УДК 004.056 DOI: 10.37468/2307-1400-2023-1-86-92

ОБЗОР СУЩЕСТВУЮЩИХ МЕТОДОВ ОБНАРУЖЕНИЯ ДУБЛИКАТОВ ИСХОДНОГО КОДА

Яковлев Артем Владимирович [1]
Израилов Константин Евгеньевич [2]

[1] Санкт-Петербургский государственный университет телекоммуникаций им. проф. М.А. Бонч-Бруевича, Санкт-Петербург, Россия

[2] Санкт-Петербургский Федеральный исследовательский центр Российской академии наук, Санкт-Петербург, Россия

АННОТАЦИЯ

Данная работа посвящена решению задачи выявления дубликатов программного обеспечения. Для этого делается краткий обзор существующих методов поиска, состоящих из следующих: текстовый, лексический, синтаксический, метрический, семантический. Затем производится сравнительный анализ методов по следующим критериям: точность, полнота, скорость, ресурсоэкономность, простора реализации; результаты сравнения приводятся в табличном виде. Также рассматриваются перспективные подходы поиска дубликатов, а именно, следующие: машинное обучение, анализ графов, анализ синтаксических деревьев, анализ динамических характеристик, анализ пространственных характеристик, анализ абстрактного синтаксиса. Указываются пути продолжения исследования.

Ключевые слова: информационная безопасность, поиск дубликатов, исходный код, защита информации, авторские права.

SURVEY OF EXISTING METHODS FOR DETECTING SOURCE CODE DUPLICATES

Yakovlev Artem V. [1]
Izrailov Konstantin E. [2]

[1] The Bonch-Bruevich Saint-Petersburg state university of telecommunications, St. Petersburg, Russia

[2] SPCRAS, St. Petersburg, Russia

ABSTRACT

This work is devoted to solving the identifying software duplicates problem. To do this, a short survey of existing search methods is made, consisting of the following: textual, lexical, syntactic, metric, semantic. Then a comparative analysis of the methods is carried out according to the following criteria: accuracy, completeness, speed, resource efficiency, scope of implementation; the comparison results are given in tabular form. Also, promising approaches for searching for duplicates are considered, namely, the following: machine learning, graph analysis, syntax tree analysis, dynamic characteristics analysis, spatial characteristics analysis, abstract syntax analysis. Ways to continue investigation are indicated.

Keywords: information security, duplicate search, source code, information protection, copyright.

Введение

Современный мир невозможно представить без использования программного обеспечения. Оно проникает во все сферы государства, жизни людей и бизнеса, от маленьких приложений для смартфонов до крупных корпоративных систем. Но за каждым преимуществом программного обеспечения стоит потенциальная угроза, связанная с его использованием [1]. Нарушение различных аспектов безопасности

программного обеспечения может привести к серьезным последствиям для пользователей и организаций, а также нанести ущерб интеллектуальной собственности [2]. Одним из таких аспектов является повторное использование исходного кода [3], как в рамках одного продукта, так и при его краже. Если во втором случае нарушение авторских прав очевидно, то первый случай чреват клонированием в таком коде потенциальных уязвимостей [4]; при этом исправление уязвимости в одной версии кода может никак не отразиться на второй. Также можно вспомнить инцидент с европейской ракетой Ariane 5, которая взорвалась на 40 секунде после старта из-за копирования и дублирования кода управляющей программы предыдущей модели ракеты.

Исходя из вышесказанного существует необходимость в разработке эффективных методов и инструментов для анализа исходного кода программного обеспечения [5, 6], которые позволят выявлять наличие дубликатов в различных программах и улучшать качество программного обеспечения. Исследование дубликатов кода может быть полезным не только для обеспечения безопасности, но и для повышения производительности, обнаружения ошибок [7] и упрощения процесса разработки программного обеспечения [8].

Данное научное исследование заключается в обзоре и сравнительном анализе существующих методов поиска дубликатов в исходном коде программного обеспечения, что позволит делать обоснованный выбор среди них. Также рассматриваются перспективные подходы к обнаружению дубликатов, стремясь качественно повысить эффективность такого поиска.

Современные методы поиска дубликатов кодов

В настоящее время существует множество методов сравнения кода, которые могут быть использованы для поиска дубликатов программного обеспечения, основными типами которых являются следующие [9, 10, 11, 12]:

1) текстовый – метод основывается на сравнении хеш-кодов строк, содержащихся в исходных кодах; если хеш-коды строк совпадают, то эти строки считаются клонами; метод наиболее простой, но также наименее точный и не учитывает контекст и семантику кода;

2) лексический – метод сравнивает подпоследовательности токенов (лексических единиц), полученных при разборе исходного кода; если такие подпоследовательности совпадают, то соответствующие участки кода считаются клонами; этот метод более точен, чем текстовый метод, но также может игнорировать семантику кода;

3) синтаксический – метод использует абстрактное синтаксическое дерево (далее – АСД), полученное из исходного кода, для поиска совпадающих поддеревьев; если поддеревья АСД совпадают, то соответствующие участки кода считаются клонами; этот метод более точен, чем лексический метод, поскольку учитывает синтаксис кода, но также может игнорировать семантику;

4) метрический – метод вычисляет метрики для АСД и графов зависимостей между операторами программы (далее – ГЗП), которые представляют структуру кода, и затем сравнивает векторы этих метрик; если векторы совпадают, то соответствующие участки кода считаются клонами; этот метод может быть более точным, чем лексический и синтаксический методы, поскольку учитывает структуру кода, но также может быть более сложным в реализации;

5) семантический – метод использует ГЗП для поиска похожих подграфов; если подграфы совпадают, то соответствующие участки кода считаются клонами; этот метод является наиболее точным, поскольку учитывает и структуру, и семантику кода, но также является наиболее сложным в реализации и требует значительных вычислительных ресурсов.

Исходя из сделанного обзора существующих методов можно произвести их сравнительный анализ по следующим критериям:

К_1) точность – насколько мало количество ложных срабатываний при обнаружении дубликатов;

К_2) полнота – насколько мало количество пропущенных дубликатов;

К_3) скорость – насколько быстро работает метод, что особенно актуально для крупных программных продуктов;

К_4) ресурсоэкономность – насколько мало количество ресурсов требуется для обнаружения дубликатов (время ЦПУ, оперативная память, объем хранилища для промежуточных данных, величина сетевого трафика и т.п.);

К_5) простота реализации – какой минимально необходимый уровень предъявляется к разработчикам метода.

Произведем далее сравнение методов по выделенным критериям, оценивая последние по следующей шкале: 2 – полное соответствие критерию; 1 – частичное соответствие критерию; 0 – нет соответствия критерию (очевидно, что более высокие значения критериев будут говорить о предпочтительности использования метода).

Результаты критериального сравнения приведены в таблице 1, в которой по каждому критерию указаны следующие их суммарные значения: в последнем столбце – по всем критериям для каждого метода и в последней строке – по всем методам для каждого критерия.

Согласно анализу результатов критериального сравнения (см. Таблицу 1) можно сделать следующие выводы.

Во-первых, наиболее удовлетворяющий всем критериям является текстовый (8 баллов); это говорит о том, что несмотря на кажущуюся тривиальность, он может быть наиболее востребован для применения.

Во-вторых, наименее удовлетворительным методом можно считать семантический (5 баллов), видимо вследствие высокого ресурсопотребления и сложности реализации.

В-третьих, наиболее удовлетворительным всеми методами критерием является точность определения дубликатов (9 баллов), за которой, впрочем, следует полнота определения (8 баллов); это позволяет сделать вывод о достаточно высокой результативности (оцененной, например, с помощью F-меры, являющейся производной от точности и полноты) всех представленных методов.

И, в-четвертых, хуже всех с точки зрения удовлетворения методами оказались ресурсоэкономность и простота реализации (по 5 баллов). Так, если достичь простоты путем усовершенствования методов вряд ли получится, то снижение потребляемых ресурсов можно достичь путем проведения оптимизационных мероприятий над конкретными реализациями решений по поиску дубликатов кода.

Перспективные подходы обнаружения дубликатов

С постоянным ростом объемов программного обеспечения появляется необходимость в более совершенных методах обнаружения дубликатов, которые могут значительно улучшить качество и безопасность кода. На сегодняшний день существуют несколько таких перспективных методов, основанных на следующих подходах.

Машинное обучение

Использование алгоритмов машинного обучения [13] может помочь в создании модели, кото-

Таблица 1 – Критериальное сравнение существующих методов обнаружения дубликатов кода

Тип метода	Критерии					Сумма баллов
	К_1	К_2	К_3	К_4	К_5	
Текстовый	1	1	2	2	2	8
Лексический	2	1	1	1	2	7
Синтаксический	2	2	1	1	1	7
Метрический	2	2	1	1	0	6
Семантический	2	2	1	0	0	5
Сумма баллов	9	8	6	5	5	33

рая будет способна определять схожесть между программами. Для этого необходимо предоставить модели достаточное количество данных для обучения. В этом случае алгоритмы классификации или нейронные сети будут принимать на вход данные, включающие информацию о программных кодах, а также метаданные (например, длину кода или количество функций в программе).

Методы, основанные на машинном обучении, могут применяться в различных областях, например, при обнаружении плагиата или во время разработки новых программных продуктов. К примеру, разработчики могут использовать этот метод для проверки на уникальность кода, что может улучшить качество программы и сократить время ее разработки.

Анализ графов

Основная идея метода заключается в том, что программа может быть представлена в виде графа, в котором узлы соответствуют компонентам программы, а ребра связывают их в соответствии с их взаимодействием. Дубликаты в программах могут быть обнаружены с помощью анализа графовой структуры исходных кодов, которая поможет выявить одинаковые участки у нескольких программ.

Возможным практическим применением методов, основанных на анализе графов, может являться обнаружение дубликатов в больших и сложных проектах, где традиционные методы могут быть неэффективными. Это может оказаться особо полезно, например, при разработке программного обеспечения для банков, медицинских учреждений, космической промышленности или других организаций, где требуется высокая точность и надежность программного кода.

Анализ синтаксических деревьев

Методы, основанные на анализе синтаксических деревьев, используются для анализа структуры кода и выявления дубликатов в программном обеспечении. Для этого сначала строится синтаксическое дерево для каждой программы, которое отображает ее структуру и логику. Затем деревья сравниваются, идентифицируются участки кода, которые совпадают или очень похожи между программами, и отмечаются как потенциальные дубликаты.

Решения могут быть полезны в различных областях, таких как разработка программного обеспечения, тестирование, обнаружение уязвимостей и др. Например, при разработке кода он поможет программистам избежать дублирования кода и уменьшить клонирование ошибок, что, в свою очередь, повысит эффективность и безопасность программы. В тестировании данный метод может использоваться для поиска ошибок в коде и выявления проблем, связанных с дублированием кода, что также может помочь повысить качество программного обеспечения.

Анализ динамических характеристик

Методы, основанные на анализе динамических характеристик, используют информацию о поведении программы во время ее выполнения для выявления дубликатов. Это может включать в себя анализ времени выполнения, использование памяти, сетевой активности и других характеристик программы. В результате этого метода можно выявить дубликаты кода, которые могут быть не видны на уровне кода, например, из-за незначительных изменений или разных имен переменных.

Анализ пространственных характеристик

Методы, основанные на анализе пространственных характеристик, могут использоваться для выявления дубликатов программного обеспечения путем анализа расположения функций и переменных в программе. Этот подход может быть особенно полезен для обнаружения дубликатов в крупных и сложных проектах, где использование методов, основанных на анализе синтаксических деревьев или динамических характеристик, может быть менее эффективным или сверхресурсоемким.

Одним из основных принципов методов, основанных на анализе пространственных характеристик, является то, что дубликаты часто имеют

сходную структуру и расположение функций и переменных в коде. Алгоритмы анализа пространственных характеристик могут производить сравнение не только синтаксиса кода, но и его структуры, что делает их более точными и эффективными.

Анализ абстрактного синтаксиса

Методы, основанные на анализе абстрактного синтаксиса, могут использоваться для выявления дубликатов программного обеспечения путем анализа структуры кода и выявления участков кода, которые имеют похожую функциональность, несмотря на отличия в синтаксисе. Это достигается путем создания абстрактного синтаксического дерева, которое представляет собой представление программы в виде дерева, где каждый узел представляет отдельный оператор или выражение. Абстрактный синтаксический анализ может быть использован для сравнения различных участков кода и выявления дубликатов, которые могут отличаться по синтаксису, но имеют похожую функциональность.

Заключение

Проблема дубликатов в программном обеспечении имеет значительное значение в современном мире, где программное обеспечение играет важную роль во всех сферах жизнедеятельности. Наличие дубликатов в коде может привести к серьезным последствиям, таким как уязвимости безопасности, кража интеллектуальной собственности, а также замедление процесса разработки и увеличение затрат на его сопровождение. В связи с этим, разработка эффективных методов и инструментов для анализа исходного кода является важной задачей. Исследование дубликатов кода может помочь улучшить качество программного обеспечения, повысить его производительность и обнаружить возможные ошибки [14]. Таким образом, исследование дубликатов кода является важным направлением в развитии программной инженерии, которое поможет обеспечить безопасность [15] и качество программного обеспечения в нашей современной жизни.

В данной работе произведен обзор существующих методов поиска дубликатов, имеющих существенные отличия, обоснованный выбор которых даст количественный прирост к эффективности поиска. Также описаны перспективные методы поиска, которые могут дать качественный скачок в решении задачи определения дублируемого исходного кода.

Продолжение исследования может пойти по следующим направлениям. Во-первых, необходимо дать более формальные оценки приведенным метода поиска дубликатов кода. Во-вторых, должны быть оценены реальные возможности применения указанных перспективных методов. И, в-третьих, может оказаться востребованным создание нового метода поиска дубликатов, включающего в себя достоинства существующих и избавленного от наиболее существенных их недостатков. Все это планируется осуществить авторами в дальнейшей научной работе.

Исследование выполнено при финансовой поддержке РФФИ в рамках научного проекта № 19-29-06099.

Список литературы

1. *Абдуллин Т.И., Баев В.Д., Буйневич М.В., Бурзунов Д.Д., Васильева И.Н., Галиуллина Э.Ф. и др.* Цифровые технологии и проблемы информационной безопасности: монография. – СПб: СПГЭУ, 2021. – 163 с.

2. *Сойников М.А.* Взыскание ущерба, причиненного преступлением против интеллектуальной собственности: процессуальные аспекты // Lex Russica (Русский закон). – 2019. – № 12 (157). – С. 80-86.

3. *Слета В.Д.* Поддержка повторного использования кода на основе онтологического подхода // Современные информационные технологии. – 2010. – № 11. – С. 178-181.

4. *Косолапов Ю.В.* Об обнаружении атак типа повторного использования исполнимого кода // Моделирование и анализ информационных систем. – 2019. – Т. 26. – № 2. – С. 213-228.

5. *Буйневич М.В., Израилов К.Е.* Основы

кибербезопасности: способы анализа программ: учебное пособие. – СПб.: Санкт-Петербургский университет ГПС МЧС России, 2022. – 92 с.

6. *Буйневич М.В., Израилов К.Е.* Основы кибербезопасности: способы защиты от анализа программ: учебное пособие. – СПб.: Санкт-Петербургский университет ГПС МЧС России, 2022. – 76 с.

7. *Израилов К.Е.* Методика оценки эффективности средств алгоритмизации, используемых для поиска уязвимостей // Информатизация и связь. – 2014. – № 3. – С. 39-42.

8. *Романов Н.Е., Израилов К.Е., Покусов В.В.* Система поддержки интеллектуального программирования: машинное обучение feat. быстрая разработка безопасных программ // Информатизация и связь. – 2021. – № 5. – С. 7-17. – DOI: 10.34219/2078-8320-2021-12-5-7-16.

9. *Лисс А.Р., Андрианов И.А.* Анализ и разработка методов поиска дубликатов в программном коде // Известия СПбГЭТУ ЛЭТИ. – 2010. – № 7. – С. 55-61.

10. *Демидова Л.А., Советов П.Н., Горчаков А.В.* Кластеризация представлений текстов программ на основе цепей Маркова // Вестник Рязанского государственного радиотехнического университета. – 2022. – № 81. – С. 51-64.

11. *Степанов Д.С., Ицыксон В.М.* Поиск дубликатов ошибок компиляторов методом генерации программ-свидетелей // Программная инженерия. – 2023. – Т. 14. – № 4. – С. 165-174.

12. *Гусев С.С.* Методы анализа произвольных текстов и исходных кодов программ с точки зрения наличия идентичных фрагментов // Анализ, моделирование, управление, развитие социально-экономических систем (АМУР-2022): сборник научных трудов XVI Международной школы-симпозиума АМУР-2022 (Симферополь-Судак, 14–27 сентября 2022 года), 2022. – С. 124-132.

13. *Kotenko I., Izrailov K., Buinevich M.* Static Analysis of Information Systems for IoT Cyber Security: A Survey of Machine Learning Approaches // Sensors. – 2022. – Vol. 22. – Iss. 4. – PP. 1335. – DOI: 10.3390/s22041335

14. *Израилов К.Е.* Моделирование программы с уязвимостями с позиции эволюции ее представлений. Часть 1. Схема жизненного цикла // Труды учебных заведений связи. – 2023. – Т. 9. – № 1. – С. 75–93. –DOI:10.31854/1813-324X-2023-9-1-75-93.

15. *Kotenko I., Izrailov K., Buinevich M.* Analytical Modeling for Identification of the Machine Code Architecture of Cyberphysical Devices in Smart Homes // Sensors. – 2022. – Vol. 22. – Iss. 3. – PP. 1017. – DOI: 10.3390/s22031017

References

1. *Abdullin T. I., Baev V. D., Buynevich M. V., Burzunov D. D., Vasilieva I. N., and Galiullina E. F.* Digital technologies and problems of information security: monograph. – St. Petersburg: SPGEU, 2021. – 163 p.

2. *Soinikov M.A.* Recovery of damage caused by a crime against intellectual property: procedural aspects // Lex Russica (Russian law). – 2019. – No. 12 (157). – P. 80-86.

3. *Sleta V.D.* Support for code reuse based on the ontological approach // Modern information technologies. – 2010. – No. 11. – P. 178-181.

4. *Kosolapov Yu.V.* On detecting attacks such as reuse of executable code // Modeling and analysis of information systems. – 2019. – Vol. 26. – No. 2. – P. 213-228.

5. *Buynevich M.V., Izrailov K.E.* Fundamentals of Cybersecurity: Ways to Analyze Programs: A Study Guide. – St. Petersburg: St. Petersburg University of the State Fire Service of the Ministry of Emergency Situations of Russia, 2022. – 92 p.

6. *Buynevich M.V., Izrailov K.E.* Fundamentals of cybersecurity: ways to protect against program analysis: a tutorial. – St. Petersburg: St. Petersburg University of the State Fire Service of the Ministry of Emergency Situations of Russia, 2022. – 76 p.

7. *Izrailov K.E.* Methodology for evaluating the effectiveness of algorithmization tools used to search for vulnerabilities // Informatization and communication. – 2014. – No. 3. – P. 39-42.

8. *Romanov N.E., Izrailov K.E., Pokusov V.V.* Intelligent Programming Support System: Machine

Learning feat. rapid development of secure programs // Informatization and communication. – 2021. – No. 5. – P. 7-17. – DOI: 10.34219/2078-8320-2021-12-5-7-16.

9. *Liss A.R., Andrianov I.A.* Analysis and development of methods for searching for duplicates in the program code. – 2010. – No. 7. – P. 55-61.

10. *Demidova L.A., Sovetov P.N., Gorchakov A.V.* Clustering representations of program texts based on Markov chains // Bulletin of the Ryazan State Radio Engineering University. – 2022. – No. 81. – P. 51-64.

11. *Stepanov D.S., Itsykson V.M.* Finding Duplicate Compiler Errors by Generating Witness Programs // Software Engineering. – 2023. – Vol. 14. – No. 4. – P. 165-174.

12. *Gusev S.S.* Methods for analyzing arbitrary texts and program source codes from the point of view of the presence of identical fragments // Analysis, modeling, management, development of socio-economic systems (AMUR-2022): collection of scientific papers of the XVI International School-Symposium AMUR-2022 (Simferopol-Sudak, 14 – September 27, 2022), 2022. – P. 124-132.

13. *Kotenko I., Izrailov K., Buinevich M.* Static Analysis of Information Systems for IoT Cyber Security: A Survey of Machine Learning Approaches // Sensors. – 2022. – Vol. 22. – Iss. 4. – PP. 1335. – DOI: 10.3390/s22041335

14. *Izrailov K.E.* Modeling a program with vulnerabilities from the standpoint of the evolution of its ideas. Part 1. Scheme of the life cycle // Proceedings of educational institutions of communication. – 2023. – Vol. 9. – No. 1. – P. 75–93. –DOI:10.31854/1813-324X-2023-9-1-75-93.

15. *Kotenko I., Izrailov K., Buinevich M.* Analytical Modeling for Identification of the Machine Code Architecture of Cyberphysical Devices in Smart Homes // Sensors. – 2022. – Vol. 22. – Iss. 3. – PP. 1017. – DOI: 10.3390/s22031017

Статья поступила в редакцию 12 февраля 2023 г.
Принята к публикации 17 марта 2023 г.

Ссылка для цитирования: Яковлев А.В., Израилов К.Е. Обзор существующих методов обнаружения дубликатов исходного кода // Национальная безопасность и стратегическое планирование. 2023. № 1(44). С. 86-92. DOI: https://doi.org/10.37468/2307-1400-2023-1-86-92

For citation: Yakovlev A.V., Izrailov K.E. Survey of existing methods for detecting source code duplicates // National security and strategic planning. 2023. № 1(41). pp. 86-92. DOI: https://doi.org/10.37468/2307-1400-2023-1-86-92

Сведения об авторах:

ЯКОВЛЕВ АРТЕМ ВЛАДИМИРОВИЧ – *студент Санкт-Петербургского государственного университета телекоммуникаций им. проф. М.А. Бонч-Бруевича (193232, Россия, Санкт-Петербург, пр. Большевиков, д. 22/1), https://orcid.org/0009-0006-5590-5095*
e-mail: 89062703467@mail.ru
ИЗРАИЛОВ КОНСТАНТИН ЕВГЕНЬЕВИЧ – *кандидат технических наук, доцент, старший научный сотрудник лаборатории проблем компьютерной безопасности Санкт-Петербургского Федерального исследовательского центра Российской академии наук (199178, Россия, Санкт-Петербург, В.О., 14-я линия, 39), https://orcid.org/0000-0002-9412-5693*
e-mail: konstantin.izrailov@mail.ru

Information about authors:

YAKOVLEV ARTEM V. – *student of The Bonch-Bruevich Saint-Petersburg state university of telecommunications (193232, Russia, St. Petersburg, Prospekt Bolshevikov, No. 22/1), https://orcid.org/0009-0006-5590-5095*
e-mail: 89062703467@mail.ru
IZRAILOV KONSTANTIN E. – *Candidate of technical sciences, Associate professor, Senior Researcher at the Computer Security Problems Laboratory of the St. Petersburg Federal Research Center of the Russian Academy of Sciences (199178, Russia, St. Petersburg, 14-th Linia, VI, No. 39), https://orcid.org/0000-0002-9412-5693*
e-mail: konstantin.izrailov@mail.ru