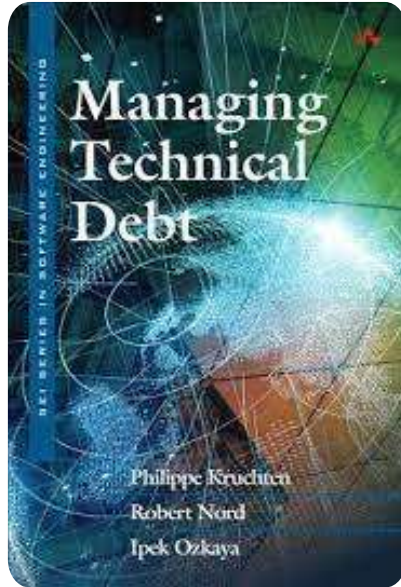


Episodio 481



Ipek Ozkaya

Managing Technical Debt:
Reducing Friction in Software
Development



¿Deuda técnica?

“Refleja el costo implícito del retrabajo adicional causado por elegir una solución fácil en lugar de utilizar un enfoque que llevaría más tiempo en su desarrollo e implementación”

¿Por qué deberíamos preocuparnos por la deuda técnica?

Managed
technical
debt?

¿No sería mejor simplemente eliminarla por completo?

¿"Administrarla"?



Nueve principios para la gestión de la deuda técnica



Principio 1:

La deuda técnica cosifica un concepto abstracto.

- Cosificar = hacer concreto.
- ¿Qué acciones se podrían llevar a cabo? ¿Cómo?
- Analizar, evaluar y tomar decisiones en base a esas acciones para gestionarla.

Principio 2:

Si no incurre en ningún tipo de interés, entonces probablemente no tenga una deuda técnica real.

- Se parte de una deuda técnica potencial y, en un futuro, observando las consecuencias se distingue si es una deuda técnica real o no.

Principio 3:

Todos los sistemas tienen deuda técnica.

- Son complejos y evolucionan de maneras que no anticipamos.
- Un sistema SW mal administrado, no sabrá cómo evaluar realmente esa deuda técnica.

Principio 4:

La deuda técnica debe rastrearse hasta el sistema.

- Lo que quiere decir este principio es que deberíamos ser capaces de señalar en que parte del sistema tenemos que rehacer el trabajo si no gestiono esa deuda técnica.

Principio 5:

La deuda técnica no es sinónimo de mala calidad.

- Siempre hay deuda técnica pero lo que hace que haya mala calidad es no saber gestionarla o saber en qué partes puede haber deuda sin perjudicar la calidad.

Principio 6:

La arquitectura tiene un mayor coste debido a la deuda técnica.

- Aquí la deuda se refiere a la construcción de la arquitectura y el diseño. Al haber aspectos acumulados de diseño se convierten en capas sobre las que se implementan otras y por tanto cuesta más arreglarlo.

Principio 7:

Todo el código importa.

- Invertir código en tests.
- El código ajeno es importante.
- Ejemplo de la API externa y desastre de Columbia.

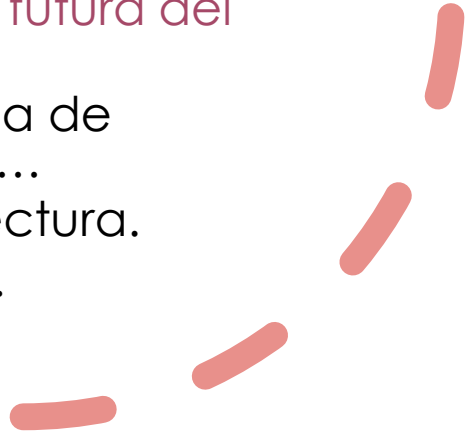
Principio 8:

La deuda técnica no tiene una medida absoluta.

- Difícil de medir ya que los sistemas evolucionan rápidamente.
- Podemos medir ciertas consecuencias, pero no comparar con otros sistemas.
- Actualmente, ya se usan herramientas de medición.

Principio 9:

La deuda técnica depende de la evolución futura del sistema.

- ¿Depender del futuro sin saberlo? Toma de decisiones en planificación, desarrollo...
 - Evolución del sistema, no de la arquitectura.
 - Equipos pequeños vs Equipos grandes.
- 

Conclusión



Aunque puede ser tentador ignorar la deuda técnica a corto plazo, a largo plazo puede resultar en un software menos estable, seguro y escalable.



Es por ello muy importante que los desarrolladores trabajen para identificarla y gestionarla.

Así, pueden garantizar un software de alta calidad y más sostenible a largo plazo.