DOI 10.31891/2307-5732-2021-301-5-166-169
УДК 004.051

МАРТИНЮК О. Р., ЯШИНА О. М., РАДЕЛЬЧУК Г. І., КУСТОВСЬКИЙ Р. С.
Хмельницький національний університет

ПОРІВНЯННЯ ПРОГРАМНИХ МЕТРИК ДЛЯ ОЦІНКИ ЯКОСТІ ПРОГРАМНИХ ПРОДУКТІВ

У статті наведено результати досліджень різних типів метрик програмного забезпечення та конкретних представників даних типів. Описано сферу їх застосування та зручність використання як при оцінці якості готового продукту, так і на стадії розробки.

Ключові слова: управління якістю, контроль якості, програмне забезпечення, метрика програмного забезпечення, програмний продукт.

MARTYNIUK OLEKSANDR RUSLANOVYCH, YASHYNA OKSANA MYKOLAIVNA,
RADELCHUK GALYNA IVANIVNA, KUSTOVSKYI ROMAN SERGIYOVYCH
Khmelnytskyi National University

COMPARISON OF SOFTWARE METRICS FOR QUALITY ESTIMATION OF THE SOFTWARE PRODUCTS

There is a well known principle, that you can manage only what you can measure – therefore adequate measurement and estimation of different elements in any engineering area is a very important development factor, and software engineering is not an exception. Software metrics represent numeric values of software characteristics and are used for analysis of different aspects of the product in order to predict possible problems. Software metrics can be applied to any stage of the software life cycle. For instance, on a stage of requirement analysis and the problem formulation, these metrics help to estimate the required amount of work and, respectively, the amount of financial and work resources. In addition, the metrics can be used directly in estimation of the program code complexity to assess it's performance, effectiveness and the ability to modify it if needed. Software metrics are an important part of the whole software quality management process as they can be easily automated and applied in order to achieve some valuable technical characteristics. Moreover, they can be combined with another quality management tools and are a beneficial addition to any software product development model. In general, measuring the attributes of software during any phase of development gives a number of beneficial results to the developers as it helps to use the time, finances and effort effectively. This is achieved because of early problems identification, estimation and prevention and because of that the size, cost, quality and simplicity of the maintenance of the product is always under control. This article describes the results of research on different software metrics types and the actual representatives of these types. It assesses the usage area and the convenience of it's usage on evaluating the quality of both a finished product and a product in the development stage.

Keywords: quality management, quality control, software, software metric, software product.

Вступ. Постановка проблеми

Адекватне вимірювання та оцінка різних елементів у будь-якій галузі інженерії є важливим фактором розробки. Інженерія програмного забезпечення не є виключенням, оскільки загальновідомим є принцип – керувати можна лише тим, що можна виміряти.

Метрики програмного забезпечення (ПЗ) представляють собою чисельні значення характеристик ПЗ та використовуються для аналізу різних аспектів продукту з метою передбачення можливих проблем. Програмні метрики можуть бути використані на будь-якому етапі життєвого циклу ПЗ. Наприклад, на етапі аналізу вимог та постановки задачі, метрики допомагають оцінити необхідний обсяг робіт та, відповідно, фінансові і трудові ресурси. Також метрики можуть використовуватись безпосередньо для оцінки складності програмного коду, для аналізу його швидкодії, ефективності та можливості адаптації у разі необхідності. Програмні метрики є важливою частиною всього процесу управління якістю ПЗ, оскільки вони можуть бути легко автоматизовані та застосовані з метою досягнення важливих технічних характеристик. Більше того, вони можуть бути поєднані з іншими засобами управління якістю і є корисним доповненням до моделі розробки будь-якого ПЗ.

Загалом, вимірювання характеристик ПЗ на будь-якій стадії розробки приносить низку корисних результатів для розробників, оскільки допомагає ефективно використовувати час, фінанси та зусилля. Це досягається шляхом раннього розпізнання, оцінки та запобігання проблемам; за рахунок цього розмір, вартість, якість та простота підтримки продукту завжди буде під контролем.

Аналіз останніх досліджень та публікацій

Основою дослідження є праці науковців у сфері управління якістю ПЗ, управління проектами ПЗ та міжнародні стандарти серії ISO/IEC щодо оцінки і вимірювання якості програмних продуктів. Згідно з дослідженнями, які описані у працях [1, 2], оцінка та забезпечення якості ПЗ є широкою темою, тому існує багато різних підходів до оцінки якості програмних продуктів. Загалом, поняття якості трактується як ступінь відповідності кінцевого продукту вимогам замовника. У галузі ПЗ визначені наступні основні критерії якості [3]:

- правильність;
- ефективність;
- цілісність;
- зручність використання;
- зручність підтримки;

– гнучкість;
– зручність тестування;
– можливість повторного використання.

У процесі розробки програмного продукту враховуються усі важливі критерії та їх пріоритети, у результаті чого підбирається модель управління якістю ПЗ, яка передбачає використання конкретних принципів розробки, метрик та методів тестування.

Метою роботи є порівняння та аналіз метрик програмного забезпечення і типів, до яких вони відносяться, на предмет сфери застосування та зручності використання.

Виклад основного матеріалу

Метрики програмного забезпечення – це стандартизовані вимірювання певних характеристик програмної системи чи процесу з метою визначення їх ступеня та міри впливу на продукт у цілому. Метою обрахунку програмних метрик є отримання об'єктивних та зрозумілих вимірювань для подальшого використання при плануванні бюджету, плану розробки, тестуванні, оптимізації та документації ПЗ [4]. Застосування метрик надає низку переваг для розробників, оскільки це спрощує прийняття вагомих рішень щодо архітектури програмного продукту, використання тих чи інших фреймворків, мов програмування тощо. Також це дозволяє швидко розпізнавати та усувати можливі несправності продукту, дає чітке і повне розуміння швидкодії програми та шляхи її оптимізації.

Тим не менше, оскільки процес проектування та розробки програмного продукту є доволі складним і комплексним, метрики ПЗ виступають лише інструментом при аналізі проекту спеціалістом та мають певні обмеження. Наприклад, застосування метрик не завжди є простим процесом, в багатьох випадках це складно та дорого; окрім того, кожен проект є унікальним. Саме тому загальновідомі метрики можуть бути використані лише для вимірювання базових характеристик програмного коду, а для глибшого аналізу слід розробити метрику конкретно для даного програмного продукту.

Метрики програмного забезпечення є емпіричними, а тому не покривають весь спектр атрибутів коду і відображають не абсолютну оцінку ПЗ, а лише ймовірність виникнення проблем та збоїв [5].

Методологія даного дослідження полягає у використанні наступних критеріїв порівняння метрик ПЗ:

– основна ідея метрики та галузь її застосування;
– простота реалізації та гнучкість використання;
– основні переваги та недоліки метрик ПЗ.

Класифікація розглянутих у даній статті програмних метрик представлена на рис. 1.

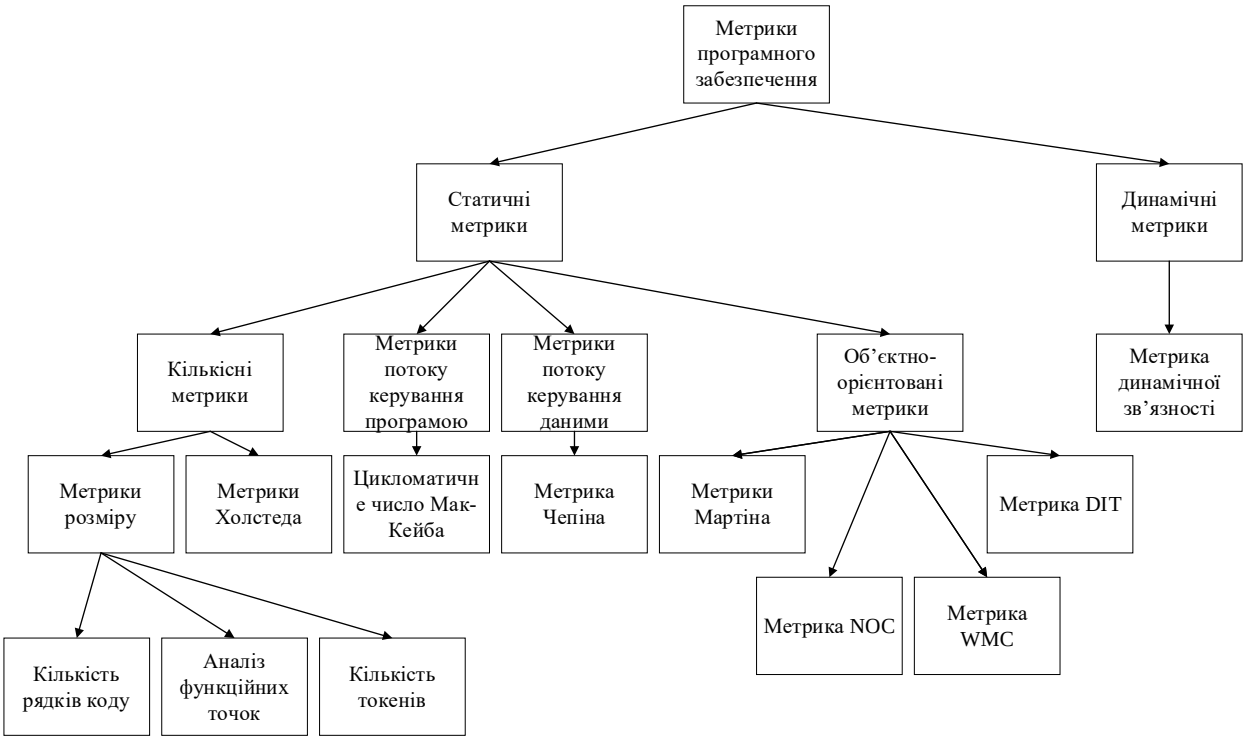

Рис. 1. Класифікація метрик програмного забезпечення

У сучасних комплексних проектах знаходять своє застосування представники всіх типів програмних метрик, однак кожна з них має свої особливості застосування та може трактуватись по-різному, залежно від специфіки конкретного програмного продукту.

Метрика динамічної зв'язності (Dynamic Coupling Metric – DCM) використовується для вимірювання зв'язності між парою об'єктів чи класів під час роботи програмної системи [6]. Перевагою цієї

метрики є інформативність, оскільки її можна використовувати на рівні модулів та виміряти ступінь залежності між цими модулями. До недоліків метрики відноситься складність використання та неможливість її застосування на ранніх етапах розробки ПЗ, оскільки у наявності має бути робоча система з відомою архітектурою.

Кількість рядків коду (Source Lines of Code – SLoC) – це метрика розміру програмного продукту, основна ідея якої полягає у підрахунку кількості рядків коду, без врахування коментарів та порожніх рядків [7]. Ця метрика розроблена для оцінки зусиль, витрачених на розробку програмного модуля та для порівняння продуктивності розробників. Метрика SLoC є однією з найпростіших і ранніх метрик, тому вона має низку обмежень. Основними з цих обмежень є те, що одна й та сама функціональність може бути як записана в один рядок, так і розбита на декілька; також потрібно враховувати специфіку окремих мов програмування. Враховуючи це, дану метрику не можна вважати гнучкою, проте вона є простою у реалізації та має широку галузь застосування. Розрізняють два типи рядків – фізичні та логічні; фізичні рядки представляють кількість усіх рядків коду, а логічні – кількість команд програми.

Кількість токенів (Token Count) – концепція цієї метрики полягає у тому, щоб розглядати програму як колекцію токенів. Токенами вважаються унікальні оператори та операнди, і вся логіка роботи програми може бути визначена за допомогою цих базових конструкцій [1]. Розмір програми вважається сумою кількості унікальних операторів та кількості унікальних операндів. Метрика Token Count має широку галузь застосування, оскільки вона може бути інтегрована у будь-який проект. Проте ця метрика не враховує специфіки архітектури конкретної програмної системи, а тому вона може використовуватись лише як додатковий інструмент, а не як самостійний критерій оцінки ПЗ.

Аналіз функціональних точок (Function Point) – виражає обсяг бізнес-логіки ПЗ за допомогою аналізу функцій та модулів системи. Усі функції програми розподіляються на п’ять типів: функції введення, функції виведення, запити, функції роботи з файлами та зовнішні інтерфейси. Після розподілу на категорії проводиться аналіз алгоритмічної складності кожної функції, на основі чого формується розмір програмного продукту [2]. Перевагою цієї метрики є її універсальність – результати оцінки не залежать від мови програмування та архітектури додатку; проте ця метрика є доволі складною у використанні.

Метрики Холстеда (Halstead Metrics) також відносяться до кількісних метрик, проте вони відображають значно більше інформації у порівнянні з попередніми метриками. Цей метод передбачає підрахунок загальної кількості операторів, загальної кількості операндів, кількості унікальних операторів та кількості унікальних операндів; за допомогою цих показників розраховуються значення розміру програми, розміру словника, складність програми, обсяг зусиль тощо [1]. До переваг Halstead Metrics можна віднести відносну простоту використання та універсальність, оскільки ці метрики не залежать ні від мови програмування, ні від складності алгоритму, що описується. До недоліків метрик Холстеда можна віднести те, що вони є статичними, тому вони обраховуються безпосередньо за допомогою коду, і, отже, не відображають атрибути програмної системи під час її виконання.

Найпоширенішим представником ***метрик потоку керування програмою*** є цикломатична складність програми, або ***цикломатичне число Мак-Кейба*** (McCabe’s Cyclomatic Metric). Концепція цієї метрики полягає у представленні програми як зв’язного орієнтованого графа, вузли якого відображають частини вихідного коду, а ребра – потік керування, який виконується під час роботи програми. Цикломатичне число рівне числу лінійно незалежних шляхів роботи програми у його відображенні за допомогою графа. Ця метрика описує складність роботи програми та число можливих відгалужень алгоритму роботи під час її виконання.

Метрика Чепіна (Chapin’s Metric) – метрика ***потоку керування даними***, суть якої полягає в оцінці інформаційної єдності окремо взятого програмного модуля за допомогою аналізу використання змінних введення та виведення. При цьому змінні поділяються на чотири типи: вхідні змінні, створені змінні, керуючі змінні та паразитні змінні. Ця метрика є доволі корисною при аналізі програмних модулів на оптимальність роботи з даними, проте її складно обчислити, а поділ змінних потребує від розробника додаткового аналізу.

Широкого використання набувають ***об’єктно-орієнтовані метрики***, оскільки об’єктно-орієнтована парадигма програмування на сьогодні є досить поширеною. Серед цих метрик можна виділити наступні. ***Метрики Мартіна*** (Martin’s Metrics) допомагають оцінити рівень нестабільності класу, рівень абстракції та залежність нестабільності класу від його абстракції. ***Метрика WMC*** (Weighted Methods per Class) представляє собою сумарну складність усіх методів класу, ***метрика DIT*** (Depth of Inheritance Tree) – глибину дерева наслідування, ***метрика NOC*** (Number of children) – кількість класів, що наслідують даний [8].

Об’єктно-орієнтовані метрики дають змогу оцінити оптимальність описаних класів та інтерфейсів з метою забезпечення хорошого рівня абстракції та зв’язності між об’єктами класів, однак ці метрики є вузькоспеціалізованими і можуть бути використані лише при аналізі ПЗ у межах об’єктно-орієнтованої парадигми програмування.

В цілому очевидно, що для управління якістю конкретного програмного продукту не можна обмежуватись використанням однієї метрики чи одного типу метрик, оскільки вони покликані вирішувати різні завдання проектування ПЗ. Частина програмних метрик є універсальними, а інші використовуються лише при розробці програм у певній парадигмі програмування. Тому оптимальним підходом є комбінування різних програмних метрик та розробка власних при потребі глибшої та якіснішої оцінки якості ПЗ.

Висновки

У дослідженні виконано порівняння та аналіз різних типів метрик ПЗ та конкретних представників цих типів на предмет галузі застосування, концепції окремих метрик, а також зручності та гнучкості використання. Визначені основні переваги та недоліки метрик ПЗ при використанні у проектах та особливості їх реалізації.

За результатами дослідження встановлено, що жодна метрика не може вирішити завдання виміру якості програмного продукту у повній мірі, а тому оптимальним є використання різних типів метрик для отримання детальної оцінки ПЗ. Підхід, який полягає у поєднанні різних програмних метрик залежно від парадигми, архітектури, а також етапу життєвого циклу ПЗ, дозволяє досягати високої якості програмного продукту.

Література

1. A Study of Software Metrics. ResearchGate. URL: https://www.researchgate.net/-publication/322070697_A_Study_of_Software_Metrics

2. Analysis of Software Quality Using Software Metrics. URL: https://www.researchgate.net/publication/328830202_Analysis_of_Software_Quality_Using_Software_Metrics

3. Software Quality Factors and Software Quality Metrics. URL: https://www.researchgate.net/publication/263582173_Software_Quality_Factors_and_Software_Quality_Metrics_to_Enhance_Software_Quality_Assurance

4. Galin D. Software quality. Concepts and practice. Publisher: Wiley-IEEE Press, 2018. 720 p.

5. Garst Smith Howard T. Software quality assurance: A guide for developers and auditors. Publisher: CRC Press Inc, 2020. 480 p.

6. Suryn W. Software quality engineering. A practitioner's approach. Publisher: Wiley-IEEE Computer Society Pr, 2014. 208 p.

7. ISO/IEC 25023:2016 Systems and software engineering – Systems and software Quality Requirements and Evaluation (SQuaRE) – Measurement of system and software product quality. ISO.org. URL: https://www.iso.org/ru/standard/35747.html

8. Рыжков Е. Программный код и его метрики [Электронный ресурс] / Е. Рыжков // Веб-портал Хабр. – Режим доступа : https://habr.com/ru/company/intel/blog/106082/

References

1. A Study of Software Metrics. ResearchGate. URL: https://www.researchgate.net/-publication/322070697_A_Study_of_Software_Metrics

2. Analysis of Software Quality Using Software Metrics. URL: https://www.researchgate.net/publication/328830202_Analysis_of_Software_Quality_Using_Software_Metrics

3. Software Quality Factors and Software Quality Metrics. URL: https://www.researchgate.net/publication/263582173_Software_Quality_Factors_and_Software_Quality_Metrics_to_Enhance_Software_Quality_Assurance

4. Galin D. Software quality. Concepts and practice. Publisher: Wiley-IEEE Press, 2018. 720 p.

5. Garst Smith Howard T. Software quality assurance: A guide for developers and auditors. Publisher: CRC Press Inc, 2020. 480 p.

6. Suryn W. Software quality engineering. A practitioner's approach. Publisher: Wiley-IEEE Computer Society Pr, 2014. 208 p.

7. ISO/IEC 25023:2016 Systems and software engineering – Systems and software Quality Requirements and Evaluation (SQuaRE) – Measurement of system and software product quality. ISO.org. URL: https://www.iso.org/ru/standard/35747.html

8. Ryzhkov E. Programmnyj kod i ego metriki [Elektronnyj resurs] / E. Ryzhkov // Veb-portal Habr. – Rezhim dostupa : https://habr.com/ru/company/intel/blog/106082/

Рецензія/Peer review : 25.09.2021 р. Надрукована/Printed :10.10.2021 р.