УДК 004.41

***Д. Л. ОРЛОВСЬКИЙ, І. В. ЛЮТЕНКО, С. О. ЛІЛІКОВИЧ***

# ПІДХІД ТА ЗАСОБИ ВИМІРЮВАННЯ СУБ'ЄКТНОЇ ДИВЕРСНОСТІ В УМОВАХ БАГАТОВЕРСІЙНОЇ РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Розглядається можливість забезпечення диверсифікації програмних проектів за рахунок підвищення рівня суб'єктної диверсності. Висунута гіпотеза про можливість оцінки різноманітності розробників через набір оцінок, що відбивають підхід кандидата до розробки, тип його особистості. Запропоновано підхід для обчислення рівня суб'єктної диверсності, формалізовано вимоги до програмного забезпечення запропонованого підходу, розглянуто існуючі програмні рішення.

**Ключові слова:** диверсність, багатоверсійна розробка, оцінки, якість програмного забезпечення, метод, вимоги.

Рассматривается возможность обеспечения диверсификации программных проектов за счёт повышения уровня субъектной диверсности. Выдвинута гипотеза о возможности оценки разнообразия разработчиков через набор оценок, отражающих подход кандидата к разработке, тип его личности. Предложен подход для вычисления уровня субъектной диверсности, формализованы требования к программному обеспечению предложенного подхода, рассмотрены существующие программные решения.

**Ключевые слова:** диверсность, многоверсионная разработка, оценки, качество программного обеспечения, метод, требования.

The possibility of diversifying software projects by improving the level of subject diversity is reviewed. Improvement of subject diversity level leads to software quality improvement, so a hypothesis on the possibility of assessing the diversity of developers through a set of assessments was made. The assessment set includes candidate's attitude to the software development process and candidate's individual type. Thus, the assessment set does not contain project-specific parameters, so subject diversity evaluation is project independent too. Approach for calculating the level of subject diversity was offered as well as toolkit software requirements were formalized. After all, review of existing software was made.

**Keywords:** diversely, multi-version development, evaluation, software quality, method, requirements.

**Вступ.** Розробка програмного забезпечення пов'язана з проблемами якості, вартості та надійності. Деякі програми містять мільйони рядків вихідного коду, які, як очікується, повинні правильно виконуватися в умовах, що змінюються.

Одним з найскладніших процесів при розробці ПЗ є пошук і виправлення помилок, що впливають на надійність та якість програмного продукту. Навіть якщо архітектура продукту в цілому вибрана правильно, більш локальні помилки наявні завжди: друкарські помилки, завжди помилкові або істинні умови, використання змінної до ініціалізації або інші помилки що призводять до невизначеної поведінки, тощо.

Наприклад, статичний аналізатор вихідного коду «PVS-Studio» знайшов безліч помилок в Open-Source проектах, в тому числі в Chromium, QT, Clang, Apache, MySQL, FAR Manager, Audacity. В цілому команда розробників «PVS-Studio» проаналізувала 224 популярних програмних продукти, які використовуються по всьому світу, та знайшла 9055 помилок. В середньому це складає більш ніж 40 помилок на кожний проект [1].

Разом з тим, деякий клас помилок дуже складно знайти, як за допомогою будь-якого з існуючих інструментів, так і вручну. Ці помилки можуть і призводять до зниження якості програмних продуктів. В якості прикладу такої вразливості можна навести «Heartbleed» – вразливість, що була присутня в широко використовуваній та відкритій бібліотеці OpenSSL. Уразливість проявлялася при обробці так званих heartbeat запитів від клієнтів. Ці запити використовуються для підтримки захищеного з'єднання в той час як воно не використовується, тобто клієнт та сервер час від часу обмінюються heartbeat пакетами.

Кожний з heartbeat пакетів транспортує корисні дані заявленого розміру, отже після отримання heartbeat пакету сервер виділяє стільки пам'яті під корисні дані, скільки було заявлено клієнтом. Занулення виділеної з кучі пам'яті не виконується, тому якщо сервер OpenSSL виділить пам'яті більше ніж фактично було отримано даних в heartbeat пакеті, то в цей буфер попадуть попередні значення що знаходились в кучі. Так як OpenSSL самостійно виконує керування кучею, то в буфер гарантовано попадають дані, пов'язані саме з роботою захищених протоколів передачі даних.

Після формування дефектного буферу сервер OpenSSL формує heartbeat-відповідь, де в якості тіла пакету використовуються дані з цього вищезазначеного буферу. Таким чином зловмисник, сформувавши описаним чином мережевий пакет heartbeat міг отримати частину даних з кучі OpenSSL. Незважаючи на те що вразливість здається занадто обмеженою, адже подібним чином не можна вилучити значну кількість даних, зловмисникам успішно вдавалося діставати конфіденційні дані.

Таким чином, вразливість «Heartbleed» являється наслідком відсутності перевірки границь масиву [2].

Одним з підходів до зниження кількості помилок є використання принципу диверсифікації, який полягає в розробці та інтеграції одразу декількох версій програмного продукту, або його частин [3]. Як правило за принципом диверсіфікації розроблюються критичні, гарантозданті програмні продукти [4].

Забезпечення багатоверсійності полягає в забезпеченні різноманітності виконавців, мов та нотацій, інструментів розробки.

Ціллю статті є висвітлення питання підвищення якості багатоверсійного програмного забезпечення через підвищення рівня саме суб'єктної диверсності.

*ISSN 2079-0023 (print)*
*ISSN 2410-2857 (online)*

**Роль суб'єктної диверсифікації в сучасній розробці.** Диверсність що забезпечується різноманіттям розробників прийнято називати суб'єктною диверсністю [3].

Для оцінки рівня диверсифікації програмної системи існують формалізовані підходи, які дозволяють оцінити різноманітність програмних платформ.

Використання формальних метрик програмного коду дозволяють чисельно оцінити різноманітність програмних реалізацій [5], а в разі використання ПЛІС подібна методика може бути поширена на код, який визначає принципову електричну схему апаратної платформи [4].

Зазначені підходи добре відомі та дозволяють довести наявність різноманітності апаратних і програмних платформ на етапі розробки та здачі проекту.

У той же час вимірюванню рівню різноманітності розробників приділяється менше уваги, що пов'язано з відсутністю математично доведеної залежності між складом команди розробників та рівнем диверсифікації продукту. Розробка такого апарату може дозволити оцінити рівень різноманітності розробників, що впливає на рівень диверсифікації проекту в цілому, можливо ще до початку розробки.

**Огляд запропонованого підходу.** В ході роботи авторами висунута гіпотеза про те, що різноманітність розробників можна оцінити через:

1. Формальні метрики вихідного коду ( $S$ ). На етапі підготовки команди кандидати отримують однакові тестові завдання, які необхідно реалізувати з однаковим набором інструментів. Якщо завдання досить тривіальне, то можна вважати, що будь-які відмінності в реалізації досягаються за рахунок відмінностей між виконавцями. Важливо зазначити, що якість програмного коду з цієї точки зору не має значення, адже розглядаються саме відмінності з точки зору реалізації завдання, тому для аналізу програмного коду можна використовувати метрики, які спеціально орієнтовані на виявлення відмінностей, що вносяться кандидатом та складають його персональний відбиток на реалізації типового завдання [6].
2. Професійні оцінки ( $P$ ). Різноманітність розробників можна оцінити за допомогою декількох запитань, які допоможуть оцінити ставлення кандидата до роботи. При цьому такий підхід не слід розцінювати як свідомий вибір парадигми розробки: він виражається, наприклад, тільки в тому як програміст використовує наявну кодову базу, або в тому, на які міркування спирається архітектор при проектуванні системи. Отримати подібні оцінки можна не тільки за допомогою опитування, а і за допомогою аналізу дій які виконує програміст в ході виконання роботи, при чому цей аналіз може бути автоматизований.
3. Тип особистості ( $C$ ). Особливості особистості можуть впливати на розробку невизначеним чином. Беручи до уваги ряд показників, що вказують на той чи певний тип особистості, можна спробувати врахувати цей вплив. Для визначення цих оцінок зручно використовувати існуючі визнані методики, наприклад MBTI [7].

При використанні запитань з варіантами відповіді «так» і «ні» оцінки $P$ легко можуть бути представлені у вигляді бінарної множини. Відмінності між двома множинами $P_A$ і $P_B$, які утворені кандидатами $A$ і $B$ відповідно, можуть бути оцінені за допомогою оператора $d$ :

$$d_w(P_A, P_B) = |\{x \mid x = k_i \forall P_{Ai} \neq P_{Bi}\}|.$$

Тут $k_i$ – це ваговий коефіцієнт, який визначає важливість $i$ -ї відмінності у відповідях кандидатів.

Оцінки $S$ і $C$ залежать від використовуваної методики, але як правило носять абсолютний характер і можуть бути підсумовані з оцінками $P$ для знаходження узагальненого рівня різноманітності розробників $C_1$ і $C_m$ – значення $V(C_1, C_m)$ :

$$V(C_l, C_m) = K_1 \sum_{i=0}^{j} \left[ w_i \left| s_i^{C_l} - s_i^{C_m} \right| \right] +$$
$$+ K_2 d_w(C^{C_l}, C^{C_m}) + K_3 d_w(P^{C_l}, P^{C_m}).$$

Тут $w_i$ являє ваговий коефіцієнт $i$ -ї формальної метрики програмного коду.

**Програмне забезпечення.** Для перевірки висунутої гіпотези необхідно провести опитування та ранжування респондентів згідно до наведених критеріїв. До інструменту проведення опитування висуваються наступні вимоги:

1. Можливість проводити ряд опитувань послідовно;
2. Автоматичне підбиття результатів там де це можливо, наприклад автоматичне юніт-тестування програмного коду, аналіз програмного коду на основі набору формальних метрик, розрахунок показників за методиками аналізу особистості. Оператор системи повинен мати можливість протестувати декілька наборів метрик для виявлення найбільш ефективних з них;
3. Збереження результатів, API доступу до розрахованих показників та проміжних результатів.

Автори розглядали наступні популярні продукти як такі що можуть бути використані в якості засобу для проведення необхідного опитування:

1. Google Forms;
2. Survey Monkey;
3. Survey Gizmo;
4. SurveyPlanet.

Порівняння певних, важливих в даному випадку, властивостей цих продуктів наведено в табл. 1.

Таблиця 1 – Порівняння програмних продуктів що можуть бути використані для проведення опитування

| Назва | Критерії | | | |
|---|---|---|---|---|
| | Можливість об'єднати декілька опитувань | API для автоматизації розрахунків | API доступу до даних | Ціна |
| Google Forms | Так | Є | Є | Безкоштовно |
| Survey Gizmo | Так | Немає | Вивантаження результатів в файл CSV | 300€ на рік за користувача |
| Survey Monkey | Так | SPSS | Вивантаження результатів в файл CSV | $75 на рік |
| SurveyPlanet | Так, підтримується розгалуження | Немає | Вивантаження результатів в файл CSV, JSON, Word, Excel | $20 на місяць |

Очевидно що більшість наведених програмних продуктів не відповідають висунутим функціональним вимогам. Google Forms може бути використаний, проте реалізація специфічних функцій, таких як автоматичне тестування та аналіз програмного коду що надають респонденти, уявляється складною, такою що знаходиться за рамками передбачених можливостей Google Forms, задачею. Тому авторами було прийняте рішення про розробку програмного забезпечення для перевірки висунутої гіпотези дослідження.

Авторами розроблений програмний продукт який орієнтований на збір статистичної інформації для перевірки висунутої гіпотези. Згідно з задекларованими вище критеріями $S, P, C$ цей програмний продукт збирає відповіді користувачів для подальшого аналізу. Взяти участь в опитуванні можна мережею інтернет за адресою http://cw51.lilikovych.name/apply.

**Подальші роботи.** Пропонується використовувати даний підхід для формування складу команд розробників гарантоздатного програмного забезпечення. Для цього необхідно:

1. Обрати конкретні метрики програмного коду. Наразі пропонується близько 50 різних метрик [8] таких що можуть бути використані для оцінки рівня диверсіфікації. Необхідно оцінити їх ефективність та автоматизувати їх обчислення;
2. Обрати методику визначення типу особистості або інших персональних властивостей кандидата (критерії $P, C$). Пропонується використовувати MBTI та спеціально розроблені запитання;
4. Оцінити кореляцію між критеріями $S, P, C$ для використання інших оцінок, крім формальних метрик вихідного коду;
5. Налаштувати вагові коефіцієнти $w, k, K$ або розробити множину коефіцієнтів для різних типів проектів;
6. Обрати алгоритм ранжування розробників за отриманими оцінками для подальшого формування груп.

**Висновки.** Очевидно, що запропонований метод оцінки потребує розробки або адаптації існуючих вищезазначених методів, таких як MBTI. В даний час відповідні формальні метрики програмного коду не можуть гарантувати точне визначення авторського підходу, однак навіть на нетривіальних завданнях показують успішні результати точність яких досягає 63% [6, 8].

Вибір конкретних методів визначення як типу особистості кандидата, так і вибір набору формальних метрик програмного коду залишається за рамками даної публікації, адже конкретні висновки про ефективність можна робити лише після апробації цих алгоритмів.

Як і в багатьох інших підходах цього класу, заснованих на використанні галузевого досвіду, для забезпечення правильних результатів необхідно буде налаштовувати коефіцієнти $w, k, K$. Разом із тим описаний підхід не потребує попереднього навчання та може застосовуватися ще до початку розробки проекту для якого забезпечується необхідний рівень суб'єктної диверсності.

**Список літератури**

1. An always up-to-date list of articles describing errors that we find in open source projects with PVS-Studio analyzer / . // PVS-Studio – Режим доступу: http://www.viva64.com/en/inspections. – Дата звертання: 18 листопада 2016.
2. *Z. Durumeric*. The matter of heartbleed / *Z. Durumeric, J. Kasten, D. Adrian* [et al.] // Conference on Internet Measurement Conference. – 2014. - С. 475–488.
3. *Sommerville I.* Software engineering / *I. Sommerville*. – Pearson Education, 2007. – 792 с.
4. *Скляр В. В.* Метрики оценки сложности проектов ПЛИС, реализующих алгоритмы управления технологическим оборудованием АЭС / *В. В. Скляр, В. А. Головир* // Радіоелектронні і комп'ютерні системи. – 2006. – № 2007. – С. 82–86.
5. *Харченко В. С.* Использование метрик Холстеда при оценке безопасности критического программного обеспечения / *В. Харченко, В. Скляр, А. Гордеев* [та ін.] // Радіоелектронні і комп'ютерні системи. – 2003. – № 4. – С. 145–150.
6. *Маевский Д. А.* Определение авторства программного обеспечения по исходному коду программ / *Д. Маевский, Ю. Чербаджи.* // Радіоелектронні і комп'ютерні системи. – 2014. – № 6. – С. 64–68.

7. *Hilsenroth M. J.* Comprehensive handbook of psychological assessment / *M. J. Hilsenroth, D. L. Segal, M. Hersen.* – Hoboken, N.J: John Wiley & Sons, 2004. – 688 c.
8. *Ding H.* Extraction of Java program fingerprints for software authorship identification / *H. Ding, M. H. Samadzadeh.* // Journal of Systems and Software. – 2004. – № 72. – С. 49–57.

**References (transliterated)**

1. An always up-to-date list of articles describing errors that we find in open source projects with PVS-Studio analyzer [WWW Document]. Available at: http://www.viva64.com/en/inspections/ (accessed 18.11.2016)
2. Durumeric, Z., Kasten, J., Adrian, D., Halderman, J.A., Bailey, M., Li, F., Weaver, N., Amann, J., Beekman, J., Payer, M.. [The matter of heartbleed]. *Internet Measurement Conference*. ACM, 2014, pp. 475–488. doi: 10.1145/2663716.2663755
3. Ian Sommerville. *Software Engineering*. Pearson Education Publ., 2007. 792p.
4. Sklyar, V.V., Holovyr, V.A. Metriki otsenki slozhnosti proektov PLIS, realizuyushchikh algoritmy upravleniya tekhnologicheskim oborudovaniem AES [Complexity estimation metrics for FPGA projects which implements algorithms for control of NPP's technological equipment]. *Radioelektronni i komp"yuterni systemy.* 2006, no. 7, pp. 82–86.
5. Kharchenko, V.S., Sklyar, V.V., Gordeev, A.A., Tokarev, V.I., Gerasimenko, A.D., Belyy, Yu.A. Ispol'zovanie metrik Kholsteda pri otsenke bezopasnosti kriticheskogo programmnogo obespecheniya [Halstead metrics usage for assessing the safety of critical software]. *Radioelektronni i komp"yuterni systemy.* 2003, no. 4, pp. 145–150.
6. Maevskiy, D., Cherbadzhi, Yu. Opredelenie avtorstva programmnogo obespecheniya po iskhodnomu kodu programm [Defining software authorship on the source code of programs]. *Radioelektronni i komp"yuterni systemy.* 2014, no. 6, pp. 64–68.
7. Hilsenroth M. *Comprehensive handbook of psychological assessment*. N. J, Hoboken, John Wiley & Sons Publ., 2004. 688 p.
8. Ding, H., Samadzadeh, M.H. Extraction of Java program fingerprints for software authorship identification. *Journal of Systems and Software.* 2004, vol. 72, no. 1, pp. 49–57. doi: 10.1016/S0164-1212(03)00049-9

*Надійшла (received) 28.11.2016*

*Бібліографічні описи / Библиографические описания / Bibliographic descriptions*

**Підхід та засоби вимірювання суб'єктної диверсності в умовах багатоверсійної розробки програмного забезпечення / Д. Л. Орловський, І. В. Лютенко, С. О. Лілікович** // Вісник НТУ «ХПІ». Серія: Системний аналіз, управління та інформаційні технології. – Х. : НТУ «ХПІ», 2016. – № 37 (1209). – С. 55–58. – Бібліогр.: 8 назв. – ISSN 2079-0023.

**Поход и средства измерения субъектной диверсности в условиях многоверсионной разработки программного обеспечения / Д. Л. Орловский, И. В. Лютенко, С. А. Лиликович** // Вісник НТУ «ХПІ». Серія: Системний аналіз, управління та інформаційні технології. – Харків : НТУ «ХПІ», 2016. – № 37 (1209). – С. 55–58. – Библиогр.: 8 назв. – ISSN 2079-0023.

**Approach and tools for measurement of subject diversity in context of multi-version software development / D. L. Orlovskyi, I. V. Lutenko, S. O. Lilikovych** // Bulletin of NTU "KhPI". Series: System analysis, control and information technology. – Kharkov : NTU "KhPI", 2016. – No. 37 (1209). – P. 55–58. – Bibliogr.: 8. – ISSN 2079-0023.

*Відомості про авторів / Сведения об авторах / About the Authors*

***Орловський Дмитро Леонідович*** – доцент кафедри програмної інженерії та інформаційних технологій управління, Національний технічний університет «Харківський політехнічний інститут»; тел.: 70-76-474; e-mail: ordm@kpi.kharkov.ua.

***Орловский Дмитрий Леонидович*** – доцент кафедры программной инженерии и информационных технологий управления, Национальный технический университет «Харьковский политехнический институт»; тел.: 70-76-474; e-mail: ordm@kpi.kharkov.ua.

***Orlovskyi Dmytro Leonidovych*** – assistant professor at the department of computer science and software engineering, National Technical University "Kharkiv Polytechnic Institute"; tel.: 70-76-474; e-mail: ordm@kpi.kharkov.ua.

***Лютенко Ірина Вікторівна*** – старший викладач кафедри програмної інженерії та інформаційних технологій управління, Національний технічний університет «Харківський політехнічний інститут»; тел.: (068) 342-24-48; e-mail: liv@kpi.kharkov.ua.

***Лютенко Ирина Викторовна*** – старший преподаватель кафедры программной инженерии и информационных технологий управления, Национальный технический университет «Харьковский политехнический институт»; тел.: (068) 342-24-48; e-mail: liv@kpi.kharkov.ua.

***Liutenko Iryna Viktorivna*** – senior lecturer at the department of computer science and software engineering, National Technical University "Kharkiv Polytechnic Institute"; tel.: (068) 342-24-48; e-mail: liv@kpi.kharkov.ua.

***Лілікович Сергій Олександрович*** – Національний технічний університет «Харківський політехнічний інститут», студент; тел.: (067) 317-29-68; e-mail: sergiy.lilikovych@gmail.com.

***Лиликович Сергей Александрович*** – Национальный технический университет «Харьковский политехнический институт», студент; тел.: (067) 317-29-68; e-mail: sergiy.lilikovych@gmail.com.

***Lilikovych Serhii Oleksandrovych*** – National Technical University "Kharkiv Polytechnic Institute", student; tel.: (067) 317-29-68; e-mail: sergiy.lilikovych@gmail.com.