Curso 24/25

Software engineering at Google

By Hyrum Wright

Google

Grupo de seminario ES_305

- Pablo Urones Clavera - UO264915
- Gael Horta Calzada - UO295662
- Andrés Carballo Pérez - UO287983

Tabla de contenido

Ley de Hyrum y las Dependencias Implícitas

La Ley de Hyrum dice *“Con suficientes usuarios de una API, cualquier comportamiento observable será usado por alguien.”* y refleja que incluso cambios menores en el código pueden romper sistemas que dependen de detalles no documentados. Esto complica el mantenimiento y exige herramientas y pruebas masivas para evitar regresiones inesperadas.

Ingeniería de Software vs. Programación

En Google, la ingeniería de software va más allá de escribir código: implica diseño, mantenimiento, colaboración y escalabilidad. Se prioriza resolver problemas, no solo producir líneas de código. Hyrum compara programar con la carpintería, diciendo que un programador sería un carpintero y una casa sería un software.

Desafíos de Escala y Eficiencia en Google

Google enfrenta retos únicos por el tamaño de su código base y sus sistemas. Necesitan herramientas que funcionen a gran escala, como análisis estático, testing masivo y migraciones automatizadas, siempre buscando costes sublineales al crecer.

Cultura de Equipo, Compartición de Conocimiento y el "Mito del Genio"

Google promueve una cultura donde el trabajo en equipo y la comunicación abierta superan al individualismo. El “mito del genio” es reemplazado por revisión de código, documentación y feedback constante. Los sistemas exitosos no los construye un genio solitario.

Medición de la Productividad en Ingeniería

La productividad no se mide solo con métricas simples. Google habla de metas, señales y métricas, y analiza si esas métricas realmente cambian comportamientos útiles. Además, evitan que los ingenieros se optimicen en exceso hacia métricas engañosas. No se trata de contar líneas de código, sino de lograr resultados útiles.

Guías de estilo y prácticas de testing

Las guías de estilo son importantes para la **consistencia y legibilidad**. Google publica sus guías (ej., la de C++). Se enfocan en la legibilidad (ej., limitando el uso de auto en C++ donde el tipo explícito mejora la comprensión) y se apoyan fuertemente en **herramientas automáticas** (linters, formateadores como clang-format) para verificar y aplicar reglas objetivas (como el formato del código), liberando a los ingenieros de preocuparse por detalles menores y permitiendo que las revisiones de código se centren en aspectos más sustanciales. Sin embargo, algunas reglas requieren juicio humano.

Wright destaca anti-patrones comunes como el exceso de código repetitivo (boilerplate) o no probar el comportamiento correcto del sistema. Promueve el "test más pequeño posible" **que verifique el comportamiento básico** y sirva como primera validación de la API. Los tests son vistos como documentación ejecutable y la primera forma de experimentar la API como lo haría un usuario.

Un **flaky test** es aquel que puede pasar o fallar bajo las mismas condiciones aparentes. Son extremadamente costosos en Google por dos razones:

- **Costo Computacional:** A menudo requieren múltiples ejecuciones para confirmar su estado, consumiendo recursos.
- **Costo Humano:** Cuando un flaky test falla en el cambio de otro equipo, ese equipo pierde tiempo investigando una falla no relacionada con su cambio, lo cual es un desperdicio significativo a escala.

Análisis estático

El análisis estático permite examinar el código fuente sin necesidad de ejecutarlo, lo que facilita la comprensión del comportamiento del programa y la detección temprana de errores. En Google, se emplean tanto análisis estático como dinámico, aunque el primero ha demostrado ser especialmente eficaz para identificar problemas en las primeras etapas del ciclo de desarrollo.

Una de las herramientas clave que utilizan es **Tricorder**, una plataforma de análisis estático escalable desarrollada por Google. Esta herramienta permite identificar errores y patrones propensos a fallos, además de ofrecer sugerencias de mejora. Tricorder facilita que equipos que trabajan con distintos lenguajes puedan crear sus propias herramientas de análisis, integrándolas automáticamente en los procesos de revisión de código. Incluso es capaz de corregir errores directamente durante el desarrollo.

Cambios de código a gran escala

Los cambios de gran escala afectan a miles de archivos y no pueden aplicarse de una sola vez.

Para facilitar su implementación y validación, se automatizan al máximo y se dividen en bloques pequeños.

Conclusión

La conversación con Hyrum Wright subraya que la ingeniería de software en Google es una disciplina multifacética que va mucho más allá de la simple escritura de código. Implica lidiar con una escala inmensa, cultivar una cultura de colaboración y transparencia, desarrollar procesos y herramientas sofisticadas para el mantenimiento, la medición y las pruebas, y tomar decisiones conscientes que equilibren la eficiencia de escritura con la crucial legibilidad y mantenibilidad a largo plazo. Los principios y prácticas discutidos, aunque desarrollados en el contexto específico de Google, ofrecen lecciones valiosas para cualquier organización que aspire a construir y mantener software de manera efectiva y sostenible.