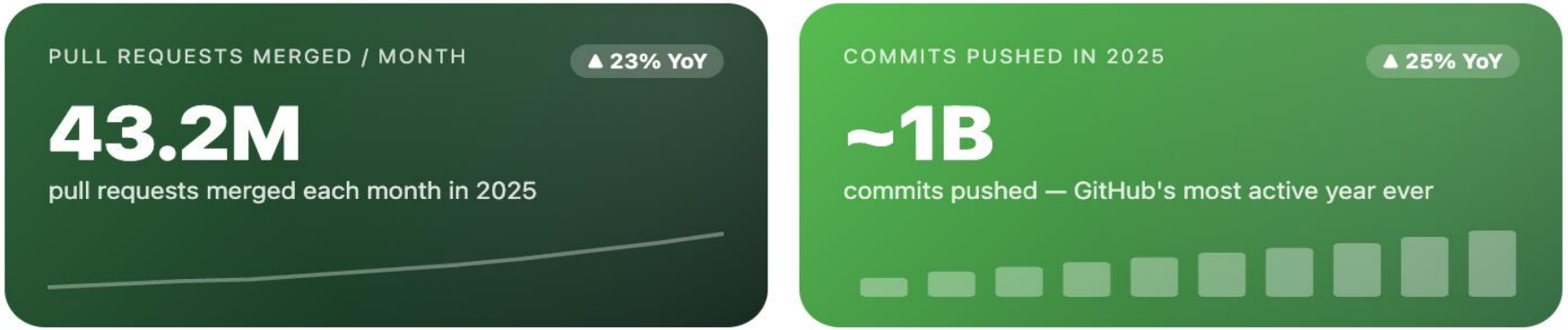
Beyond Code Coverage: Functionality Testing with Playwright

MARLENE MHANGAMI
SENIOR Developer Advocate
Microsoft/GitHub

github.com

GITHUB OCTOVERSE 2025

More Code Is Being Created



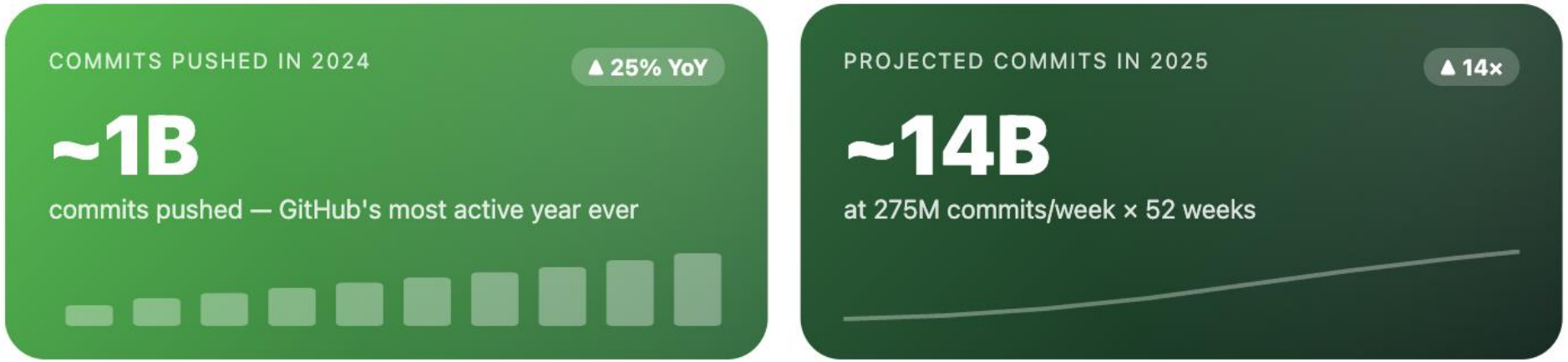
Source: GitHub Octoverse 2025

github.com

2026 GROWTH RATES

Growth Is Accelerating

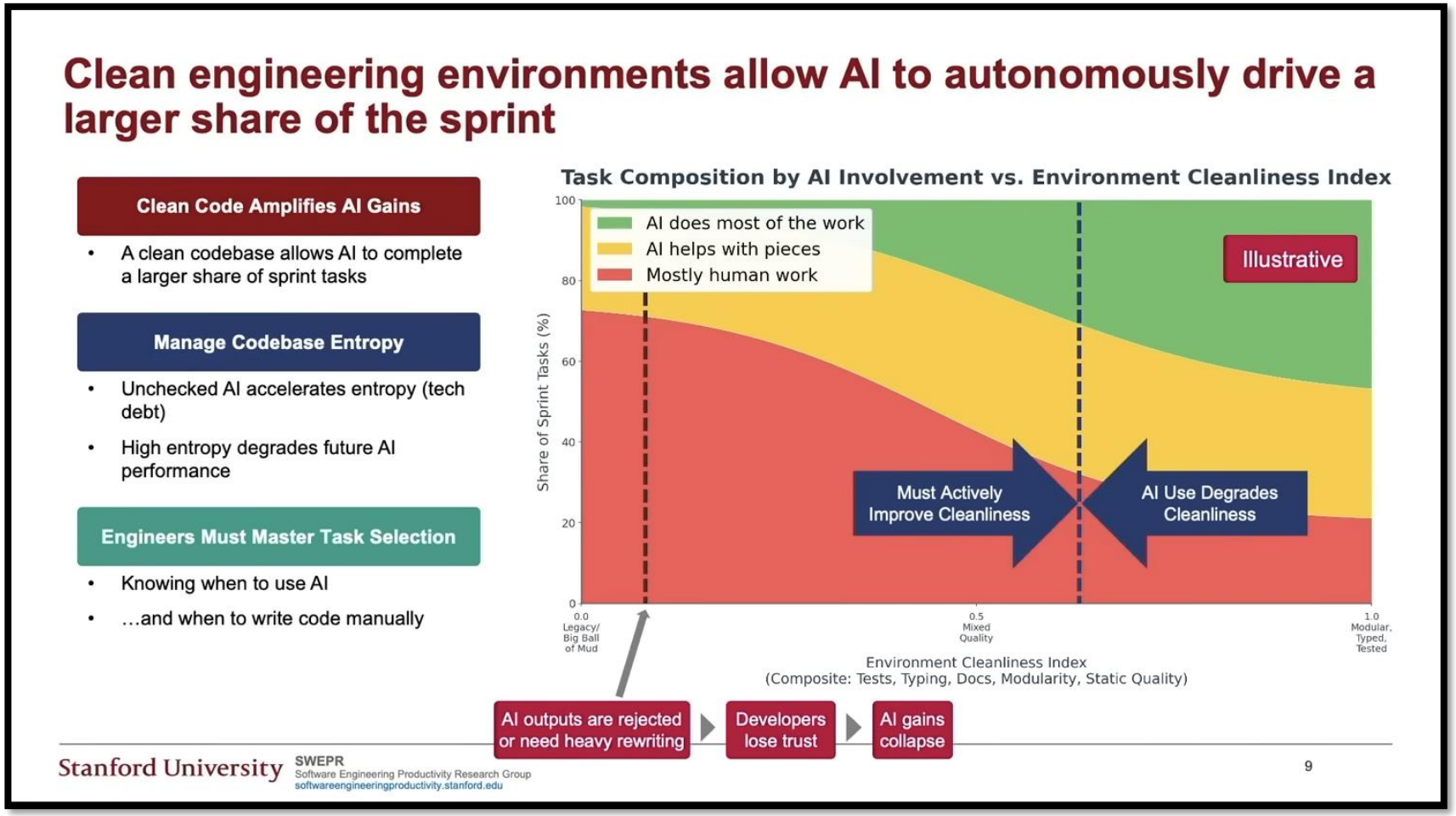
A growing share of these commits are co-authored by AI agents



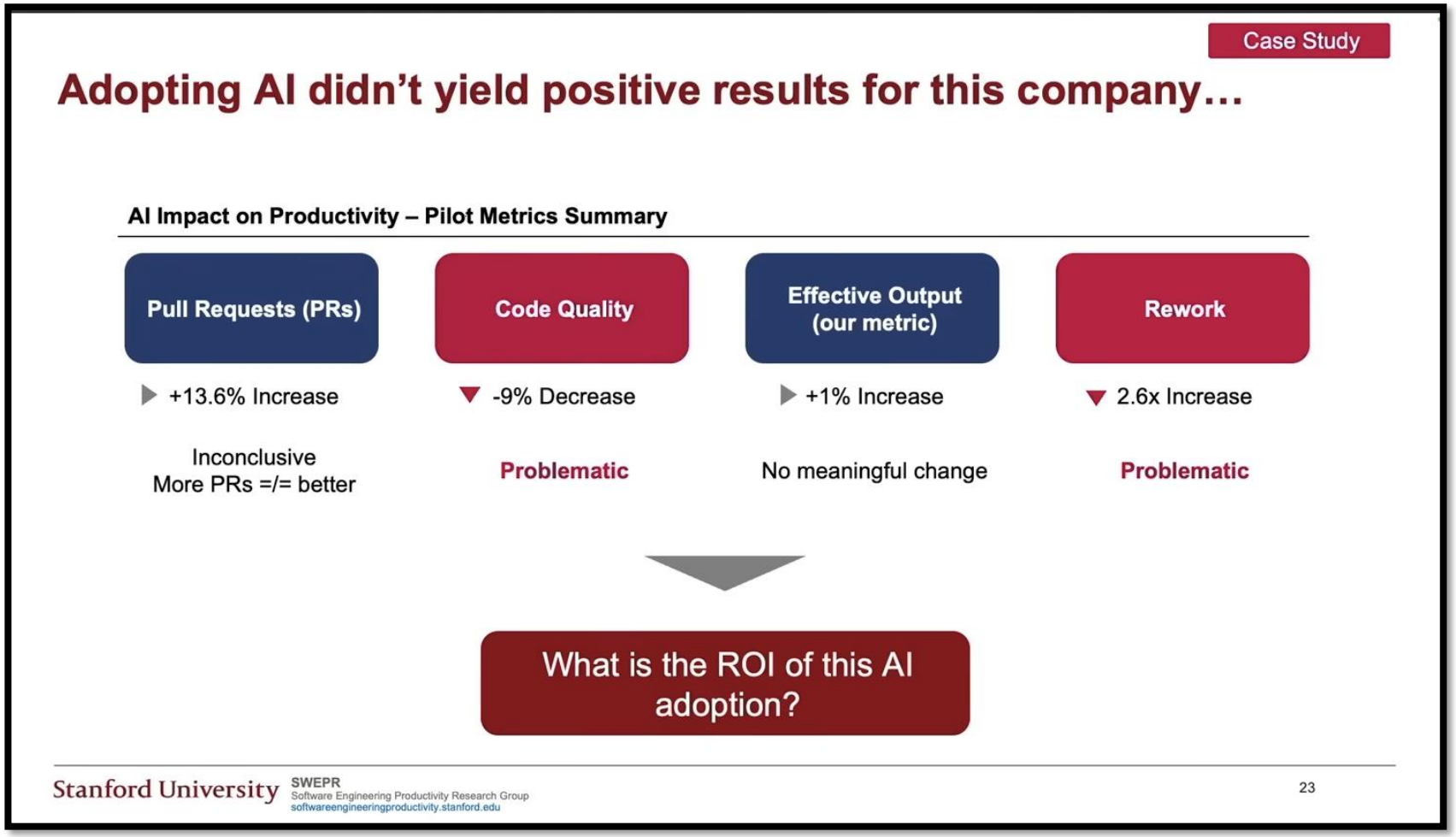
Kyle Daigle - GitHub COO

Does AI Make Developers More Productive?

Clean code amplifies AI gains



Unchecked AI amplifies entropy



Priorities for AI-assisted teams

Clean engineering environments tend to unlock more AI productivity gains

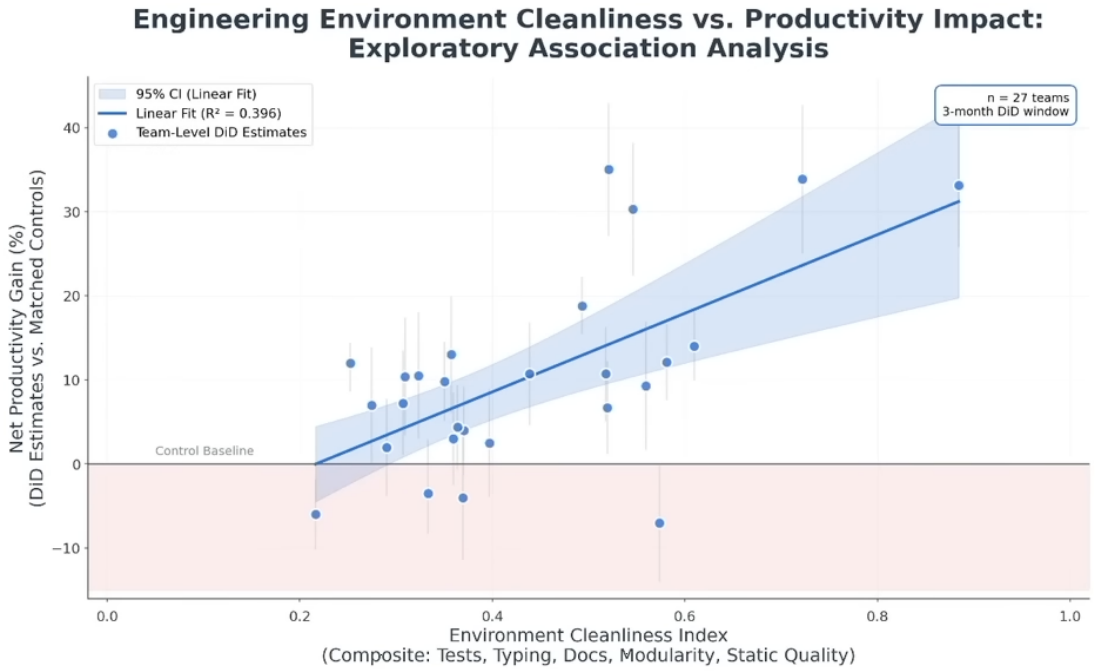
Environment Cleanliness Index

Composite score:

- Test coverage
- Type coverage
- Documentation quality & coverage
- Modularity
- Static code quality

Invest in codebase hygiene to unlock AI productivity gains

- Large, messy systems is where AI is most tempting & most dangerous
- Prioritize AI-heavy workflows in teams that already have strong engineering foundations

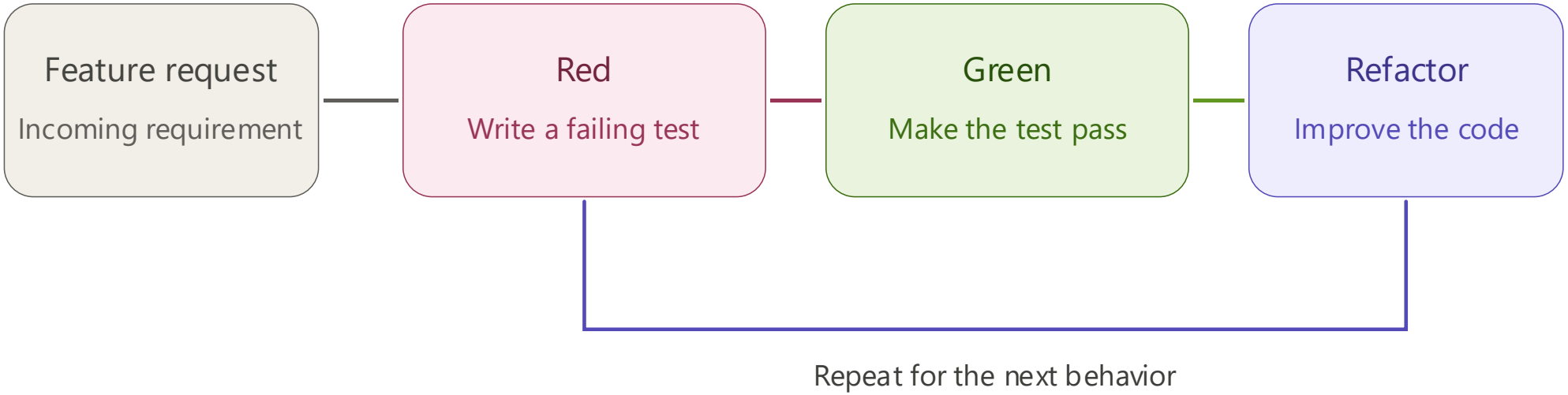


Methods: Same 27 teams and DiD effects as in previous slide. Environment Cleanliness Index is a 0 - 1 composite of test coverage, static code quality, type coverage, documentation coverage, and modularity. Plot shows OLS fit of DiD % gains on cleanliness with HC3-robust CIs; linear $R^2 \approx 0.3$ - 0.4. Exploratory, observational; cleanliness is treated as a correlate, not a causal estimate.

Stanford University SWEPR Software Engineering Productivity Research Group softwareengineeringproductivity.stanford.edu

8

How Can Developers Create And Maintain Clean Code?

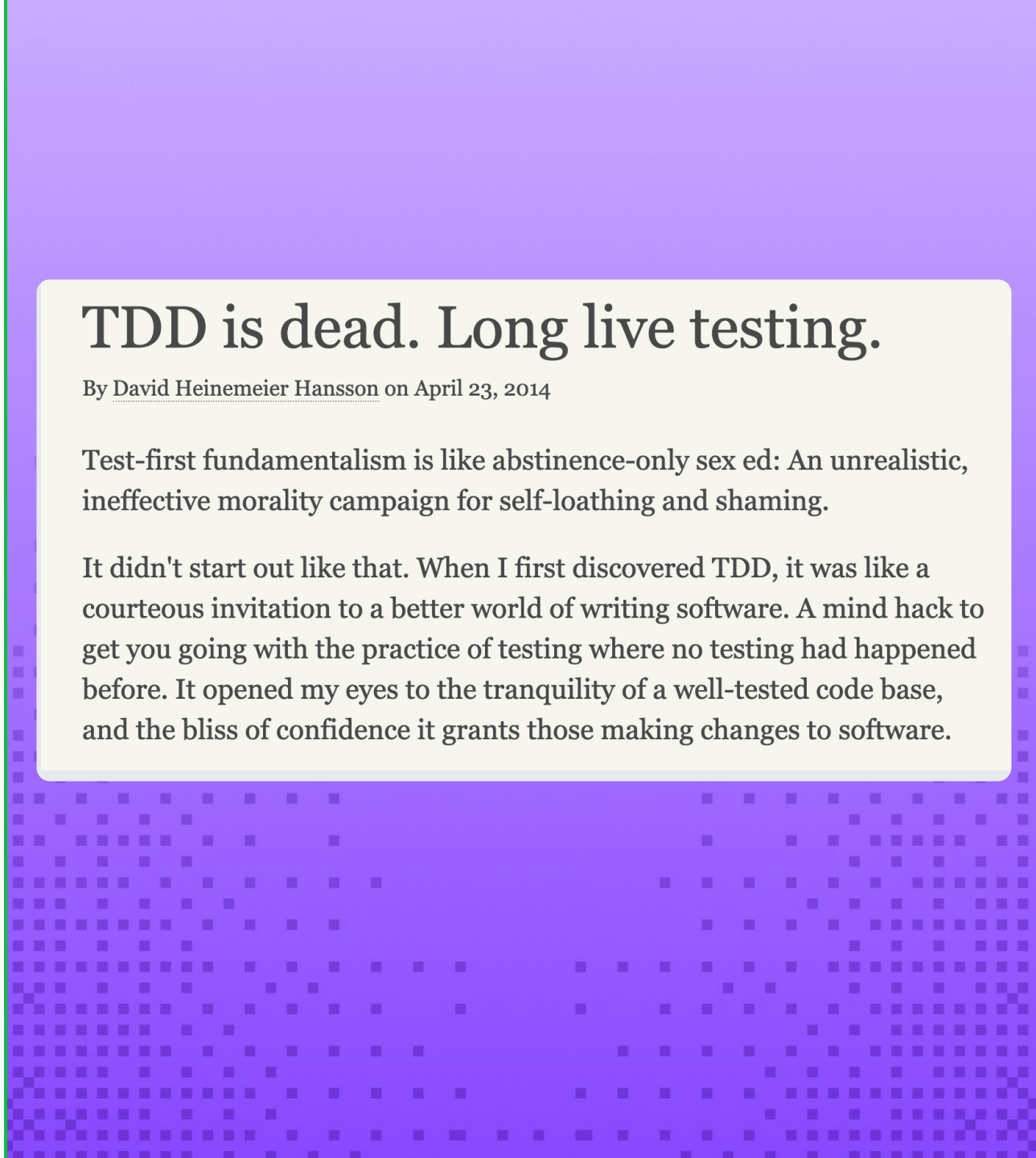


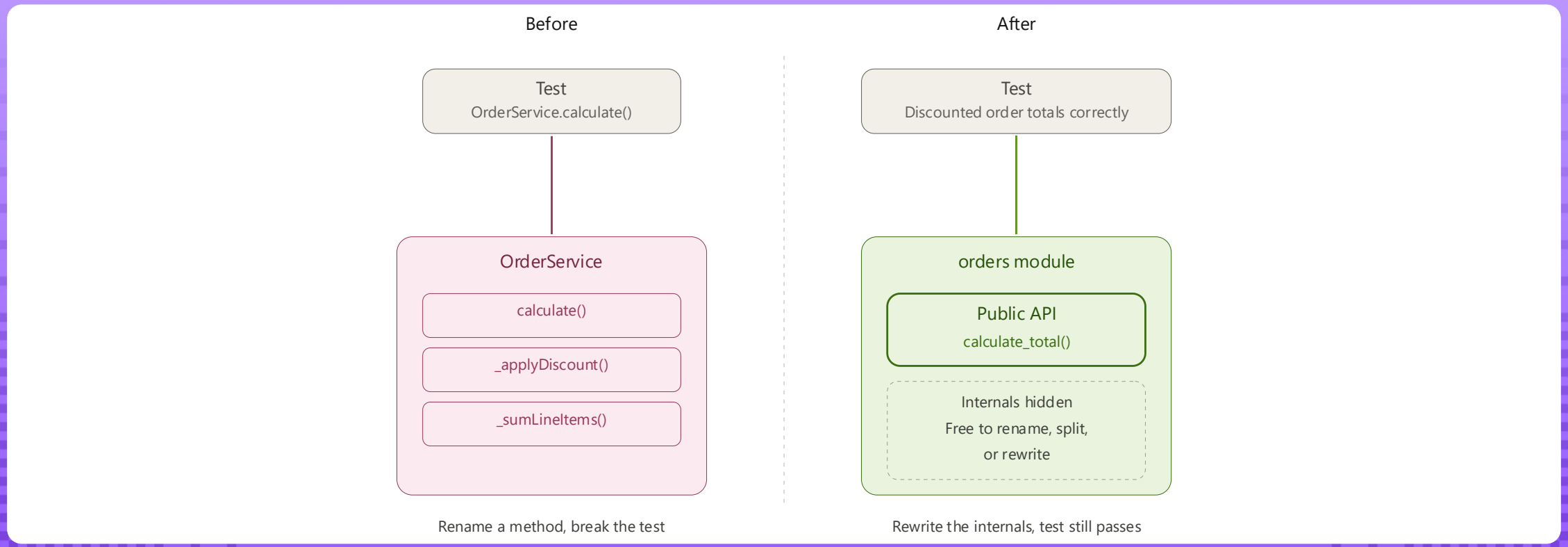
Red/Green TDD Loop

Test Driven Development (TDD)

“The current fanatical TDD experience leads to a primary focus on unit tests… rebalance the testing spectrum from unit to system”

Over-indexing on code coverage





‘We are not testing a method on a class we are testing a behavior of the system’

TDD, Where Did It All Go Wrong (Ian Cooper)

You’re absolutely right!

Self-affirming test

```python
def add_tax(price):
    return price * 1.05 # bug: should be 1.20

def test_add_tax():
    assert add_tax(100) == add_tax(100)
```

Always passes. Tells you nothing about whether the tax is correct.

Behavioral test

```python
def test_add_tax_applies_uk_vat():
    # UK VAT is 20%
    assert add_tax(100) == 120
```

Encodes the requirement. Catches the bug instead of affirming it.

Playwright For Functionality Testing

Playwright is an open-source testing framework by Microsoft that automates end to end testing in the browser by simulating user interactions

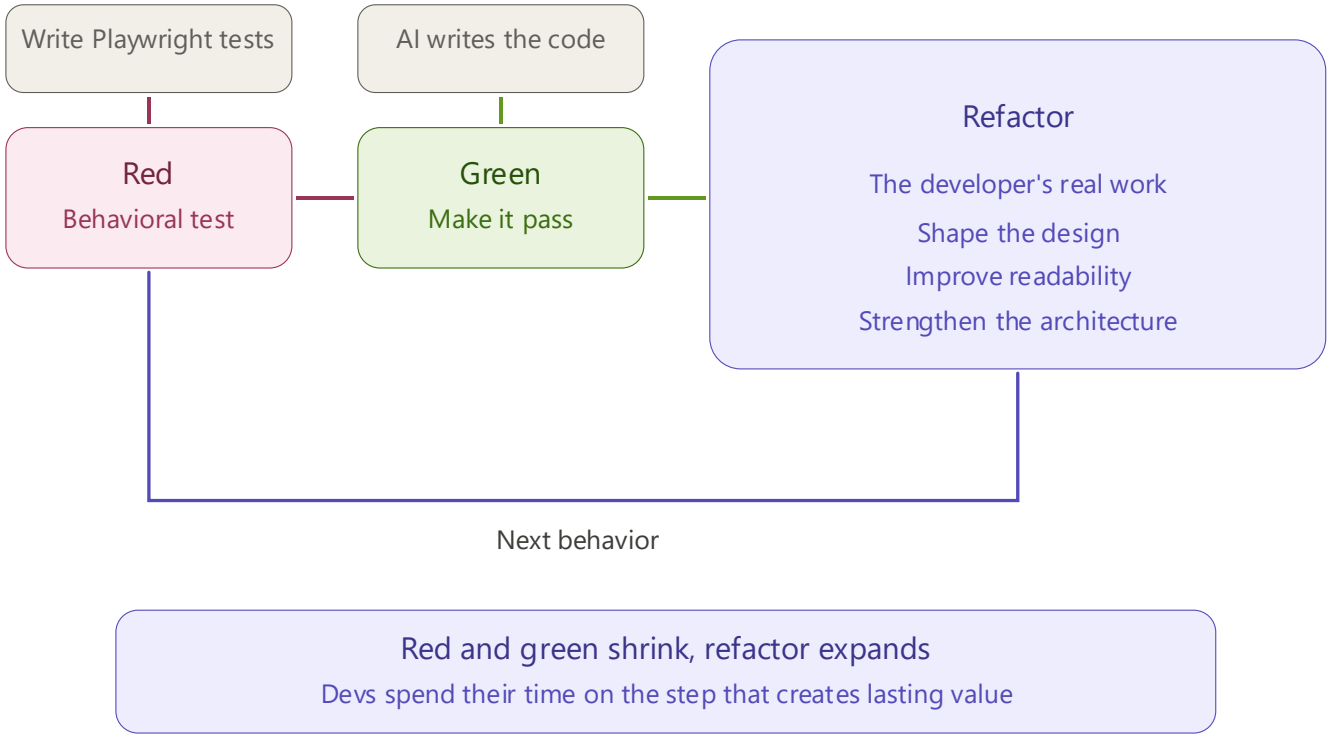
https://playwright.dev/

AI GENERATES TESTS

Playwright test

```python
from playwright.sync_api import Page, expect

def test_search_filters_toys(page: Page):
    page.goto("/toys")

    # Search by toy name
    search_input = page.get_by_placeholder("search")
    expect(search_input).to_be_visible()

    search_input.fill("Furby")

    expect(page.get_by_role("heading",
    name="Furby")).to_be_visible()
```

Python and Typescript support



Playwright with AI for TDD

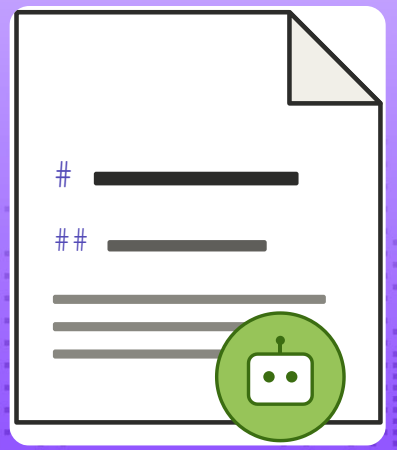
Using Playwright with coding agents

MCP Server

npx @playwright/mcp@latest

CLI

npm install -g @playwright/cli@latest



Agents

npx playwright init-agents --loop=vscode

Planner, Generator and Healer (agent.md files)

DEMO

Helpful Commands

Installation:

- Python: pytest tests/search_spec.py
- Typescript: npx playwright test

Run tests:

- Python: pytest tests/search_spec.py
- Typescript: npx playwright test

https://playwright.dev/

AI GENERATES CODE

Green: Generate code quickly to make tests pass

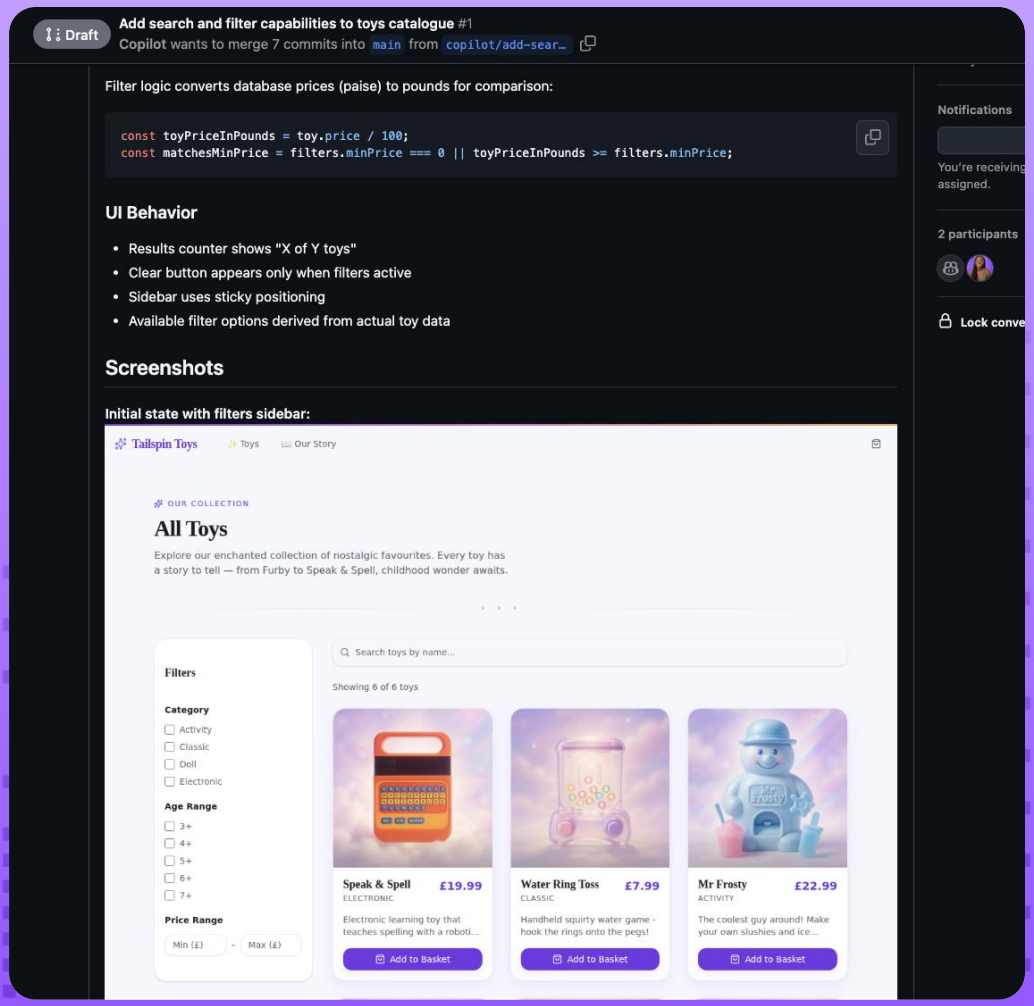
Prompt: Generate the code required for the tests to pass, try and match the implementation style of the codebase. Do not edit the tests.

```
'use client';

import { useState } from 'react';
import { Toy } from '@prisma/client';
import { ProductCard } from '@/components/toys/product-card';
import { SearchBar } from '@/components/toys/search-bar';

interface ToysCatalogueProps {
  toys: Toy[];
}

/** Displays the toys collection in a grid layout */
/** Displays the toys collection with client-side search filt
export function ToysCatalogue({ toys }: ToysCatalogueProps) {
  const [search, setSearch] = useState('');

  const filtered = toys.filter((toy) => {
    const q = search.toLowerCase();
    return (
      toy.name.toLowerCase().includes(q) ||
      toy.category.toLowerCase().includes(q) ||
      toy.shortTagline.toLowerCase().includes(q)
    );
  });
```

Refactor: Review and fix any issues with the implementation

```
import { Toy } from '@prisma/client';
import { ProductCard } from '@/components/toys/product-card';

interface ToysCatalogueProps {
  toys: Toy[];
}

/** Displays the toys collection in a grid layout */
export function ToysCatalogue({ toys }: ToysCatalogueProps) {
  return (
```

```
'use client';

import { useState } from 'react';
import { Toy } from '@prisma/client';
import { ProductCard } from '@/components/toys/product-card';
import { SearchBar } from '@/components/toys/search-bar';

interface ToysCatalogueProps {
  toys: Toy[];
}

/** Displays the toys collection with client-side search filtering */
export function ToysCatalogue({ toys }: ToysCatalogueProps) {
  const [search, setSearch] = useState('');

  const filtered = toys.filter((toy) => {
    const q = search.toLowerCase();
    return (
      toy.name.toLowerCase().includes(q) ||
      toy.category.toLowerCase().includes(q) ||
      toy.shortTagline.toLowerCase().includes(q)
    );
  });

  return (
```



Best Practices

- Add screenshots to PRs
- Use headless mode for multi-tasking
- Commit before running Healer
- Generate tests one feature at a time

https://playwright.dev/

Resources

Slides: aka.ms/playwright-ai-engineer

Documenation: playwright.dev

Github repo:

Connect:

LinkedIn: marlenemhangami

X: @marlene_zw

Bluesky: marlene@bluesky.social

Website: marlene.ai