## ЕМПІРИЧНА ІНЖЕНЕРІЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

004.415.2.045 (076.5)

**О.П. Дишлевий, М.А. Книшик**
**Національний авіаційний університет**

# МЕТРИКА ПОКРИТТЯ КОДУ ЯК ЗАСІБ ВИЗНАЧЕННЯ ЯКОСТІ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

*Стаття присвячена розробці методики вимірювання покриття коду та засобу, що реалізує цю методику. Розроблена методика включає вимірювання трьох основних метрик покриття коду, а також власної метрику, що була виведена шляхом аналізу результатів вимірювань набору проектів. Був проведений аналіз вже існуючих засобів вимірювання покриття коду з метою виявлення їх переваг та недоліків, які були враховані вході розробки.*

*Статья посвящена разработке методики измерения покрытия кода и средства, реализующего эту методику. Разработанная методика включает измерение трех основных метрик покрытия кода, а также собственной метрику, которая была выведена путем анализа результатов измерений набора проектов. Был проведен анализ существующих средств измерения покрытия кода с целью выявления их преимуществ и недостатков, которые были учтены в ходе разработки.*

*The article is dedicated to the development of code coverage measurement technique and a tool that implements this technique. The techniqueinvolves measurements of three basic code coverage metrics, as well as own metric, which was deduced from analysis ofmeasurementresults of a set of projects. Existing code coverage measurement tools were reviewed in order to identify their strengths and weaknesses, which were taken into account during the development process.*

***Ключові слова:*** *якість програмного забезпечення; метрики; покриття коду; тестування.*

**Вступ**

Разом із збільшенням попиту на комп'ютерну техніку та мобільні пристрої зростає і потреба в програмах для них.

Проте не всі проекти з розробки програмного забезпеченняє успішними. По даним досліджень TheStandishGroup лише 32% проектів ПЗ завершуються успішно, без перевищення бюджету, строків та інших проблем. Ще 24% є повністю провальними [1].

Одна з головних причин їх провалу – недостатня ефективність тестування продукту, що є основним інструментом контролю його якості.Тому надзвичайно актуальною є проблема оцінки, контролю та підвищення якості програмних продуктів, що розробляються. Для такої оцінки в більшості проектів зараз використовують метрики покриття коду. При введенні вимірювання покриття на проекті, важливо уважно поставитись до вибору програмного засобу для цього. Існуючі засоби вимірювання метрик покриття коду мають ряд недоліків, що можуть значно зменшити ефективність їх використання.

Ця стаття присвячена задачі розробки методики обчислення метрик покриття коду та засобу, що реалізує її, враховуючи недоліки вже існуючих.

**Огляд останніх досліджень**

Однією з перших методик для систематичного тестування програмного забезпечення є техніка покриття коду. На сьогоднішній день існує багато метрик покриття коду. Найпоширенішими з них є:

- покриття рядків;
- покриття гілок;
- покриття шляхів.

**Цілі статті**

Основними цілями статті є:

- аналіз проблем вимірювання метрик покриття коду;
- розробка методики вимірювання метрик покриття коду;
- розробка засобу, що реалізує власну методику.

**Базові метрики покриття коду**

**Покриття рядків.** Покриття рядків є найпростішою для розуміння та вимірювання метрикою покриття коду. Рядок покритий, якщо він виконується [2].

Слід зазначити, що під рядком тут і далі мається на увазі найменший окремий елемент мови програмування (statement), а не фізичний рядок файлу вихідного коду.

Основним недоліком даної метрики є її нечутливість до деяких структур, таких як операторів умов.

**Покриття гілок.** Значно повніше уявлення про покриття дає метрика покриття гілок. Мета використання метрики - переконатися, що всі гілки коду виконуються. Тобто кожна умова хоча б один раз має прийняти значення *true*та хоча б раз значення *false* [2].

**Покриття шляхів.** Метрика покриття шляхів дозволяє переконатися, що кожен з можливих шляхів у кожній функції був пройдений. Під шляхом розуміється унікальна послідовність гілок коду від входу у підпрограму до виходу з неї [2].

**Розробка загальної метрики покриття коду**

Для отримання більш повного розуміння результатів вимірювання, корисно обчислити метрику, що включає в себе значення трьох базових.При цьому необхідно брати до уваги різну вагомість кожної метрики.

На основі аналізу результатів вимірювань набору проектів було виявлено, що значення покриття шляхів в середньому становить 25%від значення покриття рядків, а значить є у 4 рази вагомішим. Той самий показник для покриття гілок становить 80%, тобто у 1,25 разів менше.

Враховуючи ці співвідношення були знайдені вагові коефіцієнти кожної з метрик. Отримана формула загального покриття(TC) має наступний вигляд:

$$TC = 0{,}16 * SC + 0{,}2 * BC + 0{,}64 * PC,$$

де SC – покриття рядків, BC – покриття гілок, PC – покриття шляхів.

**Огляд існуючих засобів вимірювання метрик покриття коду**

Введення в проекті вимірювання покриття може стати досить затратною задачею. В першу чергу це «заслуга» засобів для обчислення цих метрик. Було розглянуто декілька найпопулярніших вимірювачів для мови програмування Javaяк однієї з найпоширеніших сьогодні.

**Clover.** Clover – засіб вимірювання покриття коду, що представлений у вигляді плагіна для Eclipse та IntelliJ IDEA. Також існують версії для ApacheAnt та Maven. Для вимірювань використовує інструментування вихідного коду. Засіб дозволяє обчислити покриття рядків, гілок та методів. Також є можливість обчислення загального покриття за формулою [3]:

$$TPC = \frac{(BT + BF + SC + MC)}{2 * B + S + M} * 100\%$$

де
BT – гілки, що приймають значення true ;
BF – гілки, що приймають значення false;
SC – покриті рядки;
MC – відвідані методи;
B – загальна кількість розгалужень;
S – загальна кількість рядків;
M – загальна кількість методів.

Основні недоліки Clover:

- відсутність окремого графічного застосунку, що прив’язує до певних середовищ розробки або вимагає роботи з командним рядком;
- досить велика вартість. Clover– комерційне програмне забезпечення та вимагає ліцензування.

**EMMA.** Засіб з відкритим вихідним кодом, що, на відміну від Clover, інструментує байт код. Інструментування може бути виконане як статично так і динамічно. Для пов’язуваня байт коду з вихідним програма зберігає файл з метаданими.

EMMAне має власного графічного інтерфейсу і являє собою набір утиліт для командного рядка. Також є можливість роботи з Ant. Офіційних плагінів для середовищ розробки немає, проте існують деякі від сторонніх розробників для Eclipseта IntelliJIDEA.

EMMAобчислює покриття класів, методів, базових блоків та рядків [4].

Основні недоліки EMMA це:

- відсутність графічного інтерфейсу;
- відсутність підтримки покриття гілок, не кажучи вже про покриття шляхів.

**JaCoCo.** JaCoCo – засіб з відкритим вихідним кодом, що працює за методом інструментування байт коду. Інструментування проходить динамічно.

Засіб вимірює покриття рядків, гілок, методів, типів та цикломатичну складність.JaCoCo випускається для командного рядка, Ant та як плагін для Eclipse (EclEmma) [5].

Недоліки:

- відсутність графічного інтерфейсу;
- заміна покриття шляхів більш простою в реалізації цикломатичною складністю.

**Cobertura.** Cobertura – програма з відкритим вихідним кодом, що працює з java байт кодом, інструментуючи його перед запуском. Дозволяє виміряти покриття рядків, гілок та цикломатичну складність Мак-Кейба. Програма запускається з командного рядка або за допомогою Ant [6].

Недоліки програми:

- відсутність графічного інтерфейсу;
- цикломатична складність замість покриття шляхів.

**Jmockit.** Jmockit – бібліотека для тестування з відкритим кодом, шо має інструменти для обчислення покриття. Вона здійснює ІБК на льоту. Робота з бібліотекою здійснюється за допомогою Antабо через консоль Java.

Jmockitобчислює покриття рядків, шляхів та даних. Під покриттям даних розуміють перевірку, чи був зчитаний об’єкт чи статичне поле після останньої операції присвоєння [7].

Основні недоліки:

- відсутність графічного інтерфейсу;
- відсутність метрики покриття рядків;
- неспеціалізованість засобу.

## Розробка методики вимірювання метрик покриття коду

**Аналіз підходів до вимірювання покриття коду.** Існує два основних підходи до вимірювання покриття коду:

- Інструментування вихідного коду;
- Інструментування байт коду [8].

За першим методом вносяться певні зміни у вихідний код, після чого змінений код компілюється та запускається. Під час виконання цей код змінює значення певних лічильників, значення яких і будуть показувати покритий елемент коду чи ні.

Основними недоліками даного підходу є:

- необхідність зберігання двох версій вихідного коду;
- необхідність слідкувати, щоб інструментований код не впливав на роботу оригінального;
- використання пам’яті: для кожного лічильника необхідно створити окрему змінну [8].

За методом інструментування байт коду вихідний код компілюється, а зміни вносяться вже в код скомпільованих класів. Можливі два варіанти інструментування: статичне та динамічне.

При статичному інструментуванні, байт код програми змінюється до того, як її буде запущено. При цьому потрібно зберегти окрему копію змінених файлів.

При динамічному інструментуванні, код змінюється на льоту при зверненні до класу.

Головним недоліком даного способу є складність встановлення взаємозв’язку між байт кодом та вихідним, що значно ускладнює вимірювання. Також результати такого вимірювання дуже важко проконтролювати [8].

На основі аналізу двох підходів до вимірювання та засобів, що їх реалізують, було обрано один, більш зручний та точний. Інструментування байт коду виконується швидше ніж вихідного, проте останнє має кілька значних переваг:

- робота безпосередньо з кодом;
- простота реалізації;
- точність вимірювання;
- можливість контролю результатів [9].

Тому для розробки засобу вимірювання метрик покриття обрано саме інструментування вихідного коду.

## Методика обчислення базових метрик покриття

Обравши підхід, можна перейти безпосередньо до алгоритму вимірювання метрик покриття коду. Алгоритм має наступний вигляд:

1. Створити масив лічильників.
2. Вставити в точках інтересу спеціальний код, що буде інкрементувати відповідні елементи масиву.
3. Виконати інструментований код
4. Після завершення виконання зчитати значення лічильників.

Для обчислення покриття рядків створюємо статичний масив лічильників з розмірністю, що відповідає загальній кількості рядків, вставляємо перед кожним рядком конструкції типу COUNTER.S[0]++ та запускаємо. Інкремент вставляється *перед* рядком на випадок, якщо той викличе виняткову ситуацію.

Для обчислення покриття гілок інструментуються оператори умов.Конструкціяif((aBoolean&& ++BT[0]!=0 | true ) || ( ++BF[0] == 0 &false)) прийматиме ті ж самі значення що і if (aBoolean), одночасно даючи нам інформацію про виконання тієї чи іншої гілки.

Якщо aBoolean == true, тоді && також обчислюється, що інкрементуєBT[0]. Потім переконуємось, що ввесь вираз буде дорівнювати true.

Так само, якщо aBoolean == false, тоді виконання переключається до правої частини оператору ||, де інкрементується BF[0]. Також переконуємось, що вираз дорівнює falsei буде виконана гілка else.

Варто зауважити, що цей спосіб обчислення ПГ прийнятний для мов програмування, які реалізують укорочену оцінку булевих виразів: коли значення виразу може бути однозначно визначене, його обчислення зупиняється. Інакше будуть інкрементуватись обидва лічильники.

Для вимірювання покриття шляхів під час вимірювання покриття гілок кожнагілка в коді нумерується. Послідовність проходження цих точок записується у вигляді списку в статичну колекцію, якщо такої послідовності в ній ще немає. Після завершення виконання програми покриття шляхів розраховується за формулою:

$$P = \frac{P_{cov}}{2^B} * 100\%,$$

де $P_{cov}$ – кількість пройдених шляхів, B – кількість розгалужень. Тобто кількість пройдених шляхів ділиться на їхню загальну кількість.

## Розробка засобу вимірювання метрик покриття коду

Для вирішення проблем вимірювання метрик покриття було розроблено MCover – вимірювач, який враховує виявлені недоліки існуючих засобів.

MCover має простий та зрозумілий графічний інтерфейс, тому не потребує додаткового навчання. Програма здатна виміряти покриття рядків, гілок та шляхів та обчислити на їх основі загальне покриття. Результати можна представити графічно, що полегшує їх сприйняття та інтерпретацію.

Користувач задає шлях до каталогу з файламивихідного коду та вказує каталог, в якому будуть збережені інструментовані файли. Вихідний код копіюється до каталогу призначення. Під час копіювання MCover одночасно аналізує його, та додає свої оператори перед кожним рядком та в оператори умов, рахуючи їх. Створюється два статичних масиви з розмірністю, що відповідає кількості рядків та гілок.

Після цього MCover знаходить в інструментованому коді всі можливі точки входу, а користувач обирає одну з них. Програма запускається, а додані оператори, якщо вони виконуються, змінюють відповідні значення масивів. Незмінені значення відповідають непокритим елементам. Виходячи з даних у масивах можна обчислити значення трьох базових метрик покриття.Після завершення програми, що тестується, вимірювач покаже покриття для виконаних дій.

Приклад результатів вимірювання зображений на рис. 1.

Коли значення метрик виміряно, користувачможе в будь-який момент переглянутиграфічнепредставленнярезультатів у виглядідіаграми (рис 2). Висотастовпчиків та їхколірзалежитьвідзначеннявідповідної метрики. Кольоризмінюєтьсявідчервоного (>0%) до жовтого (50%) та віджовтого до зеленого (100%). Колірпрямокутника, на якомувідображаютьсястовпцізмінюєтьсяаналогічним чином та відповідаєзначенню метрики загального покритя.

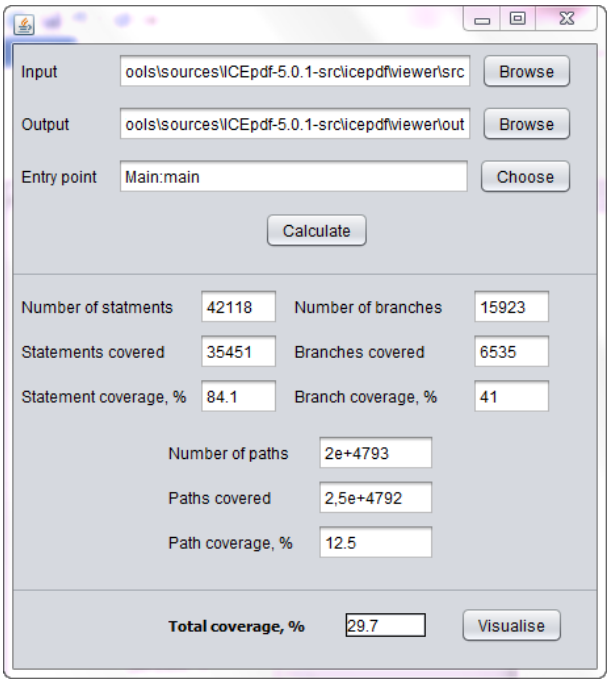


Рис.1. Приклад результатів вимірювання MCover

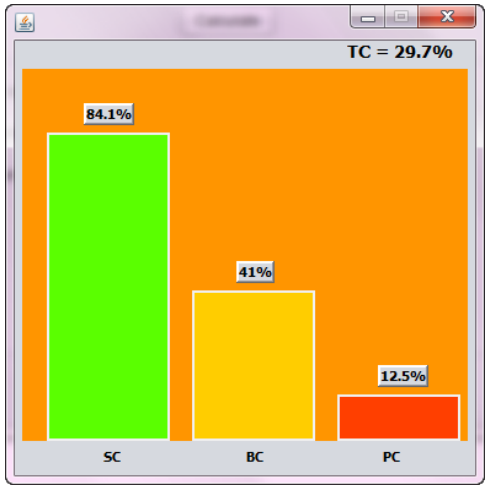


Рис. 2. Візуалізація результатів MCover

**Порівняння результатів вимірювання метрик покриття коду**

Вимірювання покриття не вимагає надто точних результатів. Вимірювачі працюють за різними принципами та часто навіть дають різні визначення одним і тим самим метрикам. Тому результати оцінки покриття різними програмами може дещорізнитися. Тим не меншбуло зроблено кілька контрольних вимірювань для перевірки роботи програми MCover.

Результати порівнювались з наступними засобами:

- Clover
- JaCoCo
- Cobertura
- jMockit

Для вимірювання було відібрано три проекти різних розмірів:

- ICEpdf – програма для перегляду документів PDF
- CleanSheets – табличний процесор,аналог MSExcel
- Jcrontab – планувальник задач

*Таблиця*

*Порівняння результатів вимірювання метрик покриття коду різними засобами*

| Засіб | Покриття рядків, % | | | Покриття гілок, % | | | Покриття шляхів, % | | | Загальне покриття, % | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ICEpdf | Clean Sheets | Jcrontab | ICEpdf | Clean Sheets | Jcrontab | ICEpdf | Clean Sheets | Jcrontab | ICEpdf | Clean Sheets | Jcrontab |
| Clover | 15,7 | 40,2 | 26,6 | 12,8 | 27 | 19,9 | - | - | - | 15,7 | 39,6 | 25,8 |
| JaCoCo | 19,4 | 47,1 | 29,1 | 12,4 | 25,4 | 18,1 | - | - | - | - | - | - |
| Cobertura | 18,93 | 47,11 | 29,47 | 12,32 | 24,64 | 18,83 | - | - | - | - | - | - |
| jMockit | 17 | 46 | 28 | - | - | - | 6 | 13 | 7 | - | - | - |
| MCover | 16,2 | 45 | 27,8 | 12,5 | 25,1 | 19,1 | 5,1 | 12,6 | 6,3 | 8,4 | 20,3 | 12,3 |

Як можна побачити з таблиці, порівняльні вимірювання не показали значних відхилень значень базових метрик MCover від інших засобів вимірювання метрик покриття коду.

Значення загального покриття значно відрізняється від відповідного значення у Clover, але є більш обґрунтованим, адже враховує покриття шляхів та різну вагу кожної метрики.

Маючи точність вимірювання на рівні інших засобів, MCover значно зручніший у використанні, тому що є окремим застосунком з графічним інтерфейсом та не потребує запуску вимірювання через середовище розробки чи командний рядок.

**Висновки**

Метрики покриття коду – корисний інструмент контролю якості тестування програмного забезпечення. Було розроблено методику вимірювання та реалізовано її у засобі MCover. Він враховує недоліки вже існуючих засобів, тому сприятиме підвищенню ефективності вимірювання метрик покриття.

Даний програмний продукт підвищить ефективність та зручність тестування програмного забезпечення, що дозволить утримувати якість продукту на належному рівні, зменшивши при цьому навантаження на людські та апаратні ресурси.

Зпоявою таких програмних засобів вимірювання покриття не буде вимагати попереднього навчання. Вимірювач не буде прив'язаний до певної платформи чи програмного забезпечення, маючи при цьому простий графічний інтерфейс.

**Список використаних джерел**


1. The Standish Group. Resolution type of projects and project cost [Electronicresource] – Access mode: http://www.standishgroup.com/sample_research/
2. Paul Johnson - Testing and Code Coverage[Electronicresource] – Access mode: http://homepage.hispeed.ch/pjcj/testing_and_code_coverage/paper.html
3. Atlassian Documentetion. About Code Coverage[Electronicresource] – Access mode: https://confluence.atlassian.com/display/CLOVER/About+Code+Coverage
4. EMMA Documrntation[Electronicresource] – Access mode:http://emma.sourceforge.net/
5. JaCoCoDocumrntation[Electronicresource] – Access mode: http://www.eclemma.org/jacoco/trunk/doc/
6. Cobertura Documrntation [Electronicresource] – Access mode: http://cobertura.github.io/cobertura/
7. The JMockit Testing Toolkit Tutorial – Measuring code coverage with JMockit Coverage [Electronicresource] – Access mode : http://jmockit.googlecode.com/svn/trunk/www/tutorial/CodeCoverage.html
8. Dmitriy Evdokimov –Light and Dark side of Code Instrumentation.
9. Atlassian Documentetion. Why does Cloveruse Source Code Instrumentation? [Electronicresource] – Access mode: https://confluence.atlassian.com/pages/viewpage.action?pageId=79986998
10. Інженерія програмного забезпечення 2012. Тези доповідей Міжнародної науково-практичної конференції аспірантів та студентів с. 11 / М.А. Книшик. Вимірювання метрик покриття коду, Київ 2012 – 37 с.
11. Brian Marick. How to Misuse Code Coverage, 1997 – 13 p.
12. Martin Fowler. Test Coverage, 2012 [Electronicresource] – Access mode: http://martinfowler.com/bliki/TestCoverage.html
13. Дмитрий Фазуненко. «70% чего?» или различные метрики измерения тестового покрытия [Электронный ресурс] – Режим доступа: http://addconf.ru/files/ADD-2/presentations/DmitryFazunenko.pdf


**Відомості про авторів**:

**Дишлевий Олексій Петрович** – старший викладач кафедри інженерії програмного забезпечення Інституту комп'ютерних інформаційних технологій. Наукові інтереси: емпірична інженерія програмного забезпечення.
E-mail: oleksiy.dyshlevyy@livenau.net

Книшик Марта Андріївна – студенка 4 курсу кафедри кафедри інженерії програмного забезпечення Інституту комп'ютерних інформаційних технологій. Наукові інтереси: емпірична інженерія програмного забезпечення.
E-mail: martaknyshyk@gmail.com