Науковий вісник НЛТУ України
Scientific Bulletin of UNFU
https://doi.org/10.36930/40330314
Article received 10.05.2023
Article accepted 25.05.2023
UDC 004.[05+42]
Correspondence author: O. M. Kuzmych oleh.kuzmych.mnpzm.2021@lpnu.ua
ISSN 1994-7836 (print)
ISSN 2519-2477 (online)

О. М. Кузьмич, В. Я. Лахай, М. М. Сенів
Національний університет "Львівська політехніка", м. Львів, Україна

УДОСКОНАЛЕНИЙ ПІДХІД ДО АВТОМАТИЗОВАНОГО ІНТЕГРАЦІЙНОГО ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ЗА УМОВ ЗАСТОСУВАННЯ БЕЗСЕРВЕРНОЇ АРХІТЕКТУРИ

Проаналізовано наявні підходи до автоматизованого інтеграційного тестування програмного забезпечення (ПЗ) за умов застосування безсерверної архітектури, а також причини їх появи. З'ясовано, що через популяризацію хмарних технологій та перехід до нових архітектур розроблення ПЗ, розроблення та підтримка ПЗ ускладнились. Оскільки програми стали розподіленими на велику кількість частин, а Відомо [?], що чим більше в системі елементів, тим більше можливих комбінацій дефектів, які спричиняють виникнення помилок. Вирішенням цієї проблеми мали слугувати тести, які будуть відловлювати помилки, проте більшість підходів до тестування ПЗ не є адаптованими для нових архітектур, а ті, які адаптовані, є повільними, дорогими та неавтоматизованими. Проаналізовано еволюцію тестування, пов'язану з переходом до нових архітектур. Аналіз показав як при переході до нових архітектур змінилась піраміда тестування через збільшення важливості інтеграційного тестування. Проаналізувавши наявні підходи до автоматизованого інтеграційного тестування ПЗ за умов застосування безсерверної архітектури, визначено їх переваги та недоліки, що дало змогу сформулювати вимоги до автоматизованого інтеграційного тестування ПЗ за умов застосування безсерверної архітектури. Прийнято рішення розробити власний підхід на підставі цих вимог. Сформовано удосконалений підхід до автоматизованого інтеграційного тестування ПЗ за умов застосування безсерверної архітектури, який повинен скорочувати грошові витрати на інтеграційне тестування, зменшувати тривалість інтеграційного тестування та давати його достовірні результати. Розроблено демонстраційне безсерверне ПЗ з використанням сервісів хмарного провайдера Amazon Web Services (AWS). Для автоматизації інтеграційного тестування використано конвеєр безперервної інтеграції (з англ. Continuous Integration, CI) від компанії GitLab. Результати тестування демонстраційного ПЗ з використанням запропонованого вдосконаленого підходу було порівняно з результатами тестування ПЗ з використанням наявних підходів.

Ключові слова: безсерверність; сота тестування; піраміда тестування; веб-сервіси Amazon (AWS).

Вступ / Introduction

Упродовж останнього десятиліття концепція хмарних технологій внесла неабиякі зміни в індустрію розроблення програмного забезпечення (ПЗ). Головні зміни пов'язані зі створенням та популярністю нових типів архітектур, а саме безсерверних та мікросервісних. Нові типи архітектур є розподіленими, що зумовлює серйозні складнощі в тестуванні програмного забезпечення, оскільки чим більше в системі елементів, тим більше можливих комбінацій дефектів, які спричиняють виникнення помилок, які потрібно тестувати [1].

Раніше для забезпечення якості процесу тестування програмного забезпечення використовували підходи та методи до його тестування. Ці методи здебільшого зосереджувались на модульному тестуванні, проте зараз вони є неактуальними внаслідок зміни типу архітектури. Оскільки з переходом до розподілених архітектур кількість елементів істотно зросла, важливість модульного тестування знизилася. Натомість важливість інтеграційного тестування виросла в рази, оскільки та величезна кількість сервісів має інтегруватись і працювати як одне ціле.

Здавалося, забезпечити відповідну якість тестування мають підходи до інтеграційного тестування, як до виду тестування, який зазнав найбільших змін від переходу до нових типів архітектур. Проте ці підходи не забезпечують такого самого рівня якості процесу тестування як і раніше. Отже, актуальною науково-прикладною задачею є розроблення нових удосконалених підходів до інтеграційного тестування програмного забезпечення за умов застосування безсерверної архітектури.

Об'єкт дослідження – автоматизоване інтеграційне тестування ПЗ.

Предмет дослідження – методи і засоби вдосконалення автоматизованого інтеграційного тестування про-

Інформація про авторів:
Кузьмич Олег Миколайович, здобувач, кафедра програмного забезпечення.
Email:oleg.kuzmuch@gmail.com; https://orcid.org/0000-0002-9749-6385
Лахай Владислав Ярославович, здобувач, кафедра програмного забезпечення.
Email: vlad.luckhi@gmail.com; https://orcid.org/0000-0003-1346-5731
Сенів Максим Михайлович, канд. техн. наук, доцент, кафедра програмного забезпечення.
Email: maksym.m.seniv@lpnu.ua; https://orcid.org/0000-0003-1044-4628

Цитування за ДСТУ: Кузьмич О. М., Лахай В. Я., Сенів М. М. Удосконалений підхід до автоматизованого інтеграційного тестування ПЗ за умов застосування безсерверної архітектури. Науковий вісник НЛТУ України. 2023, т. 33, № 3. С. 97–101.
Citation APA: Kuzmych, O. M., Lakhai, V. Ya., & Seniv, M. M. (2023). Improved approach to automated software integration testing under the conditions of serverless architecture. *Scientific Bulletin of UNFU*, 33(3), 97–101. https://doi.org/10.36930/40330314

грамного забезпечення за умов застосування безсерверної архітектури.

Мета роботи – удосконалити підхід до автоматизованого інтеграційного тестування програмного забезпечення за умов застосування безсерверної архітектури.

Для досягнення зазначеної мети визначено такі *основні завдання дослідження*:

- проаналізувати наявні підходи інтеграційного тестування;
- сформулювати відповідні до розробленого підходу вимоги;
- розробити власний підхід, який буде відповідати сформульованим вимогам.
- розробити демонстраційне ПЗ та протестувати ефективність створеного підходу

Аналіз останніх досліджень та публікацій. Останнім часом зростає вплив хмарних технологій на розроблення ПЗ, також під час розроблення ПЗ відбуваються зміни, які є наслідком переходу до нових типів архітектур. Особливу увагу приділяють тому, що складність розроблення ПЗ зросла, а підходи, які мають зменшувати цю складність, втратили свою актуальність через перехід до нових типів архітектур [2].

У роботах [3, 4] описують зміни, які відбулись з тестуванням при переході до нових архітектур. Автори зазначають, що першим представленням тестування програмного забезпечення є піраміда тестів. Піраміда тестування розділена на три рівні, кожен з яких відповідає типу тестування. Загалом модель піраміди показує, що тестування має бути різнобічним, але співвідношення тестів має бути таке, що модульних тестів повинно бути в рази більше, ніж наскрізних тестів. Помилки, що були знайдені під час модульного тестування, дешевше виправити, ніж помилки рівня наскрізних тестів. Чим ближче до вершини піраміди, тим більша впевненість у правильності роботи продукту [3, 4].

На тестовій піраміді зображено в якій послідовності проводяться тести в циклі CI/CD (англ. *Continuous Integration/Continuous Delivery* – безперервна інтеграція/безперервна доставка), також показано як зростає ціна та знижується швидкість тестування. Ця модель тестування була актуальною, коли ПЗ будувалось за монолітною архітектурою. Однак, коли з'явилися нові типи архітектур, виявилось, що модульні тести не забезпечують впевненість у правильній роботі ПЗ. Порівняно з традиційними, програми побудовані на нових архітектурах, зазвичай мають менше коду та більше робочих процесів багатьох модулів (сервісів) [3, 4, 5].

Відповідно, потрібно віддати перевагу інтеграційним тестам над модульними, оскільки можна бути впевненими, що частини розподіленої програми працюють належно одна з одною. Унаслідок цих змін автори роботи [5] пропонують змінити тестову піраміду на тестову соту. Автори зазначають, що тестова сота також розділена на три рівні, кожен з яких відповідає типу тестування. Проте в цій моделі розмір рівня вказує не на кількість тестів, а на їхню важливість. В усіх інших аспектах ця модель аналогічна до піраміди [5].

У цій роботі концентрація буде на інтеграційному тестуванні, оскільки воно набуло найбільшої цінності і потребує найбільших змін після переходу до нових архітектур ПЗ.

Інтеграційне тестування має вирішальне значення, оскільки воно виконується на ранній стадії розроблення та допомагає запобігти серйозним проблемам, які можуть виникнути пізніше.

Науковці в роботах [6, 7, 8] провели дослідження та виділили два види інтеграційного тестування: поступове та великого вибуху. Поступове тестування вони визначили як вид тестування, де всі логічно пов'язані модулі інтегруються, а потім проводиться тестування, щоб перевірити належну функціональність програми відповідно до вимог. Після цього інші пов'язані модулі поступово інтегруються, і процес триває, доки всі інтегровані логічно пов'язані модулі не будуть успішно протестовані. Вид інтеграційного тестування Big Bang – це вид інтеграційного тестування, який передбачає об'єднання більшості розроблених модулів у більшу систему, яка потім тестується загалом. Цей метод дуже ефективний для економії часу. Тестові випадки та їх результати мають бути правильно записані, щоб спростити процес інтеграції та дозволити команді тестування досягти своїх цілей [6, 7, 8].

Останнім часом з'явились адаптовані до хмарних технологій та нових архітектур підходи інтеграційного тестування [9]. Так, у роботі [9] розглядають три підходи до інтеграційного тестування, які адаптовані до безсерверної архітектури: локальний, гібридний та хмарний. Локальний підхід полягає у проведенні інтеграційного тестування локально. На вхід тестового сервісу будуть подаватись вхідні дані, створені вручну, а на виході результат буде звірятись з бажаним [9].

Переваги: тести проходять швидко; тести не потребують великих затрат; тести легко писати.

Недоліки:

- тести не перевіряють реальну інтеграцію сервісу;
- тести не є достовірними, оскільки не працюють з реальним сервісом [9].

Гібридний підхід – підхід, який полягає у проведенні інтеграційного тестування локально, проте тестований сервіс треба розмістити в хмарі і взаємодіяти з ним через запити [9].

Переваги: тести швидші за повністю хмарні; тести є достовірними.

Недоліки:

- тести є дорогими, оскільки піднімається реальний сервіс у хмарі;
- тести повільніші за локальні, оскільки потрібен час для підняття сервісу і відповіді від нього;
- складність реалізації тестів через взаємодію двох середовищ [9].
- хмарний підхід – підхід, який полягає у проведенні інтеграційного тестування в хмарі. Усі сервіси, так як і тести, розміщуються в хмарі і виконуються в тестовому середовищі (англ. environment) [9].

Переваги: тести є достовірними; тести легко писати.

Недоліки:

- тести є дорогими, оскільки піднімаються реальні сервіси в хмарі на окремому тестовому середовищі;
- тести повільні, оскільки потрібен час для розгортання сервісів і тестів, які їх тестують [9].

З наведеного вище можна зробити висновок, що проблема недостатньої якості інтеграційного тестування програмного забезпечення є надзвичайно актуальною на тлі переходу галузі на нові архітектури розроблення програмного забезпечення. З аналізу наявних підходів до інтеграційного тестування праць, які є адаптованими до безсерверної архітектури, стає зрозуміло, що вирішення цієї проблеми полягає у створенні вдосконаленого підходу до автоматизованого інтеграційного тестування програмного забезпечення за умов застосування безсерверної архітектури.

Результати дослідження та їх обговорення / Research result and their discussion

Розглянемо процедуру формування та опис удосконаленого підходу до автоматизованого інтеграційного тестування ПЗ за умов застосування безсерверної архітектури. Для вирішення проблеми інтеграційного тестування ПЗ за умов застосування безсерверної архітектури, удосконалений підхід повинен відповідати певним вимогам, які сформовані на підставі мінусів уже наявних підходів. Насамперед інтеграційне тестування має бути: автоматизованим; якомога дешевшим і швидшим; а також має давати достовірні результати, що інтегровані модулі ПЗ працюють належно.

Для забезпечення автоматизації інтеграційного тестування ПЗ застосуємо CI pipeline, яке буде одним з кроків у конвеєрі. Для забезпечення дешевизни і швидкості виконання інтеграційні тести будуть проводитись локально, а всі сервіси, які можливо, будуть емулюватись за допомогою спеціальних бібліотек. Це дасть змогу досягти високої, але не максимальної, впевненості надійності інтеграційних тестів. Для досягнення максимальної достовірності тестування пропонується розбити інтеграційні тести на дві частини. Зобразимо це на модифікованій соті тестування (рис. 1).

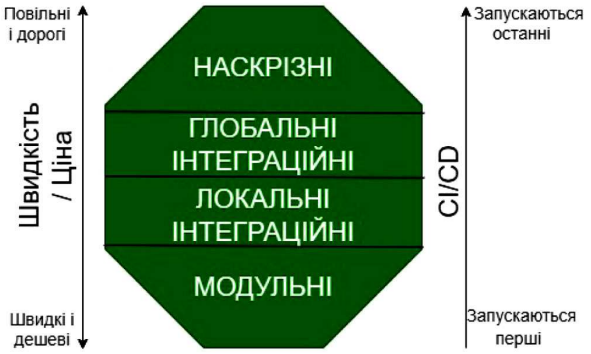

Рис. 1. Модифікована сота тестування / Modified honeycomb testing

На рис. 1 можемо побачити, що інтеграційні тести розділились на локальні інтеграційні тести та глобальні інтеграційні тести.

Локальні інтеграційні тести – це інтеграційні тести, які будуть проводитись локально, а всі сервіси, які можливо, будуть емулюватись за допомогою спеціальних бібліотек.

Глобальні інтеграційні тести – це інтеграційні тести, які будуть проводитись з реальними сервісами в хмарі. Таке розділення дасть змогу забезпечити дешевизну і швидкість тестування при використанні локальних інтеграційних тестів і надасть впевненості, що результат тестування показує чи правильно інтегровані тестові сервіси під час використання глобальних інтеграційних тестів. Також таке розділення добре накладається на процес розроблення ПЗ, оскільки не завжди і не все програмне забезпечення потрібно тестувати глобальними інтеграційними тестами.

Підсумувавши все зазначене вище, отримаємо підхід, який відповідає всім вимогам, що були сформульовані на підставі недоліків наявних підходів інтеграційного тестування. Спираючись на сучасний процес розроблення ПЗ з розділенням на три основні середовища, а саме: DEV – середовище розроблення, TEST – середовище тестування, PROD – середовище продакшену, вкажемо в якому середовищі повинно відбуватись інтеграційне тестування.

Оскільки локальне інтеграційне тестування швидке та недороге, то логічно його буде проводити так само часто як і модульне. Це дасть змогу виявляти помилки інтеграції ще на початку розроблення сервісів. Для цього локальне інтеграційне тестування потрібно проводити перед розгортанням у середовище розроблення.

Глобальні інтеграційні тести будуть відбуватись в тестовому середовищі. Зобразимо місце цих інтеграційних тестів у конвеєрах середовищ розроблення (рис. 2) та тестування (рис. 3).

Сама логіка інтеграційного тесту працює за правилом трьох AAA (англ. *Arrange, Act, Assert*). Також потрібно додати два додаткові кроки до інтеграційного тесту. Це крок затримки і крок очищення, оскільки тестування проводиться на реальних сервісах, в яких є час відгуку.

Взявши до уваги цю логіку, інтеграційний тест буде працювати так:

- CI pipeline запускає скрипт інтеграційного тесту;
- Скрипт відпрацьовує:
  - створити необхідні для сценарію умови Arrange;
  - подати на вхід тестового середовища вхідні дані Act;
  - чекати певний відтинок часу;
  - читати результат Act;
  - перевірити, що результат є бажаним Assert;
  - почистити дані, які були потрібні для тестування.

Рис. 2. CI-середовища розроблення / CI development environment

Рис. 3. CI-середовища тестування / CI testing environment

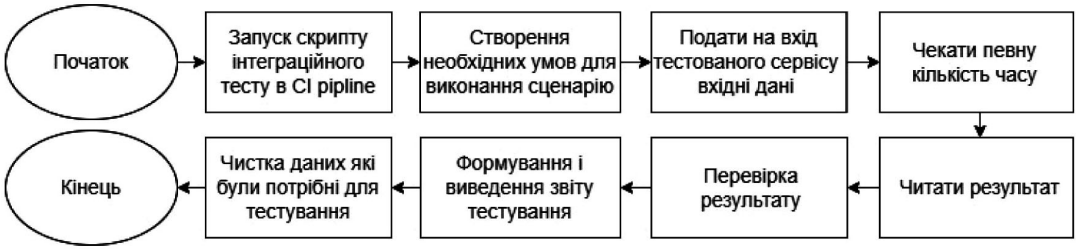

Рис. 4. Блок-схема логіки інтеграційного тесту / Block diagram of integration test logic

Розглянемо особливості розроблення ПЗ за допомогою запропонованого підходу. Створимо проєкт в GitLab з гілками dev, test, main для майбутнього накладання на них CI-конвеєрів, які будуть розгортати проект у відповідному до гілки середовищі AWS.

Для демонстрації було обрано розробити безсерверний застосунок з такою архітектурою (рис. 5).

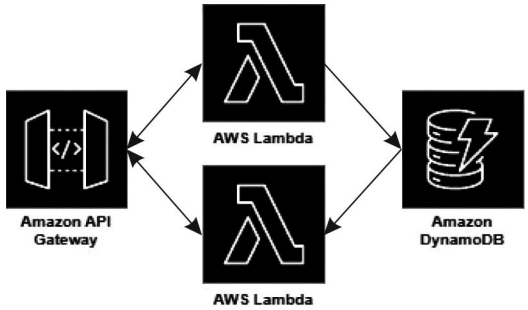

Рис. 5. Архітектура демонстраційного ПЗ / Architecture of the demo software

Застосунок складається з трьох сервісів AWS: Api Gateway, AWS Lambda, DynamoDB. Розроблюваний застосунок буде приймати POST-запит на сервіс API Gateway, який буде тригерити Lambda для запису інформації в базу DynamoDB. Також застосунок зможе отримувати інформацію з бази даних. Запит GET. ПЗ будемо розробляти за технологією CDK, що дасть змогу легко автоматизувати розгортання застосунку у хмарі.

Для перевірки ефективності кожного з підходів інтеграційні тести будуть тестувати одну і ту саму інтеграцію сервісів Lambda з DynamoDB, які використовуються для запису інформації в базу даних.

На першому кроці розробимо інтеграційні тести за локальним підходом, згідно з яким тестування проводиться локально на машині розробника, а для симуляції сервісів використовується mock (з англ. висміювання).

Наступним кроком буде розроблення інтеграційних тестів відповідно до вдосконаленого підходу, тестування проводиться локально на машині розробника, а для симуляції сервісів використовується бібліотека moto.

Код інтеграційного тесту працює так:

- імпортуються необхідні бібліотеки, серед яких і moto;
- відбуваються приготування перед тестуванням, а саме створення симульованої DynamoDB таблиці;
- відбувається запис інформації в базу даних;
- перевіряється чи записана інформація з'явилася;
- видається відповідний результат;
- тестова таблиця знищується.

Ця структура відповідає розробленій логіці інтеграційного тесту, яку зображено на рис. 4. Останнім кроком буде розроблення інтеграційних тестів відповідно до гібридного підходу, де тестування проводиться локально на машині розробника, проте перед тестуванням тестований сервіс розгортається в хмарі.

Код гібридного підходу інтеграційних тестів сильно схожий на код інтеграційних тестів за власним підходом. Різниця полягає тільки в етапі підготовки до тестування. В удосконаленому підході на етапі підготовки локально створюється емуляція бази даних, а в гібридному – ця база даних розгортається в хмарі.

Створимо pipeline середовища DEV згідно з послідовністю дій, наведеною на рис. 2. GitLab pipeline визначають у файлі.gitlab-ci.yml. Створимо pipeline середовища TEST за послідовністю дій, наведеною на рис. 3. Як видно з отриманих результатів, тести, які розроблені за власним підходом і за локальним підходом, працюють значно швидше. Це логічно, оскільки для гібридного інтеграційного тестування потрібен час на розгортання тестового сервісу.

Таблиця. Швидкість виконання інтеграційних тестів за різними підходами / Speed of execution of integration tests by full approaches

Назва підходу, за яким розроблено інтеграційний тест	Час проведення інтеграційного тесту, у секундах
Локальний	0.499
Розроблений	0.504
Гібридний	42.99 + 0.948 = 43.938

Примітка: 42.99 – час розгортання тестованого сервісу в хмарі, а 0.948 – час виконання тесту.

Швидкість виконання тестів за власним підходом і за локальним підходом однакова, проте якість зовсім різна. Це пов'язано з тим, що при використанні власного підходу створюється реальна таблиця за допомогою moto, з можливість отримати назад елемент із DynamoDB. Ці дві відмінності від звичайного локального тестування, з використанням mock, сприяють більшій впевненості розробників ПЗ в достовірності результатів інтеграційного тестування.

Глобальні інтеграційні тести не розроблялись в цій роботі, оскільки вони не відрізняються від тестів при застосуванні хмарного підходу. Відмінність полягає тільки в тому, що розроблюваний підхід дає рекомендації застосовувати глобальні інтеграційні тести на тестовому середовищі, для впевненості, що сервіси працюють правильно разом. Це дасть змогу з'економити величезний відтинок часу як тестувальниів, так і розробників ПЗ, оскільки вони не повинні довго чекати за кожного надсилання змін до репозиторію, який, водночас, починає роботу CI конвеєра.

Також важливо зазначити, що за гібридного та глобального тестування можливі зняття коштів з акаунту AWS, на відміну від розробленого підходу, який не працює з реальними сервісами, а симулює їх. Кількість знятих коштів залежить від кількості використаних ресурсів, це одна з головних концепцій безсерверності.

Обговорення результатів дослідження. Упродовж останніх років питання підходу до тестування безсерверного ПЗ актуалізувалася через стрімкий перехід індустрії з монолітної на безсерверну архітектуру. До розгляду цього питання долучилась низка авторів у роботах [2, 3, 4, 5]. Кожен з них частково описав причини та наслідки змін, які відбулись з розробкою ПЗ при переході до безсерверної архітектури. Одним з цих наслідків стала трансформація піраміди в соту тестування, через збільшення важливості інтеграційного тестування при переході до безсерверної архітектури. За ці декілька років популярності хмарних технологій і нових архітектур почали з'являтись адаптовані до них підходи інтеграційного тестування [6, 7, 8, 9]. Три такі підходи описав автор в роботі [9]. Однак у кожного з цих підходів є неділіки, а саме: недостатня швидкість тестування, дороговизна та недостовірність результатів. Також в усіх трьох підходів є недолік, що вони не є автоматизованими. Запропонований удосконалений підхід покликаний усунути ці недоліки.

Отже, за результатами виконаної роботи можна сформулювати такі наукову новизну та практичну значущість результатів дослідження.

Наукова новизна отриманих результатів дослідження – удосконалено підхід до інтеграційного тестування безсерверних застосунків, який, на відміну від наявних, дає змогу автоматизувати цей процес, зменшує його тривалість та вартість.

Практична значущість результатів дослідження – можливість використання розробленого підходу для проведення інтеграційного тестування наявного чи розроблюваного ПЗ, яке побудовано з використанням безсерверної архітектури. Використання підходу дасть змогу досягти пришвидшення у 87 разів порівняно з використанням гібридного підходу. Також зменшаться грошові витрати на інтеграційне тестування в середньому на 1 % коштів від загальних витрат на місяць.

Висновки / Conclusions

Запропоновано удосконалений підхід до автоматизованого інтеграційного тестування безсерверних застосунків, який дає змогу зменшити його тривалість та вартість. За результатами дослідження можна зробити такі основні висновки.

1. Проаналізовано еволюцію тестування, пов'язану з переходом до нових архітектур, та вказано важливість саме інтеграційного тестування внаслідок того, що сучасне ПЗ стало розподіленим (програма складається з багатьох компонентів, які працюють разом). Визначено проблему інтеграційного тестування, яка полягає у складності розроблення тестів, швидкості і дороговизні виконання.
2. Проаналізовано наявні підходи до інтеграційного тестування, які є адаптованими до безсерверної архітектури. На підставі аналізу стало зрозуміло, що вирішення проблеми якісного інтеграційного тестування полягає у створенні удосконаленого підходу до автоматизованого інтеграційного тестування ПЗ за умов застосування безсерверної архітектури. Сформульовано вимоги, яким повинен відповідати удосконалений підхід.
3. Розроблено власний удосконалений підхід до інтеграційного тестування, який, на відміну від наявних, дає змогу автоматизувати процес інтеграційного тестування безсерверних застосунків та зменшує тривалість інтеграційного тестування.
4. Розроблено демонстраційне ПЗ на безсерверній архітектурі, яке представлене у вигляді безсерверного REST API з можливістю запису і читання з хмарної бази даних. Для демонстраційного ПЗ розроблено інтеграційні тести за трьома підходами: удосконаленим, локальним і гібридним.
5. Проведено порівняння результатів тестування, які свідчать про пришвидшення у 87 разів порівняно з використанням гібридного підходу та економію в середньому 1 % коштів від загальних витрат на місяць.

У подальших наукових роботах плануємо розробити удосконалений підхід до наскрізного тестування.

References

1. Ugrynovsky, B. (2022). Methods and means of increasing the reliability of software, taking into account the process of its software aging, Ph.D. dissertation, Lviv Polytechnic National Univ., Lviv, Ukraine.
2. Lakhai, V., Kuzmych, O., & Seniv, M. (2022). An improved approach to the development of software with increased requirements for flexibility and reliability in terms of creating small and medium-sized projects. 17th International Conference on Computer Sciences and Information Technologies (CSIT), Lviv, Ukraine. https://doi.org/10.1109/csit56902.2022.10000787
3. Radziwill, N. (2020). Reframing the Test Pyramid for Digitally Transformed Organizations. Software Quality Professional. https://doi.org/10.48550/arXiv.2011.00655
4. Cohn, M. (2010). Succeeding with agile: Software development using Scrum. Upper Saddle River, NJ: Addison-Wesley.
5. Contan, A. (2018). Test automation pyramid from theory to practice. International Conference on Automation, Quality and Testing, Robotics (AQTR). https://doi.org/10.1109/AQTR.2018.8402699
6. Sotiriadis, S., & Lehmets, A. (2017). Unit and Integration Testing of Modular Cloud Services. 31st International Conference on Advanced Information Networking and Applications (AINA), Taipei, Taiwan. https://doi.org/10.1109/AINA.2017.57
7. Singh, R. (2012). An Approach for Integration Testing in Online Retail Applications. *International Journal of Computer Science and Information Technology*, 4(3), 141–158. https://doi.org/10.5121/ijcsit.2012.4312
8. Anwar, N. (2022). Review Paper on Various Software Testing Techniques & Strategies. *Global Journal of Computer Science and Technology*, 43–49. https://doi.org/10.34257/gjcstcvol19is2pg43
9. Eetu, R. (2022). Testing Approaches And Tools For AWS Lambda Serverless-Based Applications. 2022 International Conference on Pervasive Computing and Communications Workshops and other Affiliated Events (PerCom Workshops), Pisa, Italy. https://doi.org/10.1109/percomworkshops53856.2022.9767473

O. M. Kuzmych, V. Ya. Lakhai, M. M. Seniv

Lviv Polytechnic National University, Lviv, Ukraine

IMPROVED APPROACH TO AUTOMATED SOFTWARE INTEGRATION TESTING UNDER THE CONDITIONS OF SERVERLESS ARCHITECTURE

With the popularization of cloud technologies and the transition to new software development architectures, software development and maintenance have become incredibly complex. As programs began to be divided into a large number of parts, and it is known that the more elements in a system, the more possible combinations of defects that cause errors. The solution to this problem should be tests that will detect errors, but most approaches to software testing are not adapted to new architectures, and those that are adapted are slow, expensive and unautomated. Therefore, this work is relevant, as there is a great need for an improved approach to automated integration testing of software under the conditions of serverless architecture. Measurement and comparison methods are used in the work, as the own approach is quantitatively compared with existing ones. In this work, the evolution of testing associated with the transition to new architectures is analyzed for the first time. The analysis showed the growing importance of integration testing. The analysis of existing approaches to software integration testing enabled determining their advantages and disadvantages, which, in turn, allowed formulating the requirements that an improved approach to automated software integration testing should have under the conditions of using a serverless architecture. Based on these requirements, we decided to develop our own approach. An improved approach to automated integration testing of software under serverless architecture conditions has been developed, which should be automated, reduce the monetary costs spent on integration testing, reduce the duration of integration testing and ensure reliable test results. Demo software was developed, on which the effectiveness of the formulated approach was tested in comparison with existing ones. In further research, it is promising to develop an improved approach to end-to-end software testing under the conditions of using a serverless architecture, which will allow forming a complete approach to testing serverless applications.

Keywords: serverless; honeycomb testing; pyramid testing; Amazon Web Services (AWS).