УДК 004.412:519.237.5

ОЦІНЮВАННЯ СКЛАДНОСТІ ОБ'ЄКТНО-ОРІЄНТОВАНОГО ПРОЄКТУВАННЯ JAVA-ЗАСТОСУНКІВ, СТВОРЕНИХ ЗА ДОПОМОГОЮ ФРЕЙМВОРКУ SPRING, З ВИКОРИСТАННЯМ МЕТРИК RFC, WMC ТА LCOM

Латанська Л.О., к.ф.-м.н., доцент[1]; Давидов Д.В.[2]
[1,2] Національний університет кораблебудування імені адмірала Макарова
[1,2] Україна, Миколаїв
[1] liudmyla.latanska@nuos.edu.ua; [2] den0802200303@gmail.com

Анотація. *В роботі для оцінювання складності об'єктно-орієнтованого проєктування Java-застосунків, створених за допомогою фреймворку Spring, використано нормалізовані дані з СК метрик, що характеризують семантику класів, та побудовано еліпсоїд передбачення.*
Ключові слова: *еліпсоїд передбачення; складність ООП; семантика класів; Java-застосунки; Spring Framework.*

Вступна частина. Оцінювання складності об'єктно-орієнтованого проєктування (ООП) відіграє важливу роль в індустрії розробки програмного забезпечення, оскільки допомагає вчасно виявляти потенційні проблеми архітектури, передбачити можливі труднощі супроводу та визначити потребу у рефакторингу.

Серед існуючих підходів до оцінювання складності ООП можна виділити метричний аналіз. Одними з найпоширеніших метрик для оцінювання складності об'єктно-орієнтованого проєктування є метрики Чидамбера-Кемерера [1], зокрема RFC (відгук для класу), WMC (зважені методи на клас) та LCOM (ступінь зв'язності методів), які характеризують семантику класів. Перевищення допустимих значень цих показників [2, 3] може свідчити про занадто високу ступінь взаємодії одного класу із іншими класами, надмірну складність класів, низьку зв'язність або перевантаження окремих компонентів системи. Традиційно складність ООП визначається на основі окремих метрик. Нами пропонується використовувати комплексний підхід, який враховує взаємозв'язок між метриками RFC, WMC та LCOM і дозволяє більш об'єктивно оцінити структурні властивості системи, виявити надмірну складність або низьку когезію та оптимізувати архітектуру програмного продукту.

Особливої актуальності оцінювання складності ООП набуває для Java-застосунків, створених за допомогою фреймворку Spring, де об'єктно-орієнтована структура та інтенсивна взаємодія між компонентами створюють додаткові виклики для вказаного оцінювання. Тому побудова математичної моделі для оцінювання складності об'єктно-орієнтованого проєктування Java-застосунків, створених за допомогою фреймворку Spring, на основі семантики класів є доцільним напрямом дослідження, що сприятиме удосконаленню методів аналізу та підвищенню ефективності процесу проєктування.

Метою дослідження є побудова еліпсоїда передбачення для оцінювання складності ООП Java-застосунків, створених за допомогою фреймворку Spring, із урахуванням семантики класів.

Основна частина. Для проведення дослідження на ресурсі GitHub (https://github.com) знайдено 79 Java-застосунків, що створені за допомогою фреймворку Spring. З використанням програмного забезпечення CK Metrics [4] отримано набір даних з метрик RFC, WMC, LCOM на рівні застосунку. У таблиці 1 наведено описову статистику цих даних.

Таблиця 1 – Описова статистика набору даних

Метрика	Min	Max	Mean	RMSD
Кількість класів	4	760	85,152	72,898
RFC	0,666	14,864	6,654	2,375
WMC	1,333	17,222	6,047	2,575
LCOM	0,083	420,481	22,268	23,02

Перед перевіркою тривимірних даних на наявність викидів проведено тест Мардіа, що базується на оцінюванні багатовимірної асиметрії та ексцесу. Згідно з цим тестом, набір даних не є гаусівським. Для нормалізації даних використано тривимірне перетворення Бокса-Кокса. Для пошуку викидів виконано порівняння квадрату відстані Махаланобіса для кожної багатовимірної точки даних із значенням статистики [5]

$$\frac{3(N^2-1)}{N(N-3)}F_{3,N-3,\alpha},$$

де N – кількість точок даних; $F_{3,N-3,\alpha}$ – квантиль F-розподілу з 3 та N-3 ступенями вільності, α – рівень значущості.

У нашому випадку у 79 точках даних не було виявлено жодного викиду.

Для нормалізованих даних з метрик RFC, WMC та LCOM побудовано еліпсоїд передбачення. Методика побудови еліпсоїда передбачення та алгоритм його використання для оцінювання складності ООП Java-застосунків, створених за допомогою фреймворку Spring, із урахуванням семантики класів аналогічні представленим в [5].

Висновки. Для оцінювання складності ООП Java-застосунків, створених за допомогою фреймворку Spring, побудовано математичну модель у вигляді еліпсоїда передбачення для нормалізованих даних з метрик RFC, WMC та LCOM на рівні застосунку. У подальших етапах планується розробити програмне забезпечення для реалізації моделі.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

[1] Chidamber, S., & Kemerer, C. (1994). A metrics suite for object oriented design. *IEEE Transactions on Software Engineering, 20*(6), 476-493.

[2] Deligiannis, I., Shepperd, M., Roumeliotis, M., & Stamelos, I. (2003). An empirical investigation of an object-oriented design heuristic for maintainability. *Journal of Systems and Software*, 65(2), 127-139. https://doi.org/10.1016/S0164-1212(02)00054-7

[3] Subramanyam, R., & Krishnan, M. (2003). Empirical analysis of CK metrics for object-oriented design complexity: Implications for software defects. *IEEE Transactions on Software Engineering*, 29(4), 297-310. https://doi.org/10.1109/TSE.2003.1191795

[4] Aniche, M. (2015). *Java code metrics calculator (CK) [GitHub repository]*. Retrieved October 15, 2025, from https://github.com/mauricioaniche/ck

[5] Prykhodko, A. S., & Malakhov, E. V. (2024). Determining object-oriented design complexity due to the identifications of classes of open-source web applications created using PHP frameworks. *Radio Electronics, Computer Science, Control,* 2, 160-167. https://doi.org/10.15588/1607-3274-2024-2-16

Latanska L.O., Davydov D.V.

Evaluation of the complexity of object-oriented design of Java applications created with the Spring Framework using RFC, WMC and LCOM metrics

Abstract. *In this work, normalized data from the SC metrics characterizing class semantics was used to assess the complexity of object-oriented design of Java applications created using the Spring framework, and a prediction ellipsoid was constructed.*

Keywords: *prediction ellipsoid; OOP complexity; class semantics; Java applications; Spring Framework.*