BEHAVIORAL CODE ANALYSIS

Yago Fernández López – UO289549

Rubén Pérez Dafonte – UO289337

Análisis de Comportamiento de Código

Introducción

El Behavioral Code Analysis (Análisis de Comportamiento de Código) es un tipo de análisis que combina el aspecto técnico (código) con el social (las personas y como se organizan). Básicamente, el análisis de comportamiento de código se encarga de analizar el código desde la perspectiva de las personas en busca de: deudas técnicas, hotspots, fallos en la organización de los equipos, etc.

Base

Como mencionamos antes, el análisis de comportamiento de código busca como interaccionan las organizaciones con este; es una mezcla de aspectos sociales y técnicos. Este tipo de análisis nos permite comprender mejor como está hecho nuestro código y como interactuamos con él. Además, también nos permite identificar ciertos problemas del código como: deuda técnica, complejidad en el código o hotspots.

Deuda Técnica y Hotspots

Ahora bien, ¿Qué es la deuda técnica y los hotspots? La deuda técnica es, resumidamente, aquellos aspectos del código que hacemos mal, voluntaria o involuntariamente, y que pueden acarrear problemas. Los hotspots son trozos de código con una alta complejidad, que requieren mucho trabajo de desarrollo para mejorarlos. El análisis de comportamiento de código es muy bueno detectando este tipo de errores.

Organización de los Equipos

El análisis de comportamiento de código ayuda a los equipos a organizar mejor. Es muy habitual que haya problemas de coordinación en equipos grandes, en los que un equipo debe esperar a que otro acabe para realizar algún cambio; sin embargo, esto no siempre es visible fácilmente. Este tipo de análisis permite identificar estos problemas y adecuar el trabajo de los equipos para evitarlos.

Se puede utilizar tanto en proyectos Green Field (nuevos) como Brown Field (heredados). En los primeros, es especialmente importante hacer que la arquitectura se base en la

organización de los equipos, y no al revés. En este aspecto, también hay que tener en cuenta a la empresa u organización con la que trabajamos, ya que estas suelen, sin querer, impedir que dicha organización sea efectiva, poniendo fechas límites inasumibles o eliminando equipos con un alto conocimiento en una parte del código crucial.

Familiaridad y Brechas de Conocimiento

Este análisis permite observar e identificar las brechas de conocimiento del equipo. Una brecha de conocimiento es la diferencia entre el conocimiento necesario y del que disponemos para hacer algo. Esto acaba distorsionando la visión del código del equipo y dificulta escribir software de calidad.

Integración en el Ciclo de Vida del Software

Es muy recomendable integrar el análisis de comportamiento de código en el ciclo de vida del software, sobre todo al hacer pull request o en reuniones para evitar acumular deuda técnica o crear hotspots. Este tipo de análisis también permite encontrar deuda técnica en los test, que es un proceso crucial del ciclo de vida del software; esto último es especialmente importante en proyectos Legacy.

Ética y Moral

Debido al carácter social del análisis de comportamiento de código, hay una serie de cuestiones morales y éticas para tener en cuenta; este tipo de análisis se puede utilizar para clasificar a los desarrolladores. Esto puede causar: problemas de rendimiento, problemas dentro de los equipos, distorsión de las métricas, etc.

Regla del Boy Scout

“Deja el código más limpio de como te lo encontraste”; bueno, no siempre. Si la zona de código que se va a modificar es estable o no suele cambiarse a menudo, se puede empeorar el código un poco. Por el contrario, si dicha zona recibe cambios recurrentes y es propensa al cambio, es recomendable aplicar esta regla, ya que facilitará los futuros cambios. El análisis de comportamiento de código nos ayuda a diferenciar entre ambas zonas.

El Futuro

El futuro del análisis de comportamiento de código pasa por la integración de la IA y, en concreto, del Machine Learning, dentro de este. Esto facilitará los análisis y permitirá ahorrar tiempo a los desarrolladores, aunque dichos análisis deberán contar con una supervisión humana.

Los Límites Actuales

Como este tipo de análisis es bastante reciente, tiene algunos límites. El primero es que no todas las herramientas que permiten realizarlo están disponibles para todos los sistemas de control de versiones. El segundo es que la IA todavía no se ha integrado plenamente con este. El último es que este tipo de análisis todavía no es capaz de detectar con la suficiente seguridad los change coupling, componentes que cambian a la vez debido a las dependencias entre estos.

Herramientas

Las herramientas más útiles para realizar el análisis de comportamiento de código son:

- Code Maat: Una herramienta open-source que permite ver la evolución del softwate[1].
- CodeScene: Existe una Community Edition que permite realizar análisis de comportamiento de código.
- Git: También se pueden realizar este tipo de análisis mediante línea de comandos en Git.

Conclusión

El análisis de comportamiento de código es un tipo de análisis especialmente importante hoy en día, donde podemos encontrar múltiples equipos trabajando en un proyecto conjuntamente. Este análisis permite identificar problemas con bastante antelación, así como mejorar la organización de los equipos y, en última instancia, la eficiencia de estos y la calidad del software.

Bibliografía

https://se-radio.net/2023/03/episode-554-adam-tornhill-on-behavioral-code-analysis/

https://towardsdatascience.com/behavioural-code-analysis-65a226e1a601

https://codescene.com/blog/validation-for-behavioral-code-analysis/

https://www.adamtornhill.com/code/codemaat.htm

[1] Si alguien está interesado, el repositorio de Code Maat es el siguiente: https://github.com/adamtornhill/code-maat