УДК 004.415.533

О.А. РЕМІННИЙ
Вінницький національний технічний університет

СТВОРЕННЯ ЗВІТІВ ПРО ПЕРЕБІГ АВТОМАТИЗОВАНИХ ТЕСТІВ КОРИСТУВАЦЬКИХ ІНТЕРФЕЙСІВ

В даній роботі описано методику формування звітів про перебіг тестових сценаріїв на основі аспектно-орієнтованого програмування. Методика дозволяє залишити незмінною кількість програмного коду в рішенні автоматизованого тестування, а також зменшити час на знаходження помилки в тестевих сценаріях. Проведені експериментальні дослідження дозволили проаналізувати процес аналізу до та після використання аспектів і показали значне покращення результатів.

Ключові слова: автоматизоване тестування, аспектно-орієнтоване програмування

O.A. REMINNYI
Vinnytsya national technical university

CREATING REPORTS ABOUT GUI AUTOMATED TESTS EXECUTION

Abstract – In this paper we describe a technique to generate reports on the progress of the test scenarios based on aspect-oriented programming. As a basis we use business methods (also known as keywords in GUI automation testing) and wrap them with the technology-specific aspects. Aspects allow us to create comprehensive reports on business methods execution without changing the body and leaving all the code only with business specifics.

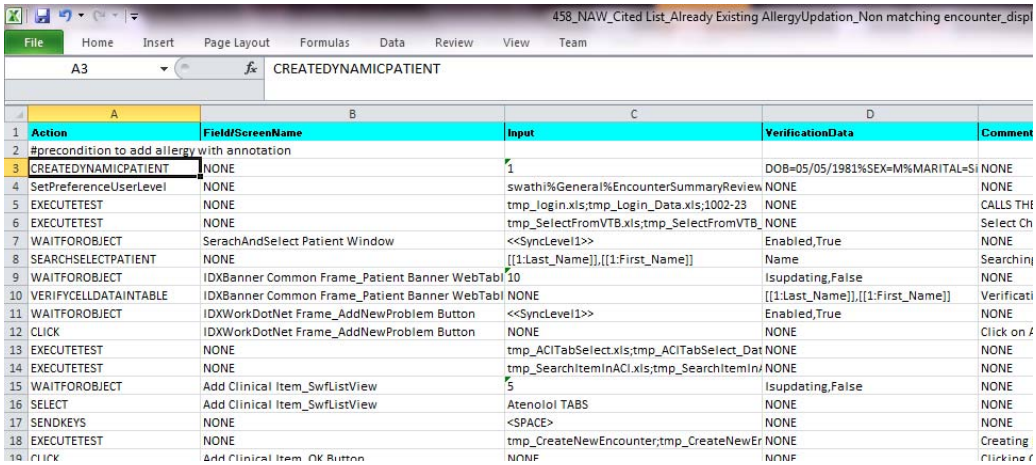
Results were calculated on a basis of two different implementation processes. We measured the time, needed to find the error source with aspect-built report and without those on a basis of Coded UI technology GUI automation testing solution. Achieved result showed two times time improvement for discovering the error source in the automation solution codebase.

Keywords: automated testing, aspect-oriented programming

Вступ

Зменшення часу на тестування є основною ціллю впровадження автоматизованого тестування в циклі розробки програмних продуктів. Однак і після впровадження автоматизованого тестування в циклі розробки є трудозатратні зони, в яких можлива оптимізація.

Для характеристики часу розробки нових тестових сценаріїв на основі існуючих бізнес методів, більш детально розглянемо процес написання нових тестових скриптів. В залежності від специфіки проекту автоматизованого тестування, системи взаємодії з користувацьким інтерфейсом системи, яка тестується, можуть бути імплементовані різні підходи до використання бізнес методів. В деяких випадках опис може бути на рівні метапрограми – наприклад в певних текстових, xml чи Excel таблицях. На Рис. 1 представлено реальний приклад працюючого скрипта, який виконується певним середовищем автоматизованого тестування. Ключові слова в цьому випадку можуть бути як бізнес методами (CREATEDYNAMICPATIENT, SEARCHSELECTPATIENT), так і певними ключовими командами взаємодії з користувацьким інтерфейсом, як то CLICK для кнопок чи SENDKEYS для певних текстових полів.



458_NAW_Cited List_Already Existing AllergyUpdation_Non matching encounter_displ

File Home Insert Page Layout Formulas Data Review View Team

A3 | CREATEDYNAMICPATIENT

	A	B	C	D	
1	**Action**	**Field/ScreenName**	**Input**	**VerificationData**	**Comment**
2	#precondition to add allergy with annotation				
3	CREATEDYNAMICPATIENT	NONE	1	DOB=05/05/1981%SEX=M%MARITAL=Si	NONE
4	SetPreferenceUserLevel	NONE	swathi%General%EncounterSummaryReview	NONE	NONE
5	EXECUTETEST	NONE	tmp_login.xls;tmp_Login_Data.xls;1002-23	NONE	CALLS THI
6	EXECUTETEST	NONE	tmp_SelectFromVTB.xls;tmp_SelectFromVTB_	NONE	Select Ch
7	WAITFOROBJECT	SerachAndSelect Patient Window	<<SyncLevel1>>	Enabled,True	NONE
8	SEARCHSELECTPATIENT	NONE	[[1:Last_Name]],[[1:First_Name]]	Name	Searchin
9	WAITFOROBJECT	IDXBanner Common Frame_Patient Banner WebTabl	10	Isupdating,False	NONE
10	VERIFYCELLDATAINTABLE	IDXBanner Common Frame_Patient Banner WebTabl	NONE	[[1:Last_Name]],[[1:First_Name]]	Verificati
11	WAITFOROBJECT	IDXWorkDotNet Frame_AddNewProblem Button	<<SyncLevel1>>	Enabled,True	NONE
12	CLICK	IDXWorkDotNet Frame_AddNewProblem Button	NONE	NONE	Click on A
13	EXECUTETEST	NONE	tmp_ACITabSelect.xls;tmp_ACITabSelect_Dat	NONE	NONE
14	EXECUTETEST	NONE	tmp_SearchItemInACI.xls;tmp_SearchItemIn/	NONE	NONE
15	WAITFOROBJECT	Add Clinical Item_SwfListView	5	Isupdating,False	NONE
16	SELECT	Add Clinical Item_SwfListView	Atenolol TABS	NONE	NONE
17	SENDKEYS	NONE	<SPACE>	NONE	NONE
18	EXECUTETEST	NONE	tmp_CreateNewEncounter;tmp_CreateNewEr	NONE	Creating
19	CLICK	Add Clinical Item_OK Button	NONE	NONE	Clicking (

Рис. 1. Приклад Excel скрипта

Загалом така імплементація ключовими словами є загальновідома, загальновизнана і популярна[1]. Однак верифікація скриптів значно ускладнюється. Потрібен додатковий запуск всього скрипта, щоб перевірити чи правильно сформована послідовність кроків. Опис тестового скрипта за допомогою стрічкових констант не дає можливості перевірити правильність написання ключових слів, помилка може бути спричинена просто орфографічною помилкою. Параметри, які передаються ключовим словам, також не мають ніякого механізму перевірки на їх кількість та тип, як то пропонується в сучасних середовищах програмування, як наприклад Microsoft Visual Studio.

Аналіз процесу розробки рішення автоматизованого тестування

Час, який витрачається тестером на формування сценаріїв, можна описати як суму двох основних частин: часу на опис та на налагодження сценарію:

$$t_{Wf} = ManError * tImplement_{Wf} + tDebug_{Wf},\ ManError > 1, \quad (1)$$

де *ManError* – коефіцієнт можливості людської помилки в зв'язку з тим, що середовище ніяк не перевіряє та не підказує можливі правильні кроки в побудові тестового сценарію, не підкреслює орфографічні помилки в написі тестових сценаріїв, не перевіряє типи параметрів (які лише під час виконання перетворюються зі стрічки Excel документу в реальний параметр кроку;

$tImplement_{Wf}$ – час на розробку тестового сценарію;

$tDebug_{Wf}$ – час на налагодження тестового сценарію. Цей час залежить від того, наскільки просто виявити місце, де відбулося помилка виконання в тесті. Для процесу налагодження тестових сценаріїв, важливим фактором є час пошуку помилки в звіті про виконання тестового сценарію. Емпіричні заміри дали наступні результати (Рис. 3, суцільна лінія). Як видно з графіку, найчастіше для пошуку джерела однієї помилки виконання тесту використовується приблизно 10 хвилин. Для великих тестових рішень такі значення можуть бути значно покращені, що призведе до загального зменшення часу на пошук помилок при налагодженні.

Побудова нової системи звітів тестових сценаріїв

Вже вказано, що початковий аналіз коду автоматизованого тестування можна покращити, оскільки це одна з основних операцій, як при налагодженні тестових сценаріїв, так і при їх запуску в режимі тестування. Дані задачі є частими при розробці рішень автоматизованого тестування [2].

В першу чергу, покращення читабельності звітів можливе за рахунок повсюдного логування інформації про перебіг тестів. Однак, постійний виклик одних і тих же функцій є кроком на шляху до дублювання коду, що не може покращувати кінцеву якість рішення.

В таких випадках, логічним рішенням є використання аспектно-орієнтованого програмування для імплементації компонентів інформаційної технології, призначених для написання звітів.

Використання аспектно-орієнтованого програмування для покращення читабельності звітів

Аспектно-орієнтоване програмування (АОП) – парадигма програмування, заснована на ідеї поділу функціональності для поліпшення розбиття програми на модулі [3].

Методологія АОП була запропонована групою інженерів дослідницького центру Xerox PARC під керівництвом Грегора Кічалеса (Gregor Kiczales).

Ведення логів і обробка виключень – типові приклади наскрізної функціональності. Інші приклади: трасування; аутентифікація та перевірка прав доступу; контрактне програмування (зокрема , перевірка перед та пост умов). Для програми, написаної в парадигмі ООП, будь-яка функціональність, за якою не була проведена декомпозиція, є наскрізною.

Однак, як стверджують деякі автори [4] , АОП може успішно застосовуватися і для вирішення завдань захисту, багатопоточності, управління трансакціями і багатьох інших.

Всі мови АОП надають засоби для виділення наскрізної функціональності в окрему сутність. Так як AspectJ є родоначальником цього напрямку, використовувані в цьому розширенні концепції поширилися на більшість мов АОП.

Основні поняття АОП:

Аспект (англ. aspect) – модуль або клас, який реалізує наскрізну функціональність. Аспект змінює поведінку решти коду, застосовуючи пораду в точках з'єднання, визначених деякими зрізом.

Порада (англ. advice) – засіб оформлення коду, який повинен бути викликаний з точки з'єднання. Порада може бути виконана до, після або замість точки з'єднання.

Точка з'єднання (англ. join point) – точка в виконуваній програмі, де слід застосувати пораду. Багато реалізації АОП дозволяють використовувати виклики методів і звернення до полів об'єкта в якості точок з'єднання.

Зріз (англ. pointcut) – набір точок з'єднання. Зріз визначає, чи підходить дана точка з'єднання до даної поради. Найзручніші реалізації АОП використовують для визначення зрізів синтаксис основної мови (наприклад, у AspectJ застосовуються Java-сигнатури) і дозволяють їх повторне використання за допомогою перейменування і комбінування.

Впровадження (англ. introduction, введення) – зміна структури класу та / або зміна ієрархії успадкування для додавання функціональності аспекту в чужорідний код. Зазвичай реалізується за допомогою деякого метаоб'єктного протоколу (англ. metaobject protocol, MOP).

Наприклад, нам потрібно огорнути конкретний метод блоками перехвату помилок (try/catch), або внести запис в систему логування/звітування про вхід/вихід з конкретного методу. В рамках розробки рішення автоматизованих тестів, використання аспектів дозволяє поліпшити якість коду, а також робить його більш читабельним і зрозумілим. Для власних потреб було використано PostSharp фреймворк [5]. Цей продукт має безкоштовну Starter Edition ліцензію, яка дозволяє комерційне використання продукту.

Формально, кожен бізнес метод в такому випадку можна розкласти наступним чином:

$$LOC_{BM} = LOC_{BM(L)} + \sum FM_A \quad (2)$$

де $LOC_{BM(L)}$ – логіка бізнес методу, яка безпосередньо належить до виконання специфічних логічних операцій з користувацьким інтерфейсом,

FM_A – аспект, який розроблений в рамках фреймворку, і логіка якого відповідно накладена на бізнес метод.

Оскільки $FM_A \subseteq FM$ і відповідно FM є константою, то твердженням (2) можна знехтувати і вважати завжди що $LOC_{BM} = LOC_{BM(L)}$.

На Рис. 2 зображено взаємодію рішення автоматизованого тестування та системи, яка тестується. Зафарбовані блоки – імплементація аспектів, відповідальних за передачу інформації в звіт про виконання тесту.

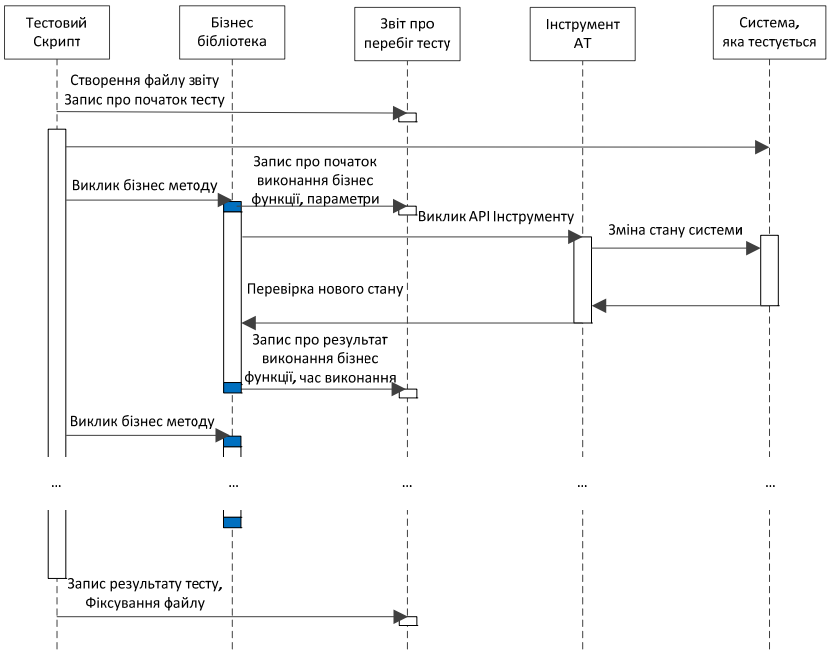


Рис. 2. Взаємодія рішення автоматизованого тестування, інструменту та системи, яка тестується

Оцінка часу на аналіз звітів запуску тестових сценаріїв

Час на аналіз звітів для виявлення помилок має важливе значення як при розробці, так і при підтримці рішень автоматизованого тестування. Використання описаних підходів дозволило зробити нові заміри часу на знаходження джерела помилок в тестових сценаріях. Результати наведено на Рис. 3. Експеримент проводився на основі замірів аналізу 20-и різних тестових скриптів, з різною кількістю помилок в кожному з випадків.

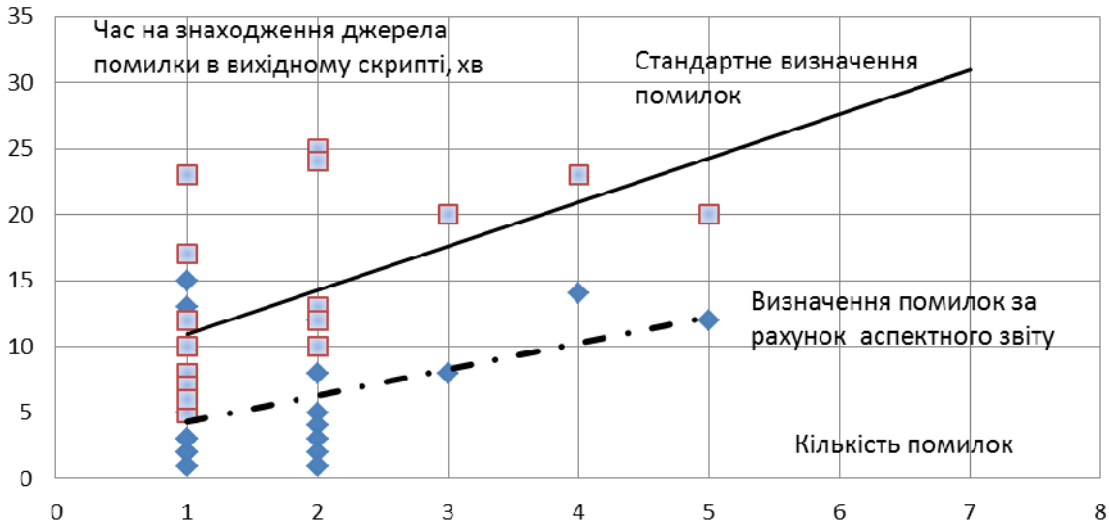


Рис. 3. Тренд витрат часу при пошуку помилок в звітах тестів, за моделлю без аспектів (суцільна лінія), та з аспектами (штрих-пунктир)

Як видно з емпірично побудованих графіків, час на пошук помилки при різній кількості помилок в рішенні автоматизованого тестування зменшився в середньому вдвічі. Звідси, справедливим стає наступне твердження з приводу показника покращення якості *ImFAn* при оцінці часу *tFan* при різних процесах визначення помилок в тестових сценаріях P_1 та P_2:

$$ImFAn = \frac{tFAn(P_1)}{tFAn(P_2)} \approx 2. \quad (3)$$

Важливо відмітити, що в зв'язку з (2), у нас не відбувається будь-якої зміни коду самого рішення автоматизованого тестування, тобто показники $LOC_{BM(i)}$ залишаються сталими.

Висновки

В роботі описано техніку створення звітів про хід тестових сценаріїв, побудованих за рахунок аспектно-орієнтованого програмування. В якості основи використовуються бізнес-методи (також відомі як ключові слова в автоматизації тестування користувацьких інтерфейсів), які огортаються аспектами на основі конкретних технологій. Аспекти дозволяють створювати комплексні звіти про виконання бізнес-методів, не змінюючи код самих методів і залишаючи там специфікацію лише бізнес логіки.

Результати були розраховані на основі двох різних процесів імплементації автоматичних тестів. Було виміряно час, необхідний для знаходження джерела помилки з аспектно-побудованими звітами та без них. Використання аспектно-побудованих звітів дало змогу залишити незмінним кількість програмного коду бізнес методів, і зменшити час, який витрачається на аналіз звітів, вдвічі.

Література

1. Котляров В.П. Основы тестирования программного обеспечения / В.П. Котляров, Т.В. Коликова– 2006 – 248 с.

2. Дивак М.П. Програмне забезпечення для тестування графічного користувацького інтерфейсу (GUI) / М.П. Дивак, В.В Чича, Л.С. Оліярник // Вісник Хмельницького національного університету. – 2012. – № 2. – С. 160–164.

3. Groves M.D. AOP in .NET: Practical Aspect-Oriented Programming / Groves M.D., P. Haack. – Manning. – 2013. – 296 p.

4. Gradecki J.D. Mastering Aspect J: Aspect-Oriented Programming in Java / Gradecki J. D., Lesiecki N. – Wiley. – 2003. – 456 p.

5. PostSharp AOP [Electronic resource] // Postsharp AOP tool, 2013. – Mode of access : http://www.postsharp.net/ – Title from the screen.

References

1. Kotlyarov V. P. Osnovy testyrovanyya prohrammnoho obespechenyya/ Kotlyarov V. P., Kolykova T. V. – 2006 – 248s.

2. Dyvak M.P. Prohramne zabezpechennya dlya testuvannya hrafichnoho korystuvats'koho interfeysu (GUI) / M.P. Dyvak, V.V Chycha, L.S. Oliyarnyk // - Visnyk Khmel'nyts'koho natsional'noho universytetu. - #2. - 2012. – c.160 - 164.

3. Groves M.D. AOP in .NET: Practical Aspect-Oriented Programming / Groves M.D., P. Haack. - Manning. - 2013. - 296 p.

4. Gradecki J. D. Mastering AspectJ: Aspect-Oriented Programming in Java / Gradecki J. D., Lesiecki N. - Wiley. - 2003. - 456p.

5. PostSharp AOP [Electronic resource] // Postsharp AOP tool, 2013. – Mode of access: http://www.postsharp.net/ – Title from the screen.

Рецензія/Peer review : 19.5.2013 р. Надрукована/Printed :6.2.2014 р.
Рецензент: д.т.н., проф. кафедри АІВТ ВНТУ Бісікало О.В.