готових рішень і високу безпеку, підходить для проєктів класичної архітектури. Express.js надає гнучкість і свободу вибору, але вимагає більшого рівня технічної компетенції та уваги до підбору сторонніх інструментів.

1. Choma D., Chwaleba K., Dzieńkowski M. The Efficiency and Reliability of Backend Technologies: Express, Django, and Spring Boot // IAPGOŚ, 4/2023. — P. 73–78. — DOI: 10.35784/iapgos.4279. — Available online: **http://doi.org/10.35784/iapgos.4279** (дата звернення: 1.11.2025).
2. BitByteSoft. Express JS vs Django: A Comparison of Development Approaches and Performance. Available at: **https://bitbytesoft.com/express-js-vs-django-comparison** (accessed November 1, 2025).
3. IEEE Xplore Digital Library. A Comparative Analysis of Node.js and Django for Scalable Web Applications. IEEE. Режим доступу: **https://ieeexplore.ieee.org** (дата звернення: 1.11.2025).
4. Abhishek Jha. Node.js vs Django: A Performance and Scalability Comparison. Режим доступу: https://d197for5662m48.cloudfront.net/documents/publicationstatus/256355/preprint_pdf/f6990ad1813ca8ba2683cb9ee1b01228.pdf (дата звернення: 11.11.2025).
5. Coursera. Node.js vs Django: What's the Difference? 2025, 23 липня. Режим доступу: https://www.coursera.org/articles/node-js-vs-django (дата звернення: 1.11.2025)

РОЗРІЗНЕННЯ БЕЗКОШТОВНИХ ЗАСТОСУНКІВ І ПРОБНИХ ВЕРСІЙ ПЛАТНИХ MACOS-ЗАСТОСУНКІВ НА ПУБЛІЧНИХ МАРКЕТПЛЕЙСАХ / DISTINGUISHING FREE APPS AND TRIAL VERSIONS OF PAID MACOS APPS ON PUBLIC APP MARKETPLACES

Літвінчук З.В., Франків О.О, Петелєв Є.Р, Кривоблоцький С.І, Стулова Н.С / Litvinchuk Z.V., Frankiv O.O, Peteliev Y.R., Krivoblotsky S.I., Stulova N.S

Національний університет «Києво-Могилянська академія» / National University of Kyiv-Mohyla Academy

04070, м. Київ, вул. Г. Сковороди 2, каф. інформатики, тел. (044) 425 60 64

E-mail: z.litvinchuk@ukma.edu.ua, o.frankiv@ukma.edu.ua, zhenya.peteliev@macpaw.com, krivoblotsky@macpaw.com, nata.stulova@macpaw.com

The lack of transparency in app monetization on public marketplaces misleads users and complicates policy enforcement. The paper presents an algorithm to distinguish truly free macOS apps from those that involve in-app purchases by classifying apps into free, freemium, and paid categories. The research was based on developing a hybrid method that combined technical analysis of Mach-O binaries, embedded frameworks, and receipt file with language-based processing of local bundle resources and external marketplace text content. It yielded a feature set for a supervised machine learning classifier. To evaluate the suggested solution, we used a dataset consisting of 1,219 apps and benchmarked supervised classifiers. The tree-based ensembles performed best: Random Forest and Gradient Boosting achieved ~90% accuracy, precision, recall, and F1-score on an 80/20 train–test split. The findings contribute to classification systems automation, enhancing applicability and transparency in the software distribution ecosystem.

Стрімкий розвиток маркетплейсів програмного забезпечення привів до їх комерціалізації. Межа між справді безкоштовними застосунками та продуктами з платними обмеженнями розмивається. У цих умовах прозорість монетизації та коректність маркування стають критично важливими: різноманіття моделей монетизації (підписки, внутрішні покупки, пробні періоди), непослідовне розкриття платних функцій ускладнюють ручну перевірку й підвищують ризики для користувачів

і модерації. Це зумовило потребу в автоматизованому підході до категоризації застосунків за моделлю монетизації.

Класифікація застосунків є актуальною незалежно від операційної системи. У [1] виконано класифікацію Android-застосунків методами статичного аналізу текстових рядків із декомпільованих APK та перевірки дозволів, ознаки доповнено даними з Android Market. Для iOS [2] і macOS [3, 4] поєднують статичний і динамічний аналіз: із виконуваних файлів Mach-O виокремлюють використані фреймворки, класи та методи, а під час виконання відстежують доступ до даних і мережевий трафік. Для macOS додатково застосовують структурний аналіз Mach-O.

Для здійснення дослідження застосунки було поділено на три групи:

3. *free*: доступ до повного функціоналу без додаткових плат;
4. *freemium*: базові можливості залишаються безкоштовними, в той час як розширені функції можуть бути розблоковані через внутрішні покупки або підписки;
5. *paid*: застосунок доступний лише після одноразової покупки.

В окремих випадках ці моделі можуть комбінуватись для максимізації бізнес-вигоди.

Реалізований алгоритм складається з трьох послідовних етапів: технічного аналізу, лінгвістичного аналізу та підсумкової класифікації на основі виокремлених ознак.

Під час технічного аналізу опрацьовуються складові пакета macOS-застосунку: виконуваний файл у форматі Mach-O, вбудовані фреймворки разом з їхніми залежностями та, за наявності, файл MASReceipt із даними про покупки в Mac App Store. Основний бінарний файл і фреймворки перевіряються на наявність платіжної логіки: визначаються платіжні фреймворки (StoreKit, сторонні SDK), виокремлюються відповідні класи й методи, аналізуються символьні таблиці. У текстових сегментах здійснюється пошук адрес API платіжних сервісів та URL-адрес сторінок оплати за допомогою регулярних виразів і словника доменів; той самий аналіз застосовується до вбудованих фреймворків. Якщо застосунок завантажено з Mac App Store, MASReceipt розшифровується для отримання записів щодо покупок і підписок. На виході формується набір технічних ознак: бінарні індикатори наявності платіжних фреймворків, класів і методів; лічильники адрес API та URL-адрес оплати/підтвердження транзакцій; агреговані показники з MASReceipt.

На етапі лінгвістичного аналізу опрацьовуються локальні ресурси пакета (Info.plist, файли локалізацій, UserDefaults, logs) та матеріали маркетплейсів (описи й відгуки на Mac App Store і MacUpdate), а також контент веб-сторінок політики конфіденційності та умов використання. Зовнішні тексти збираються автоматизовано: для Mac App Store - через iTunes Search API з доповненням вмістом HTML-сторінки; для MacUpdate опис і відгуки агрегуються методом web scraping із відтворенням пагінації. Попередня обробка передбачає токенізацію, лематизацію та розбиття CamelCase-ідентифікаторів. Далі, використовуючи перелік платіжних ключових слів, підраховуються їхні частоти для кожного джерела, формуючи джерелозалежний вектор мовних ознак (ключове слово × джерело).

Результати технічного й лінгвістичного аналізу об’єднуються в єдину числову таблицю - вхідні дані класифікатора. Датасет зібрано з Mac App Store та MacUpdate, він налічує 1219 macOS-застосунків і поділений у співвідношенні 80/20 на тренувальну та тестову частини. Первинну розмітку виконано з використанням даних агрегатора AppFigures. Розподіл класів: 39,2 % - free, 58,8 % - freemium, 2 % - paid.

Для класифікації використано моделі машинного навчання з учителем: Decision Tree Classifier, Support Vector Machine з лінійним і поліноміальним ядрами, k-Nearest Neighbors, Gaussian Naïve Bayes, Random Forest Classifier, Extremely Randomized Trees та Histogram-based Gradient Boosting. Якість оцінювалась за Accuracy, Precision, Recall і F1 як зважені середні. Найвищі результати показали деревні ансамблі: Random Forest і Histogram-based Gradient Boosting продемонстрували близько 90 % за Accuracy, Precision і Recall та F1 на рівні 0,89–0,90. Поліноміальна версія Support Vector Classifier перевершила лінійну, що відповідає потребі враховувати нелінійні залежності у

змішаному просторі технічних та мовних ознак. Базові деревоподібні та ймовірнісні моделі, а також SVM із лінійним ядром поступилися ансамблям, особливо на малочисельному класі paid із низьким recall.

Практичне розв'язання проблем непрозорої монетизації дозволяє надійно розмежовувати справді безкоштовні застосунки (*free*) від *freemium* і *paid* та коректно маркувати їх на маркетплейсах. Рішення орієнтоване на інтеграцію у процеси модерації та забезпечення відповідності нормативним вимогам, щоб автоматично виявляти суперечності між заявленою моделлю й фактичною поведінкою застосунку, знижуючи ризики для користувачів і платформи.

Використані джерела:

1. Sanz B., Santos I., Laorden C., Ugarte-Pedrero X., Garcia Bringas P. On the automatic categorisation of Android applications // *2012 IEEE Consumer Communications and Networking Conference (CCNC)*. 2012. P. 149–153. DOI: 10.1109/CCNC.2012.6181075

2. Bhatt A. J., Gupta C., Mittal S. iABC-AL: Active learning-based privacy leaks threat detection for iOS applications. *Journal of King Saud University – Computer and Information Sciences*. 2021. T. 33, № 7. P. 769–786. https://doi.org/10.1016/j.jksuci.2018.05.008

3. Pajouh, H.H., Dehghantanha, A., Khayami, R. et al. Intelligent OS X malware threat detection with code inspection. *Journal of Computer Virology and Hacking Techniques*, 14. 2018. P. 213–223. https://doi.org/10.1007/s11416-017-0307-5

4. Burgardt C. A. P. Malware detection in macOS using supervised learning : dissertation (Master's degree in Computer Science). Recife : Federal University of Pernambuco, 2022.

МОДЕЛЮВАННЯ АСИНХРОННИХ ПРОЦЕСІВ ТА УПРАВЛІННЯ СТАНОМ ЗА ДОПОМОГОЮ МЕРЕЖ ПЕТРІ

Давиденко А.М.
Національний університет «Києво-Могилянська академія»
04655, м. Київ, вулиця Григорія Сковороди, 2, НаУКМА, Факультет інформатики,
andrii.davydenko@ukma.edu.ua

This work explores the formal modeling of asynchronous systems, using classic Place/Transition (P/T) nets and data-driven Coloured Petri Nets (CPNs). Formal models are presented for three distinct processing patterns: a P/T net for a simple mutual exclusion queue, a CPN for context-based concurrent processing, and a CPN for timestamp-based state management.

Моделювання сучасних інформаційних систем, особливо асинхронних та розподілених, вимагає формалізмів, здатних адекватно описувати паралелізм та керовану даними логіку [4, 5]. У цій роботі демонструється застосування мереж Місць/Переходів (М/П) та Кольорових мереж Петрі (КМП) для моделювання попередньо визначених трьох класів розподілених систем [1].

Другий клас розподілених систем моделюється класичною мережею М/П [3]. Ця система моделює просту чергу, де вхідні повідомлення обробляються послідовно. "Маркер-замок" гарантує, що в кожний момент часу може оброблятися лише одне повідомлення (рис. 1, зліва). Мережа М/П визначається як кортеж $N=(P,T,W,M_0)$. Множина місць $P=\{p_{queue},p_{lock},p_{proc}\}$, де p_{queue} – буфер необроблених повідомлень, p_{lock} – доступність обробника (м'ютекс), p_{proc} – стан активного оброблення повідомлення. Множина переходів $T=\{t_{start},t_{finish}\}$, де t_{start} – подія початку оброблення повідомлення, t_{finish} – подія завершення оброблення повідомлення. Вагова функція дуг $W:(P\times T)\cup(T\times P)\to N$ визначає їх кратність. $W(p_{queue},t_{start})=W(p_{lock},t_{start})=W(t_{start},p_{proc})=¿$ $W(p_{proc},t_{finish})=¿$ $W(t_{finish},p_{lock})=1$, $W(a,b)=0$ для всіх інших пар (a,b). Початкове маркування $M_0:P\to N$ визначає початковий стан мережі. $M_0(p_{queue})=k$, де $k\geq 0$ – початкова кількість повідомлень. $M_0(p_{lock})=1$, $M_0(p_{proc})=0$.