# Моделирование испытаний программных средств с использованием методов мягких вычислений

А. С. Марков
Кафедра информационной безопасности
ФУ при Правительстве РФ
asmarkov@fa.ru

Г. А. Марков
Кафедра «Информационная безопасность»
МГТУ им. Н.Э.Баумана
gm@cnpo.ru

С. А. Петренко
Кафедра информационной безопасности
СПбГЭТУ «ЛЭТИ»
s.petrenko@rambler.ru

***Аннотация*. В работе показана востребованность применения моделей оценки надежности и безопасности функционирования программ на ранних этапах испытаний. Подчеркнуто, что рост надежности и безопасности функционирования современных программных систем может иметь немонотонной характер. Приведены примеры направлений повышения адекватности и точности моделирования испытаний программ. Показаны возможность и условия применения мягких вычислений для оценки надёжности и безопасности функционирования программного обеспечения информационных систем.**

***Ключевые слова: надежность функционирования программ; испытания программ; модели роста надежности; модели отладки; нечеткие модели; искусственные нейронные сети; мягкие вычисления***

## I. Введение

Утверждение обязательных требований по формализации результатов оценки соответствия программных средств защиты информации высокого уровня доверия [1] обусловило актуальность применения математических моделей оценки надежности и безопасности функционирования программного обеспечения (ПО). Несмотря на то, что исследование математических моделей оценки надежности ПО началось еще во второй половине прошлого века [2, 3], данная тематика остается востребованной и в наши дни [4–15]. Указанное связано с развитием новых технологий производства программ (например, открытого ПО) и с внедрением новых международных стандартов в области программной инженерии и информационной безопасности (ISO 15408, ISO 33001, IEC 61508, IEC 61511 и др.) [8, 16, 17]. Надо понимать, что понятие надежности ПО эквивалентно понятию безопасности ПО в случае принятия того, что выявляемые дефекты и уязвимости не идентифицируются как преднамеренные. Поэтому рассматриваемые в работе модели испытаний ПО будем по традиции именовать моделями надёжности. Как известно, разработка, выбор или синтез математических моделей напрямую зависят от подсистемы сбора статистики, внедренной в жизненный цикл ПО. Исследованию возможности и способов применения мягких вычислений в рамках математического моделирования испытаний и доработки программ посвящена данная статья.

## II. Модель отладки

В настоящее время известен ряд таксономий математических моделей надёжности программ [2, 18–20]. На наш взгляд – в прикладном плане – удобно использовать классификацию, связанную с целями этапов тестирования и испытаний ПО [20], согласно которой можно выделить 4 вида математических моделей:

- модели планирования испытаний, учитывающие сложность программ (program complexity models);
- модели отладки, ориентированные на множественные доработки, а также покрытие областей входных данных (data-domains models);
- временные модели роста надежности, ориентированные на время опытной и боевой эксплуатации (time-domains models);
- модели полноты испытаний (test confidence models).

Наиболее чувствительными к системе сбора статистики следует назвать модели отладки, так как начальные этапы испытаний характеризуются множественными изменениями версий объекта испытаний, в том числе в различных средах. Первыми моделями указанного класса являются модели Нельсона и ЛаПадулы, плюс их модификации [3, 21–24], предполагающие строго монотонно возрастающий рост надежности ПО, что не всегда соответствует особенностям современных многоверсионных технологий программирования. Для исключения указанного недостатка, авторами достаточно глубоко была проработана реализация вероятностной модели роста надежности, позволяющей учесть ряд позитивных и негативных факторов отладки, а именно [20]:

$$P_u = P_\infty - \left(P_\infty - P_0\right)\prod_{j=1}^{u}\left(1 - A_j / P_\infty\right), \qquad (1)$$

где: $P_0$ – начальный уровень надежности, $P_\infty = \frac{A_j}{A_j + B_j}$ – предельный уровень надежности, $0 \le P_0 < P_\infty \le 1$, $u$ – число модификаций; $A_j$ – коэффициент «эффективности» $j$-ой модификации; $B_j$ – коэффициент «негативности» $j$-ой модификации.

В зависимости от имеющейся статистики, эволюция моделирования испытаний и отладки программ может идти, например, путем:

- уточнения параметров вероятностной модели (1) в зависимости от значимости класса целевого обновления программ;
- уточнения параметров вероятностной модели (1) в зависимости от уровня сложности доработки кода;
- Байсовских модификаций модели в случае уточнения стохастической природы ее параметров;
- построения интервальной модели на базе аппарата теории нечетких множеств;
- создания модели с помощью технологий искусственных нейронных сетей и др.

Последние два варианта принято относить к мягким вычислениям, которые будут рассмотрены ниже. Байсовские модификации усложняют наглядность и простоту моделей, поэтому они вышли за рамки статьи. Однако указанный теоретический подход представлен в литературе [25–28]. Кратко коснемся вопросов уточнения модели (1) путем модификации ее параметров.

## III. Уточнение аналитических моделей

Что касается уточнения параметров модели (1), то наиболее наглядной является бигеминальная модель отладки, учитывающая противоположные классы модификаций программ [20]:

$$P_u = P_\infty - (P_\infty - P_0)\prod_{j=1}^{u}(1 - \sum_{i=1}^{2} a_i k_{ij} / P_\infty), \qquad (2)$$

где: $a_1$ – коэффициент эффективности модификации программы с целью исправления ошибки, $a_2$ – коэффициент эффективности модификации программы с целью добавления функциональных возможностей, $k_{ij}$ – объем $j$-ой модификации с целью исправления или обновления.

Бигеминальная модель (2) зависит от 4-х параметров $(P_0, P_\infty, a_1, a_2)$, расчет которых не представляет труда, например, с помощью метода максимального правдоподобия [20]. На область определения коэффициента объема модификации $k_{ij}$ не накладываются ограничения, что обеспечивает полноту описания процесса испытания и отладки. В случае статистики о модификации исходного кода, легко задействовать метрики сложности. Для машиноориентированных и процедурных (отчасти и объектно-ориентированных) языков это легко реализуется путем использования эвристических и статистических метрик сложности (Холстеда, МакКейба, Чепена и др.). Для визуальных систем программирования введение популярных метрик сложности не всегда может соответствовать ожиданиям: в итоге их можно просто свести к длине или объему модифицируемых фрагментов.

Опыт моделирования показал возможность повышения адекватности и точности за счет учета феномена немонотонности роста надежности многоверсионного ПО. Графически процесс изменения надежности ПО представляется ступенчатой функцией (рис. 1).

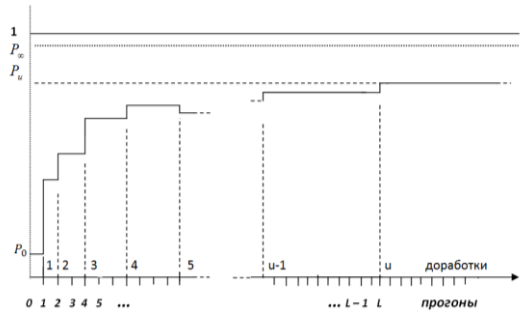


Рис. 1. Изменение степени надежности по результатам доработок

## IV. Использование мягких вычислений

К использованию мягких вычислений прибегают в выраженных случаях имеющейся статистики о тестировании и отладке ПО. К наиболее популярным подходам, как ранее указывалось, относят применение технологий искусственных нейронных систем [18, 29–31] и аппарата нечетких множеств [32–35].

### *A. 4.1. Использование технологий нейронных сетей*

Переход к использованию нейронных сетей связан со сложностью учета доработок в многоверсионном ПО (например, при открытом кода, где программисты распределены во времени, стилях, технологиях и квалификации), в результате чего на начальных этапах не понятны какие-либо закономерности. Надо понимать, что нейронные технологии реализуются при наличии достаточно представительной по объему статистики.

Для проверки данного подхода путем проведения компьютерного эксперимента были выбран пакет Neuroshell2 (Pro) [30]. В рамках исследования авторы традиционно проверили возможность моделирования с помощью допустимых топологий нейронных сетей и подбора оптимального числа их параметров (число нейронов, слоев, вида связей и т.д.). Исходные данные и результаты компьютерного эксперимента представлены на рис. 2–5.

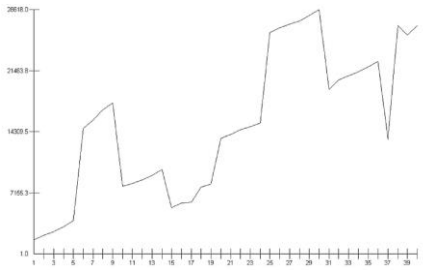

Рис. 2. Исходные данные (количество успешных испытаний)

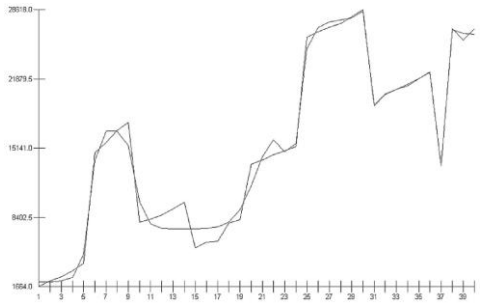

Рис. 3. Применение простой нейронной сети

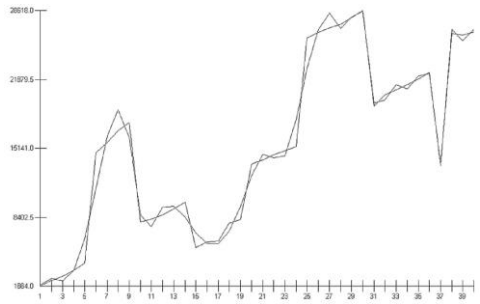

Рис. 4. Применение сети с обходными соединениями

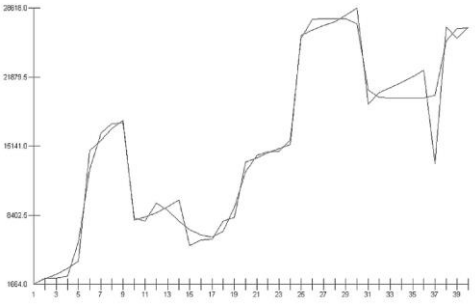

Рис. 5. Применение рекуррентных сетей с отрицательной обратной связью

Исследование продемонстрировало, что с поставленной задачей справились даже простая 4-х слойная нейронная сеть и сеть с обходными соединениями [30]. Надо понимать, что использование технологий нейронных сетей требует поддержку соответствующей системы сбора данных и накопления весьма представительной статистики.

### *B. 4.2. Применение аппарата теории нечетких множеств*

Использование аппарата теории нечетких множеств удобно, когда нет требований к высокой точности оценок для принятия решений, а степень неопределенности имеющихся статистических данных можно учесть экспертными нечеткими оценками. Так, чуть упростив модель (1):

$$P_u = P_\infty - \left(P_\infty - P_0\right)\left(1 - \frac{a}{P_\infty}\right)^u,$$

удобно перейти к нечеткой модели:

$$\boldsymbol{P} = \left\{\left(P_u, \mu_P(P_u)\right)\right\},$$

где: $P_u$ – степень надежности программы в случае $u$-модификаций; $\mu_P\left(P_u\right)$ – функция принадлежности.

Указав уровень доверия $\alpha$, можно получить в общем виде и искомую интервальную нечеткую модель:

$$\boldsymbol{P} = \left\{P_m \mid \mu_M(m) \geq \alpha\right\}.$$

В основе построения нечеткой модели лежит определение ряда нечетких множеств, как-то:

- нечеткое множество $\boldsymbol{A} = \left\{\left(\mathrm{x}_i, \mu_A(\mathrm{x}_i)\right)\right\}$, представляющее собой совокупность упорядоченных пар модификаций программы – $x_i$ универсального множества $X$ и функций принадлежности $\mu_A(\mathrm{x}_i)$, характеризующих наличие модификации программ (доработки);
- нечеткое множество учитываемых $\boldsymbol{U} = \left\{\left(u, \mu_U(u)\right)\right\}$, где $\mu_U(u)$ – функция принадлежности, означающая степень уверенности в том, что число учитываемых модификаций равно $u$.

Данный подход, в том числе техники получения функций принадлежности, подробно описаны в [20].

## V. Заключение

Несмотря на многолетние изыскания в области тестирования, надежности и качества ПО, внедрение и развитие математических моделей испытаний остается востребованным, что в том числе определено рядом современных международных стандартов, касающихся как собственно методического обеспечения, так и процедур безопасного жизненного цикла программ. В настоящее время имеется представительное множество математических моделей, выбор и синтез которых зависит от конкретных ситуаций, включая оценку уровня надёжности ПО, разрабатываемого на базе современных и перспективных систем производства программ.

В работе показана возможность использования мягких вычислений для выраженных случаев имеющейся статистики о процессе испытаний современного ПО. Результаты, полученные с помощью методов мягких вычислений, могут быть использованы в качестве начальных при расчете параметров аналитических моделей.

Данное исследование соответствует целям разработки и внедрения линейки новых стандартов, гармонизированных национальному стандарту по безопасной разработке программ – ГОСТ Р 56939 [36, 37].

## Список литературы


[1] Barabanov A., Markov A. Modern Trends in The Regulatory Framework of the Information Security Compliance Assessment in Russia Based on Common Criteria // ACM International Conference Proceeding Series 8. Ser. "Proceedings of the 8th International Conference on Security of Information and Networks, SIN 2015" 2015. P. 30-33. DOI: 10.1145/2799979.2799980.

[2] Gokhale S.S., Marinos P.N., Trivedi K.S. Important milestones in software reliability modeling. In: Proc. of Software Engineering and Knowledge Engineering (SEKE 96), Lake Tahoe. NV, 1996. p. 345-352.

[3] Teyer T.A., Lipow M., Nelson E.C. Software Reliability. A Study of Large Project Reality. TRW Systems and Energy, Inc, 1978. 326 p.

[4] Anniprincy B., Sridhar S. Prediction of software reliability using COBB-Douglas model in SRGM. Journal of Theoretical and Applied Information Technology. 2014, vol. 62, no 2, pp. 355-363.

[5] Бубнов В.П., Сергеев С.А. Нестационарные модели локального сервера автоматизированной системы мониторинга искусственных сооружений // Труды СПИИРАН. 2016. № 2 (45). С. 102-115. DOI: 10.15622/sp.45.6.

[6] Danilov A.I., Khomonenko A.D., Danilov A.A. Dynamic software testing models. // Proceedings of International Conference on Soft Computing and Measurements, SCM 2015 18. 2015. P. 72-74. DOI: 10.1109/SCM.2015.7190414.

[7] Ivutin A.N., Larkin E.V., Perepelkin D.A. Software errors and reliability of embedded software. In: 2016 IEEE Conference on Quality Management, Transport and Information Security, Information Technologies (IT&MQ&IS), IEEE, 2016. p. 69-71. DOI: 10.1109/ITMQIS.2016.7751926.

[8] Kostogryzov A. Modeling software tools complex for evaluation of information systems operation quality (CEISOQ). // Lecture Notes in Computer Science. 2001, vol. 2052, pp. 90-101. DOI: 10.1007/3-540-45116-1_12.

[9] Krymsky V.G., Ivanov I.V. Application of Interval-Valued Probabilities and Unified Scheme of Non-Homogeneous Poisson Process Models to Software Failure Prognostics. In: Podofillini L, Sudret B, Stojadinovic B, Zio E, Kröger W, editors. Safety and Reliability of Complex Engineered Systems: ESREL 2015, CRC Press, 2015, pp. 2403–2411.

[10] Smagin V.A., Novikov A.N., Smagin S.Yu. A probabilistic model of the control of technical systems. Automatic Control and Computer Sciences. 2010, vol. 44, no 6, 324-329. DOI: 10.3103/S0146411610060027.

[11] Subburaj R. Software Reliability Engineering, McGraw Hill Education, 2014. 458 p.

[12] Tamura Y., Yamada S. Cost optimization based on decision-making and reliability modeling for big data on cloud computing. Communications in Dependability and Quality Management. 2015, vol. 18, no 4, pp. 5-19.

[13] Yamada S. Software Reliability Modeling: Fundamentals and Applications. Springer Japan, 2014. 90 p. DOI: 10.1007/978-4-431-54565-1

[14] Zeephongsekul P., Jayasinghe C.L., Fiondella L. Vidhyashree Nagaraju Maximum-Likelihood Estimation of Parameters of NHPP Software Reliability Models Using Expectation Conditional Maximization Algorithm. IEEE Transactions on Reliability. 2016, vol. 65, no 3, pp. 1571-1583. DOI: 10.1109/TR.2016.2570557.

[15] Zhao C., Qiu J., Liu G., Lv K. Planning, tracking and projecting method for testability growth based on in time correction. In: Proceedings of the Institution of Mechanical Engineers, Part O: Journal of Risk and Reliability. 2015, vol. 230, no 2, pp. 228-236.

[16] Рибер Г., Малмквист К., Щербаков А. Многоуровневый подход к оценке безопасности программных средств // Вопросы кибербезопасности. 2014. № 1 (2). С. 36-39. DOI: 10.21681/2311-3456-2014-2-36-39.

[17] Barabanov A.V., Markov A.S., Tsirlov V.L. Methodological Framework for Analysis and Synthesis of a Set of Secure Software Development Controls // Journal of Theoretical and Applied Information Technology, 2016, vol. 88, No 1, pp. 77-88.

[18] Kaswan K.S., Choudhary S., Sharma K. Software Reliability Modeling using Soft Computing Techniques: Critical Review. J Inform Tech Softw Eng. 2015, 5, 144. doi:10.4172/2165-7866.1000144.

[19] Maevsky D., Kharchenko V., Kolisnyk M., Maevskaya E. Software reliability models and assessment techniques review: Classification issues. In: 2017 9th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS), IEEE, 2017. p. 894–899. DOI: 10.1109/IDAACS.2017.8095216

[20] Markov A., Barabanov A., Tsirlov V. Models for Testing Modifiable Systems. In Book: Probabilistic Modeling in System Engineering, by ed. Andrey Kostogryzov. InTech, 2018.

[21] Andersson B., Persson M. Software Reliability Prediction – An Evaluation of a Novel Technique. SEBIT, 2004. 32 p.

[22] Kapur P.K., Pham H., Gupta A., Jha P.C. Software Reliability Assessment with OR Applications. London: Springer-Verlag, 2013. 548 p. DOI: 10.1007/978-0-85729-204-9.

[23] Tian J. Software Quality Engineering: Testing, Quality Assurance and Quantifiable Improvement. Wiley-IEEE Computer Society Press, 2005. 440 p.

[24] Xie M., Dai Y.-S., Poh K.-L. Computing Systems Reliability. Models and Analysis. Kluwer Academic Publishers, 2004. 293 p. DOI: 10.1007/b100619.

[25] Rana R., Staron M., Berger C., Hansson J., Nilsson M., Meding W. Analyzing defect inflow distribution and applying Bayesian inference method for software defect prediction in large software projects. Journal of Systems and Software. 2016, 117, 229-244. DOI: 10.1016/j.jss.2014.08.033

[26] Stieber H.A. Estimating the Total Number of Software Faults Reliability Models and Mutation Testing a Bayesian Approach. In: 2015 IEEE 39th Annual Computer Software and Applications Conference, IEEE, 2015, pp. 423-426. DOI: 10.1109/COMPSAC.2015.180

[27] Utkin L.V., Zatenko S.I., Coolen F.P.A. New interval Bayesian models for software reliability based on non-homogeneous Poisson processes. Automation and Remote Control. 2010, vol. 71, no 5, pp. 935-944. DOI: 10.1134/S0005117910050218.

[28] Wang L.J., Hu Q.P., Xie M. Bayesian analysis for NHPP-based software fault detection and correction processes. In: 2015 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM), IEEE, 2015, pp. 1046-1050.

[29] Bisi M., Goyal N.K. Artificial Neural Network Applications for Software Reliability Prediction (Performability Engineering Series). Wiley-Scrivener, 312 p.

[30] Марков Г.А. Эксперимент по использованию нейросетевых технологий для проведения испытаний открытого программного обеспечения // Вопросы кибербезопасности. 2013. № 2. С. 47-52. DOI: 10.21681/2311-3456-2013-2-47-52.

[31] Starodubtsev Yu.I., Grechishnikov E.V., Komolov D.V. Use of neural networks to ensure stability of communication networks in conditions of external impacts. Telecommunications and Radio Engineering. 2011. V. 70. N 14. P. 1263-1275.

[32] Junhong G., Xiaozong Y., Hongwei L. Software Reliability Nonlinear Modeling and Its Fuzzy Evaluation. In: 4th WSEAS Int. Conf. on Non-Linear Analysis, Non-Linear Systems and Chaos (NOLASC'05), ACM, 2005. pp. 49-54.

[33] Kumar R., Khatter K., Kalia A. Measuring software reliability: a fuzzy model. In: ACM SIGSOFT Software Engineering Notes. 2011, vol. 36, no 6, pp. 1-6. DOI: 10.1145/2047414.2047425.

[34] Vorobiev E.G., Petrenko S.A., Kovaleva I.V., Abrosimov I.K. Analysis of computer security incidents using fuzzy logic // Proceedings of 2017 20th IEEE International Conference on Soft Computing and Measurements, SCM 2017 20. 2017, pp. 369-371. DOI: 10.1109/SCM.2017.7970587.

[35] Vorobiev E.G., Petrenko S.A., Kovaleva I.V., Abrosimov I.K. Organization of the entrusted calculations in crucial objects of informatization under uncertainty // Proceedings of 2017 20th IEEE International Conference on Soft Computing and Measurements, SCM 2017 20. 2017, pp. 299 - 300. DOI: 10.1109/SCM.2017.7970566.

[36] Barabanov A., Markov A., Tsirlov V. Procedure for Substantiated Development of Measures to Design Secure Software for Automated Process Control Systems // 2016 International Siberian Conference on Control and Communications, SIBCON 2016 - Proceedings 2016. P. 7491660. DOI: 10.1109/SIBCON.2016.7491660.

[37] Barabanov A., Markov A., Fadin A., Tsirlov V., Shakhalov I. Synthesis of Secure Software Development Controls. // ACM International Conference Proceeding Series 8. Ser. "Proceedings of the 8th International Conference on Security of Information and Networks, SIN 2015" 2015. P. 93-97. DOI: 10.1145/2799979.2799998.