Regression testing is a crucial step in software development, as it helps to ensure that software is delivered continuously with quality and minimal failures after changes to the production code. During this phase, developers rely on the test results to determine whether a program has a bug resulting from recent changes. However, the presence of flaky tests can make this evaluation unreliable. Flaky tests are a type of test with an intermittent behavior that alternates between passing and failing when executed in the same codebase, without any changes. This non-deterministic behavior frustrates developers, as it makes it challenging to identify and fix the root cause of the problem. Additionally, flaky tests are difficult to debug and can cause delays in the release cycles, halting the development process.

Flaky tests can be a significant challenge in software development and identifying them is essential for ensuring the reliability and accuracy of test results. Dynamic and static approaches can be used to identify flaky tests, with each approach having its advantages and disadvantages.

Dynamic approaches involve re-executing test cases a fixed number of times, which can be expensive and error prone. It can also be difficult to determine how many executions are enough to identify flakiness accurately. Static approaches, on the other hand, do not require code re-execution and rely on machine learning methods to predict flakiness likelihood based on various features obtained from the code.

Recently, an alternative approach for predicting flaky tests has been proposed based on identifying test smells. Test smells are associated with potential design problems in the test code, and their presence may impact software quality and lead to test flakiness. The alternative approach uses a set of predictors composed only of metrics collected statically, such as the size of the test case, the number of smells in the test code, and binary features related to the presence or absence of 19 test smells. The study found that this approach had better performance than the vocabulary-based model for cross-project prediction, achieving a F-measure of 0.83 with Random Forest.