ДОДАТОК А

Фрагмент коду реалізації компіляції TypeScript у Solidity

```
import { // Використання імпорту
  ClassDeclaration,
  Decorator,
  ParameterDeclaration,
  Statement,
} from "ts-morph";

export const compileClassToContract = (classDeclaration: ClassDeclaration)
=> {
  return `contract ${classDeclaration?.getName()} {`; // створення классу
декларації
};

export const compileProperty = (property: PropertyDeclaration) => {
  const propertyName = property.getName();
  const propertyType = property.getType().getText();
  return `\t${propertyType} public ${propertyName};\n`;
};

export const compileMethodParams = (parameter: ParameterDeclaration) => {
  const paramName = parameter.getName();
  const paramType = parameter.getType().getText();
  return `${paramType} ${paramName}`;
};

export const compileMethodBody = (statement: Statement) => {
  const statementText = statement.getText();
  return `${statementText.replace("this.", "")}`;
};

export const lineBreak = (str: string) => {
  return `${str}\n}`;
};

export const spaceStart = (str: string, spaces: number) => {
```

```
  return `${" ".repeat(spaces)}${str}`;
};

export function addLineBreaks(str?: string, count = 1) {
  if (!str?.trim()) return "";
  const lineBreaks = "\n".repeat(count);
  return `${str}${lineBreaks}`;
}

export class ClassParser {
  private options: ClassParserOptions;

  constructor( // робота с конструктором
    private readonly cls: ClassDeclaration,
    options?: Partial<ClassParserOptions>,
  ) {
    this.options = { ...defaultOptions, ...options };
  }

  public compile() {
    const contractName = addLineBreaks(this.getContractName());
    const properties = addLineBreaks(
      this.getProperties()
        .map((property) => spaceStart(property, this.options.tabSize))
        .join("\n"),
      2,
    );

    const constructor = addLineBreaks(
      spaceStart(this.getConstructor(), this.options.tabSize),
      2,
    );
    const methods = this.getMethods().join("\n");

    return `${contractName}${properties}${constructor}${methods}}`;
  }

  private getContractName() {  //видача імені контракту
    return `contract ${this.cls.getName()} {`;
```

```
  }

  private getProperties() {
    return this.cls?.getProperties().map((property) => {
      const propertyName = property.getName();
      const propertyType = property.getType().getText();
      const viewModifiers = this.getDecoratorNames(
        property.getDecorators(),
      ).toLowerCase();

      return `${propertyType} ${viewModifiers} ${propertyName};`;
    });
  }

  private getMethods() {
    return this.cls?.getMethods().map((method) => {
      const methodParams = method
        .getParameters()
        .map(this.getMethodParameters)
        .join(", ");

      const viewModifiers = this.getDecoratorNames(
        method.getDecorators(),
      ).toLowerCase();
      const body =
method.getStatements().map(this.getMethodBody).join("\n");

      return spaceStart(
        `function ${method.getName()}(${methodParams}) ${viewModifiers}
{\n` + //використання та надання параметрів
          `${spaceStart(body, 2 * this.options.tabSize)}\n` +
          `${spaceStart("}", this.options.tabSize)}\n`,
        this.options.tabSize,
      );
    });
  }

  private getMethodBody(statement: Statement) {
    const statementText = statement.getText();
```

```
    return `${statementText.replace("this.", "")}`;
  }

  private getDecoratorNames(decorators: Decorator[]) {
    return decorators.map((decorator) => decorator.getName()).join(" ");
  }

  private getMethodParameters(param: ParameterDeclaration) {
    const paramName = param.getName();
    const paramType = param.getType().getText();
    return `${paramType} ${paramName}`;
  }

  private getConstructor() {
    const constructors = this.cls.getConstructors();

    if (constructors.length === 0) return "";

    const constructor = constructors[0];

    const params = constructor
      .getParameters()
      .map(this.getMethodParameters)
      .join(", ");

    const body =
constructor.getStatements().map(this.getMethodBody).join("\n");

    if (!body) return "";

    return (
      `constructor(${params}) {\n` +
      `${spaceStart(body, 2 * this.options.tabSize)}\n` +
      `${spaceStart("}", this.options.tabSize)}`
    );
  }
}
export type ClassParserOptions = { tabSize: number; }; const
defaultOptions: ClassParserOptions = { tabSize:
```

ДОДАТОК Б

Фрагмент версії реалізації паралельної компіляції

```
/* Файл index.ts */
import 'reflect-metadata';
import { Project } from "ts-morph";
import * as fs from "node:fs/promises";
import { fork } from 'child_process';
import { cpus } from 'os';
import path from 'path';

const project = new Project();
project.addSourceFilesAtPaths("contracts/**/*.ts");
const files = project.getSourceFiles();
const distFolder = "./compiled";

function createWorker() {
  return fork(path.join(__dirname, 'worker.ts'));
}

async function compileClass(classDeclaration: any, index: number):
Promise<string> {
  return new Promise((resolve) => {
    const worker = createWorker();
    worker.on('message', (result: string) => {
      resolve(result);
      worker.kill();
    });
    worker.send({ classDeclaration, index });
  });
}

async function processFile(file: any) {
  const classes = file.getClasses();
  const compiled = await Promise.all(
    classes.map((classDeclaration: any, index: number) =>
      compileClass(classDeclaration, index)
    )
  );
  return { file, compiled };
}

async function ensureDistFolder() {
  try {
    await fs.access(distFolder);
  } catch (e) {
    await fs.mkdir(distFolder);
  }
}

async function writeFile(file: any, compiled: string[]) {
  await fs.writeFile(
    `${distFolder}/${file.getBaseNameWithoutExtension()}.sol`,
    compiled.join('\n\n')
  );
```

```
}

(async () => {
  await ensureDistFolder();

  const processedFiles = await Promise.all(
    files.map(file => processFile(file))
  );

  await Promise.all(
    processedFiles.map(({ file, compiled }) =>
      writeFile(file, compiled)
    )
  );
})();

/* Файл worker.ts */
import { parentPort } from 'worker_threads';
import { ClassParser } from "@/parsers/class-parser";

parentPort?.on('message', (data: { classDeclaration: any, index: number })
=> {
  const { classDeclaration, index } = data;
  const contract = new ClassParser(classDeclaration).compile();
  const pragma = index === 0 ? "pragma solidity ^0.8.0;" : "";
  parentPort?.postMessage(`${pragma}\n\n${contract}`);
});
```

ДОДАТОК Г

Слайди презентації

TypeScript для розробки смарт-контрактів

Дослідження можливостей мови програмування TypeScript для розробки смарт-контрактів та аналіз покращення інструментів компіляції TypeScript програм в Solidity.

Кравцов Ігор, Офатенко Артем ІПЗм-23-3, Кабанік Юрій ІПЗм-23-2

Науковий керівник: МАРЇН Сергій, доц. Кафедри ПІ

Made with GAMMA

Дослідження

Актуальність та стан розвитку галузі

Стрімкий розвиток dApps вимагає нових інструментів. Solidity є домінуючою мовою для EVM. TypeScript — стандарт для інфраструктури dApps. Цей дуалізм створює потребу у надійній взаємодії та компіляції.

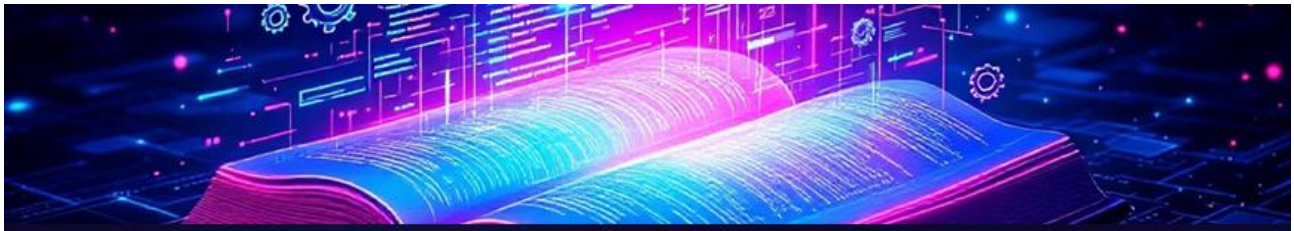
Визначення напряму дослідження

Вивчення інтеграції TypeScript у розробку смарт-контрактів для EVM. Аналіз існуючих інструментів взаємодії з Solidity. Обґрунтування викликів та перспектив прямої компіляції TypeScript у Solidity.

Об'єкт дослідження

Процеси розробки та компіляції ПЗ у блокчейн-середовищі. Взаємодія між TypeScript та Solidity. Інструментарій взаємодії та теоретична можливість транскомпіляції TypeScript у Solidity для EVM.

Made with GAMMA



Огляд літератури (аналогів)

Основні джерела та теорії

Компільоробудування (лексичний, синтаксичний аналіз). Віртуальні машини блокчейну (EVM). Формальна верифікація смарт-контрактів та компіляторів.

Важливі практичні джерела

- Hardhat, Truffle, ethers.js
- TypeChain, Abitype, Wagmi, Viem
- Azle для ICP, AssemblyScript для WASM

Прогалини у наявних дослідженнях

Відсутність зрілої та широко прийнятої компіляції TypeScript у Solidity. Фундаментальний семантичний розрив та обмеження EVM. Необхідність подальших теоретичних досліджень.

Постановка задачі

Проблема

Недостатня інтеграція TypeScript у Solidity. Відсутність прямої компіляції. Складність розробки, когнітивне навантаження.

Очікувані результати

Всебічний аналіз поточного стану TypeScript у Web3. Окреслення теоретичної прогалини та технічних викликів. Формулювання концептуальних підходів та рекомендацій.

Вплив

Закладення основи для майбутніх реалізацій. Підвищення ефективності та безпеки розробки смарт-контрактів.

Методологія

Методи дослідження

Теоретичний аналіз літератури та публікацій. Порівняльний аналіз TypeScript та Solidity. Системний аналіз архітектур Web3-інструментів. Методи індукції та дедукції.

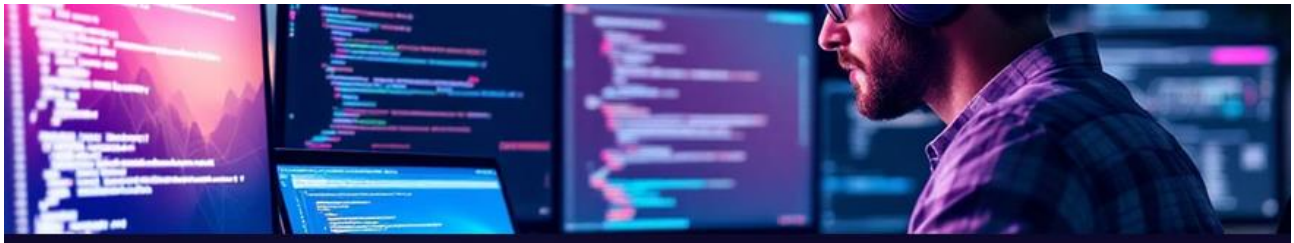
Використаний інструментарій та технології

- Компілятори: tsc, solc
- Фреймворки: Hardhat, Truffle
- Бібліотеки: ethers.js, web3.js
- Генерація типів: TypeChain, Abitype, Wagmi, Viem
- Аналіз коду: ts-morph, reflect-metadata
- Досвід проєктів: Azle, AssemblyScript

Архітектура системи для проведення експериментального дослідження

У контексті цього було розроблено прототип програмного забезпечення, що приймає на вхід TypeScript код, написаний спеціальним образом. Він проходить через процес парсингу для отримування AST-дерева, завдяки якому можна отримати головні програмні вузли TypeScript класу і перетворити його на Solidity контракт.

- **TypeScript Compiler API**
- **Робота з абстрактним синтаксичним деревом**
- **Structures**
- **Обробка типів**
- **Перетворення AST вузлів на Solidy контракт**
- Можливість обробляти паралельно декілька TypeScript файлів



Опис програмного забезпечення

Процес розробки є гібридним. Поєднує традиційне ПЗ та блокчейн-розробку.

Смарт-контракти	Solidity
Логіка dApp, тести, скрипти	TypeScript (Node.js)
Фреймворки розробки	Hardhat, Truffle
Взаємодія з блокчейном	ethers.js, web3.js
Генерація типових файлів	TypeChain, Abitype, Wagmi, Viem
Аналіз коду	ts-morph, reflect-metadata

Зміст проведеного експерименту

Методи

Порівняльний аналіз мов, системний аналіз інструментів, теоретичний аналіз архітектур.

Вхідні дані

Коди TypeScript, Solidity, ABI-файли, документація інструментів, наукові публікації.

Критерії оцінки

Типова безпека, ефективність взаємодії, складність розробки, потенційні витрати газу, досвід розробника.

Послідовність

Теоретичне обґрунтування, аналіз фундаментальних принципів, вивчення практичних підходів, визначення прогалин.

Вимірювання

Якісний характер (оцінка складності, переваг). Кількісні показники відсутні.

Аналіз отриманих результатів

Дослідження підтвердило актуальність TypeScript у Web3-розробці.

1

Актуальність TypeScript

Підтверджено його важливу роль у Web3-розробці.

2

Відсутність прямої компіляції

Немає зрілого рішення для TypeScript в Solidity.

3

Паралельна компіляція

Прагматичний та ефективний підхід, але не усуває семантичного розриву.

Результати впливають на існуючі теорії, підтверджуючи важливість проміжних представлень. Посилюється фокус на вдосконаленні інструментів для типобезпечної взаємодії. Майбутні дослідження повинні зосереджуватися на інноваційних підходах до семантичної трансляції та оптимізації для EVM.

Підсумки

Реалістичність результатів

Відображають поточний стан Web3-розробки. Пряма компіляція TypeScript у Solidity наразі не є практичним рішенням.

Корисність дослідження

Чітке окреслення теоретичної прогалини. Дорожня карта для майбутніх інновацій. Фокус на перспективних напрямках.

Можливий розвиток досліджень

Дизайн та прототипування проміжних представлень (IR). Розробка дослідницьких компіляторів. Інтеграція формальної верифікації. Компіляція TypeScript у WASM для блокчейнів.

ДОДАТОК Д

Апробація результатів роботи

ЛЬВІВСЬКИЙ НАУКОВИЙ ФОРУМ

СЕРТИФІКАТ

№ 25-1625

Даний сертифікат підтверджує, що

Kravtsov I., Ofatenko A., Kabannik Y., Mariyn S.

взяв(ла) участь у роботі

XV Міжнародній науково-практичній конференції

«Пріоритетні напрями досліджень в науковій та освітній діяльності»

тривалістю 15 годин / 0,5 кредиту ЄКТС

29-30 травня 2025 року

Голова оргкомітету конференції

В. Бондар

ДОДАТОК Е

Результат перевірки на академічний плагіат

StrikePlagiarism.com

Дата звіту 6/10/2025
Дата редагування ---

Звіт не був оцінений

Звіт подібності

метадані

Назва організації
Kharkiv National University of Radio Electronics
Заголовок
2025_М_ПІ_ІПЗМ-23-3_Офатенко_А_О_скорочений
Автор
Офатенко Артем Олегович
Науковий керівник / Експерт
Євген Кардаш
підрозділ
каф. ПІ

Обсяг знайдених подібностей

Коефіцієнт подібності визначає, який відсоток тексту по відношенню до загального обсягу тексту було знайдено в різних джерелах. Зверніть увагу, що високі значення коефіцієнта не автоматично означають плагіат. Звіт має аналізувати компетентна / уповноважена особа.

1.08% 1.08% КП 1

0.36% 0.36% КЦ

25
Довжина фрази для коефіцієнта подібності 2

14502
Кількість слів

116191
Кількість символів

ДОДАТОК Ж

Експертний висновок результатів перевірки кваліфікаційної роботи на відповідність оформлення вимогам ДСТУ 3008: 2015

Експертний висновок результатів перевірки кваліфікаційної роботи

студент (посада) | програмної інженерії (кафедра) | ІПЗм-23-3 (група)

Офатенко Артем Олегович

(прізвище, ім'я, по батькові)

Зауваження

Пункт ДСТУ 3008-2015	Зміст пункту	Сторінка кваліфікаційної роботи
1	2	3
	7.1 Загальні положення	
	7.3 Нумерація сторінок звіту	
	7.4 Нумерація розділів, підрозділів, пунктів, підпунктів	
	7.5 Рисунки	
	7.6 Таблиці	
	7.7 Переліки	
	7.8 Примітки	
	7.9 Виноски	
	7.10 Формули та рівняння	
	7.11 Посилання	
	7.13 Список авторів	
	7.14 Скорочення та умовні познаки	
	7.15 Додатки	

зауважень немає

Експерт _____ (підпис) Олена ОЛІЙНИК (прізвище, ініціали)

11.06.2025