This Year in Uber’s AI-Driven Developer Productivity Revolution

Uber

Adam Huda

Ty Smith

Uber’s Scale

$160 billon
Annualized run-rate gross bookings

156 million
Monthly active platform consumers

10,000 +
Cities

30 million
Trips per day

7.4 million
Monthly active drivers and couriers globally

70
Countries

Public figures for the quarter ended June 20, 2024

Platform Leverage

Feature Teams

Backend, Frontend

Rider App Platform

Earner App Platform

Eats App Platform

Developer Platform

Part of Platform Engineering

4,500
Feature engineers

200
Developer Platform engineers

22:1
Support ratio

3+
Engineering sites

Developer Experience

- IDEs & Tools
- Training & Documentation
- App & Service Frameworks
- Internal Libraries & Guardrails
- Monorepo Tooling & Build System

6 Monorepos for Swift, Kotlin, Typescript, Go, Java, and Python

5k+ Microservices

+8 Net promoter score

3 Major mobile apps

6 Minor mobile apps

Technical Debt

- Backlog of updates
- Fragmentation
- Test coverage

100million+

Lines of code across all of the monorepos

Macro Trends

- Flat headcount
- Backfills not guaranteed

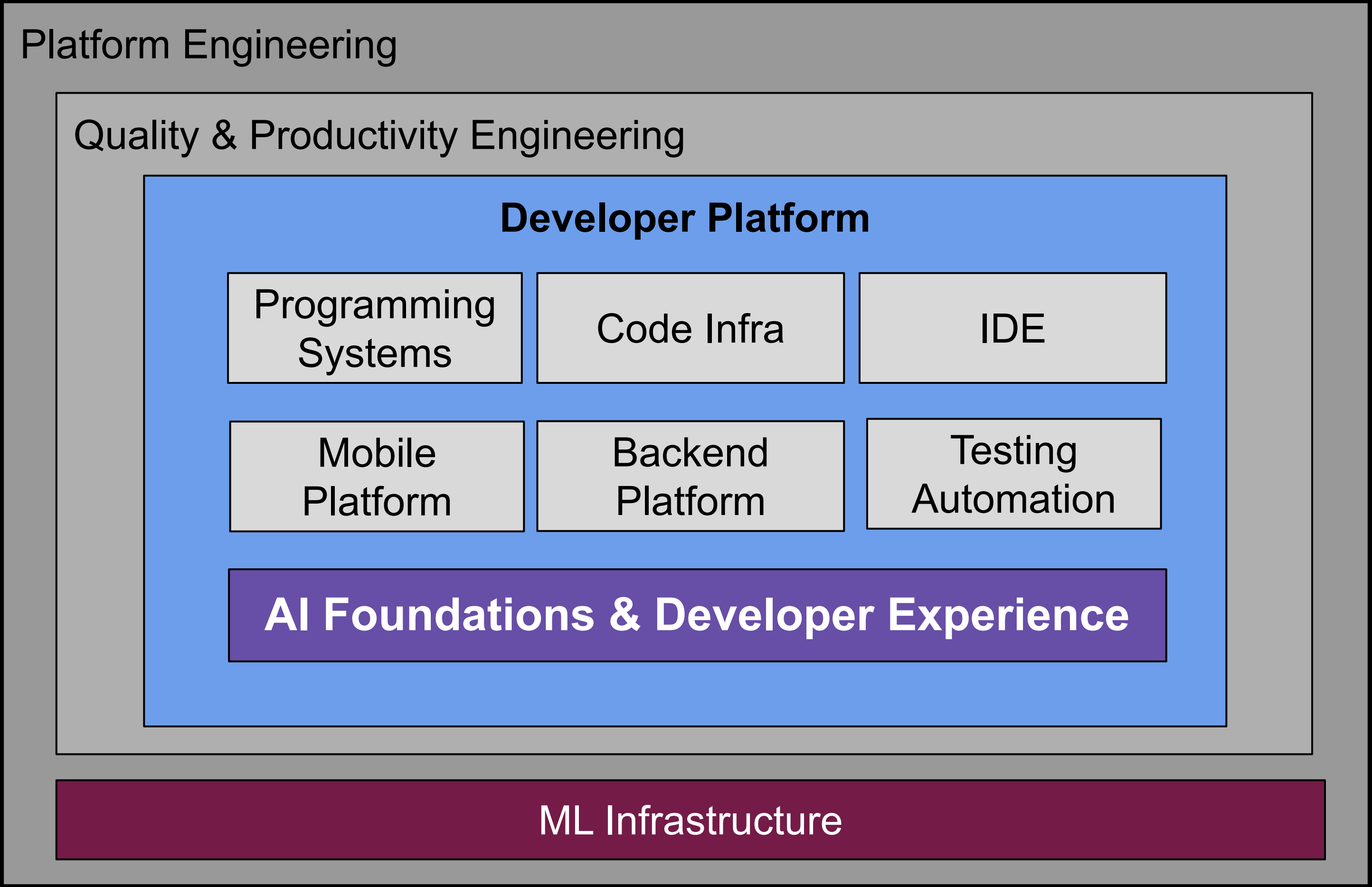
Emergence of AI as Leverage

Can’t scale people with the growth and maintenance needs of the codebase

Can we position Developer Platform to be AI-driven?

Org High-level View

Created a **centralized AI DevEx team** that specializes in AI applied to the SDLC



Timeline Applied-AI Developer Tools

Inaugural Hackdayz
Oct 2022

The ChatGPT moment
Nov 2022

Exploring generative AI
Hackdayz Summer 2023

Building automation and becoming AI-driven
Hackdayz Winter 2024

Focus on agentic systems
Hackdayz Summer 2024

Agentic Systems

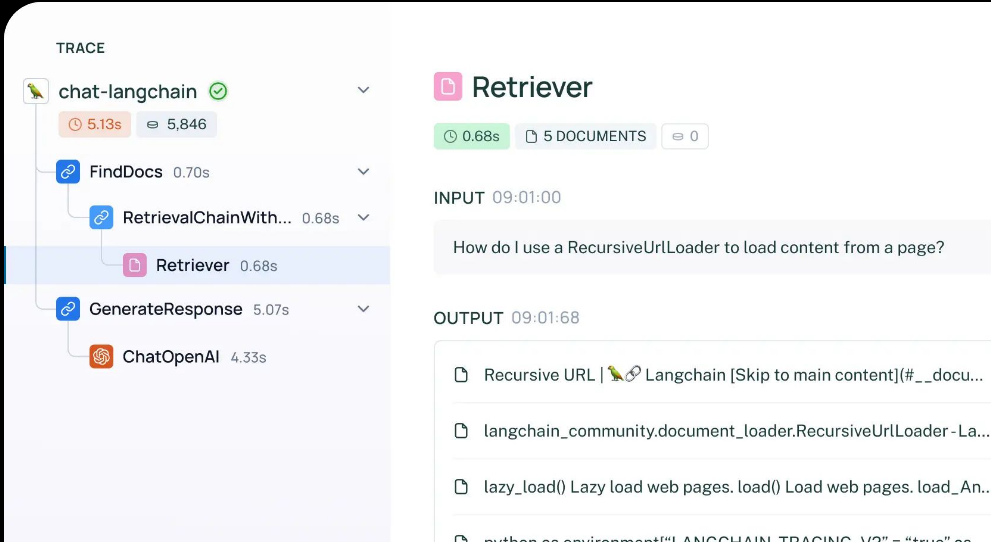
Multi-step systems to interact with LLMs

Breaks down a problem space into manageable tasks

You'll see some examples in our upcoming stories

LangChain

LangGraph



Applied-AI SDLC Stories

1. Coding Assistants
2. Generating Tests
3. Java to Kotlin Migration

DESIGN → DEVELOP → RELEASE → OPERATE → MAINTAIN

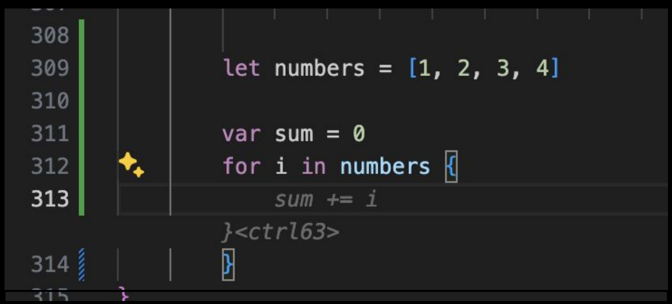
Coding Assistants

Code Assistants

Basics

- Native Plugin with Language Intelligence
- Model Backend

Contributes to **UX** and **Result Quality**



Broken Example Caused By IDE validation

IDE Plugin

Language Intelligence
Semantics, syntax, references & symbol navigation

Verification
Deterministic fixes for generated suggestions.

Extensibility
Adapt to development environments, workflows, & custom tools

Model backend

Foundational model and context awareness

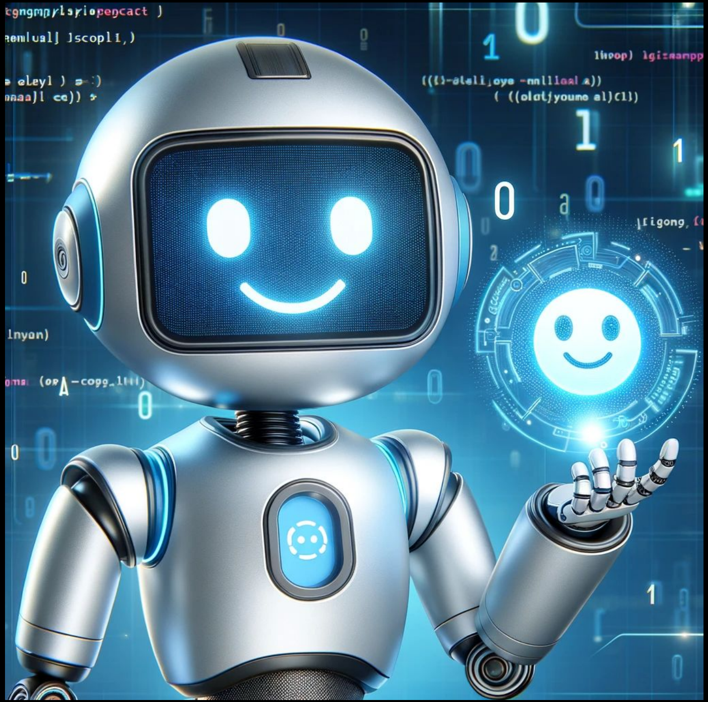
Hypothesis

An Uber trained IDE assistant is needed

Custom Code Assistants

Requirements

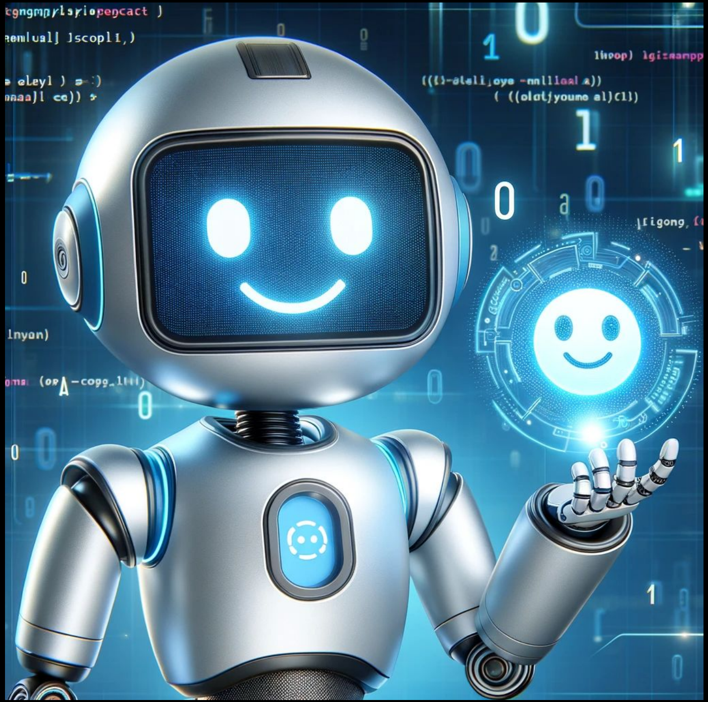
- Uber aware
- Fast
- Cost effective
- Per user analytics
- Workflow integrated



Custom Code Assistants

Buildout

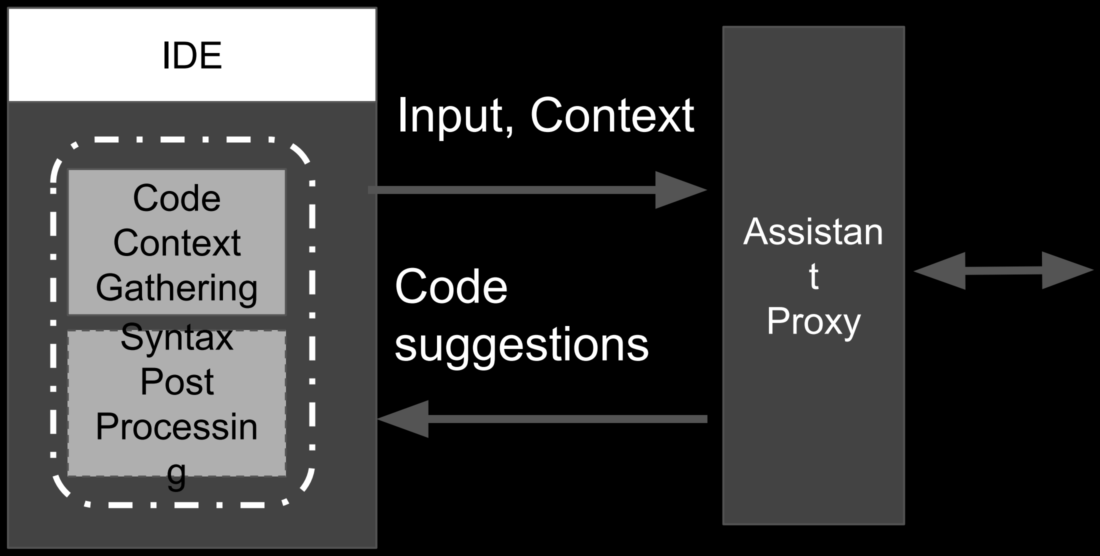
- MVP in Hackathon
- Evaluate LLMs
- Internal evangelism
- Wide variety of investments

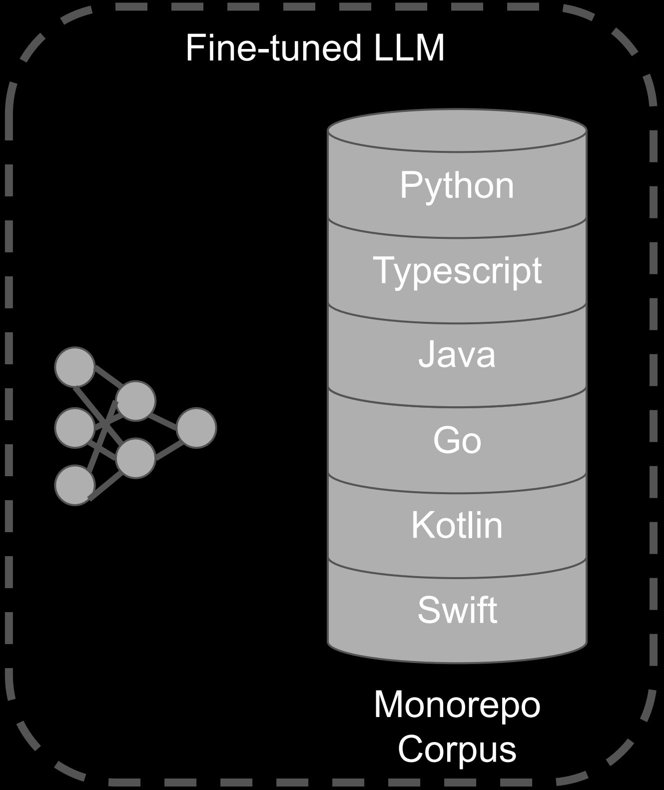


In-House Coding Assistant

Goals:

- Increase acceptance rate by +10%
- latency <1s for 100 tokens

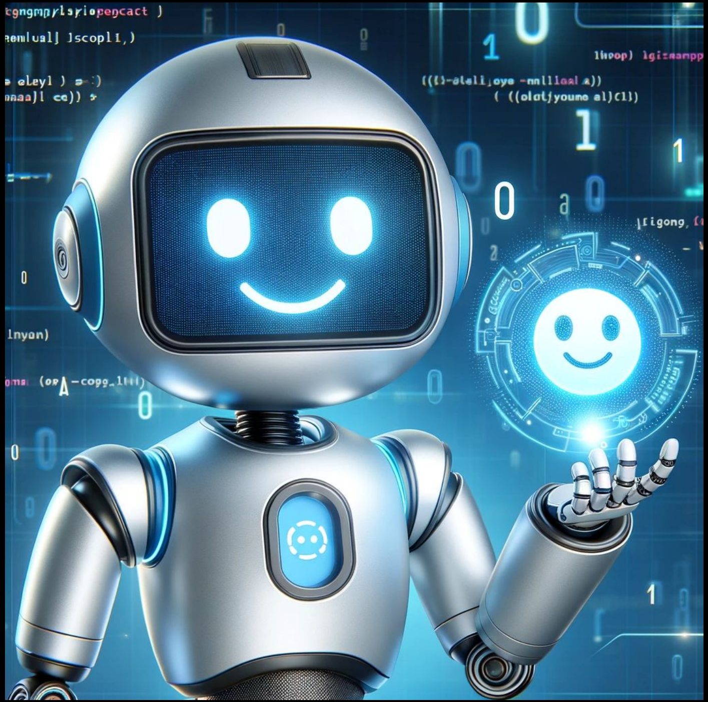




Code Assistants

Downsides

- 6 months of work
- Underfunded
- Always playing catch up



What we learned

- MVPs are easy, productionisation is hard
- Latency requirements vary per tool
- User Experience matters
- UI surface cannibalization is a risk
- Follow ecosystem principle
- Continuously evaluate landscape

Focused on GitHub Copilot adoption & evangelism

Building on Industry Tools

Reusable Components

Code Context Gathering

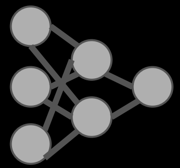
Summarize & rank code context to provide best input to use-cases:

- Data Race Fixer
- Linter Warning FIxer
- Crash Fixer

Gather telemetry

Assistan
t
Proxy

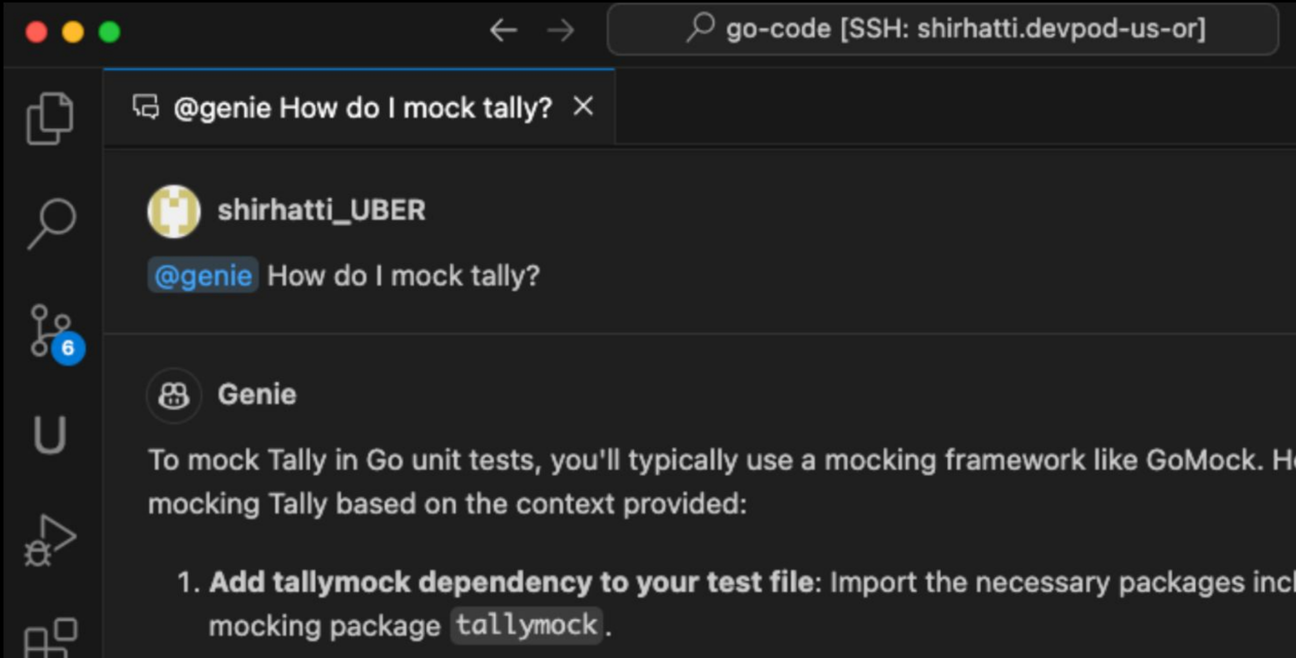
Fine-tuned LLM



Custom model with knowledge of internal libraries, custom frameworks, and company-specific best practices

Extend with chat @partcipants

@genie for monorepo knowledge base queries



Hands-on workshops

Running once per month to spread knowledge

Experience iterating with LLMs and probabilistic outputs

Internal Evangelism Content

Chat participants

@workspace
@vscode

Extensible!

Chat commands

/doc
/explain
/fix
/tests
…

Chat context

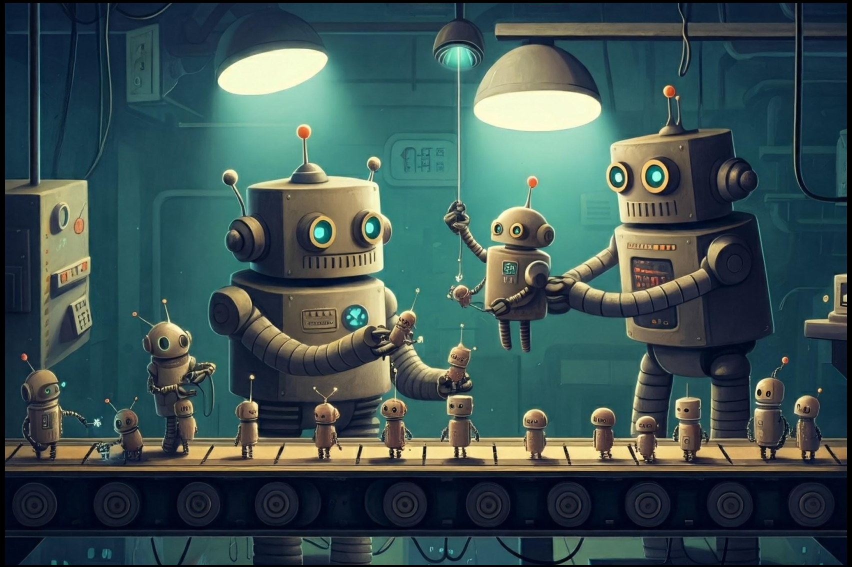
#codebase
#editor
#file
#selection
….

A series of codelabs that teach these engineers about providing context, using chat participants, and commands to refactor code and generate tests

Coding Assistants

Future

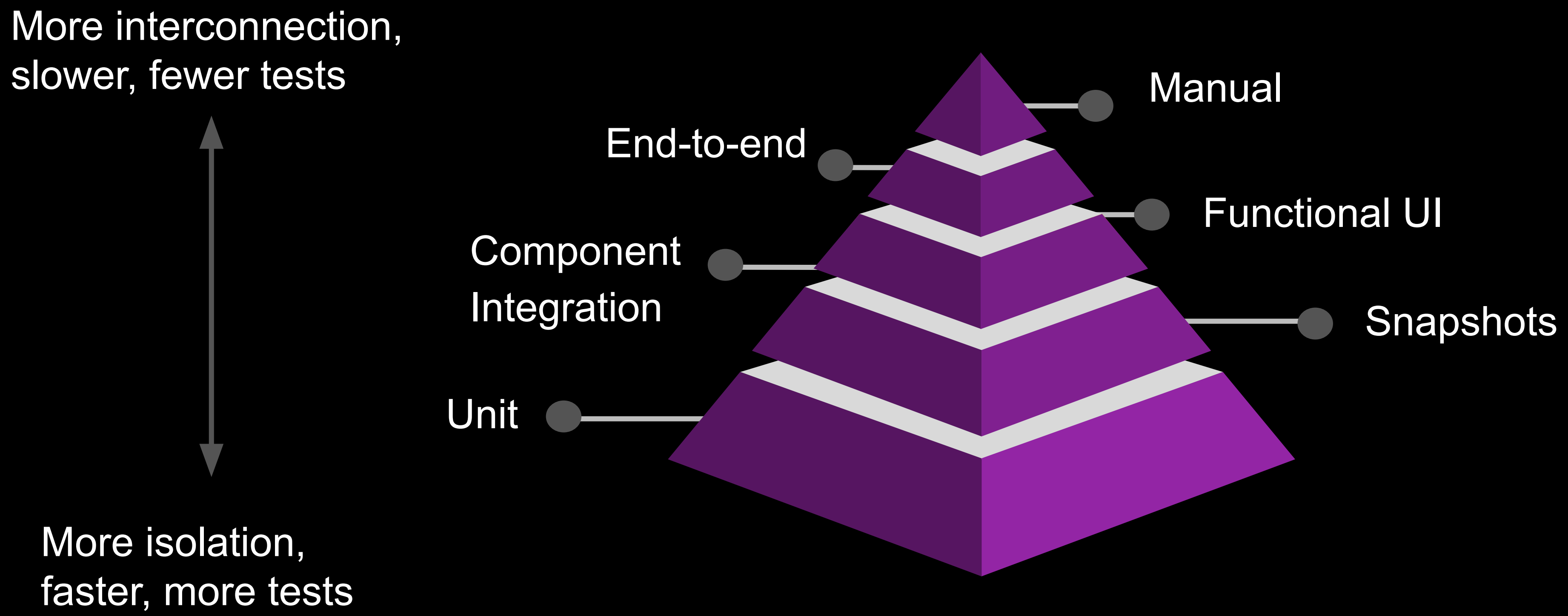
- Platform Native
- Vendor Fine Tuning
- Extensibility Increases
- Open-source clients
- Multi Vendor Landscape
- Enterprise Requirements



DESIGN DEVELOP RELEASE OPERATE MAINTAIN

Generating Tests

Classic Testing Strategy

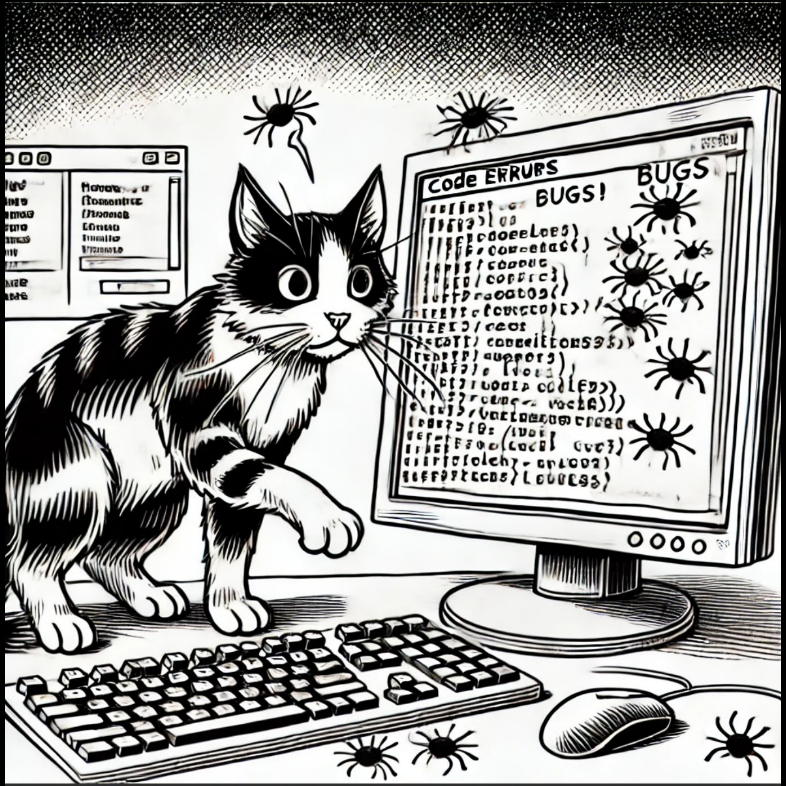


Testing Challenges

Writing good tests can be hard

Maintaining many different types of tests is tedious

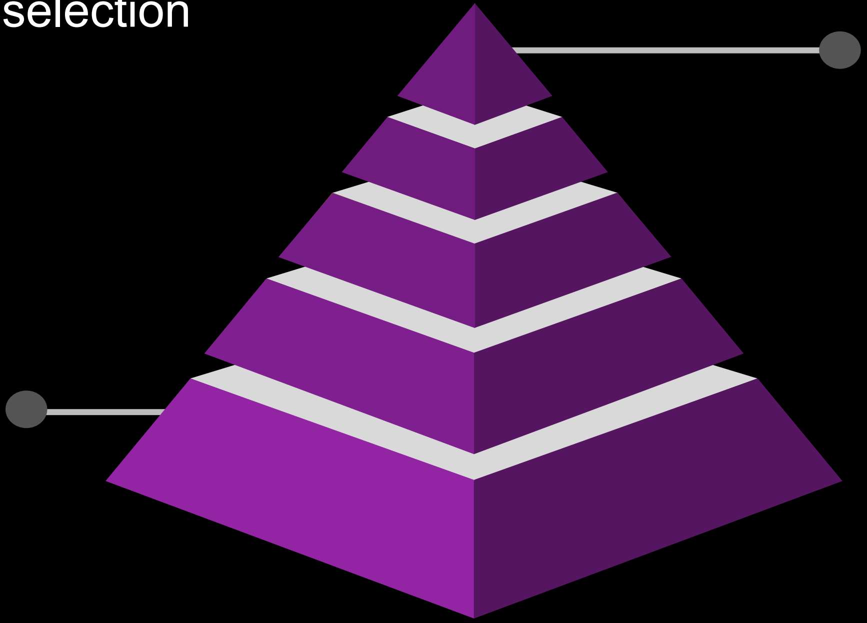
Permutations of languages, cities, experiments, platforms



AI-Driven Testing Pyramid

- Analyze quality of tests
- Predictive test selection

End-to-end AI testing agents

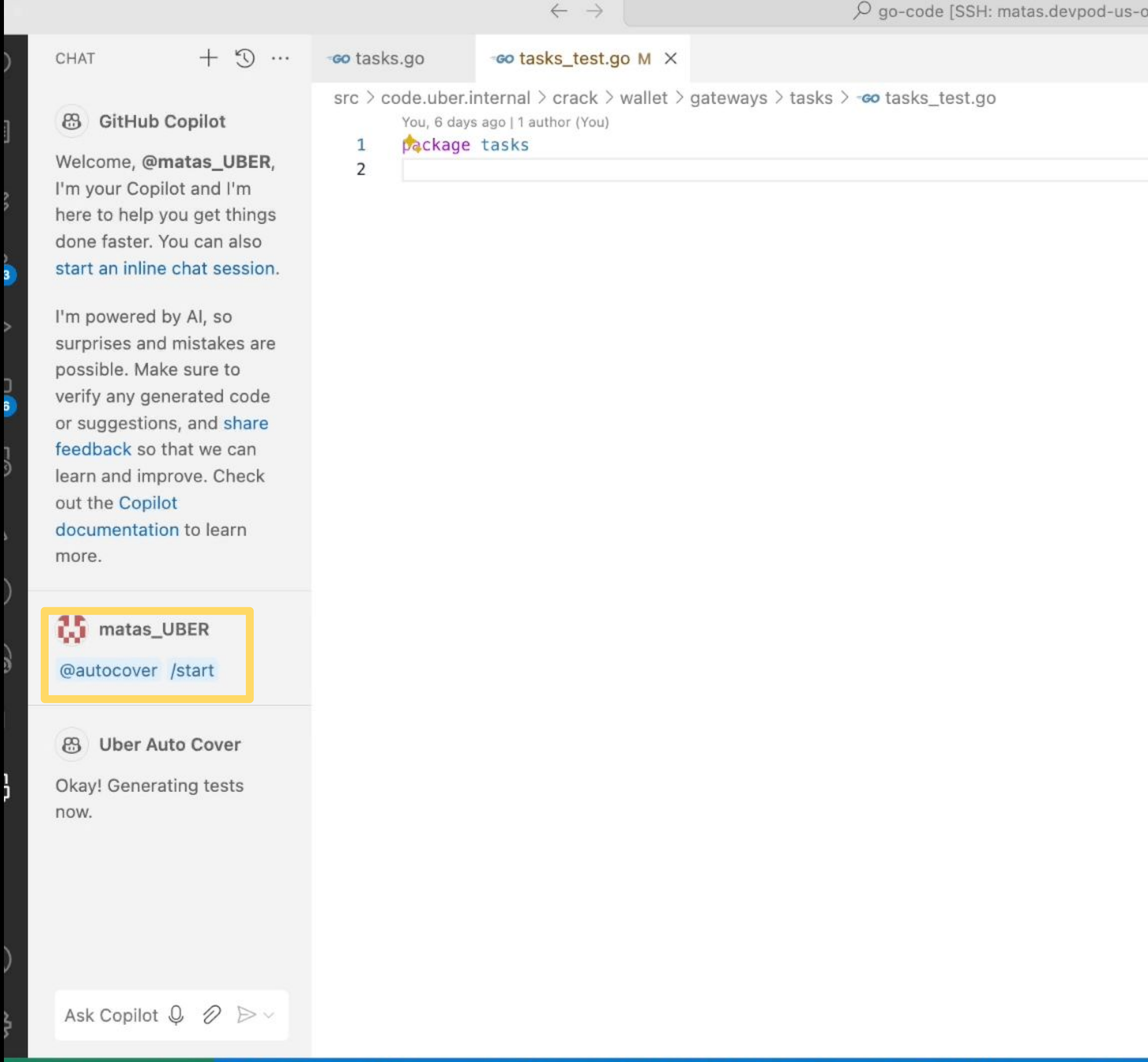


AI-powered code generation of unit tests

AutoCover Requirements

- Keep the developer-in-the loop
- Focus on regression tests
- Increase coverage

AutoCover in action

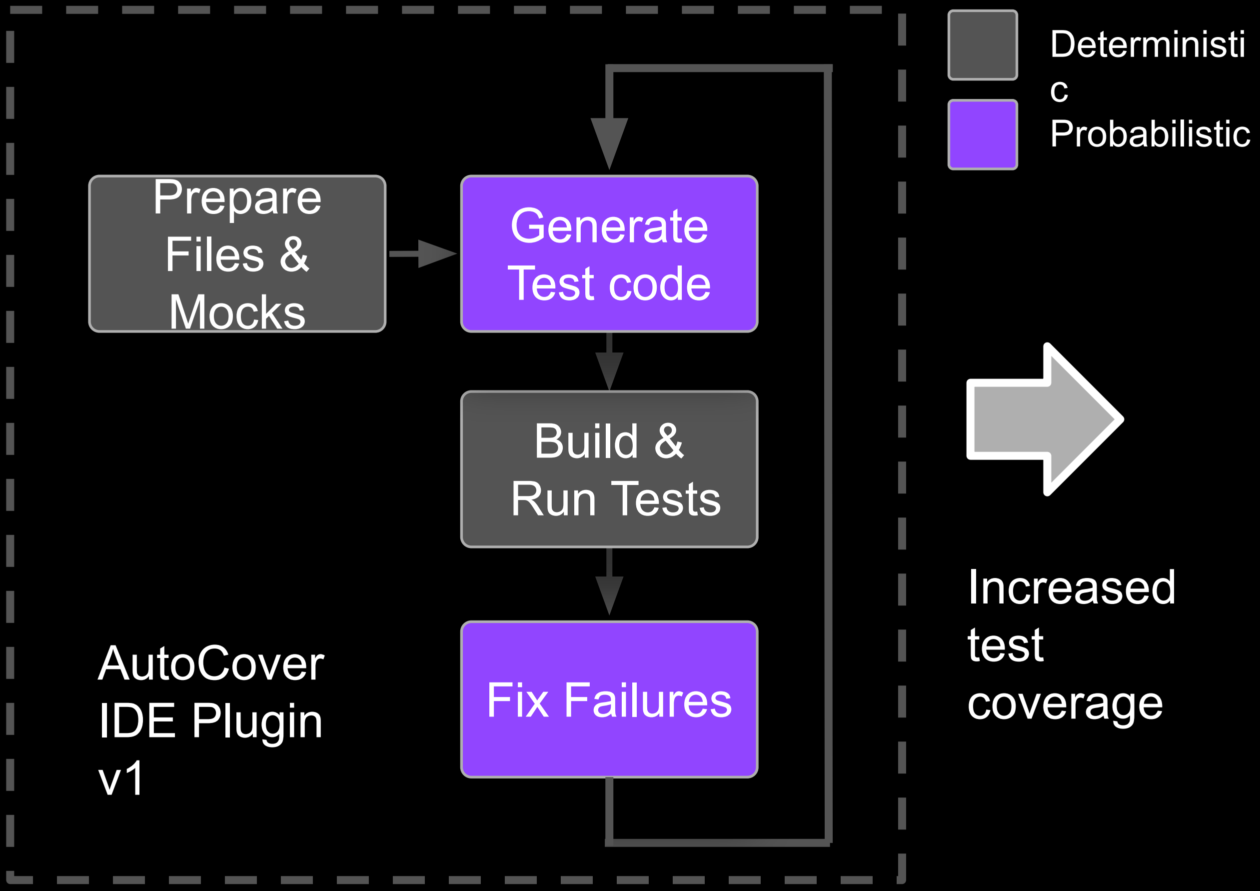


Tests are streaming in...

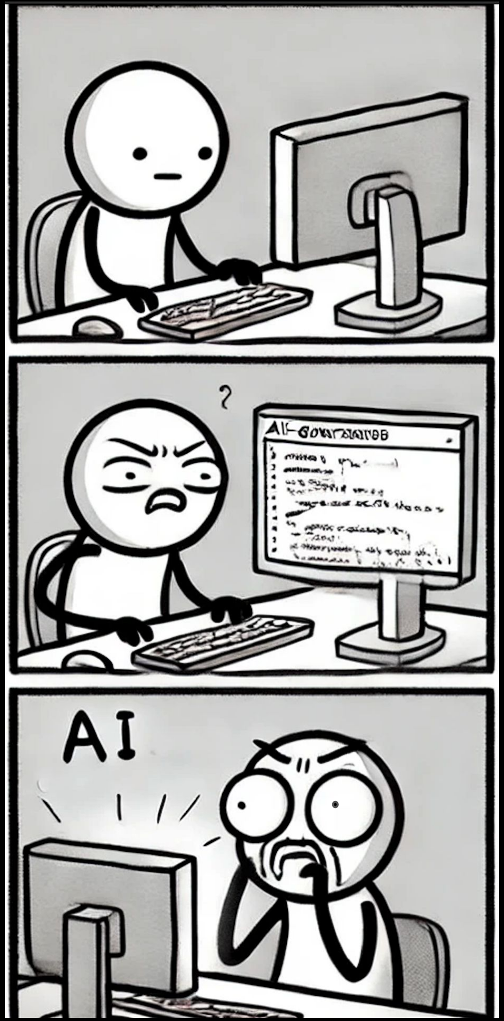
Break up the problem

Mimic human heuristics

Agentic design, built with LangGraph



Automating away all this tedious work…

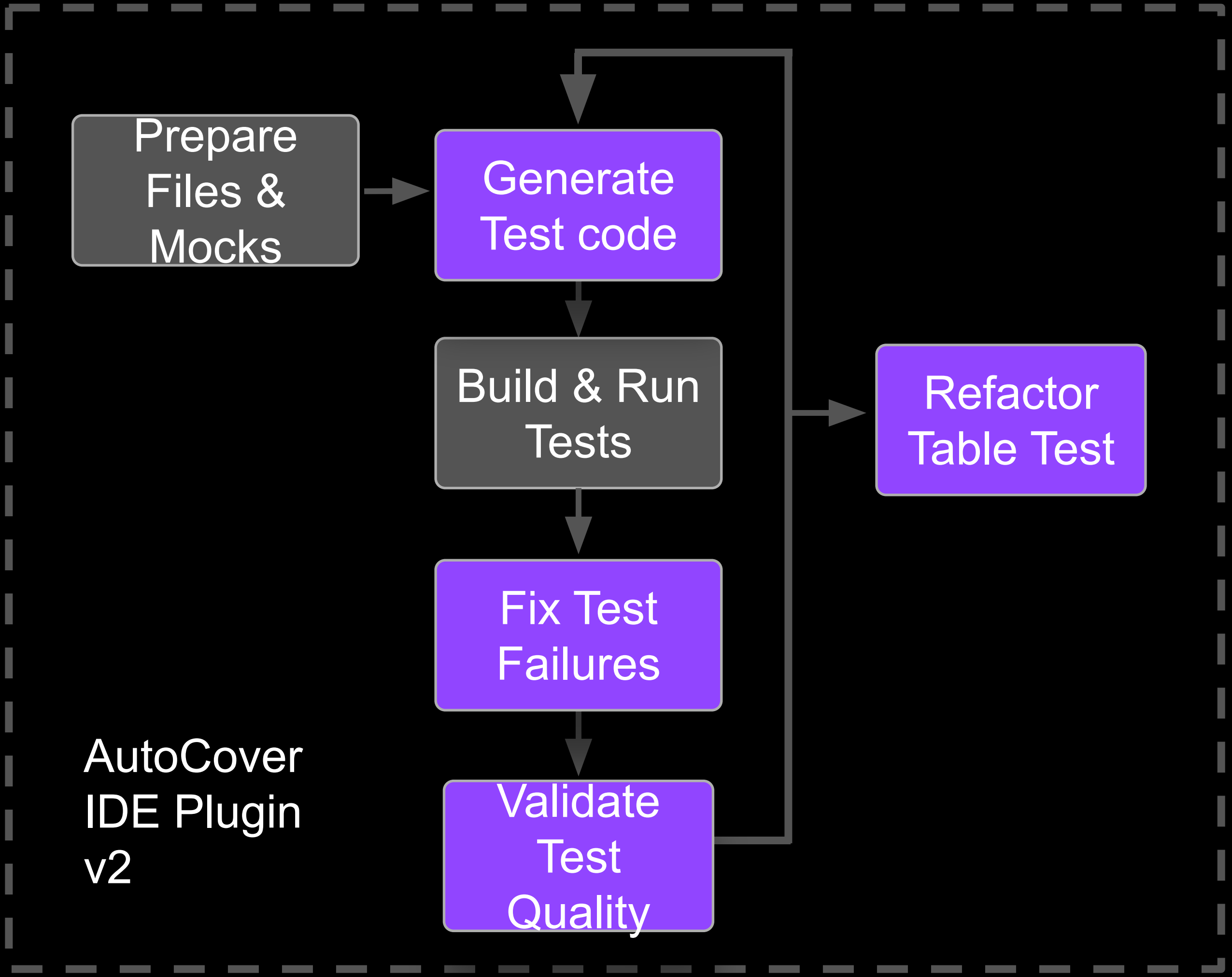


Is it writing “change detector” tests?

This needs to be labeled “generated with AI!”

Validation step: Check assertions against intent

Refactor step: Adopt best practices like the table pattern



What’s next

IDE/CLI for humans

- Table test refactor
- Validate test quality

Headless mode

- Shift-right, runs on CI
- Improve quality of existing test code

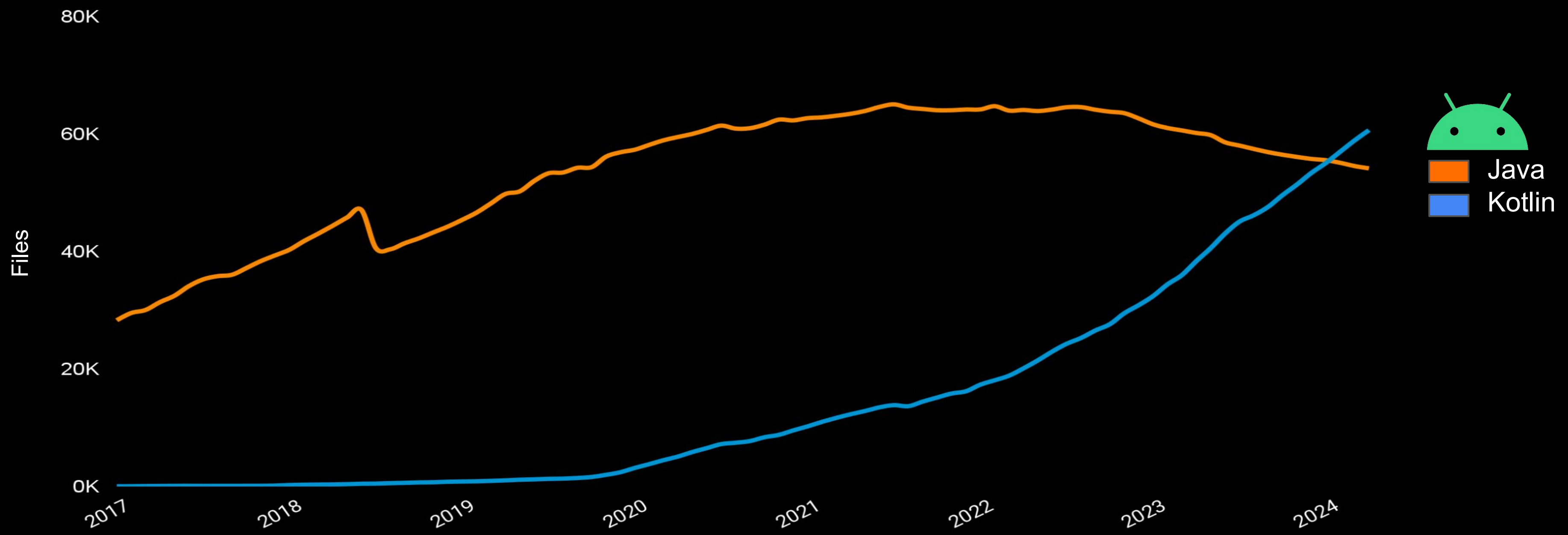
Mutation testing step

- Inject bugs (“mutants”) into the source code
- See if it finds bugs
- Generate mutants with AI

DESIGN → DEVELOP → RELEASE → OPERATE → **MAINTAIN**

Migrating To Kotlin

Kotlin History at Uber

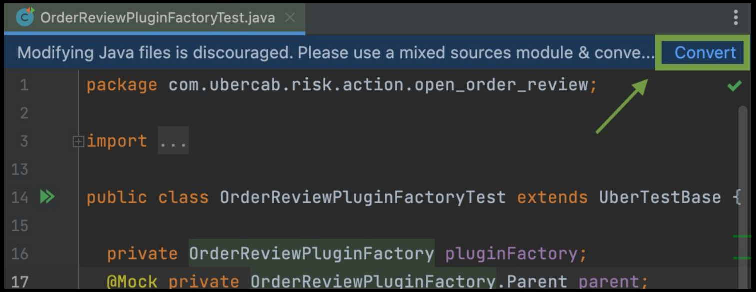


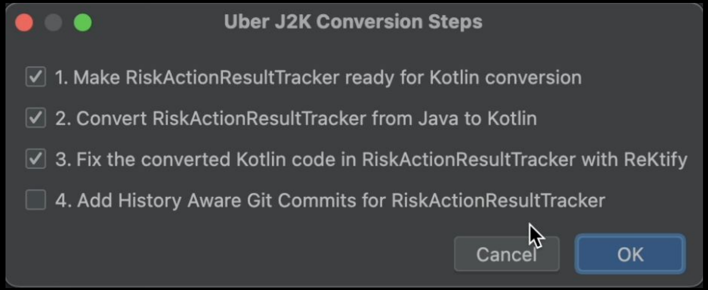
2017	2018	2019	2020	2021	2022	2023	2024
Minimal platform usage Google standardizes	Published evaluation	Limited product support	Full Product support	Organic Adoption	Manual Migration s	Assisted Migrations Mixed Sources	Banned Java Joined Kotlin Foundation

Kotlin Migrations

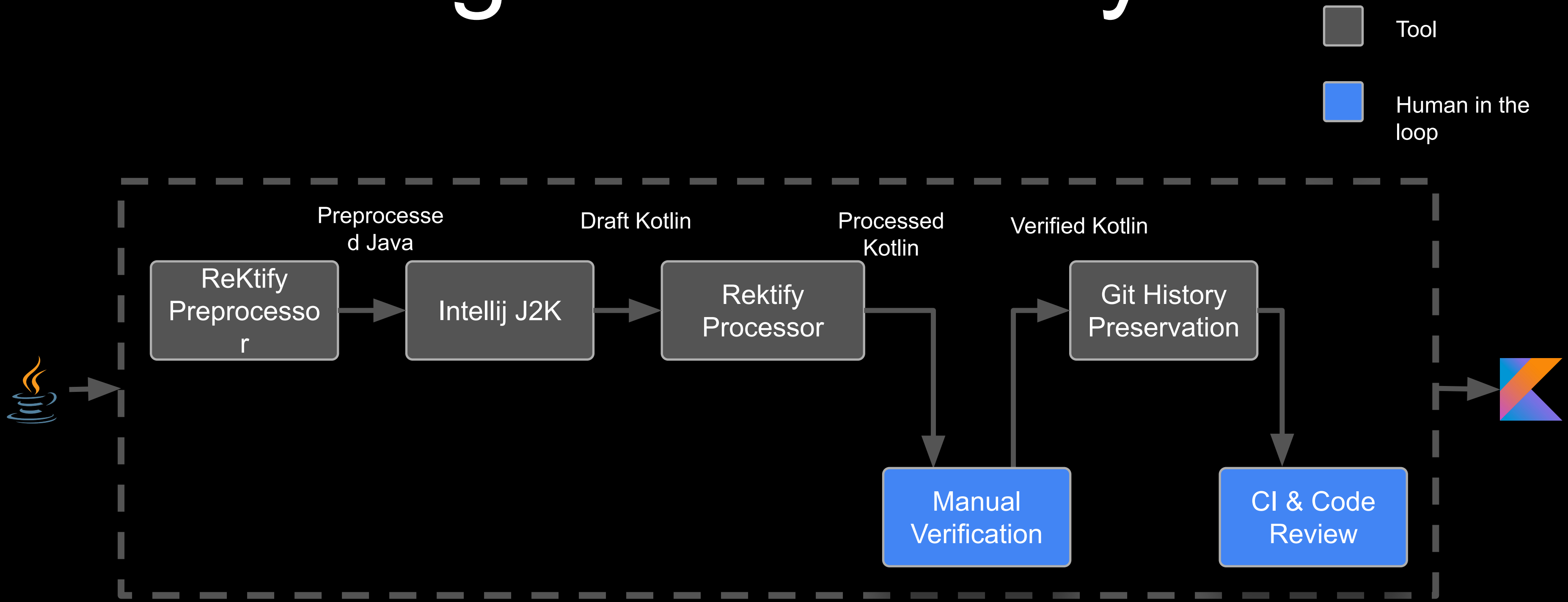
Decentralized

- Workflow incentives
- Industry standard
- Developer assisted





Kotlin Migrations Today



Rektify Processors

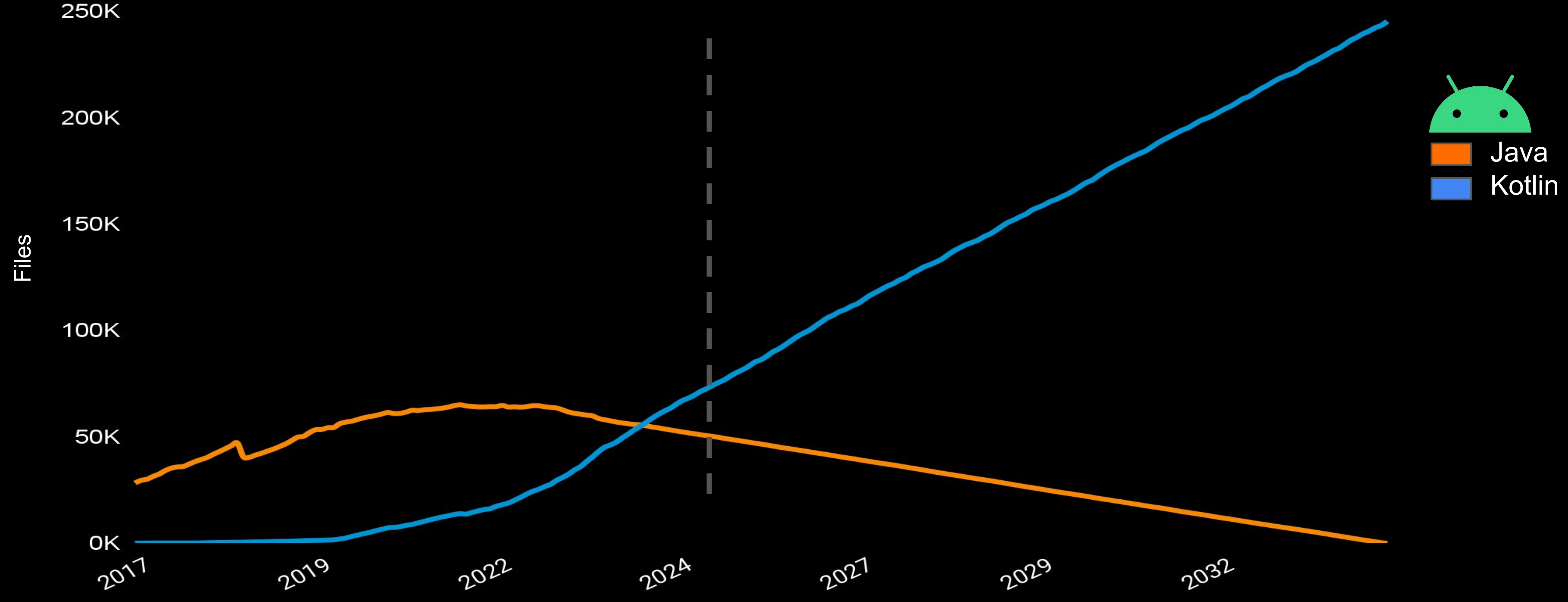
Pre Processors

- Nullable Annotations

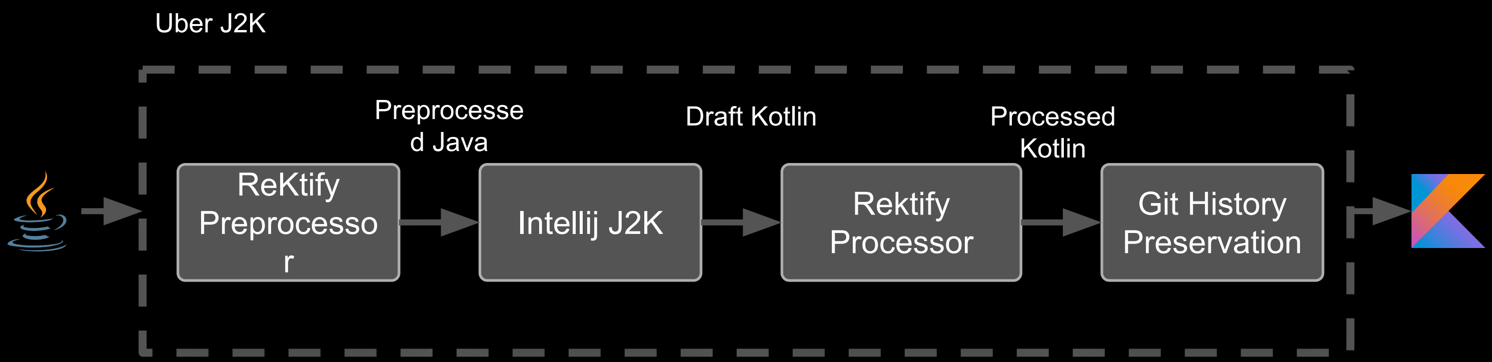
Post Processors

- AndroidTextUtilsRule
- CaptorAnnotationRule
- GuavaStringUtilsRule
- LambdaExpressionRule
- MockAnnotationRule
- RemoveInitMocksRule
- AutoDisposeRule

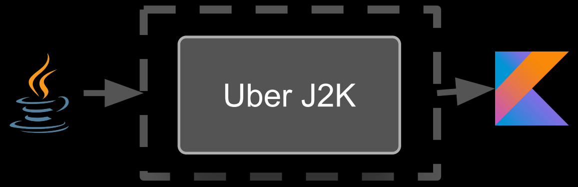
Kotlin Migrations



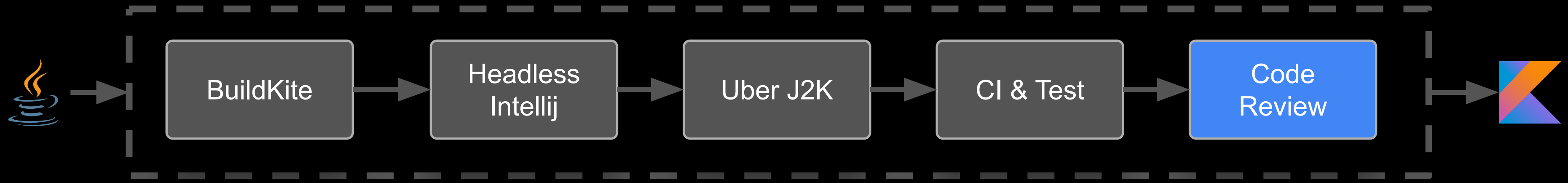
Automated Kotlin Migrations



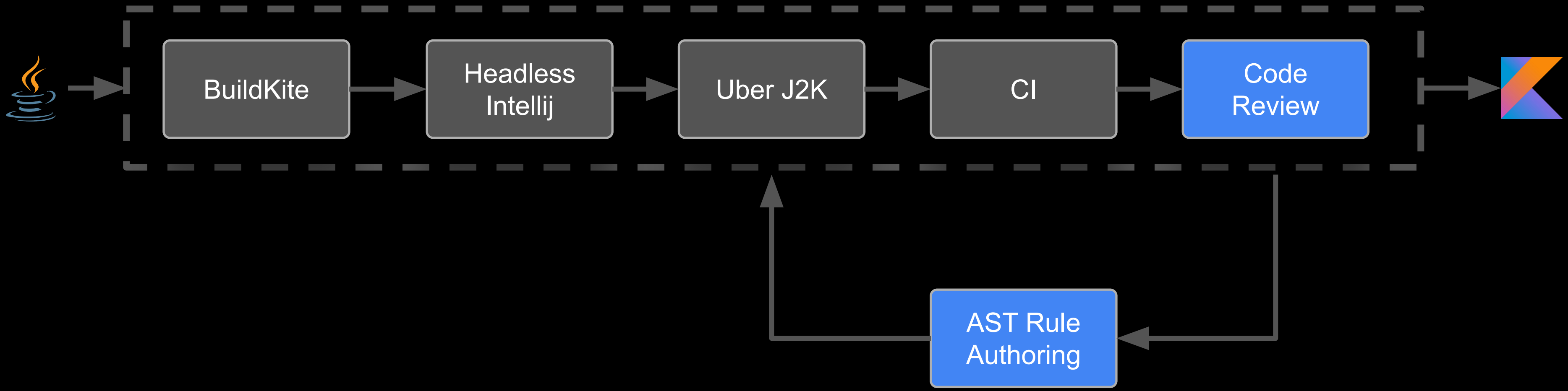
Automated Kotlin Migrations



Automated Kotlin Migrations



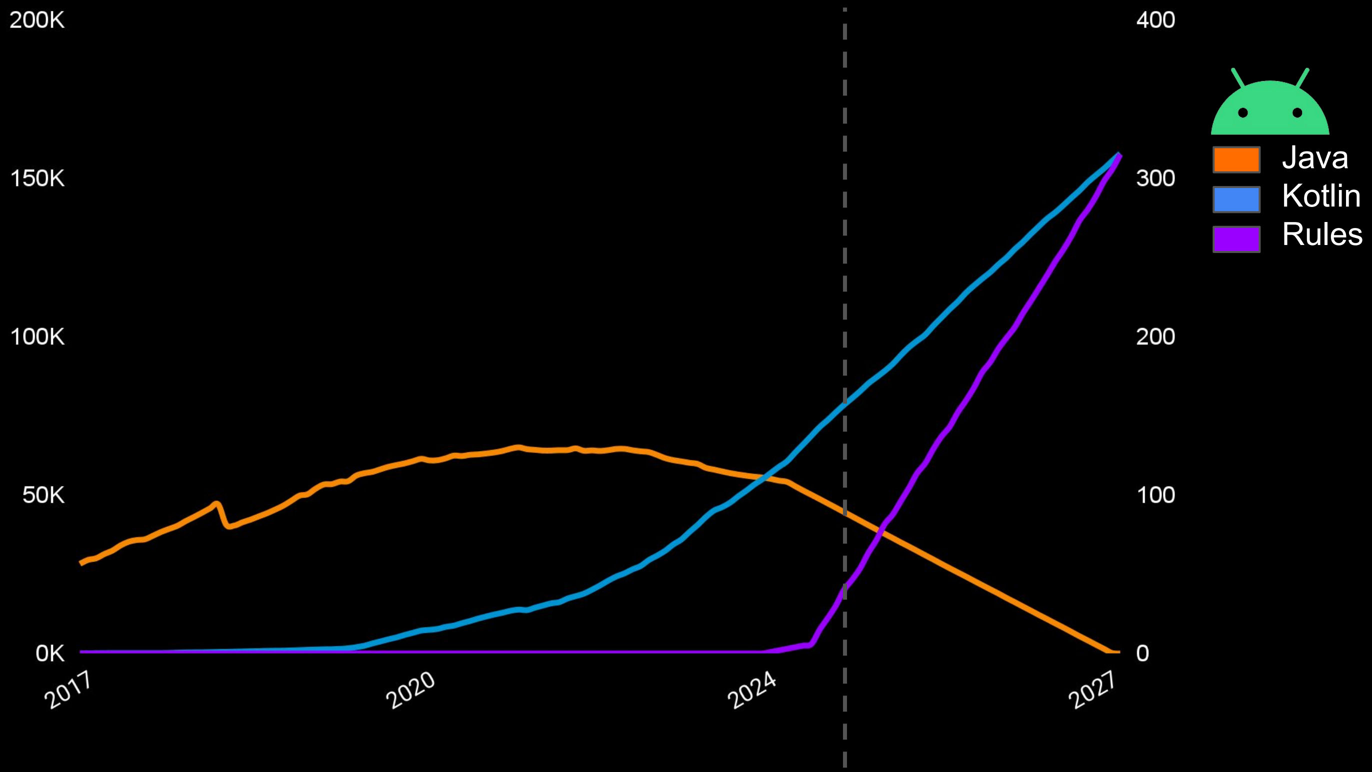
Automated Kotlin Migrations



Kotlin Migrations

Centralized

- Industry group
- ~3 Years



Can AI go faster?

LLM Kotlin Migration

Positives

- Flexible use cases
- Fast to deploy

Negatives

- Hallucinations possible
- High risk failures
- Humans are fallible

Combining AST + LLMs

AST + LLM

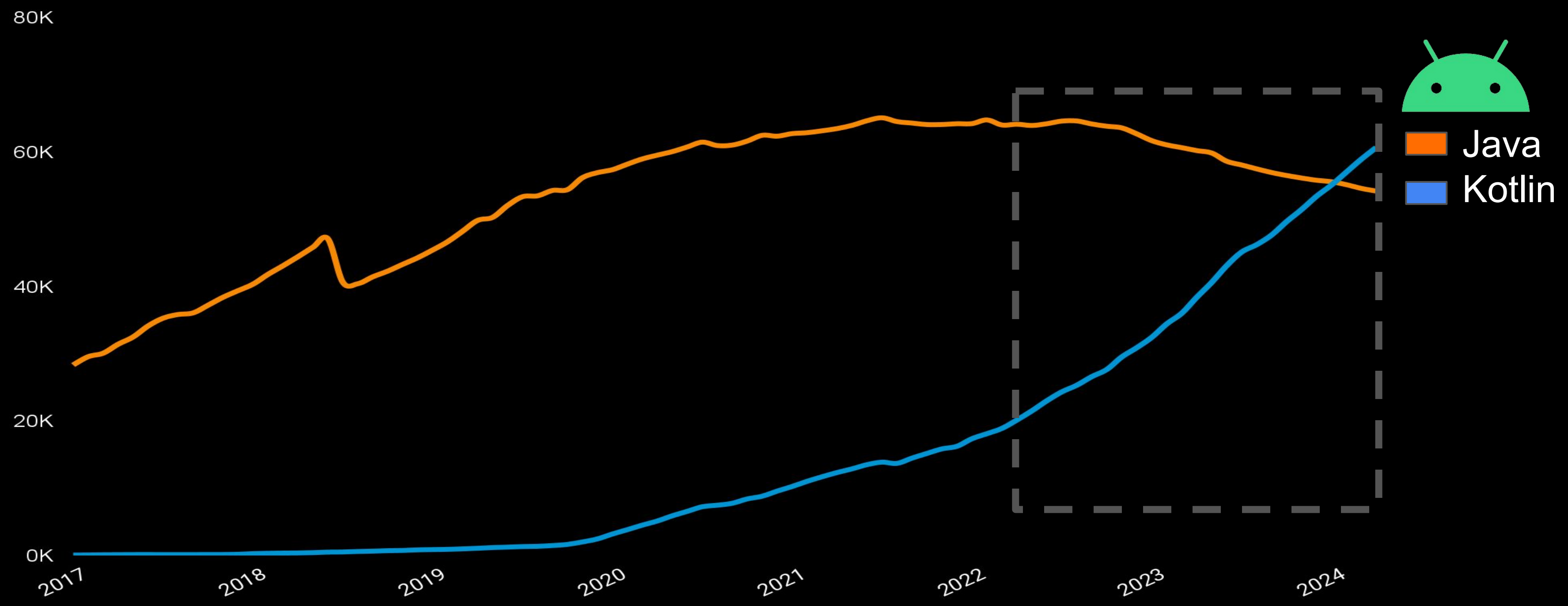
Positives

- Prior Art
- Deterministic
- Faster than human authored Rules

Negatives

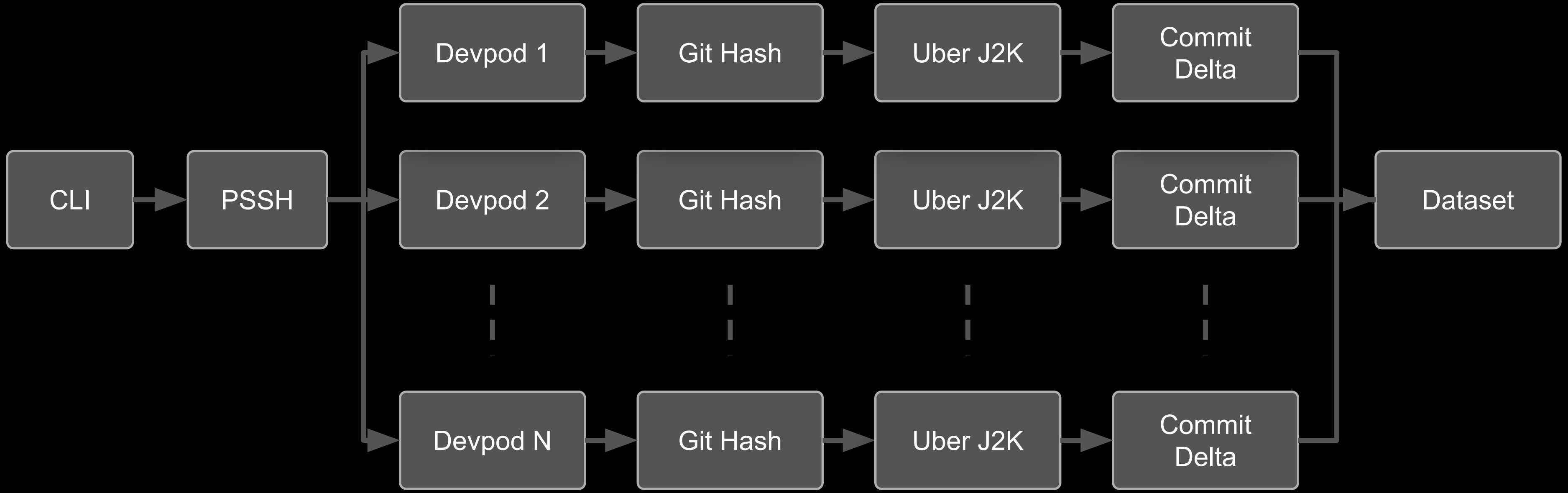
- Slower than LLM only

Dataset



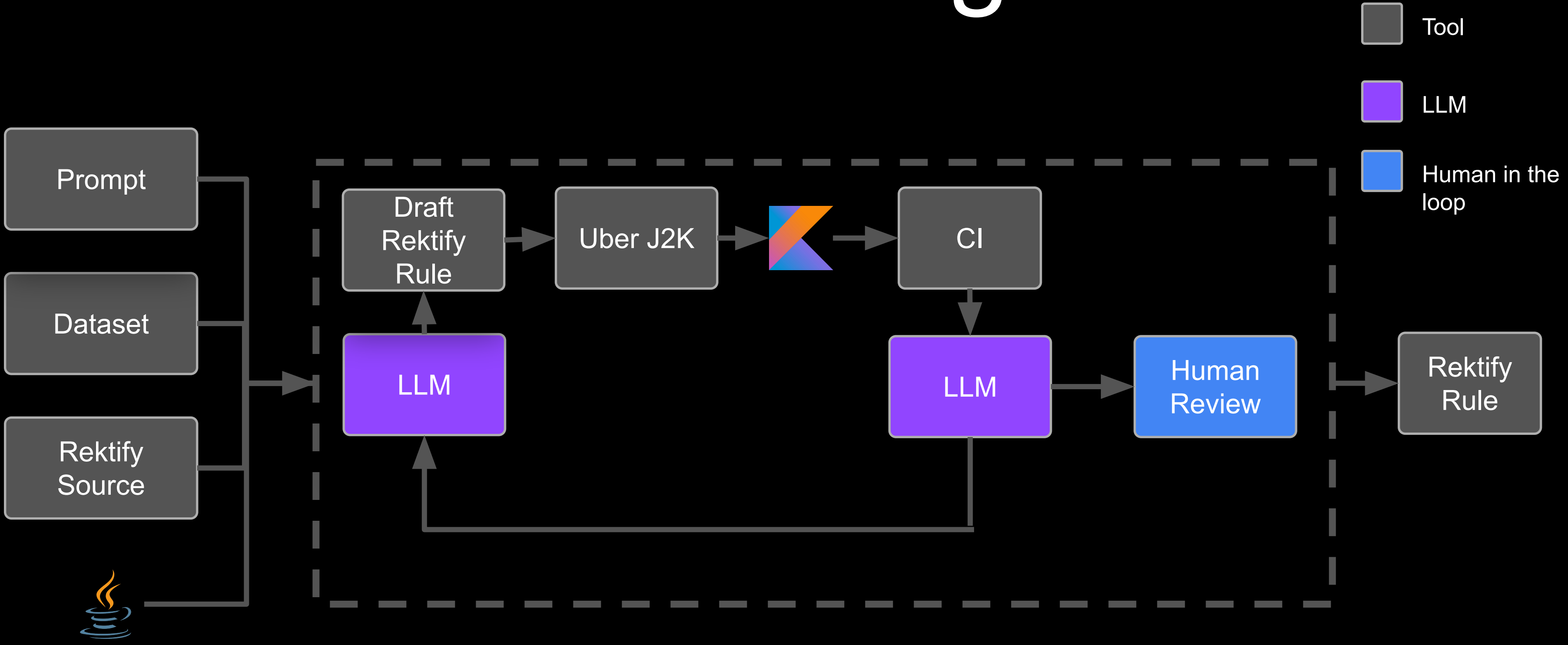
80K
60K
40K
20K
0K
2017
2018
2019
2020
2021
2022
2023
2024
Java
Kotlin

Dataset



CLI
PSSH
Devpod 1
Git Hash
Uber J2K
Commit Delta
Devpod 2
Git Hash
Uber J2K
Commit Delta
Devpod N
Git Hash
Uber J2K
Commit Delta
Dataset

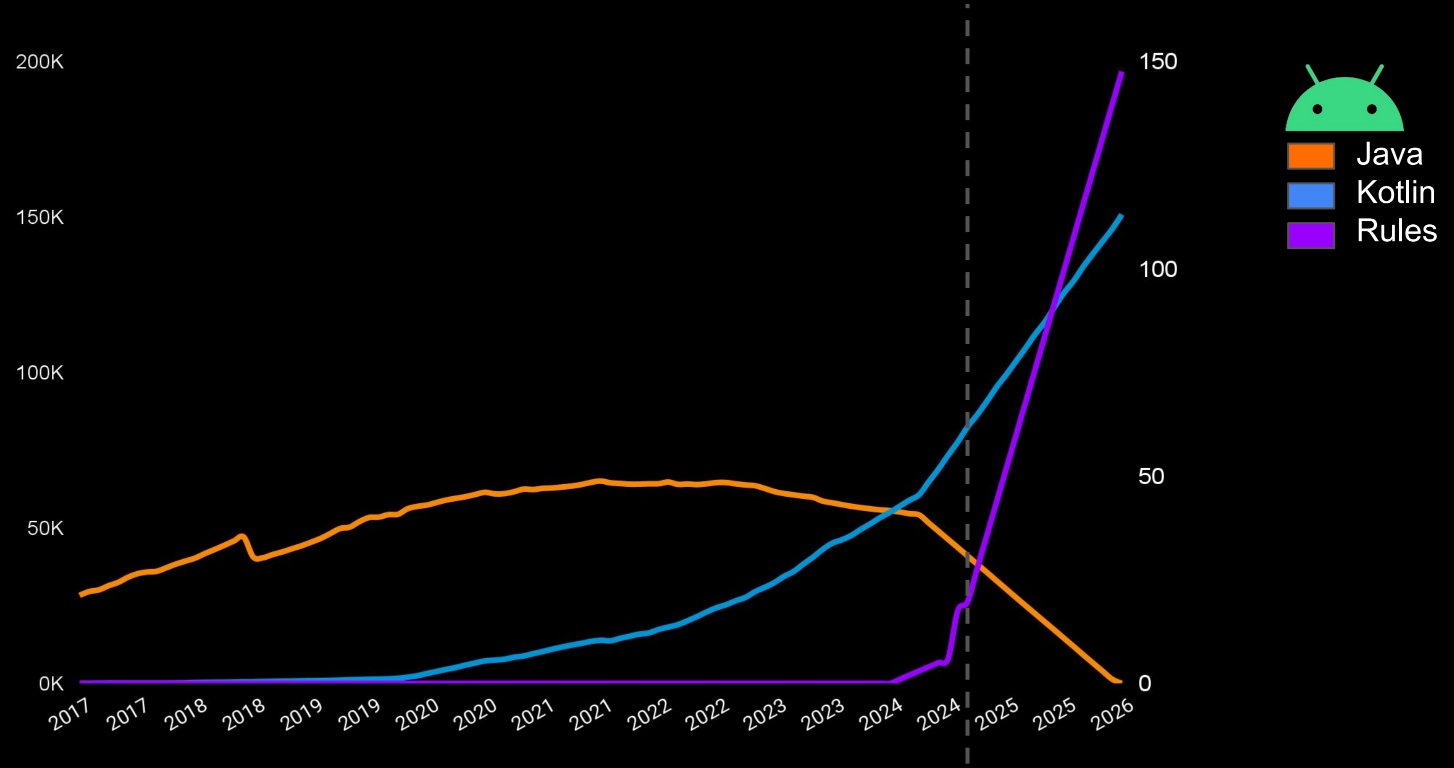
LLM+AST Kotlin Migrations



Kotlin Migrations

LLM+AST

- 50% Faster
- 18 Months



Finishing the Migration

Challenges

- Rollout risk
- Begin in low risk areas
- Categorization of rules
- Noise fatigue

Questions

- LLM fallbacks
- Batch size
- Speed of innovation

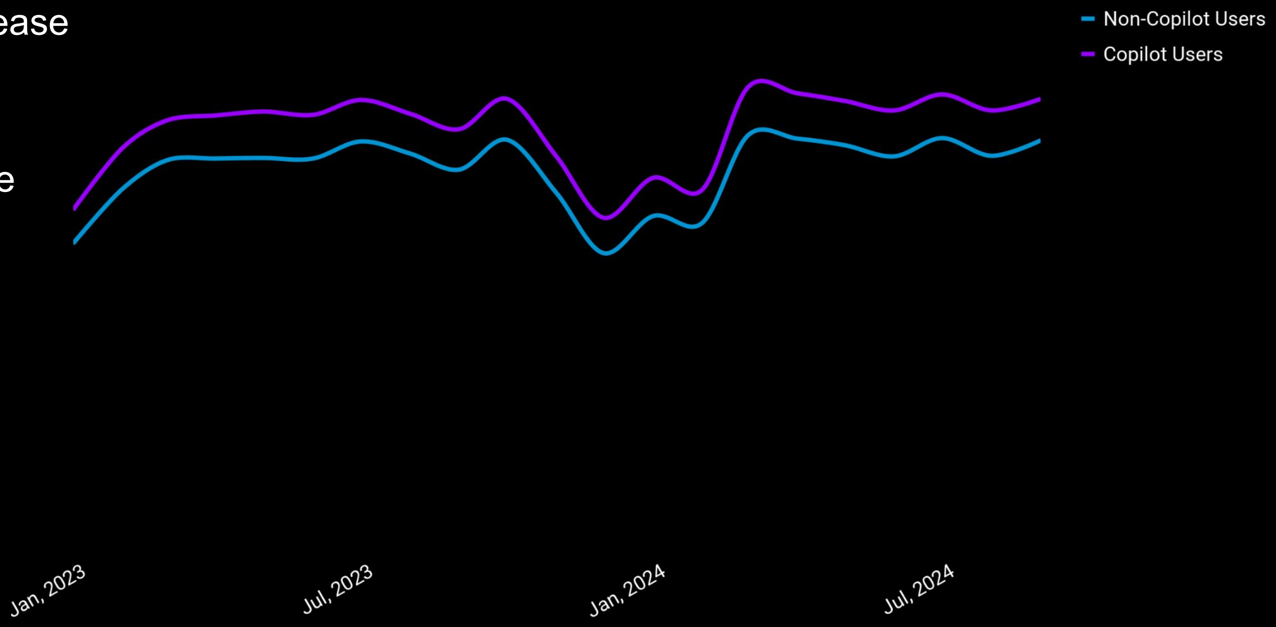
Wrap up

Measuring AI impact

~10% PR velocity increase

~30% Acceptance Rate

~60% Adoption



Measuring AI impact

Challenges

- Fragmentation
- Organizational Cost

Risks

- Side Effects
- Wrong Investments
- Missed Investments

Measurement Philosophy

Lead with qualitative

Normalize quantitative impact on developer hours saved to prioritize bets

63%

Developers report **significant** increase in productivity

1000+years

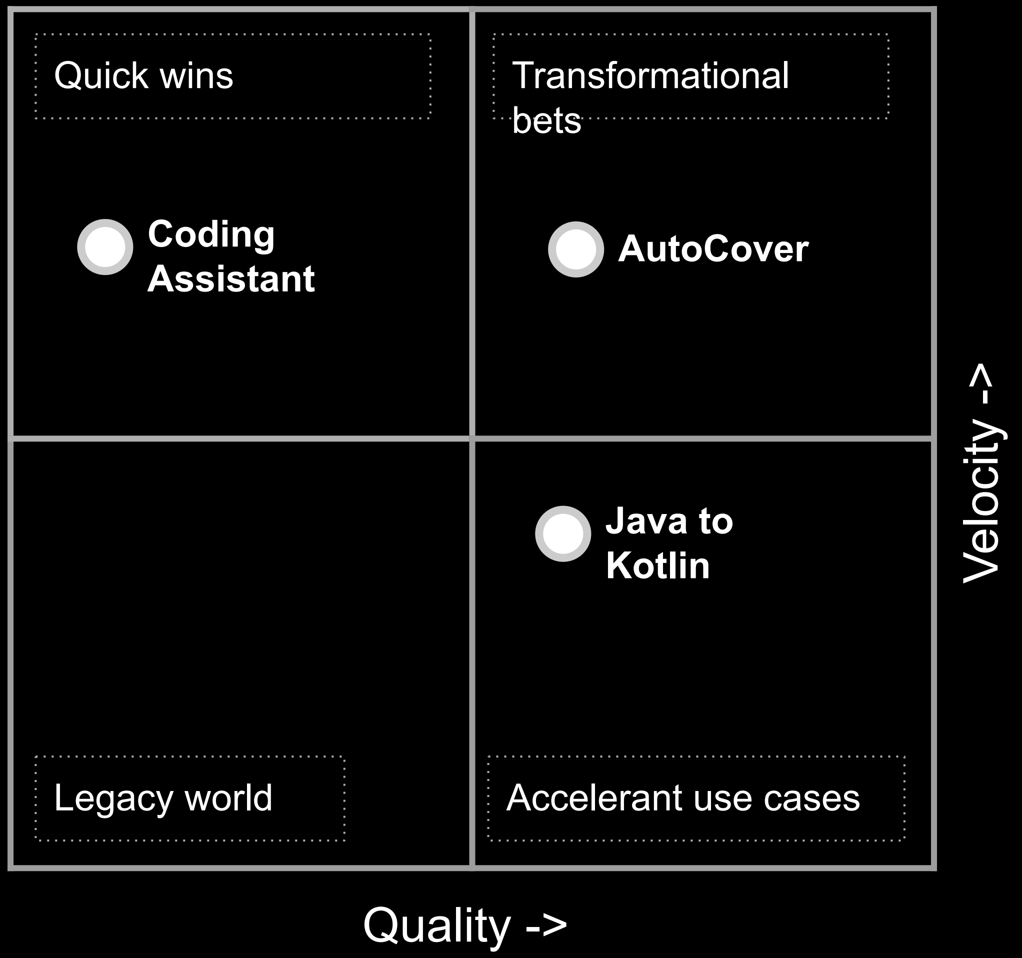
Potential impact of automating away of technical debt

Opportunities

Quality and velocity

Manage expectations and hype

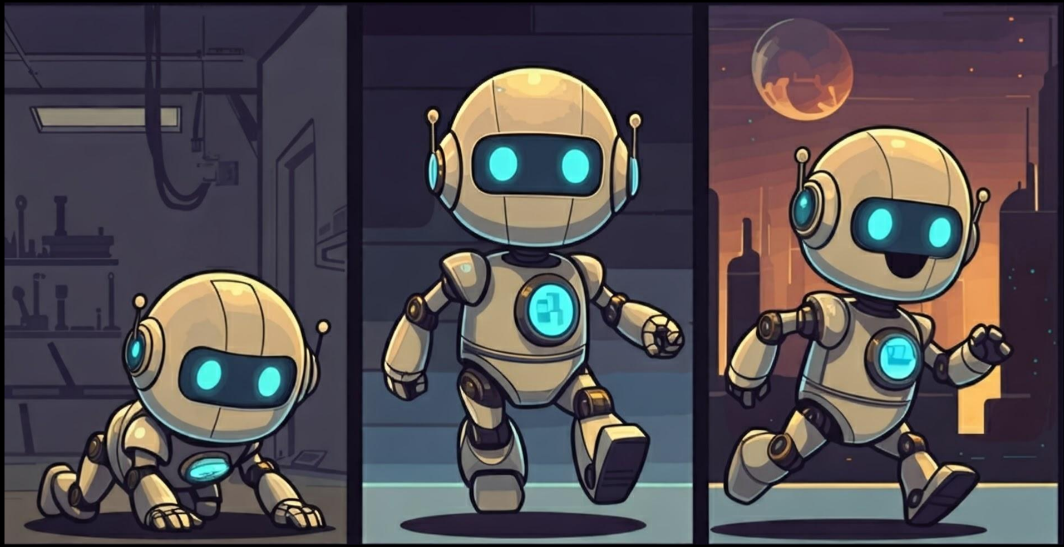
Combine deterministic approaches with new probabilistic capabilities



Crawl, walk, run

Find the sweet spot of what's possible now and will be possible soon

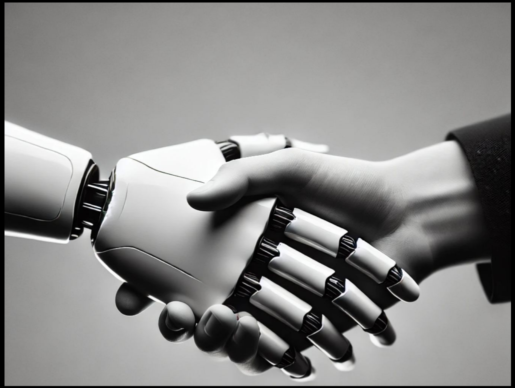
But be ready for what's coming



Demand for good software is near ∞

AIs

- Reduce toil for migrations
- Increase test coverage
- Give humans more options
- Help humans think about problems



Humans

- More time building
- Focus time on the craft of software engineering
- Break down complex problems
- Define architecture
- Set best practices

Questions?

Adam Huda
Sr. Eng Manager
AI Foundations
& Developer Experience

Ty Smith
Principal Eng
Developer Platform