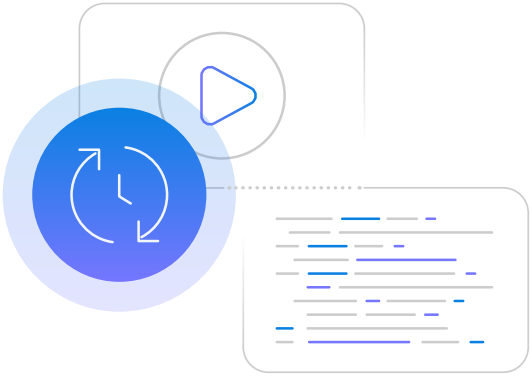
Diffblue Cover Replay: Shift More Tests Left to Accelerate Development



Use AI for Code to shift end-to-end tests left; improve software quality by unit testing data-driven code, and applications with complex internal states.

The code isn't always enough

Diffblue Cover Replay improves software quality by using existing functional tests and live runtime behavior to create new unit tests for data-driven code and applications, and applications with complex internal states.

Some code is difficult to unit test effectively because the way it behaves at runtime depends on variables which can't easily be inferred from the code itself - for example, literals might be complex strings that are only seen when loaded from an external source such as UI input, an API call or a database query.

Using functional (or end-to-end) tests instead is a common approach, but compared to unit tests they're unwieldy, slow, and identify issues too late in the development process.

Cover Replay helps to solve this problem. It instruments your application and captures data flowing to disk by recording selected executions of the code, providing the information needed to write new unit tests that validate real-world application behavior. Testing automatically becomes faster and more effective without any changes to your code.

Shift more tests left

Diffblue Cover's core unit test writing engine uses the unique record of live application behavior provided by Cover Replay to automatically write new unit tests for behavior that couldn't otherwise be covered.

No matter how complex your code is, if you can record an execution that covers a method, you can extract a test for that method.

Cover Replay lets you realize more value from Cover Core by accelerating and expanding the shift left of quality. It increases overall coverage levels, accelerates development and helps to minimize the risk of change by finding and fixing regressions sooner.

COVER REPLAY SPOTLIGHT

Increase unit test coverage

Cover Replay provides the information needed for Cover Core to automatically write new unit tests that improve overall software quality and help you meet coverage gates.

Reduce regressions and risk

Shifting more testing left means you can find and fix regressions sooner, and more easily. Some behavior captured by Cover Replay may never have been tested before - even later in the process.

Speed up software delivery

The closer you get to production, the harder and slower effective testing becomes. That's the point of shift left and the power of unit testing. Cover Replay accelerates software delivery by reducing the time spent on QA.

Go beyond the happy path

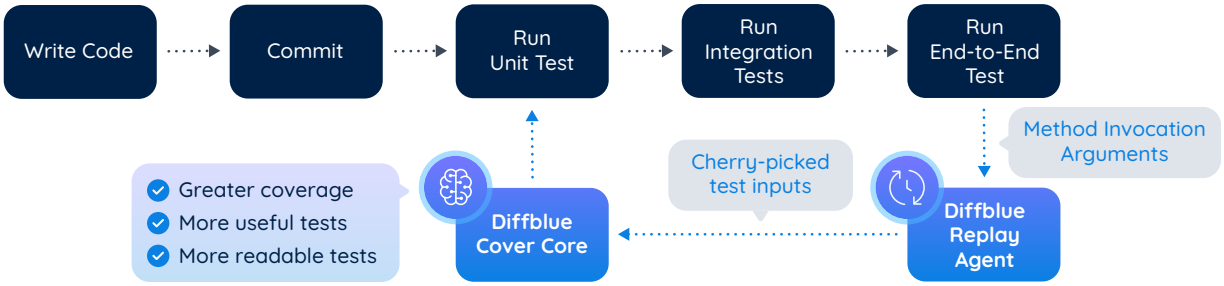
Cover Replay gives you the means to identify and unit test a range of runtime scenarios that might not otherwise be thought of, reducing the risk of change even further.

Fit your workflow

Like most features of Diffblue Cover, Cover Replay can be used locally by a developer (e.g. on their laptop) or integrated into a CI pipeline for more predictable, repeatable, low-effort unit testing.

How It Works

Cover Replay instruments your application and captures actual runtime code behavior - typically via functional, integration or end-to-end tests - which provides Cover Core with the information needed to automatically write more unit tests.



Cover Replay Example

The Netflix-Graph project uses graphs to represent a database. The method getConnection requires a NFBuildGraph object, which in turn depends on the NFGraphSpec object, which represents a database schema. A complete unit test for this complex chain of objects cannot be created by code analysis alone. Without Cover Replay, Diffblue Cover was able to write this test:

```
@Test
public void testGetConnection4() {
 // Arrange
 NFPropertySpec nfPropertySpec = mock(NFPropertySpec.class);
 when(nfPropertySpec.getName()).thenReturn("Property Name");
 when(nfPropertySpec.isSingle()).thenReturn(true);

 // Act and Assert
 assertEquals(-1, (new NFBuildGraph(new NFGraphSpec(new NF
 NodeSpec("Node spec ", nfPropertySpec))))
     .getConnection("Node spec ", 1, "Property Name"));
 verify(nfPropertySpec).isSingle();
 verify(nfPropertySpec).getName();
```

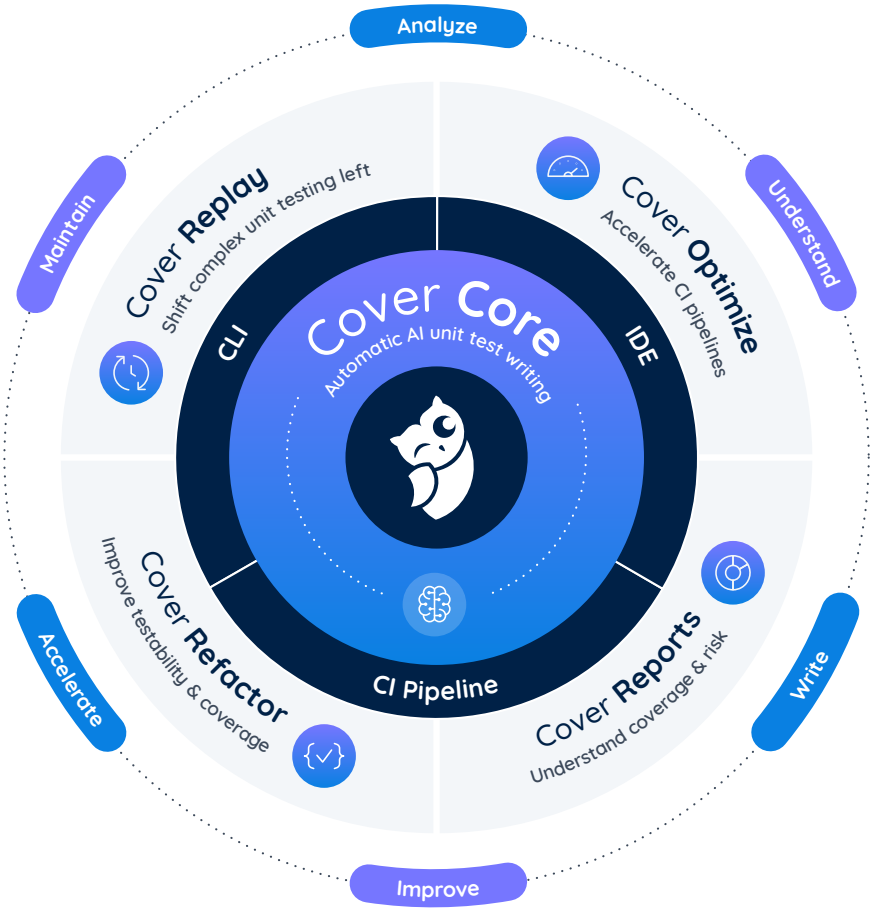
Using **Replay**, a much more comprehensive unit test can be written automatically because Diffblue Core can now see all the objects required to write the test:

```
@Test
public void testGetConnection2() {
 // Arrange
 NFCompressedGraphIntPointers nfCompressedGraphIntPointers = new NFCompressedGraphIntPointers();
 nfCompressedGraphIntPointers.addPointers("node-type-a", new int[6280]);

 NFPropertySpec nfPropertySpec = new NFPropertySpec("b-to-many-a-compact-global", "node-type-a", true, true, false);
 NFPropertySpec nfPropertySpec1 = new NFPropertySpec("b-to-many-a-hashed-global", "node-type-a", true, true, true);
 NFPropertySpec nfPropertySpec2 = new NFPropertySpec("b-to-many-a-compact-per-model", "node-type-a", false, true, false);

 NFNodeSpec nfNodeSpec = new NFNodeSpec("node-type-b", nfPropertySpec, nfPropertySpec1, nfPropertySpec2,
     new NFPropertySpec("b-to-many-a-hashed-per-model", "node-type-a", false, true, true));

 NFPropertySpec nfPropertySpec3 = new NFPropertySpec("a-to-one-b-global", "node-type-b", true, false, false);

 NFGraphSpec spec = new NFGraphSpec(nfNodeSpec, new NFNodeSpec("node-type-a", nfPropertySpec3,
     new NFPropertySpec("a-to-one-b-per-model", "node-type-b", false, false, false)));
 NFGraphModelHolder modelHolder = new NFGraphModelHolder();

 // Act and Assert
 assertEquals(0, (new NFCompressedGraph(spec, modelHolder, new SimpleByteArray(new byte[1444289]), 1444289L,
     nfCompressedGraphIntPointers)).getConnection("node-type-a", 0, "a-to-one-b-global"));
}
```

> "Citi Markets uses its deep software expertise to move faster and be more competitive. We find value in Diffblue's auto-generation of test cases. It helps drive test consistency and coverage of our software - freeing up developers to focus on delivering higher quality software, faster – and improves our developers' experience"
>
> Jonathan Lofthouse, Managing Director & Global Head of Markets Technology, Citi

The Diffblue Cover Platform

Diffblue Cover includes a range of features that let you extract more value from Java unit testing:



Diffblue Cover helps you increase business agility and accelerate transformation

Optimize the velocity and quality of Java teams; catch regressions early

Reduce software development costs and increase productivity

Untangle the complexity of refactoring legacy code

Accelerate modernization and cloud migration of core applications

ABOUT DIFFBLUE

Founded by leading computer scientists from the University of Oxford, Diffblue is changing the way code is developed. The company's flagship developer tool, Diffblue Cover, uses AI to automatically write unit tests that help Java development teams and organizations deliver better, more modern software at higher speed. Diffblue: AI for Code. Learn more at Diffblue.com or contact us at info@diffblue.com

diffblue
AI for Code