MANAGING TECHNICAL DEBT

Episodio 481

Software Engineering Radio

María Teresa Fernández Coro, Pablo Fernando Urones Clavera
y Pedro García-Cañal Sánchez.

Contenido

Ipek Ozkaya

Invitada del programa, es la coautora del libro *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?

Ipek habla de cómo los desarrolladores tienden a preocuparse por la nueva funcionalidad que se encuentran desarrollando, sus defectos, problemas de seguridad, etc.; Y, sin embargo, tienden a no preocuparse tanto por los problemas fundamentales de diseño que realmente les afectan a largo plazo; Es decir, la deuda técnica.

Si esta no se gestiona de forma continuada en el tiempo y se ignora, cuando quieran o tengan que lidiar con ella, porque será necesario que lo hagan de una forma u otra, puede que sea demasiado tarde.

Managed Technical Debt?

Lo que se podría traducir al español como *administración* o *gestión* de la deuda técnica.

¿Es necesario realmente? ¿No sería mejor simplemente eliminarla por completo?

Como bien explica Ipek en la entrevista, esta no puede eliminarse. O se tiende a gestionarla y tratarla a medida que se desarrolla un sistema software o se acumulará y será mucho más compleja de gestionar.

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 conviertes 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

La deuda técnica es un concepto fundamental en el mundo del desarrollo de software. Si bien puede ser tentador ignorarla a corto plazo, a largo plazo puede resultar en un software menos estable, menos seguro y menos escalable.

Es importante que los desarrolladores y líderes de equipos comprendan su importancia y trabajen para identificar, priorizar y abordarla de la manera conveniente. Al hacerlo, pueden garantizar un software de alta calidad y más sostenible a largo plazo.