УДК 681.3

А. Батюк, Н. Васіна
Національний університет “Львівська політехніка”,
кафедра автоматизованих систем управління

ТЕХНОЛОГІЯ ФУНКЦІОНАЛЬНОГО ТЕСТУВАННЯ ПРОГРАМНОГО ПРОДУКТУ ЗА ДОПОМОГОЮ СКРИПТІВ

Щоб програмне забезпечення безпомилково працювало в призначеному середовищі, необхідне тестування, за допомогою якого мають не тільки ефективно знаходитися дефекти, але також, наскільки можливо, швидко і дешево виконуватись тести. Автоматизація тестування дасть змогу значно зменшити зусилля, необхідні для адекватного тестування або значно покращити тестування, проведення якого обмежено часом.

Software must be tested to have confidence that it will work as it should in its intended environment. Software testing needs to be effective at finding any defects which are there, but it should also be efficient, performing the tests as quickly and cheaply as possible. Automating software testing can significantly reduce the effort required for adequate testing, or significantly increase the testing which can be done in limited time.

Вступ. Як показує практика, якість програмного забезпечення дуже часто визначається під час його експлуатації. Водночас замовник вимагає від фірм-розробників гарантій для отриманого програмного забезпечення. Сучасні засоби розроблення інформаційних систем дають змогу побудувати “каркас” веб-ужитку або програмного продукту швидко, але не завжди якісно. Тому керівники компаній, що займаються випуском програмного забезпечення, все більше уваги приділяють тестуванню своїх продуктів, чітко усвідомлюючи взаємозв’язок між процесом тестування та якістю кінцевого програмного продукту. Чим пізніше починається тестування програмного продукту, тим збільшуються ризики, що система загалом буде ненадійною. А вартість виправлень в коді кінцевого програмного продукту буде дуже високою. Отже, тестування перестало бути низькопріоритетним ужитковим процесом. Тепер процес тестування починає свій життєвий цикл паралельно з розробленням складних програмних систем.

Очевидно, якщо фірма створить програмне забезпечення низької якості, то наслідком буде втрата існуючих та потенційних клієнтів і можливості подальшого співробітництва з ними. А від цього залежить фінансовий стан фірми і, можливо, навіть саме існування фірми в майбутньому.

Однією з причин неправильного ставлення до процесу тестування розробників програмних продуктів є складність самого процесу тестування. Адже він передбачає комплекс різноманітних задач і дій для перевірки відсутності помилок і виявлення дефектів в програмних системах, оцінки продуктивності, контролю наявності та повноти документації, оцінки якості прийнятих проектних рішень. Процес тестування передбачає існування об’єкта тестування (програмне забезпечення) та еталона, з яким цей об’єкт порівнюється. Програмне забезпечення – це складний об’єкт, властивості якого змінюються на різних стадіях розроблення. Дуже важливо узгодити вимоги до програмного забезпечення між стороною замовника і стороною розробника ще до початку проектування. Основною властивістю вимог до програмного забезпечення є можливість їхньої перевірки, тобто можливість тестування на відповідність вимогам. Вимоги є двох типів: функціональні (які функції і з яким результатом виконуватиме програмне забезпечення) та нефункціональні (наприклад, обмеження на час розв’язання задачі, швидкість доступу до даних, вимоги до ресурсів тощо). Потім

логічний набір тестових вимог групується в тестові сценарії, перевірка яких дасть однозначну відповідь про коректність роботи певних функцій в системі. Якщо всі вимоги не буде зазначено вчасно, то замовник може отримати не те, що хотів, а програмне забезпечення неможливо буде коректно протестувати через відсутність еталона. У замовника і розробника повинна бути можливість порівняти поточне функціонування системи з її еталонною, очікуваною поведінкою. Обговорення технічного завдання, технічного проекту, архітектури системи з замовником також допомагає знайти помилки і уточнити еталон. Головне правило для розробника – тісна співпраця з замовником для знаходження спільних рішень в процесі створення програмного забезпечення. Отже, тестування необхідно проводити на відповідність наперед визначеним вимогам функціональності, продуктивності, безпеки тощо. Періодичне тестування інформаційної системи протягом життєвого циклу дасть змогу забезпечити постійну відповідність експлуатаційних показників заданим. Оскільки об'єкт тестування складний, то необхідно застосовувати системний підхід для планування, організування, проведення тестування. Враховуючи високу вартість інформаційних систем, що розробляються, необхідно вже на стадії проектування оцінювати показники продуктивності на основі методів моделювання та спеціальних тестів.

Визначення, здавалось би, такого простого терміна, як "тестування", різноманітні джерела дають різні. У цій статті ми хотіли б звернутися до двох міжнародних стандартів та їхнього тлумачення поняття "тестування".

Згідно з **RUP (Rational Unified Process) методологією** [1] тестування орієнтовано на оцінку якості за допомогою таких методів:

– пошук і документування дефектів якості;

– загальні рекомендації відносно якості;

– перевірка виконання основних очікувань і вимог із застосуванням конкретних прикладів;

– перевірка, чи продукт функціонує саме так, як було запроектовано;

– перевірка, чи вимоги виконано відповідним чином.

Згідно з міжнародним стандартом **IEEE Std 829-1983 (Standard for Software Test Documentation)** [2] тестування – це процес аналізу програмного забезпечення, спрямований на виявлення відмінностей між його властивостями, які реально існують, і тими, що від нього вимагаються (дефект) і на оцінку властивостей програмного забезпечення.

Узагальнюючи ці визначення, можна сказати, що тестування – це процес, який полягає в перевірці відповідності програмного забезпечення заявленим характеристикам і вимогам, вимогам експлуатації в різних середовищах, з різними навантаженнями, вимогам з безпеки, вимогам з ергономіки і зручності користування [3]. Якщо виконувати тестування поетапно, то так набагато ефективніше контролювати розроблення програмного забезпечення і знизити ризик помилок та виправлень для всього проекту розробки.

Кожен процес тестування має своє спрямування і мету, згідно з якою перевіряється певна особливість ужитку чи веб-сайту. Процес тестування прийнято документувати за допомогою тестового плану і тест-кейсів. Тестовий план – це організована структура тестових даних, що визначає стратегію тестування, методи і засоби тестування, порядок тестування, ресурси для тестування та інші його особливості. А тест-кейси – це мінімальні елементарні операції перевірки функціональності для кожного елемента чи функції ужитку. Описуються вони послідовними покроковими операціями. Хоча тестування – це досить творчий процес, його необхідно планувати. Тобто, на кожному етапі робіт повинні бути вибрані критерій якості і критерій завершення тестування. Тестувальник перевірятиме програмне забезпечення на відповідність критерію якості. Критерій завершення тестування – відповідь на запитання: за яких умов тестування може бути завершене, якщо вичерпані всі відведені ресурси. Тестовий план дійсно потрібний, бо для перевірки складного об'єкта необхідно застосовувати різні стратегії, які дають змогу досягти максимального результату при існуючих обмеженнях на ресурси. Крім того, план тестування дає змогу наперед визначити, що необхідно підготувати для проведення тестування. Тестовий план визначається

міжнародним стандартом **IEEE Std 829-1983,** що передбачає три обов'язкові розділи з такими описами:

– що буде тестуватися (тестові вимоги, тестові варіанти);

– якими методами і наскільки докладно тестуватиметься система;

– план-графік робіт і необхідні ресурси (персонал, техніка).

Додатково описуються критерії успішного/неуспішного завершення тестування, критерії завершення тестування, ризики, непередбачені ситуації, наводяться посилання на проектну документацію.

Тестування загалом вирішує ряд дуже важливих задач:

– забезпечує якість кінцевого продукту;

– підтверджує, що всі визначені функціональні вимоги реалізовані, ужиток вимогам відповідає і немає помилок в програмному коді;

– підтверджує, що ужиток здатен виконуватись у всіх визначених режимах, підтримує визначені операційні системи, веб-браузери;

– гарантує, що дані, які зберігаються і обробляються ужитком, надійно захищені від стороннього доступу чи "зламу";

– визначає, яке максимальне навантаження на сервер, локальну мережу чи базу даних ужиток може коректно опрацювати;

– дає змогу впевнитися в тому, що користувач зможе "інтуїтивно" використовувати програмний продукт, не плутаючись у складних інтерфейсах.

Згідно з **RUP (Rational Unified Process) методологією** [1] прийнята базова класифікація видів тестування:

1. Функціональне тестування:
   - безпосередньо функціональне тестування (Function testing);
   - тестування цілісності даних (Data integrity testing);
   - тестування на різних платформах (Configuration testing);
   - тестування стійкості до відмов (Failover & recovery testing);
   - тестування доступу (Security testing);
   - інсталяційне тестування (Installation testing);
   - тестування користувацького інтерфейсу (User interface testing).
2. Навантажувальне тестування:
   - безпосередньо навантажувальне тестування (Load testing);
   - тестування продуктивності (Performance testing);
   - тестування робочого циклу (Business cycle testing);
   - тестування з великим користувацьким навантаженням (Stress testing);
   - тестування з великим обсягом даних (Volume testing).

Функціональне тестування – це тестування програмного забезпечення на відповідність зовнішнім специфікаціям з метою перевірки виконання функціональних вимог. Навантажувальне – це тестування для виявлення характеристик функціонування програмного забезпечення у разі зміни навантаження (інтенсивність звернень, переповнення бази даних тощо), визначення впливу працюючого ужитку на системне середовище. Про кожний вид тестування існує велика кількість інформації, але в цій статті зупинимось на автоматизованому тестуванні, яке сьогодні є дуже популярним серед виробників програмного забезпечення.

Визначення доцільності впровадження автоматизованого тестування для програмних продуктів

Необхідно визначити, у чому сенс поняття "автоматизоване тестування". Автоматизоване тестування – це такий вид тестування, який виконується за допомогою засобів автоматизації (програмних скриптів) в автоматичному режимі. Програмний скрипт – це тестова програма, яка

автоматизує тестові перевірки, що описуються в тест-кейсах. Автоматизоване тестування вигідно застосовувати, коли продукт часто змінюється, але має велику функціональність. Так можна відслідковувати стабільність роботи всіх функцій після внесення змін у програмний продукт. Автоматизоване тестування вигідно застосовувати після створення кожної нової версії програмного продукту в циклі розроблення ужитку. При цьому участь людини не є необхідною, адже програмний скрипт може запускатись навіть вночі.

Автоматизоване тестування займає окреме місце серед наведених вище типів тестування, тому що автоматизувати можна як функціональне, так і навантажувальне тестування. Використання автоматизованого тестування – це інвестиції в майбутнє для підприємств-виробників програмного забезпечення: воно значно підвищує якість продуктів, що випускаються, а це, своєю чергою, виводить підприємство на новий, набагато вищий рівень виробництва програмного забезпечення. Правда, при цьому можуть виникати і певні труднощі та проблеми, адже перехід від "ручного" до автоматизованого тестування може забрати багато фінансових і часових ресурсів, а також потребувати переналаштування бізнес-процесів фірми:

– початок кожного автоматизованого тестування починається з ручного тестування, коли людина повинна показати роботу, що, де і як тестувати;

– створення автоматичних тестів вимагає від тестувальника програмістських знань, тому що професійна автоматизація неможлива без безпосередньої роботи з кодом програмного скрипта та знань об'єктно-орієнтованого програмування;

– чутливість автоматизованих тестів до програмного і апаратного середовища;

– автоматизацію недоцільно використовувати, коли об'єкт може протестувати тільки людина. Засіб автоматизованого тестування – це робот з певним штучним інтелектом. А відомо, що найскладнішою задачею штучного інтелекту є розпізнавання складних об'єктів і моделей поведінки. Тобто, протестувати програмний продукт на ергономіку за допомогою автоматизованого тестування неможливо.

Всі вищеперераховані складні випадки необхідно брати до уваги. Проте, якщо жоден з них не стане перепоною на шляху впровадження автоматизованого тестування і фірма почне випускати якісне автоматично протестоване програмне забезпечення набагато швидше, ніж конкуренти, то інвестування в автоматизацію виправдається не раз. Існує єдине правило, коли недоцільно застосовувати автоматизоване тестування: коли немає впевненості, що тестується один і той самий об'єкт в однакових умовах. Процес розроблення програмного забезпечення повинен передбачати такі періоди, коли зміни до коду програмного продукту не вводяться, і тестувальники можуть спокійно працювати.

Найдоцільніше автоматизувати функціональне (Function testing) та навантажувальне тестування (Load testing), але і то не завжди. Як і в будь-якій справі, приймаючи рішення про автоматизацію, необхідно керуватись принципом економічності. Якщо компанія впевнена, що випускається і в майбутньому випускатиметься багато версій програмного продукту або програмний продукт є складним для "ручного" тестування, то в такому випадку автоматизація є доцільною. Процес автоматизації функціонального тестування програмного продукту забезпечить отримання прямих фінансових вигод за рахунок економії ресурсів, а також підвищення якості робіт, які виконуються для тестування програмного забезпечення. Автоматизація зробить процес тестування інформаційної системи багаторазовим та повторюваним без участі реального тестувальника, тим самим звільняючи ресурси тестувальників від рутинних, монотонних "ручних" перевірок функціональності.

Для того, щоб застосувати технологію автоматизованого тестування, необхідно виконати такі обов'язкові дії:

– провести аудит всієї існуючої проектної документації програмного продукту чи веб-ужитку для визначення доцільності автоматизованого тестування;

– вибрати з проектної документації вимоги, які необхідно перевірити в тестуванні;

– розробити план і порядок тестування;

– створити покрокові тест-кейси для перевірки всіх вимог;

– визначити необхідні умови та ресурси для проведення тестування (програмно-апаратна платформа та людські ресурси);

– вибрати засіб чи інструментарій для автоматизованого тестування згідно з визначеними умовами;

– створити за допомогою вибраного інструментарію програмний скрипт для проходження всіх тест-кейсів;

– запустити програмний скрипт на виконання;

– результати виконання тестів зберегти для звітування.

Найістотнішими перевагами автоматизованого тестування є такі:

– автоматизоване тестування економить час, тому що програма-робот швидше виконує операції чи відсилає запити за протоколом, ніж людина – це дає змогу швидше знайти помилки і виправити їх;

– автоматизоване тестування виключає людський фактор, адже програма-робот виконуватиме постійно повторювані рутинні операції без збоїв;

– автоматизоване тестування можна використовувати в програмних продуктах, які не мають графічного користувацького інтерфейсу – це доцільно на ранніх стадіях розробки, коли тестуватимуться виконавчі модулі чи функції, які повинні правильно працювати;

– автоматизоване тестування дає змогу емулювати роботу багатьох користувачів, що є дуже важливим для навантажувального тестування;

– всі засоби автоматизованого тестування мають інструментарій для фіксації знайдених помилок і результатів, допомагаючи моделювати різноманітні помилкові ситуації, заповнювати звіти, будувати діаграми. Зберігання створених наборів тестів є дуже важливим аспектом в автоматизованому тестуванні. До тестів необхідно ставитися як до вихідного коду, тобто використовувати версії для можливості відтворення попередніх тестів, для повторного використання вже існуючих.

Отже, основні переваги і недоліки автоматизованого тестування визначено. Окремо можна сказати про людський фактор під час цього складного процесу. Як вже говорилося вище, автоматизоване тестування виконує програма-робот. Не треба забувати, що будь-який засіб автоматизованого тестування – це тільки інструмент. Тому велику увагу треба приділити плануванню, а саме створенню тестового плану і тест-кейсів. Акуратне і адекватне планування – запорука успішного автоматизованого тестування. Засіб автоматизованого тестування не писатиме тестовий план і тест-кейси, а тільки їх виконуватиме. В тест-кейсах не повинно бути нічого, крім перевірки вимог (явних і неявних), які необхідно поділити за пріоритетністю. Неадекватний тест-план і тест-кейси зведуть автоматизацію до даремної втрати часу. Тому автотест, як і будь-яка інша звичайна програма, потребує акуратного створення та постійної підтримки. А це все потребує часу, деколи не меншого, ніж на розроблення самого програмного продукту. Коли вже зрозуміло, що і навіщо тестувати, створено тестовий план і тест-кейси, необхідно з'ясувати, як тестування зробити ефективнішим, швидшим, якіснішим. Застосування "ручного" тестування до складного програмного забезпечення є дорогим, а результати після кожного "ручного" виконання тестів зникають і їх важко відтворити. Для того, щоб збільшити об'єм перевірок і підвищити якість тестування, забезпечити можливість повторного виконання тестів при внесенні змін в програмне забезпечення, застосовують засоби автоматизації. Тому тестувальник повинен володіти знаннями про можливі засоби автоматизованого тестування, знати їх переваги і недоліки для того, щоб вміло вибрати для своїх потреб найкращий інструмент. Адже не всі вони однаково працюють з усіма різноманітними платформами і протоколами, технологіями створення ужитків і обміну даними.

Аналіз існуючих засобів автоматизації

Тепер, коли ми розуміємо основні ідеї автоматизації тестування, його переваги і недоліки та знаємо, на що звертати увагу при плануванні, можна приступати до аналізу основних засобів

автоматизованого тестування. Об'єкти автоматизації тестування – це системи, що реалізують роботу клієнтської сторони. Ключовою особливістю тестування клієнт-серверних систем є можливість перевірки коректності функціонування і швидкодії системи через роботу саме клієнтської сторони. Отже, якщо ретельно перевірити ці можливості, то можна отримати гарантію працездатності системи для кінцевого користувача. Тестування – це завжди експеримент. Для його проведення потрібно мати репозиторій – структуроване сховище даних. Репозиторій має бути структурований так, щоб забезпечувати розв'язання поставлених практичних завдань. Як і в будь-якому експерименті, під час тестування необхідно збирати інформацію, обробляти результати. Доцільно використовувати версії тестових даних, створюючи та зберігаючи архіви, копії. Існує загальний поділ видів тестування: статичне і динамічне. При статичному тестуванні виконується аналіз коду, структур даних. Динамічне потребує виконання програмного забезпечення, для чого потрібні засоби автоматизованого тестування і допоміжні ужитки. Відомо дуже багато інструментів побудови і автоматичної генерації тестів, засобів моніторингу ресурсів під час виконання тестів, візуалізації результатів, статистичної обробки результатів тестування. Завдяки наявності на світовому ринку програмного забезпечення значної кількості засобів та інструментів автоматизованого тестування і певної їх схожості, перш ніж звертатися до конкретних продуктів, варто розглянути загальні принципи їхньої роботи. Для інструментів функціонального автоматизованого тестування існує схема "із чим – що – як". Щоб робот міг робити те, що вам потрібно, йому треба "пояснити", із чим працювати, що і як конкретно робити, у якій послідовності. Тобто вам необхідно створити скрипт, що містить опис тестових кроків, логіки тесту й глобальних змінних.

Програмне забезпечення для автоматизованого тестування можна умовно поділити за провідними фірмами-виробниками. Найвідоміші і потужні програмні засоби належать IBM – це так званий Rational Software [4] пакет. Також не поступається йому програмне забезпечення від Mercury Interactive [5], Segue Software [6] та Compuware [7]. Тому дуже важливим питанням є вибір найоптимальнішого засобу автоматизації для конкретних потреб фірми, керівники якої вирішили застосовувати для своїх програмних продуктів автоматизоване тестування. На вибір засобу також може вплинути його ціна, адже інструментарії автоматизованого тестування, які представлені на ринку і орієнтовані на інформаційні системи, відрізняються високою складністю, а отже, і високою вартістю. Розглянемо можливості програмних засобів для автоматизації найвідоміших світових фірм.

Продукти Rational Software від IBM – це повний життєвий цикл для автоматичного тестування, що підтримує управління вимогами, візуальне моделювання та безпосередньо автоматизоване тестування. Ці продукти підтримують:

– такі мови програмування, як C, C++, Ada, Java;

– функціональне, регресивне, навантажувальне тестування;

– створення і запуск тест-кейсів за допомогою інструкції користувача;

– запис дій користувача у програмний скрипт з подальшою можливістю його редагування та запуску;

– збереження всіх результатів тестів в репозиторії даних з можливістю доступу для користувача;

– емуляцію роботи багатьох користувачів (для навантажувального тестування).

IBM Rational TestManager значно підвищує ефективність процесу тестування. Він забезпечує команду тестувальників засобами планування, проектування, виконання і аналізу результатів тестування. Об'єднання операцій тестування в єдиний процес дає можливість команді проектувальників налагодити ефективний процес контролю за якістю програмного продукту. В IBM Rational TestManager ведеться план тестування. Джерелом конкретних сценаріїв можуть бути елементи візуальних моделей чи вимоги. За допомогою IBM Rational TestManager організовується єдиний робочий простір, що об'єднує елементи тестового плану, сценарії, тестові скрипти, звіти результатів тестування та інші необхідні дані. IBM Rational TestManager зв'язує набір тестів

конкретної ітерації процесу розроблення програмного забезпечення з конфігурацією інформаційної системи. Інтеграція IBM Rational TestManager з IBM Rational Robot дає змогу ефективно автоматизувати процес тестування завдяки об'єднанню можливостей планування та управління тестуванням з потужними засобами запису і запуску автоматизованих тестів інформаційної системи. Інтеграцією IBM Rational TestManager з IBM Rational Rose можна використовувати функціональні елементи візуальної моделі і якості джерела сценаріїв тестування. Інтеграція IBM Rational TestManager з IBM Rational RequisutePro забезпечує використання бази вимог для визначення інших сценаріїв тестування. Інтеграцією IBM Rational TestManager з IBM Rational ClearQuest має можливість прямого занесення опису дефектів до бази запитів на зміну проекту. Інтеграція IBM Rational TestManager з IBM Rational SoDA використовується для автоматичної генерації документів з усього репозиторія тестування.

Серед цих істотних переваг Rational Software існує важливе обмеження – цей пакет програм підтримує тільки версії операційної системи Windows.

Mercury Interactive пропонує продукти для тестування Windows- та Web-ужитків. Test Director – це продукт для планування тестування, його виконання та відслідковування функціональних дефектів. Цей інструмент доцільно використовувати як для "ручного", так і для автоматизованого тестування. Test Director працює з такими системами управління базами даних: Oracle, Sybase, MS SQL Server, MS Access та володіє вдалим інструментом для зберігання результатів і звітів, отриманих при тестуванні. Test Director орієнтований тільки на операційну систему Windows.

Програмний продукт від Mercury Interactive WinRunner використовується для запису і відтворення дій користувача за допомогою програмних скриптів, написаних спеціальною мовою TSL. XRunner – це аналогічний інструментарій для запису та відтворення дій користувачів, проте він орієнтований на операційну систему Unix.

Перевагами потужного засобу Mercury QuickTest Professional є зручний і зрозумілий користувацький інтерфейс для створення тестів без ручного виправлення скрипта, а недоліками є необхідність ліцензій для розповсюдження і незручність роботи з нестандартними об'єктами. QuickTest призначений для автоматизованого функціонального тестування.

Ще одним засобом для автоматизованого тестування є LoadRunner, призначенням якого є навантажувальне тестування та стрес-тестування. Працює з операційними системами Windows та Unix. Аналіз закордонної практики підтвердив, що недоцільно проводити навантажувальні експерименти на реальних діючих інформаційних системах. Спеціальні засоби тестування дають змогу знизити затрати та забезпечити інформаційну безпеку під час таких робіт. Якщо до готового робочого програмного продукту в майбутньому будуть звертатись одночасно декілька тисяч користувачів, то без LoadRunner неможливо вирішити проблему навантажувального тестування. Він гнучко емулює роботу тисячі користувачів і створює сценарії для перевірки роботи серверів, баз даних, компонентів мережі для визначення вузьких місць системи. LoadRunner – це дуже зручний інструмент, однозначний лідер, що володіє найширшим спектром можливостей.

Продукти компанії Segue Software створені для підтримки систем управління якістю та надійністю під час життєвого циклу розроблення програмного продукту. Засоби автоматизованого тестування Segue Software володіють такими можливостями:

- планування, документування та управління тестовими діями;
- забезпечення спілкування між тестувальниками і групою програмістів;
- визначення проблем під час життєвого циклу програмного забезпечення;
- запуск тестів та подальше зберігання їх результатів окремо від програмних скриптів.

Segue SilkTest – це цікавий й відносно зручний засіб, що надає широкі можливості для ручної роботи із стандартними й нестандартними об'єктами об'єктно-орієнтованою мовою 4Test. За допомогою інструментаріїв Segue Software можна автоматизувати регресивне, навантажувальне та unit-тестування.

В табл.1 наведені порівняльні характеристики засобів автоматизованого тестування [8]:

Таблиця1

Порівняльна характеристика засобів автоматизованого тестування

Види тестування, що підтримує засіб тестування	Засоби автоматизованого тестування			
	Rational Software	Mercury Interactive	Segue	Compuware (QACenter)
Тестування користувацького інтерфейсу	+	+	+	+
Навантажувальне тестування Клієнта	+	+		+
Навантажувальне тестування бази даних		+	+	+
Тестування Web-клієнта		+	+	
Навантажувальне тестування Web-сервера		+		
План тестування	+	+	+	+
Відслідковування помилок	+		+	+
Методологія	+			+
Прямий доступ до даних		+	+	+

Rational Software не підтримує навантажувальне тестування бази даних. Тому в табл.2 [8], де наведено оцінку роботи засобів автоматизованого тестування з різними системами управління базами даних (СУБД) та іншими програмними засобами, Rational Software відсутній.

Таблиця 2

Оцінка роботи засобів автоматизованого тестування з різними системами управління

Види систем управління	Засоби автоматизованого тестування		
	Mercury Interactive	Segue	Compuware (QACenter)
ODBC	П	С	С
Oracle	Х	П	Х
Sybase	Х	?	С
SQL-Server	Х	?	С
Tuxedo	С	?	С
Web	Х	?	С
PeopleSoft	Х	?	Х
SAP	П	?	П
Citrix	?	?	П
CharBased	Х	?	С
BAAN	?	?	П

У табл.2 використано такі позначення для оцінки роботи:

– П – погана;

– Х – хороша;

– С – середня;

– ? – недостатньо досліджена.

На основі аналізу табл.1 і табл.2 можна зробити висновок про те, що засіб автоматизованого тестування QACenter фірми Compuware займає одне з провідних місць серед продуктів даного класу. Крім того, QACenter орієнтований на тестування інформаційних систем з архітектурою “клієнт–сервер”, тобто є повноцінним аналогом засобу тестування. Тепер доцільно розглянути його можливості докладніше. Серед основних характеристик QACenter виділяють такі:

– архітектура для багатьох платформ, що підтримує ужитки в DOS, Windows різних версій, які працюють по протоколах TCP/IP, NETBIOS, IPX/SPX або в середовищі Web;

– можливість тестування ужитків з текстовим і графічним інтерфейсом користувача, зокрема і Web-сторінки;

– підтримує тестування ужитків на об'єктному рівні графічного користувацького інтерфейсу, зокрема й для програмних засобів 4-го покоління (4GL), таких як UNIFACE, Oracle Developer/2000, PowerBilder та ін.;

– прямий доступ до Oracle, Sybase, SQLServer, а також доступ ODBC, HTTP тощо; управління всім процесом тестування;

– підтримка видів тестування, таких як: тестування компонентів, інтегроване й регресійне тестування, тестування продуктивності/працездатності в умовах максимального навантаження, можливість тестування на всіх етапах життєвого циклу системи.

QACenter реалізований у вигляді декількох інструментаріїв:

– QADirector забезпечує управління процесом тестування, підтримує автоматизоване тестування як "клієнт-серверних" систем, так і систем на базі мейнфреймів (mainframe). Є репозиторій, що дає змогу декільком користувачам одночасно переглядати, аналізувати результати тестування та обмінюватися ними;

– QARun автоматизує трудомісткий та багаторазово повторюваний процес створення й виконання тестів, а також аналізує їхні результати. Містить структуровану мову опису послідовності дій;

– QATrack призначений для реєстрації та документування проблем під час їх виникнення, а також збирання статистики за кількістю успішних/невдалих тестів;

– QAStress призначений для проведення тестування клієнтських додатків в умовах великого навантаження на систему; вимірює час відгуку системи;

– QALoad забезпечує моделювання реальних умов експлуатації клієнт-серверного ужитку без залучення кінцевих користувачів шляхом створення віртуальних користувачів замість фізичних на рівні програмного забезпечення проміжного шару (midleware). QALoad може будувати графіки та генерувати звіти для оцінки продуктивності об'єкта тестування.

Висновки. Окрім програмних продуктів наведених вище світових компаній, існує ще досить велика кількість інструментаріїв менш відомих фірм, а також програмних продуктів з "відкритим" кодом (open source), таких як Ant, JUnit, JPrade. Комерційні продукти відомих фірм мають один істотний недолік: вони не такі ефективні, якими могли б бути. Це пов'язано з тим, що вони не здатні наслідувати новітні технології та важко пристосовуються до нових процесів в галузі програмного забезпечення. Крім того, засоби автоматизованого тестування, представлені на ринку та орієнтовані на інформаційні системи економічного спрямування, відрізняються високою складністю й, відповідно, високою вартістю. Знання того, що необхідно тестувати і з якою метою, полегшує вибір програмного продукту для автоматизованого тестування. Проте, як видно з наведеного вище аналізу найвідоміших інструментаріїв, жоден не підтримує роботи зі всіма існуючими операційними системами, мовами програмування, видами тестування, протоколами тощо. Це і є основною причиною складності вибору засобу автоматизованого тестування та застосування самої автоматизації. Тому дуже часто фірми, що приймають рішення автоматизовувати тестування для своїх програмних продуктів зі специфічними вимогами, використовують програмне забезпечення з "відкритим" кодом і вдосконалюють потрібними можливостями. Після того, як інструментарій готовий для використання, створюються і запускаються програмні скрипти та аналізуються результати. Якщо ж необхідно проводити автоматизоване тестування в операційній системі Windows, то зручніше використовувати пакет Rational Software від IBM, а для автоматизації під Unix – програмні продукти від Mercury Interactive. У такому випадку автоматизоване тестування буде доцільним, і результат перевищить очікування.

Отже, якщо необхідно швидко підготувати нові версії програмного продукту без зниження якості, якщо є прагнення зробити більше за менший проміжок часу і при цьому повністю протестувати програмне забезпечення та позбутися таких недоліків "ручного" тестування, як трудомісткість, вплив людського фактора, то необхідно автоматизувати тестування. Обмежений обсяг статті не дає нам можливості докладніше описати повний комплекс заходів автоматизованого

тестування, які б запевнили замовника в тому, що програмне забезпечення добре протестовано. Крім того, завжди існують особливості тестування, специфічні для конкретного програмного продукту. Не всі компанії-розробники володіють необхідними засобами автоматизованого тестування. Також висока вартість засобів автоматизованого тестування змушує розробників програмних продуктів не створювати чи купувати подібні засоби, а звертатися за послугами тестування в спеціалізовані випробувальні лабораторії. Часто самі замовники використовують незалежні організації для проведення "аутсорсингу" тестування ужитків, щоб впевнитися в якості створеного для них програмного продукту. Як правило, "аутсорсингу" тестування потребують біллінгові системи та системи масового обслуговування користувачів.

1. http: // www. ibm. com / developerworks / rational / library / content / RationalEdge / feb03 / RUPphilosophy_TheRationalEdge_Feb2003.pdf. 2. http://archives.maillist.ru/77933/525924.html. 3. http: // www. alexfill. ru / testing.htm. 4. http: // rational. aplana.ru / products /default. asp. 5. http: // www. mercury. com /us /. 6. http://www.borland.com/resources/en/pdf/products/silk/silktest_datasheet.pdf. 7. http: // www. compuware. com / products / qacenter / entserver. htm. 8. http: // www. gicpsvt.ru /index. php?page=ptest_analyze