Инженерный вестник Дона, №10 (2024)
ivdon.ru/ru/magazine/archive/n10y2024/9538

Один из подходов проведения анализа программного кода студенческих работ

Е.А. Павлова, М.В. Аврискин, А.В. Мельникова, М.С. Воробьева,
Тюменский государственный университет, Тюмень

Аннотация: При оценке работ студентов особенно актуальным становится анализ текстовых работ, в частности анализ программного кода. В статье рассмотрен подход для оценки динамики изменения признаков программного кода студентов. Проанализированы различные метрики программного кода и выделены ключевые: количественные, метрики сложности потока управления программы, индикатор качества TIOBE. Для определения порогового значения по каждой из метрик и их категоризации использован набор текстовых данных с исходными кодами программ, размещённых на сайте, посвящённому практическому программированию. Полученные результаты были использованы для проведения анализа программных кодов студентов с помощью разработанного сервиса, позволяющего оценить работы по ключевым признакам, увидеть динамику изменения показателей программного кода, понять положение студента в группе с точки зрения полученных значений.
Ключевые слова: машинное обучение, анализ текстовых данных, анализ программного кода, цифровой след, визуализация данных.

Введение. Обработка текстовых данных приобретает всё большую популярность в разных сферах деятельности. Для анализа текстов на естественном языке используются синтаксические анализаторы, имеющие различные реализации [1], в том числе, адаптированные для разбора исходного кода на языке программирования высокого уровня [2]. В рамках курсов по программированию студенты ИТ направлений выполняют лабораторные работы, практические задания, контрольные работы, индивидуальные задания, связанные с написанием программного кода и разработкой программ разного уровня сложности. Исходный код студентов может быть проанализирован с разных сторон, например, с точки зрения нахождение похожих фрагментов кода как по структуре, так и по количественным признакам. Преподавателям необходимо регулярно проверять большое число работ студентов, оценивая не только правильность выполнения программ, но и динамику студентов в выработке стиля написания качественного программного кода. С другой стороны, студентам

может быть интересно оценить развитие навыков программирования, сравнить себя с одногруппниками.

Цель работы – разработка сервиса, позволяющего на основании программных кодов студентов получить динамику развития программистских навыков студентов.

Для достижения этой цели необходимо решить следующие задачи:

- исследовать подходы к оценке качества программного кода;
- определить набор ключевые признаков кода, отражающих его качество;
- провести анализ ключевых признаков для определения пороговых значений и их категоризации;
- разработать инструменты для разного вида пользователей (преподаватель, студент).

Материалы и методы исследования. На первом этапе исследования были выделены три группы ключевых метрик, характеризующих программный код [3]: количественные метрики, метрики сложности потока управления программы, индикатор качества TIOBE.

К количественным метрикам относятся базовые характеристики исходного кода, связанные с количеством рассматриваемых элементов. В исследовании были рассчитаны следующие метрики:

- ✓ *количество строк кода;*
- ✓ *доля строк с комментариями* — отношение количества строк, содержащих комментарии к общему числу строк (в процентах);
- ✓ *доля пустых строк* — отношение количества строк без символов к общему числу строк (в процентах);
- ✓ *средняя длина строки* — усредненное значение от количества символов в строке;

✓ *средняя длина идентификатора* — усредненное количество символов для описания переменных;

✓ *средняя длина блоков кода с учетом вложенности* — отношение произведения максимальной вложенности блоков во всех методах и количества символов в блоках всех методов к количеству методов;

✓ *метрики Холстеда* [4].

Основной *метрикой сложности потока управления программы* является *цикломатическая сложность программы* (или цикломатическое число Мак–Кейба) [5]. Несмотря на наличие возможных модификаций классического метода цикломатической сложности программы, в большинстве существующих инструментов используется именно цикломатическое число Мак–Кейба ввиду простоты и универсальности данной метрики.

Индикатор качества TIOBE [6] – это набор метрик, используемых компанией *TIOBE* для получения комбинированной взвешенной оценки на базе измеряемых характеристик кода, зафиксированных стандартом ISO 25010 [7].

После выявления признаков важным этапом анализа программного кода является токенизация исходного текста (определение ключевых слов для каждого из типов анализа программного кода) для дальнейшего расчета метрик на основании формул, связанных с количеством ключевых слов и их типов/видов в зависимости от типа текстового токена [8]. Например, для оценки цикломатической сложности программного кода используются токены операторов языка программирования (ветвления, цикла, прерывания). Для оценки расчетных метрик используются токены лексем (идентификаторов переменных и методов, разделители, комментарии и т.д.). Реализация токенизации текста и разбиение входной последовательности

символов на лексемы с последующей классификацией и унификации была выполнена с использованием библиотеки python-pygments [9].

Для определения порогового значения для каждой из метрик и их категоризации был использован набор текстовых данных с исходными кодами решений практических задач по программированию, размещённых на сайте HackerRank. Датасет включал программные коды на языке C# по 22 разнообразным заданиям. Всего 329927 экземпляров программных кодов, прошедшие юнит–тестирование в рамках поставленной для них задачи [10].

Общая схема проведения анализа программного кода включает три этапа: получение, обработка и анализ данных (рис. 1).

Рис. 1. – Общая схема проведения анализа программного кода

Для каждого вида задания из выбранного набора данных после обработки (очистки и приведения к общему формату) было проведено измерение метрик программного кода, на основании которых были рассчитаны минимум, максимум и медианное значение, а также частотные показатели на основании разделения выборки по значениям метрик с заданным шагом дискретизации. В таблице 1 показан расчет метрик на примере одной из задач.

Таблица 1

Статистические характеристики метрик

Метрика	минимум	максимум	медиана
Цикломатическая сложность	2	18	5
Объем по Холстеду	54	1132	235.43
Сложность по Холстеду	0.95	65	13.34
Индекс поддерживаемости, %	23.46	100	82.958

После расчётов для всех метрик по каждому заданию были взяты средние значения частотных диапазонов и разделены на 5 категорий от худшего показателя к лучшему. Результаты распределения метрик по категориям указаны в таблице 2.

Таблица 2

Распределение метрик по категориям

Метрики	Категории				
	A	B	C	D	E
Цикломатическая сложность	<5	5–10	10–15	15–20	>20
Сложность по Холстеду	0–25	25–45	45–60	60–80	>80
Индекс поддерживаемости кода, %	80–100	60–80	45–60	30–45	0–30
Комментарии в коде, %	>15	10–15	5–10	3–5	0–3

Результаты и их обсуждение. Для проведения анализа программных кодов студентов был разработан сервис, с помощью которого можно оценить работы по ключевым признакам, увидеть динамику изменения показателей программного кода, понять положение студента в группе с точки зрения полученных значений (рис. 2).

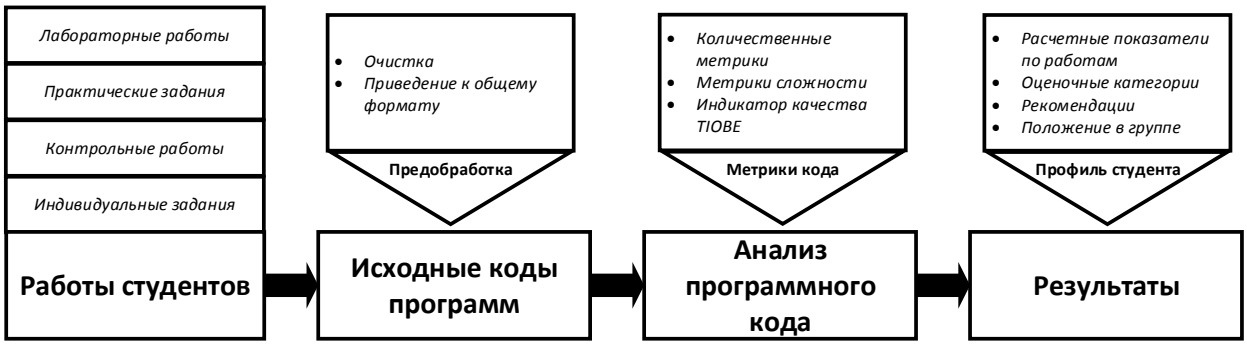

Рис. 2. – Схема работы сервиса для анализа программных кодов

В качестве исходного набора данных был использован архив с набором программных кодов на языке C++ 349 студентов 1 курса, поступивших в Институт математики и компьютерных наук в 2019-2020 уч. году, 2020–2021

уч. году, 2021-2022 уч. году, 2022-2023 уч. году на направления «Компьютерная безопасность» и «Математическое обеспечение и администрирование информационных систем».

Исходные данные были получены из аналитической системы с иерархической структурой хранилища, которое содержит как сырые, так и подготовленные для дальнейшего анализа данные (пользовательский уровень). Доступ к данным пользовательского уровня организован с помощью прикладного уровня интерфейса [11]. Архив содержал 3468 файлов, каждый студент выполнил от 5 до 10 работ (таблица 3).

Таблица 3

Распределение студентов и файлов по учебным годам

Уч. годы		Количество сданных лабораторных работ						Итого
		5	6	7	8	9	10	
2019-2020 уч.год	студенты	23	22	20	18	12	11	106
	файлы	115	132	140	144	108	110	749
2020-2021 уч.год	студенты	22	20	23	21	15	12	113
	файлы	110	120	161	168	135	120	814
2021-2022 уч.год	студенты	28	22	24	19	19	18	130
	файлы	140	132	168	152	171	180	943
2022-2023 уч.год	студенты	30	26	22	20	18	18	134
	файлы	150	156	154	160	180	180	962

По полученным обезличенным экземплярам программного кода студентов после предобработки были рассчитаны показатели: количественные метрики и цикломатическая сложность (рис. 3).

№ Признак	*Лабораторные работы*										
	№1	*№2*	*№3*	*№4*	*№5*	*№6*	*№7*	*№8*	*№9*	*№10*	*Среднее*
1 Число строк кода (LOC)	99	51	36	177	184	138	100	51	38	250	**111.4**
2 Доля комментариев в коде (CR)	0%	9.88%	4.64%	1.72%	0%	2.61%	0%	9.87%	4.61%	0%	**3.33%**
3 Цикломатическая сложность (CC)	0	10	0	8	21	9	0	10	0	17	**7.5**
4 Сложность по Холстеду (HC)	32.42	41.67	25.82	56.25	90.62	73.95	32.42	41.67	25.82	79.02	**49.97**
5 Объем по Холстеду (HV)	4142.81	1795.84	897.4	7154.33	6499.32	3965.08	4149.26	1801.33	902.64	8200.11	**3950.81**
6 Индекс поддерживаемости кода (MI)	**53.25%**	**66.04%**	**77.59%**	**39.15%**	**36.03%**	**46.03%**	**53.08%**	**66.02%**	**76.68%**	**31.44%**	**54.53%**

Рис. 3. – Метрики программных кодов студента

Затем, на основе выделенных ранее пороговых значений для каждой метрики, программные коды были разделены на категории с указанием рекомендаций по каждой работе студента.

Для любой работы студента можно проследить динамику изменения по каждой метрике. Например, по признаку «Индекс поддерживаемости кода» у выбранного студента наблюдается качественный рост, так как даже при усложнении заданий (4 и 10 работы) индекс после падания начинает расти (рис. 4). При этом можно увидеть, что этот показатель является одним из лучших в группе, и ещё у двух студентов схожие значения по данной метрике.

Рис. 4. – Динамика работ студента по метрике «Индекс поддерживаемости кода»

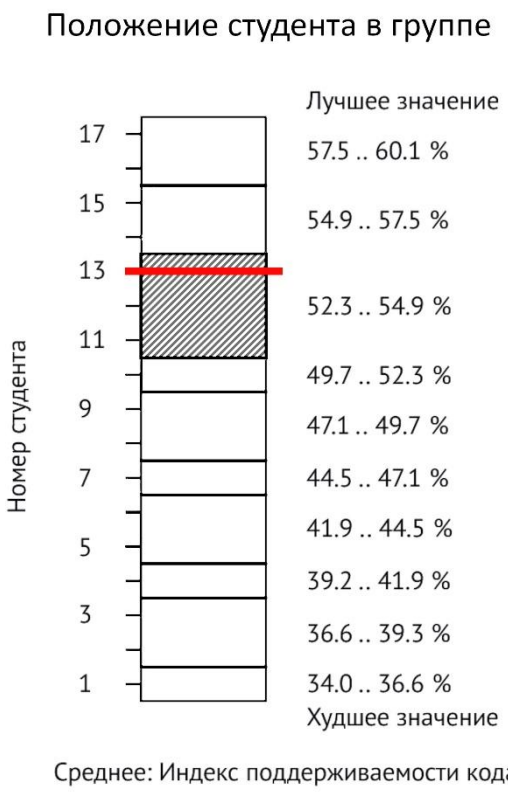

Рис. 5. – Сравнение студента с группой по метрике «Индекс поддерживаемости кода»

При сравнении программных кодов студентов 1 курса за разные годы (рис. 6) обнаружилось, что по каждому признаку, в частности, по индексу поддерживаемости кода, показатели снизились, хотя и сохраняли одинаковую тенденцию к росту (или снижению). Сравнивались программные коды одной группы студентов направления «Математическое обеспечение и администрирование информационных систем» (контрольная группа, 21 человек, 2022 – 2023 уч.г.) и одной группы студентов (28 человек, 2023 – 2024 уч. г.), которая была составлена из учащихся разных направлений. Сравнение производилось по трём лабораторным работам.

Рис. 6. – Сравнение динамики двух групп за разные годы по метрике «Индекс поддерживаемости кода»

Для общего сравнения кодов студентов одной группы по пяти признакам используется отображение на лепестковой диаграмме (рис. 7), с помощью которой можно:

- Увидеть положение конкретного студента относительно всей группы: близки ли значения его признаков к среднему по группе, есть ли выбросы и по каким признакам. Например, для студента № 2506 значения показателей индекс поддерживаемости кода (MI), объём по Холстеду (NV), сложность по Холстеду (HC), цикломатическая сложность (CC) находятся примерно в тех же значениях, что и у большинства в группе, а доля комментариев в коде (CR) намного ниже медианного значения группы.

- Сравнить группы между собой, выделив ключевые закономерности.

- Определить, какие признаки имеют наибольшее или наименьшее значение.

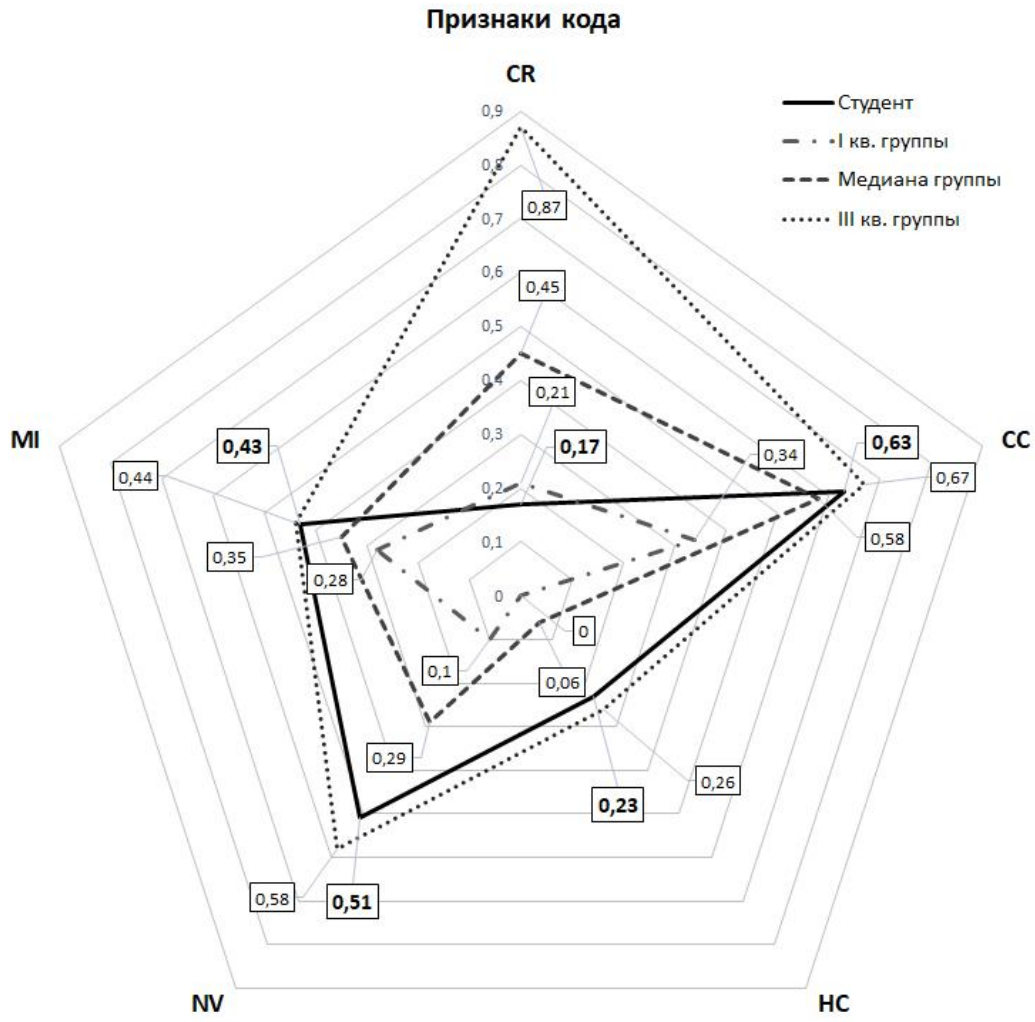

Рис. 7. – Сравнение признаков кода студента с показателями группы

Выводы. Разработанный сервис анализа программного кода позволяет преподавателям оценить программные коды студентов по ключевым признакам, увидеть траекторию изменения признаков программного кода, проиллюстрировать положение студента в группе по разным признакам кода, при этом студент, оценивая свою позицию по лепестковой диаграмме, видит, что нужно «подтянуть» при написании кода. Видя положение студентов в группе, преподаватель может выделять мини-коллективы с похожими признаками. Данный подход способствует более точному подбору пакета

индивидуальных заданий, направленных, в том числе, на корректировку особенностей кода студента.

Литература

1. Харламов А.А., Ермоленко Т.В., Дорохина Г.В. Сравнительный анализ организации систем синтаксических парсеров // Инженерный вестник Дона, 2013, №4. URL: ivdon.ru/ru/magazine/archive/n4y2013/2015.

2. Чубейко С.В., Цуриков А.Н., Палагута В.С. Анализ синтаксического разбора текста с помощью парсер-комбинаторов // Инженерный вестник Дона, 2018, №4. URL: ivdon.ru/ru/magazine/archive/n4y2018/5335.

3. Программный код и его метрики (блог компании Intel). URL: habr.com/ru/company/intel/blog/106082/

4. Wiley–IEEE Computer Society,New York, USA. Software Metrics and Software Metrology// Chapter 7. Halstead Metrics, Analysis of their Design, pp. 145–159, 2010.

5. McCabe T.J. A complexity measure // IEEE Transactions on Software Engineering, vol. SE–2, no. 4, pp. 308–320, 1976.

6. Jansen Paul. The TIOBE Quality Indicator. URL: tiobe.com/files/TIOBEQualityIndicator_v4_8.pdf.

7. ISO/IEC 25010:2011. Systems and software engineering // Systems and software Quality Requirements and Evaluation (SQuaRE) — System and software quality models. URL: iso.org/standard/35733.html.

8. Enslen E., Hill E., Pollock L. L., Vijay–Shanker K. Mining source code to automatically split identifiers for software analysis // In Proceedings of the 6th International Working Conference on Mining Software Repositories, pp. 71–80, 2009.

9. Документация по использованию доступных лексеров в библиотеке pygments // Available lexers — Pygments. URL: pygments.org/docs/lexers/.

10. Pinter Adam, Szenasi Sandor. Preprocessed C# Source Codes for Machine Learning. URL: zenodo.org/record/3264761.

11. Иваненко О. А., Бакланов И. А., Чемакин Т. А., Воробьев А. М. Информационные сервисы для сопровождения индивидуальных образовательных траекторий студентов вуза // Информатизация образования и методика электронного обучения: материалы IV Междунар. науч. конф. Красноярск: Сибирский федеральный университет, 2020. С. 115–118.

References

1. Harlamov A.A., Ermolenko T.V., Dorohina G.V. 8. Inzhenernyj vestnik Dona, 2013, №4. URL: ivdon.ru/ru/magazine/archive/n4y2013/2015

2. Chubeyko S.V., Tsurikov A.N., Palaguta V.S. 8. Inzhenernyj vestnik Dona, 2018, №4. URL: ivdon.ru/ru/magazine/archive/n4y2018/5335.

3. Programmnyj kod i ego metriki (blog kompanii Intel). URL: habr.com/ru/company/intel/blog/106082/.

4. Wiley–IEEE Computer Society, New York, USA. Software Metrics and Software Metrology. Chapter 7. Halstead Metrics, Analysis of their Design, pp. 145–159, 2010.

5. McCabe T.J. A complexity measure. IEEE Transactions on Software Engineering, vol. SE–2, no. 4, pp. 308–320, 1976.

6. Jansen Paul. The TIOBE Quality Indicator. URL: tiobe.com/files/TIOBEQualityIndicator_v4_8.pdf.

7. ISO/IEC 25010:2011. Systems and software engineering. Systems and software Quality Requirements and Evaluation (SquaRE). System and software quality models. URL: iso.org/standard/35733.html.

8. Enslen E., Hill E., Pollock L. L., Vijay–Shanker K. Mining source code to automatically split identifiers for software analysis. In Proceedings of the 6th

International Working Conference on Mining Software Repositories, pp. 71–80, 2009.

9. Dokumentatsiya po ispol'zovaniyu dostupnykh lekserov v biblioteke pygments. [Documentation on using the available lexers in the pygments library]. Available lexers. Pygments. URL: pygments.org/docs/lexers/.

10. Pinter Adam, Szenasi Sandor. Preprocessed C# Source Codes for Machine Learning. URL: zenodo.org/record/3264761.

11. Ivanenko O. A., Baklanov I. A., Chemakin T. A., Vorob'ev A. M. Informatizatsiya obrazovaniya i metodika elektronnogo obucheniya: materialy IV Mezhdunar. nauch. konf. Krasnoyarsk: Sibirskiy federal'nyy universitet, 2020. S. 115–118.

Дата поступления: 10.07.2024

Дата публикации: 12.10.2024