

---

# SOFTWARE ENGINEERING AT GOOGLE

by Hyrum Wright

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

+

o

•

# Hyrum Wright

- Ingeniero de software en Google
- Responsable técnico C++
- Profesor en la Universidad de Carnegie Mellon
- Editor y contribuidor del libro "Software Engineering at Google"



# Ley de Hyrum

With a sufficient number of users of an API,  
it does not matter what you promise in the contract:  
all observable behaviors of your system  
will be depended on by somebody.

- Hyrum's Law

# Managers. Programar $\neq$ Ingeniería de Software

- Managers tradicionales  Cómo hacer las cosas
- Buenos managers  Por qué se hace. Confianza en el equipo
  
- Programar  Escribir código
- Ing. Software  Construir sistemas sostenibles y eficientes

# Público objetivo. Desafíos en Google

- Profesionales, estudiantes, universidades.
- No se trata de inventar nuevas cosas, sino de hacerlas escalables.
- Desafíos técnicos como:
  - Probar millones de líneas de código eficientemente
  - Organizar un código base masivo
  - Coste de mantenimiento del código



# Mito del genio. Trabajo en equipo

- "Los grandes sistemas son creados por una sola persona brillante."
- Colaboración mediante revisiones de código, diseño compartido, comunicación...
- Escrito una vez, leído muchas veces
- Claridad de código
- Futuros trabajadores



## WHAT PART OF

```
attachEvent("onreadystatechange",H),e.attachE
boolean Number String Function Array Date RegE
|={};function F(e){var t=_[e]={};return b.ea
t[1])===!1&&e.stopOnFalse){r=!1;break}n=!1,u&
?o=u.length:r&&(s=t,c(r));return this},remove
ction(){return u=[],this},disable:function():
re:function(){return p.fireWith(this,argument
ending",r=(state:function(){return n},always:
romise)?e.promise().done(n.resolve).fail(n.re
id(function){n=s},t[1^e][2].disable,t[2][2].
=0,n=h.call(arguments),r=n.length,i=1!==r|e&
(r),l=Array(r);r>t;t++)n[t]&&b.isFunction(n[t
/><table></table><a href='/'>a</a><input typ
/TagName("input")[0],r.style.cssText="top:1px
test(r.getAttribute("style")),hrefNormalized:
```

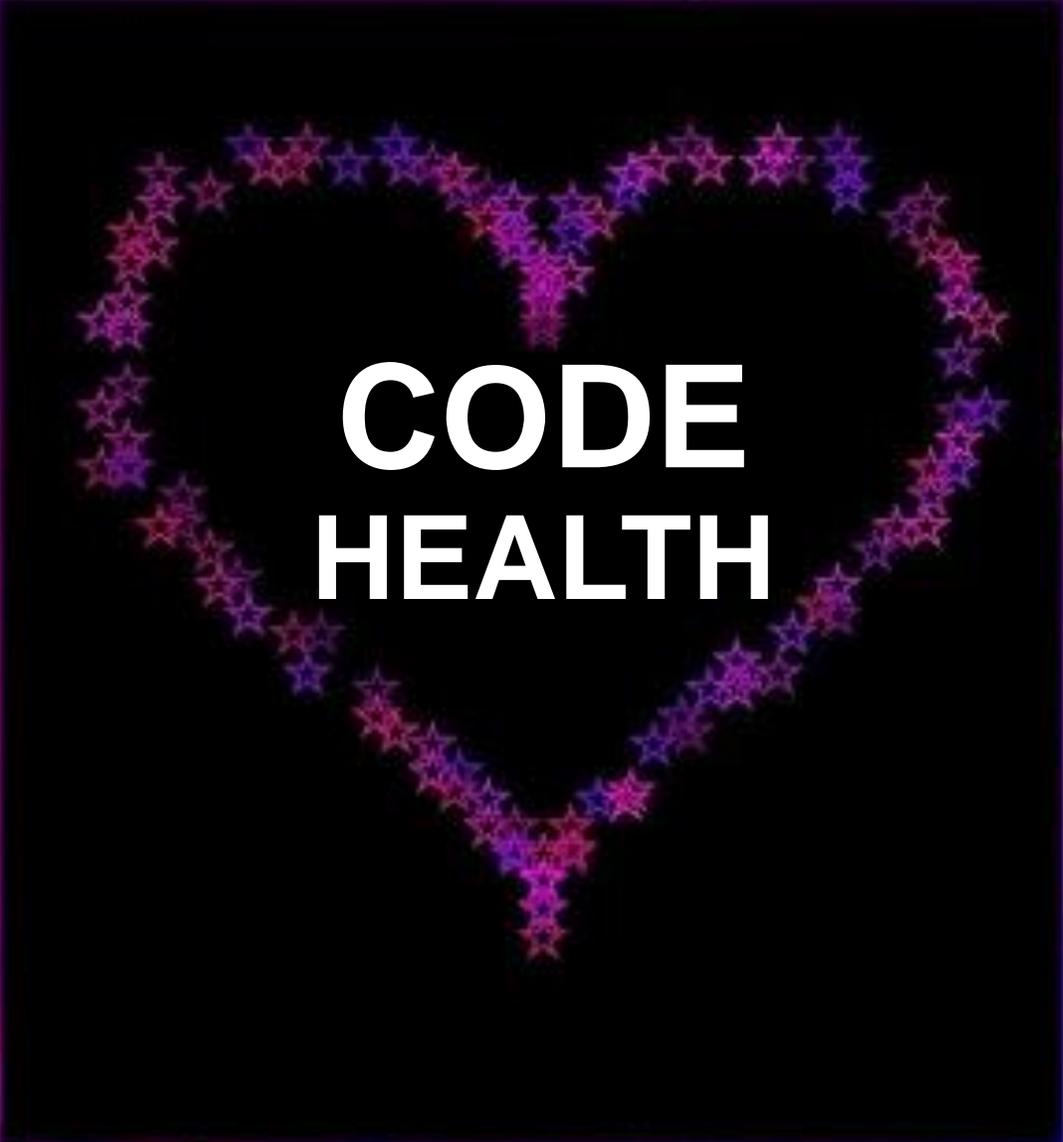
DON'T YOU UNDERSTAND

# Ocultar información. Productividad

- Tendencia a trabajar en "secreto"
  - Produce retrasos o llevan a problemas más complejos
  - Solución en Google:
    - Compartir desde el inicio
    - Fomentar la colaboración temprana
- Métricas malas o engañosas
  - Estrategia de Google:
    - Definir que se quiere lograr
    - Identificar indicadores de progreso
    - Datos concretos que reflejen el impac



```
mainpy - Sublime Text
mainpy
1
2 from tkinter import Canvas, Tk, Frame
3 from math import cos, sin
4 from random import randint, choice
5
6 class Heart(Frame):
7     def __init__(self, master):
8         super().__init__(master)
9
10        self.canvas = Canvas(master, bg = 'black')
11        self.canvas.place(relx = 0, rely = 0, relwidth = 1, relheight=1)
12
13
14        self.objects = []
15        self.num = 0
16        self.chars = ['>', 'v', '*', '@', 'x', 's', 'e', 'o', '0', '1']
17        self.char = '*'
18
19        self.create_obj()
20        self.update()
21
22    def create_obj(self):
23        for i in range(200):
24            obj = self.canvas.create_text(0,0, font= ('Arial',24))
25            self.canvas.coords(obj, 500, 250)
26            self.objects.append(obj)
27
28    def draw(self, obj, x,y, color, char):
29        self.canvas.itemconfig(obj, fill = color, text=char)
30        self.canvas.move(obj, x,y)
31
32    def update(self):
33        for t in range(0,200,1):
34            wa = .3*(int(16*now*(left) .3))
```



# STYLE GUIDES



Formateadores  
automáticos

Legibilidad

Ojo con el uso de auto



# Test techniques

- Flaky tests
- Code coverage
- Mutation testing
- Test Doubles

# FLAKY TESTS

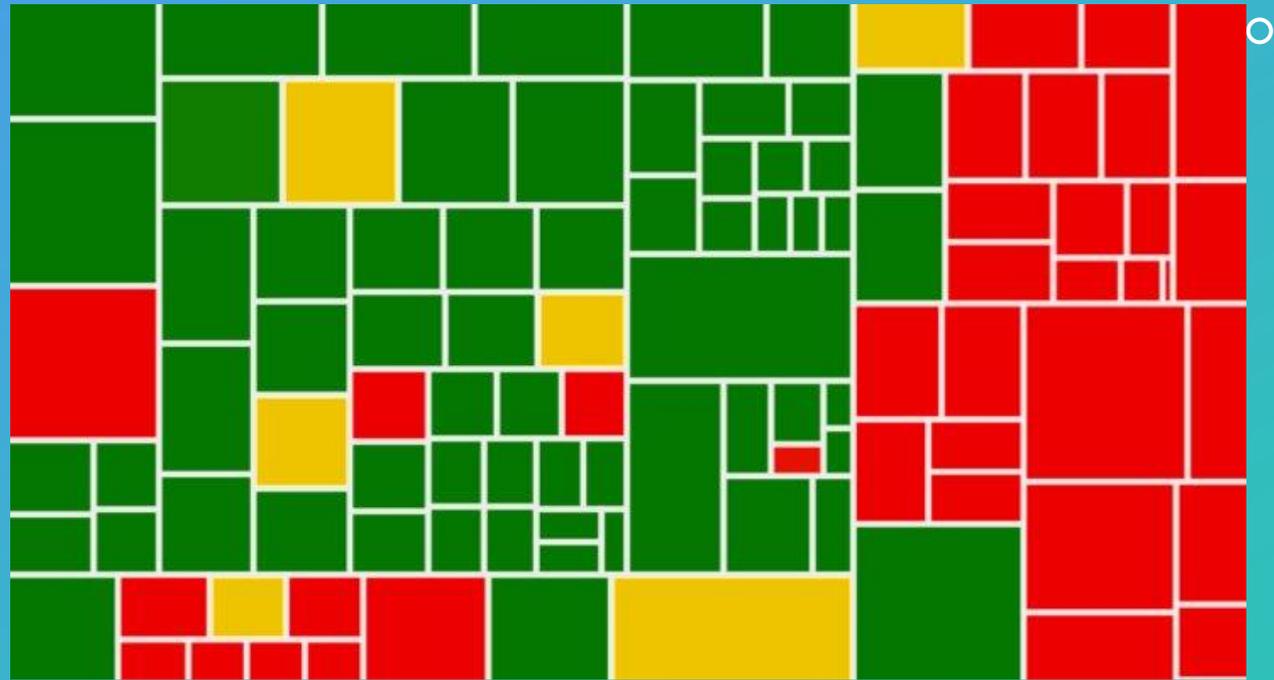


Bastante caros

Coste computacional y humano

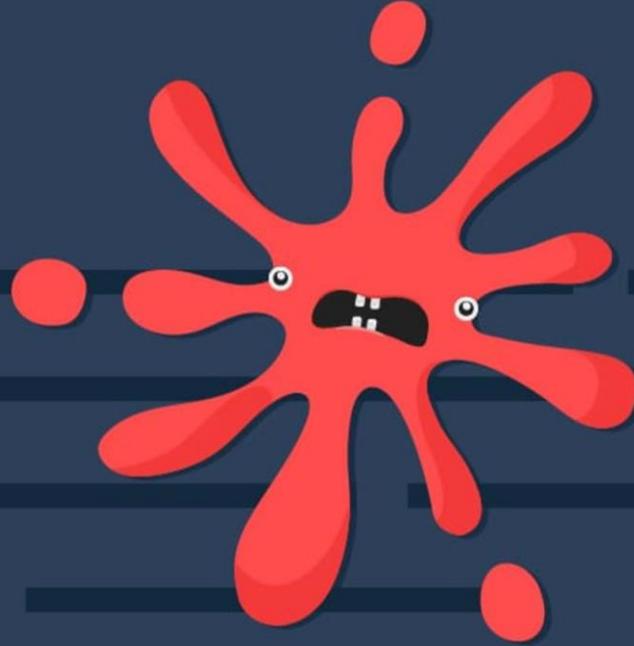
Flaky tests en  
Google

# CODE COVERAGE



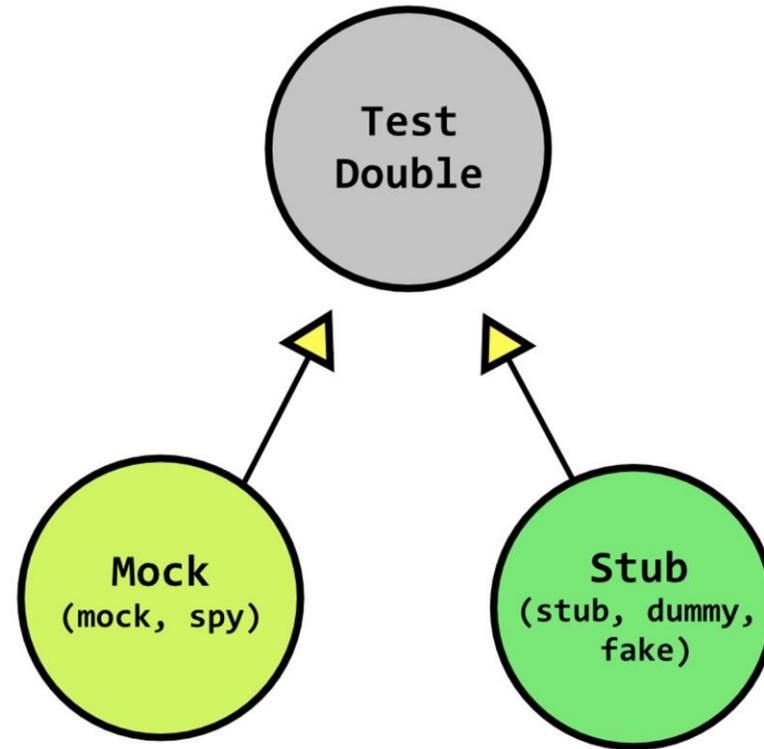
</> ...

# MUTATION TESTING



---

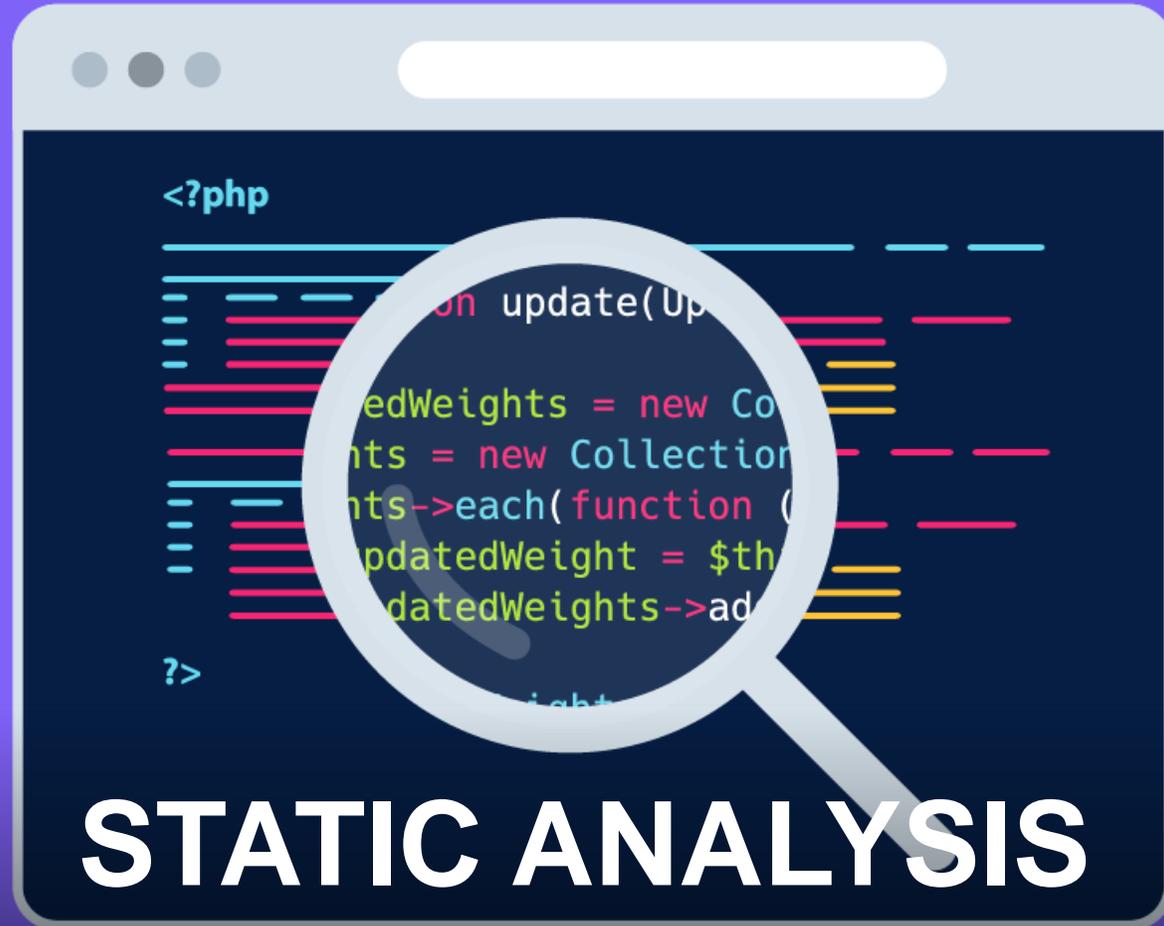
# Test doubles



¿Cómo lo hacen en

Google ?

- ❖ Grado de testeo
- ❖ Técnicas
- ❖ Mantenerse en el scope



**STATIC ANALYSIS**

# Tricorder



Plataforma para realizar análisis estático escalable.



Recopila datos y los muestra en el momento oportuno.



Pieza central de la gestión del proceso de análisis.

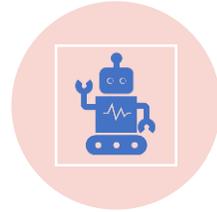


The Google logo, featuring the word "Google" in its characteristic multi-colored font: blue 'G', red 'o', yellow 'o', blue 'g', green 'l', and red 'e'.

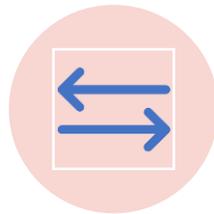
# Cambios a gran escala (LSC)



MODIFICACIONES  
QUE AFECTAN MILES  
DE ARCHIVOS.



AUTOMATIZAR EL  
PROCESO SI ES  
POSIBLE.



EJ: MIGRACIÓN DE  
APIS, LLAMADAS A  
FUNCIONES...



# Validación de LSC



PRUEBAS  
AUTOMATIZADAS.



DIVIDIR EN PARTES  
MÁS PEQUEÑAS.



REVISIÓN DE CÓDIGO.



GENERAR CONFIANZA  
A LOS TRABAJADORES.



RETROALIMENTACIÓN.

# Restricciones

No suelen gustar.

Facilitan la escalabilidad.

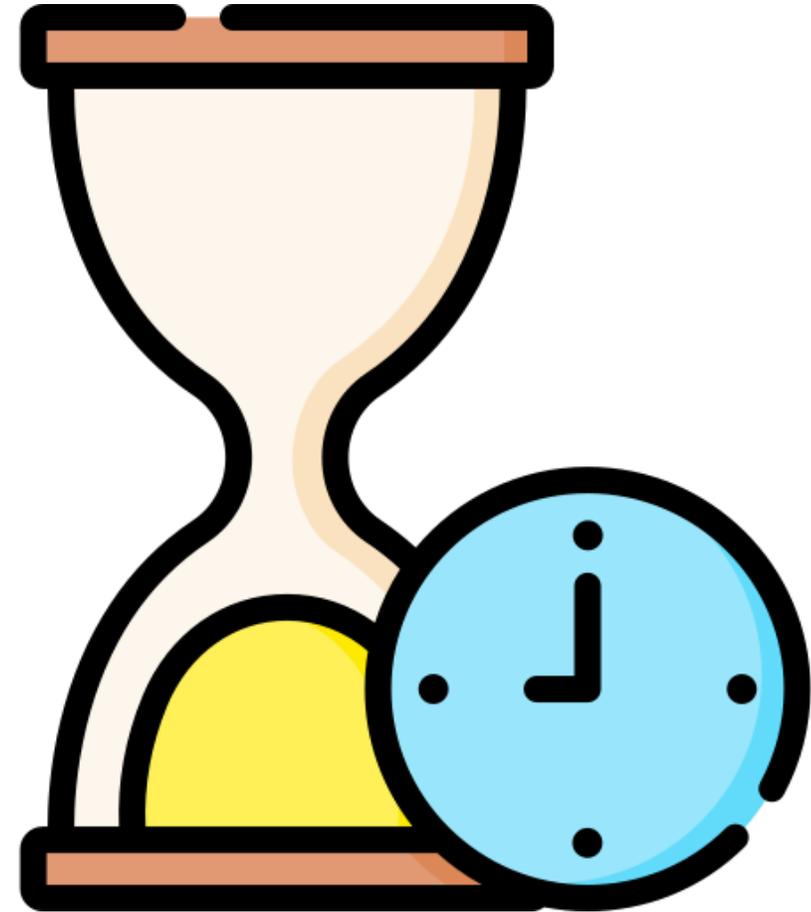
Pueden ser positivas.



# Tiempo

- Cuanto debe durar un software.
- Influye en muchas decisiones.

🔊 *"Es un tema influyente, pero rara vez discutido en diseño de software." — Hyrum Wright*



# Consejos de Hyrum Wright

## Para juniors

- Pensar más allá del software
- Enfocarse en el panorama general

## Para mejorar conocimientos

- Mejorar el ecosistema y la cultura.
- Facilitar el trabajo a los demás.

