Depurar

José Galaviz

Premisa

Un programa no trivial tiene, al menos, un error (bug)

Referencia

- Beizer, Boris, *Software Testing Techniques,* 2a Ed., Dreamtech, 1990.
- Establece una clasificación de errores de programación.
- Más tarde: IEEE Standard Classification for Software Anomalies (IEEE Std 1044, 2009)

Clasificación

1. Requirements and Features.
2. Functionality as Implemented.
3. Structural Bugs.
4. Data.
5. Implementation (standards violation, and documentation).
6. Integration.
7. System and Software Architecture.
8. Test Definition or Execution Bugs.
9. Other Bugs, Unspecified.

Tipos (1):

- Requirements and Features. Errores durante la etapa de captura de requisitos.
- Functionality as Implemented. La implementación no satisface los requisitos funcionales.
- Structural Bugs. Errores relacionados con la estructura del programa.

Tipos (2):

- Data. Errores en la definición, estructura o uso de los datos.
- Implementation. Violaciones en las convenciones de codificación, formatos, estilos y en la documentación.
- Integration. Errores vinculados con la integración de componentes, con la compatibilidad de sus interfaces.

Tipos (3):

- System and Software Architecture. Errores no directamente relacionados con el programa, sino con su plataforma de ejecución.
- Test Definition or Execution Bugs. Errores en la definición, diseño y ejecución de las pruebas.
- Other Bugs, Unspecified.

Estadísticas

Categoría	% Vinter	% Beizer
Requirements	**23.5**	8.1
Functionality	**24.3**	16.2
Structural	**20.9**	25.2
Data	**9.6**	22.4
Implementation	4.3	9.9
Integration	5.2	9.0
Architecture	0.9	1.7
Test	6.9	2.8
Otros	4.3	4.7
Total	100	100

Más

- El 85% de los errores se pueden corregir sin modificar más de una rutina.
- Cerca del 36% de los errores del programa son en realidad errores de escritura (*typos*).
- 17% de los errores se deben a un entendimiento incorrecto del diseño.
- Corregir: 85% toman unas cuantas horas, 14% de unas horas a unos días, 1% requieren más tiempo.

Y más

- Promedio de la industria: 1 a 25 errores por cada 1000 líneas de código (a la entrega).
- Microsoft se precia de tener 10 a 20 en producción y 0.5 en entrega.

Lo mejor que podemos hacer con los errores
es no cometerlos.

(pero somos humanos)

Y bien… nos encontramos un error

- Eso significa que obtuvimos un resultado que no era el esperado.
- Presupone que sabemos cómo debería funcionar.
- A depurar (*debugging*).

Premisa

Tu NO quieres corregir el error

Si rompes el fondo del balde,
ciertamente ya no verás el agua
escapar por el agujero en su lado.
Pero habrás perdido un balde
y no sólo una carga de agua.

-- Lao Tse

Lo que realmente quieres

- Es saber por qué se produce el resultado erróneo.
- Corregir es sólo una consecuencia.

La esencia de depurar

- Proceso de búsqueda detallado.
- Heurística.

Proceso sistemático que permita reducir lo más rápido posible el espacio de búsqueda

¿Con qué contamos?

- El código fuente.
- Los datos de entrada de las pruebas.
- Los resultados de las pruebas.
  - Esperados (*a priori*).
  - Obtenidos (*a posteriori*).
    - Correctos (obtenidos == esperados).
    - Incorrectos (obtenidos != esperados).
- La correspondencia entre los datos de entrada y los de salida.

Proceder

- Búsqueda más rápida: búsqueda binaria.
- Acotar el código que engendra el error.
- Cerrar el cerco a su alrededor.
- Usando el método científico.

Método científico

- Hipótesis: “Yo creo que el error se presenta siempre que ocurre… en los datos de entrada”
- Diseño experimental para confirmar o refutar la hipótesis.
- Conjunto de datos de entrada mínimos que hacen que el error aparezca.
- Refinar la hipótesis.

Consejos

- Conservar la versión previa a la sesión de depuración en lugar seguro.
- Salvar frecuentemente., control de versiones.
- Bitácora de trabajo.
- Cansancio.
  - Uno acaba estropeando más el programa.
  - Andando en círculos.

Durante la depuración

Los novatos añaden código correctivo.

Los expertos remueven código defectuoso.

Un poco de psicología

- Para el programador el código que escribe es una extensión de sí mismo.
- Aceptar que tiene errores es difícil: humildad.
- Cuando corriges tú programa te corriges a tí mismo.
- Introspección.

Cada error es tu amigo

¿Qué estaba yo pensando cuando escribí el código equivocado?, ¿qué creía?, ¿es un patrón recurrente?

¿Desesperado?

Antes de arruinar más el código...

- Déjalo un rato, al menos 15 minutos.
- Haz otra cosa.
- Toma agua, come algo.
- Duermete.
- Explícale el código a un cuate.

Código erróneo, código feo

- El código equivocado normalmente es el más feo y tú sabes cuál es.
- Que está enredado: que tiene muchos if o tres for anidados con breaks.
- Que viola la encapsulación, que accede a variables globales.
- Que es largo.

Al final

- Ya que corregiste el error.
- Ya que reflexionaste acerca de tí.
- Limpia el mugrero.
  - Los comentarios de bitácora.
  - ¿Los comentarios corresponden?
  - Las variables de verificación, los printf.
  - Reacomoda, borra lo que no usas.

Herramientas

- Opciones del compilador: los warnings cuentan (`-Wall`).
- Escribir mensajes y contenidos de variables:
  - `stderr`, compilación condicional (`ifdef`).
  - logging.
- Análisis estático de código: `cppcheck`.
- Análisis dinámico: `valgrind`.
- Debugger: `gdb`, `ddd`.
- **LEER el código.**