УДК 004.4

А.М. Акименко, канд. фіз.-мат. наук, доцент
Т.П. Бивойно, асистент
Чернігівський державний технологічний університет, м.Чернігів, Україна

ВИКОРИСТАННЯ МЕТРИК ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ КІЛЬКІСНОГО АНАЛІЗУ РИЗИКІВ ПРОГРАМНОГО ПРОЕКТУ

Розглянуто процес кількісного аналізу ризиків розробки програмного проекту. Визначено набір метрик, які можуть бути використані для проведення кількісного аналізу. Отримано формули для підрахунку кількісних показників ризиків, пов'язаних з функціональністю та складністю програмного проекту.

Постановка проблеми. Сучасна програмна індустрія за останній час накопичила значну колекцію моделей і метрик, що оцінюють окремі виробничі та експлуатаційні властивості програмного забезпечення (ПЗ). Проте неврахування області застосування розроблюваного ПЗ, ігнорування етапів життєвого циклу програмного забезпечення і, нарешті, необґрунтоване їх використання в різнопланових процедурах прийняття виробничих рішень, не дає можливості розробникам ПЗ отримувати якісну інформацію стосовно стану розробки програмного продукту, виявляти та зменшувати ризики, які виникають ще на початкових стадіях життєвого циклу.

Ці обставини потребують ретельного підходу до процесу вибору та застосування методів оцінки якості ПЗ, встановлення порядку їх використання, накопичення та інтеграції різнорідної метричної інформації для прийняття своєчасних виробничих рішень. Однією з найважливіших проблем є пошук універсальних інструментів, які б дали можливість отримати кількісну оцінку ризиків, що виникають під час створення програмного забезпечення, ще на ранніх стадіях життєвого циклу.

Кількісний аналіз ризиків проекту. *Кількісний аналіз ризиків* – процес числового аналізу впливу визначених ризиків на мету проекту.

Кількісний аналіз ризиків є однією з шести складових процесу управління ризиками, який визначено в PMBOK (Project Management Body Of Knowledge) [1]. Деякою мірою це ключовий етап, оскільки після нього розробник ПЗ має можливість приймати чіткі управлінські рішення, пов'язані з плануванням реакції на ризики та, відповідно, зменшувати їх вплив на кінцеві результати розробки.

Кількісний аналіз виконують з ризиками, для яких у процесі якісного аналізу було встановлено пріоритети їх виникнення та суттєвість впливу. Призначення кількісного аналізу ризиків – визначення результату такого впливу та присвоєння кількісного значення ризикам.

За кількісного аналізу ризиків найчастіше використовують методи теорії ймовірностей, моделювання Монте-Карло й аналіз дерева рішень. Ці методи застосовують для:

- обчислення ймовірності досягнення результатів проекту;
- ідентифікації ризиків, які потребують найбільшої уваги, через обчислення їхнього відносного внеску у загальний ризик проекту;
- обчислення реалістичних витрат, графіка та результату при заданих значеннях ризиків;
- визначення найкращих управлінських рішень у разі невизначеності.

Формулювання цілей статті (постановка завдання). Метою цієї статті є вивчення питання про визначення кількісних показників для ризиків програмного проекту, пов'язаних з функціональністю та складністю програмного забезпечення.

Метрики програмного забезпечення. Сьогодні застосовується широкий спектр метрик програмного забезпечення. Природно, що існуючі метрики мають ті чи інші аспекти використання. Деякі з них дуже рідко використовуються на практиці, для інших

неможливо автоматизувати процес вимірювання, деякі мають дуже вузьку спеціалізацію. Однак існують метрики, які застосовуються досить часто, особливо в області об'єктно-орієнтованого програмування.

Цей клас метрик з'явився в процесі розвитку об'єктно-орієнтованих мов програмування. У цій групі найбільш часто використовуваними є набори метрик Чідамбера-Кемерера, Лоренца-Кідда та Ф. Абреу.

Набір Чідамбера-Кемерера найбільш часто цитується в програмній індустрії та наукових дослідженнях.

Метрики Лоренца і Кідда – результат практичного, промислового підходу до оцінки об'єктно-орієнтованих(ОО) проектів.

Набір метрик MOOD (Metrics for Object Oriented Design), запропонований Ф. Абреу в 1994 році, – інший приклад академічного підходу до оцінки якості ОО-проектування.

Оскільки метою статті є вивчення питання про кількісний аналіз ризиків розробки реальних програмних проектів, які можуть бути ідентифіковані на ранніх стадіях життєвого циклу, тому найбільш цікавим, з цієї точки зору, буде набір метрик який: по-перше, базується на результатах практичних досліджень у сфері розробки ОО-проектів, по-друге, орієнтований на логічну складову програмного продукту, а саме, структуру класів, яка і відображає статичне представлення концептуальної підмоделі системи в термінах об'єктно-орієнтованої моделі. Таким набором є набір метрик Лоренца-Кідда, який ми розглянемо більш детально.

Лоренц і Кідд виділяють три групи метрик:

1. Метрики, орієнтовані на класи.
2. Операційно-орієнтовані метрики.
3. Метрики для ОО-проектів.

Розглянемо першу групу.

Метрики, що включені до першої групи, розподілені на чотири категорії: метрики розміру, метрики спадкування, внутрішні та зовнішні метрики.

Метрики розміру засновані на підрахунку властивостей і операцій для окремих класів, а також їхніх середніх значень для всієї ОО-системи. Метрики спадкування акцентують увагу на способі повторного використання операцій в ієрархії класів. Внутрішні метрики класів розглядають питання зв'язності і кодування. Зовнішні метрики досліджують зчеплення і повторне використання.

Метрика 1. Розмір класу CS (Class Size). Загальний розмір класу визначається за допомогою таких вимірів:

– загальна кількість операцій (разом з приватними й успадкованими операціями), які інкапсулюються всередині класу;

– кількість властивостей (разом з приватними й успадкованими властивостями), які інкапсулюються класом.

Рекомендоване значення $CS \leq 20$.

Метрика 2. Кількість операцій, перевизначених підкласом, NOO (Number of Operations Overridden by a Subclass). Перевизначенням називають випадок, коли підклас заміщує операцію, успадковану від суперкласу, своєю власною версією.

Рекомендоване значення $NOO \leq 3$.

Метрика 3. Кількість операцій, доданих підкласом, NOA (Number of Operations Added by a Subclass). Рекомендоване значення $NOA \leq 4$ (для класа-листа).

Метрика 4. Індекс спеціалізації SI (Specialization Index). Забезпечує оцінку ступеня спеціалізації кожного підкласу. Розраховується за формулою:

$$SI = (NOO \times \text{рівень}) / M_{заг}, \quad (1)$$

де *рівень* – номер рівня в ієрархії класів, на якому знаходиться підклас, $M_{заг}$ – загальна кількість методів класу.

Рекомендоване значення SI ≤0,15.

Друга група «Операційно-орієнтовані метрики» орієнтована на оцінку операцій у класах.

Метрика 5. Середній розмір операції OS_{AVG} (Average Operation Size). Рекомендоване значення $OS_{AVG} \leq 9$.

Метрика 6. Складність операції OC (Operation Complexity). Рекомендоване значення OC ≤ 65.

Метрика 7. Середня кількість параметрів на операцію NP_{AVG} (Average Number of Parameters per operation). Рекомендоване значення $NP_{AVG} = 0,7$.

Третя група метрик призначена для визначення прогнозної оцінки розміру програмного продукту.

Метрика 8. Кількість описів сценаріїв NSS (Number of Scenario Scripts). Для визначення цієї метрики необхідно задіяти інші діаграми, які використовуються для опису моделі програмної системи (діаграма варіантів використання). Рекомендоване значення NSS – не менше одного сценарію на публічний протокол підсистеми, що відображає основні функціональні вимоги до підсистеми.

Метрика 9. Кількість ключових класів NKC (Number of Key Classes). Ключовий клас прямо пов'язаний з проблемною областю, для якої призначена система. Тому значення NKC достовірно відображає майбутній обсяг розробки. В типовій ОО-системі на частку ключових класів припадає від 20 до 40 відсотків від загальної кількості класів.

Рекомендоване значення: 0,2 ≤ NKC ≤ 0,4 від загальної кількості класів системи.

Метрика 10. Кількість підсистем NSUB (Number of SUBsystem). Рекомендоване значення NSUB > 3.

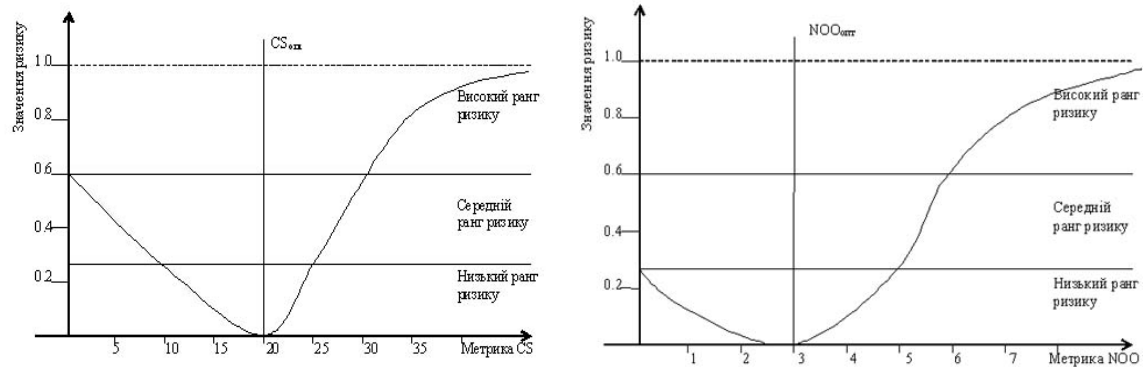
Визначення кількісних показників ризику. Оскільки всі метрики з наведеного набору пов'язані зі структурою класів ОО-системи, є очевидною можливість використовувати, як вхідну інформацію для визначення значення показників метрик діаграми класів, розробленої за допомогою мови моделювання UML. Як відомо, діаграми UML можуть бути використані в процесах ідентифікації та якісного аналізу ризиків розробки ПЗ на ранніх стадіях життєвого циклу [2, 3]. Діаграма класів, у свою чергу, відображає попередню архітектуру програмного проекту та дозволяє ідентифікувати ризики, пов'язані з функціональністю та складністю [3] програмного продукту.

Розглянемо питання про ризики, пов'язані з метриками набору Лоренцо-Кідда, та визначення їх кількісних характеристик.

Метрика CS. Велике значення метрики CS вказує на те, що клас має дуже багато обов'язків. Це зменшує можливість повторного використання класу та ускладнюють його реалізацію та тестування. Для отримання більш якісної кількісної характеристики метрики має сенс обрахування середньозваженого значення для всього проекту $CS_{сзв}$. У цьому випадку, чим менше середнє значення метрики, тим більша ймовірність повторного використання класу. Кількісне значення ризику $\mathbf{P_{CS}}$ для відповідних значень метрики визначають згідно з графіком, наведеним на рисунку1, *а*. Значення $\mathbf{P_{CS}}$ дозволяє визначити ранг ризику, пов'язаному зі складністю ПЗ.

Метрика NOO. Велике значення NOO говорить про те, що існують проблеми проектування. Зрозуміло, що підклас повинен розширяти операції суперкласу. Розширення проявляється у вигляді нових імен операцій. Якщо ж NOO велика, то розробник порушує абстракцію суперкласу. Це послаблює ієрархію класів, ускладнює тестування та модифікацію програмного забезпечення. Іншими словами, збільшує складність програмного продукту. Кількісне значення ризику $\mathbf{P_{NOO}}$ для відповідних значень метрики ви-

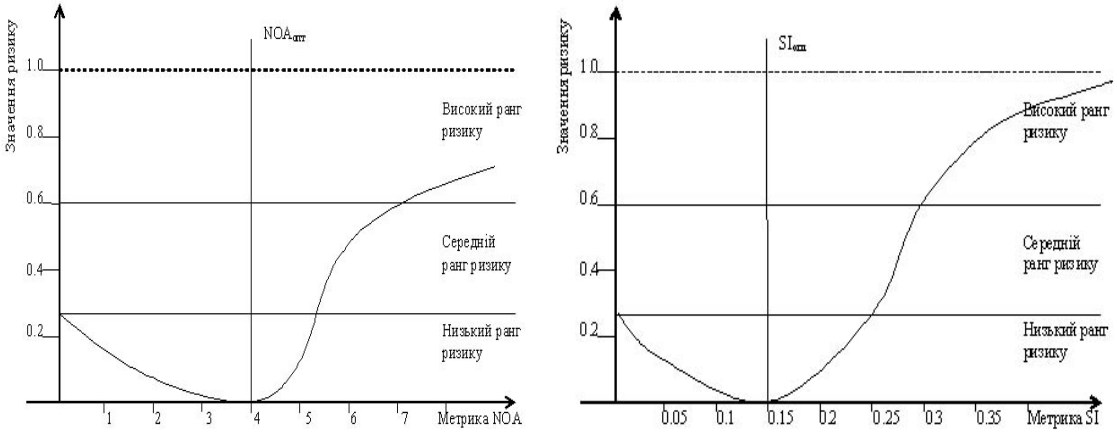
значають згідно з графіком, наведеним на рисунку 1, *б*. Значення $\mathbf{P_{NOO}}$ дозволяє визначити ранг ризику, пов'язаному зі складністю ПЗ.



а) б)

Рис. 1. Визначення параметрів ризиків P_{CS} та P_{NOO}

Метрика NOA. Підкласи спеціалізуються додаванням приватних операцій і властивостей (атрибутів). Зі зростанням NOA підклас віддаляється від абстракції суперкласу. Під час збільшення глибини ієрархії класів повинно зменшуватися і значення NOA на нижчих рівнях ієрархії. Кількісне значення ризику $\mathbf{P_{NOA}}$ для відповідних значень метрики визначають згідно з графіком, наведеним на рисунку 2, *а*. Значення $\mathbf{P_{NOA}}$ дозволяє визначити ранг ризику, пов'язаному зі складністю ПЗ.



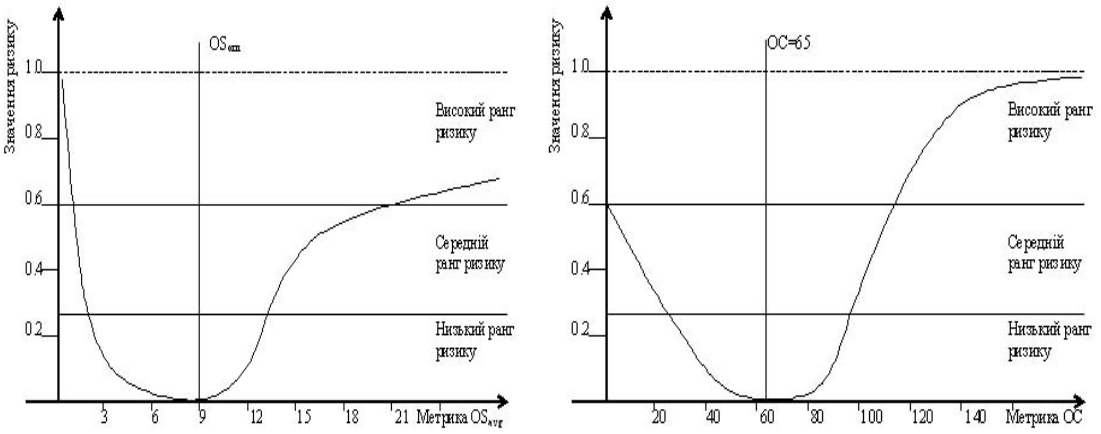
а) б)

Рис. 2. Визначення параметрів ризиків P_{NOA} та P_{SI}

Метрика SI. Чим вище значення SI, тим більша ймовірність того, що в ієрархії класів є класи, що порушують абстракцію суперкласу. Кількісне значення ризику $\mathbf{P_{SI}}$ для відповідних значень метрики визначають згідно з графіком, наведеним на рисунку 2*б*. Значення $\mathbf{P_{SI}}$ дозволяє визначити ранг ризику, пов'язаному зі складністю ПЗ.

Метрика $\mathbf{OS_{AVG}}$. Збільшення значення метрики означає, що обов'язки розміщені в класі не досить вдало. Кількісне значення ризику $\mathbf{P_{OS}}$ для відповідних значень метрики визначають згідно з графіком, наведеним на рисунку 3, *а*. Значення $\mathbf{P_{OS}}$ дозволяє визначити ранг ризику, пов'язаному з функціональністю ПЗ.

Метрика ОС. Бажано, щоб значення метрики не перевищувало рекомендованого. Кількісне значення ризику $\mathbf{P_{OC}}$ для відповідних значень метрики визначають згідно з графіком, наведеним на рисунку 3, *б*. Значення $\mathbf{P_{OC}}$ дозволяє визначити ранг ризику, пов'язаному з функціональністю ПЗ.



а) б)

Рис. 3. Визначення параметрів ризиків P_{OS} та P_{OC}

Метрика NP_{AVG}. Чим більше параметрів має операція, тим складніша взаємодія між об'єктами. З цієї причини значення NP_{AVG} не повинне перевищувати рекомендоване значення. Кількісне значення ризику $\mathbf{P_{NP}}$ для відповідних значень метрики визначають згідно з графіком, наведеним на рисунку 4. Значення $\mathbf{P_{NP}}$ дозволяє визначити ранг ризику, пов'язаному з функціональністю ПЗ та має вплив на складність програмного продукту.

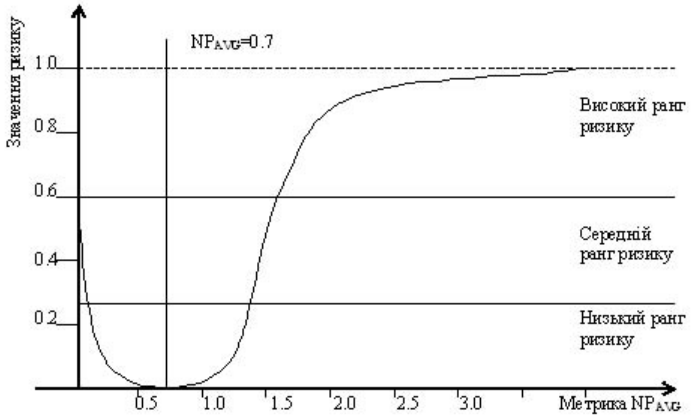
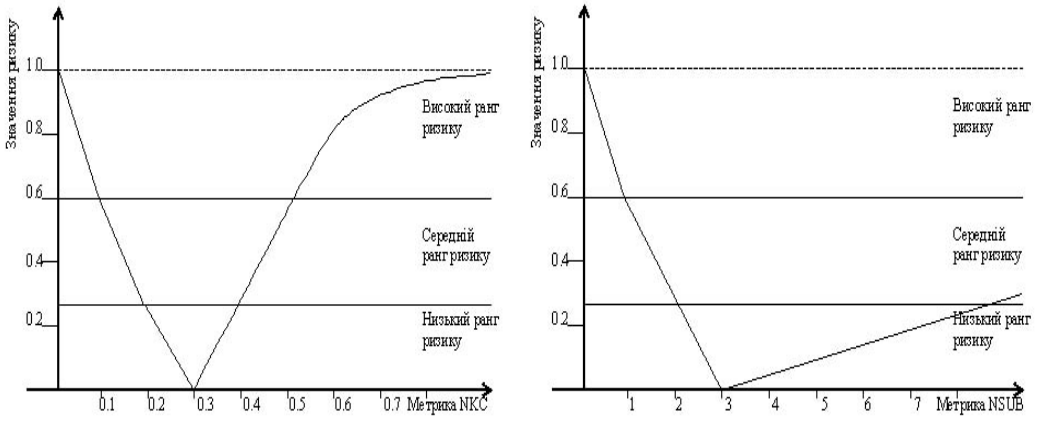


Рис. 4. Визначення параметра ризику P_{NP}

Метрика NKC. Значення NKC достовірно відображає майбутній обсяг розробки. Кількісне значення ризику $\mathbf{P_{NKC}}$ для відповідних значень метрики визначають згідно з графіком, наведеним на рисунку 5, *а*. Значення $\mathbf{P_{NKC}}$ дозволяє визначити ранг ризику, пов'язаному зі складністю ПЗ.



а) б)

Рис. 5. Визначення параметра ризику P_{NKC} та P_{NSUB}

Метрика NSUB. Кількість підсистем забезпечує розуміння наступних питань: розміщення ресурсів, планування, загальні затрати на інтеграцію. Кількісне значення ризику $\mathbf{P_{NSUB}}$ для відповідних значень метрики визначають згідно з графіком, наведеним на рисунку 5, *б*. Значення $\mathbf{P_{NSUB}}$ дозволяє визначити ранг ризику, пов'язаному зі складністю ПЗ.

Згідно зі статистичним способом підрахунку ймовірності та на основі вищевикладеного, ми маємо можливість визначити формули для отримання кількісних характеристик ризиків, пов'язаних зі складністю програмного продукту, що розробляється, та його функціональністю. Отже, маємо формулу для визначення $\mathbf{P_{СКЛ}}$:

$$P_{СКЛ} = \frac{P_{CS} + P_{NOO} + P_{NOA} + P_{SI} + P_{NKC} + P_{NSUB} + K_{NP}^{C} P_{NP}}{7}, \quad (2)$$

де K_{NP}^{C} – коефіцієнт впливу метрики P_{NP} на складність програмного продукту.

Формула для визначення $\mathbf{P_{ФУНК}}$ має вигляд:

$$P_{ФУНК} = \frac{P_{OC} + P_{OS} + K_{NP}^{Ф} P_{NP}}{3}, \quad (3)$$

де $K_{NP}^{Ф}$ – коефіцієнт впливу метрики P_{NP} на функціональність програмного продукту. Коефіцієнти K_{NP}^{C} та $K_{NP}^{Ф}$ пов'язані формулою

$$K_{NP}^{C} = 1 - K_{NP}^{Ф}. \quad (4)$$

Висновки дослідження:

- застосування метрик для оцінки об'єктно-орієнтованого програмного забезпечення дає можливість оцінити кількісні показники ризиків програмного проекту;
- метрики набору Лоренца-Кідда дають можливість визначити кількісні значення параметрів ризиків, пов'язаних зі складністю та функціональністю програмного забезпечення;
- застосування набору метрик Лоренца-Кідда спрощує ідентифікацію ризиків та дозволяє приймати управлінські рішення, ґрунтуючись на чітко визначених кількісних даних.

Список використаних джерел

1. Project Management Body of Knowledge (PMBOK), PMI Standard Committee.
2. Акименко А. М. Використання UML під час проектування складних програмних систем / А. М. Акименко // Вісник Чернігівського державного технологічного університету. Серія "Технічні науки": наук. зб. – 2011. – № 49. – С. 164-170.
3. Акименко А. М. Ідентифікація ризиків програмного проекту / А. М. Акименко // Вісник Чернігівського державного технологічного університету. Серія "Технічні науки": наук. зб. – 2011. – № 53. – С. 171-175.

УДК 004.02

А.Н. Волокита, канд. техн. наук
Ву Дык Тхинь, аспирант
А.Ю. Якушев, студент
Национальный технический университет Украины «КПИ», г. Киев, Украина

ОБНАРУЖЕНИЕ ВТОРЖЕНИЙ В РАСПРЕДЕЛЕННЫЕ КОМПЬЮТЕРНЫЕ СИСТЕМЫ НА ОСНОВЕ ГЕНЕТИЧЕСКОГО ПРОГРАММИРОВАНИЯ

В данной статье предложено использование генетического программирования (ГП) для решения задачи обнаружения вторжений в распределенные компьютерные системы. Сделано анализ существующих методов оптимизации выполнения ГП, проведено экспериментальное исследование эффективности ГП при поиске символьного выражения третьего порядка.

Постановка проблемы. Распределенная компьютерная система (РКС) – группа объединенных в сеть компьютеров, представляющаяся их пользователям единой объединен-