

**ПРАВИТЕЛЬСТВО РОССИЙСКОЙ ФЕДЕРАЦИИ  
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ  
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ  
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ  
«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»**

Факультет компьютерных наук  
Магистерская программа  
«Исследования и предпринимательство  
в искусственном интеллекте»

СОГЛАСОВАНО

Научный руководитель,  
доцент ФКН департамента  
программной инженерии, к.т.н.

\_\_\_\_\_ С. Л. Макаров  
« \_\_\_\_ » \_\_\_\_\_ 2025 г.

УТВЕРЖДАЮ

Академический руководитель  
образовательной программы  
«Исследования и предпринимательство  
в искусственном интеллекте»

\_\_\_\_\_ Д. С. Лялин  
« \_\_\_\_ » \_\_\_\_\_ 2025 г.

**Курсовая работа  
(проектная)**

на тему: **Разработка мультиагентной текстовой стратегической игры на  
основе оркестрируемых языковых моделей**

по направлению подготовки 01.04.02 «Прикладная математика и информатика»

ВЫПОЛНИЛ

студент группы ИПИИ  
образовательной программы  
01.04.02 «Прикладная  
математика и информатика»

\_\_\_\_\_ Н. С. Пеганов  
« \_\_\_\_ » \_\_\_\_\_ 2025 г.

# Реферат

Работа посвящена разработке мультиагентной текстовой стратегической игры на основе оркестрируемых языковых моделей и исследованию применимости современных LLM в качестве автоматизированного мастера игры в жанре военно-политических игр (ВПИ).

В работе рассмотрен полный цикл создания подобной системы, включая проектирование архитектуры мультиагентной среды, оркестрацию языковых моделей для выполнения различных игровых функций, а также анализ результатов пилотного запуска с реальными пользователями. Особое внимание уделяется методам преодоления типичных проблем языковых моделей в контексте игрового процесса: галлюцинации, сохранение долгосрочного контекста и вычислительные ограничения.

Результаты исследования демонстрируют потенциал современных LLM для автоматизации роли вердера в текстовых стратегических играх, выявляют ключевые технические и игровые ограничения существующих подходов, а также предлагают набор технических решений для повышения качества игрового опыта, включая применение систем на основе RAG и локальных моделей для снижения стоимости эксплуатации игровой системы.

Данная работа состоит из 11 страниц, 2 глав, 5 листингов, 1 таблицы, 2 приложений. Использовано 6 источников.

**Ключевые слова:** статический анализ; поиск ошибок; обработка исключений; Java; Kotlin; JVM; байткод.

# Abstract

This paper is dedicated to **smth**<sup>1</sup>.

In this work ...<sup>2</sup>.

**(TODO: Дописать)**

The paper contains 11 pages, 2 chapters, 5 listings, 1 table, 2 appendices. 6 sources are used.

**Keywords:** static analysis; search for defects; exception handling; Java; Kotlin; JVM; bytecode.

---

<sup>1</sup>TODO: Дописать

<sup>2</sup>TODO: Дописать

# Содержание

Реферат . . . . .	2
Abstract . . . . .	3
Используемые определения и термины . . . . .	5
Введение . . . . .	6
Глава 1 Обзор источников . . . . .	7
1.1 Какая-то подглава . . . . .	7
1.1.1 Какая-то подподглава . . . . .	7
1.1.1.1 Какой-то параграф . . . . .	7
1.1.1.2 Какой-то параграф . . . . .	7
1.1.1.3 Какой-то параграф . . . . .	7
1.1.1.4 Какой-то параграф . . . . .	7
1.1.2 Какая-то подподглава . . . . .	7
1.1.2.1 Какой-то параграф . . . . .	7
1.1.2.2 Какой-то параграф . . . . .	7
1.1.2.3 Какой-то параграф . . . . .	7
1.1.2.4 Какой-то параграф . . . . .	8
Выводы по главе . . . . .	8
Глава 2 Какая-нибудь ещё глава . . . . .	9
Заключение . . . . .	10
Список использованных источников . . . . .	11
Приложение А . . . . .	12
Приложение Б . . . . .	15

## Используемые определения и термины

**Common Vulnerabilities and Exposures (CVE)** – база данных общеизвестных уязвимостей информационной безопасности.

**Common Weakness Enumeration (CWE)** – общий перечень и система классификации слабых мест и уязвимостей программного обеспечения.

**Java** – строго типизированный объектно-ориентированный язык программирования общего назначения, разработанный компанией Sun Microsystems.

**Kotlin** – статически типизированный, объектно-ориентированный язык программирования, работающий поверх Java Virtual Machine и разрабатываемый компанией JetBrains.

**Абстрактное синтаксическое дерево (АСД, Abstract Syntax Tree, AST)** – одна из форм промежуточного представления программ в виде древовидной структуры.

**Анализ потока данных (Data Flow Analysis, DFA)** – один из основных методов анализа программ, позволяющий определить в каждой точке программы некоторую информацию о данных, которыми оперирует код.

**Байткод** – одна из форм промежуточного представления программ в виде инструкций, которые близки к машинным и могут быть интерпретированы при помощи виртуальной машины.

**Виртуальная машина Java (Java Virtual Machine, JVM)** – основная часть исполняющей системы Java, исполняющая байткод, полученный из исходного кода программы, на конкретной платформе путём трансляции байткода в машинные инструкции.

**Граф потока управления (ГПУ, Control Flow Graph, CFG)** – множество всех возможных путей выполнения программы, представленное в виде графа.

**Промежуточное представление (Intermediate Representation, IR)** – структура данных или код, используемый внутри компилятора или виртуальной машины для представления программ.

**Статический анализ кода** – анализ исходного кода на предмет ошибок и недочётов без непосредственного выполнения анализируемых программ.

# Введение

Пример введения.

Это пример ссылки на статью [1].

А это пример ссылки на онлайн-ресурс [2].

А это пример нескольких ссылок [3—6].

# Глава 1. Обзор источников

Текст главы 1

## 1.1. Какая-то подглава

Текст подглавы

### 1.1.1. Какая-то подподглава

Текст подподглавы

#### 1.1.1.1. Какой-то параграф

Текст параграфа

#### 1.1.1.2. Какой-то параграф

Текст параграфа

#### 1.1.1.3. Какой-то параграф

Текст параграфа

#### 1.1.1.4. Какой-то параграф

Текст параграфа

### 1.1.2. Какая-то подподглава

Текст подподглавы

#### 1.1.2.1. Какой-то параграф

Текст параграфа

#### 1.1.2.2. Какой-то параграф

Текст параграфа

#### 1.1.2.3. Какой-то параграф

Текст параграфа

#### **1.1.2.4. Какой-то параграф**

Текст параграфа

#### **Выводы по главе**

Текст Текст Текст Текст Текст Текст Текст



## Глава 2. Какая-нибудь ещё глава

Текст главы 2

## Заключение

Текст заключения

## СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. *Shelekhov V. I., Kuksenko S. V.* Data flow analysis of Java programs in the presence of exceptions // International Andrei Ershov Memorial Conference on Perspectives of System Informatics. — Springer. 1999. — С. 389—395.
2. Common Weakness Enumeration [Электронный ресурс] : CWE-703: Improper Check or Handling of Exceptional Conditions. — URL: <https://cwe.mitre.org/data/definitions/703.html> (дата обр. 31.12.2021).
3. SonarRules [Электронный ресурс] : Java static code analysis. — URL: <https://rules.sonarsource.com/java/tag/error-handling/> (дата обр. 22.04.2022).
4. SpotBugs [Электронный ресурс] : Bug descriptions. — URL: <https://spotbugs.readthedocs.io/en/latest/bugDescriptions.html> (дата обр. 22.04.2022).
5. Infer [Электронный ресурс] : List of all issue types. — URL: <https://fbinfer.com/docs/all-issue-types/> (дата обр. 22.04.2022).
6. Detekt [Электронный ресурс] : Exceptions Rule Set. — URL: <https://detekt.dev/exceptions.html> (дата обр. 22.04.2022).

## Пример приложения

Пример приложения. Какой-то текст. Какой-то текст. Какой-то текст. Какой-то текст. Какой-то текст. Какой-то текст. Какой-то текст. Какой-то текст. Какой-то текст. Какой-то текст.

Ссылка на приложение Б.

Тут ссылка на листинг 1.

А тут ссылка на листинг 3.

```

1 | @Deprecated("Reason")
2 | fun findScriptDefinition(project: Project, script: SourceCode): ScriptDefinition?
   | {
3 |     val scriptDefinitionProvider = ScriptDefinitionProvider.getInstance(project) ?:
   |         return null
4 |     ?: throw IllegalStateException("Unable to get script definition: ...")
5 |
6 |     return scriptDefinitionProvider.findDefinition(script) ?:
   |         scriptDefinitionProvider.getDefaultDefinition() // Comment
7 | }

```

Листинг 1 — Пример какого-то кода на Kotlin

```

1 | class Main {
2 |     public static ScriptDefinition findScriptDefinition(Project project, SourceCode
   |         script) {
3 |         ScriptDefinitionProvider scriptDefinitionProvider = ScriptDefinitionProvider.
   |             getInstance(project);
4 |         if (scriptDefinitionProvider == null) {
5 |             if (null == null) {
6 |                 throw IllegalStateException("Unable to get script definition: ...");
7 |             } else {
8 |                 return null;
9 |             }
10 |        }
11 |
12 |        ScriptDefinition definition = scriptDefinitionProvider.findDefinition(script);
13 |        if (definition == null) {
14 |            return scriptDefinitionProvider.getDefaultDefinition(); // Comment
15 |        } else {
16 |            return definition;
17 |        }
18 |    }
19 | }

```

Листинг 2 — Пример какого-то кода на Java

```

...
13 aload_2
14 dup
15 ifnonnull 28
18 new #17 // NullPointerException
21 dup
22 ldc #19 // String null cannot be cast to non-null String
24 invokespecial #23 // NullPointerException."<init>"(String)
27 athrow
...
46 aload_2
47 dup
48 ifnonnull 61
51 new #17 // NullPointerException
54 dup
55 ldc #19 // String null cannot be cast to non-null String
57 invokespecial #23 // NullPointerException."<init>"(String)
60 athrow
...

```

Листинг 3 — Пример JVM-байткода

```

...
13: aload_2
14: dup
15: ifnonnull 28
18: new #17 // NullPointerException
21: dup
22: ldc #19 // String null cannot be cast to non-null String
24: invokespecial #23 // NullPointerException."<init>"(String)
27: athrow
...
46: aload_2
47: dup
48: ifnonnull 61
51: new #17 // NullPointerException
54: dup
55: ldc #19 // String null cannot be cast to non-null String
57: invokespecial #23 // NullPointerException."<init>"(String)
60: athrow
...

```

Листинг 4 — Пример JVM-байткода 2

А тут ссылка на таблицу 1.

Таблица 1 — Пример таблицы

Col1	Col2	Col2	Col3
1	6	87837	787
2	7	78	5415
3	545	778	7507
4	545	18744	7560
5	88	788	6344

```
1 procedure RUN(packages, hashes)
2   queue[svace.parallel_max]
3   for item ∈ zip(packages, hashes)
4     ps = create(item)
5     if !queue.full()
6       queue.put(ps)
7     else
8       first = queue.get()
9       first.wait()
10    end if
11  end for
12 end procedure
```

Листинг 5 — Пример псевдокода на алгоритмическом языке

## Ещё один пример приложения

Пример приложения