КОМПЬЮТЕРНОЕ МОДЕЛИРОВАНИЕ И АВТОМАТИЗАЦИЯ ПРОЕКТИРОВАНИЯ

УДК 004.413.4 Научно-исследовательская статья
DOI: 10.35330/1991-6639-2023-6-116-74-82
EDN: EYLUNJ

Тестирование программ и математическая модель поиска ошибок в программном комплексе

И. В. Кучумов

Компания «Яндекс», отдел разработок
119021, Россия, Москва, ул. Льва Толстого, 16

Аннотация. Тестирование программ актуально для аудита качества программы и его соответствия исходным спецификациям, требованиям надежности, функциональности, полноты комплекса и др. В последнее время актуален и учет соответствия программного продукта потребительским и рыночным условиям. Это требует новых подходов и методов, инструментов и технологий верификации и тестирования программ в реальном коде и реальном режиме эксплуатации. Цель работы – системный анализ тестирующих сред и моделирование процесса отладки, тестирования. С помощью общих системных методов (анализ-синтез, композиция-декомпозиция, моделирование и др.), математического моделирования получены следующие результаты: 1) проведен анализ целей, типов, методов тестирования; 2) проведена классификация методов; 3) при определенных исходных гипотезах относительно распределения ошибок в программном комплексе построена и исследована математическая модель оценивания количества ошибок (уязвимостей) в программной системе, их динамики с использованием аппарата класса обыкновенных дифференциальных уравнений «с насыщением». Предложены варианты развития постановок задач (гипотез), моделей, алгоритмов идентификации моделей для улучшения доказательности и охвата более широкого класса тестовых ситуаций. Результаты исследований можно использовать для практического аудита, управления процессом тестирования.

Ключевые слова: тестирование, надежность, анализ, программа, ошибки, математическая модель

Поступила 30.11.2023, одобрена после рецензирования 07.12.2023, принята к публикации 09.12.2023

Для цитирования. Кучумов И. В. Тестирование программ и математическая модель поиска ошибок в программном комплексе // Известия Кабардино-Балкарского научного центра РАН. 2023. № 6(116). С. 74–82. DOI: 10.35330/1991-6639-2023-6-116-74-82

MSC: 68N30 Research article

Software testing and mathematical error finding model

I.V. Kuchumov

Yandex company, development department
119021, Russia, Moscow, 16 Lev Tolstoy street

Abstract. Program testing is important to audit the quality of the program and its compliance with the initial specifications, reliability requirements, functionality, fullness of the complex, etc. Recently taking into account the compliance of the software product with consumer and market conditions is also relevant. This requires new approaches and methods, tools and technologies for verifying and testing programs in real code and real operation mode. This work is devoted to system analysis of testing environments and modeling of the debugging and testing process. Using general system methods

(analysis-synthesis, composition-decomposition, modeling, etc.), mathematical modeling the following results were obtained: 1) an analysis of goals, types, testing methods was carried out; 2) classification of methods was carried out; 3) with certain initial hypotheses regarding the distribution of errors in the software system, a mathematical model for estimating the number of errors (vulnerabilities) in the software system, their dynamics using the apparatus of the class of ordinary differential equations "with saturation" was built and investigated. There are presented variants for development of problem statements (hypotheses), models, algorithms for identification of models for improvement of evidence and coverage of a wider class of test situations. Research results can be used for practical audit, control of the testing process.

***Keywords**:* testing, reliability, analysis, program, errors, mathematical model

Submitted 30.11.2023, approved after reviewing 07.12.2023, accepted for publication 09.12.2023

For citation. Kuchumov I.V. Software testing and mathematical error finding model. *News of the Kabardino-Balkarian Scientific Center of RAS.* 2023. No. 6(116). Pp. 74–82. DOI: 10.35330/1991-6639-2023-6-116-74-82

Введение

Процессы валидации программных комплексов необходимы для выяснения соответствия комплекса и отдельных его модулей спецификациям технического задания**.** Аппарат, подход и методология верификации или тестирования программ используются разнообразные.

Верификация – процесс проверки программы на соответствие заданным спецификациям, проверки кода до запуска, на первых этапах разработки кода [1]. Тестирование – процедура проверки работоспособности уже написанного кода в реальной среде, ее надежности [2] и функциональной полноты.

Конечная цель верификации и валидации – установление степени соответствия программы (ПО) своему назначению. Тестировщик не делает акцент на исходном коде, его заботит надежность программ [3], особенно предназначенных для применения в системах автоматизации.

Цели и задачи исследования

Цели исследования – системный анализ процесса верификации, тестирования программных комплексов и классификация используемых при этом подходов, а также решение проблемы оценки динамики изменений количества ошибок в программном комплексе. Для полной верификации, проверки соответствия программы проектным требованиям (функциональным, интерфейсным, инфологическим, вычислительным и экономическим) необходимы формализация и алгебраические подходы. Поэтому проблема сводима к «приземленной» задаче тестирования или проверке функциональности, результативности и устойчивости программ по разработанным заранее тестировщиком тестов или контрольным задачам с известными решениями. Активно применяются и эвристики.

Методы исследования

Применяются методы системного анализа и синтеза, математического моделирования и статистики, методы таксономии, классификации. Для исследования надежности и тестирования программ выделены ключевые признаки (характеристики):

1) безотказность в течение времени «наработки на отказ»;

2) восстанавливаемость, способность комплекса восстанавливать работоспособность после сбоев;

3) развиваемость ПО (модулей, структур, интерфейса, управления);

4) параметризуемость, возможность реализации различных сценариев и режимов использования ПО лишь с помощью параметров.

Универсальных критериев оценки надежности программ нет, хотя такие оценки необходимы чаще априори и для прогноза. Пользуются часто эмпирическими (полуэмпирическими) моделями, эвристиками, экспертными процедурами идентификации важности и приоритетности различных факторов надежности. Например, для антивирусных пакетов применяют такие факторы (критерии), как «обновляемость», «срок лицензии» и др.

Результаты исследований

1. Системный анализ и классификация методов тестирования ПО

При тестировании используются:

1) модульный подход, тестирование отдельных модулей или классов функциональных объектов;

2) компонентный подход с тестированием модулей, объединенных для создания компонентов, подсистем ПО единым интерфейсом;

3) тестирование полностью интегрированного варианта, тестирование взаимодействий (инфологических связей) компонентов.

Тестирование проводится в ручном или автоматизированном режиме.

Есть и комбинаторные методы тестирования для охвата всевозможных взаимодействий, связей входных параметров, в частности, попарное или двухстороннее тестирование [4]. Используется часто и регрессионное тестирование с повторным запуском тестов.

Автоматизация тестирования позволяет:

1) минимизировать дублирование ситуаций тестового набора;

2) минимизировать время тестирования;

3) применять тестовые сценарии для различных ситуаций и программ;

4) оперативно модифицировать и расширить функций ПО;

5) реализовать «самотестирование» (тестирование программистом) и др.

Устойчивое решение с помощью программ целевых задач должно быть для пользователя достаточно интерактивным и комфортным, а его уязвимость для перехвата управления и иных рисков должна быть мала. Используют два принципиальных метода: теоретический, верифицирующий, доказательный и практический, на тестах.

Часто проводят «слабо» формальный подход к верификации – с использованием специфики верифицируемого объекта [5] или динамических статистических рядов [6].

Тестирование бывает функционально ориентированное и нефункциональное. Ключевая цель функционального тестирования – проверка соответствия ПО заявленным в ТЗ функциональным спецификациям, т.е. проверка выполнимости всех функций ПО, прямо или даже косвенно акцентированных в ТЗ. Функциональное тестирование базируется на методике, документе испытаний функционала (подробных сценариях реальных действий пользователя) ПО [7]. Программа и методика испытаний имитирует эксплуатацию продукта в реальном времени.

Нефункциональное тестирование ориентировано на оценку качества ПО, например, оценку времени, памяти или эргономики. Проверяется соответствие ПО нефункциональным сценариям и требованиям**,** например, совместимости, кросс-платформенности, единству интерфейса и др.

Экстремальное программирование, Scrum и другие методологии позволяют активно и полно вести аудит и анализ качества программы с помощью тестирования. С помощью сопряженного и массового программирования распространяются компетенции.

Тестирование интерфейсного и навигационного комфорта является также важным критерием надежности. Чем шире аудитория, тем внимательнее к пожеланиям пользователей программный продукт.

Конфигурационное тестирование приложения (для веб-приложения – тестирование кросс-платформенности) обеспечивает его работоспособность на различных платформах. Для конфигурационного тестирования нужно больше вычислительных ресурсов. При сложной архитектуре ПО используют интеграционное («сквозное») тестирование, но лишь после компонентного тестирования с использованием его результатов.

Тестирование безопасности выявляет уязвимости, риск-ситуации и возможный ущерб от действий злоумышленников при получении управления, доступа к данным. Объемное тестирование позволяет экспериментировать с предельными объемами обрабатываемой информации или с усилением контроля использования памяти [8].

Для каждого параметра (фактора) качества ПО применяют нагрузочное тестирование или проверку реагирования на внешние реакции, воздействия среды. Часто проверяется работа в экстремальных ситуациях – с ограничением аппаратных и архитектурных ресурсов или при максимальной нагрузке (стрессовое тестирование в диапазоне предельных возможностей) функционирования ПО. Здесь важно проверять устойчивость, надежность, способность ПО к самовосстановлению.

2. Моделирование поиска и оценки количества ошибок в программном комплексе

Для тестировщика, как и для программиста, важно знать динамические оценки количества ошибок в программе (потенциальных или оставшихся, текущих). Важно уметь сокращать время поиска латентных ошибок [9].

Многие модели надежности базируются на вероятностных гипотезах и законах распределения. Например, классическая модель Джелински – Моранда строится на гипотезах, согласно которым количество ошибок между последовательными моментами их обнаружения не меняется, они распределены в программе равновероятно и независимо.

Рассмотрим формализацию задачи поиска ошибок в программе (программном комплексе).

Пусть $u(x,t)$ – количество ошибок, выявленных в программном продукте к моменту времени t с участием x тестировщиков. Тогда можно записать следующую модель тестирования (класса систем «с насыщением»):

$$u_t(x,t) + u_x(x,t) = k\big(u_{max} - u(x,t)\big),$$

где k – коэффициент интенсивности поиска ошибок, u_{max} – максимальное количество ошибок в программном комплексе, которое заранее обычно неизвестно и подлежит идентификации или экспертной оценке, u_t, u_x – составляющие темпа поиска ошибок, количества ошибок, обнаруживаемых в единицу времени одним тестировщиком по времени и по количеству участвующих тестировщиков.

Задаются начальное и граничное условия:

$$u(x,0) = u_0(x), \qquad u(0,t) = u_1(t),$$
$$u(0,t) = u_1(t), \qquad u_0(0) = u_1(0),$$
$$0 \le x \le X, \qquad 0 \le t \le T.$$

В силу непрерывности $u(x,t)$ получаем $u_{max} = u(x_0, t_0)$, $(x_0; t_0) \in [0; X] \times [0; T]$.

Аналогично модели Джелински – Моранда функцию риска наличия ошибок можно задать в форме:

$$R(x,t) = k\big(u_{max} - u(x,t)\big),$$

$$R_{ij} = R(x_i, t_j) = k(u_{max} - u_{ij}), \qquad u_{ij} = u(x_i, t_j),$$

где k – экспертным или экспериментальным способом задаваемый параметр (масштабирующий параметр).

Рассмотрим для простоты логики рассуждений случай с одним тестировщиком: $m = 1$, $p = p(t)$, $u = u(t)$. Можно считать, что плотность вероятности отказа программного комплекса из-за наличия ошибок определяется по экспоненциальному закону:

$$p(x,t) = k(u_{max} - u(x,t))e^{-k(u_{max}-u(x,t))t}.$$

Учитывая плотность вероятности отказа ПО, функцию правдоподобия метода максимального правдоподобия или наименьших модулей (см., например, [10]) можно записать в следующем виде:

$$L = \prod_{i=1}^{n} p(t_i).$$

Или, логарифмируя с учетом $u_{max} \neq u_i$, $i = 1,2,\dots,n$, запишем функцию:

$$L = \sum_{i=1}^{n} ln(k(u_{max} - u_i)e^{-k(u_{max}-u_i)t_i} =$$

$$= \sum_{i=1}^{n} (ln(k(u_{max} - u_i)) - k(u_{max} - u_i)t_i).$$

Максимум этой функции ищем из системы уравнений:

$$\begin{cases} \dfrac{\partial L}{\partial u_{max}} = 0, \\ \dfrac{\partial L}{\partial k} = 0. \end{cases}$$

Вычисляя соответствующие выражения для производных, запишем систему соотношений вида:

$$\begin{cases} \displaystyle\sum_{i=1}^{n} \frac{1}{u_{max} - u_i} - k\sum_{i=1}^{n} t_i = 0, \\ \dfrac{n}{k} - u_{max}\displaystyle\sum_{i=1}^{n} t_i + \sum_{i=1}^{n} u_i t_i = 0. \end{cases}$$

Если ввести обозначение

$$\sum_{i=1}^{n} t_i = L,$$

то систему можно представить в виде:

$$\begin{cases} \displaystyle\sum_{i=1}^{n} \frac{1}{u_{max} - u_i} - kC = 0, \\ u_{max}C - \dfrac{n}{k} = \displaystyle\sum_{i=1}^{n} u_i t_i. \end{cases}$$

Из первого уравнения находим параметр

$$k = \frac{1}{C}\sum_{i=1}^{n}\frac{1}{u_{max} - u_i}.$$

Из второго уравнения после подстановки найденного выражения k получаем выражение:

$$u_{max}C - \frac{nC}{\sum_{i=1}^{n}\frac{1}{u_{max} - u_i}} = \sum_{i=1}^{n} u_i t_i$$

или

$$u_{max}C\sum_{i=1}^{n}\frac{1}{u_{max} - u_i} - \sum_{i=1}^{n} u_i t_i \cdot \sum_{i=1}^{n}\frac{1}{u_{max} - u_i} = nC.$$

Если обозначить

$$\varphi(u_{max}) = \sum_{i=1}^{n}\frac{1}{u_{max} - u_i},$$

то можно записать:

$$Cu_{max}\varphi(u_{max}) - \varphi(u_{max})\sum_{i=1}^{n} u_i t_i = nC.$$

Для этого трансцендентного уравнения можно применить метод итераций согласно нижеприведенной схеме:

$$u_{max}^{(n)} = \psi\left(u_{max}^{(n-1)}\right),\ \ n = 0,1,\dots;\ \ u_{max}^{(0)} = \alpha,\ \ 0 < \alpha < 1,$$

где

$$\psi(u_{max}) = \frac{1}{C}\left(\frac{nC}{\varphi(u_{max})} - \sum_{i=1}^{n} u_i t_i\right) =$$

$$= \frac{n}{\varphi(u_{max})} - \frac{1}{C}\sum_{i=1}^{n} u_i t_i\,.$$

Отметим, что решение u_{max} ищется целочисленное, поэтому при итерировании следует брать целочисленные значения $[u_{max}^{(n)}]$, т.е. значения функции антье от $u_{max}^{(n)}$.

Обсуждение

Предложенная модель и проведенный анализ позволяют реализовать ситуационное моделирование процесса тестирования, в частности, эксперимент нижеследующего типа.

По значениям $u_i(x_i)$ или величинам текущих ошибок в программе нужно идентифицировать максимальный объем оставшихся ошибок x_{imax}, $i = 1, 2, \dots, n$. Эта величина позволит настроить процесс тестирования по привлекаемым ресурсам, например, количеству тестировщиков. Например, для простоты возьмем случай одного тестировщика $u(x,t) = u(t)$. Благодаря непрерывности $u(t)$ найдем $u_{max} = u(t_{max}),\ t_{max} \in [t_0; T]$. Модель тестирования примет вид:

$$u'(t) = k\big(u_{max} - u(t)\big),$$

и его решение имеет вид:

$$u(t) = u_0 e^{-kt} + u_{max}(1 - e^{-kt}).$$

Подставляя $t = t_{max}$ получим:

$$u_{max} = u_0 e^{-kt_{max}} + u_{max}(1 - e^{-kt_{max}}).$$

Очевидное единственное решение – стационарное: $u_{max} = u_0$.

Отметим, что проведенное исследование можно развивать. В частности, интересно рассмотреть случай, когда в промежутке последовательных ошибок $[t_{i-1}; t_i], i = 2, \dots, n$ изменяется еще и параметр k, т.е. исследовать уравнение вида:

$$u_t(x,t) + u_x(x,t) = k(t)\big(u_{max} - u(x,t)\big).$$

Модель можно улучшать (математически, формально), усложняя гипотезы и совершенствуя идентификационный алгоритм.

Заключение

Тестирование, используя минимально достаточный тестовый набор, позволяет охватывать всевозможные ветви логики алгоритма, различные классы входных параметров, которые выделяются эмпирически или эвристически. Все значения, комбинации, естественно, невозможно охватить. Искусство тестировщика заключается в минимизации риска оставить какой-то класс и даже подкласс без внимания. Современные методы тестирования (попарное, экстремальное и др.) позволяют это реализовать. Тестирование проводится непрерывно, динамически может усложняться, поэтому моделирование и прогнозирование позволит облегчить этот сложный, но творческий процесс.

СПИСОК ЛИТЕРАТУРЫ

1. *Мерзлякова Е. Ю., Янченко Е. В.* Обзор методов верификации и оценки качества программного обеспечения // Вестник СибГУТИ. 2023. Т. 17. № 1. С. 92–106. DOI: 10.55648/1998-6920-2023-17-1-92-106

2. *Поначугин А. В.* Определение надежности программного обеспечения в структуре современной информационной системы // Кибернетика и программирование. 2019. № 2. С. 65–72. DOI:10.25136/2306-4196.2019.2.20341

3. *Лаврищева Е. М., Зеленов С. В., Пакулин Н. В.* Методы оценки надежности программных и технических систем // Труды ИСП РАН. 2019. Т. 31. № 5. С. 95–108. DOI: 10.15514/ISPRAS-2019-31(5)-7

4. *Шевчук В. И.* Попарное тестирование программного обеспечения // Universum: технические науки. 2023. № 7(112). DOI: 10.32743/UniTech.2023.112.7.15767

5. *Кашкевич А. М., Баданина Ю. В., Филимонов А. С., Долгих А. И.* Верификация программных комплексов на примере статически определимой призматической балки // Известия ВУЗов (сер. «Машиностроение»). 2023. № 5. С. 29–36. DOI:10.18698/0536-1044-2023-5-29-36

6. *Радионова Ю. А., Савкин А. Л.* Построение модели прогноза пиковых нагрузок рабочего процесса на основе анализа временного ряда // Автоматизация процессов управления. 2020. № 2(60). С. 53–60. DOI: 10.35752/1991-2927-2020-2-60-53-61

7. *Максимов М. И., Горина Е. А.* Agile-методология как драйвер эффективной корпоративной культуры // Региональная и отраслевая экономика. 2023. № 1. С. 102–111. DOI:10.47576/2949-1916_2023_1_102

8. *Самарин Н. Н.* Модель безопасного функционирования программного обеспечения, формализующая контроль использования памяти и обращений к ней процессора // Наукоемкие технологии в космических исследованиях Земли. 2021. Т. 13. № 1. С. 68–79. DOI:10.36724/2409-5419-2021-13-1-68-79

9. *Шакирова А. И., Хасьянов А. Ф., Даутов Э. Ф.* Сокращение времени тестирования программного обеспечения // Современные наукоемкие технологии. 2019. № 7. С. 104–109.

10. *Горяинов В. Б., Горяинова Е. Р.* Сравнение оценок максимального правдоподобия и наименьших модулей параметров процесса авторегрессии со случайными коэффициентами // Вестник МГТУ им. Н. Э. Баумана (сер. «Естественные науки»). 2015. № 3. С. 20–30.

REFERENCES

1. Merzlyakova E.Yu., Yanchenko E.V. Review of verification methods and software quality assessment. *Vestnik SibGUTI* [Bulletin of SibGUTI]. 2023. Vol. 17. No. 1. Pp. 92–106. DOI:10.55648/1998-6920-2023-17-1-92-106. (In Russian)

2. Ponachugin A.V. Software update in the structure of the modern information system. *Kibernetika i programmirovaniye* [Cybernetics and programming]. 2019. № 2. Pp. 65–72. DOI:10.25136/2306-4196.2019.2.203413. (In Russian)

3. Lavrishcheva E.M., Zelenov S.V., Pakulin N.V. Methods for assessing the reliability of software and technical systems. *Trudy ISP RAN* [Proceedings of ISP RAS]. 2019. Vol. 31. No. 5. Pp. 95–108. DOI: 10.15514/ISPRAS-2019-31(5)-7. (In Russian)

4. Shevchuk V.I. Pairwise testing of software. *Universum: tekhnicheskiye nauki* [Universum: Technical sciences]. 2023. No. 7(112). DOI: 10.32743/UniTech.2023.112.7.15767. (In Russian)

5. Kashkevich A.M., Badanina Yu.V., Filimonov A.S., Dolgikh A.I. Verification of software systems using the example of a statically determinate prismatic beam. *Izvestiya VUZov (ser. "Mashinostroyeniye")* [Higher education institutions news ("Mechanical Engineering")]. 2023. No. 5. Pp. 29–36. DOI:10.18698/0536-1044-2023-5-29-36. (In Russian)

6. Radionova Yu.A., Savkin A.L. Construction of a model for forecasting peak loads of a working process based on time series analysis. *Avtomatizatsiya protsessov upravleniya* [Automation of control processes]. 2020. No. 2(60). Pp. 53–60. DOI: 10.35752/1991-2927-2020-2-60-53-61. (In Russian)

7. Maksimov M.I., Gorina E.A. Agile methodology as a driver of effective corporate culture. *Regional'naya i otraslevaya ekonomika* [Regional and industrial economics]. 2023. No. 1. Pp. 102–111. DOI:10.47576/2949-1916_2023_1_102. (In Russian)

8. Samarin N.N. A model for the safe functioning of software that formalizes control of memory use and processor access to it. *Naukoyemkiye tekhnologii v kosmicheskikh issledovaniyakh Zemli* [Science-intensive technologies in space research of the Earth]. 2021. Vol. 13. No. 1. Pp. 68–79. DOI:10.36724/2409-5419-2021-13-1-68-79. (In Russian)

9. Shakirova A.I., Khasyanov A.F., Dautov E.F. Reducing the time of software testing. *Sovremennyye naukoyemkiye tekhnologii* [Modern science-intensive technologies]. 2019. No. 7. Pp. 104–109. (In Russian)

10. Goryainov V.B., Goryainova E.R. Comparison of maximum likelihood estimates and least modules of autoregression process parameters with random coefficients. *Vestnik MGTU im. N.E. Baumana (ser. "Yestestvennyye nauki")* [Bulletin of the Bauman Moscow State Technical University (ser. "Natural Sciences")]. 2015. No. 3. Pp. 20–30. (In Russian)

Информация об авторе

Кучумов Илья Вадимович, руководитель отдела разработки, компания «Яндекс»;
119021, Россия, Москва, ул. Льва Толстого, 16;
kuchumov.ilya@gmail.com; ORCID: https://orcid.org/0009-0003-6470-5587

Information about the author

Kuchumov Ilya Vadimovich, Head of the Development Department, Yandex company;
119021, Russia, Moscow, 16 Lev Tolstoy street;
kuchumov.ilya@gmail.com; ORCID: https://orcid.org/0009-0003-6470-5587