УДК 004.415.53

# РЕШЕНИЕ ПРОБЛЕМЫ НЕСТАБИЛЬНОСТИ АВТОМАТИЗИРОВАННЫХ ТЕСТОВ ПОЛЬЗОВАТЕЛЬСКОГО ИНТЕРФЕЙСА WEB-ПРИЛОЖЕНИЙ

**Кириллов С.С.**

*«КонсультантПлюс», Москва, e-mail: sskmbox@gmail.com*

Одной из основных проблем в области автоматизации тестирования пользовательского интерфейса web-приложений является общая нестабильность браузерных тестов. Данная особенность значительно затрудняет оценку работоспособности программного обеспечения в процессе его разработки. Кроме того, недетерминированное поведение тестов снижает доверие к ним со стороны различных команд, включая команды разработки и тестирования. В качестве объекта исследования выступил тестовый проект, построенный на базе инструмента для взаимодействия с браузером – Selenium WebDriver в связке с языком программирования Java. Как правило, с ростом количества тестов наблюдается увеличение числа проблемных тестов. В этом случае команда тестирования вынуждена тратить больше усилий на поддержку работоспособности проекта, включая время на выявление причин падения и исправление. Целью исследования является снижение доли нестабильных тестов в общем объеме тестовых сценариев. Научная новизна представленной работы состоит в том, что найден алгоритм снижения доли недетерминированных тестов, эффективность которого основана на очередности внедрения каждого этапа. Данная методика предполагает мониторинг доли проблемных тестов на всех этапах внедрения, что позволяет оценить вклад каждого из описанных подходов. Кроме того, автором предложено сочетание собственных решений, в том числе для подготовки и управления тестовым окружением, вкупе с применением программного обеспечения с открытым исходным кодом, например Selenide. В рамках данной статьи проведен анализ причин нестабильности автоматизированных тестов пользовательского интерфейса web-приложений. Рассмотрены способы борьбы с недетерминированным поведением в тестах. На основе полученных данных предложена комплексная методика, направленная на снижение доли нестабильных тестов в общем наборе сценариев. Реализация данной методики на практике позволила в несколько раз сократить долю проблемных тестов в проекте, что дало импульс для дальнейших преобразований в области автоматизации тестирования. Полученные в ходе проведенного исследования методические рекомендации позволяют лицам, принимающим решения в области внедрения систем тестирования программного обеспечения, более взвешенно подходить к построению системы автоматизации тестирования пользовательского интерфейса.

**Ключевые слова: автоматизированное тестирование, нестабильные тесты, пользовательский интерфейс, тестирование программного обеспечения, java, selenide, selenium webdriver, web**

# SOLVING A PROBLEM OF UI-BASED FLAKY TESTS ON WEB APPLICATIONS

**Kirillov S.S.**

*«ConsultantPlus», Moscow, e-mail: sskmbox@gmail.com*

One of the main problems in the field of automated web application UI testing is the general instability of browser tests. This feature greatly complicates the assessment of software performance during its development. Besides, the nondeterministic behavior of tests decreases their credibility among various teams, including development and testing teams. A test project based on the Selenium WebDriver browser interaction tool coupled with the Java programming language was used as the object of the study. As a rule, as the number of tests grows, the number of problematic tests increases. In this case, the testing team has to spend more effort on maintaining the project's operability, including the time to identify the causes of crashes and fix them. The objective of the study is to reduce the share of flaky tests in the total volume of test scenarios. This paper analyzes the causes of instability in automated web application UI testing. Ways to eliminate nondeterministic test behavior are considered. Based on the obtained data, a complex methodology aimed at reducing the share of unstable tests in the overall set of scenarios is proposed. The novelty of the research is in the algorithm for reducing the proportion of non-deterministic tests, the effectiveness of which is based on the order of implementation of each stage. This methodology includes tracking the proportion of flaky tests at each stage of implementation. It makes it possible to evaluate the contribution of each solution of the described approaches. In addition, the author proposed a combination of their own solutions including preparing and managing a test environment, coupled with the use of open-source software, like Selenide. Implementation of this technique allowed to reduce the share of problematic tests in a project by several times, which gave an impetus for further transformations in the field of test automation. The methodological recommendations obtained in the course of this study allow decision makers in the field of software testing systems implementation to take a more balanced approach to building a UI automated testing system.

**Keywords: automated testing, flaky tests, user interface, software testing, java, selenide, selenium webdriver, web**

Согласно классическому подходу к построению эффективной стратегии автоматизации тестовое покрытие можно представить в виде пирамиды. В основании пирамиды находятся модульные тесты, которые занимают наибольшую долю в общем объеме тестовых сценариев. Далее следуют тесты сервисов, и на вершине находятся функциональные тесты пользовательского интерфейса [1, 2]. По сравнению с модульными и сервисными, тесты на интерфейс требуют от специалистов значительных усилий на их разработку и сопровождение, а запуск таких тестов занимает больше времени.

В первую очередь сложность в сопровождении таких тестов обусловлена их общей нестабильностью. К нестабильным тестам относят тесты, результаты повторного запуска которых отличаются друг от друга, несмотря на отсутствие изменений в коде тестируемого приложения. В зарубежных источниках такие тесты называют «flaky» [3–6].

С проблемой нестабильности тестов сталкивается большинство разработчиков программного обеспечения. Так, например, в 2017 г. компания Google опубликовала данные, согласно которым общий объем ненадежных тестов внутри компании составлял около 2%. Однако для тестов пользовательского интерфейса доля таких тестов существенно выше. Так, например, для тестов Selenium WebDriver в связке с языком программирования Java объем нестабильных тестов составлял около 10%, а в связке с Python – около 20%. Для мобильных устройств на базе Android доля недетерминированных тестов еще выше – 25% [7].

Это указывает на масштаб проблемы в области автоматизации тестирования с использованием инструментов на основе Selenium WebDriver. Наличие большого количества нестабильных тестов в проекте снижает уровень доверия к ним со стороны команд тестирования, разработки и управления. Именно в решении данной проблемы и состоит задача исследования.

*Причины нестабильности тестов*

Для решения поставленной задачи важно понимать причины недетерминированного поведения в тестах. За последние годы было проведено несколько исследований, описывающих природу такого явления. В таблице представлен перечень наиболее частых причин сбоев автоматизированных тестов на примере программного обеспечения с открытым исходным кодом [8–10].

Согласно результатам исследования основная причина нестабильности большинства автоматизированных тестов – ожидание наступления событий [3, 5]. Большинство современных web-приложений имеют динамический контент, что затрудняет взаимодействие с элементами web-страниц в тестах. Часть объектов может загрузиться раньше, другая часть позже, что приводит к частым падениям в тестах.

Распространенными причинами недетерминированного поведения в тестах являются ошибки в тестовой логике. В основном это проявляется в виде зависимости тестов от очередности их запуска или состояния гонки при использовании многопоточных операций. Локализация таких ошибок требует больших усилий от команды проекта, так как необходимо определить, в какой момент происходит сбой.

Основные причины нестабильности автоматизированных тестов программного обеспечения с открытым исходным кодом

| Категории | Причины нестабильности в тестах |
|---|---|
| Ожидание | – Рендеринг ресурсов<br>– Проблемы анимации на странице<br>– Загрузка сетевых ресурсов |
| Тестовая логика | – Порядок запуска тестов<br>– Параллельные операции<br>– Ошибки системного времени |
| Окружение | – Ошибки, связанные с платформой<br>– Сетевые ошибки<br>– Утечки ресурсов |
| Инструменты тестирования | – Сбои при обращении к локаторам<br>– Проблемы инструмента запуска тестов |

Также можно выделить ошибки, связанные с окружением (версия браузера, драйвера и т.д.) и инструментами тестирования, например Selenium WebDriver.

*Способы борьбы с недетерминированным поведением в тестах*

Существуют различные методы борьбы с недетерминированным поведением в тестах программного обеспечения. Часть из них связана с тестируемым приложением, другая часть – непосредственно с тестами или окружением. Наиболее популярные методы [3, 4, 9]:

*1. Преобразование ожиданий в тестах*

Многие разработчики тестов, столкнувшись с проблемой ожидания, предпочитают останавливать процесс выполнения фрагмента кода на фиксированное время, после чего программа продолжает свою работу. Данный подход существенно замедляет тесты и не гарантирует их работоспособность. Более эффективным способом является ожидание конкретного события на странице в течение определенного времени.

*2. Изоляция тестов*

Принцип изолированности тестов лежит в основе большинства успешных проектов в области автоматизации тестирования. Независимые тесты не оказывают влияния друг на друга, что, с одной стороны, сокращает долю недетерминированных тестов в проекте, а с другой, позволяет легче идентифицировать проблему в случае сбоя. Кроме того, благодаря изоляции, тесты

можно запускать параллельно в несколько потоков, тем самым ускоряя время их прохождения. Сложность реализации данного подхода состоит в том, что необходимо решить ряд вопросов, связанных с подготовкой окружения и тестовых данных, а также восстановлением исходного состояния системы перед тестами.

*3. Карантин*

Карантин подразумевает перенос недетерминированных сценариев в отдельный набор тестов. С одной стороны, это повышает уровень доверия к основному набору тестов, а с другой стороны, дает возможность более детально изучить проблему. Недостатком карантина является то, что про тесты в нем часто забывают. Выходом из данной ситуации может быть установка лимита по времени нахождения теста на карантине.

### *Методика борьбы с нестабильными тестами на проекте*

В рамках данного исследования рассмотрен проект, состоящий из более чем 1 000 автоматизированных тестов пользовательского интерфейса web-приложения. Проект построен на базе инструмента для взаимодействия с браузером – Selenium WebDriver в связке с языком программирования Java. Тестовый стенд поддерживает одновременный запуск последних версий браузеров Chrome, Firefox и Edge с помощью распределенной системы серверов Selenium Grid.

На начальном этапе развития проекта тесты запускались в один поток на последних версиях браузеров. По мере увеличения количества тестовых сценариев начала проявляться проблема недетерминированного поведения, которая усугублялась внедрением многопоточности.

С целью отслеживания состояние тестируемого продукта, а также определения доли нестабильных тестов в проекте, результаты каждого запуска записываются в базу данных (БД). Помимо основных показателей, в БД передается информация о каждом тесте, включая информацию о текущем статусе теста и количестве успешных запусков. На основе полученных данных производится оценка доли ненадежных тестов в общем объеме сценариев. Так, если за последние 20 запусков тест хотя бы раз упал с ложноотрицательным результатом, то он помечается как нестабильный. В противном случае тест исключается из списка. Диапазон отслеживания может варьироваться в зависимости от частоты запуска тестов.

На некоторых этапах исследования доля недетерминированных тестов в проекте достигала пиковых значений в объеме 15% от общего количества тестов, что серьезно затрудняло оценку состояния тестируемого приложения. В числе основных причин падения тестов можно выделить:

− Ожидания событий.
− Влияние тестов друг на друга.
− Проблемы окружения.

В настоящее время область преодоления недетерминированного поведения в тестах недостаточно изучена. Кроме того, стек технологий каждой компании имеет свою специфику, что отражается на способах борьбы с нестабильными тестами. Поэтому эффективность каждого конкретного способа борьбы с нестабильными тестами может отличаться в различных проектах.

Проанализировав существующие решения, а также используя собственный опыт и наработки, была разработана методика для поэтапного решения проблемы недетерминированного поведения автоматизированных тестов пользовательского интерфейса. Основные этапы реализации методики представлены на рис. 1.

Рассмотрим данные этапы подробнее.

### *Внедрение стандарта*

На первом этапе был внедрен стандарт, в котором описаны основные требования к тестам на проекте. Стандарт предполагает соблюдение ряда принципов, среди которых:

*1. Атомарность*

Существует корреляция между объемом тестового метода и вероятностью появления недетерминированного поведения [7]. Наличие большого количества проверок и переусложненная логика в тестах замедляют ход их выполнения. Поэтому было введено ограничение – не более 30 строк кода для одного теста.


*Рис. 1. Методика борьбы с нестабильными тестами пользовательского интерфейса*

Таким образом, каждый тест отвечает за работоспособность небольшого сегмента функциональности и проходит максимально быстро. В случае появления ошибки другие тесты смогут продолжить свою работу и дадут более полное представление о состоянии приложения.

*2. Соответствие правилам оформления*

Выработка общих подходов и правил к написанию тестового кода позволяет снизить влияние энтропии на проект, а также ускоряет процесс доработки тестов в случае изменений в коде тестируемого приложения. Данный принцип был сформулирован в виде подробной инструкции о том, как должен быть оформлен код тестов. За основу был взят путеводитель от компании Google [11].

Внедрение стандарта не дает быстрых результатов, однако по мере добавления новых тестов в проект, а также преобразования старых количество нестабильных тестов постепенно снижается. В этой связи было запланировано дополнительное время на внесение правок в существующие тесты.

*Преобразование механизма ожиданий в тестах*

Далее команда тестирования сконцентрировалась на решении проблемы ожидания событий в браузере. Одним из распространенных методов ожидания события в тесте является принудительная остановка потока на заданное время. Данный подход имеет ряд недостатков. С одной стороны, это приводит к увеличению продолжительности теста, а с другой, не дает гарантии, что ожидаемое событие не произойдет позже и тест все равно упадет. Selenium WebDriver имеет в своем арсенале более продвинутые механизмы явных и неявных ожиданий. Однако эти способы либо недостаточно эффективны, либо требуют больше строк программного кода в тестах.

Столкнувшись с особенностями ожиданий в Selenium WebDriver, было принято решение внедрить в существующий тестовый проект инструмент автоматизации тестов – Selenide. Данный инструмент представляет собой высокоуровневую надстройку над Selenium WebDriver и предлагает пользователям ряд полезных функций. Одной из таких функций является механизм «умных ожиданий», когда любое действие с элементом на странице в браузере подразумевает встроенный таймаут (до 4 с по умолчанию). Данное решение позволило свести к минимуму количество недетерминированных тестов, связанных с ожиданием событий.

*Подготовка тестовой среды*

В зависимости от очередности запуска автоматизированных тестов в многопоточной среде, результаты могли отличаться. С данной проблемой сталкиваются многие специалисты [6], поэтому одной из приоритетных задач стала максимальная изоляция тестов друг от друга. В качестве решения были использованы следующие подходы:

– Наполнение базы данных тестовыми данными.

– Создание пула пользователей.

– Генерация пользователей с особым набором характеристик.

– Очистка кэша web-приложения и браузера.

Помимо этого, для стабильной работы автоматизированных тестов пользовательского интерфейса необходима совместимость версий браузера и драйвера (Selenium WebDriver). С этой целью в проект была интегрирована библиотека WebDriverManager [12], с помощью которой можно загружать любую версию драйвера для наиболее популярных браузеров. Также, с помощью библиотеки, удалось решить проблему обновления драйвера на удаленных узлах Jenkins.

*Замена внешних сервисов*

Наличие большого количества внешних зависимостей снижает стабильность и увеличивает время выполнения тестов. С целью усиления контроля за тестируемым приложением, часть внешних сервисов заменили эмуляторами. Так, например, тестирование работы почтовых сервисов было переведено на локальный SMTP-сервис.

*Перезапуск тестов*

Существует практика многократного перезапуска нестабильных тестов, и в случае успеха такой тест признается успешным [4]. Однако это лишь маскирует реальную проблему в приложении и увеличивает время выполнения тестов. Компромиссом стало внедрение в тестовое приложение обработчика событий, который однократно перезапускает упавший тест. С одной стороны, это решение повышает стабильность тестов, а с другой стороны, сохраняет возможность получить сигнал о наличии проблемы в случае падения теста.

Особое внимание уделяется новым тестам, так как именно в этой категории, согласно исследованиям, наблюдается наибольшая доля нестабильных тестов [3]. Каждый новый тест проходит многократную проверку – 100 итераций.

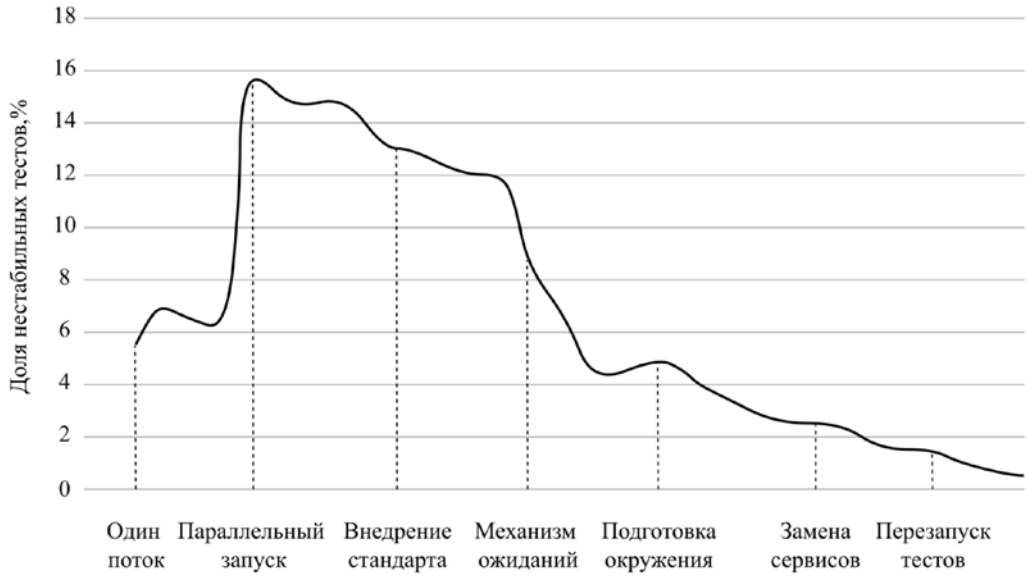


*Рис. 2. Применение методологии борьбы с нестабильными тестами*

*Оценка эффективности*

Для оценки эффективности представленной методики, на каждом этапе внедрения фиксировалось среднее значение доли недетерминированных тестов в проекте. Результаты наблюдения представлены на рис. 2.

Как видно из представленных данных, наибольшее сокращение доли нестабильных тестов обеспечило преобразование механизма ожиданий, а также изоляция тестов друг от друга. Также стоит отметить, что на начальном этапе внедрение стандарта не дало существенного сокращения нестабильных тестов. Однако по мере преобразования существующих тестов эффект от стандартизованного подхода к оформлению кода тестов только усиливался, что позволило в конечном итоге добиться снижения доли недетерминированных тестов до менее чем 1 %.

**Заключение**

Недетерминированное поведение в тестах является серьезной проблемой для большинства специалистов области автоматизации тестирования программного обеспечения. Особенно остро она проявляется при тестировании пользовательского интерфейса веб-приложений с использованием инструментов для взаимодействия с браузером, например Selenium WebDriver.

Несмотря на актуальность проблемы, механизмы преодоления нестабильного поведения в тестах программного обеспечения пока недостаточно изучены и требуют дополнительных исследований. Поэтому научное значение работы заключается в систематизации существующей информации, а также в ее дополнении, используя эмпирические данные и собственные наработки, которые легли в основу комплексного подхода к решению проблемы ненадежных тестов. Суть предложенной методики заключается в следующем:

− Формализуется процесс написания новых и преобразования существующих тестов за счет внедрения единого стандарта. Стандарт предполагает введение ограничения по объему каждого теста, а также строгие требования к оформлению. Кроме того, с целью минимизации негативных последствий для проекта автором предложен обязательный многократный запуск новых тестов перед их добавлением в тестовый набор.

− Автор отошел от общепринятых подходов к управлению ожиданиями событий в тестах: использование стандартных методов Selenium WebDriver, а также приостановка выполнения потока теста на заданное время. Оба подхода несут в себе риски сохранения нестабильного поведения в тестах, поэтому автор предлагает внедрение в существующий тестовый проект фреймворка Selenide, одной из особенностей которого является встроенный механизм умных ожиданий.

− В отличие от стандартных способов снижения зависимости тестов друг от друга, когда часть сценариев запускается в разное время, либо в проекте прописывают

очередность запуска тестов, автором реализован комплексный механизм управления тестовой средой за счет предварительного наполнения БД тестовыми данными, создания пула пользователей, генерации новых пользователей.

− Минимальное использование внешних сервисов в проекте путем их замены на локальные решения и симуляторы.

− Автор предложил отойти от практики многократного перезапуска каждого упавшего теста при высокой доле нестабильных тестов в проекте, так как это лишь маскирует проблему. В качестве оптимального решения был выбран вариант с единичным перезапуском теста, и только тогда, когда тестовый проект уже стабилизировался.

Говоря о практической значимости предложенной методики, необходимо отметить, что в ходе исследования она продемонстрировала высокую эффективность. Благодаря ее внедрению в действующий тестовый проект, команде тестирования удалось добиться поставленной цели исследования, а именно: существенно сократить долю ненадежных тестов в общем объеме сценариев с 15 % до менее чем 1 %.

Принципы, заложенные в методике, будут особенно полезны специалистам в области автоматизации тестирования пользовательского интерфейса web-приложений, как в существующих проектах, так и при проектировании новых, что значительно облегчит сопровождение тестов и их интеграцию в процесс разработки программного обеспечения.

Перспективы дальнейшего исследования заключаются в изучении новых подходов к преодолению недетерминированного поведения в тестах, а также в быстром выявлении причин нестабильности.

**Список литературы**

1. Cohn M. Succeeding with Agile: Software Development Using Scrum (1st. ed.). Addison-Wesley Professional, 2009. 512 p.

2. Software Testing Anti-patterns. [Электронный ресурс]. URL: http://blog.codepipes.com/testing/software-testing-anti-patterns.html (дата обращения: 06.06.2022).

3. Luo Q., Hariri F., Eloussi L. and Marinov D. An empirical analysis of flaky tests. In Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering (FSE 2014). Association for Computing Machinery, New York, NY, USA. P. 643–653. DOI: 10.1145/2635868.2635920.

4. Micco J. Flaky tests at Google and how we mitigate them. [Электронный ресурс]. URL: https://testing.googleblog.com/2016/05/flaky-tests-at-google-and-how-we.html (дата обращения: 06.06.2022).

5. Romano A., Song Z., Grandhi S., Yang W. and Wang W. An Empirical Analysis of UI-Based Flaky Tests, 2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE). 2021. P. 1585–1597. DOI: 10.1109/ICSE43902.2021.00141.

6. Zhang S., Jalali D., Wuttke J., Muşlu K., Lam W., Ernst M., Notkin D. Empirically revisiting the test independence assumption. Proceedings of the 2014 International Symposium on Software Testing and Analysis. 2014. P. 385–396. DOI: 10.1145/2610384.2610404.

7. Listfield J. Where do our flaky tests come from? [Электронный ресурс]. URL: https://testing.googleblog.com/2017/04/where-do-our-flaky-tests-come-from.html (дата обращения: 06.06.2022).

8. Eck M., Palomba F., Castelluccio M., Bacchelli A. Understanding flaky tests: The developer's perspective. Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering, ser. ESEC/FSE 2019. New York, NY, USA: Association for Computing Machinery. 2019. P. 830–840. DOI: 10.1145/3338906.3338945.

9. Fowler M. Eradicating Non-Determinism in Tests. [Электронный ресурс]. URL: https://martinfowler.com/articles/nonDeterminism.html (дата обращения: 06.06.2022).

10. The Code Gang. Flaky Tests – A War that Never Ends. URL: https://hackernoon.com/flaky-tests-a-war-that-never-ends-9aa32fdef359 (дата обращения: 06.06.2022).

11. Google Style Guides. [Электронный ресурс]. URL: https://google.github.io/styleguide/ (дата обращения: 06.06.2022).

12. Garcia B., Munoz-Organero M., Alario-Hoyos C. et al. Automated driver management for Selenium WebDriver. Empir Software Eng. 2021. Vol. 26. P. 107. DOI: 10.1007/s10664-021-09975-3.