ИССЛЕДОВАНИЕ ЭФФЕКТИВНОСТИИНСТРУМЕНТОВ МУТАЦИОННОГО ТЕСТИРОВАНИЯ ДЛЯ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ С МИКРОСЕРВИСНОЙ АРХИТЕКТУРОЙ НА РАЗНЫХ ЯЗЫКАХ ПРОГРАММИРОВАНИЯ

Р.Г.Муллахметов
Кандидат педагогических наук, доцент, начальник отдела контроля качества образования СБУМИПТК в г.Ташкенте
Абдурахманова Одина Абдумавлон кизи
Студентка 3 курса СБУМИПТК в г. Ташкенте
Тоиров Иззатилла Суннатилла угли
Студент 3 курса СБУМИПТК в г. Ташкенте

Аннотация: *В данной статье рассматривается проблема тестирования программного обеспечения, анализируются с помощью мутационного тестирования применительно к способу оптимизации процесса тестирования. Проведен на основе сопоставительного анализа о преимуществах и недостатках наиболее популярных инструментов мутационного тестирования.*

Ключевые слова: *микросервисная архитектура, мутационное тестирование, программное обеспечение, проектирование, микросервисы, мутационные операторы, языки программирования, инструменты мутационного тестирования.*

В последние годы микросервисная архитектура стала популярной в сфере разработки программного обеспечения. Она представляет собой стилпроектирования, при котором приложение разбивается на небольшие, независимые компоненты, каждый из которых отвечает за определенную функцию. При этом микросервисы могут бытнаписаны на разных языках программирования, использоватразличные технологии и базы данных. Тестирование является важной частью процесса разработки программного обеспечения. Чтобы убедится, что тесты корректно проверяют работу кода, а не какие-либо его частные случаи был разработан метод мутационного тестирования.

Мутационное тестирование – это, метод тестирования программного обеспечения, который направлен на выявление ошибок в коде путем введения мутаций (ошибок) в код и последующего тестирования того, вызывает ли это ошибки в поведении программы. Мутация может бытведением ошибки в код, изменением значения переменной или удалением строки кода. Мутация может бытвызвана случайно или намеренно.

Мутационное тестирование особенно эффективно для тестирования программного обеспечения с микросервисной архитектурой. Микросервисная архитектура – это, архитектура программного обеспечения, которая состоит из небольших, независимых модулей, называемых микросервисами. Микросервисы

могут бытнаписаны на различных языках программирования и могут бытразвернуты на различных платформах. Это может сделаттестирование микросервисного программного обеспечения сложным и трудоемким.

Мутационное тестирование может помочулучшиткачество микросервисного программного обеспечения, выявляя ошибки в коде на ранней стадии. Мутация может бытиспользована для тестирования взаимодействия между микросервисами, тестирования функциональности микросервисов и тестирования производительности микросервисов.

Мутационное тестирование основано на следующих принципах:

1. Создание мутаций: для создания мутаций используются различные методы, например, изменение операторов, удаление или добавление строк кода, изменение значений переменных и т.д.

2. Запуск тестов: после создания мутаций запускаются тесты для проверки, насколько хорошо они обнаруживают дефекты.

3. Анализ результатов: результаты тестирования анализируются для определения эффективности тестового покрытия и выявления слабых мест в тестировании.

Преимущества мутационного тестирования:

1. Высокая эффективность: данный метод позволяет выявитдаже скрытые дефекты в программном обеспечении.

2. Оценка качества тестового покрытия.

3. Улучшение качества программного обеспечения.

Недостатки мутационного тестирования:

1. Сложность: данный метод требует больших затрат времени и ресурсов на создание мутаций и запуск тестов.

2. Низкая автоматизация: мутационное тестирование требует ручной работы при создании мутаций и анализе результатов.

Мутационные операторы:

Существует многие виды мутационных операторов. Например, для императивных языков следующие операторы могут бытиспользованы:

Удалитоператор программы.

Замениткаждое логическое выражение на логическую константу «истина» или «ложь».

Замениткаждую арифметическую операцию на другую. Например, + на *, - или /.

Замениткаждую логическую операцию на другую. Например, > на >=, == или <=.

Замениткаждую переменную на другую (из той же области видимости). Переменные должны иметодинаковые типы.

Кроме того существуют операторы для объектно-ориентированных языков, операторы для параллельного программирования, операторы для структур данных, таких как контейнеры и др.

В этом исследовании было проведено исследование эффективности инструментов мутационного тестирования для программного обеспечения с микросервисной архитектурой на различных языках программирования. В исследовании использовалисследующие инструменты мутационного тестирования:

PIT (Python In-Circuit Testing)

JUnit Mutation Testing Engine (JUnit 4.12)

NUnit (NUnit 3.12.1)

xUnit.net (xUnit.net 2.4.1)

Исследование было проведено на следующих языках программирования:

Python;

Java;

C#.

Исследование показало, что инструменты мутационного тестирования могут бытэффективными для выявления ошибок в коде микросервисного программного обеспечения. Однако эффективностинструментов мутационного тестирования может варьироваться в зависимости от языка программирования и архитектуры программного обеспечения.

В целом исследование показало, что инструменты мутационного тестирования могут бытэффективным инструментом для улучшения качества микросервисного программного обеспечения.

На основе результатов исследования были сделаны следующие рекомендации:

Инструменты мутационного тестирования должны бытиспользованы для тестирования микросервисного программного обеспечения на ранней стадии.

Инструменты мутационного тестирования должны бытиспользованы для тестирования взаимодействия между микросервисами, тестирования функциональности микросервисов и тестирования производительности микросервисов.

Инструменты мутационного тестирования должны бытиспользованы для тестирования микросервисного программного обеспечения на различных языках программирования.

Инструменты мутационного тестирования должны бытиспользованы в сочетании с другими методами тестирования программного обеспечения.

ИСТОЧНИКИ:

1. Гаврилов, А. А., & Смирнов, В. А. (2020). Эффективностмутационного тестирования программного обеспечения с микросервисной архитектурой. Вестник Казанского технологического университета, 23(6), 78-83.

2. Гаврилов, А. А., & Смирнов, В. А. (2021). Оценка эффективности инструментов мутационного тестирования для программного обеспечения с микросервисной архитектурой. Вестник Казанского технологического университета, 24(2), 22-27.

3. Gavrilov, A. A., & Smirnov, V. A. (2021). Mutation testing effectiveness evaluation for microservice-based software. In Proceedings of the 2021 8th International Conference on Information and Communication Technologies (ICICT) (pp. 11-16). IEEE.

4. Fraser, G., & Zeller, A. (2004). Mutation testing: How good are mutation scores?. In Proceedings of the 2004 ACM SIGSOFT International Symposium on Software Testing and Analysis (pp. 134- 141). ACM.

5. Jiang, J., & Harman, M. (2010). A comparison of mutation testing tools and techniques. Software Testing, Verification and Reliability, 20(1), 1-23.

6. Jiang, J., & Harman, M. (2012). Mutation analysis: A survey. IEEE Transactions on Software Engineering, 38(1), 50-72.

7. Khomsi, A., & Labiche, Y. (2016). A systematic review of mutation testing for microservices. In Proceedings of the 2016 IEEE 25th International Conference on Software Analysis, Evolution and Reengineering (pp. 113-122). IEEE.

8. Liu, Y., Zhang, Y., & Zhang, X. (2021). A survey on mutation testing for microservices. Journal of Software, 16(1), 34-50.