# COYOTE C++

The State of the Art in White-Box Automated Testing

CODEMIND Corporation

## Intelligent Unit Testing with COYOTE

Bill Gates once said, "we have as many testers as we have developers. And testers spend all their time testing, and developers spend half their time testing. We're more of a testing, a quality software organization than we're a software organization." This quote captures the growing significance of testing in software development and the need for efficient testing methodologies. In today's software landscape, where complexity and software size continue to increase, testing plays a critical role in ensuring quality and reliability.

Traditional testing approaches often rely heavily on manual efforts, consuming valuable time and resources. However, the rise of safety-critical software and the demand for rigorous testing have underscored the importance of automation in the testing process. Pure automated testing has become a necessity, eliminating the need for manual tasks and enabling testing teams to focus on core development activities.

In this paper, we introduce COYOTE, the advanced intelligent tool specifically designed for white-box unit testing of C/C++ programs. By leveraging the power of automation, COYOTE revolutionizes the testing landscape, offering a seamless transition to pure automated testing. This paper explores the remarkable capabilities of COYOTE and demonstrates how it can maximize testing efficiency, productivity, and code quality.

## The Growing Need for Automated Testing

Traditionally, black-box testing has been predominantly used for integration testing and system testing, accounting for approximately 30% of the total testing field. On the other hand, white-box testing, which primarily focuses on unit testing, one of the most rigorous and fundamental testing practices in the field of software development, comprises nearly 70% of the testing efforts. While black-box testing can be performed with a focus on the external behavior of the software and is generally easy to conduct, white-box testing often requires a significant investment of human resources and costs since white-box testing's deeper inspection of the internal workings necessitates a more substantial investment of human resources, specialized skills, and associated costs.

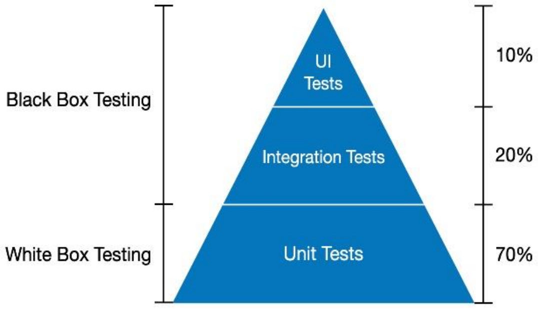


**Figure 1.** Testing landscape

Fortunately, with the introduction of automated testing, there is an opportunity to remarkably reduce human efforts. COYOTE, our advanced intelligent tool for white-box unit testing, addresses this challenge by streamlining the testing process and maximizing efficiency. By automating test case generation and test code preparation, COYOTE significantly reduces the manual effort and associated costs, making white-box unit testing more accessible and cost-effective. This is a completely different dimension of automation than what other tools have tried before.

## Harnessing the Unmatched Power of COYOTE

COYOTE is a powerful tool that revolutionizes white-box unit testing by leveraging advanced technologies such as symbolic execution and automated test case generation. In this section, we explore how COYOTE seamlessly integrates into existing workflows, simplifying the testing process and achieving unprecedented code coverage.

### Technical Overview of COYOTE

COYOTE employs symbolic execution, a innovative technique that allows it to explore different code paths and systematically generate test cases. By representing program inputs symbolically, COYOTE can analyze multiple execution paths simultaneously, uncovering potential vulnerabilities and ensuring thorough software testing.

### Manual Testing Challenges and the Need for Automation

In manual white-box testing, there are indeed significant tasks involved. It requires a deep understanding of the internal structure and workings of the source code. Test cases need to be written for each function, and test harnesses such as drivers and stubs implemented as functions must be developed.

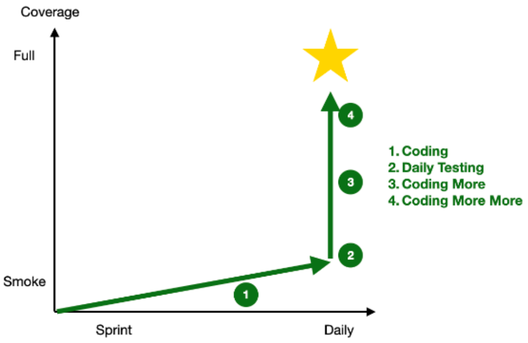


**Figure 2.** Manual approach

The tests are then executed, and based on the results, additional coding is done to cover any uncovered functions. Considering the various possible scenarios for a function, it becomes increasingly complex to contemplate all possible combinations of inputs. This process needs to be repeated regularly, often on a daily basis.

In an ideal software development context, the goal is to have automated testing that eliminates the manual effort of writing test cases for each function and developing drivers and stubs. The vision is to simplify the process by generating test cases and code directly from the source code being tested, with a simple click. This intelligent approach would ensure comprehensive coverage of the software. However, despite this ideal vision, such advanced technology is not yet a reality.

While we haven't reached the absolute ideal level of fully automated testing, significant progress has been made towards achieving it. We are excited to announce that our COYOTE technology has brought us remarkably close to this goal. With COYOTE, we have achieved the highest level of automation currently available in the industry. Unlike previous attempts that have been around for a long time, our technology has successfully achieved this unprecedented level of automation, which is a significant accomplishment.

### Simplifying Testing with COYOTE

With COYOTE, the generation of test cases and the development of test harnesses, such as driver and stub code, are seamlessly performed behind the scenes. Users simply need to review the analyzed results provided by COYOTE and make any necessary adjustments or modifications to settings or code. This level of automation empowers users to focus their efforts on reviewing and refining the testing process

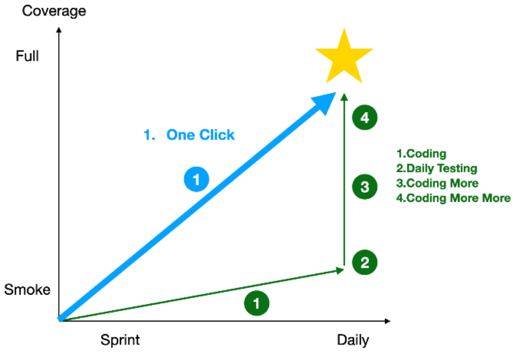


**Figure 3.** Ideal automated approach

rather than spending extensive time on manual tasks.

By leveraging COYOTE's capabilities, developers and testers can achieve a substantial increase in code coverage, reaching levels of 90% or higher. This surpasses what can typically be achieved through manual testing alone. COYOTE's automation enables thorough exploration of code paths, ensuring comprehensive testing of critical functionalities. As a result, users can have confidence in the quality and reliability of their software while optimizing their testing efforts.

### Achieving Unprecedented Code Coverage

Practically, achieving 100% code coverage through white-box testing is often not feasible due to several reasons in manual or automated mode. Unreachable code, which refers to portions of the code that are not executed under normal circumstances, may not be covered during testing. Time and resource constraints play a role as exhaustive testing efforts require considerable time and resources, especially for large and complex software systems. Non-deterministic behavior of certain components adds difficulty to precise testing, as their output or behavior cannot be reliably predicted. Additionally, the presence of external dependencies, such as libraries or APIs, introduces challenges in achieving full coverage of code interacting with these components.

COYOTE's advanced algorithms and intelligent test case generation techniques enable it to achieve exceptional code coverage. In tests conducted at the prestigious KAIST Research Center, COYOTE achieved an impressive 92% statement coverage and 87% branch coverage. These results surpass the capabilities of other existing tools in the field of automated white-box testing. With COYOTE, organizations can gain unparalleled confidence in the quality and reliability of their software.

While striving for high code coverage is important in white-box testing, it is crucial to consider these practical limitations and prioritize testing efforts to ensure thorough coverage of critical components and functionalities. Automated testing tools can help in achieving effective coverage, even if complete coverage is not feasible.

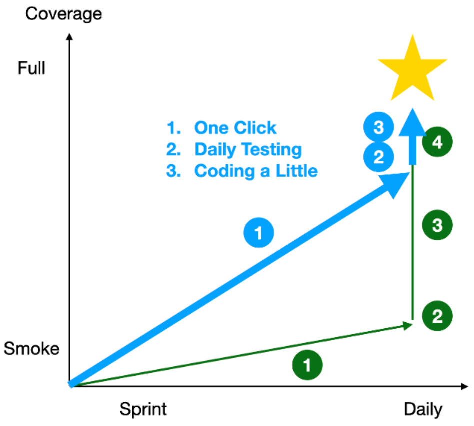
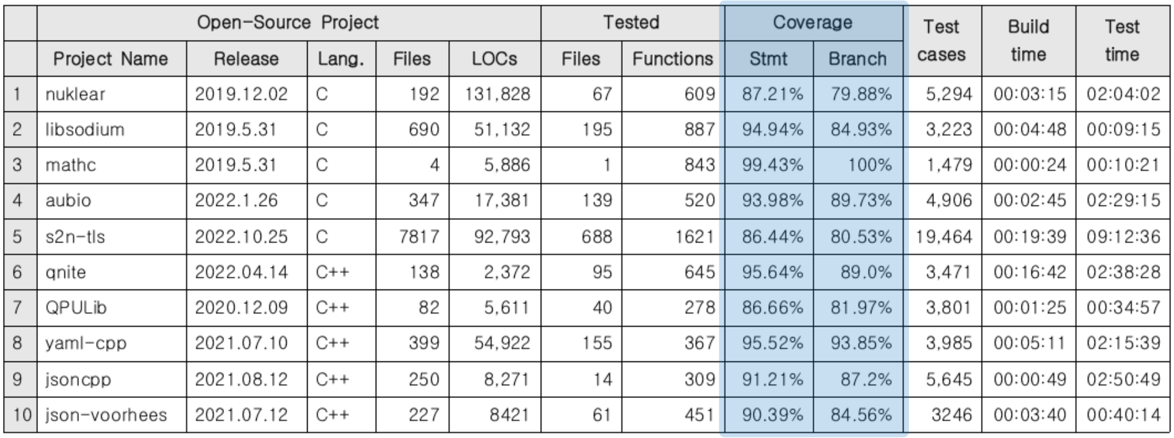


**Figure 4.** COYOTE approach



| | Open–Source Project | | | | | Tested | | Coverage | | Test cases | Build time | Test time |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Project Name | Release | Lang. | Files | LOCs | Files | Functions | Stmt | Branch | | | |
| 1 | nuklear | 2019.12.02 | C | 192 | 131,828 | 67 | 609 | 87.21% | 79.88% | 5,294 | 00:03:15 | 02:04:02 |
| 2 | libsodium | 2019.5.31 | C | 690 | 51,132 | 195 | 887 | 94.94% | 84.93% | 3,223 | 00:04:48 | 00:09:15 |
| 3 | mathc | 2019.5.31 | C | 4 | 5,886 | 1 | 843 | 99.43% | 100% | 1,479 | 00:00:24 | 00:10:21 |
| 4 | aubio | 2022.1.26 | C | 347 | 17,381 | 139 | 520 | 93.98% | 89.73% | 4,906 | 00:02:45 | 02:29:15 |
| 5 | s2n–tls | 2022.10.25 | C | 7817 | 92,793 | 688 | 1621 | 86.44% | 80.53% | 19,464 | 00:19:39 | 09:12:36 |
| 6 | qnite | 2022.04.14 | C++ | 138 | 2,372 | 95 | 645 | 95.64% | 89.0% | 3,471 | 00:16:42 | 02:38:28 |
| 7 | QPULib | 2020.12.09 | C++ | 82 | 5,611 | 40 | 278 | 86.66% | 81.97% | 3,801 | 00:01:25 | 00:34:57 |
| 8 | yaml–cpp | 2021.07.10 | C++ | 399 | 54,922 | 155 | 367 | 95.52% | 93.85% | 3,985 | 00:05:11 | 02:15:39 |
| 9 | jsoncpp | 2021.08.12 | C++ | 250 | 8,271 | 14 | 309 | 91.21% | 87.2% | 5,645 | 00:00:49 | 02:50:49 |
| 10 | json–voorhees | 2021.07.12 | C++ | 227 | 8421 | 61 | 451 | 90.39% | 84.56% | 3246 | 00:03:40 | 00:40:14 |

**Figure 5.** Table of White box testing results for 'B' automated software testing tool. Adapted from *Performance and Functionality Evaluation of White-box Software Testing Tools, Part 2*, 2023, CSRC Weblog. https://csrc.kaist.ac.kr/blog/2023/01/25/performance-and-functionality-evaluation-of-white-box-software-testing-tools-part-2/.

### Empowering Software Development

By leveraging COYOTE for white-box unit testing, developers can allocate more time and expertise to critical aspects of software development. The seamless integration of COYOTE into existing workflows enhances productivity and code quality, while the advanced symbolic execution and automated test case generation capabilities ensure thorough testing and the identification of potential issues.

## Realizing Tangible Benefits and Achieving Excellence in Software Development

COYOTE excels in white-box unit testing for C/C++ programs, streamlining the development process by automating tasks such as code comprehension, test case creation, and harness preparation. This powerful tool allows developers to focus on critical aspects of software development, improving efficiency and productivity.

One of COYOTE's notable strengths lies in its remarkable ability to enhance test coverage and minimize defects. One of the biggest automotive companies in Korea has witnessed substantial productivity gains by integrating COYOTE into their testing labs. COYOTE's intelligent testing technology has led to ten-fold improvements compared to existing tools in the market. The company has achieved exceptional test coverage ranging from 80% to 90% without coding, surpassing the limitations of traditional tools struggling to exceed 10% to 20%. COYOTE's advanced features address gaps in automated testing, ensuring comprehensive scrutiny of critical functionalities.

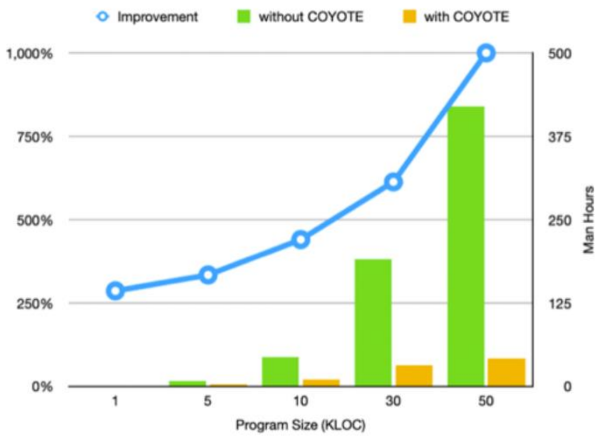


**Figure 6.** Productivity improvements

Moreover, COYOTE's safety focus makes it an ideal solution for industries adhering to rigorous standards. By employing COYOTE, organizations can ensure compliance with the demanding ISO 26262 standard for automobile development. The combination of higher code coverage and automated testing capabilities provided by COYOTE offers automotive companies a reliable and efficient means of guaranteeing software safety and compliance.

COYOTE's versatility extends beyond the automotive industry. Efforts are underway to introduce COYOTE to other sectors such as national defense, aviation, and rail. Its advanced features and capabilities make it a versatile tool that can address the testing needs of various domains, elevating productivity, and software quality across industries.

In conclusion, COYOTE revolutionizes white-box unit testing, enabling developers to achieve higher test coverage, ensure safety compliance, and enhance productivity. The practical use cases and tangible benefits of COYOTE have been witnessed in the automotive industry, with substantial improvements in test coverage and defect reduction. As COYOTE continues to make waves, expanding its presence across industries, it remains a powerful tool that empowers software development teams to deliver reliable and high-quality software. Embrace the power of COYOTE in your software development workflow to harness the benefits of automated testing and elevate your development processes to new heights.

CODEMIND
Into the Source