УДК 04.416.6
DOI https://doi.org/10.36994/2788-5518-2024-02-08-06

ПРОБЛЕМА ПРОГНОЗУВАННЯ РОЗРОБКИ НОВОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ: ВИКОРИСТАННЯ РЕПОЗИТОРІЇВ ВІДКРИТОГО КОДУ ДЛЯ ПОБУДОВИ ПРОГНОСТИЧНИХ МОДЕЛЕЙ

Загорулько А. В., *аспірант, Відкритий міжнародний університет розвитку людини «Україна», Київ, Україна. https://orcid.org/0009-0004-9478-5642, azago85@gmail.com*

Павленко В. І., *кандидат фізико-математичних наук, доцент, Відкритий міжнародний університет розвитку людини «Україна», Київ, Україна. https://orcid.org/0000-0002-3958-0415, pavlenko.v@i.ua*

Анотація. *У поточному дослідженні було розглянуто проблему підвищення ефективності процесу розробки нового програмного забезпечення шляхом використання прогностичних моделей, побудованих на основі подій, що фіксуються протягом усього життєвого циклу продукту (SDLC). У сучасному конкурентному середовищі розробки програмного забезпечення оптимізація використання ресурсів і врахування попереднього досвіду є критичними для зменшення часу виходу продукту на ринок та підвищення його якості. В роботі пропонується використовувати події SDLC як надійну основу для планування термінів і обсягів випуску нових версій продукту. Також розглядається доцільність використання репозиторіїв відкритого коду як джерела подій SDLC у випадках, коли продукт перебуває на ранніх стадіях розробки й ще не має власної історії подій. Для дослідження було вибрано низку проєктів з відкритим кодом, що містять значну кількість історичних даних щодо змін коду, вирішення задач і виявлених дефектів, а саме: Spring, Mockito, Quarkus, DBeaver.*

Для апробації підходу було розроблено модель прогнозування дефектів, яка враховує обсяг виконаних задач, змін коду та його цикломатичну складність. Навчання моделі здійснювалося на основі даних проєктів групи Spring, а перевірка її точності – на базі даних репозиторіїв проєктів Mockito, Quarkus, DBeaver. У процесі дослідження було підтверджено ефективність моделі та можливість її застосування для прогнозування й покращення процесу розробки нового програмного забезпечення.

Модель також продемонструвала здатність адаптуватися до нових даних, що відкриває можливість її інтеграції у процеси безперервної інтеграції та розгортання для автоматизованого аналізу якості коду й планування випусків програмного продукту.

Результати дослідження свідчать про значний потенціал застосування машинного навчання для покращення SDLC, а також підтверджують доцільність використання репозиторіїв відкритого коду як надійного джерела даних.

Ключові слова: *CI, CD, прогнозування, програмне забезпечення, машинне навчання, SDLC.*

THE PROBLEM OF FORECASTING NEW SOFTWARE DEVELOPMENT: USING OPEN SOURCE REPOSITORIES TO BUILD FORECASTING MODELS

Anton Zahorulko, *Ph.D. student, Open International University of Human Development "Ukraine", Kyiv, Ukraine. https://orcid.org/0009-0004-9478-5642, azago85@gmail.com*

Volodymyr Pavlenko, *Ph.D., Open International University of Human Development "Ukraine", Kyiv, Ukraine. https://orcid.org/0000-0002-3958-0415, pavlenko.v@i.ua*

Abstract. *This study addresses the problem of improving the efficiency of the software development process by utilizing predictive models based on events recorded throughout the entire Software Development Life Cycle (SDLC). In today's competitive software development environment, optimizing resource utilization and leveraging past development experience are critical for reducing time-to-market and enhancing product quality.*

The study proposes using SDLC events as a reliable foundation for accurately planning release timelines and volumes of new product versions. It also examines the feasibility of utilizing open-source repositories as sources of SDLC events in cases where a product is in its early development stages and lacks its own event history. For the research, a selection of open-source projects containing a substantial amount of historical data on code changes, task resolution, and defect identification was analyzed, including Spring, Mockito, Quarkus, and DBeaver.

To validate the approach, a defect prediction model was developed, considering the volume of completed tasks, code changes, and its cyclomatic complexity. The model was trained using data from the Spring project group, while its accuracy was tested on repository data from Mockito, Quarkus, and DBeaver. The

study confirmed the model's effectiveness and its applicability for predicting and improving the software development process.

Additionally, the model demonstrated adaptability to new data, enabling its integration into continuous integration and deployment (CI/CD) pipelines for automated code quality analysis and release planning. The research findings indicate a significant potential for applying machine learning to enhance SDLC and confirm the feasibility of using open-source repositories as a reliable data source.

Key words: *CI, CD, forecasting, software, machine learning, SDLC.*

Вступ

В умовах швидкого розвитку технологій та зростання конкуренції у сфері розробки програмного забезпечення важливою стає оптимізація використання ресурсів і досвіду, накопиченого під час розробки попередніх продуктів. Можливість випуску програмного продукту належної якості в заплановані строки є одним з ключових факторів конкурентоспроможності продукту, даючи можливість отримати найбільший ефект від об'єднання зусиль технічного та маркетингового складників.

Вчасний випуск нових версій програмного продукту великою мірою залежить від ефективного управління процесом розробки, що включає планування, аналіз і прогнозування життєвого циклу програмного забезпечення (SDLC).

Події SDLC, такі як випуск нових версій продукту, розробка вимог, виконання задач розробниками, виправлення дефектів, зміна коду, тестування функціоналу продукту командою контролю якості, моніторинг помилок збірки, розгортання та функціонування продукту у виробничому середовищі, можуть слугувати базовими метриками ефективності SDLC. Зберігання та аналіз цих подій створює основу для прогнозування витрат ресурсів та аналізу ризиків, пов'язаних з випуском нових версій.

Прогнозування випуску ранніх версій програмного продукту вважається досить складною проблемою у зв'язку з відсутністю наявних даних подій SDLC для аналізу, тому ситуація з розбіжністю анонсованої та фактичної дати випуску перших версій продукту є досить частим випадком у світі розробки програмного забезпечення.

Для вирішення цієї проблеми пропонується використання сторонніх джерел даних подій SDLC. Одним з таких джерел даних можна вважати репозиторії відкритого коду [1; 2].

Репозиторії відкритого коду, для прикладу GitHub [3], який налічує понад 300 мільйонів відкритих репозиторіїв, є досить змістовним джерелом даних подій SDLC, що можуть бути використані для побудови та навчання прогностичних моделей. Серед наявних проєктів з відкритим кодом є багато таких, що широко використовуються як бібліотеки та платформи для розробки сучасного програмного забезпечення. Таким прикладом є Spring Framework та Spring Boot [4; 5], які є платформою для реалізації сервісних додатків, у тому числі заснованих на мікросервісній архітектурі.

Таблиця 1

Основні метрики репозиторіїв Spring Framework та Spring Boot

	Spring Boot	**Spring Framework**
Задачі	42623	30994
Версії	338	379
Зміни коду	42774	11880
Зміни файлів	178335	53270
Наявні дані з	2013	2003

Аналіз даних, що надають репозиторії відкритого коду, є доволі поширеним способом апробації гіпотез та побудови кореляцій, для прикладу, у статті [1] дослідники аналізують динаміку розвитку проєкту в часі у розрізі розвитку та співпраці спільноти розробників навколо проєкту, у статті [2] пропонується інструмент для виявлення критичних місць, що виникають під час еволюції кодової бази та вимагають більших зусиль з обслуговування.

Метою поточного дослідження є апробація використання репозиторіїв відкритого коду для побудови та тренування прогностичної моделі для оцінки кількості дефектів на основі обсягу виконаних задач, кількості змін коду та цикломатичної складності зміненого коду. Також оцінюється можливість застосування розробленої моделі на прикладі інших репозиторіїв відкритого коду.

Методологія дослідження

У поточному дослідженні розглядаються репозиторії відкритого коду, засновані на мові програмування Java. Для створення та тренування моделі були використані дані проєкту Spring-Framework [4] та Spring-Boot [5].

Для проведення дослідження **створено програмне забезпечення, що дозволяє**:

- Експортувати дані для побудови та навчання моделі за допомогою публічного GitHub REST API.
- Завантажити отримані дані в базу даних.
- Обрахувати цикломатичну складність та кількість змін Java класів.

– Підготувати дані для створення та тренування моделі у розрізі конкретної версії.
– Створити треновану модель на базі алгоритмів машинного навчання [6].
– Оцінити якість моделі стосовно множини даних, зібраних на базі інших репозиторіїв відкритого коду.
– Перевірити вплив додаткового навчання моделі на власних даних на точність прогнозування.

У контексті дослідження експортовано таку інформацію SDLC:
– задачі
– події задач
– зміни вихідного коду (commits)
– файли змін вихідного коду (стан файлу вихідного коду після внесення змін).

У поточному дослідженні побудовано модель даних (рис. 1), що включає у себе інформацію стосовно версії програмного продукту, задач, що були виконані впродовж створення нової версії, інформацію стосовно змін коду та складності цих змін на основі подій SDLC.

Рис. 1. Модель даних для побудови прогностичної моделі

Для прогнозування кількості дефектів у версії сформовані такі метрики відповідно до кожної наявної версії у розрізі даних:
– загальна кількість задач (фактор);
– загальна кількість змінених ліній коду (фактор);
– загальна цикломатична складність зміненого коду (фактор);
– загальна кількість дефектів (прогнозована величина).

Серед критеріїв вибору алгоритму машинного навчання можна виділити такі як:
– обмежена кількість записів (менше 100 версій);
– невелика кількість факторів (3 фактори);
– необхідність високої точності та стабільності моделі;
– можлива наявність нелінійних зв'язків.

Виходячи з наведених вище критеріїв, серед алгоритмів машинного навчання ми вибрали Random Forest [6] та Lasso [6].

Алгоритм **Random Forest** дозволяє опрацювати нелінійні залежності, є досить стабільним та точним для наявної множини даних та кількості факторів, не потребує нормалізації та масштабування даних, толерує нерівномірність розподілу даних, менш чутливий до шуму.

Алгоритм **Lasso** є досить стабільним та точним для наявної множини, характеру (числові) даних та кількості факторів, зменшує ризик мультиколінеарності між факторами та менш чутливий до шуму даних.

Результати дослідження

Для виконання дослідження ми експортували події SDLC з репозиторіїв Spring Boot та Spring Framework [4; 5] та створили прогностичну модель оцінки кількості дефектів залежно від кількості задач, включених до версії продукту, загальної кількості змінених строчок коду та цикломатичної складності змінених Java класів.

Побудована прогностична модель (табл. 2) дозволила досягти середньої абсолютної похибки (MAE) у 3%, а також виявити, що всі три фактори є значущими для прогнозування та мають приблизно однаковий вплив на результат.

Таблиця 2

Характеристики побудованих прогностичних моделей

	Random Forest	**Lasso**
MAE, %	2,67	3,28
Фактор кількості задач, %	36	0
Фактор кількості змінених строчок коду, %	34	75
Фактор цикломатичної складності змін, %	30	25

Для перевірки моделі ми залучили дані репозиторіїв відкритого коду, що належать групі проєктів Spring (Spring-Framework) та інші (табл. 3), що не належать групі проєктів Spring. Залучення репозиторіїв, що не належать до групи проєктів Spring, є важливим для перевірки можливості застосування побудованої моделі для аналізу SDLC програмних продуктів, створених іншими командами розробки.

Таблиця 3

Основні метрики репозиторіїв, залучених для перевірки отриманих моделей

	Mockito	**DBeaver**	**Quarkus**
Задачі	42623	30994	42266
Версії	338	379	343
Зміни коду	42774	11880	37027
Зміни файлів	178335	53270	239281
Наявні дані з	2013	2003	2018

Результати перевірки прогностичних моделей, побудованих за допомогою репозиторію відкритого коду Spring-Boot, для репозиторіїв відкритого коду наведені нижче в таблицях 4, 5.

Таблиця 4

Результати перевірки побудованої прогностичної моделі на базі алгоритму Random Forest

Random Forest	**Spring-Framework**	**Mockito**	**DBeaver**	**Quarkus**
MAE, %	4	2,40	8,18	3,63
MAE, % з донавчанням моделі на даних репозиторію, що перевіряється	3,75	2,21	7,44	3,45

Таблиця 5

Результати перевірки побудованої прогностичної моделі на базі алгоритму Lasso

Lasso	**Spring-Framework**	**Mockito**	**DBeaver**	**Quarkus**
MAE, %	6,54	4,65	7,71	4,5

Аналіз результатів

Отримані результати показують достатню точність побудованих моделей, яка варіюється від 3% до 8% залежно від алгоритму та репозиторіїв відкритого коду.

Модель побудована за допомогою алгоритму Random Forest є точнішою, ніж модель на базі алгоритму Lasso.

Експерименти з донавчанням Lasso моделі не проводились, оскільки цей алгоритм не підтримує таку можливість, єдиний варіант навчання в такому випадку – це створення нової моделі на базі множини даних обох репозиторіїв, що не має сенсу в контексті поточного дослідження.

Можна відзначити, що алгоритм Lasso, на відміну від алгоритму Random Forest, знехтував фактором кількості задач у версії та істотно виділив фактор змінених ліній коду як основного чинника впливу на кількість дефектів у коді. Це вказує на відношення колінеарності між факторами.

У випадку використання алгоритму Random Forest з можливістю донавчання експерименти показали, що точність моделі покращилась у середньому на 7%, що підтверджує ефективність цієї процедури.

Висновки

Використання репозиторіїв відкритого коду для аналізу подій SDLC і, як результат, створення прогностичних моделей є ефективним підходом для вирішення проблеми прогнозування SDLC для нових продуктів та команд, що не мають власної історії розробки.

Дослідження підтвердило доцільність використання репозиторіїв відкритого коду як надійного джерела даних для поліпшення якості та планування випуску нових версій програмного продукту.

Оновлення та адаптація побудованих прогностичних моделей з використанням даних конкретного продукту для врахування поточних особливостей розробки дозволяє забезпечити безперервний зворотний зв'язок з моделями та підвищує їх прогностичну точність.

Можна припустити, що інтеграція прогностичних моделей у середовище CI/CD для автоматизованого контролю якості програмного забезпечення дозволить підвищити рівень ефективності SDLC загалом.

Література

1. Suhee Jo., Ryeonggu Kwon, Gihwon Kwon. Probabilistic Model Checking GitHub Repositories for Software Project Analysis. *Applied Sciences*, 2024. DOI: 10.3390/app14031260.

2. Armando Sousa, Gisele Ribeiro, Guilherme Avelino, Lincoln S., Rocha Ricardo de, Sousa Britto. SysRepoAnalysis: A tool to analyze and identify critical areas of source code repositories. 2022. DOI: 10.1145/3555228.3555281.

3. Kyle Daigle. Octoverse: The state of open source and rise of AI in 2023. URL: https://github.blog/news-insights/research/the-state-of-open-source-and-ai (дата звернення: 01.12.2024).

4. Spring Framework. URL: https://github.com/spring-projects/spring-framework (дата звернення: 01.12.2024).

5. Spring Boot. URL: https://github.com/spring-projects/spring-boot (дата звернення: 01.12.2024).

6. Bonaccorso G. Machine Learning Algorithms. Second edition. O'Reilly Media, Inc., 2018. 522 p.

7. DBbeaver/DBeaver: Free universal database tool and SQL. URL: https://github.com/DBeaver/DBeaver (дата звернення: 01.12.2024).

8. Quarkusio/Quarkus. URL: https://github.com/quarkusio/quarkus (дата звернення: 01.12.2024).

9. Mockito/Mockito. URL: https://github.com/mockito/mockito (дата звернення: 01.12.2024).

References

1. Suhee, Jo., Ryeonggu, Kwon, Gihwon, Kwon. Probabilistic Model Checking GitHub Repositories for Software Project Analysis. *Applied Sciences*, 2024. DOI: 10.3390/app14031260.

2. Armando, Sousa, Gisele, Ribeiro, Guilherme, Avelino, Lincoln, S., Rocha, Ricardo, de, Sousa, Britto. SysRepoAnalysis: A tool to analyze and identify critical areas of source code repositories. 2022. DOI: 10.1145/3555228.3555281.

3. Kyle Daigle. Octoverse: The state of open source and rise of AI in 2023. June 2023. Retrieved from: https://github.blog/news-insights/research/the-state-of-open-source-and-ai.

4. Spring Framework. December 2024. Retrieved from: https://github.com/spring-projects/spring-framework.

5. Spring Boot. December 2024. Retrieved from: https://github.com/spring-projects/spring-boot

6. Bonaccorso G. Machine Learning Algorithms. Second Edition. O'Reilly Media, Inc., 2018. 522 p.

7. DBbeaver/DBeaver: Free universal database tool and SQL. December 2024 Retrieved from: https://github.com/DBeaver/DBeaver.

8. Quarkusio/Quarkus. December 2024. Retrieved from: https://github.com/quarkusio/quarkus.

9. Mockito/Mockito. December 2024. Retrieved from: https://github.com/mockito/mockito.