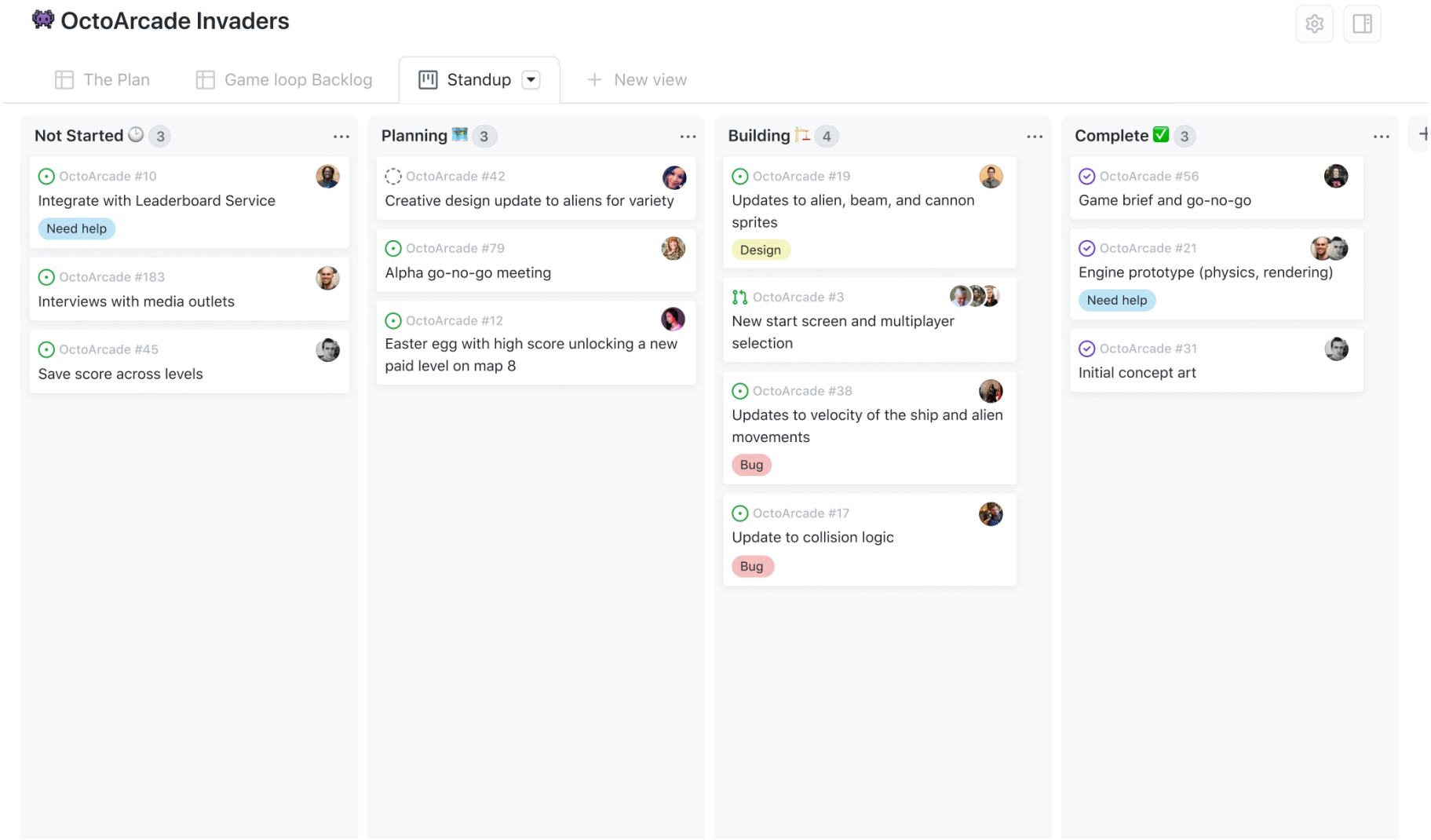


Figure 4.1 Workflow of a change request

Estados de un issue/ticket de GC y GI



Ejemplo de tablero de issues



Todo proceso de gestión de incidencias debe al menos tener:

- Prioridad de la incidencia
- Estado en la que se encuentra
- Tipo de incidencia
- Roles en la gestión de dicha incidencia

Índice

Proceso general en caso de error

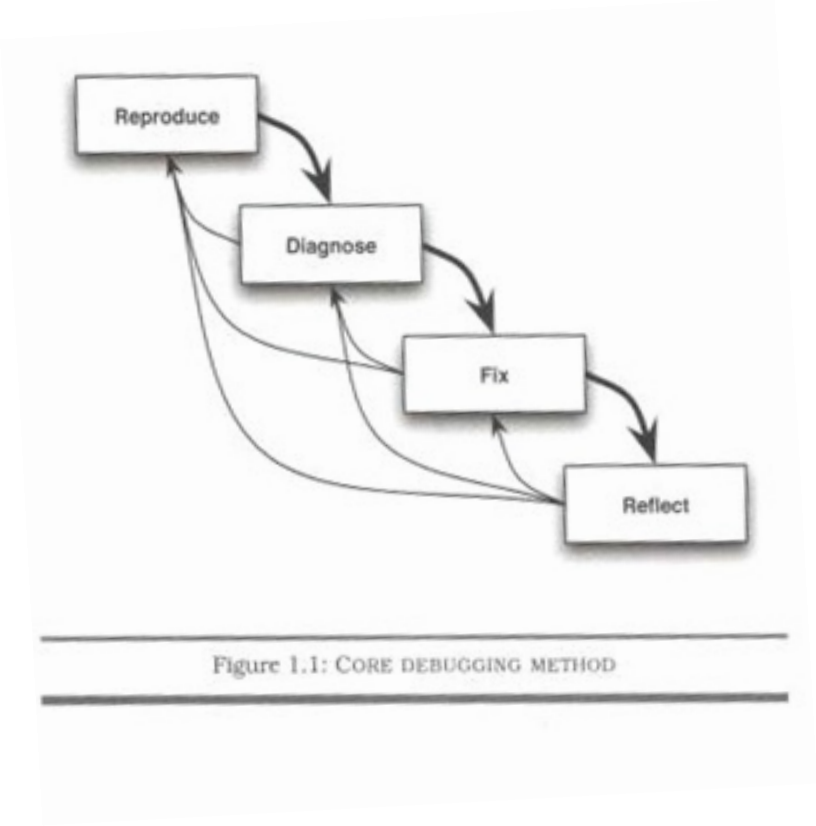


Figure 1.1: CORE DEBUGGING METHOD

- 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

Ley de Linus:

"dado un número suficientemente elevado de ojos, todos los errores se convierten en obvios“

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 incidencias
  - Para auditar *bugs* (e.g en las entregas)
  - Para priorizar y construir la "*work item list*"
  - Para comunicación en el equipo
  - Herramienta de gestión y toma de decisiones

Todo proyecto de software debe contar con un sistema de gestión de incidencias

Bug tracking system

Pero un sistema de gestión de incidencias es el medio, no el fin. Debe haber buenas prácticas al usarlo.

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 pulso el botón de confirmar la votación	Al pulsar el botón de votación en una votación de selección múltiple con el navegador Firefox versión XX, salta la excepción “Overflow Exception” (copio mensaje de error)
Cuando entro en la sección de visualizar los resultados, uno de los elementos a visualizar me sale con símbolos raros	Al intentar visualizar los resultados de la votación XXX, en los resultados de la repuesta YYY sale con un número de caracteres así: C%%%&&&

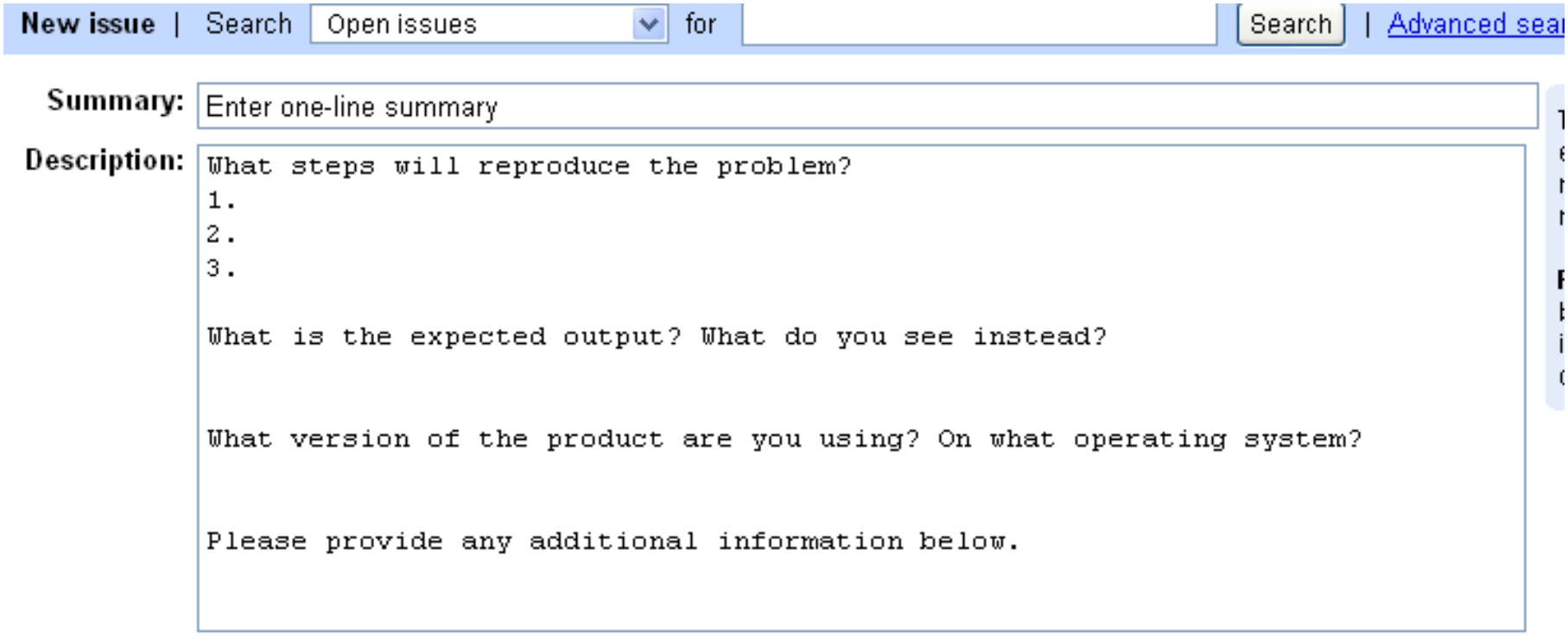
Cómo informar de una incidencia

Información de los usuarios

- Usar una plantilla
- Automatizar al máximo

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



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.

Índice

Depuración

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

Primer paso: reproducir

- Dos escenarios: tengo toda la información, no la tengo
- Algunas recomendaciones si no la tengo:
  - 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)

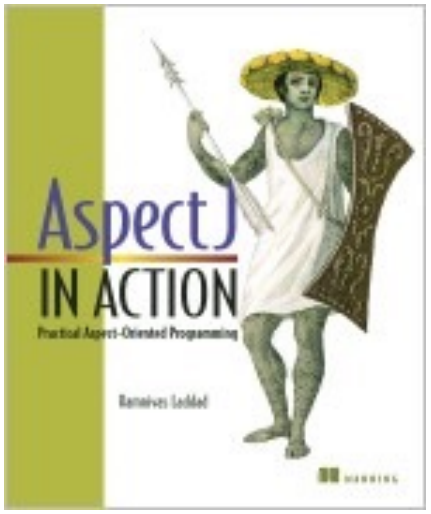
Bug tracking system

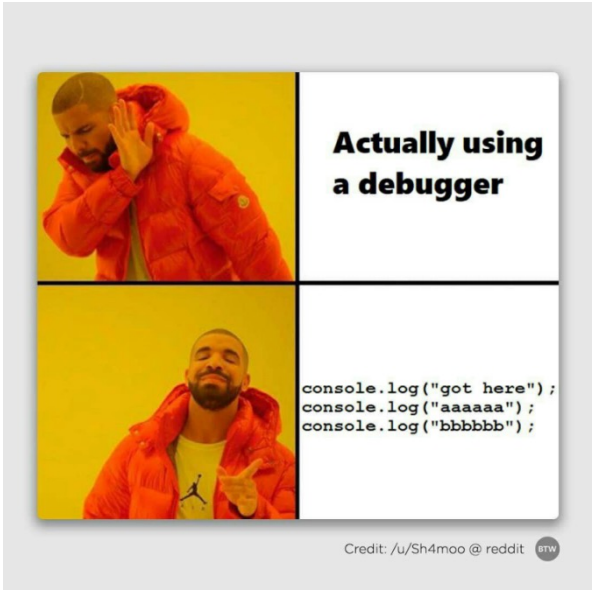
¿Logging sí o no?

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





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.

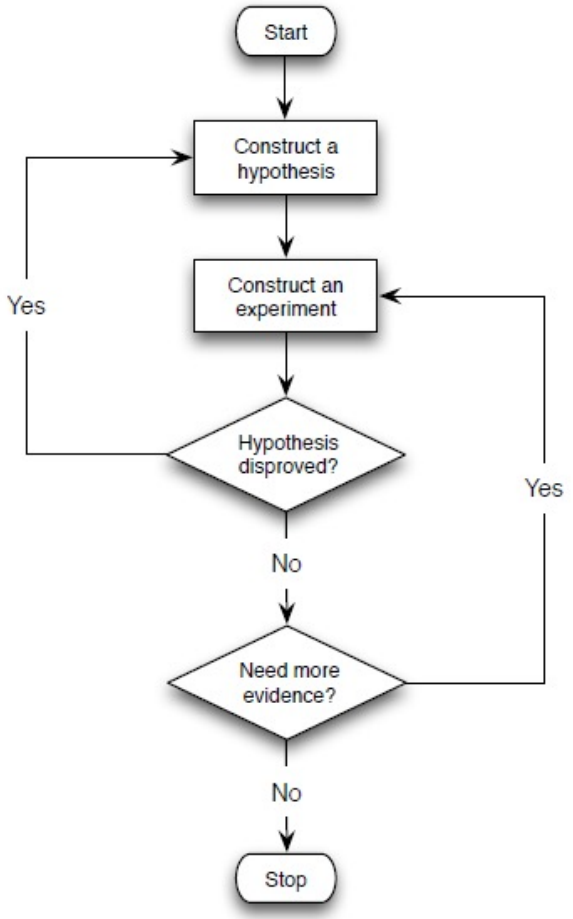


Figure 3.1: A Debugging Method

Un buen método para construir hipótesis

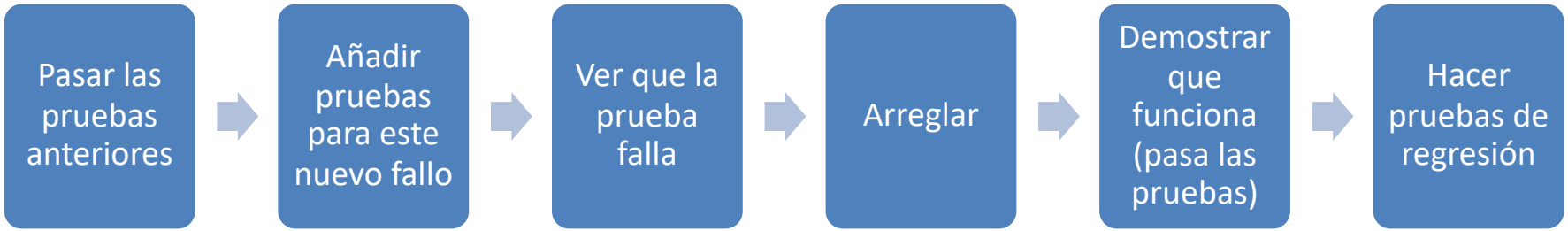
Índice

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 y cherrypick*
- Pasos para arreglar:



- Subir el cambio a "preproducción" o enviar como un "patch" y documentar y gestionar en el "*work item list*"

Tercer paso: arreglar

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

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, con tipos de incidencias, con prioridades y con roles
  - Informar de un error es una tarea crucial
  - Para depurar hace falta método y estrategias

Índice

Bibliografía