diffblue
AI for Code

White paper

The Next Generation of AI for Code

How autonomous code writing helps enterprises accelerate change and increase productivity

Introduction

If you lead an organization that runs a lot of Java in production, you may sometimes set up for work feeling like an ancient Roman architect...

Watch your head

It was a tradition 2,000 years ago that the architect and construction leader both stood under a new major edifice the day the scaffolding was removed. A design flaw or construction error meant you perished under tons of rock and concrete.

If this feeling sounds familiar, you will want to make sure your IT project has been thoroughly tested before pushing code to production. Unit tests are a key part of that process. They help to ensure business continuity. Quick to run at an early stage in development, unit tests have become a key enabler of fast delivery of high-quality code.

The next generation of AI for Code

Java remains one of the most important coding languages in the enterprise. Millions of developers around the world use it. Most large enterprises rely on it. But the size and complexity of applications in such organizations make effective unit testing extremely difficult.

Diffblue has created a new platform that automates Java test unit writing through the power of artificial intelligence (AI). It autonomously writes unit tests 250x faster[1] than your developers, producing tests that can be read by humans. All without manual intervention.

The benefits of automating unit test creation are compelling. Expensive developers can be freed to write code that moves the top line instead of spending up to a third of their time[1] doing a task better performed by AI.

Although many AI projects are as-yet unproven science experiments in real-world business settings, Diffblue's AI engine (based on the latest advancements in Reinforcement Learning), pays off fast. Diffblue Cover:

- Writes correct, fast-running unit test code autonomously, without developer intervention
- Automatically refactors Java code to make it more testable
- Slashes the time and cost to run tests in CI (Continuous Integration)
- Understands test coverage and code risk across your organization
- Automatically shifts complex functional tests left into unit test suites

But ensuring continuity isn't enough for businesses today. Continuous change is now a fact of life for organizations that want to remain competitive. That change can mean many things: migrating applications from an old language, moving systems to the cloud, migrating applications from a mainframe, rearchitecting code monoliths into microservices, adding new features and services, improving the customer experience. And more. Unit testing - and Diffblue Cover - help across the board.

What is Unit Testing?

American software engineer Kent Beck pioneered the idea of extreme programming two decades ago. One of the core tenets of extreme programming is unit testing. Beck recognized the need for more confidence that software would run as intended – but during coding, prior to QA or production release. It's very expensive to correct software in the wild, not to mention the cost of reassuring unhappy customers. For the Java programming language, Beck created JUnit and popularized the concept of unit testing. Today most enterprises embrace unit testing as a best practice.

A unit of code – which can be just a few lines long, or hundreds – is the smallest part of a program that can be maintained and executed independently. A suite of unit tests, typically dozens, checks that the unit of code executes as intended. That ensures that when you stack the units together the program runs correctly as designed.

But despite the value, unit testing is a highly visible pain point for most enterprises. It is essential to writing high-quality software but it delays delivery times and drains developer resources from writing more functional code that powers modern organizations. Nearly 9 out of 10 people working in software development[1] say they need a better solution to testing bottlenecks. Developers don't like it either. Four out of 10 software engineers[1] would prefer to never write another unit test.

So automating unit testing not only improves an enterprise's efficiency and productivity while reducing risks, it also makes developers happier. It turns out that AI provides a win-win solution.

90%

of people in software development need a better solution to testing bottlenecks

40%

of Java developers want to stop writing unit tests

The Power of AI Technology

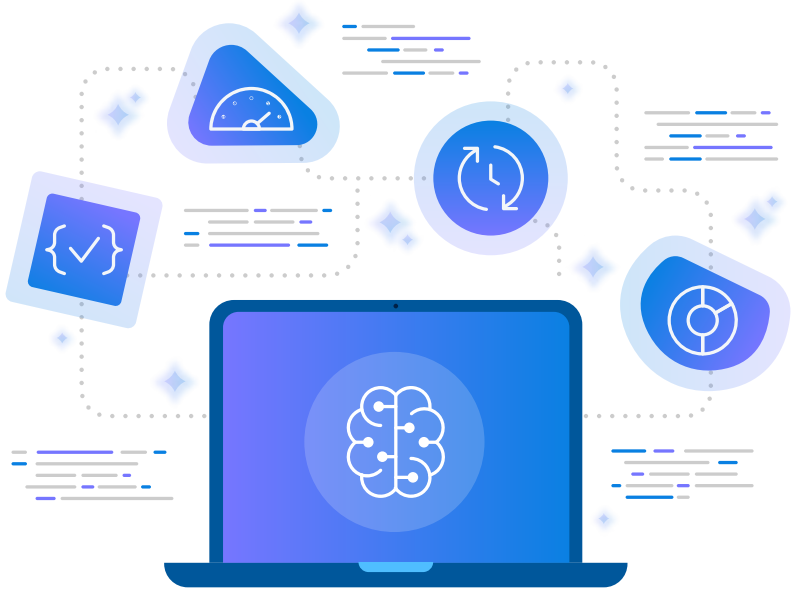
Diffblue Cover is powered by an AI technique called reinforcement learning. Instead of being pre-trained on example data, the learning happens in real-time as the model seeks to find the best answer by searching the space of possible answers. It follows the most promising trajectories and backtracks if things seem to be getting worse.

The algorithm is guided by a reward function that seeks to optimize the long-term reward. This is called “probabilistic search” – a method where the space of potential solutions is sampled. The algorithm then spends more time searching regions with a greater probability of a good solution.

The best-known example of reinforcement learning is Google’s AlphaGo algorithm, where the search task is to find the best move in the game of Go. Reinforcement learning is a popular AI approach when the number of potential solutions is so large that an exhaustive search is unfeasible. In the case of Go, there are more potential moves than atoms in the known universe. So to find a good move requires a probabilistic approach. (Coincidentally, the co-founders of Diffblue were faculty members in the same Computer Science department at Oxford University as the founders of Deepmind, creators of AlphaGo).

In AlphaGo, two neural networks are used to predict the next best move. The game-winning prediction is used for the reward function. The prediction of the best move is used to decide what moves to evaluate. The algorithm then repeats the process to maximize the game win probability.

Diffblue uses probabilistic search, but it doesn’t need costly neural networks to make predictions (that would be prohibitively expensive: according to Nature Magazine, AlphaGo used 1,202 CPUs and 176 GPUs to run). Diffblue Cover can run on a standard laptop computer.



How it works

To write a unit test, Cover evaluates the code and from that guesses at what a good test would be. It then runs the method under test. It evaluates the line coverage (how much of the code the test exercises) and other qualities of the test, and then predicts which changes to the test will trigger additional branches to produce higher coverage. Then it repeats these steps until it has found the best test in the time available (it is time-bound).

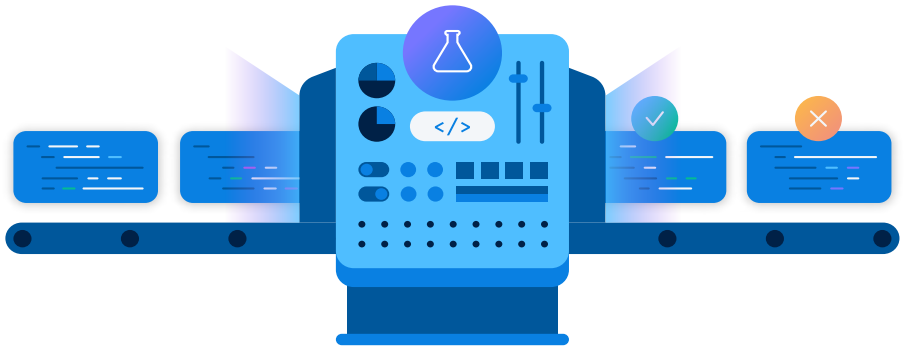
The return value(s) of the method and any side effects that were observed are used to write assertions on the test's behavior. It is not useful having good code coverage if the tests don't check what the code actually does. The result is a test that is known to work (it ran successfully) and produces specific coverage and assertion checks.

Probabilistic searches cannot guarantee they will find the best solution – or any solution – because by design they don't evaluate every possible case. In contrast to generative transformers, you don't get gibberish or an incorrect result in that situation. You know when you have failed! This approach allows for full automation in test writing because it won't produce tests that cause your pipeline to fail.

Even more coverage

Almost as valuable and useful, Diffblue Cover helps you surface code that may need to be refactored or rewritten because it's not testable - by man or machine. An example would be private methods that can't be called, so they can't be tested. Cover can implement some of the necessary code improvements for you automatically.

Some complex or data-driven code may not be technically untestable, but finding the right combination of behavior and data to meaningfully exercise the code can be difficult. Too often, such code lacks good unit tests. Integration tests or end-to-end functional testing are relied on instead. Cover uses trace recording to feed the reinforcement learning loop when it finds such code, to increase unit test coverage and accelerate cycle times.



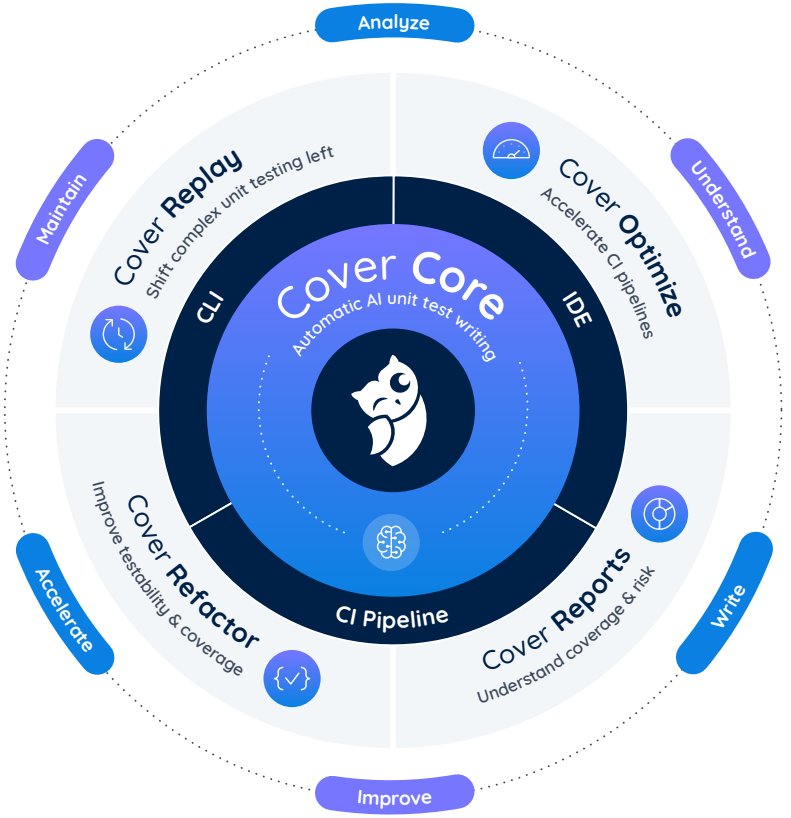
The Diffblue Cover Platform: AI that Pays Off Fast

Because it operates completely autonomously, without the manual intervention required by other AI-assisted coding tools, Diffblue Cover can deliver value fast.

Cover's core technology increases unit test coverage rapidly – it can write more tests in 8 hours than a typical developer could create in a year. The tests it writes run 100 percent of the time, and continue to deliver value as you make updates to your codebase: Cover automatically maintains your entire unit test library whenever code is modified.

Crucially, with Cover your code remains your own. It runs within a local user environment and does not require a connection to external servers, providing the security and confidence that enterprises require.

Automatic unit test writing is central to the value delivered by the Cover platform. But the core AI for Code engine supports other powerful features that further improve unit testing. Diffblue Cover is the only fully-automated, AI-powered solution that can not only write new code, but also improve existing code, accelerate CI pipelines and provide deep insights into the risks of change.



Diffblue Cover Key Features

Cover Core

The heart of the platform, Cover Core uses AI to automatically write human-like JUnit tests that are indistinguishable from those written by a developer. Cover Core lets you can find bugs sooner, deploy code changes faster, stop worrying about quality gates and coverage levels, and minimize tedious, unproductive developer effort.

Cover Reports

Cover Reports delivers valuable insights about your Java codebase. It visualizes the state of unit testing and pinpoints unique, actionable insights. Understand coverage, identify risk and prioritize effort. Benchmark and track over time.

Cover Optimize

Cover Optimize enables faster, cheaper, more flexible delivery of Java code by minimizing the time needed to run unit tests, whether on the developer desktop or in a Continuous Integration (CI) pipeline.

Cover Refactor

Cover Refactor suggests and applies fixes that improve the observability of Java code and make it more testable, thus automatically increasing coverage and reducing the risk of regressions.

Cover Replay

Cover Replay shifts more testing left and increases unit test coverage. It uses existing functional tests and live runtime behavior to automatically create new unit tests for data-driven code and applications.

About Us

Diffblue is the leading pioneer of software creation through the power of AI. Founded by researchers from the University of Oxford, Diffblue Cover uses AI for Code to solve the problem of effective unit testing. Capable of writing unit tests 250x faster than a human developer, Cover helps software teams improve code quality, expand test coverage and increase productivity, so they can ship software faster, more frequently, with fewer defects.

To find out more visit diffblue.com

1. Diffblue research