Храпов А.А. Использование ANTLR для анализа метрик холстеда в языках PYTHON и BML // Международный журнал информационных технологий и энергоэффективности. – 2025. – Т. 10 № 1(51) с. 36–41

Международный журнал информационных технологий и энергоэффективности

Сайт журнала: http://www.openaccessscience.ru/index.php/ijcse/

УДК 004.42

ИСПОЛЬЗОВАНИЕ ANTLR ДЛЯ АНАЛИЗА МЕТРИК ХОЛСТЕДА В ЯЗЫКАХ PYTHON И BML

Храпов А.А.
ФГБОУ ВО "МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ (НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)", Москва, Россия, (125993, Москва, Волоколамское ш., д. 4), e-mail: khrapov@bk.ru

В данной статье рассматривается использование ANTLR (Another Tool for Language Recognition) для анализа исходного кода, написанного на языках Python и BML (BlockSet Modeling Language), с целью автоматизированного подсчета метрик Холстеда. Описан процесс создания грамматик для языков Python и BML и использования ANTLR для генерации парсеров, которые позволяют построить синтаксическое дерево. На основании этого дерева производится автоматизированный подсчет операторов и операндов, необходимых для вычисления ключевых метрик. В статье приводятся примеры реализации на языке Python, а также выделено преимущество автоматизации подсчета метрик в сравнении с ручным подходом. Применение ANTLR обеспечивает точность и стандартизацию анализа исходного кода программы, что особенно важно для оценки эффективности различных инструментов разработки веб-приложений. Проведенное исследование демонстрирует, как можно использовать синтаксический анализатор для объективного сравнения языков Python и BML, измерения сложности и объема кода.

Ключевые слова: ANTLR, метрики Холстеда, синтаксический анализ, Python, BML, Django.

USING ANTLR TO ANALYZE HALSTED METRICS IN PYTHON AND BML LANGUAGES

Khrapov A.A.
MOSCOW AVIATION INSTITUTE (NATIONAL RESEARCH UNIVERSITY), Moscow, Russia, (125993, Moscow, Volokolamskoye shosse, 4), e-mail: khrapov@bk.ru

This article provides an overview of the most popular open source relational database management systems (DBMS), such as MySQL, PostgreSQL, MariaDB and SQLite. It covers the key aspects that distinguish these DBMS, including architecture, installation complexity, extensibility, performance monitoring, and ease of learning. The benchmarking also includes performance, compatibility, data backup and recovery capabilities, as well as support levels for ACID transactions and data integrity.

Keywords: ANTLR, Halstead metrics, parsing, Python, BML, Django.

Разработка веб-приложений требует не только создания функциональности, но и анализа эффективности используемых инструментов. Метрики Холстеда позволяют измерить сложность кода, его объем, временную нагрузку и другие характеристики [4]. Однако ручной подсчет метрик затруднителен, особенно для крупных проектов. Для автоматизации этого процесса можно использовать ANTLR (Another Tool for Language Recognition).

ANTLR – это библиотека, являющаяся мощным инструментом для описания грамматик языков и генерации кода для их синтаксического анализа [1]. В данной работе

рассматривается, как ANTLR может быть применен для анализа языков Python и BML (BlockSet), а также для автоматизированного подсчета метрик Холстеда.

Начнём с построения грамматики для высокоуровневого языка программирования Python [2]. Она включает лексические и синтаксические правила для идентификаторов, операторов и выражений. Одна из вариаций грамматики может выглядеть следующим образом:

```
grammar Python;
program: statement+;
statement: assignment | expression;
assignment: IDENTIFIER '=' expression;
expression: NUMBER | IDENTIFIER | '(' expression ')';
IDENTIFIER: [a-zA-Z_][a-zA-Z0-9_]*;
NUMBER: [0-9]+;
WHITESPACE: [ \t\r\n]+ -> skip;
```

Данная грамматика описывает простые операторы присваивания и некоторые выражения, что достаточно для демонстрации подсчета базовых метрик. В дальнейшем в грамматику будут включены более сложные конструкции языка, такие как условные операторы, циклы и вызовы функций.

Далее перейдём к построению грамматики для BML, который используется в BlockSet для декларативного описания данных и логики [3]. Она может быть описана так:

```
grammar BML;
model: '<model>' set+ '</model>';
set: '<set' 'name' '=' "" IDENTIFIER "" '>' block* '</set>';
block: '<block' 'name' '=' "" IDENTIFIER "" 'type' '=' "" IDENTIFIER "" '/>';
IDENTIFIER: [a-zA-Z_][a-zA-Z0-9_]*;
WHITESPACE: [ \t\r\n]+ -> skip;
```

Эта грамматика позволяет анализировать иерархическую структуру BML и извлекать информацию для подсчета метрик. В отличие от Python, структура BML более предсказуема и декларативна, что облегчает построение грамматики.

Для генерации парсеров ANTLR используются команды «antlr4 -Dlanguage=Python3 Python.g4» и «antlr4 -Dlanguage=Python3 BML.g4» [1].

Это создаст Python-классы для обработки синтаксического дерева, включая классы парсера и посетителя (Visitor). Сгенерированные парсеры способны преобразовывать исходный код в синтаксическое дерево, которое затем можно обрабатывать для извлечения операторов и операндов.

Сгенерированный код используется для обхода дерева синтаксиса. Его необходимо расширить логикой подсчета операторов и операндов (рисунок 1).

```python
from PythonParser import PythonParser
from PythonVisitor import PythonVisitor

class HalsteadMetricsVisitor(PythonVisitor):
    def __init__(self):
        self.operators = set()
        self.operands = set()
        self.operator_count = 0
        self.operand_count = 0

    def visitAssignment(self, ctx):
        self.operators.add('=')
        self.operator_count += 1
        self.visit(ctx.expression())
        return None

    def visitExpression(self, ctx):
        if ctx.NUMBER():
            self.operands.add(ctx.NUMBER().getText())
            self.operand_count += 1
        elif ctx.IDENTIFIER():
            self.operands.add(ctx.IDENTIFIER().getText())
            self.operand_count += 1
        return self.visitChildren(ctx)
```

Рисунок 1 – Расширения класса для подсчёта операторов и операндов Python кода

Аналогичный класс можно создать и для BML (рисунок 2).

```python
from BMLParser import BMLParser
from BMLVisitor import BMLVisitor

class HalsteadMetricsBMLVisitor(BMLVisitor):
    def __init__(self):
        self.operators = set()
        self.operands = set()
        self.operator_count = 0
        self.operand_count = 0

    def visitModel(self, ctx):
        self.operators.add('<model>')
        self.operators.add('</model>')
        self.operator_count += 2
        return self.visitChildren(ctx)

    def visitSet(self, ctx):
        self.operators.add('<set>')
        self.operators.add('</set>')
        self.operator_count += 2

        name = ctx.getChild(3).getText()
        self.operands.add(name)
        self.operand_count += 1
        return self.visitChildren(ctx)

    def visitBlock(self, ctx):
        self.operators.add('<block>')
        self.operator_count += 1

        name = ctx.getChild(3).getText()
        block_type = ctx.getChild(7).getText()
        self.operands.add(name)
        self.operands.add(block_type)
        self.operand_count += 2

        return self.visitChildren(ctx)
```

Рисунок 2 – Расширения класса для подсчёта операторов и операндов BML кода

В итоге, применение ANTLR позволит автоматически подсчитать метрики Холстеда, такие как:

- n_1 (уникальные операторы);
- n_2 (уникальные операнды);
- N_1 (общее количество операторов);
- N_2 (общее количество операндов);
- n (словарь программы);
- N (длина кода);
- Объем программы (V) и сложность (D) [4].

Для корректного подсчёта для языка Python выбран фреймворк Django – это высокоуровневый веб-фреймворк, который позволяет быстро создавать безопасные и поддерживаемые веб-приложения [5].

Например, для фрагмента Python-кода на Django (Рисунок 3) результатом подсчёта будут: $n_1 = 7$ («def», «=», «.», «()», «{}», «:»); $n_2 = 3$ (request, random_game, Game и т.д.); $N_1 = 20$; $N_2 = 30$; $n = 23$, $N = 50$, $V \approx 226$; $D \approx 6{,}56$.

```python
def home(request):
    random_game = Game.objects.order_by('?').first()
    top_games = Game.objects.order_by('-user_rating')[:10]
    recent_games = Game.objects.order_by('-id')[:10]
    return render(request, 'games/home.html', {
        'random_game': random_game,
        'top_games': top_games,
        'recent_games': recent_games,
    })
```

Рисунок 3 – фрагмент кода программы на Django

Для BML фрагмента (Рисунок 4) подсчитаны свои показатели: $n_1 = 5$ («model», «set», «block», «/set», «/model»); $n_2 = 6$ (games, top_game и т.д.); $N_1 = 7$; $N_2 = 6$; $n = 11$, $N = 13$, $V \approx 45$; $D \approx 2{,}5$.

```
<model>
    <set name="games">
        <block name="random_game" type="object" />
        <block name="top_games" type="list" />
        <block name="recent_games" type="list" />
    </set>
</model>
```

Рисунок 4 – фрагмент кода BML

Данные примеры ещё не являются полноценны веб-приложением, это лишь фрагменты, но уже можно сделать некоторые выводы:

- У BML декларативная структура, следовательно, меньший словарный запас языка;
- Вероятнее, более низкие объём и сложность кода по сравнению с Django (Python).

В заключение, можно отметить, что применение ANTLR позволит автоматизировать подсчет метрик Холстеда для различных языков, в данном случае для Python и BML. Это снизит нагрузку на разработчиков и обеспечит более высокую точность при анализе кода.

Результаты подсчета демонстрируют, как можно организовать процесс анализа кода и объективно оценить эффективность разработки с использованием разных инструментов.

Список литературы

1. Документация ANTLR. [Электронный ресурс] URL: https://www.antlr.org/. (дата обращения 09.12.2024).
2. Документация Python. [Электронный ресурс] URL: https://docs.python.org (дата обращения 10.12.2024).
3. Предпосылки формирования новой методологии разработки веб-узлов BlockSet и декларативного языка BML. Кейно П.П. [Электронный ресурс] URL: https://cyberleninka.ru/article/n/predposylki-formirovaniya-novoy-metodologii-razrabotki-veb-uzlov-blockset-i-deklarativnogo-yazyka-bml/viewer (дата обращения 10.12.2024).
4. Программные показатели Холстеда – Разработка программного обеспечения. [Электронный ресурс] URL: https://www.geeksforgeeks.org/software-engineering-halsteads-software-metrics/ (дата обращения 12.12.2024).

5. Django веб-фреймворк (Python). [Электронный ресурс] URL: https://developer.mozilla.org/ru/docs/Learn/Server-side/Django/Introduction (дата обращения 10.12.2024).

References

1. Documentation of ANTLR. [Electronic resource] URL: https://www.antlr.org/ (accessed 09.12.2024).
2. Documentation of Python. [Electronic resource] URL: https://docs.python.org (accessed 10.12.2024).
3. Prerequisites for the formation of a new development methodology BlockSet and declarative BML language. P.P.Keyno. [Electronic resource] URL: https://cyberleninka.ru/article/n/predposylki-formirovaniya-novoy-metodologii-razrabotki-veb-uzlov-blockset-i-deklarativnogo-yazyka-bml/viewer (accessed 12.12.2024).
4. Halstead's Software Metrics – Software Engineering. [Electronic resource] URL: https://www.geeksforgeeks.org/software-engineering-halsteads-software-metrics/ (accessed 10.12.2024).
5. Django web-framework (Python). [Electronic resource] URL: https://developer.mozilla.org/ru/docs/Learn/Server-side/Django/Introduction (accessed 10.12.2024).