УДК 621.396.1

СКЛАДНОЩІ НАСКРІЗНОГО E2E ТЕСТУВАННЯ ТЕЛЕКОМУНІКАЦІЙНИХ СИСТЕМ

Остапов О.А., Чуб М.М., Афанасьєва Л.О.
Навчально-науковий інститут телекомунікаційних систем КПІ ім. Ігоря Сікорського, Україна
E-mail: liana.afanasyeva@gmail.com

CHALLENGES OF END-TO-END TESTING OF TELECOMMUNICATION SYSTEMS

End-to-end (E2E) network testing is critical for service providers, it helps to identify bugs early in software development process but also presents some major challenges. Testing triangle, challenges and solutions for microservices testing are presented.

Задля забезпечення сучасних потреб клієнтів, телекомунікаційні компанії впроваджують нові послуги, використовують новітні технології і постійно розширюють свою інфраструктуру. Оскільки послуги майже ніколи не бувають атомарними, а майже завжди інтегровані одна з одною, технологічні системи, що надають ці послуги - тісно пов’язані між собою. А кожен зв’язок, інтеграція однієї системи з іншою не може відбутись без детального аналізу та інтеграційного скрізного E2E тестування [1].

Для гарантування, що впроваджувані послуги - найвищої якості, QA інженерам варто розуміти основні категорії ризиків та потенційних проблем, і як насліжок - шляхи запобігання ризиків і вирішення проблем. Основні проблеми тестування телекомунікаційних системи можна розбити на категорії:

- Організаційні;
- Інфраструктурні;
- Програмні;
- Пов’язані з даними.

Організаційні ризики і проблеми, хоч і не пов’язані напряму а ні з програмним забезпеченням, а ні з апаратними комплексами, часто є одними з найсуттєвіших. Типовий приклад - розподіл необхідних прав доступів QA інженерам, коли задля задовільнення вимог департаменту безпеки, необхідні спеціалісти банально не мають змоги почати процес E2Eтестування, чи отримують його пізніше запланованого, що, відповідно, зменшує час на проведення системних інтеграційних тестів та ретест виявлених проблем. Інша типова організаційна проблема - налагодження зручних і гнучких процесів розробки і тестування. Незважаючи на декларування гнучкого підходу до проєктування (Agile) - у великих організаціях, такі як телеком компанії, багато процесів залишаються строгими, послідовними і незграбними. Більш того, в багатьох випадках до тестування ставляться як до контролю якості, тобто тест інженерів залучають запізно, в той час, як прогресивна спільнота рухається в бік QA та TestOps підходів [2].

Інфраструктурні ризики пов’язані з неточним чи несвоєчасним розгортанням необхідної інфрастркуктури як для процесу розробки,

тестування, так і для впровадження. Не зважаючи на те, що ідеологія DevOps має вирішувати проблеми, нажаль, часто вона створює нові, такі як виділення окремих ролей для налаштування середовищ, недостатність ресурсів, нехтування тестовими середовищами, відсутність моніторингу та інженерів, що відповідають за нього

Програмні проблеми - найбільш близькі та очевидні колу QA інженерів. Сучасні розподілені системи часто базуються на мікросервісній архітектурі. Раннє тестування мікросервісів є ефективним методом виявлення помилок на початкових етапах розробки програмного забезпечення. Модифікована піраміда тестування враховує особливості контролю якості розподілених систем на мікросервісній архітектурі й включає підходи для тестування мікросервісів з нижнього рівня.

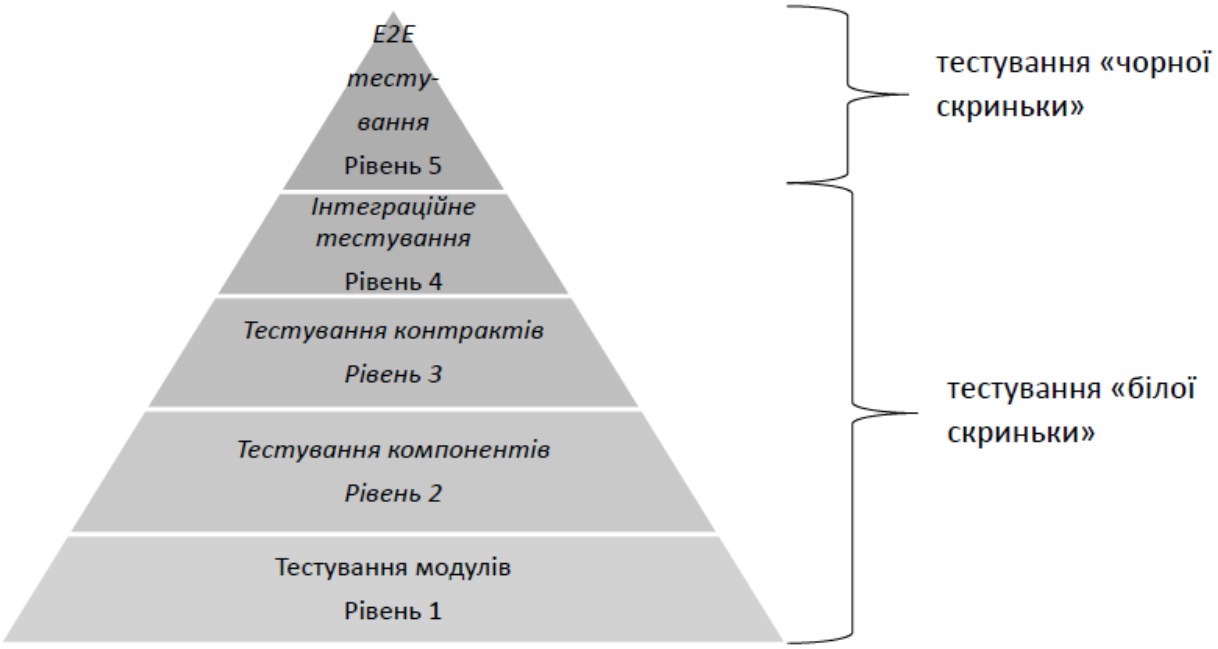

Рис.1. Трикутник тестування.

Цей підхід є частиною методології "Shift-left" у тестуванні програмного забезпечення, яка спрямована на перенесення тестування на ранні етапи розробки. Метою створення додаткових суміжних рівнів тестової піраміди для мікросервісів є виявлення різних типів проблем на початкових етапах розробки систем, що дозволяє зменшити кількість проблем в експлуатації. Кожен тип тестування спрямований на перевірку очікуваних результатів на різних рівнях програмної системи. Для розподіленої системи на мікросервісній архітектурі можна організувати тести в наступні етапи:

1. Тестування модулів - тестується невелика одиниця вихідного коду мікросервісів та перевіряється поведінка методів або функцій вихідного коду всередині мікросервісу, імітуючи залежні модулі та тестові дані.

2. Тестування компонентів - це тестування спрямоване на незалежне тестування всіх функціональних можливостей мікросервісів та API-інтерфейсів для окремо взятих мікросервісів.

3. Тестування контрактів - перевіряє узгоджені контракти між різними доменними мікросервісами; тестується частина інтеграції: між мікросервісом та підключеними базами даних, виклики API між двома мікросервісами.

4. Інтеграційне тестування - використовується для тестування та перевірки всієї функціональності шляхом тестування всіх пов'язаних мікросервісів.

5. E2E тестування - здійснюється наскрізне тестування функціональності системи з метою перевірки її відповідності функціональним бізнес-цілям користувача, клієнта або потенційного клієнта. E2E тестування здійснюється на зовнішньому інтерфейсі (UI) або викликах клієнта API, зокрема за допомогою REST клієнтів та проводиться в різних розподілених мікросервісах та додатках SPA/MFE. Даний вид тестування включає у себе перевірку інтерфейсу користувача, серверних мікросервісів, баз даних та їх внутрішніх/зовнішніх компонентів.

Для збільшення ефективності тестування на будь-якому етапі можна проводити тестування не з точки зору чорного ящика (QA інженер не знає, як влаштована система), а з точки зору білого ящика - включаючи перегляд вихідних кодів окремих мікросервісів, внутрішніх логів, даних в мережевих повідомленнях та записів в базах даних.

Остання, але не по значимості, категорія проблем - проблеми з даними. По-перше, в QA інженерів мають бути описи правильних і не правильних даних, які можна використовувати в інтеграційному тестуванні. По-друге, мають бути приклади самих даних. Раніше, в часи до GDPR, багато телекомунікаційних компаній використовували мінімально змінені дані зі своїх існуючих продуктових систем, оскільки ніщо не може бути кращим, ніж справжні дані. Зараз ситуація для тестування стала складніша, оскільки правильні дані для всіх мікросервісів, в потрібних кількостях потрібно генерувати самим інженерам. На їх же плечі лягає тягар забезпечення цілосності даних у випадку розподілених систем, керування даних для тестових команд і команд розробки.

Є різні підходи, від проактивних, системних, що мають на меті вирішити саму причину проблем. Власне, формулювання проблеми - вже перший крок до її вирішення, оскільки інженери часто надають перевагу не помічати очевидне. Інший підхід - реактивний - вирішення конкретних проблем по факту їх виявлення. Не зважаючи на очевидні переваги першого підходу, кожен має як переваги, так і недоліки.

Література

1. Dave Westerveld, API testing and development with postman : a practical guide to creating, testing, and managing APIs for automated software testing, 2021. – 341.
2. P. Ammann and J. Offutt, Introduction to Software Testing, Cambridge:Cambridge University Press, 201.