DOI 10.37882/2223–2966.2023.04.19

АВТОМАТИЗАЦИЯ ТЕСТИРОВАНИЯ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ СРЕДСТВАМИ ФРЕЙМВОРКОВ

AUTOMATION OF SOFTWARE TESTING USING FRAMEWORKS

Yu. Latin

Summary. The article discusses the features of software testing using frameworks. Separately, the advantages of frameworks for automating program testing are highlighted. The advantages and disadvantages of the most known systems are also described. In addition, the study presents a description of the author's development of "Automated Testado development and testing support system", which allows to reduce the time and financial costs of regression testing.

Keywords: program, test, framework, code, speed.

Латин Юрий Эдуардович
Генеральный директор Bell Integrator (АО Бэлл Интегратор); Казахский Государственный Национальный Университет им. Аль-Фараби, Казахстан, г. Алматы
yury.latin@gmail.com

Аннотация. В статье рассмотрены особенности тестирования программного обеспечения с использованием фреймворков. Отдельно выделены преимущества фреймворков для автоматизации тестирования программ. Также описаны достоинства и недостатки наиболее известных систем. Кроме того, в процессе исследования представлено описание авторской разработки «Автоматизированная система поддержки разработки и тестирования Testado», которая позволяет снизить временные и финансовые затраты на проведение регрессионного тестирования.

Ключевые слова: программа, тест, фреймворк, код, скорость.

Воздействие развития глобальной сети Интернет на человечество не имеет исторических аналогов. Фактически это ознаменовало начало эпохи проникновения технологий, цифровых инноваций во все сферы жизни человека. Вследствие этого программное обеспечение стало неотъемлемой частью повседневной жизни общества на современном этапе [1]. Оно касается миллионов людей в разных сферах, и это в свою очередь требует от него бесперебойной работы, надежного функционирования и генерирования ожидаемых результатов, достичь которых возможно благодаря проведению надлежащего тестирования на различных этапах его создания и функционирования.

Тестирование — это процесс анализа программного средства и документации, которая его сопровождает, с целью выявления несоответствия спецификации и имеющегося продукта для повышения его качества [2]. Необходимо отметить, что тестирование пронизывает весь жизненный цикл программного обеспечения, начиная от его проектирования и заканчивая неопределенно долгим этапом эксплуатации. Вследствие этого перед разработчиками всегда возникают проблемы упорядочения действий, касающихся анализа и оценки продукта, которые должны иметь вид связанного процесса, что позволит рационально распределять ресурсы, а также принимать обоснованные решения относительно начала и завершения процессов проверки.

Обозначенные задачи и проблемы эффективно решаются за счет использования автоматических систем для тестирования программного обеспечения. Их широкая популярность и распространение связаны с тем, что они позволяют снизить затраты на обслуживание и усилия по тестированию, а также дают возможность обеспечить более высокую рентабельность инвестиций для групп контроля качества, стремящихся оптимизировать свои гибкие процессы [3].

Особо широкое применение на сегодняшний день приобрела автоматизация тестирования программ с использованием фреймворков, которые дают возможность тестировщикам комбинировать практики и инструменты для того, чтобы достичь более высоких результатов.

Таким образом, приведенные обстоятельства обуславливают актуальность рассматриваемой проблематики и предопределяют выбор темы данной статьи.

Проблемы автоматизации тестирования программного обеспечения нашли свое отражение в трудах таких авторов как: Бурбин А.В., Полевщиков И.С., Баяндин К.Н., Muccini, H.; Bertolino, A.; Inverardi, P.

Особенности использования алгоритмических моделей знаний для автоматизации тестирования программных продуктов описываются Буйневичем М.В.,

Таблица 1. Достоинства и недостатки фреймворков, которые используются для автоматизации тестирования программ

Название системы	Достоинства	Недостатки
Линейная система автоматизации	Нет необходимости писать собственный код, поэтому опыт автоматизации тестирования не требуется Быстрый способ создания тестовых сценариев Рабочий процесс тестирования легче понять любой стороне, участвующей в тестировании, поскольку сценарии расположены последовательно Это самый простой способ приступить к автоматизированному тестированию, особенно с новым инструментом	Сценарии, разработанные с использованием этой платформы, нельзя использовать повторно Данные закодированы в тестовом сценарии, что означает, что тестовые случаи нельзя повторно запустить с несколькими наборами, и их необходимо будет изменить, если данные будут изменены Обслуживание считается хлопотным, потому что любые изменения в приложении потребуют много переделок
Модульная система автоматизации тестирования	Если в приложение вносятся какие-либо изменения, необходимо будет исправить только модуль и связанный с ним отдельный тестовый сценарий, а это означает, что остальная часть приложения может быть нетронутой Создание тестовых случаев требует меньше усилий, поскольку тестовые сценарии для разных модулей можно использовать повторно	Данные закодированы в сценарии тестирования, поскольку тесты выполняются отдельно, поэтому нельзя использовать несколько наборов данных Для настройки фреймворка необходимы знания в области программирования
Система, управляемая ключевыми словами	Необходимы минимальные навыки написания скриптов Одно ключевое слово можно использовать в нескольких тестовых сценариях, поэтому код можно использовать повторно Тестовые сценарии могут быть созданы независимо от тестируемого приложения	Первоначальная стоимость установки фреймворка высока Необходимо владеть хорошими навыками автоматизации тестирования Ключевые слова могут создавать проблемы при масштабировании тестовой операции
Система тестирования на основе библиотечной архитектуры	Как и в случае с модульной структурой, использование этой архитектуры приведет к высокому уровню модульности, что упростит обслуживание и масштабируемость тестов и сделает их более экономичными Эта структура имеет более высокую степень повторного использования, поскольку существует библиотека общих функций, которые могут быть задействованы в нескольких тестовых сценариях	Тестовые данные закодированы в сценарии. Следовательно, любые изменения данных потребуют изменений в сценариях Для написания и анализа общих функций тестовых сценариев необходимы технические знания Тестовые сценарии требуют больше времени для разработки
Структура, управляемая данными	Тесты могут выполняться с несколькими наборами данных Можно быстро протестировать несколько сценариев, меняя данные, тем самым уменьшая количество необходимых сценариев Можно избежать жесткого кодирования данных, поэтому любые изменения в тестовых сценариях не влияют на используемые данные и наоборот Экономия времени	Необходим опытный тестер, который владеет различными языками программирования, чтобы правильно использовать этот фреймворк Настройка среды, управляемой данными, занимает значительное количество времени
Гибридная система тестирования	Гибридная платформа представляет собой комбинацию любого из ранее упомянутых фреймворков, поэтому ей присущи их достоинства и недостатки	

Гановым Г.А., Израиловым К.Е., Bucchiarone, A.; Muccini, H.

Однако, несмотря на имеющиеся труды и наработки, ряд вопросов в исследуемой предметной плоскости остается открытым и требует более детального изучения и анализа. В частности, нерешенными являются проблемы, связанные с выбором наиболее приемлемой среды тестирования для программы. Также в уточнении нуждаются перспективы развития гибридной среды для автоматизированного тестирования.

Итак, с учетом вышеизложенного, цель статьи заключается в рассмотрении особенностей проведения автоматизированного тестирования программного обеспечения средствами фреймворков.

Фреймворк для автоматизации тестирования — это платформа, которая представляет собой комбинацию программ, компиляторов, функций, инструментов и т.д. Она обеспечивает среду, в которой можно выполнять сценарии автоматизированного тестирования [4]. Преимуществами использования фреймворков в процессе

тестирования программного обеспечения являются следующие.

1. Возможность повторного использования кода. Поскольку фреймворки поставляются с информацией кодирования, которая необходима для успешного проведения автоматизированного тестирования, ценные данные сохраняются для будущего использования и могут быть повторно применены в любой момент времени. Нет необходимости вставлять коды вручную или переставлять их [5].
2. Низкая стоимость. Разработка тестовых ситуаций или примеров стоит достаточно дешево, потому что фреймворки уже имеют установленные правила. Кроме того, поскольку эти коды можно использовать многократно, стоимость и время создания тестовых примеров для новых функций значительно уменьшается.
3. Минимальное ручное вмешательство: фреймворки автоматизации работают в соответствии с руководящими принципами. Т.к. максимальное покрытие уже встроено и достигнуто на начальном этапе, для запуска тестов автоматизации требуется очень мало или вообще не требуется вмешательство человека. Если процесс не удался, системы автоматизации могут быть запущены повторно с некоторыми изменениями, но данные остаются постоянными и не требуют дополнительных усилий со стороны отдельного человека или команды.
4. Повышенная эффективность. Фреймворки автоматизации тестирования повышают производительность благодаря стандартизации. Она гарантирует максимальное покрытие тестов, поскольку набор кодов с самого начала выполняется стандартным образом.
5. Исправление ошибок на ранней стадии. Используя правильные средства автоматизации тестирования, можно реализовать концепцию оценки «со сдвигом влево». Это относится к идее о том, что следует перенести тестирование как можно раньше в жизненный цикл разработки программного обеспечения.

На сегодняшний день существует шесть наиболее распространенных типов фреймворков для автоматизации тестирования, каждый из которых имеет свою архитектуру, а также преимущества и недостатки: линейная система автоматизации; модульная система автоматизации; система тестирования на основе библиотечной архитектуры; структура, управляемая данными; система, управляемая ключевыми словами; гибридная система тестирования.

В таблице сгруппированы достоинства и недостатки этих фреймворков.

Для усовершенствования процедур автоматизированного тестирования программного обеспечения, снижения временных и финансовых затрат на проведение регрессионного тестирования, автором была разработана и запатентована Автоматизированная система поддержки разработки и тестирования «Testado» [6].

Эта система дает возможность фиксировать действия пользователя и оформлять их в виде тест-кейса, который имеет структурированный вид. Также необходимо обратить внимание на тот факт, что в тест-кейсе указывается перечень осуществляемых шагов, фиксируются входные данные и ожидаемые результаты.

Кратко опишем алгоритм работы предложенной системы.

1. Начало работы с системой. На данном этапе разрабатывается и формализуется «план тестирования», который представляет собой файл проекта. В нем содержатся данные о проекте, для которого будут создаваться тест-кейсы, обозначены настройки для кодогенерации, приведены ссылки на созданные проекты с исходными кодами автотестов.
2. Проведение анализа требований к программному обеспечению, который может быть реализован в структурированном или свободном формате. На основании этого определяются объекты и виды тестирования, устанавливаются параметры тестирования и кодогенерации нагрузочных и автоматизированных тестов.
3. Запись тест-кейсов. После того как тест-кейс успешно записан, происходит его автоматическое сохранение и привязка к текущему плану тестирования. Благодаря этому он доступен для корректировки, запуска, удаления или генерации кода на его основе.
4. Настройка тест-кейсов и планов тестирования. На данном этапе пользователь имеет возможность изменять наборы входных и выходных данных, объединять тест-кейсы в группы, кастомизировать настройки кодогенерации.
5. Запуск процедуры кодогенерации, согласно указанным в плане тестирования настройкам.
6. Данный этап алгоритма задействуется в том случае, если возникает потребность покрыть unit-тестами уже существующие исходные коды программного обеспечения. Для этого следует выбрать файл проекта и обозначить подробности технологического стека, на котором реализованы исходники.
7. Кодогенерация unit-тестов проводится согласно настройкам, обозначенным в плане тестирования.

8. Запуск тест-кейсов, а также автоматических, нагрузочных и unit-тестов.
9. Запись возникающих ошибок.
10. Выгрузка записанных ошибок в файлы или их экспорт в одну из систем.

Подводя итоги проведенного исследования, отметим, что фреймворки открывают широкие возможности для повышения эффективности тестирования программ. Однако необходимо подходить взвешенно к выбору конкретного их типа, учитывая текущие потребности и возможности разработчика.

В статье представлено описание авторской «Автоматизированной системы поддержки разработки и тестирования «Testado».

ЛИТЕРАТУРА

1. Баканова Т.Ю., Лашманова М.Г., Савиных Е.А., Серова Т.Н. Методика тестирования пакета программ «Логос» // Вопросы атомной науки и техники. Серия: Математическое моделирование физических процессов. 2020. № 1. С. 66–76.
2. Sun, Baicai Integrating DSGEO into test case generation for path coverage of MPI programs // Information and software technology. 2023. Volume 153.
3. Gobert, Maxime Best practices of testing database manipulation code // Information systems. 2023. Issue 111; pp 14–19.
4. iOS Code Testing: Test-Driven Development and Behavior-Driven Development with Swift / Abhishek Mishra. Berkeley, CA: Apress, 2017. 440 p.
5. Иванников Д.В., Копий А.А. Оптимизация временных затрат тестирования программного продукта с применением технологии GIT // I-methods. 2021. Т. 13. № 2. С. 34–39.
6. Патент № 2022684404, 14.12.2022. Автоматизированная система поддержки разработки и тестирования «Testado» // Патент России № 2022681905. 2022. Бюл. № 12 / Латин Ю.Э.

Журнал «Современная наука: актуальные проблемы теории и практики»

г. Алма-Аты