Санкт-Петербургский Государственный Университет
Кафедра системного программирования

Разработка алгоритмов по оценке качества кода для онлайн-платформ Stepik и Hyperskill

Автор: И.М. Власов, 19.Б10-мм
Научный руководитель: к.т.н., доцент Т.А. Брыксин
Консультант: программист-исследователь JetBrains Research А.И. Бирилло

Санкт-Петербург, 2021

Введение

- ➔ В настоящий момент наблюдается процесс активной цифровизации образования [1, 2]
- ➔ Всё больше людей получают доступ к онлайн-курсам по программированию
- ➔ Студенты фокусируются на прохождение тестов, уделяя мало внимания качеству кода [3, 4]
- ➔ Качественный код напрямую влияет на читаемость, сложности и поддержку

[1] He Wu, Xu Guandong, Kruck SE. Online IS education for the 21st century, 2019
[2] Schmidt Joel T, Tang Min. Digitalization in Education: Challenges, Trends and Transformative Potential, 2020
[3] Keuning Hieke, Heeren Bastiaan, Jeuring Johan. Code Quality Issues in Student Programs, 2017
[4] Keuning Hieke, Heeren Bastiaan, Jeuring Johan. How Teachers Would Help Students to Improve Their Code, 2019

Обзор

- Существующие инструменты:
  - Qodana[1]
  - Codacy[2]
  - SonarQube[3]
- Плюсы:
  - Позволяют проводить комплексную проверку кода
- Минусы:
  - Ориентированы на промышленные продукты, а не на образование
  - Не учитывают ранее совершённые ошибки

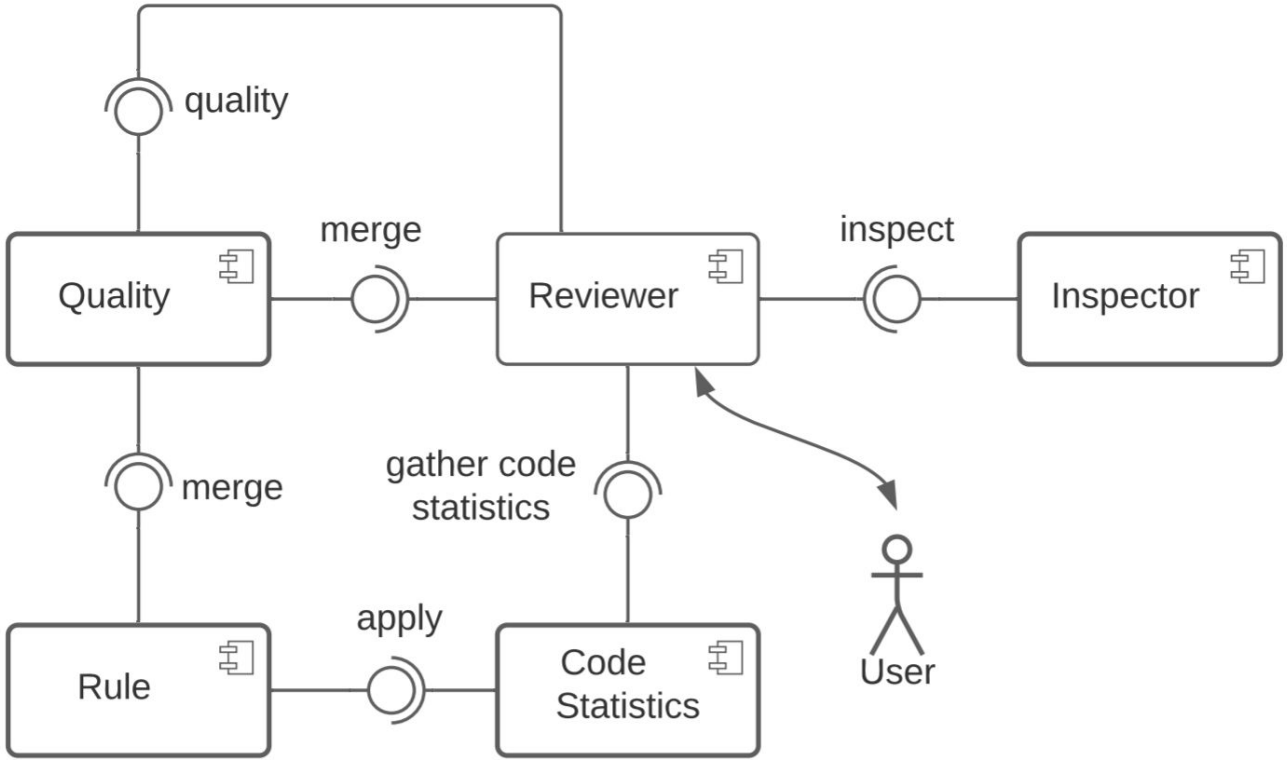
1. Официальный репозиторий Qodana: https://github.com/JetBrains/Qodana
2. Официальный сайт Codacy: https://www.codacy.com/
3. Официальный сайт SonarQube: https://www.sonarqube.org/

Hyperstyle

- ➔ Инструмент для запуска набора предварительно настроенных анализаторов кода и оценки качества кода
- ➔ Используется на платформах для онлайн-обучения программированию Stepik и Hyperskill
- ➔ Поддерживаемые языки: Python, Java, Kotlin и JavaScript

- ➔ Минусы:
  - Системы новая и разрабатывалась быстро, поэтому есть проблемы с качеством исходного кода инструмента
  - В основном умеет находит ошибки, связанные со стилем кода
  - Не учитывает ранее совершённые ошибки

Общая схема работы Hyperstyle



Результат работы Hyperstyle

- Одна из следующих оценок: EXCELLENT, GOOD, MODERATE или BAD
- Список все ошибок в формате JSON:

```
{
  "code": "WPS331",
  "text": "Found variables that are only used for return",
  "line": "return result",
  "line_number": 3,
  "column_number": 5,
  "category": "BEST_PRACTICES"
}
```

Цель и задачи

Цель: улучшение системы оценки качества кода Hyperstyle для онлайн-платформ Stepik и Hyperskill

Задачи:

1. Провести обзор исследуемой области
2. Усовершенствовать процесс непрерывной интеграции системы оценки качества кода
3. Добавить поддержку различных анализаторов кода для языка Python
4. Разработать и внедрить алгоритм штрафов за повторные ошибки, совершенные студентом
5. Оценить качество решения на реальных данных

Улучшение CI

Ошибка	Описание	Поддержка
I100	Неверный порядок импортов	✓
I101	Неверный порядок имён внутри импорта	✓
I202	Неверный перенос строки между импортами	✓
I201	Лишняя строка внутри группы импортов	✓
R504	Присвоение переменной перед `return`	✗
A003	Затенение встроенных переменных	✗
E800	Закомментированный код	✗
SC100	Орфографическая ошибка в комментариях	✓
SC200	Орфографическая ошибка в имени	✓
E402	Импорт не в начале файла	✗

Таблица 1: Поддержка игнорируемых инспекций

Новые анализаторы

- Новые плагины flake8:
  - wemake-python-styleguide (WPS)
  - flake8-broken-line
  - flake8-string-format
  - flake8-commas
  - cohesion
- Новый анализатор: Radon
  - Вычисляет maintainability index[1]

1. Формула, используемая для вычисления индекса ремонтопригодности: https://radon.readthedocs.io/en/latest/intro.html\#maintainability-index

Алгоритм штрафов

```
1  def get_penalty_coefficient(current, previous):
2      penalizing_issues = get_penalizing_issues(current, previous)
3
4      nonnormalized_penalty_coefficient = 0
5      for issue in penalizing_issues:
6          nonnormalized_penalty_coefficient += issue.number *
           ↪ get_category_coefficient(issue.category)
7
8      penalty_coefficient = nonnormalized_penalty_coefficient / (
       ↪ nonnormalized_penalty_coefficient + len(previous))
9
10     return penalty_coefficient
```

Алгоритм штрафов

Итоговая оценка \ Исходная оценка	EXCELLENT	GOOD	MODERATE	BAD
EXCELLENT	$[0, 0.5)$	—	—	—
GOOD	$[0.5, 0.7)$	$[0, 0.5)$	—	—
MODERATE	$[0.7, 0.9)$	$[0.5, 0.7)$	$[0, 0.5)$	—
BAD	$[0.9, 1]$	$[0.7, 1]$	$[0.5, 1]$	$[0, 1]$

Таблица 2: Влияние штрафного коэффициента на итоговую оценку

Тестирование

- ➔ Тестирование производилось на датасете из 163457 решений реальных пользователей
- ➔ Из них 68859 уникальные фрагменты кода
- ➔ Всего 4671 уникальный пользователь.
- ➔ Отдельно протестированы новые анализаторы и алгоритм штрафов

Тестирование

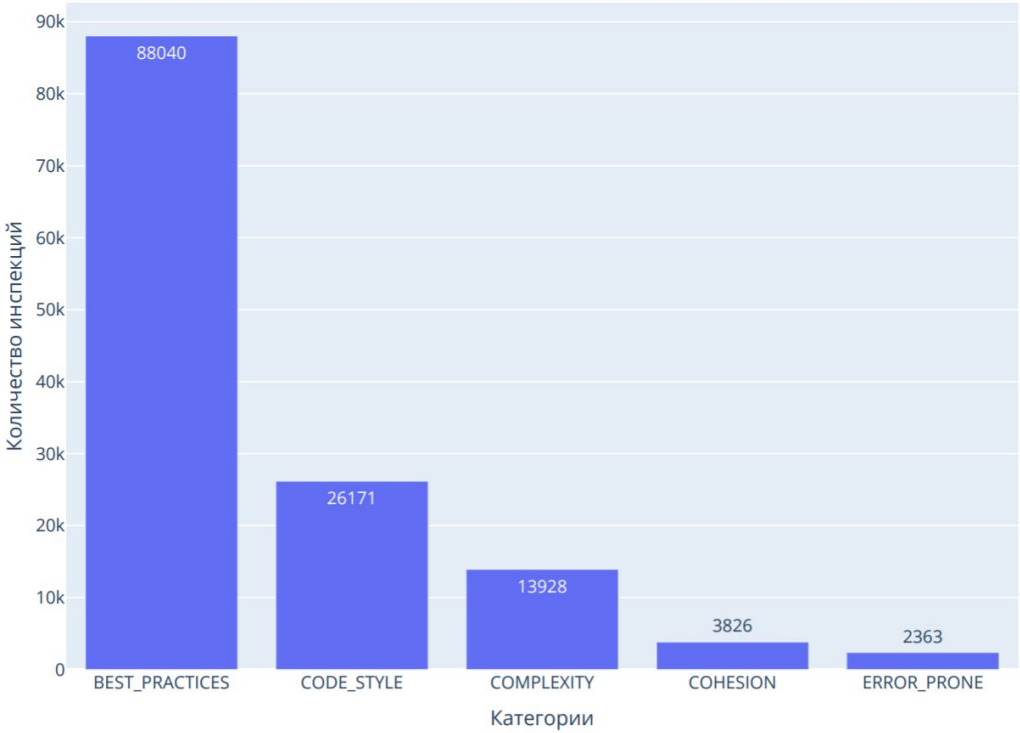


Рис. 1: Количество найденных новых инспекций, сгруппированных по категориям

Тестирование

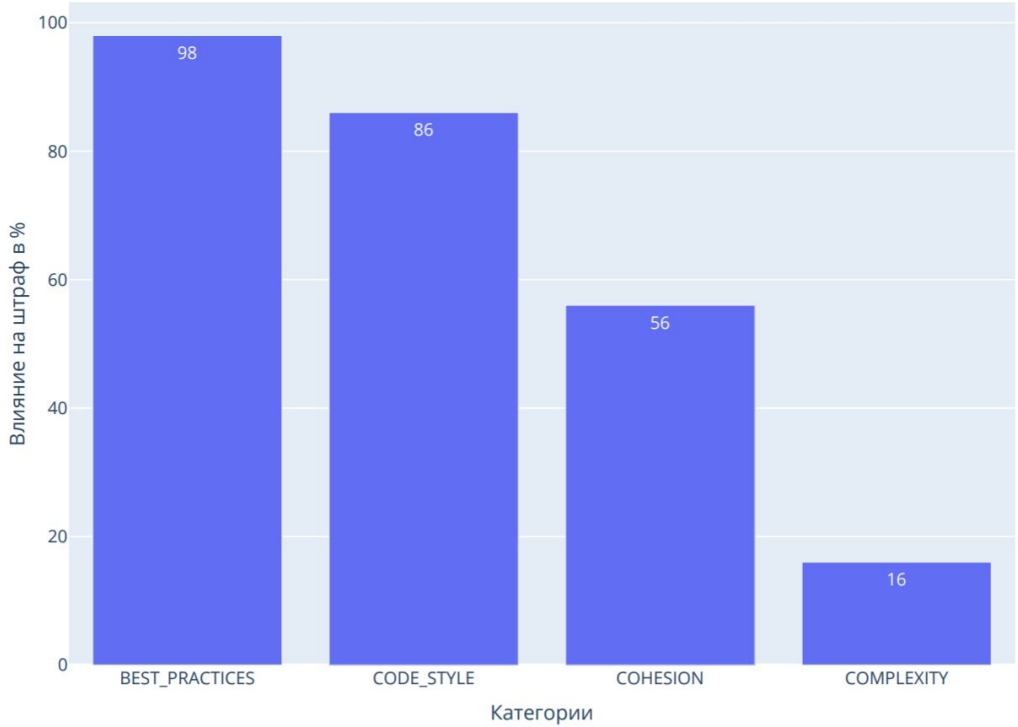


Рис. 2: Влияние алгоритма штрафов на итоговую оценку по каждой категории. Представлены медианные значения

Заключение

- ➔ Проведён обзор исследуемой области
- ➔ Усовершенствован процесс непрерывной интеграции системы оценки качества кода
- ➔ Добавлена поддержку различных анализаторов кода для языка Python
- ➔ Разработан и внедрён алгоритм штрафов за повторные ошибки, совершенные студентом
- ➔ Оценено качество решения на реальных данных

Исходный код инструмента Hyperstyle можно изучить в официальном репозитории[1] на GitHub.

1. Ссылка на репозиторий: https://github.com/hyperskill/hyperstyle

Что дальше?

- ➔ Добавить новые анализаторы кода для Java, JavaScript и Kotlin;
- ➔ Добавить персонализацию
- ➔ Добавить алгоритм выявления наиболее сложных для исправления инспекций, основанный на теории тестовых заданий (Item Response Theory, IRT) [1]
- ➔ Разработать модель машинного обучения для имитации работы статических анализаторов, таких как Qodana.

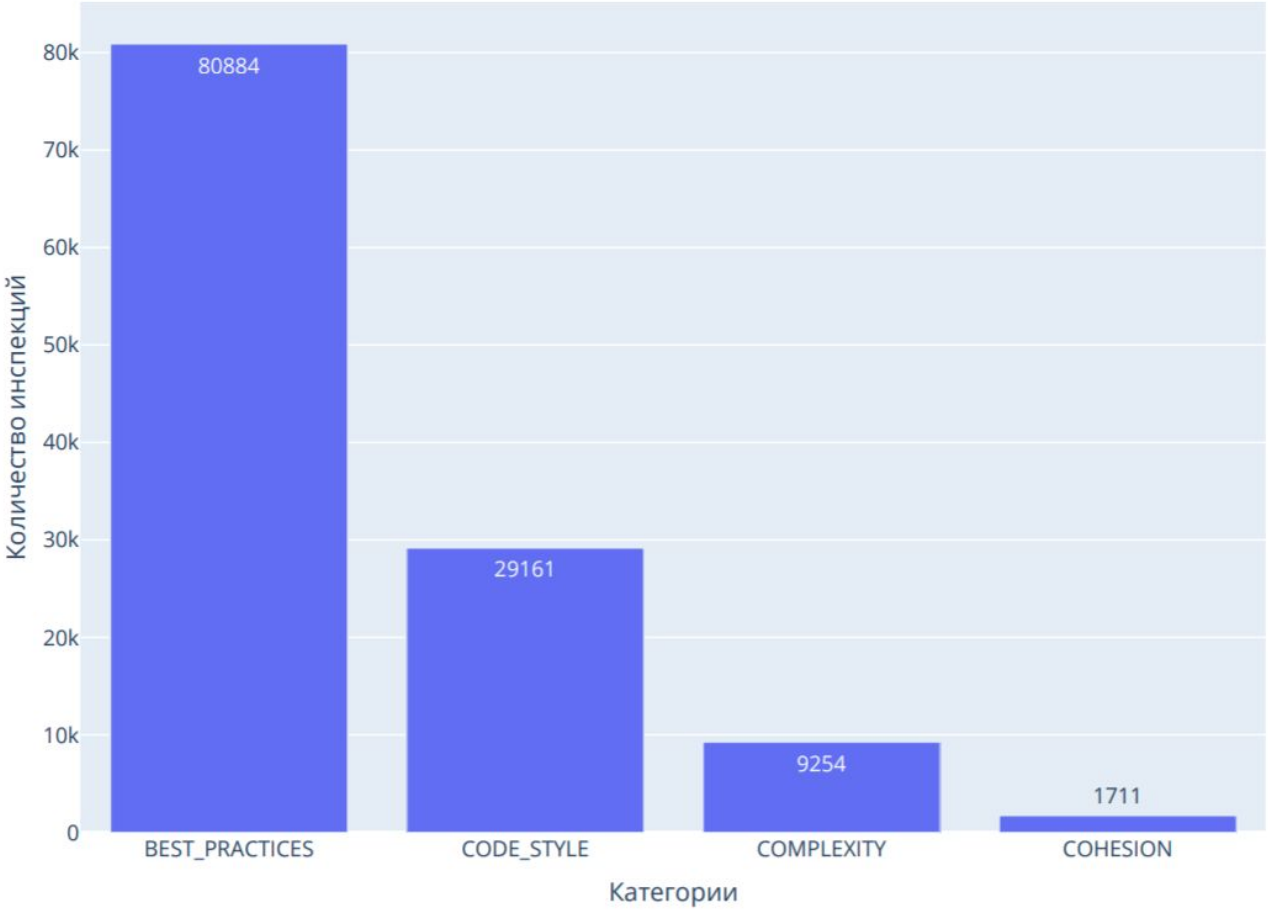
[1] Berges Marc, Hubwieser Peter. Evaluation of Source Code with Item Response Theory, 2015

```
{
    ''quality'':{
        ''code'':''GOOD'',
        ''text'':''Code quality (beta): GOOD''
    },
    ''file_review_results'':[
        {
            ''file_name'':''main.py'',
            ''quality'':{
                ''code'':''GOOD'',
                ''text'':''Code quality (beta): GOOD''
            },
            ''issues'':[
                {
                    ''code'':''WPS331'',
                    ''text'':''Found variables that are only used for
                        ↪ return'',
                    ''line'':''return result'',
                    ''line_number'':3,
                    ''column_number'':5,
                    ''category'':''BEST_PRACTICES''
                }
            ]
        }
    ]
}
```

```
1  {
2      ''python'':[
3          {
4              ''origin_class'':''SC200'',
5              ''number'':20
6          },
7          {
8              ''origin_class'':''WPS314'',
9              ''number'':3
10         }
11     ]
12 }
```

Категория	PR	DIF	IMP	Коэффициент
CODE_STYLE	5	4	5	1.0
LINE_LEN	4	5	5	1.0
BEST_PRACTICES	5	3	5	0.9
CYCLOMATIC_COMPLEXITY	4	3	3	0.7
ERROR_PRONE	3	4	2	0.6
COUPLING	4	1	4	0.6
COHESION	4	1	4	0.6
FUNC_LEN	3	3	2	0.6
BOOL_EXPR_LEN	3	2	3	0.6
COMPLEXITY	2	2	3	0.5
MAINTAINABILITY	1	1	2	0.3
METHOD_NUMBER	1	1	1	0.2
CHILDREN_NUMBER	1	1	1	0.2
WEIGHTED_METHOD	1	1	1	0.2
CLASS_RESPONSE	1	1	1	0.2
INHERITANCE_DEPTH	1	1	1	0.2

Ошибка	Описание	Категория	Количество
SC200	Орфографическая ошибка в имени	BEST_PRACTICES	64 435
WPS432	Магическое число	CODE_STYLE	17 477
WPS221	Излишне сложная строчка	COMPLEXITY	10 618
WPS336	Явная конкатенация строк вместо форматирования	BEST_PRACTICES	4965
H601	Низкая связность	COHESION	3826
SC100	Орфографическая ошибка в комментарии	BEST_PRACTICES	2719
WPS319	Неправильно расставленные скобки	CODE_STYLE	2655
WPS317	Неправильные отступы	CODE_STYLE	2575
WPS515	`open()` без менеджера контекста	BEST_PRACTICES	1783
WPS503	Ненужная `else`-ветка	BEST_PRACTICES	1611


80k
70k
60k
50k
40k
30k
20k
10k
0
Количество инспекций
80884
29161
9254
1711
BEST_PRACTICES
CODE_STYLE
COMPLEXITY
COHESION
Категории

Ошибка	Описание	Категория	Количество
SC200	Орфографическая ошибка в имени	BEST_PRACTICES	59 467
WPS432	Магическое число	CODE_STYLE	13 169
WPS221	Излишне сложная строчка	COMPLEXITY	8212
SC100	Орфографическая ошибка в комментарии	BEST_PRACTICES	5288
E226	Отсутствует пробел около оператора	CODE_STYLE	3537
WPS336	Явная конкатенация строк вместо форматирования	BEST_PRACTICES	3034
WPS515	`open()` без менеджера контекста	BEST_PRACTICES	2468
E231	Отсутствующий пробел после ‘`,`’, ‘`;`’, ‘`:`’	CODE_STYLE	2183
H601	Низкая связность	COHESION	1711
E225	Отсутствует пробел около оператора	CODE_STYLE	1532