УДК 004.8

Савчук Т.О.
Вінницький національний технічний університет

Приймак Н.В.
Вінницький національний технічний університет

ОБҐРУНТУВАННЯ ДОЦІЛЬНОСТІ ВИКОРИСТАННЯ МЕТОДУ FP-GROWTH (FPG) ПІД ЧАС РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

У роботі здійснено порівняльний аналіз методів генерації частих предметних наборів для пошуку асоціативних правил. Обґрунтовано доцільність використання методу FP-Growth (FPG) для пошуку асоціативних правил під час розробки програмного забезпечення. Наведено основні етапи процесу розробки програмного забезпечення. Описано методи інтелектуального аналізу даних, що можуть використовуватися під час розробки програмного забезпечення. Обґрунтовано доцільність пошуку асоціативних правил під час розробки програмного забезпечення.

Ключові слова: *програмне забезпечення, етапи розробки програмного забезпечення, метод FP-Growth, асоціативні правила, інтелектуальний аналіз даних.*

Постановка проблеми. Розробка програмного забезпечення (ПЗ) – це комплексний процес створення комп'ютерних програм, в якому беруть участь розробники, бізнес-аналітики, дизайнери, інженери з якості та менеджери, що утворюють команду розробки [1]. Для ефективного планування роботи такої команди та для управління проектом менеджерам потрібно визначити тривалість кожного з етапів розробки ПЗ.

З метою покращення функціональності або виправлення дефектів код програмного продукту змінюється в процесі розробки. Усі зміни фіксуються в системах управління версіями коду, які дають змогу використовувати накопичену інформацію для подальшого аналізу. Оскільки обсяг таких даних неможливо проаналізувати вручну, то виникає потреба в автоматизованому аналізі. Для цього використовують різноманітні технології інтелектуального аналізу даних.

Аналіз останніх досліджень і публікацій. Розробка програмного забезпечення здійснюється із використанням різноманітних технологій, в основу яких покладено відповідні моделі.

Будь-яка з моделей процесу розробки програмного забезпечення складається з таких етапів (рисунок 1) [2, с. 210]:

1. Аналіз, під час якого відбувається дослідження предметної галузі і визначення вимог до ПЗ. Кінцевий результат – специфікація на розроблюваний програмний продукт.

2. Проектування, під час якого відбувається визначення внутрішньої будови та функціонування майбутньої програми з точки зору розробника. Кінцевий результат – прототип розроблюваної програми.

3. Програмування (кодування, реалізація), під час якого створюється програмне забезпечення із використанням визначених мов програмування та засобів. Кінцевий результат – комп'ютерна програма.

4. Тестування та виправлення помилок, під час якого здійснюється виявлення та усунення дефектів, налагодження створеного продукту. Кінцевий результат – програма, що відповідає специфікації.

5. Документування, під час якого пишеться документація, описується майбутній продукт як з точки зору його створення, так і з точки зору його використання. Кінцевий результат – документація на розроблену програму.

Інформація, накопичена під час реалізації цих етапів, зберігається у репозитаріях з метою її подальшого аналізу. Для отримання корисних даних із накопиченої інформації доцільно застосовувати технології Data Mining [3, с. 57].

Основними методами інтелектуального аналізу даних (ІАД), що можуть використовуватися під час розробки програмного забезпечення, є [3, с. 60; 4]:

1. Інтелектуальний аналіз текстової інформації (листи, коментарі, документація), що дасть змогу визначати найактивніших учасників процесу розробки програмного забезпечення.

2. Кластеризація, що може використовуватися для визначення груп схожих модулів програм, базуючись на кількості зроблених модифікацій.

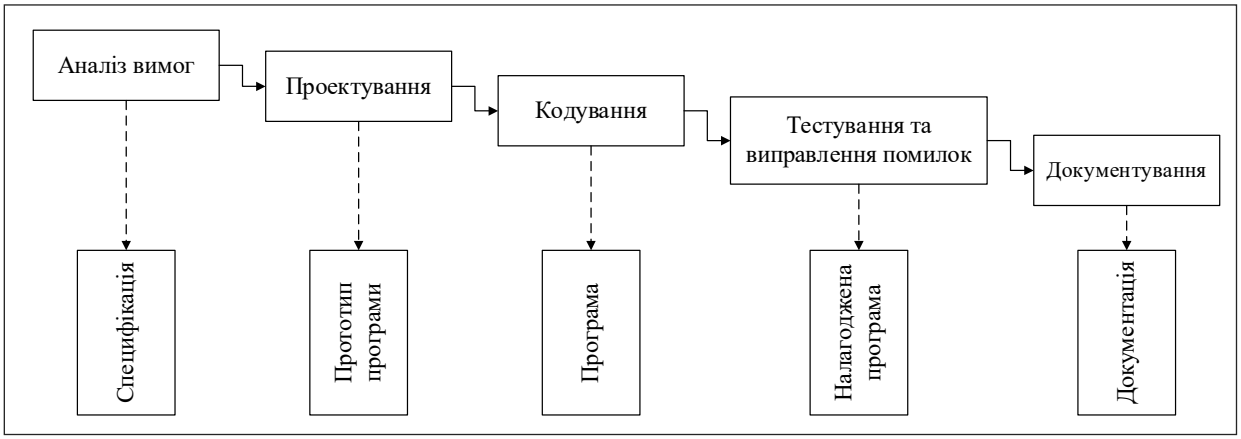

Рис. 1. Основні етапи розробки програмного забезпечення

3. Класифікація, що може використовуватися з метою визначення механізму ідентифікації вразливих модулів програмного забезпечення на основі атрибутів модулів чи системи.

4. Добування частих шаблонів або асоціативних правил, що можна використовувати для прогнозування розробника, який працюватиме над виправленням конкретної помилки в ПЗ. Такий підхід скоротить час, необхідний для підбору людських ресурсів, потрібних для виправлення дефектів.

5. Міркування на основі прецедентів, що може бути використане для знаходження необхідного набору тест-кейсів для валідації конкретного компонента, базуючись на даних, отриманих у разі тестування схожих компонентів раніше.

Пошук асоціативних правил дасть змогу досягти таких цілей під час розробки програмного забезпечення:

1) Визначення порушень в архітектурі програмного продукту.

У роботі [5] автори пропонують використовувати пошук асоціативних правил для визначення порушень в архітектурі об'єктно орієнтованого ПЗ.

Такий підхід допоможе визначати вразливі місця в архітектурі ПЗ на ранніх етапах його розробки, що дасть змогу уникнути додаткових матеріальних затрат на їх виправлення в майбутньому.

2) Визначення кількості необхідних ресурсів на розробку ПЗ.

У роботі [6] запропоновано застосовувати теорію нечітких множин та пошук асоціативних правил із використанням алгоритму Apriori для оцінки необхідних людських ресурсів у процесі розробки ПЗ.

Такий підхід дасть змогу менеджерам проекту ефективно планувати діяльність кожного учасника процесу розробки ПЗ.

3) Визначення розробника, якого буде призначено для виправлення знайденого дефекту.

У роботі [7] запропоновано використовувати методи пошуку асоціативних правил для визначення розробника, що виправлятиме знайдений дефект у програмі.

Такий підхід дасть змогу автоматизувати процес призначення розробника на виправлення конкретного дефекту та може бути використаний менеджерами проекту під час планування діяльності учасників команди розробки ПЗ.

Отже, задача пошуку асоціативних правил під час процесу розробки ПЗ є актуальною, а її вирішення допоможе полегшити процес розробки програм.

Постановка завдання. Метою цього дослідження є обґрунтування доцільності використання методу генерації частих предметних наборів FP-Growth для пошуку асоціативних правил під час розробки програмного забезпечення.

Виклад основного матеріалу дослідження. Асоціативні правила – це закономірності типу «із події X витікає подія Y», тобто $X \to Y$, при цьому обов'язково має виконуватися умова, що $X \cap Y \to \varnothing$ [8, с. 85].

Будь-яке асоціативне правило можна охарактеризувати двома числовими величинами [9, с. 244]:

1) Підтримкою $supp(X \to Y)$. Це величина, що дорівнює відношенню кількості записів $X \cup Y$ в БД до загального числа записів у БД.

2) Достовірністю $conf(X \to Y)$. Це величина, що дорівнює відношенню підтримки $supp(X \to Y)$ до підтримки $supp(X)$.

Задачу пошуку асоціативних правил під час розробки програмного забезпечення можна розділити на дві підзадачі:

1. Підзадачу генерації частих предметних наборів для пошуку АП під час розробки про-

грамного забезпечення, в яких рівень підтримки не нижче заданого експертом порогового значення $minsupp(X)$.

2. Підзадачу генерації асоціативних правил $X \to Y$ під час розробки програмного забезпечення, що мають рівень достовірності не нижче заданого експертом порогового значення $minconf(X \to Y)$.

Щоб вирішити підзадачу генерації частих предметних наборів для пошуку АП під час розробки ПЗ, використовують методи (рисунок 2), які можна класифікувати за способом цієї генерації:

– методи, що генерують кандидатів для частих предметних наборів [10, с. 487; 11–13];

– методи, що не виконують генерацію кандидатів перед пошуком частих предметних наборів [14].

Розглянемо кожен метод детальніше та здійснимо їх порівняльний аналіз (таблиця 1).

Set-oriented mining method [11].

Це найперший метод, що був запропонований для генерації частих предметних наборів. Обмеження: база даних є статичною.

Переваги: може застосовуватися до потужних БД, метод простий для розуміння.

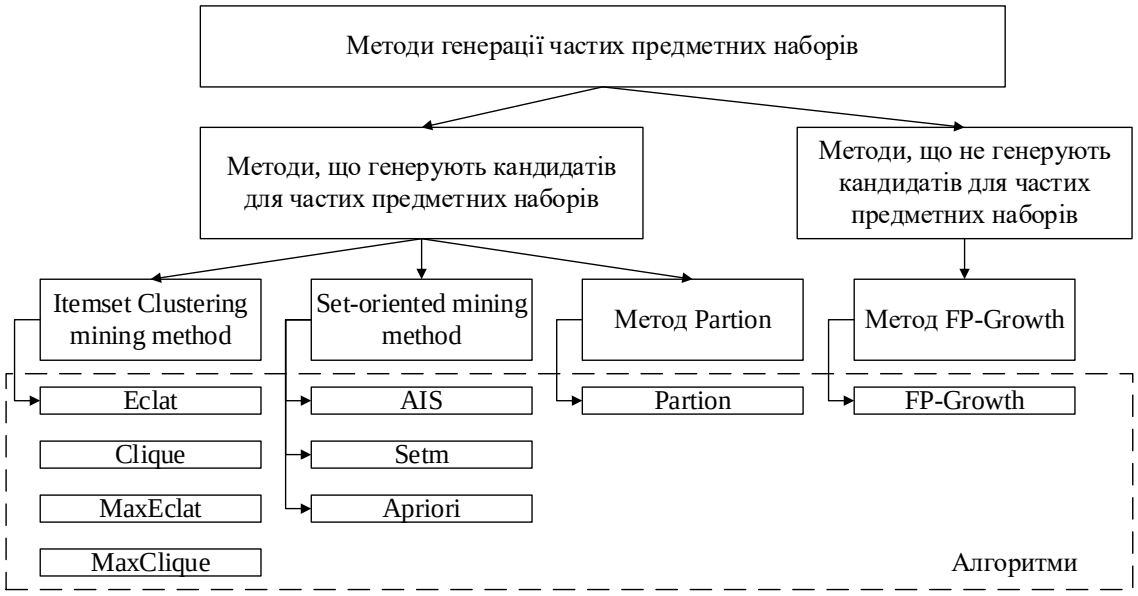

Рис. 2. Методи генерації частих предметних наборів

Таблиця 1

Порівняння характеристик методів генерації частих предметних наборів для пошуку асоціативних правил

Метод	Переваги	Недоліки
Set-oriented	– Можна застосовувати до потужних БД; – метод простий для розуміння	– Процес генерації можливих частих предметних наборів тривалий та використовує пам'ять комп'ютера; – виконується багаторазове сканування БД; – можливість роботи лише зі статичними даними
Partion	– Швидкий пошук у потужних БД; – лише два сканування БД	– Потреба у комп'ютері з великою кількістю оперативної пам'яті; – недосконалий підхід поділу БД на частини на першому етапі
Itemset Clustering	– Використання невеликої кількості оперативної пам'яті, – Немає необхідності у побудові хеш-структури, – лише одне сканування БД	– Необхідність здійснювати вибір параметрів для алгоритму кластеризаці
FP-Growth	– Розмір FP-дерева досить малий, – висока швидкість пошуку частих предметних наборів	– Розмір FP-дерево не може бути більшим, ніж пам'ять комп'ютера, – два сканування БД, що може призвести до нетривіальних витрат, – обробка лише статичних даних

Недоліки: процес генерації можливих частих предметних наборів тривалий та використовує пам'ять комп'ютера, виконується багаторазове сканування БД, можливість роботи лише зі статичними даними.

Метод Partion [12].

Метод доцільно застосовувати для пошуку асоціативних правил у потужних БД. Основною його відмінністю від Set-oriented mining method є те, що перед пошуком частих предметних наборів БД розділяється на декілька частин, кожна з яких обробляється окремо. Обмеження: дані в БД зберігаються відсортованими в лексикографічному порядку у форматі $\langle TID, it \rangle$, де TID – це унікальний ідентифікатор транзакції, що розглядається, it – окремі елементи, з яких складається транзакція.

Переваги: метод дає змогу здійснювати швидкий пошук у потужних БД, відбувається лише два сканування БД для генерації частих предметних наборів, що зменшує використання фізичної пам'яті комп'ютера.

Недоліки: виконання цього методу потрібно здійснювати на комп'ютері з великою оперативної пам'яттю, підхід поділу БД на частини на першому етапі недосконалий.

Itemset Clustering mining method [13].

Це метод, в якому спочатку за допомогою методів кластеризації (кластеризація на основі класів еквівалентності, maximal Hypergraph Clique кластеризація) генеруються потенційні максимальні часті предметні набори, що складаються, як мінімум із 2 елементів. Для генерації кінцевих частих предметних наборів використовуються методи проходження по графу: з низу до верху та гібридний. Обмеження: дані в БД мають бути представлені у вертикальному вигляді, що дасть змогу їх кластеризувати за одне проходження по БД.

Переваги: використовується мало оперативної пам'яті, оскільки відбувається лише одне сканування БД, не потрібно будувати складну хеш-структуру для генерації частих предметних наборів.

Недоліки: необхідно вибрати параметри для алгоритму кластеризації.

До методів, що не здійснюють генерацію кандидатів для частих предметних наборів, належить FP-Growth.

Метод FP-Growth [14].

За використання цього методу вся інформація перетворюється на деревовидну структуру – FP-дерево, після проходження по якому знаходяться часті предметні набори. Обмеження: FP-дерево не може бути більшим за розмірами, ніж основна пам'ять комп'ютера, на якому виконується його створення.

Переваги: розмір FP-дерева досить малий, що дає змогу уникнути затратної процедури генерації кандидатів, швидкість пошуку частих предметних наборів вища, ніж у попередніх методах.

Недоліки: неможливо побудувати FP-дерево, що матиме розмір більший, ніж основна пам'ять комп'ютера. Хоча FP-дерево досить компактне, для його побудови потрібно двічі сканувати БД, що може являти собою нетривіальні витрати, дає змогу обробляти лише статичні дані.

Для вирішення підзадачі генерації асоціативних правил $X \to Y$ використовується один загальний підхід, який полягає в такому [10, с. 490]:

1. Для кожного частого набору довжиною w елементів всі можливі комбінації з $w-1$ елементів розглядаються як умова (X), а елемент, що залишився, як наслідок (Y).

2. Для кожної комбінації X та Y перевіряється, чи значення достовірності дорівнює або більше значення заданої мінімальної достовірності $minconf(X \to Y)$.

3. Якщо умова мінімальної достовірності виконується, асоціативне правило записується у вигляді $X \to Y$.

Як видно з таблиці 1, метод FP-Growth виконує генерацію частих предметних наборів швидше, ніж інші наявні методи, а також дає змогу обробляти потужні БД, отже, його варто застосовувати для генерації частих предметних наборів для пошуку асоціативних правил під час розробки програмного забезпечення.

Висновки. Отже, було проведено аналіз наявних методів генерації частих предметних наборів, що можуть бути використані для пошуку асоціативних правил під час розробки програмного забезпечення. Було з'ясовано, що для пошуку асоціативних правил під час розробки програмного забезпечення доцільно використовувати метод FP-Growth, оскільки він відповідає таким вимогам: дає змогу обробляти потужні БД; швидкість генерації частих предметних наборів досить висока.

На підставі проведеного аналізу можна зробити висновок, що метод FP-Growth потрібно удосконалити за рахунок здійснення класифікації даних перед пошуком асоціативних правил, що скоротить час їх пошуку. Удосконалений метод може бути використаний у створенні відповідної інформаційної моделі.

Список літератури:

1. BPC, Articles and Glossary. Application Development. URL: http://www.bestpricecomputers.co.uk/glossary/application-development.html (дата звернення: 10.08.2018)
2. Glass R. Fact and Fallacie of Software Engineering. Boston. 2003, 210 p.
3. Xie T., Thummalapenta S. Data Mining for Software Engineering. Computer Volume. 2009. pp.55–62.
4. Halkidi M., Tsatsaronis G. Data mining in software engineering. Intelligent Data Analysis – IOS Press. 2011. pp. 413–441.
5. Maffort C. Mining Architectural Patterns Using Association Rules. International Conference on Software Engineering and Knowledge Engineering. 2013. pp. 234–240.
6. Azzeh M. Software Stage-Effort Estimation Using Association Rule Mining and Fuzzy Set Theory. 2011. URL: https://arxiv.org/ftp/arxiv/papers/1703/1703.04539.pdf (дата звернення: 05.09.2018).
7. Sharma M., Kumari M. Bug Assignee Prediction Using Association Rule. ICCSA 2015. Lecture Notes in Computer Science. 2015. pp.444–457.
8. Зайко Т., Олійник А. Ассоциативные правила в интеллектуальном анализе данных. Вестник Национального технического университета Харьковский политехнический институт. 2013. С. 82–96.
9. Zhang C. Association Rule Mining, Models and Algorithms. Berlin. 2002. C.244.
10. Agrawal R. Srikant P. Fast Algorithms for Mining Association Rules in Large Databases. Proceedings of the 20th International Conference on Very Large Data Bases. 1994. pp. 487–499.
11. Agrawal R., Swami A. Mining Association Rules between Sets of Items in Large Databases. International Conference on Management of Data. 1993. pp. 207–216.
12. Savasere A., Omiecinski E. An Efficient Algorithm for Mining Association Rules in Large Databases. Proceedings of the 21th International Conference on Very Large Data Bases. 1995. pp. 432–444.
13. Hipp J. Algorithms for Association Rule Mining – A General Survey and Comparison. ACM SIGKDD Explorations Newsletter. 2000. pp. 58–64.
14. Han J., Pei J., Yin Y. Mining frequent patterns without candidate generation. SIGMOD '00 Proceedings of the 2000 ACM SIGMOD international conference on Management of data. 2010. pp. 1–12.

ОБОСНОВАНИЕ ЦЕЛЕСООБРАЗНОСТИ ИСПОЛЬЗОВАНИЯ МЕТОДА FP-GROWTH (FPG) ПРИ РАЗРАБОТКЕ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

В работе осуществлен сравнительный анализ методов генерации частых предметных наборов для поиска ассоциативных правил. Обоснована целесообразность использования метода FP-Growth (FPG) для поиска ассоциативных правил при разработке программного обеспечения. Приведены основные этапы процесса разработки программного обеспечения. Описаны методы интеллектуального анализа данных, которые могут использоваться при разработке программного обеспечения. Обоснована целесообразность поиска ассоциативных правил при разработке программного обеспечения.

Ключевые слова: *программное обеспечение, этапы процесса разработки программного обеспечения, метод FP-Growth, ассоциативные правила, интеллектуальный анализ данных.*

THE FEASIBILITY STUDY OF USING METHOD FP-GROWTH (FPG) DURING THE SOFTWARE DEVELOPMENT

In the paper a comparative analysis of the methods of frequent object sets generation for the associative rules search is performed. The expediency of using the FP-Growth (FPG) method for searching associative rules in software development is grounded. The main stages of the software development process are represented. The methods of data mining, which can be used in software development, are described. The expediency of associative rules search during the software development is substantiated.

Key words: *software, stages of the software development process, method FP-Growth, associative rules, data mining.*