eBook

CIOs and CTOs

AI Tools are Making Legacy Modernization Possible

diffblue
AI for Code

Whose problem is legacy code?

Every chief tech or information officer knows that prioritizing the right investments on the path to digital transformation can provide untold benefits to the quality and agility of your business's software development.

One of your key investments is in a team of skilled developers, but even the most efficient developers get wrapped up in non-development activities, often related to the challenge of working with legacy code: the outdated, unsupported and poorly documented remnants of code written by unknown authors, with hard-to-track connections that might break something if changed. The risks inherent to modifying or replacing such code are obvious, which makes testing crucial, and time-consuming.

Code refactoring tools might help developers tackle legacy code in basic ways (e.g. renaming and safely deleting instances and classes; extracting interfaces, superclasses, and parameters from methods; renaming default namespace; removing unused references). However, these capabilities work on a very local level. They neither improve code readability, nor do they boost test coverage; therefore, the code remains very challenging to update.

Until recently, the only other way to accommodate legacy code—short of a slow, expensive, and risky total rewrite—was to maintain the status quo, make improvements gradually, and hope that the company's foundational software outlasted your tenure. But even though the developers are on the front line in the battle against legacy code, the problems it causes can be felt throughout the whole enterprise, and the power to fix it ultimately lies in the hands of the information and technology executives. Recent advancements in AI and automation have created an opportunity to repay the years of technical debt that drag down productivity and the bottom line.

Pay Back Technical Debt

For organizations where transformation might otherwise take years, strategically implementing automation to key areas of software development makes it possible to start tackling the modernization challenge. Diffblue is creating unique AI tools that quickly and accurately do the time-consuming, repetitive programming tasks that are tedious, difficult and frustrating for developers, so they can refocus their energy on what really matters: coding new features, improving the firm's software product and exceeding the expectations of the internal or external customer.

By transforming legacy software to scalable modules, companies can move from managing problems to managing outcomes.

Our Automatic Test Generation Is Like No Other

Tests are complex software programs, and writing them consumes a significant portion of a developer's time. The technology segment of test automation (TA) is crowded with frameworks and solutions for managing tests and supporting their automated execution and reporting. Now, however, there's one that can automate the writing of the unit tests themselves: Diffblue Cover intelligently creates tests within seconds, without manual coding.

The Core of Diffblue Is Our AI Engine

The core of Diffblue Cover is a mathematical reasoning and learning engine that uses a code analysis technology called bounded model checking. With

this, it is able to scan the codebase and reason about the behaviors of a program with unprecedented accuracy. Subsequently, Diffblue Cover turns these behaviors into a suite of minimal, meaningful unit tests. It generates test cases for all behaviors of the code from usual, to non-standard, and even very undesirable situations, and presents them to the technical user for consideration. This allows Diffblue Cover to make comprehensive improvements to your regression suite.

There is no use for a unit test that doesn't thoroughly examine the logic of your code, and Diffblue Cover is the only technology that understands the logic of code precisely enough to add value. Other products use inherently less-powerful technical approaches that might achieve an apparently high test coverage, but the actual tests are typically less relevant. Diffblue Cover's core technology semantically understands the behavior of a program, and will therefore deliver superior input to a programmer wanting to reason about relevant corner cases, making this the best solution for an organization that aims to create high quality unit tests for code automatically and streamline this challenging step in the software delivery lifecycle.

Other approaches to finding bugs, such as static analysis tools, suffer from lack of precision and consider non-existent executions of code, creating false positives and false negatives. Furthermore, such approaches tend to focus almost exclusively on alerts, only indicating potential problems with your code. We take a different approach: our precision technology produces real, actionable unit tests rather than just creating alarms.

As a tool, **Diffblue Cover** seamlessly integrates into a CI/CD powered workflow: the automatically generated tests are added to a test automation platform and executed on CI/CD systems (e.g. Jenkins) down the line for a huge leap in time savings and accuracy.

Earn Technical Credit

The AI at the core of Diffblue Cover makes it possible to more safely upgrade legacy code by showing the effect of certain upgrades and modifications on existing behavior. However, it also provides a level of

documentation for and an awareness of effects stemming from newly written code. With more tests, unintended changes can get identified early, and do not create problems further down the development pipeline or—worse—in production. Diffblue Cover empowers your developers to understand the impact of changes to the source code and make more informed decisions as they code, so future legacy challenges are prevented.