Errores de la programación: de los bugs a las alucinaciones
Autor: Kimi (Kimi-ai)
Fecha: 20 de septiembre de 2025

---

1. Issue de programación: la unidad de trabajo

En repositorios modernos *issue* es el término genérico que abarca bugs, mejoras, tareas y preguntas. Sirve para **planificar, discutir y rastrear** cualquier trabajo futuro; no implica necesariamente que el código esté roto .

---

2. Bug: el error que ya se ve

Un **bug** es una **manifestación observable** de un fallo: el programa se desvía de lo especificado. Ejemplos clásicos: desbordamiento de búfer, variable no inicializada, condición de carrera o referencia nula . El bug es solo **un subtipo** de *issue*.

---

3. ¿"Issue" es la forma elegante de decir error?

Sí, y además **más precisa**. Decir *"abro un issue"* evita culpabilizar al autor; deja abierta la posibilidad de que el problema sea documentación, diseño o entorno. Herramientas como SonarQube clasifican *issues* en **Bug, Vulnerabilidad, Code Smell, Hotspot**, cada uno con severidad (Blocker → Info) .

---

4. Alucinaciones de la IA: código que nunca existió

Los modelos de lenguaje pueden **inventar APIs o símbolos** que parecen válidos pero no existen; por ejemplo:
Python
复制

```
df = load_csv("sales.csv")   # función alucinada
```

Este **error inducido por IA** compila pero falla en tiempo de importación. Las causas: datos de entrenamiento sesgados, recuperación defectuosa o sobre-ajuste . **Hasta el 46 % de los textos generados por IA contienen alucinaciones** .

---

5. Pair Programming IA-Humano y SQA

El humano actúa como *driver/navigator* y la IA como *suggester*. El ciclo de aseguramiento de calidad (SQA) se vuelve:

1.*Prompt engineering* (contexto)
2.*Acceptance gate* (compila + pruebas)
3.*Exploratory testing* (casos límite)

4.*Refactor & document*
Cuando las sugerencias se revisan siempre, los defectos se reducen en 55 % .

---

6. Vibe Coding Cleanup Specialist: la nueva especialidad

Vibe coding: dejar que la IA escriba archivos completos sin supervisión. Produce prototipos rápidos, pero también **código basura** (duplicado, sin pruebas, con APIs inventadas).
Nace el **Cleanup Specialist**:

- **Skills**: arqueología de prompts, detección de alucinaciones, poda de código muerto.
- **Herramientas**: linters con hooks de IA, dashboards de deuda, explicadores de diff.
- **KPIs**: tasa de alucinaciones ↓, deuda técnica ↓, tiempo de reparación ↓.

Se prevé que en 24 meses aparezcan ofertas de trabajo para *"AI Code Janitor"*.

Conclusión

Los errores evolucionan: de tipos humanos a alucinaciones de máquina. El antídoto sigue siendo **revisión rigurosa**, ahora reforzada por **pair programming con IA** e **ingenieros especializados en limpieza**. Aceptemos la velocidad, pero **nunca saltemos la segunda pasada de ojos**.

References (English sources only)

Wikipedia, *Software error*, 2003-08-04. https://en.wikipedia.org/wiki/Software_bug
UDIT, *What are AI hallucinations?*, 2025-02-13. https://www.udit.es/que-son-las-alucinaciones-de-la-ia/
Google Cloud, *What are AI hallucinations?*, 2025. https://cloud.google.com/discover/what-are-ai-hallucinations?hl=es-419
SonarSource, *SonarQube Rules*, 2025. https://rules.sonarsource.com
Latenode, *What are AI hallucinations and how to avoid them?*, 2025-06-12. https://latenode.com/es/blog/what-are-ai-hallucinations-and-how-to-avoid-them
Testing IT, *5 Most Common Software Errors*, 2024-09-12. https://www.testingit.com.mx/blog/errores-de-software