ARTIFICIAL INTELLIGENCE IN THE CYBERSECURITY FIELD

DOI 10.20535/2411-1031.2025.13.2.344711
УДК 004(89+4’2)
ВОЛОДИМИР СОКОЛОВ

СПОСІБ ЗАСТОСУВАННЯ ШТУЧНОГО ІНТЕЛЕКТУ ДЛЯ СТВОРЕННЯ ТА РЕВЕРС-ІНЖИНІРИНГУ ГРАФІЧНИХ МОДЕЛЕЙ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

В статті представлено спосіб застосування систем генеративного штучного інтелекту (СШІ) на основі великих мовних моделей для побудови з промптів та відновлення з вихідного коду графічних моделей програмного забезпечення (ПЗ). Розроблений спосіб розглядається як основа для інтеграції СШІ та графічних систем (ГС), які традиційно використовуються для побудови графічних моделей ПЗ. В процесі дослідження розглядались такі методи та нотації графічного моделювання як BPMN, IDEF, ERD, UML та C4. В процесі аналізу форматів представлення графічних моделей різними ГС було визначено, що найбільш зручними для застосування СШІ є мовні описи моделей, на відміну від XML-подібних та бінарних форматів. Ідея способу полягає у використанні синтаксису DSL (Domain Specific Language) популярних ГС в якості проміжних мов взаємодії СШІ та ГС, що забезпечує можливості як інтелектуальної обробки мовного опису графічної моделі СШІ, так і якісного її відображення ГС. Суть способу полягає у представленні кожної графічної схеми моделі трирівневою архітектурою та застосуванні композиції функцій міжрівневої трансформації. Трирівнева архітектура представлення графічної схеми включає вхідний промпт (семантика моделі), DSL-опис схеми для обраної ГС (синтаксичне представлення) та графічне зображення у вигляді файлу експорту ГС (візуальне представлення). Функції міжрівневої трансформації включають:

– функцію трансляції промпту в DSL, що виконується СШІ;
– функцію рендерингу DSL в ГС та експорту графічного файлу;
– функцію уточнення промпту на основі оцінки людиною адекватності отриманого візуального представлення (зворотний зв'язок).

Такий спосіб дозволяє побудувати дискретну динамічну систему графічного моделювання ПЗ з ітеративним уточненням. Представлений спосіб застосування ШІ для створення та реверс-інжинірингу графічних моделей ПЗ дозволяє підвищити загальну ефективність реалізації процесів життєвого циклу (ЖЦ) ПЗ за рахунок поєднання інтелектуальної та репрезентативної функцій в процесі створення та аналізу ПЗ.

Ключові слова: штучний інтелект, графічні моделі програмного забезпечення, DSL.

Постановка проблеми. Моделювання ПЗ із застосування графічних схем є давньою практикою, яка дозволяє розробнику більш ефективно виконувати свою роботу, опрацьовуючи зручні візуальні представлення різних аспектів програмної системи. Сучасні інструменти дозволяють створювати безліч подібних графічних моделей, таких як: схеми бізнес-процесів, різноманітні діаграми UML (Unified Modeling Language), схеми алгоритмів, архітектурні схеми, функціональні та структурні схеми і т.п. Частина таких інструментів інтегрується із системами програмування, дозволяючи генерувати окремі програмні артефакти (фрагменти коду, розділи документації, окремі тести) з графічних моделей, а також частково виконувати реверс-інжиніринг графічних моделей з коду та підтримувати їх синхронізацію.

З появою систем генеративного штучного інтелекту на основі великих мовних моделей (ВММ) виникла потреба у застосуванні інтелектуальних можливостей СШІ для створення та опрацювання графічних моделей ПЗ. Але проблема полягає в тому, що ВММ можуть

“розуміти” опис графічної моделі природною мовою, але не можуть її якісно відобразити, хоча й самі здатні описати її зображення мовними конструкціями. На сьогоднішній день СШІ мають обмежені графічні здібності у порівнянні зі спеціалізованими графічними редакторами. В кращому випадку вони можуть генерувати векторні зображення, наприклад, UML-діаграм певної якості, але в графічній формі вони не зручні для подальшого опрацювання моделі. З іншого боку, на сьогодні СШІ навряд чи можуть якісно опрацювати структуровану графічну схему у формі зображення (хоча й існують прототипи мультимодельних СШІ, які поєднують комп’ютерний зір та обробку природної мови) і зрозуміти її семантику, бо, власне, це і не є завданням ВММ, які призначені для опрацювання саме мовних описів моделей, а не їх графічних представлень.

Іншою тенденцією є додавання можливостей ВММ до ГС у вигляді інтелектуальних помічників, що дозволяє на основі описів природною мовою генерувати та аналізувати задані діаграми. Але такий підхід має певні обмеження як у функціональному, так і в архітектурному аспектах:

– більшість існуючих рішень реалізують односторонню взаємодію, де ШІ виконує генерацію моделей на основі текстових описів, але не здатен повноцінно інтерпретувати або модифікувати вже створені графічні структури;

– модулі ШІ демонструють обмежену здатність до контекстного аналізу, що ускладнює обробку складних технічних описів, багаторівневих моделей та нестандартних нотацій;

– більшість графічних редакторів не підтримують повноцінну інтеграцію з СШІ через API (Application Programming Interface), що ускладнює автоматизацію редагування, трасування змін та забезпечення зворотного зв’язку.

Тому суть проблеми полягає у розриві між семантичним представленням та візуальним відображенням моделей ПЗ, що робить актуальним пошук нових способів інтеграції існуючих СШІ та ГС для підвищення ефективності розробки ПЗ.

Аналіз останніх досліджень та публікацій. Застосуванню СШІ для графічного моделювання ПЗ присвячено багато робіт та створено низку інструментів, в яких використовуються переваги інтелектуальних систем та пропонуються рішення часткових задач створення графічних моделей ПЗ.

В роботах [1]-[3] досліджується використання генеративного ШІ для створення з тексту або голосу BPMN-моделей та їх аналізу, застосування ШІ-помічників, демонструються переваги мультимодальних представлень для автоматизованого моделювання бізнес-процесів.

Робота [4] присвячена огляду сучасних практик застосування ШІ в інженерії ПЗ, включаючи автоматизоване моделювання, генерацію коду, оптимізацію архітектури та прогнозування дефектів.

В роботах [5]-[7] показано приклади використання ШІ для створення C4-діаграм архітектури ПЗ з акцентом на пояснення структури коду для команд розробників на основі текстових описів з використанням ГС PlantUML та інтеграції з DevOps.

В [8] представлено інструмент, який генерує ERD-діаграми з текстових описів баз даних, використовуючи ШІ для розпізнавання сутностей, атрибутів і зв’язків.

В статті [9] проведено аналіз викликів інтеграції ШІ з застарілими системами, зокрема через IDEF0-моделювання, для модернізації інфраструктури.

В статті [10] запропоновано систему динамічного генерування UML з вихідного коду з використанням генеративного ШІ для автоматизованого документування та аналізу ПЗ.

В роботі [11] порівнюються результати виконання реверс-інжинірингу на основі ВММ та модельного підходів для задач абстрагування програм на Python та Java до формальної мови OCL (Object Constraint Language), а також застосування комбінованого підходу для покращених результатів перевірки коректності програм.

Робота [12] присвячена аналізу впливу якості промптів на якість генерування графічних схем ШІ та представлено систему, яка демонструє здатність покращувати подібні промпти.

Таким чином, аналіз останніх публікацій свідчить про актуальність пошуку способів застосування СШІ у взаємодії з ГС в задачах моделювання ПЗ.

Метою статті є розробка способу інтеграції існуючих СШІ та ГС моделювання ПЗ, що дозволить підвищити рівень автоматизації процесів створення та реверс-інжинірингу графічних моделей ПЗ за рахунок поєднання інтелектуальної та репрезентативної функцій в єдиній системі.

Виклад основного матеріалу дослідження. Під графічними моделями ПЗ будемо розуміти візуальні представлення структури, поведінки та взаємодії компонентів програмної системи, які використовуються для аналізу, проєктування, документування та комунікації між учасниками розробки, а також для автоматизації генерування програмного коду, тестів, документації та інших програмних артефактів.

В роботі використано наступні терміни відносно графічних моделей:

– *метод* – це метод моделювання (наприклад, IDEF0 як метод функціонального моделювання);

– *нотація* – це набір графічних символів і правил зображення графічних моделей певного методу (наприклад, UML-діаграми);

– *стандарт* – це офіційно затверджений опис нотації (наприклад, UML 2.5 від OMG);

– *модель* – це конкретна реалізація нотації для певної системи, конкретна модель;

– *формат* – це спосіб збереження моделей (наприклад, XMI для UML, BPMN XML).

Технологічне підґрунтя. Моделі в ГС зберігаються у вигляді метаданих певного формату, які описують графічні зображення відповідних схем. Ці моделі можуть мати формат неструктурованих або бінарних даних, структурованих текстів (типу XML, JSON), або опису спеціальною мовою (Domain-Specific Language, DSL). Тому ідея полягає у використанні саме мовних DSL-форматів опису графічних моделей, які були б одночасно “зрозумілими” як для СШІ, так і для популярних ГС. В більш широкому сенсі завдання полягає у пошуку мовних інтерфейсів між людиною, СШІ та ГС.

Методика досліджень. Для розробки способу було виконано наступні етапи досліджень:

1. Обґрунтування та вибір методів графічного моделювання та нотацій графічних моделей, які вирішують основні завдання моделювання ПЗ.
2. Аналіз форматів представлення моделей в ГС та визначення форматів, найбільш придатних для обміну моделями між ГС та СШІ.
3. Розробка способу інтеграції СШІ з ГС.

Отримані результати. В результаті проведених експериментів та досліджень отримано наступні результати.

1. Обґрунтування та вибір методів графічного моделювання ПЗ. За результатами дослідження було обрано наступні методи графічного моделювання ПЗ та відповідні графічні нотації, які використовуються під час реалізації найбільш суттєвих процесів ЖЦ ПЗ:

– *схеми бізнес-процесів* в нотації BPMN (Business Process Model and Notation) – візуалізують бізнес-логіку, яку реалізує ПЗ, а також застосовуються для опису технологій розробки ПЗ;

– *діаграми “сутність-зв’язок”* ERD (Entity-Relationship Diagram) – призначені для загальної візуалізації сутностей, їх атрибутів і зв’язків, частіше використовуються для проєктування баз даних (БД);

– *діаграми IDEF* (Integration DEFinition) – для моделювання різних аспектів систем, які використовуються для опису бізнес-процесів, даних та функцій (IDEF0 – моделювання функцій, IDEF1X –моделювання даних та IDEF3 – документування процесів);

– *архітектурні діаграми C4* (Context, Container, Component, Code) – ієрархічна модель візуалізації архітектури ПЗ, що спирається на наявні техніки моделювання UML, ERD;

– *UML-діаграми* – використовуються для аналізу, проєктування та розробки ПЗ.

2. Аналіз форматів графічних моделей. Для аналізу форматів представлення графічних моделей було розглянуто декілька популярних ГС, а також проаналізована наявність відповідного стандарту та формальної метамоделі.

Формальна метамодель – це специфікація, яка описує типи елементів, які можуть бути в моделі (класи, атрибути, зв'язки тощо), відношення між ними (агрегація, асоціація, наслідування і т.п.), обмеження (наприклад, що атрибут має бути унікальним), семантику – що означає кожен елемент. Метамодель потрібна для валідації моделей (перевірки коректності), інтероперабельності (обміну між системами, наприклад, через DSL або XMI), автоматизації генерації артефактів (коду, документації, трансформацій) та формалізації знань (чіткого визначення понять). Наприклад, стандарт Meta-Object Facility (MOF), розроблений консорціумом OMG (Object Management Group) для формального опису мов моделювання та метамоделей, включає чотири рівні метамоделювання, що представлені в таблиці 1.

Таблиця 1 – Рівні метамоделювання MOF

Рівень	Назва	Опис
M0	Дані	Конкретні екземпляри (реальні об'єкти)
M1	Модель	Модель системи (наприклад, UML-діаграма)
M2	Метамодель	Описує структуру моделей (наприклад, UML метамодель)
M3	Мета-метамодель	Описує структуру метамоделей (наприклад, MOF)

Наявність стандартів опису метамоделі для різних нотацій представлена в таблиці 2.

Таблиця 2 – Формальні метамоделі графічних нотацій

Нотація	Формальна метамодель	Стандарт	Основні формати представлення
BPMN	BPMN 2.0 Metamodel	OMG	BPMN XML (.bpmn), XMI
ERD	Немає єдиної метамоделі	Відсутній	.erwin, Visio (.vsdx), XML, JSON, UML/XMI
UML	UML Metamodel (MOF-based)	OMG	XMI (.xmi), JSON (StarUML), PlantUML (.puml)
IDEF	Частково формалізована	FIPS PUB 183	PDF, Draw.io XML (.drawio), Visio (.vsdx), частково .erwin (IDEF1X)
C4	C4 Model v1.0 Metamodel	Відсутній	Structurizr DSL (.dsl), JSON, YAML

Для вибору форматів представлення моделей, придатних для обміну з СШІ, була проведена класифікація метамоделей за типом представлення наступним чином.

1. *Мовні метамоделі* – це метамоделі, які описані через спеціалізовані метамодельні мови, такі як MOF, Ecore (метамодельна мова Eclipse Modeling Framework (EMF)), KM3 (Kernel Meta Meta Model), яка популярна в ATL/Model Transformation, та MetaGME – мова метамоделювання в Generic Modeling Environment.

Мовні метамоделі найбільш придатні для безпосередньої інтеграції СШІ та ГС.

2. *XML-подібні метамоделі* – це метамоделі, представлені у вигляді XML-структур, які часто використовуються для обміну між інструментами, такі як XMI (XML Metadata Interchange) – стандарт OMG для серіалізації UML, BPMN, SysML моделей, BPMN 2.0 XML – XML-опис метамоделі бізнес-процесів, Archimate XML – для моделей архітектури підприємства, Ecore XML – серіалізація метамоделей EMF.

Цей тип метамоделей не дуже підходить для безпосереднього обміну моделями між СШІ та ГС внаслідок змінюваності їх структур в різних версіях та в різних ГС, але може бути використаний шляхом їх трансформації в DSL, наприклад, на основі XSLT, з використанням програмних парсерів, або через додаткові промпти до СШІ з використанням шаблонів. *XSLT*

(Extensible Stylesheet Language Transformations) – це мова для трансформації XML-документів у інші формати (HTML, текст, інший XML, JSON, PlantUML тощо).

3. *Неструктуровані та бінарні метамоделі* – це метамоделі, збережені у власних форматах інструментів, таких як Enterprise Architect (.eap, .qea) – зберігає UML/BPMN метамоделі у бінарному вигляді, MagicDraw (.mdzip) – зберігає архів XMI та бінарні дані, Rational Rose (.mdl) – використовує старий бінарний формат UML. Також IDEF0/IDEF1X часто описуються тільки графічним зображенням або в документах без формальної структури, а ERD – це нотація без єдиної метамоделі.

Ці формат не підходять для взаємодії ГС з СШІ або потребують спеціалізованих програмних трансформерів для потрібних форматів.

Формати збереження графічних моделей можна класифікувати наступним чином (див. Таблицю 3).

Таблиця 3 – Класифікація форматів графічних моделей

Тип	Формат	Приклади	Придатність для обміну
Мовні	DSL	MOF, Ecore, KM3	Висока
XML-подібні	XML	XMI, BPMN XML	Середня (залежить від ГС та версії)
Неструктуровані або бінарні	Власні	.eap, .mdzip для UML; IDEF, ERD	Низька

Властивості DSL-форматів графічних моделей, що визначають їх переваги для застосування в якості мови інтеграції СШІ та ГС, полягають у наступному:

– текстовий опис моделі “зрозумілий” одночасно для людини, СШІ та ГС;

– синтаксис мови є простим, що не потребує складних конструкцій опису моделі;

– є можливість не вказувати деталі відображення моделі (розміри, координати, шрифт), що дозволяє акцентуватись на логіці та семантиці, замість фізичного представлення моделі;

– наявність формальної метамоделі розглядається як додаткова перевага, що дозволяє виконувати верифікацію моделі.

В результаті дослідження було обрано наступні мови опису графічних моделей (DSL-формати метамоделей), найбільш придатних для інтеграції СШІ та ГС:

– *PlantUML* – найпотужніша текстова мова, яка підтримує багато типів діаграм (UML, C4, ERD та інші);

– *Mermaid* – зручна для Markdown-середовищ, підтримує понад десяток типів діаграм, включаючи блок-схеми, діаграми послідовностей, графіки Ганта та ERD;

– *Structurizr DSL* – спеціалізована мова для архітектурних моделей C4;

– *Eraser DSL* – це текстовий формат для створення діаграм у онлайн-редакторі Eraser для технічної документації (BPMN, ERD, UML);

– *Nomnoml* – легка мова для UML-діаграм класів з онлайн-рендерингом та експортом;

– *Graphviz DOT* – універсальна мова для графів, часто використовується для діаграм станів, дерев, залежностей;

– *D2* – підтримує широкий спектр діаграм, включаючи блок-схеми, архітектурні схеми, організаційні діаграми, мережеві топології, ERD та багато інших;

– *UMLet DSL* – проста мова для створення UML-діаграм;

– *dbdiagram.io DSL* – це мова моделювання БД на основі DBML (Database Markup Language), яка дозволяє описувати схеми БД, а потім генерувати ERD-діаграми;

– *QuickDBD DSL* – має простий синтаксис, схожий на Markdown (легка мова розмітки, яка дозволяє форматувати текст у простому вигляді, з можливістю конвертації в HTML), для створення ERD;

– *BPMN Sketch Miner DSL* – проста мова для BPMN-нотації;

– *Umple DSL* – мова, яка поєднує UML-моделювання з об’єктно-орієнтованим програмуванням, дозволяє генерувати як UML-діаграми, так і вихідний код (Java, PHP тощо).

Придатність обраних DSL-метамоделей для представлення моделей різних нотацій та наявність API представлено в таблиці 4, де позначка “+/–“ означає часткову придатність.

Таблиця 4 – Придатність DSL-метамоделей для графічних моделей

DSL	Придатність для створення моделі					ГС	API
	BPMN	ERD	C4	UML	IDEF		
PlantUML	+/–	+	+	+	+/–	PlantUML, IntelliJ, VS Code	+
Mermaid	+/–	+	+/–	+	–	Markdown-редактори, Live Editor, Obsidian	+
Structurizr DSL	–	–	+	+/–	–	Structurizr CLI, Structurizr Lite	+
Eraser DSL	+	+	+/–	+	+/–	Eraser Web Editor	–
Nomnoml	–	–	–	+/–	–	Nomnoml Web Editor, Markdown	+
Graphviz DOT	+/–	+	+/–	+/–	+	Graphviz Desktop, CLI, VS Code	+
D2	+/–	+	+	+/–	+/–	D2 CLI, Web Playground	+
UMLet DSL	–	–	–	+	–	UMLet Desktop	–
dbdiagram.io DSL	–	+	–	–	+/–	dbdiagram.io Web Editor	+
QuickDBD DSL	–	+	–	–	+/–	QuickDBD Web Editor	–
BPMN Sketch Miner DSL	+	–	–	–	–	BPMN Sketch Miner Web Editor	+
Umple DSL	–	–	–	+/–	–	Umple Online Editor, Eclipse Plugin	+

Проблемою є відсутність повноцінних DSL для моделей BPMN та IDEF (стандартно зберігаються в XML-подібних форматах): вони реалізовані тільки частково для окремих діаграм, але можуть бути замінені на аналогічні за рахунок інших нотацій або створення відповідних трансформерів. Іншою проблемою є відхилення від стандартів графічної нотації певними ГС для окремих діаграм або схем.

Приклади синтаксису DSL та відповідних зображень представлено на рисунках 1-3.

PlantUML	Зображення

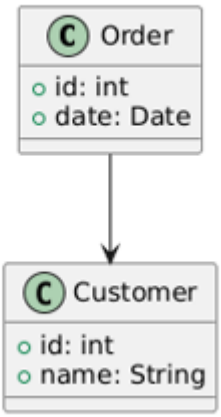
@startuml
class Order {
+id: int
+date: Date
}
class Customer {
+id: int
+name: String
}
Order --> Customer
@enduml
	

Рисунок 1 – Приклад діаграми класів UML в ГС PlantUML

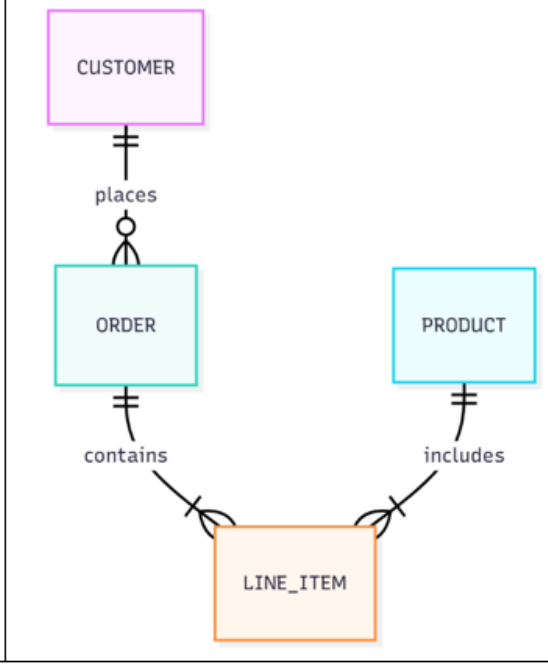
Mermaid (ERD)	Зображення
erDiagram CUSTOMER ||--o{ ORDER : places ORDER ||--|{ LINE_ITEM : contains PRODUCT ||--|{ LINE_ITEM : includes	

Рисунок 2 – Приклад діаграми ERD в ГС Mermaid

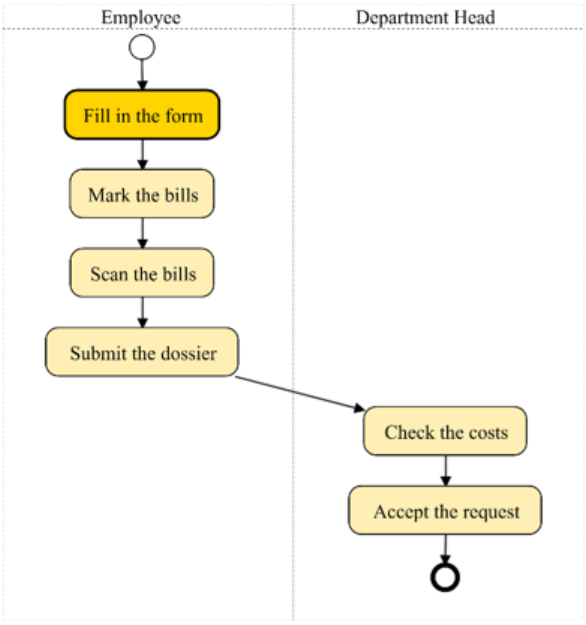
BPMN Sketch Miner	Зображення
Employee: Fill in the form Mark the bills Scan the bills Submit the dossier Department Head: Check the costs Accept the request	

Рисунок 3 – Приклад діаграми BPMN в ГС BPMN Sketch Miner

3. Розробка способу інтеграції СШІ з ГС. Суть способу, що пропонується, полягає у використанні моделі семантично-візуального рендерингу на основі композиції функцій трансформації опису графічної моделі ПЗ з природної мови у візуальне зображення. В основі способу лежить принцип трирівневої архітектури представлення графічних моделей ПЗ, що включає:

1 рівень (*Prompt*): семантичне представлення графічної моделі у вигляді промпту природною мовою для забезпечення мовного інтерфейсу між людиною та СШІ;

2 рівень (DSL): синтаксичне представлення графічної моделі у форматі DSL певної ГС для забезпечення інтерфейсу між СШІ та ГС більш формальною машиночитною мовою;

3 рівень (Image): візуальне представлення моделі у форматі графічного зображення для забезпечення інтерфейсу між ГС та людиною в якості цільової моделі та забезпечення зворотного зв'язку для оцінки відповідності графічного зображення вхідному промпту.

Виходячи з цього, ПЗ може бути описано сукупністю графічних моделей (не виключаючи й текстові описи), кожна з яких може бути представлена наступним чином:

$$GM = \langle P, D, I \rangle, \quad (1)$$

де GM – графічна модель ПЗ;

P – *Prompt*;

D – *DSL*4

I – *Image.*

Процес трансформації рівнів представлення моделі можна описати як композицію функцій:

$$f(P) = f_2(f_1(P)) = I, \quad (2)$$

де $f_1 : P \to D$ – трансляція промпту в DSL;

$f_2 : D \to I$ – рендеринг DSL у графічне зображення.

Для можливості уточнення промпту на основі оцінки адекватності візуального результату (*зворотний зв'язок*) можна використати функцію $f_3 : I \to P$, що дозволяє побудувати дискретну динамічну систему з ітеративним уточнення:

$$P_{i+1} = f_3(f_2(f_1(P_i))), \quad (3)$$

де P_i та P_{i+1} – це промпти на i та $i+1$ ітераціях відповідно.

Можна виділити наступні властивості трансформацій моделі (1) з урахуванням (2), (3):

- f_1 реалізується СШІ для обраної DSL певної ГС;
- f_2 реалізується ГС для обраного графічного формату (файлу експорту зображення);
- f_3 реалізується людиною на основі аналізу отриманого зображення.

Таким чином, забезпечується трансформація семантичного (мовного) опису графічної моделі у графічне зображення через проміжний формат DSL. Такий спосіб має наступні позитивні властивості:

- забезпечується зворотній зв'язок між формулюванням промпту природною мовою та оцінкою адекватності графічної моделі людиною;
- можливість ітеративного покращення графічної моделі шляхом переформулювання та уточнення промпту завдяки зворотному зв'язку;
- достатньо зберігати модель у форматі $\langle P, D \rangle$, що забезпечує збереження семантики і контексту та однозначну трансформацію в графічне представлення (зберігати DSL доцільно внаслідок неоднозначного його повторного генерування з промпта та можливу наявність ручного корегування DSL);
- можливість перетворення однієї семантичної моделі в різні DSL, а також можливість трансформації одного DSL в інші DSL;
- можливість перетворення DSL в різні графічні формати (залежить від можливостей експорту ГС);
- можливість генерування програмного коду з DSL для певних графічних моделей (наприклад, з діаграм класів, діяльностей);
- можливість генерування програмної документації з DSL (як опис та пояснення графічних схем, архітектури, логіки та поведінки);
- можливість реверс-інжинірингу графічних моделей з програмного коду або з документації у формат DSL з подальшим перетворенням в графічний формат;
- можливість часткового відтворення промпту з DSL.

Реалізація та застосування цього способу можлива як в ручному режимі, так і шляхом створення програмного інструменту, який інтегрує СШІ та ГС в єдиній системі на основі API.

Висновки. Представлений спосіб застосування ШІ для створення та реверс-інжинірингу графічних моделей ПЗ дозволяє отримати наступний позитивний ефект:

– дозволяє інтегрувати інтелектуальні можливості нових СШІ та функціональність існуючих популярних ГС, що підвищує загальну ефективність процесів ЖЦ ПЗ;

– трирівневе представлення графічних моделей ПЗ забезпечує як збереження вихідної семантики моделі, зрозумілої для людини, так і можливість опрацювання моделі СШІ та ГС;

– спосіб придатний для практичної реалізації інтегрованої системи, яка дозволяє за рахунок API об'єднувати в єдиній системі інтелект СШІ, функціональність ГС та потужність систем програмування.

В якості перспектив розвитку цього способу можна розглядати застосування СШІ для ефективної трансформації графічних моделей між форматом XML та DSL, а також застосування мультиагентної архітектури.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

[1] D.N. Dolha, and R.A. Buchmann, "Generative AI for BPMN Process Analysis: Experiments with Multi-modal Process Representations," in *Proc. Perspectives in Business Informatics Research*, vol. 529, pp. 19-35, 2024. doi: https://doi.org/10.1007/978-3-031-71333-0_2.

[2] J. Köpke, and A. Safan, "Introducing the BPMN-Chatbot for Efficient LLM-Based Process Modeling," in *Proc. of the Best BPM Disser. Aw., Doc. Cons., and Demonstr. & Res. Forum co-located with 22nd Inter. Conf. on Bus. Proc. Man. (BPM 2024)*, Krakow, Poland, 2024, vol. 3758, pp/86-90, 2024. [Online]. Available: https://ceur-ws.org/Vol-3758/paper-15.pdf. Accessed on: Sep. 11, 2025.

[3] J.T. Licardo, N. Tankovic, and D. Etinger, "BPMN Assistant: An LLM-Based Approach to Business Process Modeling", *arXiv preprint*, 2025. doi: https://doi.org/10.48550/arXiv.2509.24592.

[4] M. Alenezi, and M. Akour, "AI-Driven Innovations in Software Engineering: A Review of Current Practices and Future Directions," *Applied Sciences. AI in Software Engineering: Challenges, Solutions and Applications (Special Issue)*, vol. 15, iss. 3, art. 1344, 26 p., 2025. doi: https://doi.org/10.3390/app15031344.

[5] S. Heuel, "Creating Architecture Diagrams with C4 and AI", *Heuel Blog*, 2025. [Online]. Available: https://blog.heuel.org/2025/01/creating-architecture-diagrams-with-c4-and-ai/. Accessed on: Sep. 17, 2025.

[6] "C4 Model for Mobile App Architecture with AI-Powered Diagramming", *Diagrams-AI*, 2025. [Online]. Available: https://www.diagrams-ai.com/blog/c4-model-for-mobile-app-architecture/. Accessed on: Sep. 17, 2025.

[7] "C4-PlantUML: Combining PlantUML and C4 with AI Tools", *PlantUML Community*, *GitHub Repository*, 2024. [Online]. Available: https://github.com/plantuml-stdlib/C4-PlantUML. Accessed on: Aug. 05, 2025.

[8] "Free ERD Diagram Maker: AI-Generated Database Diagrams," *MyMap.AI*, 2023. [Online]. Available: https://www.mymap.ai/er-diagram-tool. Accessed on: Aug. 08, 2025.

[9] M. Singh, "Integrating Artificial Intelligence with Legacy Systems: A Systematic Analysis of Challenges and Strategic Considerations," *European Journal of Computer Science and Information Technology*, vol. 13, iss. 32, pp. 38-45, 2025. doi: https://doi.org/10.37745/ejcsit.2013/vol13n323845.

[10] S.S. Saxena, I. Alam, V. Sharma, U. Vats, and V. K. Chundury, "Dynamic creation of UML diagrams using generative AI", *Technical Disclosure Commons. Defensive Publications Series, Art. 7993*, 7 p., 2025. [Online]. Available: https://www.tdcommons.org/dpubs_series/7993.

Accessed on: Sep. 12, 2025.
[11] H.A. Siala, and K. Lano, "A Comparison of Large Language Models and Model-Driven Reverse Engineering for Reverse Engineering", *Frontiers in Computer Science*, vol. 7, art. 1516410. doi: https://doi.org/10.3389/fcomp.2025.1516410.
[12] D. Lande, and L. Strashnoy, "VizPrompt: A Framework for Structured Prompt Generation in Diagram Synthesis using Generative AI", *SSRN*, 2025. [Online]. Available: https://ssrn.com/abstract=5628310. Accessed on: Sep. 10, 2025.

Стаття надійшла 31.10.2025.

REFERENCE

[1] D.N. Dolha, and R.A. Buchmann, "Generative AI for BPMN Process Analysis: Experiments with Multi-modal Process Representations," in *Proc. Perspectives in Business Informatics Research*, vol. 529, pp. 19-35, 2024. doi: https://doi.org/10.1007/978-3-031-71333-0_2.
[2] J. Köpke, and A. Safan, "Introducing the BPMN-Chatbot for Efficient LLM-Based Process Modeling," in *Proc. of the Best BPM Disser. Aw., Doc. Cons., and Demonstr. & Res. Forum co-located with 22nd Inter. Conf. on Bus. Proc. Man. (BPM 2024)*, Krakow, Poland, 2024, vol. 3758, pp/86-90, 2024. [Online]. Available: https://ceur-ws.org/Vol-3758/paper-15.pdf. Accessed on: Sep. 11, 2025.
[3] J.T. Licardo, N. Tankovic, and D. Etinger, "BPMN Assistant: An LLM-Based Approach to Business Process Modeling", *arXiv preprint*, 2025. doi: https://doi.org/10.48550/arXiv.2509.24592.
[4] M. Alenezi, and M. Akour, "AI-Driven Innovations in Software Engineering: A Review of Current Practices and Future Directions," *Applied Sciences. AI in Software Engineering: Challenges, Solutions and Applications (Special Issue)*, vol. 15, iss. 3, art. 1344, 26 p., 2025. doi: https://doi.org/10.3390/app15031344.
[5] S. Heuel, "Creating Architecture Diagrams with C4 and AI", *Heuel Blog*, 2025. [Online]. Available: https://blog.heuel.org/2025/01/creating-architecture-diagrams-with-c4-and-ai/. Accessed on: Sep. 17, 2025.
[6] "C4 Model for Mobile App Architecture with AI-Powered Diagramming", *Diagrams-AI*, 2025. [Online]. Available: https://www.diagrams-ai.com/blog/c4-model-for-mobile-app-architecture/. Accessed on: Sep. 17, 2025.
[7] "C4-PlantUML: Combining PlantUML and C4 with AI Tools", *PlantUML Community*, *GitHub Repository*, 2024. [Online]. Available: https://github.com/plantuml-stdlib/C4-PlantUML. Accessed on: Aug. 05, 2025.
[8] "Free ERD Diagram Maker: AI-Generated Database Diagrams," *MyMap.AI*, 2023. [Online]. Available: https://www.mymap.ai/er-diagram-tool. Accessed on: Aug. 08, 2025.
[9] M. Singh, "Integrating Artificial Intelligence with Legacy Systems: A Systematic Analysis of Challenges and Strategic Considerations," *European Journal of Computer Science and Information Technology*, vol. 13, iss. 32, pp. 38-45, 2025. doi: https://doi.org/10.37745/ejcsit.2013/vol13n323845.
[10] S.S. Saxena, I. Alam, V. Sharma, U. Vats, and V. K. Chundury, "Dynamic creation of UML diagrams using generative AI", *Technical Disclosure Commons. Defensive Publications Series, Art. 7993*, 7 p., 2025. [Online]. Available: https://www.tdcommons.org/dpubs_series/7993. Accessed on: Sep. 12, 2025.
[11] H.A. Siala, and K. Lano, "A Comparison of Large Language Models and Model-Driven Reverse Engineering for Reverse Engineering", *Frontiers in Computer Science*, vol. 7,

art. 1516410. doi: https://doi.org/10.3389/fcomp.2025.1516410.

[12] D. Lande, and L. Strashnoy, “VizPrompt: A Framework for Structured Prompt Generation in Diagram Synthesis using Generative AI”, *SSRN*, 2025. [Online]. Available: https://ssrn.com/abstract=5628310. Accessed on: Sep. 10, 2025.

VOLODYMYR SOKOLOV

METHOD OF USING ARTIFICIAL INTELLIGENCE FOR CREATING AND REVERSE ENGINEERING GRAPHICAL SOFTWARE MODELS

The article presents a method of using generative artificial intelligence systems (AIS) based on large language models to build graphical software models from prompts and restore them from source code. The developed method is considered the basis for integrating AIS and graphical systems (GS), which are traditionally used to build graphical software models. In the process of research, such methods and notations of graphical modeling as BPMN, IDEF, ERD, UML and C4 were considered. In the process of analyzing the formats of representation of graphical models by different GS, it was determined that the most convenient for the use by AIS are language descriptions of models, unlike XML-like and binary formats. The idea of the method is to use the syntax of DSL (Domain Specific Language) of popular GS as intermediate languages for interaction between AIS and GS, which provides the possibility of both intelligent processing of the language description of the graphical model by AIS and its high-quality display by GS. The essence of the method is to represent each graphic model scheme by a three-level architecture and apply a composition of inter-level transformation functions. The three-level architecture of the graphic scheme representation includes an input prompt (model semantics), a DSL description of the scheme for the selected GS (syntactic representation) and a graphic image in the form of a GS export file (visual representation). The inter-level transformation functions include:

- a prompt translation function in DSL, which is performed by the AIS;
- a DSL rendering function by the GS and exporting the graphic file;
- a prompt refinement function based on a human assessment of the adequacy of the resulting visual representation (feedback).

This method allows to build a discrete dynamic system for graphical software modeling with iterative refinement. The presented method of using AI for creating and reverse-engineering graphic software models allows to increase the overall efficiency of implementing software life cycle (LC) processes by combining intellectual and representative functions in the process of creating and analyzing software.

Keywords: artificial intelligence, graphical software models, DSL.

Соколов Володимир Володимирович, кандидат технічних наук, доцент, доцент кафедри комп'ютерних наук та технологій штучного інтелекту у сфері кібербезпеки, Інститут спеціального зв’язку та захисту інформації Національного технічного університету України “Київський політехнічний інститут імені Ігоря Сікорського”, Київ, Україна. ORCID 0000-0002-5779-7167, v.sokolov@kpi.ua.

Sokolov Volodymyr, candidate of technical sciences, associate professor, associate professor at the computer science and artificial intelligence technologies in the field of cybersecurity academic department, Institute of special communication and information protection of National technical university of Ukraine “Igor Sikorsky Kyiv polytechnic institute”, Kyiv, Ukraine.