endava

MODEL-BASED TEST DESIGN AND AUTOMATION

GREGORY SOLOVEY
TEST ARCHITECT

MARK FIRTH
GLOBAL HEAD OF TESTING SERVICES

CONTENTS

01

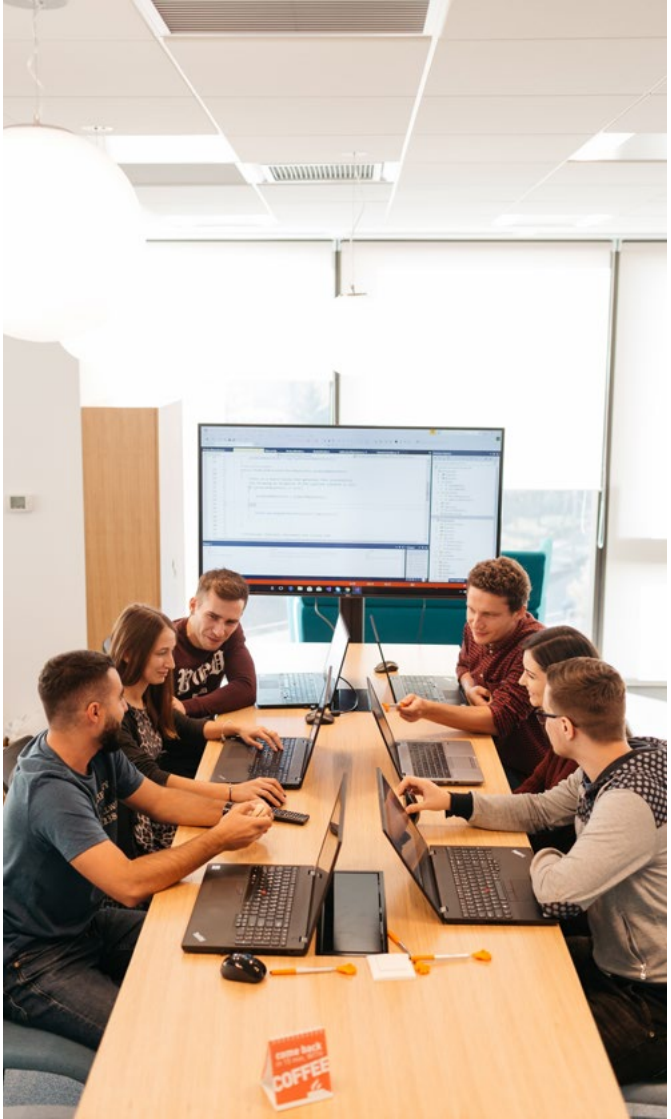
INTRODUCTION

There are a number of reasons why testing is an imperative part of product design and delivery. While some may be obvious, for example it ensures customer or user satisfaction which is increasingly important and the focus shifts to deliver excellent experiences to satisfy customer expectations, it is important not to overlook the equally important business benefits of saving money, ensuring security and delivering the promised value through product quality.

The fact is that the business will expect your products to be perfect, or as close to perfect as possible. This cannot be achieved easily, but test automation offers some solutions to a number of testing challenges, namely, how to guarantee test completeness and how to maximise return on investment (ROI) from test projects.

Endava is routinely selected as a partner of choice to assure success in test automation and deliver robust solutions that solve client's business challenges and unlock new business value. We base our approach on a collaborative conversation with our clients about their goals and challenges and leverage our expertise to identify and deliver a solution.

We wanted to share one of our responses to testing challenges we see on a regular basis, so that it might help you to improve your test automation.



02

IT HURTS WHEN I PRESS HERE

The perception of pain is subjective, and for testing, it is dependent on differences in organisational culture, automation skills, and ways of working.

A challenge described by a client does not always point to the true source of the problem. The same symptom can have many different causes.

02. "IT HURTS WHEN I PRESS HERE"

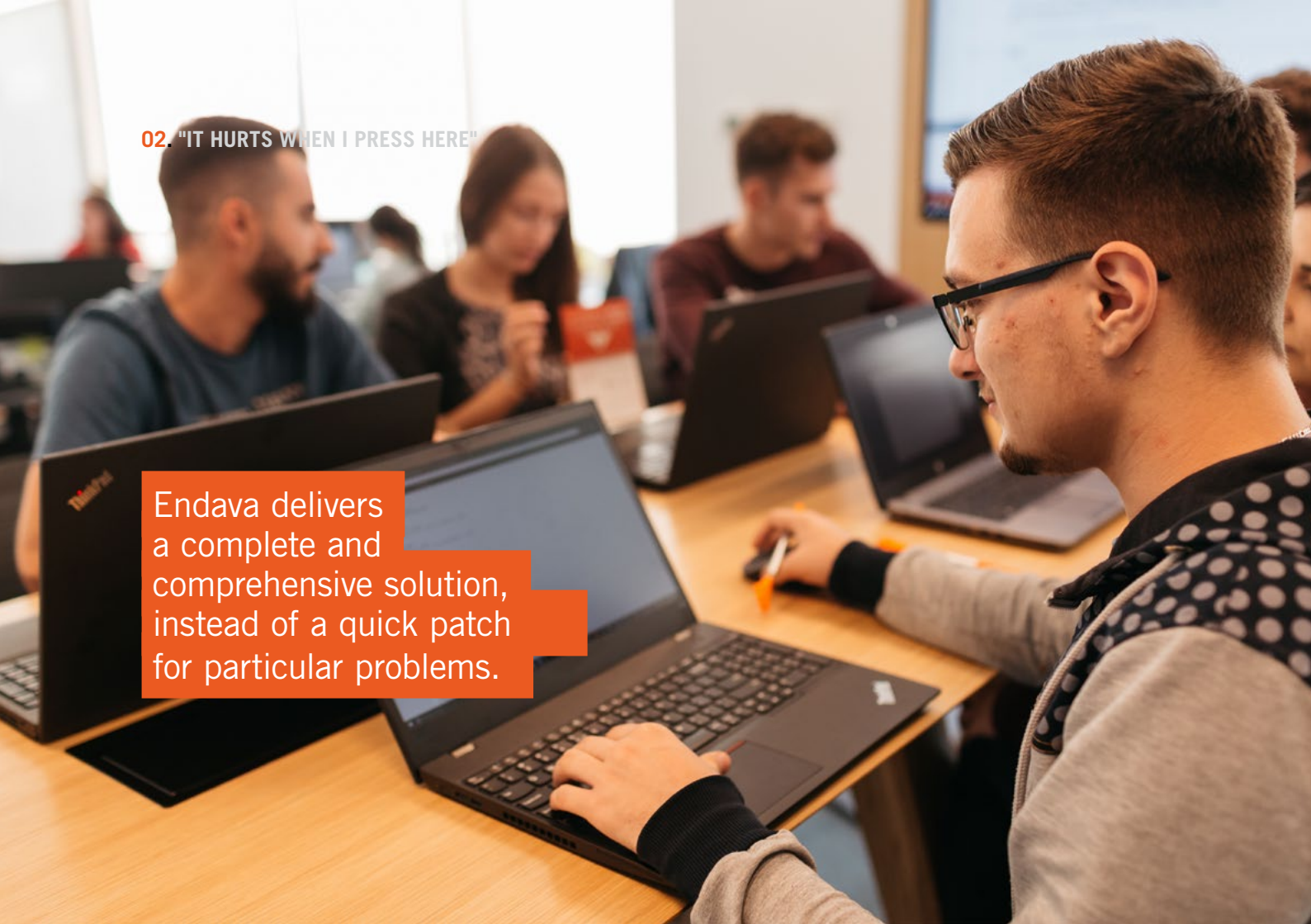
For example, the source of a bug found in production could be the result of a multitude of different scenarios, including:

- The testing of a new feature was not complete;
- The regression tests were manual and were not run completely for the latest build;
- Testing was not executed in all production environments (e.g. browsers and mobile devices);
- Performance testing was run on unrepresentative data or on a non-production-like environment.

Equally, the source of missed milestones or slow test execution could be traced to:

- Unrealistic expectations and planning;
- Poor testing practices;
- A shortage of skilled personnel.

The approach to identify and fix isolated problems does not guarantee the absence of additional, masked problems. Just like in medicine, instead of just treating the symptoms, the best approach is to bring testing to a healthy state and then continually monitor its effectiveness to drive additional improvements.


Endava delivers
a complete and
comprehensive solution,
instead of a quick patch
for particular problems.

03

THAT'S A RELIEF

When implemented effectively, test automation has the potential to yield significant ROI due to the replacement of time consuming, and sometimes unreliable and non-repeatable, manual testing. Currently, 85% of companies report that they are using test automation (**State_of_testing_report**), but at the same time, the average automation coverage is only 20% (**broken-promise-test-automation**). Test automation often fails or does not provide the expected ROI due to maintenance issues, stemming from a lack of appropriate skills and experience.

The first paragraph in Leo Tolstoy's "Anna Karenina" states: "Happy families are all alike; every unhappy family is unhappy in its own way." **To paraphrase it: Successful test projects are all alike; but every unsuccessful test project fails in its own unique way.**

> Successful test projects are all alike; every unsuccessful test project fails in its own unique way.

Endava uses a **model-based test design and automation** approach - a comprehensive solution to build and automate tests in an efficient and maintainable fashion. This approach enables automation in parallel with development and makes tests traceable to the system architecture and requirements.

This model-based test design and automation flow is presented in the picture below and the descriptions of each view follow >

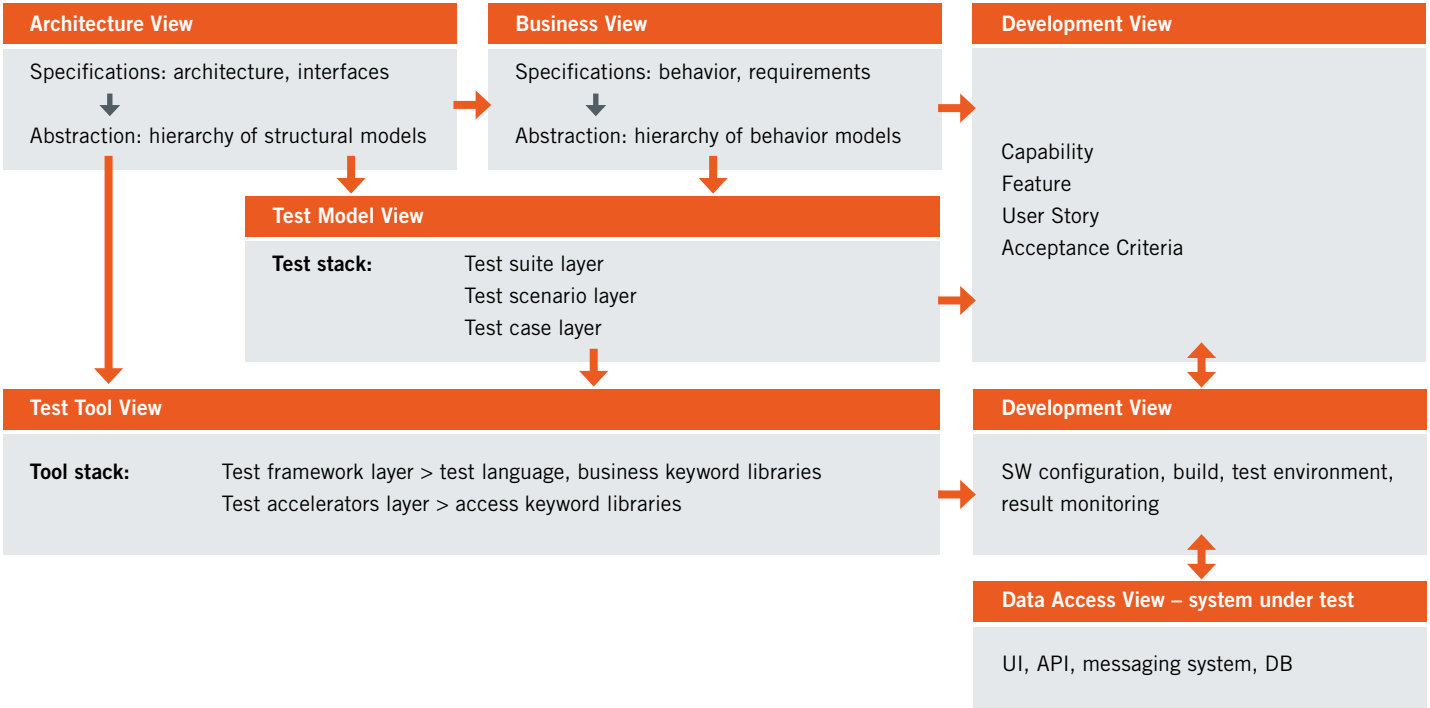


Figure 1 – *Endava Model-based test design and automation flow results*

03. "THAT'S A RELIEF!"

When creating an automated test suite, rather than simply translating manual tests into automated tests on a one-to-one basis, it is more efficient to step back and define the desired application behaviour and the existing system architecture and use these to drive how the test automation is structured.

The technical capabilities the test automation framework must exhibit are driven by the **Architecture** of the system to be tested (**Architecture View**). For example, if we want to execute test flows through a web GUI, API and database, we need a framework which is capable of interacting with those interfaces and supporting tests which interact with multiple types of entry and validation points.

The technologies behind the interfaces help us to identify the tools which will be included in the test automation framework e.g. Selenium for UI testing, Java or .NET libraries for API and database testing.

The Business View is a system presentation which describes the behaviour, responsibilities and data structures for each element of the Architecture View. On an abstract level it is organised as a hierarchy of behaviour models.

The Test Model View presents the hierarchy of test artefacts that are built to verify the business model's implementation. **Model-based test design** uses proven test design methodologies (**test design methods**), which illustrate how to build the minimum number of tests, to maximise test coverage and to improve test completeness and traceability. This approach assumes that each architecture element/ requirement is presented as a formal model (an expression, an algorithm, a state machine, etc.), and then known methods to build test cases are used to make the test design a straightforward, routine process that enables us to build complex testware within days.

03. "THAT'S A RELIEF!"

Model-based test design presents the tests as a set of abstract layers. This enables the separation of the business logic from the implementation details and, as a result, makes the automation process more efficient, decreases maintenance costs, increases test reliability and the reusability of test objects. The test stack includes the following layers:

- Test suite layer - splits tests according to the system's architectural structure and/or functional behaviour.
- Test scenario layer - presents the behaviour (functionality) of an architectural element, an application, a service, a module, etc.
- Test case layer - defines the test case format: set-up, execution, result comparison, and tear down sections.

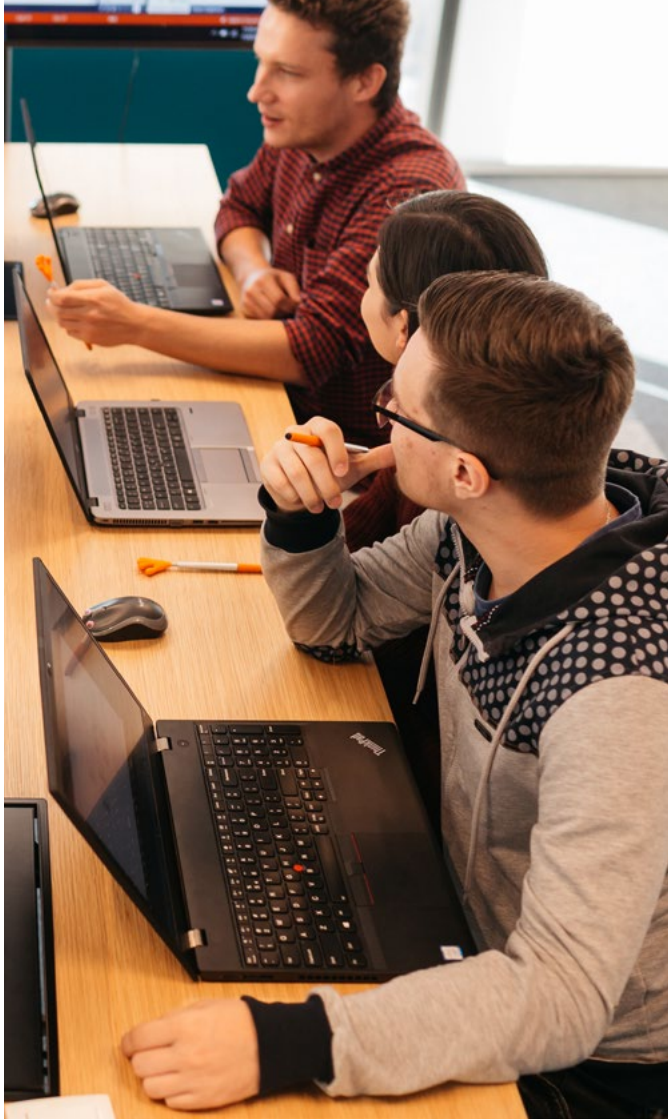
The Test Tool View presents test frameworks and underlying test automation tools. Test Frameworks (such as Robot Framework, Cucumber, SpecFlow, Junit, NUnit, TestNG) offer a "language" to represent the test hierarchy and libraries of keywords to communicate with test tools. Test tools (such as Selenium, JMeter, SoapUI) and Java or .NET libraries support the communication with a variety of interfaces (Web, GUI, API, messaging, DB) to send stimuli and capture results.

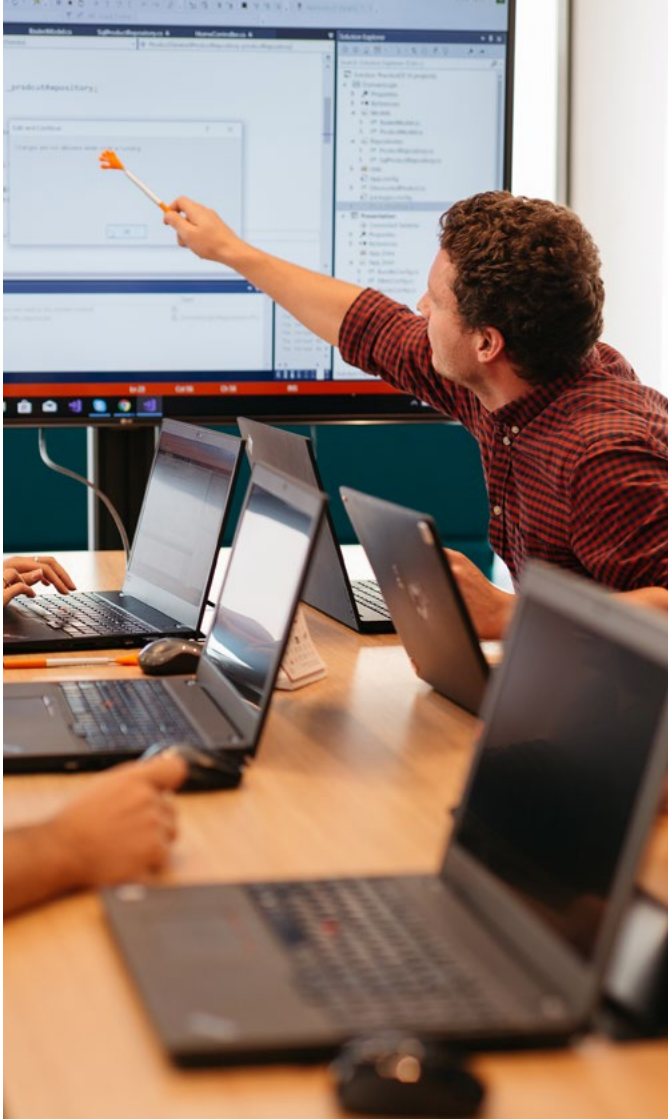
The test hierarchy from the Test Model View will be presented by Framework language. Thus, the same test presented at the business level can be performed on different browsers and devices using different access keyword libraries. In addition, various testing tools can execute the same test presented at the business level.

03. "THAT'S A RELIEF!"

The Development View represents the transformation of the Architecture/Business views into development artefacts, such as capabilities, features, epics, user stories and their acceptance criteria for Agile models, or specifications and designs in a waterfall model.

The release/feature planning phase typically starts as soon as high-level Architecture and Requirements documents are created (**Core Practices for Agile/Lean Documentation**). Those are decomposed and refined into user-stories and their respective acceptance criteria. In many traditional Agile implementations, tests are only created at the user-story level. This can lead to a "fragmented" test pack, where the big-picture view of the system is lost. *The model-based approach delivers high-level tests for each Architecture and Requirements document. Tests are refined along with the decomposition of architecture and requirements.*





03. "THAT'S A RELIEF!"

The Release View represents the result of the transformation of the development work items (from the product backlog) into release artefacts.

The code related to a user-story is moved through the CI/CD pipeline – included in a particular build, tested in particular environments, deployed to a particular release.

The Data Access View presents the variety of the system interfaces.

03. "THAT'S A RELIEF!"

Model-based test design and automation provide the following benefits:

TEST AUTOMATION CREATION:

the test coding process is replaced by a test "declaration" process. In the majority of cases, a tester isn't required to use a programming language (Java, Python, C++, etc.), but instead "declares" a test with a set of predefined business and access keywords, using the test framework language. New objects or functionality can be incorporated by simply adding new "action words." This approach allows all testers to participate in test automation by reusing "actions" from existing libraries**.**

MAINTENANCE AND EXTENSION:

business and access "action words" are defined in libraries, and therefore, **any single code change leads to a single testware update in a library**. This makes test maintenance straightforward.

DEBUGGING:

the structured test presentation **makes debugging simple**, since the test and log file formats are common. Each test case is traced back to the respective business scenario (a requirement) and structural elements. It is clear from the log file structure what the stimuli, system response and expected results were, and why a test case failed.

TEST EXECUTION:

the fact that the test structure is mapped to the system structure allows testers to selectively execute test suites according to the modules that were touched in the latest build. Such an approach can dramatically **decrease the time to test, and as a consequence, the delivery time.**

ADDITIONAL BENEFITS:

There are also additional benefits, such as improved testability, end-to-end traceability and the ability to focus test execution coverage.

04

INVEST IN YOUR HEALTH

Test quality is about faster product delivery, minimised defect escapes, increased quality, reduced test creation effort, improved maintainability, good transparency and fast feedback. While improving test automation may require considerable upfront investment, this can be easily justified by the ROI.

The benefits of model-based test design and automation solutions include significant cost savings by replacing manual effort, the reduction in maintenance costs due to the proper organisation of the testware and production bug density reduction, that can yield between 40%-80% in savings (**reduction in defect density**).

IBM estimates that a bug costs $100 to fix during the "gathering requirements" phase, $1,500 in the QA testing phase, and $10,000 in production (**software-bug-cost**).

The investment in **Endava test automation solutions** will yield visible results in just a few weeks and the full investment will be rapidly recouped.

Note:
This paper focuses on just one approach to implementing comprehensive testing. This is not the only test-related solution we use, other solutions include continuous improvement, shift left, architecture testability principles, quality and progress monitoring, test data generation, test as a service, built-in self-test, etc.

93%	reduction in test execution time
95%	cost reduction in generating environments
40%	performance improvement
98%	requests served in <0.4 seconds
70%	test coverage increase
78	different performance scenarios tested
4	years in production with no failures
8	performance tuning improvements found
20	commits per day
80%	performance gain
66%	reduction of team size
30	mins to run smoke test
100%	license cost reduction
8	automated scenarios / day

Figure 2 – *Endava Model-based test design and automation benchmarks*

05

ABOUT ENDAVA

Endava is reimagining the relationship between people and technology.

We accelerate our clients' ability to take advantage of new business models and market opportunities by ideating and delivering dynamic platforms and intelligent digital experiences that fuel the rapid, ongoing transformation of their business.

By leveraging next-generation technologies, our agile, multi-disciplinary teams provide a combination of Product & Technology Strategies, Intelligent Experiences, and World Class Engineering to help our clients become more engaging, responsive, and efficient.

CONTACT US

Want to know more?
Email us.

AUTHORS
Gregory Solovey - Test Architect
Mark Firth - Global Head of Testing Services