УДК 519.72

doi: 10.26906/SUNZ.2019.2.123

І.О. Мартінкус, К.А. Нагорний, М.В. Ткачук

Харківський національний університет імені В. Н. Каразіна, Харків, Україна

МЕТРИКИ ТА ЗАСОБИ ДЛЯ ДОСЛІДЖЕННЯ ПРОЯВІВ НАСКРІЗНОЇ ФУНКЦІОНАЛЬНОСТІ В УСПАДКОВАНИХ ПРОГРАМНИХ СИСТЕМАХ

Предметом вивчення в статті є метрики та засоби для дослідження впливу наскрізної функціональності в успадкованих програмних системах. **Метою** є підвищення ефективності використання пост об'єктно-орієнтованих технології (ПООТ) у процесі супроводу успадкованих програмних систем (УПС). **Завдання:** розглянути проблему наскрізної функціональності (НФ) при розробці та супроводу УПС, можливі типи та класифікації НФ, запропонувати метрики для дослідження негативного впливу НФ, розробити процедуру класифікації НФ, програмний засіб для реалізації запропонованого підходу, та проведення експериментальних досліджень. Використовуваними **методами** є: кількісні метрики якості програмного забезпечення, об'єктно-орієнтовані та пост об'єктно-орієнтовані методи аналізу та синтезу програмного забезпечення. Отримані такі **результати:** досліджено особливості негативного впливу НФ на супровід УПС, запропоновано множину метрик її вимірювання (питома вага НФ в вихідному коді цільової УПС, розповсюдження НФ серед програмних компонентів УПС, ступінь розсіювання НФ у компонентах УПС). Розроблено процедуру класифікації НФ та запропоновано програмний засіб для реалізації оцінки негативного впливу НФ на супроводі УПС. **Висновки.** Проведені експериментальні дослідження для тестових УПС показали, що найменший ступінь присутності НФ після модифікації структури УПС на основі ПООТ забезпечує застосування аспектно-орієнтованого підходу. Напрямком подальших досліджень є розробка методики та проведення експериментів з метою оцінки впливу застосування ПООТ на рівень присутності дефектів у програмному коді УПС.

Ключові слова: наскрізна функціональність, пост об'єктно-орієнтовані технології, метрики, успадкована програмна система.

Вступ

Постановка проблеми у загальному вигляді.

Останні тенденції в області розробки та супроводу сучасних ПС, зокрема, такі як їх постійно зростаючі структурна складність та інтенсивність внесення змін у вимоги до програмних систем, привели до того, що у практиці застосування об'єктно - орієнтованого програмування (ООП) проявилися певні кризові явища. Одне із них отримало назву феномену наскрізної функціональності (crosscutting functionality) – НФ, суть якої полягає в неможливості виділити програмний код НФ в окремий програмний клас, натомість код НФ є розсіяним з-поміж класів, в яких реалізовані інші вимоги до програмних систем (ПС). Для подолання негативних наслідків НФ були запропоновані нові технології розробки програмного забезпечення, що розширюють можливості застосування ООП за рахунок вбудованих механізмів, які дозволяють ізолювати програмний код НФ в окремому програмному модулі, та потім у не інвазійний спосіб отримати нову конфігурацію ПС. Ці нові підходи до створення ПС можуть бути узагальнено визначені терміном пост об'єктно-орієнтовані технології (ПООТ)

В роботі [1] вирішувалася науково-практична задача розробки моделей та інформаційних технологій для визначення ефективності використання пост об'єктно-орієнтованих технологій у процесі супроводу успадкованих програмних систем (УПС). В рамках цієї роботи досліджувалася проблема впливу наскрізної функціональності на супровід УПС.

Мета статті – дослідження негативного впливу НФ на успадковані програмні системи за допомогою метрик, та розробка відповідної процедури класифікації НФ та програмного засобу для реалізації запропонованого підходу

Аналіз останніх досліджень і публікацій. На сьогоднішній день існує багато досліджень, присвячених проблемі НФ взагалі, та варіативності її проявів у вихідному коді ПС зокрема. В літературі досліджуються властивості НФ, наводяться різні варіанти її типізації відповідно до цих властивостей та пропонуються відповідні засоби реакції на ті чи інші типи НФ[2, 10, 11]. Крім того в роботах [12, 13, 14] запропоновано варіант розділення НВ на сорти, яких усього налічується 13.

В цій роботі пропонуються метрики та засоби для дослідження впливу НФ на успадковані програмні системи. Для цього в першому розділі досліджується поняття НФ ті підходів щодо його класифікації. В другому розділі розробляється множина метрик для вимірювання негативного впливу НФ. В третьому розділі пропонується процедура та програмний засіб для дослідження НФ та аналіз експериментальних результатів ПС. В четвртому розділі визначаються задачі для подальших досліджень.

Основна частина

1. Наскрізна функціональність та її (негативний) вплив на процес розробки програмних систем

Наскрізна функціональність (НФ) [2] – це цілісний, на рівні опису вимог, запит кінцевого користувача до функціональності ПС, який розсіюється (crosscut) на рівні розробки ПЗ. Тобто, це така частина бізнес-логіки УПС, яка не може бути локалізована в окремий програмний модуль у вигляді вихідного коду, але залишається самостійною у вигляді вимоги до цієї УПС.

Найпростіший підхід до визначення типу НФ – це підхід, що орієнтується на природу НФ: гомогенна або гетерогенна НФ [10]. В дослідженні [11] пропонується розділення НФ на відповідні шаблони

(їх 13) які розбиті на 4 категорії: плоскі шаблони НФ (flat crosscutting patterns), шаблони наслідування (inheritance-wise concerns), «шаблони зв'язку» (communicative concerns) та інші шаблони (miscellaneous), що не увійшли до перших трьох груп. В рамках цього дослідження запропоновані стратегії щодо виявлення таких шаблонів НФ та даються рекомендації, яким чином можливо усунути реалізації НФ, що віднесені до конкретного шаблону за допомогою АОП.

У роботах [12–14] запропоновано розділяти НФ на сорти, всього таких сортів представлено 13. Кожен сорт НФ характеризується п'ятьма атрибутами, що відображають його «назву» (name), «намір» (intent), «успадковану реалізацію» (legacy implementation), «бажану реалізацію» (desired implementation), «варіанти» (instances), наприклад:

– *назва* (сорту НФ) – «Прошарок переспрямування»(Redirection layer);

– *намір* (опис) – НФ визначає інтерфейсний прошарок для об'єкта та перенаправляє виклики до цього об'єкта;

– *успадкована реалізація* (ООП) – визначення проміжного прошарку (декоратор або адаптер), та додання методів до цього прошарку, що перенаправляють їх виклики;

– *бажана реалізація* (АОП) – зріз (pointcut) та порада заміщення (around advice);

– *варіанти* – шаблон «Декоратор», шаблон «Адаптер» [15].

Для кожного сорту НФ надається порада щодо усунення її за допомогою АОП, тобто в рамках цих робіт, бажана реалізація є аспектно-орієнтована реалізація.

2. Вимірювання негативного впливу наскрізної функціональності

2.1. Питома вага НФ в вихідному коді цільової УПС. Зазначені вище способи класифікації НФ: шаблони, сорти з точки зору команди супроводу УПС можуть давати загальне розуміння присутності НФ, відноситься до деякого класу, та деякі можливі стратегії її видалення. Але ці класи не дають кількісного уявлення про складність ураження системи НФ. З огляду на це в рамках даного дослідження пропонується додатковий клас, що має розрізняти різні реалізації НФ за ступенем поширення її в УПС: «слабке ураження», «ураження середньої тяжкості», «сильне ураження».

Ступінь ураження УПС можна виразити через коефіцієнт питомої ваги НФ (Crosscutting Functionality Ratio – CFR), що запропоновано у [3]:

$$CFR = \frac{C_{cf}}{N}, \quad \text{CFR} \in [0;1] \qquad (1)$$

де CFR – коефіцієнт питомої ваги НФ в УПС,

C_{cf} – кількість класів базової бізнес-логіки УПС, де присутній вихідний код НФ,

N – загальна кількість класів в УПС.

Якщо CFR = 1, це свідчить про те, що НФ уражені всі класи УПС. Якщо CFR = 0, присутності НФ в системі не виявлено.

При цьому граничні показники коефіцієнта питомої ваги НФ, пропонується обирати за принципом звичних знань для нормалізованих показників, згідно із [4], що наведені у табл.1:

Таблиця 1 – **Граничні значення для нормалізованих показників**

Граничне значення	Семантичне значення
0.25	Одна чверть
0.33	Одна третина
0.5	Половина
0.66	Дві третини
0.75	Три чверті

Таким чином, граничні значення діапазонів ураження УПС наскрізною функціональністю будуть наступні:

– якщо значення метрики $0 \le CFR < 0.33$ – то дана УПС має слабке ураження, тобто НФ присутня менше ніж у третині її класів;

– якщо значення метрики потрапляє в діапазон $0.33 \le CFR < 0.75$ – то дана УПС має ураження середньої тяжкості, тобто НФ присутня у трьох чвертях її класів;

– якщо значення метрики $0.75 \le CFR \le 1$ – то дана УПС має сильне ураження, тобто НФ присутня майже в усіх її класах;

Рис. 1 в термінах діаграми класів, нотації UML 2.0 [5] відображає існуючі (запропоновані в інших роботах) класи НФ, та з огляду на їх недоліки до існуючої класифікації додано новий клас за ступенем ураження УПС наскрізною функціональністю [3].

Рис. 1. Класифікація наскрізної функціональності

2.2. Розповсюдження НФ серед програмних компонентів УПС.

В [6] запропоновано декілька метрик, та також присутнє порівняння їх із уже відомими. Одна із запропонованих у зазначеній роботі метрик є ступеня розсіювання (Degree of Scattering) *DS*, що вимірює ступінь розсіювання НФ серед класів УПС. Вона показує на скільки таке розсіювання рівномірне:

$$DS = 1 - \frac{N}{N-1}\sum_{n=2}^{N}\left(\frac{LOC(cf,n)}{LOC(cf)} - \frac{1}{N}\right)^2, \; DS \in [0;1], \qquad (2)$$

де N – кількість класів бізнес-логіки, $N > 1$,

LOC(cf,n) – кількість строк коду у класі *n*, що відносяться до НФ,

LOC(cf) – загальна кількість строк коду, що відноситься до НФ.

Якщо *DS* = 1, це свідчить про те, що НФ рівномірно розсіяна серед уражених їй класів УПС. Якщо ж *DS* = 0, то така НФ повністю ізольована. Ця метрика є досить придатною для подальших розрахунків, за виключенням одного недоліку – вона не відображає кількість класів, що уражені в УПС. Тобто не є зрозумілим, у якій саме частині УПС присутня НФ, яка наприклад рівномірно розсіяна.

2.3. Ступінь розсіювання НФ у компонентах УПС. Беручи до уваги різні аспекти НФ, що можуть бути виміряні за допомогою зазначених вище метрик, в рамках даного дослідження пропонується поєднати деякі із них (1) та (2). Таким чином, комбінована метрика буде показувати на скільки рівномірно розсіяна (scattered) НФ серед уражених їй класів УПС [7]:

$$CFL = DS \cdot CFR,\ CFL \in [0;1], \qquad (3)$$

де *CFL* – ступінь присутності НФ (Crosscutting Functionality Level),

DS – ступінь розсіювання НФ (Degree of Scattering), вираз (2),

CFR – коефіцієнт питомої ваги НФ (Crosscutting Functionality Ratio), вираз (1).

Якщо *CFL* = 1, це свідчить про те, що НФ рівномірно розсіяна серед усіх класів УПС, тобто уражені усі класи. Якщо *CFL* = 0, то НФ повністю ізольована в одному класі.

3. Процедура та програмний засіб для дослідження НФ та аналіз експериментальних результатів УПС

Для класифікації та дослідження наскрізної функціональності слід обрати один з методів або алгоритмів класифікації. До найбільш відомих методів класифікації відносяться: дерево рішень, метод Байєса та метод SVM(support vector machine) [9]. У якості цільового методу класифікації було обрано метод дерева рішень. Процедура визначення класу НФ у нотації IDEF0 зображена на рис. 2.

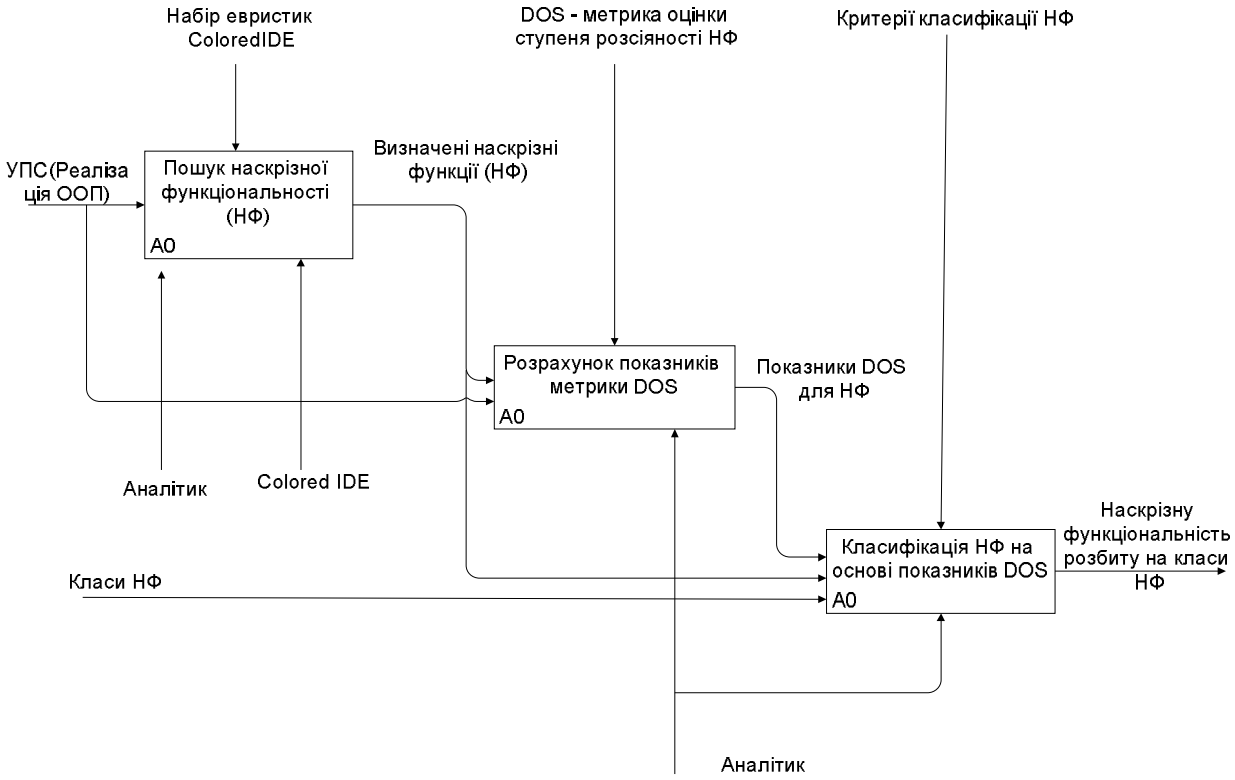

Рис. 2. Процедура класифікації НФ

Для автоматизації застосування запропонованих метрик визначення впливу НФ було розроблено програмний засіб, який реалізує запропоновану процедуру, фрагмент діаграми класів якого представлений на рис 3.

На основі обробки повного об'єму експериментальних даних, приклад розрахунків яких був приведений у попередньому параграфі, були отримані остаточні результати застосування запропонованої інформаційної технології визначення ефективності використання ПООТ в процесі супроводу ПС [7,8]. Для цього були обрані чотири УПС відповідного класу та сфери застосування. Початкові характеристики цих систем для першої фази експерименту наведені в табл. 2.

Таким чином, УПС №1 відповідно до загальної процедури запропонованої інформаційної технології віднесена до першого (I) типу ПС, вона є стабільною з точки зору стану запитів на модифікацію, тобто стан її ФВ визначається як «низький», а ранг ФВ представляється найнижчим «нейтральним» пріоритетом, та «низькою» функціональною складністю.

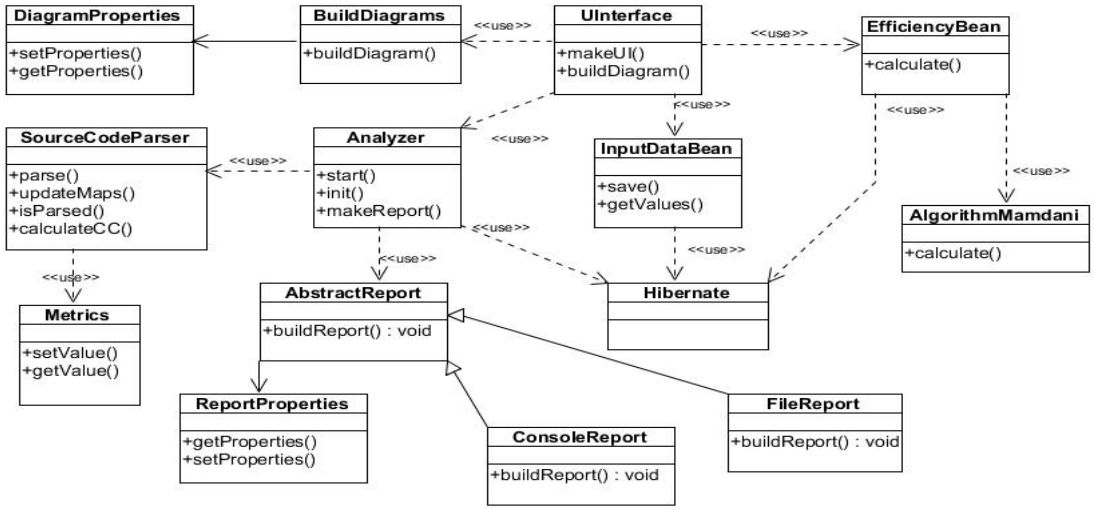

Рис. 3. Фрагмент діаграми класів розробленого програмного засобу

Таблиця 2 – **Вхідні характеристики досліджуваних УПС**

№ УПС	Кількість java- класів що реалізують бізнес-логіку	Ранг ФВ	Структурна складність	Тип УПС
1	58	Низький	Проста	I
2	97	Високий	Проста	II
3	119	Низький	Складна	III
4	334	Високий	Складна	IV

Також УПС №1 має «просту» структурну складність, що вказує на те що показники по всім метрикам складності програмного коду не перевищують їх середнього граничного значення для більшості структурних елементів (методів, класів).

УПС №2 віднесена до другого (II) типу ПС, вона є «нестабільною» з точки зору стану запитів на модифікацію, тому що має «високий» ранг ФВ, що характеризується здебільшого максимальним «невідкладним» пріоритетом виконання вимог та максимальною «високою» функціональною складністю вимог. У той самий час, УПС №2 має «просту» структурну складність, що як у випадку із УПС №1, вказує на те, що показники метрик програмного коду не перевищують середні граничні умови для більшості структурних елементів цієї програмної системи.

УПС №3 віднесена до третього (III) типу ПС, вона є «стабільною» з точки зору стану запитів на модифікацію, т.я. має «низький» ранг ФВ, але вона має «високу» структурну складність, що вказує на те, що показники метрик програмного коду перевищують середні граничні умови, тобто належать до «складного» сегменту, для більшості структурних елементів цієї програмної системи. Це, в свою чергу, вказує на те, що УПС містить недосконалі проектні рішення та зумовлює ускладненість її супроводу.

УПС №4 віднесена до четвертого (IV), найскладнішого типу ПС, вона є «не стабільною» з точки зору стану запитів на модифікацію, тому що має «високий» ранг ФВ. УПС №4 має «високу» структурну складність, що вказує на те, що показники метрик програмного коду перевищують середні граничні умови, тобто належать до «складного» сегменту УПС з токи зору більшості структурних елементів цієї програмної системи. Відповідно, ця УПС містить недосконалі проектні рішення, які з урахуванням функціональної складності і невідкладності реалізації нових ФВ дуже сильно ускладнюють супровід такої УПС.

Для зазначених УПС, відповідно до четвертого етапу методики проведення експерименту (представленої у попередньому параграфі), був визначений коефіцієнт питимої ваги НФ (див. Табл. 3). Слід зазначити, що для розрахунку показників НФ, до розгляду приймаються незалежні від каркасу (framework) класи, а саме, java-класи бізнес–логіки.

Згідно із даними у Табл. 3 та відповідно до запропонованої класифікації НФ, всі розглянуті вище УПС мають «ураження» НФ середньої тяжкості.

Таблиця 3 – **Коефіцієнт питомої ваги в досліджуваних УПС**

№ УПС	Загальна кількість класів бізнес-логіки	Кількість «уражених» класів бізнес-логіки	Кількість класів бізнес-логіки «вільних» від НФ	Показник коефіцієнта питомої ваги НФ
1	58	21	37	36,21%
2	97	59	51	60,82%
3	119	76	43	63,87%
4	334	156	178	46,71%

Для УПС III та IV, такі ураження є досить суттєвими, так як НФ має властивості «заплутуватись» та «перемішуватись» (див. п. 1.2 першого розділу роботи), що у поєднанні із складною структурою цих УПС, додатково ускладнюють процес супроводу системи.

Після відповідних ПООТ - модифікацій вихідного коду усіх УПС, розрахований ступінь присутності НФ (*CFL* – Crosscutting Functionality Level) представлено графіками на рис. 4. Із результатів розрахунку *CFL* видно, що найменший ступінь присутності НФ після ПООТ - модифікації структури УПС забезпечує застосування аспектно-орієнтованого підходу (АОП), який є близьким до 0,05%, тобто використання АОП дозволяє ізолювати НФ в найменшій кількості додаткових АОП - модулів.

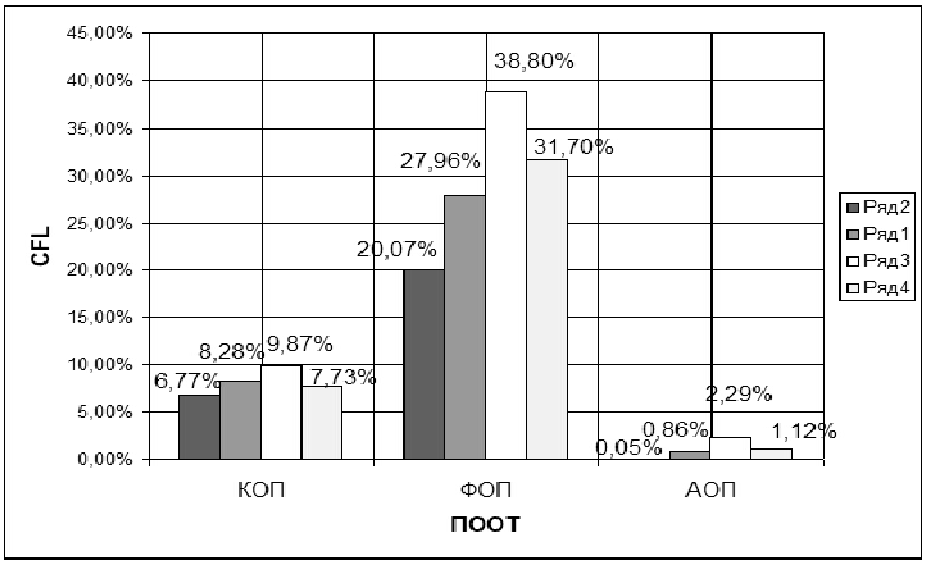

Рис. 4. Показник ступеню присутності НФ (CFL)

Висновки і напрямки подальших досліджень

В роботі було розглянуто особливості негативного впливу НФ на супровід успадкованих програмних систем та запропоновано множину метрик його вимірювання, а саме для оцінки:

1) питомої ваги НФ в вихідному коді цільової УПС;

2) ступеня розповсюдження НФ серед програмних компонентів УПС;

3) ступеня розсіювання НФ у компонентах УПС.

Розроблено процедуру класифікації НФ та запропоновано програмний засіб для реалізації оцінки негативного впливу НФ на супроводі УПС. Проведені експериментальні дослідження для тестових УПС показали що найменший ступінь присутності НФ після ПООТ - модифікації структури УПС забезпечує застосування аспектно-орієнтованого підходу.

Подальші дослідження мають бути направлені на розробку методики та проведення експериментів з метою оцінки впливу застосування ПООТ на рівень присутності дефектів у програмному коді успадкованої програмної системи.

СПИСОК ЛІТЕРАТУРИ

1. Нагорний, К.А. Моделі та інструментальні засоби супроводу програмних систем на основі пост об'єктно-орієнтованих технологій: дис. … канд. техн. наук: 05.13.06 / Національний технічний університет «Харківський політехнічний інститут», Харків, Україна, 2016. 182 с.
2. Kaur, A., Johari, K. Identification of Crosscutting Concerns: A Survey. / Kaur A., Johari K. // International Journal of Engineering Science and Technology. – 2009. – vol. 1(3). – pp. 166-172.
3. Ткачук, Н.В., Нагорный, К.А. Об одном подходе к оценке эффективности применения пост объектно-ориентированных технологий при сопровождении программных систем / Ткачук Н.В., Нагорный К.А. // Проблемы программирования. – К.: НАН України. – 2010. – № 2-3 (спец. выпуск). – с. 252 - 260.
4. Lanza, M., Marinescu, R. Object-Oriented Metrics in Practice: Using Software Metrics to Characterize, Evaluate, and Improve the Design of Object-Oriented Systems/ Lanza, M., Marinescu, R. – Germany: Springer-Verlag, 2006. – 207 p.
5. Unified Modeling Language (UML) Resource Page //Офіційний вєб-ресурс уніфікованої мови моделювання UML, що розробляється консорціумом OMG – Object Management Group [Електронний ресурс] – Режим доступу: http://www.uml.org
6. Eaddy, M., Aho, A., Murphy, G. C. Identifying, Assigning, and Quantifying Crosscutting Concerns / Eaddy M., Aho A., Murphy G. C. // Proceedings of the ACoM '07: the First International Workshop on Assessment of Contemporary Modularization Techniques – 2007.
7. Нагорный К. А. Разработка и применение методики оценки эффективности пост объектно-ориентированных технологий. / Нагорный К.А. // Східно-Європейський журнал передових технологій. – 2013. – № 3/10 (63). – с. 21–25.
8. Нагорний К. А. Експериментальне дослідження ефективності пост об'єктна-орієнтованих технологій розробки програмних систем. / К. А. Нагорний, М. В. Ткачук, Ю. М. Жадан // Матеріали XXII міжнародної науково-практичної

конференції «Інформаційні технології: наука, техніка, технологія, освіта, здоров'я», НТУ «ХПІ», Харків, 15-17 жовтня 2014 р. – с. 15.
9. Черезов Д. С. Обзор основных методов классификации и кластеризации данных / Черезов Д. С., Тюкачев Н. А., Воронежский государственный университет, 2009.
10. Apel, S., Batory, D., Rosenmüller, M. On the Structure of Crosscutting Concerns: Using Aspects or Collaborations? / Apel S., Batory D., Rosenmüller M. // Proceedings of the AOPLE'06: the 1st Workshop on Aspect-Oriented Product Line Engineering co-located with the GPCE'06: the 5th International Conference on Generative Programming and Component Engineering. – Portland, OR, USA. – Oct. 2006.
11. Figueiredo, E. Concern-Oriented Heuristic Assessment of Design Stability. Submitted for the Degree of PhD. / Figueiredo E. – Lancaster University, Lancaster, UK, October 2009. – 237 p.
12. Marin, M., Moonen, L., Deursen, A.V. A Classification of Crosscutting Concerns. / Marin M., Moonen L., Deursen A.V. // Proceedings of the ICSM'2005: the 21st IEEE International Conference on Software Maintenance, IEEE Computer Society. – 2005. – pp. 673-676.
13. Marin, M., Moonen, L., Deursen, A.V. An Approach to Aspect Refactoring Based on Crosscutting Concerns Types. / Marin M., Moonen L., Deursen A.V. // Proceedings of the MACS'2005: Workshop on Modeling and Analysis of Concerns in Software, ACM. – St. Louis, Missouri. – 2005. – pp. 1-5.
14. Marin, M. Formalizing typical crosscutting concerns. / Marin M. // Proceedings of the 21st IEEE International Conference on Software Maintenance, IEEE Computer Society. – 2005. – pp 673-676.
15. Gamma, E. et al. Design Patterns: Elements of Reusable Object-Oriented Software / Gamma E., Helm R., Johnson R., Vlissides J. – Addison-Wesley, 1994. – 395 p.

Рецензент: д-р техн. наук, проф. В. О. Філатов,
Харківський національний університет радіоелектроніки, Харків
Received (Надійшла) 20.02.2019
Accepted for publication (Прийнята до друку) 27.03.2019

Метрики и средства для исследования проявлений сквозной функциональности в унаследованных программных системах

И. О. Мартинкус, К. А. Нагорний, Н. В. Ткачук

Предметом изучения в статье являются метрики и средства для исследования влияния сквозной функциональности в унаследованных программных системах. **Целью** является повышение эффективности использования пост объектно-ориентированных технологий (ПООТ) в процессе сопровождения унаследованных программных систем (УПС). **Задачи:** рассмотреть проблему сквозной функциональности (СФ) при разработке и сопровождению УПС, возможные типы и классификации СФ, предложить метрики для исследования негативного влияния СФ, разработать процедуру классификации СФ, программное средство для реализации предлагаемого подхода и проведение экспериментальных исследований. Используемыми **методами** являются количественные метрики качества программного обеспечения, объектно-ориентированные и пост объектно-ориентированные методы анализа и синтеза программного обеспечения. Получены следующие **результаты**. Исследованы особенности негативного влияния СФ на сопровождение УПС, предложено множество метрик его измерения (удельный вес СФ в исходном коде целевой УПС, распространение СФ среди программных компонентов УПС, степень рассеивания СФ в компонентах УПС). Разработана процедура классификации СФ и предложено программное средство для реализации оценки негативного воздействия СФ на сопровождение УПС. **Выводы.** Проведенные экспериментальные исследования для тестовых УПС показали, что наименьшая степень присутствия СФ после модификации структуры УПС на основе ПООТ обеспечивает применение аспектно-ориентированного подхода. Направлением дальнейших исследований является разработка методики и проведение экспериментов с целью оценки влияния применения ПООТ на уровень присутствия дефектов в программном коде УПС.

Ключевые слова: сквозная функциональность, пост объектно-ориентированные технологии, метрики, унаследованная программная система.

Metrics and tools for exploration of the crosscutting functionality in legacy software systems

I. Martinkus, K. Nagornyi, M. Tkachuk

The **subject matter** of the article are metrics and tools for studying the influence of crosscutting functionality in legacy software systems (LSS). The **goal** is to increase the efficiency of post-object-oriented technologies (POOT) usage of in the process of maintainability legacy software systems. The **tasks** are: consider the problem of crosscutting functionality (CF) in the development and maintenance of LSS, possible types and classification of CF, purpose metrics for studying the negative impact of CF, develop a classification procedure for CF, a software tool for implementing the proposed approach and conducting experimental research. The **methods** used are quantitative software quality metrics, object-oriented and post-object-oriented methods for analyzing and synthesizing software. The following **results** were obtained. The features of the negative impact of CF on the maintenance of the LSS are investigated, a set of metrics for CF-measurement (the proportion of the CF in the source code of the target LSS, the distribution of the CF among the software components of the LSS, the degree of dispersion of the CF in the LSS components) are proposed. A procedure for CF-classifying has been developed and a software tool has been proposed for realizing an assessment of the negative impact of CF on the maintenance of LSS. **Conclusions.** Experimental LSS test cases showed that the smallest degree of presence of the CF after modifying the structure of the LSS on the basis of POOT ensures the use of an aspect-oriented approach. The direction of further research is to develop a methodology and conduct experiments to assess the impact of the application of POOT on the level of presence of defects in the LSS code.

Keywords: crosscutting functionality, post object-oriented technologies, metrics, legacy software system.