Software engineering at Google

Group 5
Ana Castro, Manuel Méndez, María Rodríguez y Héctor Triguero

Table of contents

01

Problems

02

Style Guides

03

Communication

04

Measuring productivity

05

Testing

06

Static analysis

07

Scalability

08

Time

01

Problems

Problems and challenges at Google

Challenges not about kind, but **scale**.

Products based on **pre-existing** concepts.

Main focus: Scale products effectively.

02

Style guides

Code Readability and Style Guides

Objective: Code readability

How to achieve it?

Use **style guides** and tools that mantain consistency.

Results:

Slow development but high mantainability and collaboration.

03

Communication

Communication and Collaboration

Openess in coding.

Provide early feedback and avoid hiding.

Avoid the "genius myth"

Team is stronger than an individual.

04

Productivity

Measuring Productivity

Ask leaders:

What to measure?

What to improve?

Perform the measurements **discreetly**:

So develpers don't alter their behaviour during measures.

Code Health Team: Long-term code quality over short-term output

05

Testing

Testing at Google

Boilerplate tests: Borderline conditions.

Flaky tests: They behave differently under the same conditions

Test double: They substitute a complex system and emulate the responses

Test coverage: Number of lines of code we execute in our tests

06

Static analysis

Static analysis

Examine source code without compiling or executing it

Engineers can detect **potential issues** before they manifest

Static analysis cannot catch every possible problem

Helps **identify many errors** efficiently, reducing the cost of debugging

Tricorder: Google's Static Analysis Tool

Tricorder acts as a framework that integrates **language-specific** analysis tools

Aggregates and presents **analysis results** at relevant stages

Developers receive **timely feedback** on issues.

07

Scalability

Large-Scale Change

Modifications in the codebase **too extensive** to be atomically.

Instead of requiring each team to make these changes, a **centralized team** is responsible

Testing large-scale changes in their entirety is **impractical**.

Changes are designed to be **modular**, so they become smaller and independently testable parts.

Validated through **continuous integration** and reviewed by engineers.

Constraints in Scalability

For a system to remain scalable, organizations must **impose constraints**.

Enforcing **strategic constraints**, enhances collaboration and scalability.

Well-designed constraints **empower engineers**, enabling large-scale engineering.

08

Time

Time as a Factor in Engineering Decisions

The expected **lifespan of software** influences decisions.

Long-term projects require **careful planning**

Short-term solutions may **prioritize speed**

Understanding the intended longevity helps make **informed choices**