# Behavioral Code Analysis

David Maldonado Álvarez

Jorge Toraño Herrera

Álvaro López Fueyo

La deuda técnica no hace referencia únicamente a cuestiones técnicas. El análisis de comportamiento nos ayuda a suplir estas carencias, descubriendo otras partes de la deuda técnica.

El análisis estático del código solamente nos dice los síntomas que tiene nuestro código base, pero no sus causas. Si tenemos código con aspecto terrible, de nada sirve refactorizarlo si va a volver a su estado anterior. Hemos de centrarnos en el motivo por el cuál llegó a ese punto.

No hemos de analizar archivo tras archivo como piezas individuales sino como una red a través del análisis de comportamiento.

## La deuda técnica no es técnica.

La deuda técnica se encarece con el paso del tiempo a no ser que la solventemos de forma recurrente. Normalmente es generada para garantizar que el software cumpla ciertos requisitos a corto. El problema es que se dejan de lado posibles cambios que puedan surgir en el futuro.

## Técnicas de análisis de código de comportamiento

La cantidad de cambios en un archivo es un buen indicador de la disminución de su calidad, lo que acaba ocasionando errores. Los lugares donde más trabajamos es donde la refactorización acabará teniendo un mayor efecto.

### Puntos calientes

Segmentos del programa propensos a sufrir errores. Suelen ser el resultado de código complejo y activo, zonas del programa en las cuales se pasa la mayor parte del tiempo de ejecución.

### Tendencia de la complejidad

Luego, podemos comenzar a determinar la tendencia de complejidad de cada posible objetivo de refactorización. Podemos categorizar si el punto de acceso está activo, inactivo o extinto.

## Cambio de acoplamiento

Es lógica compartida entre dos funciones o archivos en la que, si cambia uno de ellos, probablemente tendrá que cambiar el otro. Se trata de un acoplamiento invisible, implícito en el código. Se puede observar su presencia al analizar los cambios sufridos en el código a lo largo del tiempo.

# El problema no es técnico, es social.

Siempre habrá deuda técnica. Lo importante es cómo lo enfrentamos.

## Arregle la causa, no el síntoma

No hemos de tomar malas prácticas como costumbre. Cuantos más elementos tiene un equipo, más difícil acaba siendo coordinarlo y es posible que aparezcan fallos de comunicación, y la consecuente pérdida de proceso, explicada en dos aspectos principales:

Ignorancia Pluralista. Sucede porque cada persona siente que es la única que piensa de esa manera y tiene miedo de ser juzgada por el resto. Entonces todos fallan en actuar.

Difusión de la Responsabilidad. El estado de una persona, de sentirse menos responsable, menos responsable en grandes grupos. La responsabilidad se diluye, del individuo al grupo, pero si todos se sienten así, entonces nadie dará cuenta de esa responsabilidad.

Más que centrarnos en qué refactorizar, debemos prestar mucha atención a la organización y las prácticas que seguimos. Practicar la revisión del código y tener un mantenedor principal tiene efectos positivos en la calidad del código.