УДК 004.052
DOI: 10.25559/SITITO.18.202204.855-860

Оригинальная статья

Количественные характеристики безопасности программ

А. А. Куликовская[1*], Е. А. Доренская[1], Ю. А. Семенов[1,2]
[1] ФГБУ «Национальный исследовательский центр «Курчатовский институт»», г. Москва, Российская Федерация
Адрес: 123182, Российская Федерация, г. Москва, пл. Академика Курчатова, д. 1
[*] kulikovskaya@phystech.edu
[2] ФГАОУ ВО «Московский физико-технический институт (национальный исследовательский университет)», г. Долгопрудный, Российская Федерация
Адрес: 141701, Российская Федерация, Московская область, г. Долгопрудный, Институтский переулок, д. 9

Аннотация
Качество программ принято характеризовать числом ошибок на 1000 строк кода. Эта характеристика получается в результате регрессионного анализа числа выявленных ошибок в последовательных версиях кода с последующей экстраполяцией в отдаленное будущее. Данная процедура является очень трудоемкой даже для крупных компаний. Проверить достоверность такой оценки для обычных пользователей как правило довольно сложно. Такая проблема возникает из-за недоступности исходных данных. Существуют различные способы оценки числа ошибок в программе, например, модель Шумана, Муса, Ла Падула, Джелинского-Моранды, Шика-Волвертона, модель переходных вероятностей, статистическая модель Миллса, простая интуитивная модель, модель Липова, Коркорэна, Бернулли, Нельсона и т.д. Часто приходится иметь дело с программами, в которых формально нет ошибок, при этом их качество на первый взгляд не очевидно. В данной статье предложен новый метод количественной оценки качества программ, написанных на языке Perl. Данный метод позволяет выявить места в программах, где могут быть допущены ошибки. В отличие от существующих алгоритмов данный метод позволяет оценивать качество программы, анализируя стиль написания ее кода. Предлагаемый метод оценки качества применим к любым программам, написанным на алгоритмических языках высокого уровня с открытыми кодами (Python, Perl, PHP и т.д.). В том числе его можно использовать для сравнения качества и выбора программ, решающих одну и ту же задачу.

Ключевые слова: безопасность, тестирование, принципы структурного программирования

Конфликт интересов: авторы заявляют об отсутствии конфликта интересов.

Для цитирования: Куликовская А. А., Доренская Е. А., Семенов Ю. А. Количественные характеристики безопасности программ // Современные информационные технологии и ИТ-образование. 2022. Т. 18, № 4. С. 855-860. doi: https://doi.org/10.25559/SITITO.18.202204.855-860

Original article

Quantitative Security Characteristics of Perl Programs

A. A. Kulikovskaya[a*], E. A. Dorenskaya[a], Yu. A. Semenov[a,b]
[a] National Research Centre "Kurchatov Institute", Moscow, Russian Federation
Address: 1 Academician Kurchatov Square, Moscow 123182, Russian Federation
[*] kulikovskaya@phystech.edu
[b] Moscow Institute of Physics and Technology (National Research University), Dolgoprudny, Russian Federation
Address: 9 Institutskiy per., Dolgoprudny 141701, Moscow Region, Russian Federation

Abstract
The program quality is used to be characterized with error number per 1000 code lines. This parameter is calculated by a statistical regressive analysis of error numbers for successive code versions, with a subsequent extrapolation for the future. This procedure is very tedious even for large companies. It is very hard to verify this estimate for common users, as they have no initial data. There are a lot of methods to estimate code error number, e.g., models Shooman, Musa, Bell-LaPadula, Jelinski-Moranda, Schick-Wolverton, Mills, Lipov, Corcoran, Bernoulli simple intuitive software reliability model, Nelson's software reliability. But often we deal with programs that formally have no errors, at the same time their quality is not evident. The method is proposed to estimate quantitatively a code quality for Perl-routines. This method can identify weaknesses in certain program components, where errors are possible. The proposed method is based on programming style analysis. The method is applicable for any programs with open sources (Python, Perl, PHP, etc.). The method can be used for quality comparison and choice of the programs solving similar tasks.

Keywords: security, testing, structured programming principles

Conflict of interests: The authors declare no conflict of interest.

For citation: Kulikovskaya A.A., Dorenskaya E.A., Semenov Yu.A. Quantitative Security Characteristics of Perl Programs. *Modern Information Technologies and IT-Education.* 2022;18(4):855-860. doi: https://doi.org/10.25559/SITITO.18.202204.855-860

Введение

В настоящее время проблема контроля качества данных стоит достаточно остро, так как анализ информации используется как основа для принятия решений. Существуют несколько программных продуктов для оценки качества данных[1] [1-18]. Как правило это базы знаний, помогающие проконтролировать полноту, целостность, своевременность, а также не искаженность хранящейся информации. В совокупности эти характеристики могут быть показателем качества информации. Существует много видов данных – статистика, результаты измерений и т.д., одним из наиболее важных являются тексты программ.

В данной статье вводится понятие качества программного обеспечения и предлагается метод его количественной оценки.

Существует несколько классов ошибок, которые можно выявить на этапах написания программы. Для этой цели служат отладчики, статистические и динамические анализаторы кода [19], которые позволяют выявить синтаксические и некоторые алгоритмические ошибки в программе. В случае, когда программа запускается и выдает какие-то результаты, для выявления ошибок используется тестирование [20] на большом количестве данных.

Однако, существуют случаи, когда выявить ошибки с помощью тестирования невозможно. В таком случае можно воспользоваться предположениями, основанными на данных, определяющих наиболее «слабые» места в программе – некорректности, а именно определенные используемые методы или функции, в которых ошибки совершаются чаще, чем в других.

Описание предлагаемого метода

В 2006 г. ведущий специалист подразделения надежного программного обеспечения LaRS лаборатории реактивных двигателей НАСА Джерард Хольцман [21] сформулировал 10 правил создания надежного софта, которые выработались в многолетней практике подготовки критически важного программного обеспечения[2]. Эти правила, схожи с классическими принципами структурного программирования, придуманными в 60-х годах XX века, которые ориентированы на формирование кодировщиком исходного кода, который затем подвергается автоматическому анализу. В программе ищутся фрагменты, которые формально допустимы, однако крайне нежелательны, так как относятся к плохому стилю программирования. Программист, который допускает такие отклонения от принципов структурного программирования с большой вероятностью может совершать и программные ошибки. Испытание метода проведено для программ Perl, так как для них всегда доступен исходный текст кода [22-25].

К регистрируемым некорректностям относится, например, модификация индекса цикла DO внутри тела цикла. Кроме того, некоторые ошибки могут с разной вероятностью влиять на работу программы, поэтому необходимо добавить метрику (**m**) – некоторое относительное число, которое будет обозначать «вес» определенной ошибки в исследуемой программе. Полный набор метрик представлен в таблице 1.

Ниже предлагается метод количественной оценки безопасности программы (**M**), по числу выявленных некорректностей с учетом количества строк кода. Каждому виду угрозы ставится в соответствие определенное значение метрики (**m**). Например, неявная рекурсия (программа А обращается к программе В, которая в свою очередь вызывает А) будет считаться достаточно опасным инцидентом. Еще более значимым можно считать изменение индекса цикла DO (FOR) внутри цикла (программа может при определенных условиях зацикливаться). Похожая проблема может возникнуть при вычислении значения верхнего предела цикла DO (FOR). Ошибки в типах входных и выходных данных подпрограммы или функции следует также считать серьезной уязвимостью. Например, объявление имени индекса цикла global может создавать проблемы, так как индекс может модифицироваться где-то в подпрограмме или функции. Аналогично, с особой осторожностью следует относиться к процедуре вычисления индекса массива.

В результате оценка качества программа будет характеризоваться числом $\mathbf{M=(c/N_L)\sum n_i \times m_i}$, где $\mathbf{n_i}$ – число детектированных потенциальных уязвимостей типа **i**, $\mathbf{m_i}$ – метрика уязвимости типа **i**, **c** – нормировочный коэффициент, а $\mathbf{N_L}$ – число строк в исследуемой программе. Ниже, в Таблице 1, приведены названия уязвимостей, а также их весовые коэффициенты.

Таблица 1. **Значения метрик уязвимостей с весовыми коэффициентами**
Table 1. **Values of vulnerability metrics with weighting coefficients**

№	Описание	Значение метрики (m)
1	Модификация индекса цикла DO внутри тела цикла	6
2	Рекурсия	8
3	Неявная рекурсия	8
4	Недопустимое значение индекса массива (отрицательное или за пределами верхней границы	12
5	Некорректный тип входных данных	2

[1] Abiteboul S., Buneman P., Suciu D. Data on the Web: from relations to semistructured data and XML. San Francisco, CA, USA : Morgan Kaufmann Publishers Inc.; 1999. 258 p.

[2] Семенов Ю. А. 10 правил Хольцмана и рекомендации для написания программ, критичных в отношении безопасности [Электронный ресурс]. URL: http://book.itep.ru/10/holz_rules.htm (дата обращения: 07.09.2022); Wayner P. Safeguard your code: 17 security tips for developers [Электронный ресурс] // InfoWorld. Feb 4, 2013. URL: https://www.infoworld.com/article/2078701/safeguard-your-code--17-security-tips-for-developers.html (дата обращения: 07.09.2022).

№	Описание	Значение метрики (m)
6	Некорректный тип выходных данных	2
7	Нелегальное изменение значения индекса стека	10
8	Несанкционированная гибель процесса (в системах управления)	15
9	Появление нелегального процесса	12
10	Некорректный результат подпрограммы или функции	6
11	Слишком длинный программный модуль (более 60 строк в одной функции)	int(N_l/40)+2
12	Недопустимое обращение к подпрограмме или функции (нарушение прав доступа)	10
13	Превышен верхний предел времени работы подпрограммы или функции	2
14	Верхний предел цикла DO вычисляется (не задан явно)	4
15	Попытка модификации константы	5
16	Нелегальная модификация переменной или элемента массива	5
17	Нелегальная попытка доступа к файлу	8
18	Нелегальная попытка доступа к каналу передачи данных	8
19	Нелегальная попытка копирования файла	8
20	Нелегальная попытка стирания файла	9
21	Слишком много вложенных условных операторов (более 2)	5
22	Слишком много вложенных циклов (более двух)	4
23	Переменная индекса цикла объявлена с типом global	5
24	Индекс внутреннего и внешнего циклов имеют одинаковые имена	6
25	Файл открыт, но не закрыт	6
26	Отсутствует требование use strict	6

Данная таблица может пополняться. Чем больше потенциальных некорректностей выявлено, тем надежнее оценка качества исследуемой программы. Чем длиннее список возможных некорректностей (Таблица 1), тем надежнее используемый метод оценки.

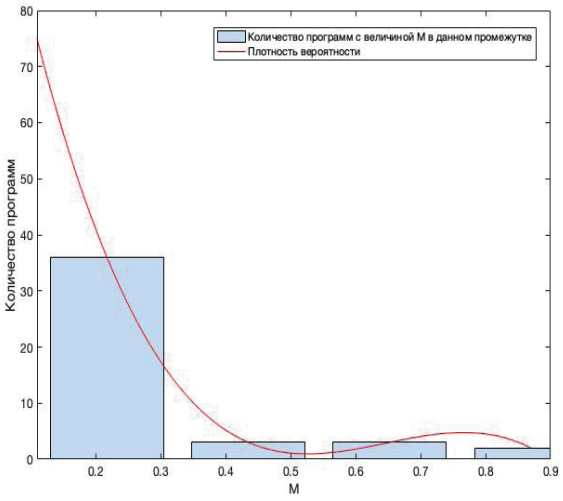
Полученные результаты

На данный момент протестировано 46 программ, написанных на языке Perl. Для представления выводов о наиболее часто встречающихся потенциальных уязвимостях количество протестированных программ должно быть достаточно велико. Ниже, на рисунке 1 приведено распределение программ по числу M (ось X), которое определяет количество программ (ось Y), имеющих заданный уровень некорректности, рассчитанный по формуле 1.

Наиболее часто встречающимися некорректностями оказались следующие:

- Модификация индекса цикла внутри тела цикла (1)
- Слишком длинный программный модуль (46)
- Верхний предел цикла DO/FOR вычисляется (не задан явно) (17)
- Файл открыт, но не закрыт (30)
- Отсутствует use strict (4)

В данной статистике наибольший вклад в значение M внесли такие некорректности, как вычисление границ циклов и длина текста тестируемых программ. Разработана аналогичная программа для оценки качества программ, написанных на языке Python.



Р и с. 1. Плотность вероятности возможных некорректностей

F i g. 1. Probability density of possible incorrectness

Выводы

Предлагаемый метод позволяет количественно оценить качество написанных программ и сравнивать две или более программ, имеющих одинаковые назначения. При этом в результате оценки разработчик получает сведения о том, как именно был получен данный количественный результат, и о позициях в коде, где обнаружены некорректности.

References

[1] Li M., Xiao D., Huang H., Zhang B. Multi-level video quality services and security guarantees based on compressive sensing in sensor-cloud system. *Journal of Network and Computer Applications*. 2022;205:103456. doi: https://doi.org/10.1016/j.jnca.2022.103456

[2] Zhang Z., Wu W., Wu D. A Multi-Mode Learning Behavior Real-time Data Acquisition Method Based on Data Quality. In: 2021 2nd International Symposium on Computer Engineering and Intelligent Communications (ISCEIC). Nanjing, China: IEEE Computer Society; 2021. p. 64-69. doi: https://doi.org/10.1109/ISCEIC53685.2021.00021

[3] Wang Y., Yu C., Hou J., Zhang Y., Fang X., Wu S. Research on the Key Issues of Big Data Quality Management, Evaluation, and Testing for Automotive Application Scenarios. *Complexity.* 2021;2021:9996011. doi: https://doi.org/10.1155/2021/9996011

[4] Pipino L.L., Lee Y.W., Wang R.Y. Data quality assessment. *Communications of the ACM*. 2002;45(4):211-218. doi: https://doi.org/10.1145/505248.506010

[5] Govender S.G., Kritzinger E., Loock M. A Framework for the Assessment of Information Security Risk, the Reduction of Information Security Cost and the Sustainability of Information Security Culture. In: Silhavy R. (ed.) Applied Informatics and Cybernetics in Intelligent Systems. CSOC 2020. Advances in Intelligent Systems and Computing. Vol. 1226. Cham: Springer; 2020. p. 69-84. doi: https://doi.org/10.1007/978-3-030-51974-2_7

[6] Aiken P.H. Reverse Engineering of Data. *IBM Systems Journal*. 1998;37(2):246-269. doi: https://doi.org/10.1147/sj.372.0246

[7] Arenas M., Bertossi L., Chomicki J. Consistent query answers in inconsistent databases. In: Proceedings of the eighteenth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems (PODS '99). New York, NY, USA: Association for Computing Machinery; 1999. p. 68-79. doi: https://doi.org/10.1145/303976.303983

[8] Carlo B., Cinzia C., Chiara F., Andrea M. Methodologies for data quality assessment and improvement. *ACM Computing Surveys*. 2009;41(3):16. doi: https://doi.org/10.1145/1541880.1541883

[9] Ballou D.P., Wang R., Pazer H.L., Tayi G.K. Modelling Information Manufacturing Systems to Determine Information Product Quality. *Management Science*. 1998;44(4):433-594. doi: https://doi.org/10.1287/mnsc.44.4.462

[10] Bovee M., Srivastava R.P., Mak B. A Conceptual Framework and Belief- Function Approach to Assessing Overall Information Quality. *International Journal of Intelligent Systems*. 2003;18(1):51-74. doi: https://doi.org/10.1002/int.10074

[11] Cappiello C., Francalanci C., Pernici B. Time-Related Factors of Data Quality in Multichannel Information Systems. *Journal of Management Information Systems*. 2003;20(3):71-91. doi: https://doi.org/10.1080/07421222.2003.11045769

[12] Eppler M.J. Managing Information Quality. Increasing the Value of Information in Knowledge-intensive Products and Processes. Heidelberg: Springer Berlin; 2003. 398 p. doi: https://doi.org/10.1007/3-540-32225-6

[13] Jarke M., Jeusfeld M.A., Quix C., Vassiliadis P. Architecture and Quality in Data Warehouses: An Extended repository Approach. *Information Systems*. 1999;24(3):229-253. doi: https://doi.org/10.1016/S0306-4379(99)00017-4

[14] Naumann F. Quality-Driven Query Answering for Integrated Information Systems. In: Lecture Notes in Computer Science. Vol. 2261. Heidelberg: Springer Berlin; 2002. 168 p. doi: https://doi.org/10.1007/3-540-45921-9

[15] Orr K. Data Quality and Systems Theory. *Communications of the ACM*. 1998;41(2):66-71. doi: https://doi.org/10.1145/269012.269023

[16] Wang R.Y. A Product Perspective on Total Data Quality Management. *Communications of the ACM*. 1998;41(2):58-65. doi: https://doi.org/10.1145/269012.269022

[17] Wang R.Y., Strong D.M. Beyond Accuracy: What Data Quality Means to Data Consumers. *Journal of Management Information Systems.* 1996;12(4):5-33. doi: https://doi.org/10.1080/07421222.1996.11518099

[18] Wand Y., Wang R.Y. Anchoring Data Quality Dimensions in Ontological Foundations. *Communication of the ACM*. 1996;39(11):86-95. doi: https://doi.org/10.1145/240455.240479

[19] Coust P. Chapter 15 – Methods and Logics for Proving Programs. *Formal Models and Semantics*. Handbook of Theoretical Computer Science. Elsevier Science; 1990. p. 843-993. doi: https://doi.org/10.1016/B978-0-444-88074-1.50020-2

[20] Dorenskaya E.A., Semenov Yu.A. About the Programming Techniques, Oriented to Minimize Errors. *Modern Information Technologies and IT-Education.* 2017;13(2):50-56. (In Russ., abstract in Eng.) doi: https://doi.org/10.25559/SITITO.2017.2.226

[21] Holzmann G.J. The power of 10: rules for developing safety-critical code. *Computer*. 2006;39(6):95-99. doi: https://doi.org/10.1109/MC.2006.212

[22] Schilling W., Alam M. A methodology for quantitative evaluation of software reliability using static analysis. In: 2008 Annual Reliability and Maintainability Symposium. Las Vegas, NV, USA: IEEE Computer Society; 2008. p. 399-404. doi: https://doi.org/10.1109/RAMS.2008.4925829

[23] de Sousa A.L.R., de Souza C.R.B., Reis R.Q. A 20-year mapping of Bayesian belief networks in software project management. *IET Software*. 2022;16(1):14-28. doi: https://doi.org/10.1049/sfw2.12043

[24] Sonnekalb T., Heinze T.S., Mäder P. Deep security analysis of program code. *Empirical Software Engineering*. 2022;27(1):2. doi: https://doi.org/10.1007/s10664-021-10029-x

[25] Villalón-Fonseca R. The nature of security: A conceptual framework for integral-comprehensive modeling of IT security and cybersecurity. *Computers & Security*. 2022;120:102805. doi: https://doi.org/10.1016/j.cose.2022.102805

Поступила 07.09.2022; одобрена после рецензирования 13.11.2022; принята к публикации 25.11.2022.
Submitted 07.09.2022; approved after reviewing 13.11.2022; accepted for publication 25.11.2022.

Об авторах:

Куликовская Анна Алексеевна, младший научный сотрудник, ФГБУ «Национальный исследовательский центр «Курчатовский институт»» (123182, Российская Федерация, г. Москва, пл. Академика Курчатова, д. 1), **ORCID: https://orcid.org/0000-0002-0214-1697**, kulikovskaya@phystech.edu

Доренская Елизавета Александровна, инженер-программист, ФГБУ «Национальный исследовательский центр «Курчатовский институт»» (123182, Российская Федерация, г. Москва, пл. Академика Курчатова, д. 1), **ORCID: https://orcid.org/0000-0002-4249-5131,** dorenskaya@itep.ru

Семенов Юрий Алексеевич, ведущий научный сотрудник Института теоретической и экспериментальной физики имени А.И. Алиханова, ФГБУ «Национальный исследовательский центр «Курчатовский институт»» (123182, Российская Федерация, г. Москва, пл. Академика Курчатова, д. 1); заместитель заведующего кафедрой информатики и вычислительных сетей, Институт нано-, био-, информационных, когнитивных и социогуманитарных наук и технологий, ФГАОУ ВО «Московский физико-технический институт (национальный исследовательский университет)» (141701, Российская Федерация, Московская область, г. Долгопрудный, Институтский переулок, д. 9), кандидат физико-математических наук, **ORCID: https://orcid.org/0000-0002-3855-3650**, semenov@itep.ru

Все авторы прочитали и одобрили окончательный вариант рукописи.

About the authors:

Anna A. Kulikovskaya, Junior Researcher, National Research Centre “Kurchatov Institute” (1 Academician Kurchatov Square, Moscow 123182, Russian Federation), **ORCID: https://orcid.org/0000-0002-0214-1697**, kulikovskaya@phystech.edu

Elizaveta A. Dorenskaya, Software Engineer, National Research Centre “Kurchatov Institute” (1 Academician Kurchatov Square, Moscow 123182, Russian Federation), **ORCID: https://orcid.org/0000-0002-4249-5131,** dorenskaya@itep.ru

Yuri A. Semenov, Lead Researcher of the Institute for Theoretical and Experimental Physics named by A.I. Alikhanov of National Research Centre “Kurchatov Institute” (1 Academician Kurchatov Square, Moscow 123182, Russian Federation); Deputy Head of the Chair for Computer Science, Institute of Nano-, Bio-, Information, Cognitive and Socio-humanistic Sciences and Technologies, Moscow Institute of Physics and Technology (National Research University) (9 Institutskiy per., Dolgoprudny 141701, Moscow Region, Russian Federation), Cand. Sci. (Phys.-Math.), **ORCID: https://orcid.org/0000-0002-3855-3650**, semenov@itep.ru

All authors have read and approved the final manuscript.