opentext™

WHITE PAPER

Script-based versus model-based functional testing

Contents

Traditional script-based and model-based testing differ in how they're applied in organizations and how they improve technical testers' efficiency.

Functional testing methods

Every organization deals with software bugs and logic issues. So, naturally, every organization has tools and methods to test their development and production codebases. Most of these enterprises follow cost-efficient and time-tested approaches that optimize quality and increase accuracy.

Among these testing methods are traditional script-based and model-based functional testing.

Script-based testing requires technical testers (automation engineers or developers) to follow a step-by-step walkthrough to test a given test case or functionality. Testers need to define test cases in advance with procedures describing the expected output. With scripted testing, a tester knows what to expect from each input and follows instructions to test and pass each feature.

Model-based testing (MBT) involves test cases generated from production models of system requirements. MBT allows technical testers to carry out tests independent of algorithmic design and development. Additionally, MBT helps automate business and software requirements for analysts and quality assurance (QA) professionals beyond traditional stochastic and heuristic methods.

But which of these testing paradigms is best? In this paper, we will:

- Explore the strengths and challenges of each.
- Show the value of a hybrid testing approach.
- Look at how OpenText™ Functional Testing can help you reach this hybrid approach.

With script-based testing, you can:

- Easily manage test coverage.
- Repeat tests several times with predefined test scripts.
- Get detailed results of each executed test case.
- Effectively test software or system requirements in sequence.

Script-based versus model-based

Traditional script-based and model-based testing differ in how they're applied in organizations and how they improve technical testers' efficiency.

Since scripted testing is a conventional approach, you need to define test cases in advance. Additionally, all requirements and specifications must be completed before testing. Script-based testing relies heavily on the knowledge and expertise of testers since they need to know what to expect at the end of the testing cycle.

On the other hand, model-based testing is a lightweight, automated method for testing hardware and software requirements. With MBT, you validate the system against predictions made by a model.

MBT is more part of the software development process than traditional scripting testing. It requires a clear goal and equitable expenditure to make an accurate and maintainable testing case. MBT allows your teams to focus on how to construct testable applications and design models based on real-world relevance from the user's point of view. Plus, it minimizes script development and maintenance.

However, there are challenges with MBT, including limited control over testing, implementation complexities, and lack of complete insight.

Because forecast models do most of the evaluations, it's difficult to control the behavior and expected output. Why? Predictions are based on system requirements independent of the algorithms used in testing.

MBT also uses models to evaluate various system requirements. This process is complex when an organization does not have scalable architecture and the proper tooling infrastructure for test maintenance and coverage.

Coming up with clear insights from MBT test coverage, fixing bugs, and building models for the application under test can be challenging for complex models running on unreliable architectures. You'll need to prepare for these challenges to get the most out of MBT.

MBT and script-based testing allow you to automate various test cases across your system architecture, easing test coverage management. With both testing paradigms, it's easy to get detailed test results, giving insight into which inputs and test cases you can improve and how.

However, both of these approaches have blind spots. Let's go through some of the advantages and disadvantages.

MBT offers some unique advantages. These can include:

- Improved test coverage.
- Increased accuracy and visibility.
- More collaboration among your teams.

Scripted testing

With script-based testing, you can:

- Easily manage test coverage.
- Repeat tests several times with predefined test scripts.
- Get detailed results of each executed test case.
- Effectively test software or system requirements in sequence.
- On the other hand, scripted testing isn't ideal when:
- Dealing with changing requirements.
- Working within time constraints—this method provides feedback slowly.
- Trying to catch bugs and issues early in the testing cycle.

Model-based testing

MBT offers some unique advantages. These can include:

- Improved test coverage.
- Increased accuracy and visibility.
- More collaboration among your teams.

The disadvantages to MBT are:

- Steep learning curve for technical testers.
- Generated models and tests sometimes miss bugs.
- Scaling issues with complex models.

Hybrid testing

MBT and script-based testing both have tradeoffs.

Scripted testing focuses on achieving the result but isn't concerned about the technical tester's satisfaction. Additionally, scripted testing doesn't cover most of the real-world problems users might face.

But MBT isn't a silver bullet. While it can improve application integrity, it's challenging to implement.

Why not do both? A hybrid approach is the best way to maintain application integrity. Hybrid testing combines the power of traditional scripted testing with automated model-based functional testing. Both methods ensure that your system requirements are tested against all bugs and any hidden issues.

OpenText Functional Testing is perfect for hybrid testing. It boasts scalable, intelligent, automated tests that cover your enterprise's web, mobile, desktop, mainframe, and composite applications.

Resources

OpenText CEO Mark Barrenechea's blog >

Next steps

Testing is an integral part of the software development process. It requires sophisticated tools to ensure that your system requirements are tested against all possible cases.

Remember, using either scripted or model-based testing will only get you so far. You need a hybrid approach for effective all-around testing.

OpenText Functional Testing provides you with a hybrid model that captures the best parts of both script-based and model-based testing—without leaving blind spots that can impact your application's integrity.

Testing can be tedious, complex, and difficult to carry out when it involves many systems in an organization. To reduce these complexities, visit our OpenText Functional Testing page to learn how to better plan your business testing strategy.

Learn more.