Закріплення юридичної сили блокчейн-записів сприятиме появі нових можливостей для інноваційних бізнес-моделей, розробок і досліджень. З часом блокчейн може стати не лише технічним інструментом, а й основою сучасної екосистеми правовідносин у сфері ІВ, де прозорість, надійність і справедливий розподіл переваг стануть нормою.

Висновок

Технологія блокчейн пропонує дієві механізми для зміцнення захисту інтелектуальної власності ще на етапі її створення. Вона забезпечує беззаперечний доказ авторства, дату виникнення, незмінність записів, прозоре управління правами через смарт-контракти та спрощує контроль за дотриманням цих прав. Хоча впровадження пов'язане з певними викликами та потребує адаптації правової бази, вироблення стандартів і технічних рішень, потенційні вигоди важко переоцінити.

Список використаних джерел:

[1] A. Atzei, N. Bartoletti, T. Cimoli, "A survey of attacks on Ethereum smart contracts (SoK)," Proceedings of the 6th International Conference on Principles of Security and Trust (POST), Springer, 2017. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9014349

[2] T. Chen, X. Li, X. Luo, X. Zhang, "Under-Optimized Smart Contracts Devour Your Money," Proceedings of the 24th USENIX Security Symposium, 2019. [Online]. Available: https://ieeexplore.ieee.org/document/10061741

[3] Поліщук Юрій, Електрона залікова книжка студента на основі технології Blockchain, Електронний архив НаУКМА, 2021. [Online]. Available: https://ekmair.ukma.edu.ua/handle/123456789/22027

[4] S. Ruj, M. Stojmenovic, "Decentralized Access Control with Anonymous Authentication of Data Stored in Clouds," Information Security and Privacy, Springer, 2017. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-662-57611-3_1019/full

[5] X. Chen, X. Huang, J. Li, D. Wong, "Emerging Technologies of Blockchain and Intellectual Property," Information Security and Privacy, Springer, 2017. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-662-57611-3_10

ДЕЯКІ ОСОБЛИВОСТІ ЗАСТОСУВАННЯ ОБ'ЄКТНО-ОРІЄНТОВАНОЇ ПАРАДИГМИ

Бублик В.В. доцент, **Д.Р.Фітель**,випускник НаУКМА, працює в Microsoft.USA
Національний університет «Києво-Могилянська академія»
04655, м. Київ, вулиця Григорія Сковороди, 2, кафедра мультимедійних систем,
E-Mail: boublik@ukma.edu.ua; тел. (044) 463 69 85

The paper explores Edsger Dijkstra's alleged harsh criticism of the object-oriented paradigm, offering a detailed critique of contemporary object-oriented programming (OOP). It demonstrates how certain attributes common in OOP implementations cannot be expressed as mathematical theories, thus making them incompatible with Dijkstra's structured programming philosophy. Key principles of the OO paradigm that aim to create precise abstractions with a clear hierarchy, akin to structured programming have been outlined. Additionally, the paper examines C++ as a successful example of OOP implementation, highlighting its ability to facilitate abstractions without making them obligatory. Finally, the paper illustrates how combining the OO paradigm with other paradigms can be advantageous. It uses a specific example of template metaprogramming in C++ applied to an object-oriented implementation of an algorithm, showcasing the vast scalability for multiple use cases without incurring runtime costs.

В доповіді аналізуються деякі механізми абстракції, властиві мові програмування С++, вживані у поєднанні процедурних, узагальнених і власне об'єктно-орієнтованих засобів мови. Тема була спровокована жвавою дискусією навколо висловлювання, приписуваного Едсгеру Дейкстрі, стосовно об'єктно-орієнтованого програмування. Ця теза Дейкстри зазвичай цитується за важко доступною роботою Боба Кроуфорда [1], на питання якого про ООП Дейстра начебто відповів: *"object oriented programs are offered as alternatives to correct ones"* і далі *"object oriented programming is an exceptionally bad idea which could only have originated in California"*. На думку Кроуфорда Дейкстра не стільки заперечував вживання об'єктів у програмуванні, скільки проти використання об'єктів для всього. Критичне ставлення Дейкстри можна зрозуміти, виходячи з його позиції про те, що програма має близьку структурну аналогію з будь-якою математичною теорією [2]. Тому, на думку Дейкстри, робота програміста суттєво не відрізняється від роботи творчого математика.

Спробуємо уявити, як могла б виглядати більш детальна критика сучасних реалізацій ООП.

Проблеми насправді починаються з самого визначення парадигми. На жаль, поширені «доступні» визначення надто загальні, і спільні ознаки зводяться до понять класу і об'єктів. Мабуть, найбільш відоме визначення – формула «ООП = Інкапсуляція + успадкування + поліморфізм» - зовсім не відповідає суті парадигми, а лише перераховує окремі атрибути, які часто притаманні її реалізаціям. Найбільше уваги, як правило, приділяють успадкуванню, яке на практиці виявляється найбільш проблематичною рисою.

Спершу дамо визначення парадигмі ООП.

- ООП ґрунтується на об'єктах, пов'язаних ієрархіями вкладення, а не алгоритмах.
- Кожен об'єкт суть екземпляр певного класу.
- Класи утворюють ієрархію спадкувань.

Програма являє собою взаємодію об'єктів, що її складають, а сама взаємодія виражається в термінах алгоритмів. Це уявлення повністю відповідає ідеології, покладеної в основу C++ як мультипарадигменної мови. Цікаво, що один з найвидатніших розробників C++, Александр Степанов, сам критично ставився до методології об'єктно-орієнтованого програмування, вважаючи її методологічно помилковою. Починати потрібно не з класів, на думку Степанова, а з алгоритмів. Лише вивчивши їх досконально, можна винаходити інтерфейси для їх впровадження [3]. Тут ще раз проявляється перевага C++, наявність процедурної частини якого дозволяє легко оперувати окремим алгоритмами.

Застосування об'єктно-орієнтованої парадигми ґрунтується на понятті об'єктної моделі, найважливішими елементами якої за Г.Бучем [3] суть: абстрагування, інкапсуляція і ієрархічність.

Ключова властивість ООП – ***абстрагування***, тобто можливість визначення нових абстракцій. Інші елементи певною мірою або служать для забезпечення абстрагування. Але, слідуючи за Дейкстрою, абстрагування полягає не розмиванні понять, а в створенні нового понятійного рівня, на якому можна бути абсолютно точним. Ієрархії об'єктів і класів складають ієрархії рівнів абстракції.

Інкапсуляція в свою чергу забезпечує вищий рівень абстракції цілої конструкції порівняно з рівнем її складових, виділяючи її істотні характеристики.

Пошук адекватних абстракцій – головне завдання об'єктно-орієнтованого програмування.

Нарешті згадаємо про іншу важливу складову, яка значною мірою визначає успішність застосування об'єктно-орієнтованої парадигми. Це узагальнене програмування, яке забезпечує новий вищий рівень абстракції.

Як приклад, звернемося до задачі пошуку екстремуму методом градієнтного спуску. В книзі [5] наведено алгоритм і шаблон функції для пошуку екстремуму функцій двох змінних.

```
template <typename Value, typename T1, typename T2, typename F, typename G>
auto gradient_descent(Value& u, T1 s, T2 eps, F f, G g) {
  auto val = f(u), delta = val;
  do {
            u -= s * g(u);
            auto new_val = f(u);
            delta = abs(new_val - val);
            val = new_val;
  } while (delta > eps);
  return u;
}
```

Застосувавши техніку варіативних шаблонів, узагальнимо наведену вище функцію до використання простору R довільної, наперед невідомої розмірності,

```
template <typename ...P> struct R;
template <typename T, typename ...P>
struct R<T, P...> {
  T x;
  R<P...> y;
};
template<>
struct R<>{};
```

яка виявлятиметься в місці використання об’єкту потрібної розмірності, наприклад,

```
R<double, double, double, double, double> u = { 1, 2, 3, 4, 5 };
```

Тим же методом визначимо цільову функцію, наприклад, $y = {x_1}^2 + {x_2}^2 + ... + {x_n}^2$.

```
template <typename ...T>
inline double f(const R<T...>& u);
template <typename T, typename ...P>
inline double f(const R<T, P...>& u) { return u.x*u.x+f(u.y); }
template <>
inline double f(const R<>& u) { return 0;}
```

І її градієнти:

```
template <typename ...T>
inline R<T...> g(const R<T...>& u);
template <typename T, typename ...P>
inline R<T, P...> g(const R<T, P...>& u) { return { 2 * u.x, g (u.y)}; }
template <>
inline R<> g(const R<>& u) { return {}; }
```

Оператори множення і віднімання, застосовані у алгоритмі, визначаємо аналогічно. Для виклику градієнтного спуску обгорнемо цільову функцію і її градієнти в лямбда вирази

```
gradient_descent(u, h, eps, [](auto u) { return f(u); }, [](auto u) { return g(u); });
```

Як висновок стверджуємо, що підтримка об’єктно-орієнтованої парадигми узагальненим програмуванням доповнюють одна другу без додаткових витрат етапу виконання.

Література:

1. Crawford, Bob. "Object-Oriented Programming: The Good, the Bad, and the Ugly." *TUG Lines*, vol. 32, Aug.-Sep. 1989, pp. 7-11.
2. E. W. Dijkstra, Hoe wiskundig programmeren is, EWD 261, https://www.cs.utexas.edu/~EWD/transcriptions/EWD02xx/EWD261.html
3. An Interview with A. Stepanov by Graziano Lo Russo, 2008-04-25 http://www.stlport.org/resources/StepanovUSA.html
4. Grady Booch, Object-Oriented Analysis and Design with Applications, Addison-Wesley Professional; 3rd edition, 2007, 720 pp.
5. Peter Gottschling, Discovering Modern C++, Addison Wesley, 2015, 472 pp.

СТВОРЕННЯ РЕКОМЕНДАЦІЙНОЇ СИСТЕМИ ДЛЯ РОЗВ’ЯЗАННЯ ЗАДАЧ З ГЕОМЕТРІЇ

Жежерун О.П., доцент,,**Смиш О.Р**.,PhD, ст.викладач// Zhezherun, O., Smysh, O
Національний університет «Києво-Могилянська академія»
вулиця Григорія Сковороди, 2, Київ, 04655, тел.: +380 44 425-77-23
E-mail: o.smysh@ukma.edu.ua, zhezherun@ukma.edu.ua

At the Department of Multimedia Systems at the National University of Kyiv-Mohyla Academy, a recommendation system is being developed to automate solving mathematical problems. This system utilizes natural language processing (NLP) techniques and the UDPipe morphosyntactic analyzer to extract meaningful elements from problem statements. Designed for students, teachers, and educators, it provides automated problem-solving, step-by-step solutions, and interactive visualizations. The system supports various tasks, including calculating geometric properties of shapes such as squares, triangles, and trapezoids, and visualizing these elements. Developed in Python, the project integrates advanced tools like Matplotlib. Tested on diverse problem sets, the system demonstrates effectiveness and flexibility in educational and research contexts. Future plans include expanding mathematical capabilities and integrating multilingual support.