Behavioural code análisis

¿Qué es el Análisis de Código de Comportamiento?

Mientras que el análisis de código estático nos da las claves de lo que falla en cada archivo individualmente, el análisis de código de comportamiento identifica patrones en la forma en la que se interactúa con el código que se está realizando. Si las propiedades del código son importantes, es aún más valioso saber cómo llega el código a dicho estado y sus tendencias. Permitiendo esto, priorizar la deuda técnica, detectar dependencias invisibles en el código y medir otros factores importantes.

Las métricas como la frecuencia de cambio o el acoplamiento son claves para saber si nuestro código está mal. Y este tipo de métricas no se pueden obtener analizando archivo por archivo individualmente, si no que se deben analizar como una red a través del análisis de comportamiento.

Motivación

- ¿Cómo mantener grandes sistemas complejos?
- ¿Qué piezas necesitan ser refactorizadas?¿Cómo identifico las más urgentes?
- ¿Qué partes es más probable que contengan errores?
- ¿Cuáles son las deudas técnicas y cómo comunicarlas a las partes interesadas?
- ¿Qué métricas podemos usar para apoyarnos?

La deuda técnica

La deuda técnica se encarece con el tiempo a menos que se vaya "pagando" recurrentemente. Se asume para poder garantizar las expectativas a corto plazo, pero nos crea una desventaja de cara al futuro. Cuanta mayor sea la deuda técnica más cerca estaremos del punto en el que sea demasiado costoso integrar nuevas funciones.

El análisis de código de comportamiento nos permite, en cierto modo, priorizar cuál de nuestra deuda técnica es más importante "pagar".

El problema no es técnico, es social

Cuantos más elementos tiene un grupo, más difícil es coordinarlo y es más probable que falle la comunicación. Además, al trabajar en grupo nuestros valores y toma de decisiones se ve influenciada por los comportamientos del grupo. Esto es el llamado efecto espectador, que se explica a través de dos aspectos sociales principales:

- Ignorancia pluralista: En un estado grupal donde la norma del grupo es públicamente aceptada, pero privadamente rechazada.
- Difusión de la responsabilidad: Una persona se siente menos responsable en grandes grupos.

Más que centrarnos en refactorizar, no se debe subestimar el aparato organizativo. Practicar la revisión de código y tener un mantenedor principal tienen efectos positivos en la calidad del código.

Técnicas de análisis de código de comportamiento

Según investigaciones nos dicen que demasiados cambios en un archivo indica la disminución de la calidad de ese archivo, dando como resultado errores.

Hotspots – Puntos críticos

Son los posibles objetivos de refactorización y que son puntos donde más errores pueden existir, siendo estos resultados de un código muy activo y complejo.

Tendencia de la complejidad

Una vez encontrados los puntos críticos podemos determinar la tendencia de complejidad de cada uno de estos posibles objetivos de refactorización. Podemos categorizar el punto crítico como activo (complejidad creciente), inactivo (complejidad constante) o extinto (complejidad decreciente)

Cambio de acoplamiento

En archivos demasiado grandes, antes de comprender todo lo que hacen sus funciones se puede aplicar otra técnica para comprender la evolución de estas, el cambio de acoplamiento. Se basa en la lógica compartida entre dos funciones o archivos, que implica que si uno cambia conlleva cambios en el otro. Esto es distinto al acoplamiento habitual de la ingeniería del software ya que es invisible, estando implícito en el código, sólo pudiendo notarse mirando los cambios realizados a lo largo del tiempo.

Un alto grado de acoplamiento no tiene por qué ser malo. Siguiendo el principio de proximidad y agrupando funciones que cambien juntas, se transmite información que no es posible de expresar sólo a través del código.

En conjunto, el hecho de que un código sea malo no significa un problema, ya que es mas importante el código en constante evolución.

Métricas

- **Complejidad:**
  Al ser el análisis de código de comportamiento independiente del lenguaje de programación, la complejidad se aproxima en función de las sangrías, ya que normalmente se acuerda una manera de codificar definida para el equipo. Cuanto más número de sangría:
  a) Más líneas, el proyecto es más grande.
  b) Más estructuras anidadas, por lo que aumenta la complejidad.

  Lo importante de esto es poner este número en relación con el tiempo, para ver las tendencias.

- **Salubridad:**
  Hay herramientas que nos muestran tendencias de salud y que nos ofrecen también una lista de hallazgos sobre la salud del código lo que nos permite, junto a los hotspots nos pueden dar una buena información.

Bibliografía

https://www.viaboxx.de/blog/behavioral-code-analysis/

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

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