



ІІТМО

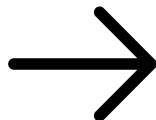
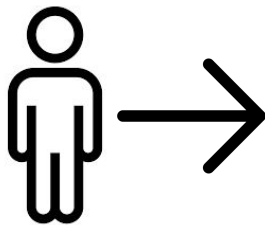
Разработка модуля рекомендации подсказок для системы автоматизированной проверки задач по программированию

Автор: Кривоносов Егор Дмитриевич

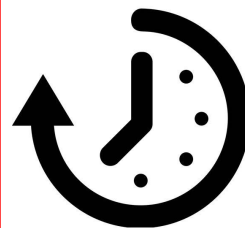
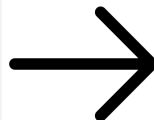
Санкт-Петербург, 2024

Проблема

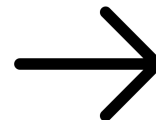
Student

A screenshot of a web-based code editor interface. The title bar says "code-n-test" and "Java". The code is as follows:

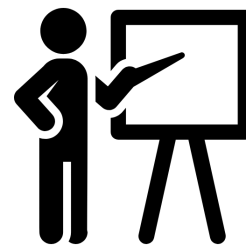
```
1 import java.util.Scanner;
2
3 public class Main {
4     public static void main(String[] args) {
5         Scanner scanner = new Scanner(System.in);
6         double a = scanner.nextDouble();
7         double b = scanner.nextDouble();
8         double c = scanner.nextDouble();
9         double discriminant = b * b - 4 * a * c;
10
11         if (discriminant > 0) {
12             double root1 = (-b + Math.sqrt(discriminant)) / (2 * a);
13             double root2 = (-b - Math.sqrt(discriminant)) / (2 * a);
14             System.out.println(root1 + ", " + root2);
15         } else if (discriminant == 0) {
16             double root = -b / (2 * a);
17             System.out.println(root);
18         }
19     }
20 }
```



1-2 weeks



Teacher





- **Цель:**

- Повысить качество и скорость оценки кода студентов за счёт автоматического выявления ошибок, генерации рекомендаций по их устранению и автоматизированного составления отчётов.

- **Решение:**

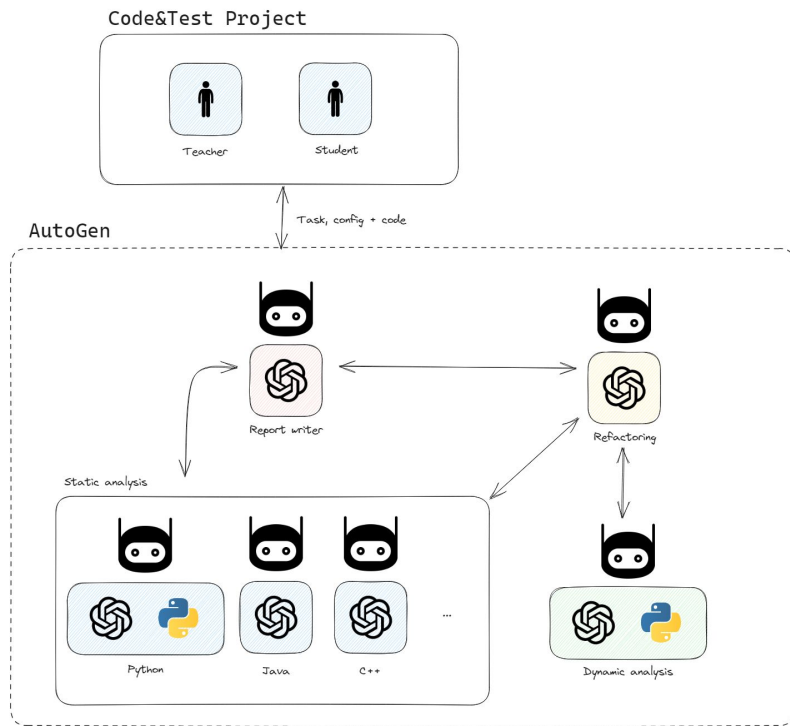
- Моя работа направлена на **разработку модуля подсказок** для обеспечения немедленной **автоматической обратной связи** по представленному коду в рамках платформы **Code&Test**.
- Этот инструмент облегчит среду обучения в режиме реального времени, позволяя студентам быстро выявлять и исправлять ошибки в написанном коде, тем самым ускоряя процесс обучения и повышая качество кода.



- Совместное использование статических и динамических анализаторов кода позволяет получить наилучший результат.
- Многие решения поддерживают только определенный язык программирования, без возможности его расширения.
- Существует очень мало решений, которые дают обратную связь по коду в понятной форме для любого студента с любым уровнем знаний в области программирования.
- В основном все решения платные или не имеют открытого доступа.

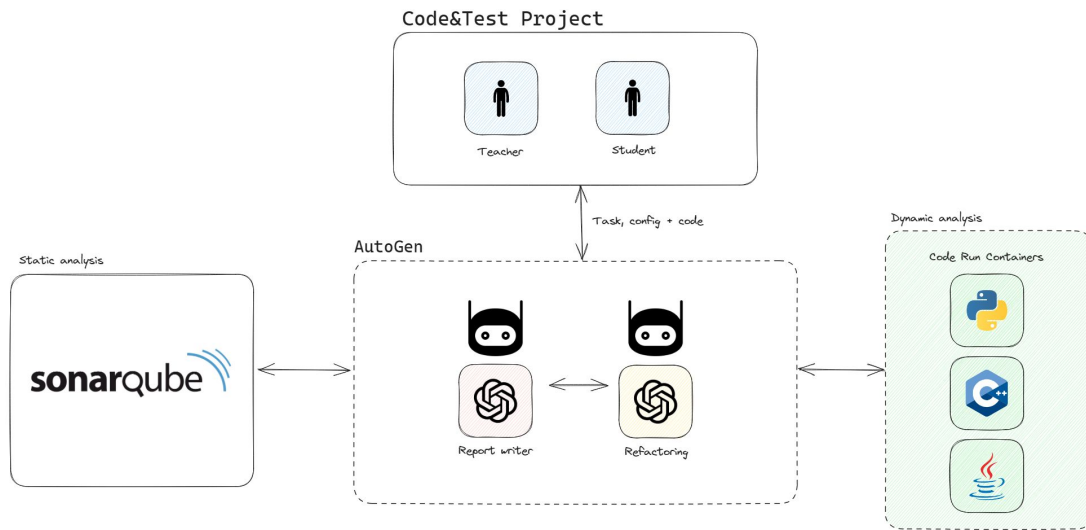


Архитектура решения



1. **4 вида агентов** со своими навыками. Навыки задаются в отдельном разделе AutoGen.
2. В качестве ИИ – будет использоваться **Gemini 1.5 Pro / ChatGPT 3.5 Turbo**.
3. Учитель предоставляет задание, тесты и ограничения (задание + конфиг), а студент решение на данное задание (код).
4. Данная архитектура поддерживает динамический запуск только кода написанного на Python.

Архитектура решения (ВКР)



- 1. **2 вида агентов** со своими навыками. Навыки задаются в отдельном разделе AutoGen.
- 2. В качестве ИИ – будет использоваться **Gemini 1.5 Pro**.
- 3. Учитель предоставляет задание, тесты и ограничения (задание + конфиг), а студент решение на данное задание (код).
- 4. В данной архитектуре запуск кода для динамического анализа будет производиться в отдельном сервисе проекта Code&Test в **Docker-контейнере**, а запуск статического анализа через Sonarqube или другую подобную систему.

Важные промпты агентов

Refactroing_Agent

The Refactoring Agent handles the code refactoring process using insights from Static and Dynamic Analysis agents to improve code quality and efficiency.

Refactor the provided code by combining feedback from static and dynamic analysis agents. Focus on optimizing the structure and efficiency of the code, while ensuring compliance with coding guidelines. Document the changes made and their justification. Save the new code to a file. Refactoring_Agent CAN ONLY GET INFORMATION FROM Static_Analysis_Python and Dynamic_Analysis_Python.

Report_Writer_Agent



Take into account the language that the user gave you when writing the report. Compile a detailed report based on the outputs from the Refactoring, Static, and Dynamic Analysis agents. Include explanations of errors, results of the code refactoring, and performance metrics. Present the report in a user-friendly format, ensuring it is comprehensive and informative. Take into account the language that the user gave you when writing the report.

Report structure:

1. Static analysis
[detailed analysis information is taken from the Static_Analysis_Python agent and duplicated]
2. Dynamic analysis (metrics)
[detailed analysis information is taken from the Dynamic_Analysis_Python agent and duplicated]
3. Refactoring
[updated code + information about changes]
4. Conclusion

You forward the final report to the chat manager and say "Terminate".

Результат в AutoGen Studio

AGENTS

Отчет

1. Статический анализ

Код, представленный для решения задачи умножения матриц, соответствует стандарту PEP8. Однако, были выявлены некоторые моменты для улучшения:

1. Рекомендуется добавить комментарии к функции `multiply_matrices`.
2. Рекомендуется использовать более информативные имена переменных.
3. Рекомендуется использовать безопасные методы преобразования входных данных.
4. Добавление дополнительных проверок на входные данные.

2. Динамический анализ (метрики)

Динамический анализ кода показал хорошую производительность и использование памяти при умножении матриц. Результаты обработки входных данных и выполнения операций умножения были эффективными.

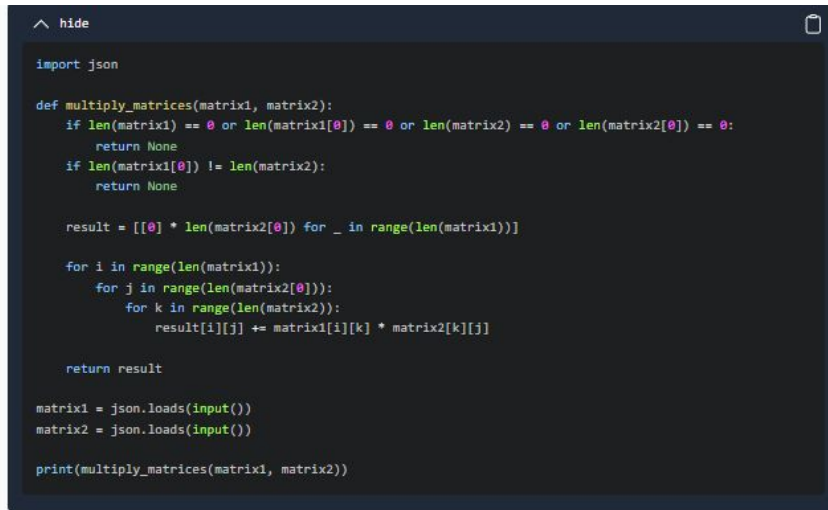
3. Рефакторинг

Проведен рефакторинг кода для улучшения его качества и безопасности:

- Добавлены комментарии к функции `multiply_matrices`.
- Использованы информативные имена переменных.
- Применено безопасное преобразование входных данных с использованием `json.loads`.
- Добавлены проверки на нулевые размеры матриц для избежания ошибок.

hide

Результат в AutoGen Studio



```
^ hide

import json

def multiply_matrices(matrix1, matrix2):
    if len(matrix1) == 0 or len(matrix1[0]) == 0 or len(matrix2) == 0 or len(matrix2[0]) == 0:
        return None
    if len(matrix1[0]) != len(matrix2):
        return None

    result = [[0] * len(matrix2[0]) for _ in range(len(matrix1))]

    for i in range(len(matrix1)):
        for j in range(len(matrix2[0])):
            for k in range(len(matrix2)):
                result[i][j] += matrix1[i][k] * matrix2[k][j]

    return result

matrix1 = json.loads(input())
matrix2 = json.loads(input())

print(multiply_matrices(matrix1, matrix2))
```

4. Заключение

Код был успешно отрефакторен с учетом рекомендаций статического анализа, что улучшило его читаемость, безопасность и надежность. Динамический анализ показал хорошую производительность кода при умножении матриц.

На данный момент **получилось реализовать** Архитектуру из 4 агентов и протестировать все в AutoGen с помощью модели **ChatGPT-3-Turbo**.

Результат показал, что использовать генеративные нейронные сети для статического анализа и динамического анализа не очень релевантно, т.к. они всегда дают разный результат, ещё не все модели позволяют запустить код с тестами. Также на это может тратиться достаточно большое количество времени.

Возможность писать отчет на любом языке – **успешно**.

Сам отчет написан всегда в **Markdown**, что позволит достаточно просто интегрировать **UI** платформы Code&Test без дополнительного изменения.

Gemini 1.5 Pro – лучше в написании текстов (отчетов по коду), чем **ChatGPT 4**.

ChatGPT 4 – лучше справляется с рефакторингом кода, чем **Gemini 1.5 Pro**.



- В рамках ВКР уже планируется реализовать архитектуру с 2 агентами (Report Writer и Refactoring, использовать **Gemini 1.5 Pro**). Использовать уже готовый Статический анализатор (например **Sonarqube**) и библиотеку для Динамического анализа кода и сбора метрик в отдельных Docker-контейнерах, совмещая с проверкой кода на антиплагиат.
- Запускать AutoGen через библиотеку **Autogen4j**, а не **AutoGen Studio**.
- Добавить поддержку **большего** количества языков программирования.
- Добавить возможность **создавать преподавателю тест-кейсы** кода студентов на основе написанной формулы самим преподавателем и настройки переменных для автоматической генерации тестов.



Спасибо за ваше внимание!

itMO *re than a*
UNIVERSITY