ORES
Научно-практический электронный журнал
Оригинальные исследования (ОРИС), *ISSN 2222-0402*
www.ores.su
original-research.ru

УДК 002.304

КАТАЛОГИЗАЦИЯ И ТИПИЗАЦИЯ ПРОГРАММНЫХ СРЕДСТВ ДЛЯ АВТОМАТИЧЕСКОЙ ГЕНЕРАЦИИ АВТОТЕСТОВ

Кудинов Никита Георгиевич,
руководитель Центра R&D, Технопарк Магика,
генеральный директор, ООО «ЗРЕНИЕ 2.0» (резидент Фонда «Сколково»)
аспирант по направлению «Медиакоммуникация и журналистика»
Донской государственный технический университет
Россия, г. Ростов-на-Дону

Катунин Сергей Дмитриевич,
Студент, магистрант
1 курс, факультет «Отдел магистратуры», кафедра «Медиатехнологии»
Донской Государственный Технический Университет
Россия, г. Ростов-на-Дону

Аннотация

Целью статьи является проведение глубокого анализа и сравнительного обзора программных средств для автоматической генерации автотестов в сфере разработки программного обеспечения. Авторы систематизируют инструменты по различным критериям, включая языки программирования, платформы и функциональные возможности. Методы исследования включают каталогизацию и детальный анализ выбранных инструментов. На основе этого проводится сравнительный анализ, выявляя преимущества и недостатки каждого инструмента. Выводы статьи включают рекомендации по выбору инструментов в зависимости от конкретных потребностей проекта, подчеркивая уникальные характеристики каждого инструмента и их применимость в различных сценариях разработки.

Ключевые слова: автоматизация тестирования, программные средства, автотесты, сравнительный анализ, рекомендации, выбор инструментов.

CATALOGING AND TYPING OF SOFTWARE TOOLS FOR AUTOMATIC GENERATION OF AUTOTESTS

Nikita G. Kudinov,
Head of R&D Center, Technopark Magika,
General Director, LLC "Sight 2.0" (Skolkovo Foundation resident)
Postgraduate student in the field of Media Communication and Journalism
Don State Technical University
Russia, Rostov-on-Don

Sergey D. Katunin,
student, master's student
1st year, Faculty "Master's Degree Division", Department "Media Technologies"
Don State Technical University
Russia, Rostov-on-Don
archdornan47@gmail.com

ABSTRACT

The aim of the article is to conduct a comprehensive analysis and comparative overview of software tools for automatic test generation in the field of software development. The authors categorize the tools based on various criteria, including programming languages, platforms, and functional capabilities. The research methods involve cataloging and a detailed analysis of the selected tools. Subsequently, a comparative analysis is conducted, highlighting the advantages and disadvantages of each tool. The article's conclusions include recommendations for choosing tools based on the specific needs of a project, emphasizing the unique characteristics of each tool and their applicability in various development scenarios.

Keywords: test automation, software tools, automated tests, comparative analysis, recommendations, tool selection.

В современной динамичной сфере разработки программного обеспечения, где каждый день появляются новые технологии и методологии, автоматизация тестирования стала фундаментальной составляющей успешного создания высококачественных продуктов. Этот процесс, который представляет собой переход от ручного тестирования к использованию автоматизированных тестов, вносит существенные изменения в привычные практики разработки и тестирования.

Автоматизированные тесты стали неотъемлемым инструментом разработчиков, позволяя им значительно сократить время цикла разработки. Вместо того чтобы тратить часы на ручное тестирование каждого компонента приложения, команды разработки могут использовать автотесты для быстрого и эффективного выявления ошибок и недочетов. Это не только ускоряет процесс, но и обеспечивает надежность и стабильность разрабатываемого продукта.

Однако, несмотря на все преимущества автоматизации тестирования, успешная реализация этого процесса требует грамотного выбора инструментов и их дальнейшей каталогизации. Существует множество инструментов для автоматизации тестирования, каждый из которых предназначен для решения определенных задач и адаптирован под определенные технологические стеки. Именно здесь кроется необходимость систематизации и каталогизации этих инструментов, чтобы разработчики и тестировщики могли эффективно выбирать те, которые наилучшим образом соответствуют требованиям и особенностям их проектов [1].

Этот выбор включает в себя рассмотрение различных критериев, таких как языки программирования, платформы, функциональные возможности, производительность, удобство использования и расширяемость. Каждый инструмент имеет свои сильные и слабые стороны, и правильная каталогизация позволяет разработчикам принимать

информированные решения в соответствии с уникальными потребностями и характеристиками их проектов.

Автоматизированные тесты подразумевают использование различных инструментов, каждый из которых может быть ориентирован на определенный язык программирования. Например, Selenium предоставляет поддержку для Java, Python, C#, Ruby и JavaScript, что делает его универсальным для различных проектов. Appium, с другой стороны, специализируется на автоматизации мобильных приложений и поддерживает те же языки программирования. В свою очередь, TestNG и JUnit предназначены преимущественно для тестирования Java-приложений [2].

Каждый инструмент имеет свою специфику использования в зависимости от платформы, на которой разрабатывается и тестируется приложение. Selenium ориентирован на веб-приложения, в то время как Appium предоставляет средства для автоматизации тестирования мобильных приложений на платформах Android и iOS. JUnit и TestNG используются в основном для тестирования Java-приложений. Каталогизация инструментов по платформам помогает выбрать подходящий инструмент для конкретного проекта.

Важным аспектом выбора инструмента для автоматической генерации автотестов являются их функциональные возможности. Например, Selenium предоставляет широкие возможности для взаимодействия с веб-элементами и эмуляции действий пользователя. TestNG, в свою очередь, предлагает расширенные возможности по сравнению с JUnit, такие как параллельное выполнение тестов и управление зависимостями между тестами. Hypothesis и QuickCheck специализируются на автоматической генерации тестовых данных и проверке свойств программ [3].

Важным этапом в выборе инструментов для автоматической генерации автотестов является проведение сравнительного анализа основных характеристик каждого инструмента. Ниже представлен более подробный обзор пятерки инструментов, включая Selenium, Appium, JUnit, TestNG, Hypothesis и QuickCheck, с акцентом на производительность, удобство использования и расширяемость.

Selenium:

Производительность: Selenium обеспечивает высокую производительность в тестировании веб-приложений. Однако, при тестировании больших и сложных приложений, использование Selenium Grid для параллельного выполнения тестов может стать необходимостью.

Удобство использования: Selenium предоставляет простой и интуитивно понятный интерфейс для взаимодействия с веб-элементами. Веб-разработчики часто предпочитают Selenium из-за его обширной документации и поддержки сообщества.

Расширяемость: Selenium легко расширяется с использованием различных языков программирования и сторонних библиотек, что позволяет адаптировать его под конкретные потребности проекта [4].

Appium:

Производительность: Appium предоставляет высокую производительность для тестирования мобильных приложений. Возможность параллельного выполнения тестов на разных устройствах значительно ускоряет процесс.

Удобство использования: Appium предоставляет удобный и единый интерфейс для автоматизации как нативных, так и гибридных мобильных приложений. Обширная документация и активное сообщество облегчают использование инструмента.

Расширяемость: Appium является гибким и расширяемым инструментом, который легко интегрируется с различными языками программирования и фреймворками тестирования.

JUnit:

Производительность: JUnit обеспечивает хорошую производительность для тестирования Java-приложений. Возможность параллельного выполнения тестов позволяет ускорить тестовые сценарии.

Удобство использования: JUnit славится своей простотой использования и низким порогом входа. Его аннотации делают написание тестов интуитивно понятным для разработчиков.

Расширяемость: JUnit обладает расширенной системой расширений, что позволяет адаптировать его под специфические потребности проекта.

TestNG:

Производительность: TestNG предоставляет возможности параллельного выполнения тестов и легкость конфигурации, что положительно сказывается на производительности.

Удобство использования: TestNG предоставляет дополнительные функциональности по сравнению с JUnit, включая группировку тестов, зависимости и параллельное выполнение, что делает его более гибким для управления тестовыми сценариями.

Расширяемость: TestNG обладает богатой экосистемой плагинов и API, что обеспечивает высокую степень расширяемости.

Hypothesis и QuickCheck:

Производительность: Hypothesis и QuickCheck, как библиотеки для автоматической генерации тестовых данных, обеспечивают высокую производительность при проверке свойств программы.

Удобство использования: Оба инструмента предоставляют простые API для генерации тестовых данных и проверки свойств, но их использование может потребовать понимания особенностей Python и Haskell соответственно.

Расширяемость: Hypothesis и QuickCheck являются расширяемыми, позволяя разработчикам определять свои стратегии генерации данных и свойства для проверки [5].

Сравнительный анализ выявляет, что каждый инструмент обладает своими уникальными преимуществами и подходит для определенных сценариев использования. При выборе следует учитывать конкретные потребности проекта, уровень опыта команды разработки и совместимость инструмента с технологическим стеком проекта.

Выбор инструмента для автоматической генерации автотестов должен зависеть от конкретных потребностей проекта. Если важна универсальность и широкая поддержка языков программирования, Selenium может быть предпочтителен. Для мобильных приложений — Appium. В случае Java-приложений, JUnit и TestNG предоставляют хорошие возможности. Hypothesis и QuickCheck подходят для проектов на Python и Haskell соответственно, если важна автоматическая генерация тестовых данных.

Таким образом, правильный выбор инструментов зависит от уникальных требований проекта и уровня опыта команды разработки. Эффективное использование автоматизированных тестов с использованием подходящих инструментов может значительно улучшить качество и производительность разрабатываемого программного обеспечения.

Список литературы:

1. Wang S. et al. Automatic unit test generation for machine learning libraries: How far are we? // 2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE). – IEEE, 2021. – С. 1548-1560.

URL: https://ieeexplore.ieee.org/abstract/document/9402041 (дата обращения: 15.12.2023)
2. Zhang S. et al. Combined static and dynamic automated test generation //Proceedings of the 2011 international symposium on software testing and analysis. – 2011. – С. 353-363. URL: https://dl.acm.org/doi/abs/10.1145/2001420.2001463 (дата обращения: 15.12.2023)
3. Li M. et al. An automatic generation tool for unit test case based on dynamic symbolic execution //The Proceedings of the International Conference on Nuclear Engineering (ICONE) 2019.27. – The Japan Society of Mechanical Engineers, 2019. – С. 1944. URL: https://www.jstage.jst.go.jp/article/jsmeicone/2019.27/0/2019.27_1944/_article/-char/ja/ (дата обращения: 16.12.2023)
4. Mg R. P. Learning Selenium Testing Tools. – Packt Publishing Ltd, 2015. URL: https://books.google.nl/books?hl=ru&lr=&id=P7vNBgAAQBAJ&oi=fnd&pg=PP1&dq=selenium+autotest+tools&ots=tYvQtXGW-t&sig=iUVHodngqqXjecZPmrUrA2XO8JA&redir_esc=y#v=onepage&q=selenium%20autotest%20tools&f=false (дата обращения: 16.12.2023)
5. Abraham R., Erwig M. AutoTest: A tool for automatic test case generation in spreadsheets //Visual Languages and Human-Centric Computing (VL/HCC'06). – IEEE, 2006. – С. 43-50. URL: https://ieeexplore.ieee.org/abstract/document/16987602001463 (дата обращения: 16.12.2023)