DOI 10.31891/2307-5732-2024-333-2-5
УДК 004.773.2

ФЕНЯК РОМАН
Національний університет «Львівська політехніка»
https://orcid.org/0009-0002-4084-0613
e-mail: roman.y.feniak@lpnu.ua
ВИКЛЮК ЯРОСЛАВ
Національний університет «Львівська політехніка»
https://orcid.org/0000-0003-4766-4659
e-mail: yaroslav.i.vykliuk@lpnu.ua

# ПІДХОДИ ДО АВТОМАТИЗАЦІЇ ПРОЄКТУВАННЯ АРХІТЕКТУРИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ З ВИКОРИСТАННЯМ ШТУЧНОГО ІНТЕЛЕКТУ

*Автоматизація проєктування архітектури програмного забезпечення за допомогою генеративного штучного інтелекту пропонує практичне розв'язання проблем, з якими стикається сучасна розробка програмного забезпечення. У цьому документі обговорюється використання генеративного штучного інтелекту під час проєктування архітектури програмного забезпечення для покращення процесів проєктування шляхом використання досвіду успішних проєктів і розуміння причинно-наслідкових зв'язків між добре зібраними вимогами та їх досягнення за допомогою оптимальних архітектурних рішень. За допомогою генеративного штучного інтелекту архітектори програмного забезпечення можуть оптимізувати проєкти архітектури відповідно до підвищених вимог до складності та якості сучасних програмних систем. Водночас це відкриває двері для творчих рішень, які інакше не завжди можливо розробити через брак часу чи зусиль. Таким чином, чистий час, витрачений на процес проєктування, може бути скорочений за допомогою штучного інтелекту, що дає змогу архітекторам програмного забезпечення приймати кращі стратегічні рішення, зменшуючи операційну суєту. Однак, поряд із зазначеними вище перевагами, використання ШІ в архітектурі програмного забезпечення створює наступні проблеми. По-перше, це можливість інтерпретації та перевірка архітектури, розробленої штучним інтелектом: більшість цих проєктів базується на деяких складних алгоритмах, які необхідні для визначення остаточних специфікацій, тобто необхідно довести, що вони задовольняють вимоги користувача. Друге питання пов'язане з процесом інтеграції – інтеграція має стати стандартизованою операцією, завдяки якій роль фахівців ШІ у взаємодії з архітектурою програмного забезпечення буде визначена набагато чіткіше та систематичніше. Таким чином, усі початкові переваги, недоліки та можливі варіанти використання можуть бути вирішені за допомогою конкретних чітко визначених принципів інтеграції. У наступному документі розглядаються кілька наявних випадків впровадження штучного інтелекту, окреслюються плюси та мінуси таких заходів і численні шляхи подолання можливих проблем. Підсумовуючи, інтеграція генеративного штучного інтелекту в розробку архітектури програмного забезпечення забезпечує широкий спектр переваг і обіцяє більш ефективні, творчі та потужні програмні системи в короткостроковій і довгостроковій перспективі. Оскільки ця концепція продовжує розвиватися, у сфері розробки програмного забезпечення з'являться нові креативні можливості.*

*Ключові слова: проєктування архітектури програмного забезпечення, генеративний AI, автоматизація дизайну.*

FENIAK ROMAN, VYKLYUK YAROSLAV
Lviv Polytechnic National University

## APPROACHES TO THE AUTOMATION OF SOFTWARE ARCHITECTURE DESIGN USING AI

*Automating software architecture design through generative Artificial Intelligence offers a practical solution to the challenges faced in contemporary software development. This paper discusses the use of generative AI during the software architecture design to improve design processes by leveraging the experience of successful projects and understanding the cause-and-effect relationships between well-collected requirements and achieving them through optimal architectural decisions. With generative AI, software architects can optimize architecture designs adequately to the elevated complexity and quality demands of modern software systems. At the same time, it opens the door to creative solutions that otherwise are not always possible to develop due to a lack of time or effort. So the net time spent on the design process can then be reduced by AI, which enables software architects to make superior strategic decisions while yielding less operational fuss. However, along with the above advantages, the use of AI in software architecture presents the following challenges. The first one is interpretability and validation of the AI-designed architecture: most of these designs are based on some complex algorithms that are needed to put down to final specifications, meaning that it is necessary to prove that it satisfies the user requirements. The second issue is related to the integration process – integration should become a standardized operation, thanks to which the role of AI professionals in cooperation with software architecture would be determined much clearer and more systematically. Therefore, all of the introductory advantages, shortages, and possible use cases might be settled using specific well-defined integration principles. The following paper considers several existing cases where AI was introduced, outlining the pros and cons of such measures and the numerous ways to overcome possible challenges. In conclusion, the integration of generative AI into software architecture design provides a wide array of benefits and promises more efficient, creative, and powerful software systems in the short and long run. As this concept continues to evolve more creative possibilities will become available in the field of software development.*

*Keywords: software architecture design, generative AI, Software design automation.*

### Постановка проблеми

Індустрія 4.0 змінила ландшафт архітектури програмного забезпечення, наповнивши його безпрецедентною кількістю передових технологій, які обіцяють підвищення ефективності, автоматизацію процесів, поліпшення якості продукції, зниження витрат і забезпечення гнучкості та швидкості адаптації до змін на ринку. А саме технології великих даних (Big Data), штучного інтелекту (Artificial Intelligence – AI), інтернету речей (Internet of Things – IoT), та інших напрямків суттєво розширила наявні технічні можливості і надали інструментарій архітекторам програмного забезпечення для розв'язання бізнесових задач великої

складності.

Відповідно, сам процес проєктування стає складнішим і вимагає значних часових і ресурсних затрат, залучення широкого кола експертів для ефективної інтеграції новітніх технологій і їх сумісності з наявними рішеннями в корпоративній архітектурі, з урахуванням викликів у сфері кібербезпеки та потребах у взаємодії з різними хмарними сервісами. Це вказує на необхідність пошуку способів часткової автоматизації певних підпроцесів для оптимізації та підвищення ефективності розробки.

Мета цієї статті полягає в дослідженні потенціалу штучного інтелекту для автоматизації процесу проєктування архітектури програмного забезпечення, зокрема через використання генеративних алгоритмів, та визначенні можливостей та викликів, що стоять перед архітекторами ПЗ у цій новій парадигмі.

Завданнями статті є:

1) аналіз сучасних методологій і нещодавніх досліджень націлених на автоматизацію проєктування архітектури, зокрема використовуючи генеративний штучний інтелект (Generative AI);
2) ідентифікація ключових викликів і проблем, які потребують подальших досліджень;
3) розробка рекомендацій для можливої інтеграції штучного інтелекту в процеси проєктування архітектури програмного забезпечення, з метою підвищення ефективності, інноваційності та відповідності до бізнес-вимог.

**Аналіз останніх джерел**

Нещодавні дослідження вивчають різні підходи до автоматизованого проєктування архітектури ПЗ використовуючи засоби штучного інтелекту. По-перше, засновані на розлогій базі знань методи є широко використовувані в архітектурі програмного забезпечення [1]. Дослідження демонструє наростальний інтерес до технік на основі факто-орієнтованих методів проєктування архітектури програмного забезпечення. Крім того, дослідження рекомендує автоматизовувати рутинні завдання розробки ПЗ за допомогою моделей Generative AI. Однак повна автоматизація обмежена наявними моделями.

Додатковою областю дослідження є технологія автоматичного генерування альтернативних рішень в архітектурному проєктуванні [3]. Такі дослідження підкреслюють необхідність охоплення ранніх етапів архітектурного проєктування. Крім того, у дослідженнях пропонується використання технік багатокритеріальної оптимізації, таких як оптимізація з використанням алгоритму мурашиних колоній для оптимізації наявних архітектур ПЗ [4]. Навчання моделей на основі залежностей вимог і архітектурних патернів дозволяє автоматизувати здатність визначати оптимальні архітектурні рішення для різних нефункціональних вимог.

На додаток, були проведені дослідження щодо вивчення search-based software engineering, що має потенціал для автоматичного синтезу та складання архітектур програмного забезпечення, наприклад, за допомогою генетичних алгоритмів [5]. Вони можуть бути використані для створення архітектур програмного забезпечення, що складаються з багатьох проєктних шаблонів. Крім того, використання великих мовних моделей (large language models) в автоматизованій генерації коду призвело до розробки шестирівневої архітектури (Six-Tier Architecture) для розробки програмного забезпечення засобами Штучного Інтелекту [6]. Це дозволяє створити адаптивну та гнучку систему для задоволення постійно змінних потреб розробки програмного забезпечення.

Конкретно, в контексті архітектури, також було проведено дослідження штучного інтелекту, яке передбачає, що ШІ створюватиме дизайн архітектури, котрий буде здатний адаптуватися до технологічних викликів росту і масштабованості, з якими вони зіткнуться у майбутньому [6]. Теоретично, підхід на основі ШІ міг би покращити адаптивність і гнучкість архітектури. Застосування ШІ як педагогічного інструменту в архітектурній освіті було досліджено за допомогою експериментів [7]. Цей експеримент надає уявлення про роль ШІ при вивченні особливостей проєктування архітектури ПЗ та загальних поліпшеннях, що стали доступними завдяки ШІ. Таким чином, ці нещодавні дослідження показують, що інтерес дослідників розширюється в сторону поліпшення автоматизованого проєктування архітектури програмного забезпечення за допомогою генеративного ШІ на основі наявних баз знань та оптимізаційних моделей.

**Виклад основного матеріалу**

Проєктування архітектури програмного забезпечення — це важливий процес у розробці програмного забезпечення, який включає розв'язання проблем високого рівня щодо того, як система або частина системи вписується в загальну екосистему і як вона працює загалом, щоб відповідати технічним і експлуатаційним вимогам. Існує кілька методологій, які керують процесом проєктування архітектури програмного забезпечення:

- Архітектурна методологія від Software Engineering Institute (SEI) надає засоби для документування проєктних рішень в різних розрізах. У класифікації SEI, яка ґрунтується на так званих репрезентаціях (views - опис лиш тих складових архітектури в межах поточної репрезентації, які потрібні для розуміння поточного контексту і прийнятого рішення), проєктні рішення повинні бути зафіксовані та задокументовані якнайшвидше. SEI також представив метод аналізу компромісів архітектури (Architecture Tradeoff Analysis Method, ATAM)) для аналізу нефункціональних вимог системи та їх впливу на систему в цілому для досягнення оптимальних результатів. Відповідно, наявні методології SEI допомагають приймати відповідні рішення вчасно й уникати хаосу на етапі проєктування.
- З іншого боку, TOGAF (The Open Group Architecture Framework) — це фреймворк, котрий описує підхід до проєктування, планування, впровадження та керування ентерпрайз архітектурою підприємства. TOGAF заохочує узгодження бізнес-цілей з IT-стратегією та гарантує, що архітектура задовольняє потреби

організації, зокрема. TOGAF складається з кількох етапів:

- Визначення архітектурної візії на основі вимог від зацікавлених сторін.
- Визначення бізнес-архітектури, котра описує як організація має працювати, щоб досягнути своїх бізнес-цілей.
- Визначення архітектур інформаційних систем, котрі описують які дані і аплікації організації потрібні для досягнення бізнес архітектури.
- Визначення технологічної архітектури (Technology Architecture), котра займається визначенням і описом програмних і апаратних можливостей, необхідних для підтримки розгортання бізнес-служб, даних і прикладних послуг.

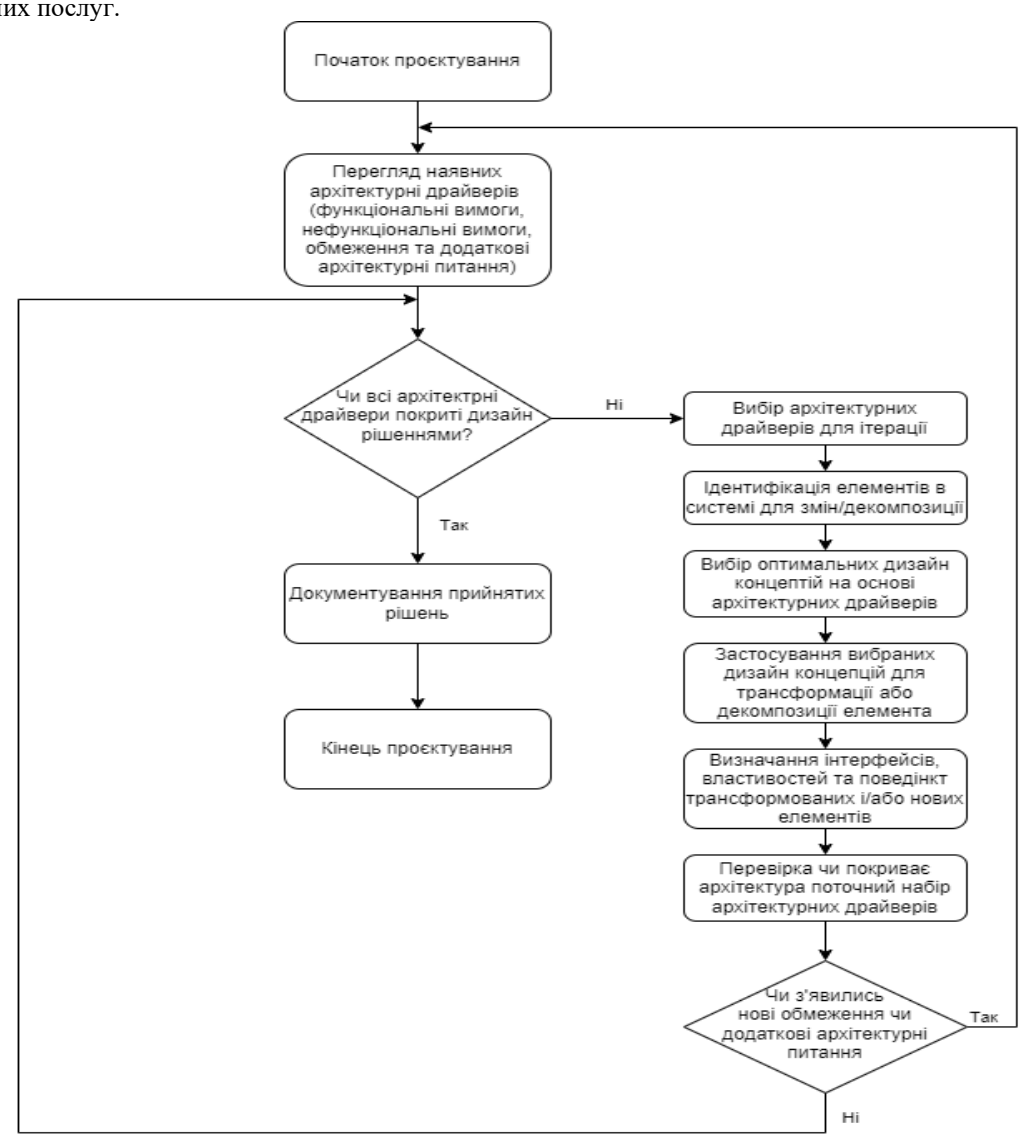


**Рис. 1. Ітеративний процес проєктування архітектури**

У той час як для методологій SEI та TOGAF, відмінності в дизайні архітектури програмного забезпечення корелюють з їх подібністю: SEI орієнтована на рівень аплікації, а TOGAF на рівень ентерпрайзу. Обидва підходи мають спільні кроки в процесі проєктування архітектури програмного забезпечення (рис. 1). Вони підкреслюють важливість розуміння та документування архітектури через різні відображення та перспективи. Вони також підкреслюють врахування нефункціональних вимог та обґрунтованих проєктних рішень на основі архітектур цих атрибутів, які відповідають вимогам бізнесу та користувачів. Важливо, що обидва методи також пропонують узгодження дизайну з організаційними цілями та стратегією.

Моделі на основі Generative AI можуть значно автоматизують процес проєктування архітектури програмного забезпечення за методологіями SEI та TOGAF. Штучний інтелект допоможе автоматизувати генерацію архітектурних діаграм за оптимальними ключовими словами та принципами проєктування. Така форма автоматизації може значно прискорити процес проєктування та запропонувати архітекторам широкий вибір архітектурних рішень, бо такі моделі будуть опиратись на ґрунтовні бази знань, матимуть пропрацьовані зв'язки між типовими архітектурними драйверами (функціональними/нефункціональними вимогами, обмеженнями та архітектурними викликами) і оптимальними вирішеннями цих завдань. Окрім того, модель розумітиме взаємосуперечливі рішення і зможе на ранніх етапах передбачити потенційні проблеми та запропонувати кращі альтернативи, де б архітектору довелось провести n-кількість ітерацій після котрої він б

можливо побачив проблемні моменти.

Загалом, автоматизований процес (рис. 2) може виглядати наступним чином, архітектор формує вхідні вимоги до проєкту разом з зацікавленими сторонами в стандартному форматі (чітко окреслені бізнес-цілі рішення, функціональні та нефункціональні вимоги до цього рішення, архітектурні обмеження котрі на це рішення накладаються та потенційні архітектурні виклики пов'язані зі специфікою проєкту). Після цього, модель ШІ опрацьовує ці вимоги та повертає чітко сформовану архітектурну візію з описом всіх прийнятих рішень і поясненням чому було зроблене те чи інше архітектурне рішення. Таким чином, архітектори зможуть ефективніше проєктувати базову архітектуру та інтенсивніше досліджувати матеріали для подальших вдосконалень, що корелює з ітераційною природою процесу проєктування.


**Рис. 2. Автоматизований процес проєктування архітектури**

Проте не слід забувати, що використання Generative AI для розробки архітектури програмного забезпечення може мати свої плюси та мінуси, залежно від можливого впливу цієї технології на проєктування архітектури програмного забезпечення.

Плюси використання Generative AI для проєктування архітектури ПЗ:

- Автоматизація та ефективність. Генеративний штучний інтелект може суттєво контролювати значну кількість робочої сили, необхідної для виконання таких завдань, як створення візуалізації дизайну та виявлення потенційних вдосконалень.
- Розширений процес прийняття рішень. ШІ може запропонувати архітекторам велику кількість пропозицій на основі проаналізованих даних, допомагаючи їм розв'язувати архітектурні проблеми та обирати оптимальні архітектурні компроміси
- Відповідність бізнес-цілям. Generative AI, який автоматизує узгодження дизайну з бізнес-цілями та планами, щоб гарантувати, що архітектура все ще актуальна, оскільки ринок розвивається та потреби компанії змінюються.
- Інноваційні дизайнерські рішення. ШІ може створити додаткові варіанти дизайну на основі оптимізованих ключових слів і принципів дизайну.
- Адаптивність і гнучкість. ШІ може розробити адаптивні та гнучкі архітектурні проєкти, які, ймовірно, будуть відповідати новому викликам пов’язаним з потенційним ростом і розширенням продукту.

Мінуси використання Generative AI для проєктування архітектури ПЗ:

- Надмірна залежність від ШІ. Архітектори можуть покладатися на рішення, запропоноване штучним інтелектом, таким чином тенденція до творчості, ймовірно, буде зупинена. Однак необхідно встановити баланс між людською інтуїцією та досвідом і дизайном ШІ.
- Якість і перевірка. Архітектури, розроблені штучним інтелектом, мають різний рівень якості, і експерти повинні ретельно перевірити та чітко знати, чи відповідає система певній узгодженій якості та цілям.
- Складність і можливість інтерпретації. Дизайн генеративної архітектури може бути для розуміння.
- Етичні міркування. Ця практика породила багато занепокоєнь щодо етики та конфіденційності та, отже, може поставити архітекторів у досить незручну ситуацію.

Інтеграція генеративного ШІ в процес архітектурного проєктування передбачає зміну парадигми в галузі в бік більш ефективних, інноваційних та персоналізованих процесів проєктування. Впровадження genAI має здійснюватися свідомо та поетапно, забезпечуючи його ефективну інтеграцію в робочий процес та організаційну культуру. Нижче наведено план загального впровадження genAI в організації, а також опис того, як ШІ може допомогти на кожному етапі дизайну архітектури виходячи за рамки проєктування рішення.

План інтеграції Generative AI в організації:

- Оцінка поточних процесів в організації: на цьому етапі слід оцінити поточні практики і методології проєктування, наявні ресурси, та використовувані інструменти, щоб визначити готовність до впровадження genAI.
- Стратегічне планування: визначення конкретних областей, у яких інструменти та моделі genAI додадуть цінності, узгодження цілей і визначення кроків для інтеграції Generative AI.
- Технологічна інфраструктура: створення необхідної інфраструктури, включаючи апаратне та програмне забезпечення, доступність до даних та інші інструменти, які потрібні для інтеграції Generative AI.
- Розвиток навичок: створення навчальних матеріалів для архітекторів для покращення їхніх здібностей щодо використання інструментів на основі genAI.
- Вибір пілотних проєктів: вибір оптимального проєкту, куди можна інтегрувати GenAI на початкових етапах, щоб протестувати нові процедури і виявити потенційні недоліки.
- Фінальна інтеграція: повна інтеграція моделей і інструментів genAI у загальні процедури розробки проєкту.

Етапи в циклі проєктування архітектури і як Generative AI може на них допомогти:

- Етап планування: Generative AI можна використовувати для структурування клієнтських вимог, проєктних обмежень і бізнес-цілей відповідно до конкретного бізнес-кейсу і доменної області в котрій має бути створено продукт чи система.
- Концептуальний етап: Generative AI можна використовувати для створення різноманітних варіантів дизайну, використовуючи попередній досвід архітектора, потреби зацікавлених сторін, історичні записи та відомі принципи дизайну для створення оптимального рішення, котре задовольнить всіх зацікавлених сторін.
- Фаза детального проєктування: Generative AI можна використовувати для створення детальних специфікацій, наприклад опису компонентів, їх особливостей і взаємозв’язків, деталей інтерфейсів та шаблонів на основі кращих практик і доменних стандартів.
- Етап оцінювання: використання генеративного штучного інтелекту для оцінки різних комбінацій архітектурних рішень і моделювання досяжності ними їх нефункціональних вимог, щоб впевнитися що поточне архітектурне рішення може в достатньому об’ємі покривати потреби зацікавлених сторін.
- Етап документування: використання штучного інтелекту для автоматизації процесу запису архітектурних рішень відповідно до очікуваного стандарту.
- Етап імплементації: Generative AI може спростити процес впровадження надаючи розробникам можливість легко шукати інформацію про обрані архітектурні рішення.

**Висновки**

Підбиваючи підсумки, можна сказати, що автоматизація проєктування програмної архітектури за допомогою генеративних алгоритмів штучного інтелекту є проривною розробкою у сфері програмної інженерії. Це досягнуто завдяки здатності штучного інтелекту генерувати архітектурні візії на основі баз знань і наданих вхідних вимог до проєкту в рамках однієї ітерації проєктування замість того щоб архітекторам витрачати безліч часу та ресурсів. Такий підхід спрощує процес проєктування та підвищує ефективність праці архітекторів, мінімізуючи ризики пов'язані з людськими помилками, відсутністю досвіду в усіх необхідних технологіях і недостатнім розумінням потенціалу рішення.

В результаті, майбутнє проєктування програмної архітектури нерозривно пов'язане з розвитком штучного інтелекту, що стає все більш революційним у розв'язанні задач дизайну програмного забезпечення. Підтримка наукових досліджень цього методу сприятиме створенню корисних рекомендацій щодо його інтеграції, заохочуючи інновації, покращуючи процеси роботи та більш ефективне використання людських ресурсів в професійній архітектурі програмного забезпечення.

**References**

1. Representation learning: a review and new perspectives. ieee transactions on pattern analysis and machine intelligence, 35(8), 1798-1828. (2013). https://doi.org/10.1109/tpami.2013.50

2. A study on the development of automatic design alternatives generation technology used in the early stages of architectural design. asia-pacific journal of convergent research interchange, 7(3), 123-132. (2021). https://doi.org/10.47116/apjcri.2021.03.11

3. Application of knowledge-based approaches in software architecture: a systematic mapping study. information and software technology, 55(5), 777-794. (2013). https://doi.org/10.1016/j.infsof.2012.11.005

4. Artificial intelligence versus software engineers: an evidence-based assessment focusing on non-functional requirements. (2023). https://doi.org/10.21203/rs.3.rs-3126005/v1

5. Model checking software architecture design. (2012) https://doi.org/10.1109/hase.2012.12

6. Interleaving human and search-based software architecture design. proceedings of the Estonian academy of Sciences, 62(1), 16. (2013). https://doi.org/10.3176/proc.2013.1.03

Search-based software engineering. acm computing surveys, 45(1), 1-61. (2012). https://doi.org/10.1145/2379776.2379787