

Онлайн ІТ-університет

МАГІСТРАТУРА NEOVERSITY

європейська вища освіта для нових ІТ-лідерів

Faculty: Artificial Intelligence &

Machine Learning

Student: Oleksandr Skriabikov

Contact No.: +380 (63) 101-2000

E-mail: oskriabikov@gmail.com

LinkedIn: linkedin.com/in/askryabikov



Короткий огляд наукової публікації

Порівняння об'єктно-орієнтованої та функційної парадигм програмування у проєктуванні програмного забезпечення

Автори: А. І. КОВАЛЬ, А. М. ЯШИНА, Г. І. РАДЕЛЬЧУК, Ю. В. ФОРКУН

Вступ

У цій публікації розглядаються дві ключові парадигми розробки ПЗ – об'єктно-орієнтована та функційна – а також їхні особливості при проєктуванні програмних систем. Мета роботи полягає в тому, щоб показати, у яких умовах кожна з парадигм є найбільш зручною та наскільки вони придатні на практиці. Наводяться типові приклади мов програмування: для ООП – Java, C++ і Python; для функційного підходу – Haskell, Lisp і F#.

Методологія

Аналіз побудовано на порівнянні концепцій двох парадигм і розгляді їхніх ключових принципів. ООП описано через інкапсуляцію, успадкування, абстракцію та поліморфізм. Функційний підхід розглядається через чисті функції, відсутність стану та принцип референційної прозорості. Також аналізуються відмінності в архітектурних підходах і способах декомпозиції.

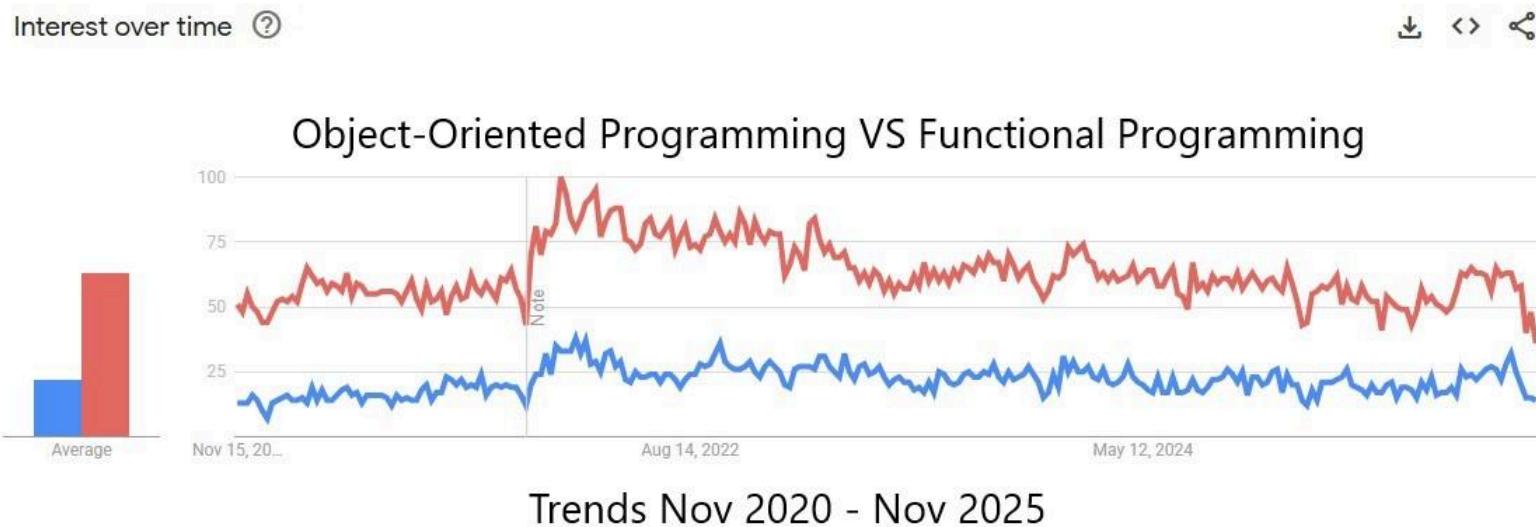
Результати

Автори пояснюють, що обидві парадигми по-своєму ефективні. ООП краще підходить для моделювання сутностей реального світу та складних структур даних. Натомість функційне програмування переважає в задачах, пов'язаних із великими обсягами обчислень, паралельністю та високою передбачуваністю результатів. Зростання популярності ФОП пов'язане з Big Data та машинним навчанням, тоді як ООП залишається стандартом для корпоративних систем.

Ключові інсайти

1. Кожна парадигма найкраще проявляється у своїй сфері: ООП – там, де важливі об'єкти та їхня поведінка, а функційний підхід – там, де критичною є чистота обчислень і стабільність результату. Це допомагає точніше підбирати інструменти під конкретну задачу.
2. В ООП дані “живуть” усередині об'єктів, і ми змінюємо їх через методи. У функційному програмуванні дані не перезаписуються – на кожному кроці створюється нова їхня версія. Такий підхід робить програми більш передбачуваними, що важливо й в аналізі даних.

3. На оновленому графіку видно, що з 2020 по 2025 рік інтерес до функційного програмування стабільно залишається значно вищим, ніж до об'єктно-орієнтованого. Водночас запити, пов'язані з ООП, утримуються на нижчому, але досить стабільному рівні без різких коливань.



Висновок

Стаття дає чітке та структуроване порівняння двох парадигм програмування й підкреслює, що вибір підходу має залежати від характеру конкретного завдання та логіки розроблюваної системи. Автори показали, що кожна з парадигм має свої сильні та слабкі сторони, а їхнє застосування визначається практичним змістом у контексті проєкту. Такий підхід допоміг поглянути на архітектуру програмного забезпечення більш зважено.