DOI: https://doi.org/10.36910/6775-2524-0560-2021-44-13
УДК 681.3
Абгарян Юра Серьожайович, спеціаліст кафедри транспортних технологій, розробник програмного забезпечення SoftServe
https://orcid.org/0000-0001-8519-2539
Запорізький національний технічний університет.

АЛГОРИТМ АГРЕГАЦІЇ ПРОГРАМНИХ МЕТРИК І ЇЇ ЗАСТОСУВАННЯ ПРИ ТЕСТУВАННІ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Абгарян Ю. С. Алгоритм агрегації програмних метрик і її застосування при тестуванні програмного забезпечення. У статті розкрито алгоритм агрегації програмних метрик і її застосування при тестуванні програмного забезпечення. Визначено генезис формування наукової думки, щодо агрегації програмних метрик. Розкрито методологію тестування програмного забезпечення з відокремленням схеми процесу тестування програмного забезпечення. Наголошено, що в якості базису для визначення рівнів агрегації програмних метрик при тестуванні програмного забезпечення спочатку слід визначити процес і складові блоки на прикладі системи тестування. Підкреслено, що агрегація програмних метрик може проводитися на рівні винесення рішень, на рівні значень відповідності та на рівні ознак і зразків. Відзначено, що агрегація на першому та другому рівнях відбувається після залучення засобу порівняння, в той час як рівні третій та четвертий проводять операції до того, як пристрій порівняння видасть результуючі дані. Описано математичні властивості методів агрегації, а саме, домен, діапазон, інваріантність та розкладання. Представлено алгоритм агрегації програмних метрик до рейтингів, використовуючи порогові значення на основі еталонних показників. Покроково описано реалізацію алгоритму та визначено параметричні значення процесу агрегації. Наголошено, що зведення окремих вимірювань до рейтингів здійснюється за допомогою дворівневого процесу, заснованого на двох типах порогів, а окремі вимірювання об'єднуються в профілі ризиків за допомогою метричних порогів. При цьому, профілі ризику агрегуються за 5-бальною зірковою шкалою за допомогою порогових значень. Агрегація дворівнева, на першому рівні агрегація здійснюється шляхом обчислення відносного розміру системи, що підпадає під кожну категорію ризику, на другому рівні об'єднання профілів ризику в рейтинг здійснюється шляхом визначення мінімального рейтингу, для якого сукупний відносний розмір усіх категорій профілю ризику не перевищує набору порогів 2-го рівня. Здійснено тестування програмного забезпечення Dia. Профіль ризику для Dia містить 73,3% коду у низькому ризику, 8,2% помірного ризику, 7.9% високого ризику та 10.7% дуже високого ризику. Використання інтерпольованої функції дає рейтингове значення 2,99, рейтинг для Dia має три зірки.
Ключові слова: агрегація, програмна метрика, параметр, ризик, тестування, програмне забезпечення, програмний код, цикломатична складність.

Абгарян Ю. С. Алгоритм агрегации программных метрик и ее применение при тестировании программного обеспечения. В статье раскрыт алгоритм агрегации программных метрик и ее применение при тестировании программного обеспечения. Определен генезис формирования научной мысли, относительно агрегации программных метрик. Раскрыта методология тестирования программного обеспечения с отделением схемы процесса тестирования программного обеспечения. Отмечено, что в качестве базиса для определения уровней агрегации программных метрик при тестировании программного обеспечения сначала следует определить процесс и составные блоки на примере системы тестирования. Подчеркнуто, что агрегация программных метрик может проводиться на уровне принятия решений, на уровне значений соответствия и на уровне признаков и образцов. Отмечено, что агрегация на первом и втором уровнях происходит после привлечения средства сравнения, в то время как уровни третий и четвертый проводят операции до того, как устройство сравнения выдает результирующие данные. Описаны математические свойства методов агрегации, а именно, домен, диапазон, инвариантность и разложения. Представлен алгоритм агрегации программных метрик к рейтингам, используя пороговые значения на основе эталонных показателей. Пошагово описана реализация алгоритма и определены параметрические значения процесса агрегации. Отмечается, что сведение отдельных измерений к рейтингам осуществляется с помощью двухуровневого процесса, основанного на двух типах порогов, а отдельные измерения объединяются в профили рисков с помощью метрических порогов. При этом, профили риска агрегируются по 5-балльной звездной шкале с помощью пороговых значений. Агрегация двухуровневая, на первом уровне агрегация осуществляется путем вычисления относительного размера системы, что подпадает под каждую категорию риска, на втором уровне объединения профилей риска в рейтинг осуществляется путем определения минимального рейтинга, для которого совокупный относительный размер всех категорий профиля риска не превышает набора порогов 2-го уровня. Осуществлено тестирование программного обеспечения Dia. Профиль риска для Dia содержит 73,3% кода в низком риске, 8,2% умеренного риска, 7.9% высокого риска и 10.7% очень высокого риска. Использование интерполированной функции дает рейтинговое значение 2,99, рейтинг для Dia имеет три звезды.
Ключевые слова: агрегация, программная метрика, параметр, риск, тестирование, программное обеспечение, программный код, цикломатическая сложность.

Abharian Yura. Algorithm of aggregation of software metrics and its application at testing of software. The algorithm of software metrics aggregation and its application in software testing is revealed in the article. The genesis of the formation of scientific thought on the aggregation of program metrics is determined. The methodology of software testing with the separation of the scheme of the software testing process is revealed. It is emphasized that as a basis for determining the levels of aggregation of software metrics in software testing, you should first determine the process and component blocks on the example of a testing system. It is emphasized that the aggregation of software metrics can be carried out at the level of decision making, at the level of compliance values and at the level of features and samples. It is noted that aggregation at the first and second levels occurs after the comparison tool is involved, while the third and fourth levels perform operations before the comparison device produces the resulting data. Mathematical properties of aggregation methods are described, namely, domain, range, invariance and decomposition. The algorithm of aggregation of program metrics to ratings, using threshold values on the basis of reference indicators is presented. The implementation of the algorithm is

described step by step and the parametric values of the aggregation process are determined. It is emphasized that the reduction of individual measurements to ratings is carried out using a two-level process based on two types of thresholds, and individual measurements are combined in the risk profile using metric thresholds. At the same time, risk profiles are aggregated on a 5-point star scale using threshold values. Two-level aggregation, at the first level aggregation is carried out by calculating the relative size of the system falling under each risk category, at the second level combining risk profiles into a rating is carried out by determining the minimum rating for which the total relative size of all risk profile categories does not exceed a set of thresholds 2nd level. Dia software tested. The risk profile for Dia contains 73.3% of the code in low risk, 8.2% in moderate risk, 7.9% in high risk and 10.7% in very high risk. Using the interpolated function gives a rating value of 2.99, the rating for Dia has three stars.

Key words: aggregation, software metrics, parameter, risk, testing, software, software code, cyclomatic complexity.

Постановка проблеми. Використання програмних метрик для аналізу та оцінки програмного забезпечення шляхом кількісного збору певних характеристик або подання програмної системи є актуальним напрямком дослідження враховуючи швидкість розвитку ІТ сфери.

Однією з основних проблем використання програмних метрик є те, як об'єднати окремі вимірювання для збору інформації про загальну систему. Це загальна проблема, оскільки більшість метрик не мають визначення на системному рівні. Наприклад, для вимірювання складності запропоновано використовувати метрику Маккейба. Однак використання цієї метрики може легко створити кілька тисяч вимірювань, які буде важко проаналізувати, щоб прийти до судження про те, наскільки складна загальна система. На основі цього, було використано та запропоновано кілька методологій для узагальнення вимірювань, але всі вони мають недоліки. Наприклад, математичне додавання не може застосовуватися до всіх показників (наприклад, зв'язок між класами об'єктів); центральні тенденції часто приховують основні розподіли; показники розподілу та нерівності важко інтерпретувати, а їх результат складно простежити щодо метричних вимірів; індивідуальні формули важко перевірити. Загалом, ці методології неможливо об'єднати вимірювання у змістовний результат який би легко пояснювався та інтерпретувався.

Аналіз останніх досліджень і публікацій. Формування наукової думки, щодо агрегації програмних метрик припадає на 1976 рік, коли було розроблено міру цикломатичної складності програми Томасом Дж. Маккейбом. Потім, на протязі багатьох років, науковці підходили до вирішення проблем визначення міри складності комп'ютерної програми та її застосування у сфері інформаційних технологій. Так, Ю. Грицюк та О. Андрущакевич [1] розробили засіб для визначення якості програмного забезпечення методами метричного аналізу. Авторами з'ясовано особливості процесу оцінювання якості програмного забезпечення, тобто проаналізовано поняття якість програмного продукту як предмет стандартизації, а також рівні подання моделі якості програмного забезпечення, що дало змогу встановити можливість її підвищення шляхом формування відповідних вимог до критеріїв оцінювання якості, вдосконалення моделей метричного аналізу його якості та методів кількісного її вимірювання на всіх етапах реалізації програмного проекту. У [2] наведено основні напрями автоматизації тестування програмного забезпечення. Виконано аналітичний огляд систем автоматизованого тестування програмного забезпечення. Інформаційні системи для тестування програмного забезпечення допомагають керувати процесом тестування, відслідковуванням помилок та формуванням звітності. А.С. Авраменко, В.С. Авраменко та Г.В. Косенюк [3] розкрили основні поняття в області тестування програмного забезпечення, визначили критерії вибору тестів, здійснили оцінку відтестованості проекту. Значна увага приділяється модульному та інтеграційному тестуванню, інтеграційному тестуванню для об'єктно-орієнтованого програмування. Розглядаються питання автоматизації тестування тощо. А. В. Ільєнко, С. С. Ільєнко та Д. С. Сташевський [4] дослідили та визначили, що помилки, які виникають при розробці та використанні сучасних високонавантажених веб-додатків є дуже небезпечними, оскільки впливають на повноцінну життєдіяльність інформаційної системи в цілому та можуть призводити до порушення конфіденційності та цілісності персональної інформації користувачів. Із зарубіжних авторів варто відзначити такі роботи як: Shatnawi Raed [5], Meng T. [6], Sicilia, M. & Sánchez-Alonso, Salvador & Mora-Cantallops, Marçal & Barriocanal, Elena [7], Norris S. [8], Liu, Zhengli & Li, Bing & Wang, Jian & Yang, Rong [9], Franco, Eduardo [10], Broy, Manfred & Kuhrmann, Marco [11], Falco, Mariana & Robiolo, Gabriela [12], Serebrenik A., Roubtsov S. [13], Heitlager I, Kuipers T, Visser J. [14] та інші. Проте, враховуючи описані наукові набутки, за темою, питання розкриття алгоритму агрегації програмних метрик і її застосування при тестуванні програмного забезпечення залишається відкритим та потребує детального опрацювання.

Постановка завдання. Розкрити алгоритм агрегації програмних метрик і її застосування при тестуванні програмного забезпечення.

Викладення основного матеріалу дослідження. У технологіях виробництва програмного забезпечення тестуванню відводиться роль основного засобу забезпечення та контролю якості продукту.

Проявляється це в тому, що процеси тестування все більше інтегруються в проектні методи, а управління тестуванням стає найважливішою складовою управління проектами.

В якості базису для визначення рівнів агрегації програмних метрик при тестуванні програмного забезпечення спочатку слід визначити процес і складові блоки на прикладі системи тестування. На рисунку 1 показана схема процесу тестування програмного забезпечення.

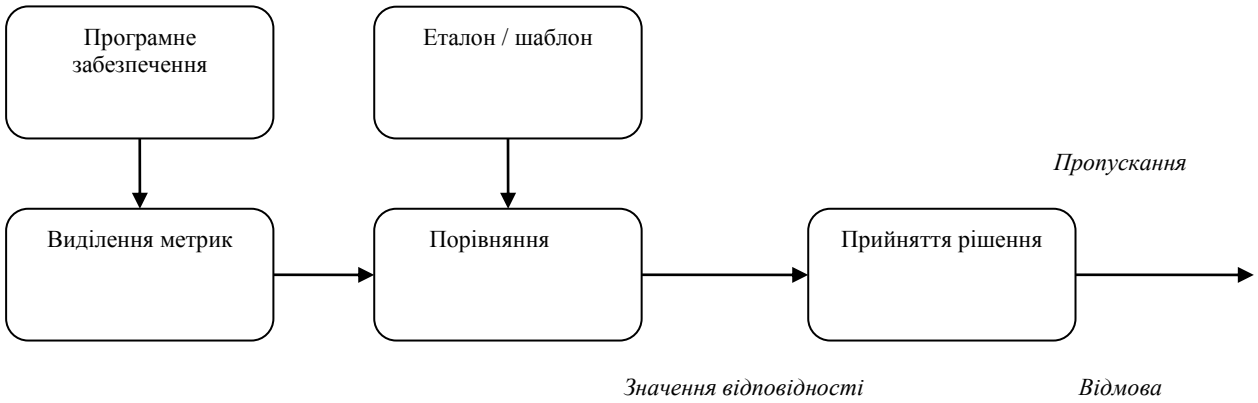
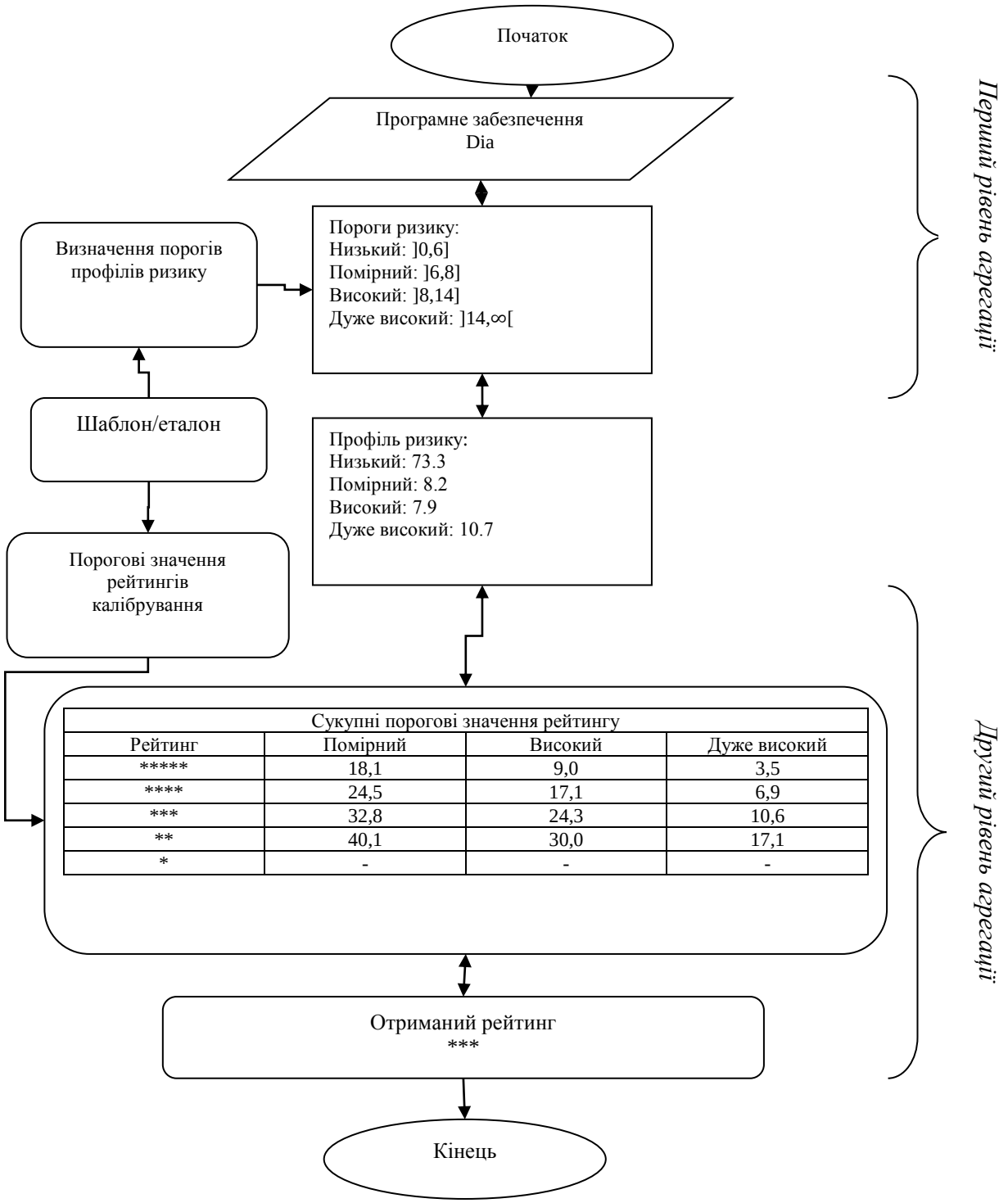


Рис. 1. Схема процесу тестування програмного забезпечення

Програмний додаток передається модулю виділення метрик. За допомогою методів обробки програмного забезпечення модуль виділення метрик перетворює зразок у вектор метрик (наприклад, цикломатична складність, кількість помилок на рядок коду, зв'язність, кількість класів, тощо), які формуються в поданні, придатному для порівняння. Пристрій порівняння отримує на вхід вектор метрик і порівнює його зі збереженим шаблоном / еталоном. Результатом є значення відповідності, що використовуються модулем винесення рішення для отримання відповіді на питання, чи відповідає запропонований зразок програмного забезпечення збереженому шаблону. Вихідними даними модуля є бінарне значення «Відповідність / невідповідність». Узагальнюючи вищеописаний процес, можна виділити такі рівні, на яких можливе застосування агрегації. Агрегація програмних метрик може проводитися на рівні винесення рішень (1), на рівні значень відповідності (2), на рівні ознак (3) і зразків (4). Відзначимо, що агрегація на рівнях (1) і (2) відбувається після залучення засобу порівняння, в той час як рівні (3) і (4) проводять операції до того, як пристрій порівняння видасть результуючі дані. Хоча інтеграція даних можлива на всіх перерахованих рівнях, агрегація на рівнях безлічі ознак, значень відповідності та винесення рішень використовується найбільш часто. а) Рівень рішень: кожен окремий процес в якості вихідних даних отримує логічне значення. Зазвичай процес їх агрегації відбувається за допомогою алгоритмів агрегації на основі логічних функцій диз'юнкції і кон'юнкції. б) Рівень значень відповідності: кожен окремий процес зазвичай видає як результат значення відповідності, але іноді це може бути і масив значень. в) Рівень ознак: кожен процес має на виході набір ознак. Процес агрегації об'єднує ці набори в одну безліч або в один вектор ознак.

г) Рівень зразка: кожен окремий процес отримує в якості вихідних даних набір зразків. При агрегації ці набори перетворюються в єдиний зразок. Математичні властивості методів агрегації, що мають значення для їх застосування при тестуванні програмного забезпечення: Домен. Область техніки агрегування визначає застосовність цієї техніки до класів програмних метрик. Економетричні індекси зазвичай застосовуються до розподілу доходів, тобто до наборів позитивних значень. Однак деякі показники програмного забезпечення можуть мати негативні значення, наприклад, індекс ремонтопридатності. Діапазон. Інтерпретація сукупного значення залежить від діапазону техніки агрегування: наприклад, 0,99 вказує на дуже високий ступінь нерівності. Значення, отримані шляхом застосування середнього значення, можуть коливатися від $-\infty$ до $+\infty$. Інваріантність. Техніка агрегації є інваріантною щодо додавання, якщо $I(x_1,\ldots,x_n) = I(x_{1+c},\ldots,x_{n+c})$ для будь-якого $x_1,\ldots,x_n$ *і* c, за умови існування $I(x_{1+c},\ldots,x_{n+c})$. Аналогічно, техніка агрегації є інваріантною щодо множення, якщо $I(x_1,\ldots,x_n) = I(x_{1*c},\ldots,x_{n*c})$ для будь-якого $x_1,\ldots,x_n$ *і* c за умови існування $I(x_{1*c},\ldots,x_{n*c})$.



Сукупні порогові значення рейтингу			
Рейтинг	Помірний	Високий	Дуже високий
*****	18,1	9,0	3,5
****	24,5	17,1	6,9
***	32,8	24,3	10,6
**	40,1	30,0	17,1
*	-	-	-

Рис. 2. Алгоритм агрегації програмних метрик до рейтингів, використовуючи порогові значення на основі еталонних показників

Агреговані рядки коду, що вимірюються для кожного файлу, інваріантні до техніки агрегації щодо додавання, дозволяють ігнорувати, наприклад, заголовки, що містять інформацію про ліцензування та включені до всіх вихідних файлів. На результати, отримані шляхом застосування техніки агрегування, інваріантної щодо множення, це не впливає, якщо враховувати відсотки від загальної кількості рядків коду, а не кількість самих рядків коду. Середнє значення не є інваріантним відносно додавання та множення. Розкладання є ключовою властивістю, необхідною для пояснення нерівності шляхом розподілу значень, які потрібно об'єднати, на неперетинні групи. В економетрії такі групи відповідають, наприклад, рівню освіти, статі чи етнічній приналежності, тоді як в еволюційному дослідженні програмного забезпечення – пакету, мові програмування та прізвищу автора.

На рисунку 2 представлено алгоритм агрегації програмних метрик до рейтингів, використовуючи порогові значення на основі еталонних показників. Зведення окремих вимірювань до рейтингів здійснюється за допомогою дворівневого процесу, заснованого на двох типах порогів. Окремі вимірювання об'єднуються в профілі ризиків за допомогою метричних порогів. Профіль ризику представляє собою відсоток загального коду, що потрапляє до кожної з чотирьох категорій ризику: низький, помірний, високий та дуже високий. Профілі ризику агрегуються за 5-бальною зірковою шкалою за допомогою порогових значень. Кожен рейтинг відкалібрований для представлення певного відсотка систем у контрольному етапі.

1) Агрегація першого рівня. Сукупність окремих вимірювань до профілів ризику з використанням порогів 1-го рівня здійснюється шляхом обчислення відносного розміру системи, що підпадає під кожну категорію ризику. Розмір вимірюється за допомогою вихідних рядків коду.

Оскільки профіль ризику складається з чотирьох категорій, необхідно чотири інтервали для класифікації всіх вимірювань. Інтервали, що визначають чотири категорії ризику для показників Маккейба, показані на рисунку 2 та представлені за допомогою позначень ISO/IEC 80000-2:2019 [15]. Ці інтервали були визначені трьома порогами, 6, 8 та 14, що представляють собою верхню межу категорій низький, помірний та високий, відповідно.

Щоб обчислити профіль ризику для системи Dia, використовуємо інтервали, показані на рисунку 2, для класифікації всіх методів на чотири категорії ризику. Потім для кожної категорії підсумовуємо розмір усіх цих методів, а потім ділимо їх на загальний розмір системи, в результаті чого відносний розмір (або відсоток) системи потрапляє до кожної категорії ризику. Наприклад, категорія низького ризику Dia обчислюється з урахуванням усіх методів, які мають значення Маккейба, які потрапляють в інтервал]0…6], тобто всі методи, які мають значення Маккейба менше або такі, що дорівнюють 6. Потім підсумовуємо рядки коду всіх цих методів (96, 361) і ділимо на загальні розміри Dia (133, 498), що в загальному складає 73.3%. Профіль ризику для Dia зображено на рисунку 2, містить 73,3% коду у низькому ризику, 8,2% помірного ризику, 7.9% високого ризику та 10.7% дуже високого ризику.

2) *Агрегація другого рівня*. Об'єднання профілів ризику в рейтинг здійснюється шляхом визначення мінімального рейтингу, для якого сукупний відносний розмір усіх категорій профілю ризику не перевищує набору порогів 2-го рівня.

Оскільки використано 5-бальну шкалу оцінок, потрібно як мінімум 4 набори порогів, що визначають верхні межі, необхідні для покриття всіх можливих значень профілю ризику. Кожен набір з порогових значень визначає сукупні верхні межі для категорій помірного, високого та дуже високого ризику. Сукупна верхня межа для категорії враховує обсяг коду для цієї категорії плюс усі вищі категорії (наприклад, сукупна верхня межа для помірного ризику враховує відсоток об'єму помірних, високих та дуже високих категорій профілю ризику). Варто наголосити, що оскільки сукупна категорія низького ризику завжди буде 100%, немає необхідності вказувати для неї порогові значення. На рисунку 2 показана таблиця, що містить пороги 2-го рівня для метрики Маккейба, відкалібровані за алгоритмом, в якому, в якості вхідних даних береться два аргументи: сукупні профілі ризику для всіх систем у контрольному етапі та розділ, що визначає бажаний розподіл систем за рейтингом. Сукупні профілі ризику обчислюються з використанням порогів 1-го рівня, для кожної окремої системи еталону. Розділ розміром N−1 визначає кількість систем для кожного рейтингу (від найвищого до найнижчого). Алгоритм починається в рядку 1 з ініціалізації змінних порогів, які будуть містити результат алгоритму калібрування (порогові значення рейтингу). Потім у рядках 2–4 кожна категорія ризику профілів ризику впорядковується та зберігається як матриця у впорядкованій змінній. Стовпці матриці складатимуться з трьох категорій ризику, а рядки представлятимуть значення для категорій ризику базового рівня. Ця матриця відіграє важливу роль, оскільки кожна позиція буде повторюватися, щоб знайти порогові значення для кожного рейтингу. Алгоритм має дві основні частини: пошук початкового набору порогів, що відповідає бажаній кількості систем для цього рейтингу, і частина оптимізації, яка відповідає за пошук найменших можливих порогів для трьох категорій ризику.

Щоб визначити рейтинг для Dia, спочатку обчислюємо сукупний профіль ризику, тобто сукупний відносний розмір для категорій помірний, високий та дуже високий. Це робиться шляхом врахування відносного розміру кожної категорії ризику плюс усі вищі категорії, що дає 26,2% для помірного ризику, 17,1% для високого ризику і 10,2 % дуже високого ризику. Потім ці значення порівнюються з пороговими оцінками Маккейба, як показано на рисунку 2, і отримано рейтинг 3 зірки. Використання інтерпольованої функції дає рейтингове значення 2,99. Рейтинг для Dia зображено на рисунку 2 – три зірки.

Відстеження рейтингу 3 зірки до окремих вимірювань досягається шляхом повторного використання порогів 2-го та 1-го рівнів. Це відстеження важливе не тільки для того, щоб пояснити

рейтинг, але й для отримання інформації про потенційні проблеми, які потім можуть бути використані для підтримки прийняття рішень.

Аналіз стійкості використаних порогів полягає у запуску алгоритму калібрування n разів, кожен з випадково вибраної підмножини систем, наявних у вихідному наборі. Результатом є n порогових таблиць на метрику. Можливі два способи оцінки цих чисел:

1) перевірити мінливість порогових значень;

2) застосовувати порогові значення до вихідного набору систем та перевіряти відмінності в рейтингах.

Для оцінки стабільності порогових значень з точки зору обчислених рейтингів, для кожної системи необхідно розрахувати абсолютні відмінності від її медіанного рейтингу протягом усіх прогонів. Модель більш стабільна на рівні рейтингів, ніж на пороговому рівні. Цього слід очікувати, оскільки відмінності в порогових значеннях різних категорій ризику для одного рейтингу можуть компенсувати один одного.

Такими чином, рейтинги мають обмежений вплив, викликаний включенням або виключенням конкретних систем або невеликих груп систем. Це свідчить про хорошу стабільність результатів, отриманих за допомогою цього еталону.

Висновки і перспективи подальших досліджень. У роботі розкрито алгоритм агрегації програмних метрик і її застосування при тестуванні програмного забезпечення. Доступні численні програмні метрики для вимірювання різних аспектів якості програмного забезпечення, ці показники визначаються на низькому рівні окремих компонентів: функцій, методів, класів. Агрегація різнонаправлених метрик дозволяє досягти високого рівня тестування та більш точного кінцевого результату.

Перспективи подальших досліджень ґрунтуються на розробці програмного забезпечення здатного агрегувати програмні метрики враховуючи направленість програмного забезпечення, що підлягає тестуванню.

Список бібліографічного опису.

1. Грицюк Ю., Андрущакевич, О. (2018). Засіб для визначення якості програмного забезпечення методами метричного аналізу. *Scientific Bulletin of UNFU*, 28, 159-171. doi: 10.15421/40280631.
2. Єгорова, О. В., & Бичок, В. (2019). Програмні засоби для тестування програмного забезпечення. *Молодий вчений*, (11 (75)), 680-684.
3. Авраменко, А. С., Авраменко, В. С., & Косенюк, Г. В. (2017). *Тестування програмного забезпечення*: навч. посіб. Черкаси: ЧНУ імені Богдана Хмельницького.
4. Ільєнко, А. В., Ільєнко, С. С. Сташевський Д. С. Програмний модуль відслідковування помилок у високонавантажених веб-додатках на базі використання авторського алгоритму логеру. *Кібербезпека: освіта, наука, техніка*, *3*(11), 61-72.

References.

1. Shatnawi, Raed. (2020). Comparison of threshold identification techniques for object-oriented software metrics. IET Software. 14. 10.1049/iet-sen.2020.0025.
2. Meng T. et al. (2020) A survey on machine learning for data fusion //Information Fusion. – 2020. – Т. 57. – С. 115-129.
3. Sicilia, M. & Sánchez-Alonso, Salvador & Mora-Cantallops, Marçal & Barriocanal, Elena. (2020). On the Source Code Structure of Quantum Code: Insights from Q# and QDK. 10.1007/978-3-030-58793-2_24.
4. Norris S. (2019) Systematically working with multimodal data: Research methods in multimodal discourse analysis. – John Wiley & Sons, 2019.
5. Liu, Zhengli & Li, Bing & Wang, Jian & Yang, Rong. (2020). Requirements engineering for crossover services: Issues, challenges and research directions. IET Software. 15. 10.1049/sfw2.12014.
6. Franco, Eduardo. (2020). A dynamical evaluation framework for technical debt management in software maintenance process. 10.11606/T.3.2020.tde-17052021-140104.
7. Broy, Manfred & Kuhrmann, Marco. (2021). Eigenschaften und Strukturen von Softwaresystemen. 10.1007/978-3-662-50263-1_2.
8. Falco, Mariana & Robiolo, Gabriela. (2021). Building a Catalogue of ISO/IEC 25010 Quality Measures Applied in an Industrial Context. Journal of Physics: Conference Series. 1828. 012077. 10.1088/1742-6596/1828/1/012077.
9. Serebrenik A, Roubtsov S, van den Brand MGJ. Dn-based architecture assessment of Java open source software systems. In ICPC '09: Proc. 17th Int. Conf. on Program Comprehension, 2009, IEEE, 2009; 198–207.
10. Heitlager I, Kuipers T, Visser J. A practical model for measuring maintainability. In Proceedings of the 6th International Conference on Quality of Information and Communications Technology. IEEE Computer Society: Washington, DC, USA, 2007; 30–39
11. ISO/IEC 80000-2:2019 Quantities and units – Part 2: Mathematics. – https://www.iso.org/standard/64973.html