УДК 004.9
DOI: 10.26102/2310-6018/2021.35.4.016

Корреляция отказов как основа применения модели Маркова для тестирования программного обеспечения

М.М. Зозуля[1], О.Я. Кравец[2]✉

[1]Военный учебно-научный центр Военно-воздушных сил «Военно-воздушная академия имени профессора Н.Е. Жуковского и Ю.А. Гагарина», Воронеж, Российская Федерация
[2]Воронежский государственный технический университет, Воронеж, Российская Федерация
csit@bk.ru✉

Резюме. Анализ существующих результатов исследований тестирования отказов, сбоев программного обеспечения во время тестирования должен учитывать актуальность программного обеспечения для тестирования отказов с использованием цепей Маркова с правом тестирования модели, разработку многоцелевого алгоритма оценки заданной цепи Маркова с правильной стратегией тестирования на основе отказов, связанных со стратегией перехода состояния на основе матрицы весов многоцелевого теста. Целью исследования является разработка набора оптимизирующих стратегий тестирования отказов программного обеспечения на основе учета корреляции связанных отказов и управляемых цепей Маркова. В данной работе на основе модели тестирования контролируемой цепи Маркова, основанной на корреляционных отказах, предложена модель тестирования контролируемой цепи Маркова, в основном для решения проблемы тестирования программного обеспечения в ситуации взаимосвязи отказов программного обеспечения. Связь между программными модулями определяется количественно для расчета многоцелевой матрицы переноса и оценки взаимосвязи связанных отказов. В интегрированной среде разработки Java Eclipse загружается CDT проекта с открытым исходным кодом, для реализации которого используется Java, а в среде Eclipse используются процедуры модульного тестирования с использованием JUNIT для разработки. Результаты эксперимента показывают, что предложенная стратегия по сравнению со стратегией тестирования управляемой цепи Маркова может значительно сократить количество тестовых случаев и повысить скорость обнаружения отказов.

Ключевые слова: тестирование отказов, программное обеспечение, управляемая модель Маркова, матрица переноса, матрица весов

Для цитирования: Зозуля М.М., Кравец О.Я. Корреляция отказов как основа применения модели Маркова для тестирования программного обеспечения. *Моделирование, оптимизация и информационные технологии.* 2021;9(4). Доступно по: https://moitvivt.ru/ru/journal/pdf?id=1098
DOI: 10.26102/2310-6018/2021.35.4.016

Failure correlation as a basis for applying the Markov model for software testing

M.M. Zozulya[1], O.Ja. Kravets[2]✉

[1]Military educational scientific center air force "air force Academy named after Professor N. E. Zhukovsky and Y. A. Gagarin», Voronezh, Russian Federation
[2]Voronezh state technical university, Voronezh, Russian Federation
csit@bk.ru✉

Abstract: The analysis of existing research results of testing failures, software failures during testing should take into account the relevance of software for testing failures using Markov chains with the right to test the model, the development of a multi-purpose algorithm for evaluating a given Markov chain with the correct testing strategy based on failures associated with a state transition strategy based on a matrix of weights of a multi-purpose test. The study aims to develop a set of optimizing software failure testing strategies based on the related failures correlation and controlled Markov chains. In this paper, based on the Markov controlled chain testing model based on correlation failures, a Markov model is proposed, mainly to solve the problem of software testing in a situation of software failures interconnection. The relationship between software modules is quantified to calculate a multi-purpose transfer matrix and assess the interrelationship of associated failures. In the Eclipse Java Integrated Development Environment, the CDT of an open-source project is loaded, for which Java is used for implementation, and in the Eclipse environment, unit testing procedures are used using JUNIT for development. The results show that this strategy, compared with the Markov controlled chain testing strategy, can significantly reduce the number of test cases and increase the speed of failure detection.

Keywords: failure testing, software, controlled Markov model, transfer matrix, weight matrix

For citation: Zozulya M.M., Kravets O.Y. Failure correlation as a basis for applying the Markov model for software testing. *Modeling, Optimization and Information Technology*. 2021;9(4). Available from: https://moitvivt.ru/ru/journal/pdf?id=1098 DOI: 10.26102/2310-6018/2021.35.4.016 (In Russ).

Введение

В последние годы, с постоянным расширением спектра программных приложений, масштабы растут, структура становится все более сложной. Компьютерные приложения с программным обеспечением в качестве ядра проникли во все области, включая промышленность, сельское хозяйство, оборону, образование, экономику, повседневную жизнь и работу людей, и играют все более важную роль. Проблемы надежности программного обеспечения [1] становятся все более и более заметными. Хотя существует множество методов и методик, которые были предложены и приняты для повышения качества программного обеспечения при проектировании программных систем (таких как оценка архитектуры программного обеспечения [2]), тестирование программного обеспечения по-прежнему является основным средством обеспечения его качества и надежности.

В настоящее время процесс тестирования программного обеспечения требует больших затрат на разработку в большинстве компаний-разработчиков программного обеспечения. Исследования показывают, что усилия по тестированию программного обеспечения часто составляют более 40 % от общей рабочей нагрузки при разработке программного обеспечения. Для некоторых программ, требующих высокой надежности и высокой безопасности (например, аэрокосмических, военных и т. д.), стоимость тестирования эквивалентна 80 % всего жизненного цикла программного обеспечения от общей стоимости проекта. Кроме того, стоимость, которая используется для исправления отказов программного обеспечения, будет расти с увеличением времени, необходимого для обнаружения отказов. Однако, тестирование программного обеспечения по-прежнему является относительно небольшой частью исследований в процессе разработки программного обеспечения.

Для эффективного повышения эффективности тестирования программного обеспечения многие исследователи выдвигают новые методы тестирования, основанные на автоматической генерации [3], приоритете случаев тестирования [4] и локализации отказов [5], а также оптимизации стратегии тестирования и др., но большинство этих

исследований основаны на предположении, что отказы независимы, то есть взаимосвязь между отказами игнорируется. В процессе фактического тестирования большое количество исследований показало, что многие отказы программного обеспечения не являются независимыми друг от друга. Между ними существует определенная взаимосвязь, то есть корреляция сбоев [6]. Отмечается [7], что программное обеспечение для космических приложений содержит в общей сложности 36 отказов, из которых частота их обнаружения практически равна нулю, если существуют только три отказа D1, D2 и D32. Они могут быть обнаружены только в случае наличия других сопутствующих отказов. Поэтому следует в полной мере использовать корреляционную информацию между отказами, что может помочь повысить эффективность тестирования программного обеспечения.

Корреляция отказов

Хотя компании-разработчики программного обеспечения стремятся разрабатывать продукты в стандарте высокой согласованности и низкой связи, но для большинства программ они все еще не могут делать абсолютно независимые модули. Взаимосвязь между отказами может проявляться в потоке управления или в потоке данных программы, оба из которых могут рассматриваться как неотъемлемое представление отказов и связанных с ними отказов. Кроме того, присущие программистам стили при программировании приведут к отказам со сходством [8]. Это связано с тем, что существуют поток управления и поток данных, которые вызывают наличие взаимосвязи между ними.

Что касается отказов корреляции, в [9] отмечено, что в реальном процессе тестирования программного обеспечения сбои программного обеспечения не являются независимыми. В [9] предложен способ использования обновленных марковских моделей для моделирования надежности программного обеспечения, связанного с отказами. Некоторые эксперты [10] предложили бинарную модель марковского процесса на основе теста, связанного с раундом. Основываясь на предположении о взаимном вмешательстве до и после теста, время рассматривалось как основа для взаимосвязи корреляции отказов, что в определенной степени объясняло причины возникновения корреляции отказов. В [11] проведены эмпирические исследования обнаруженных отказов в процессе разработки модулей. Отказы программного обеспечения интерпретировались как модель контекстно-зависимой машины и предполагалось, что взаимосвязь между сбоями программного обеспечения может быть объяснена эффектом защиты от сбоев. В [6] предложена модель надежности P-NHPP (Фазово-неоднородный процесс Пуассона). Предполагалось, что, если преждевременно отказаться от отдельного отказа корреляции, это скроет возможность обнаружения других связанных отказов, что является одной из причин сбоя программного обеспечения. Это повлияет на результаты оценки модели надежности программного обеспечения. Для обнаружения отказа корреляции в [12] проанализирована корреляция сбоев с точки зрения отказов программного обеспечения. Продемонстрирована взаимосвязь между отказами на примерах программного обеспечения в космических приложениях, и доказано, что связанные отказы не соответствуют закону обмена и ассоциативному закону, а также применены корреляционные отказы в процессе тестирования программного обеспечения. Там же предложен тестовый метод возврата обнаруженного отказа для устранения корреляции отказов. Результаты моделирования показывают, что этот метод может эффективно обнаруживать отказы корреляции, существующие в программном обеспечении, но метод возврата отказов окажет определенное влияние на эффективность тестирования.

Как указано в [13], тестовые примеры, охватывающие одни и те же или аналогичные требования к тестированию, как правило, обнаруживают одни и те же или аналогичные отказы программного обеспечения. Таким образом, полное использование соответствующей информации между отказами облегчит поиск распространенных отказов тестировщиками. Однако стратегия оптимизации тестирования программного обеспечения, предназначенная для устранения корреляции отказов, редко изучалась. Кроме того, в этих исследованиях не было количественной оценки связанных с этим отказов. **Целью исследования** является разработка набора оптимизирующих стратегий тестирования отказов программного обеспечения на основе учета корреляции связанных отказов и управляемых цепей Маркова.

Модели управляемых цепей Маркова

Модель Маркова – это статистическая модель, которая характеризуется стохастическими процессами дискретного времени. В этом процессе, при условии предоставления текущих знаний или информации, не имеет значения учитывать статус прошлого для прогнозирования будущего состояния.

Модель цепи Маркова применяется для тестирования программного обеспечения. Количество отказов, присутствующих в системе тестируемого программного обеспечения, рассматривается как состояние модели. Предположим, что в системе тестируемого программного обеспечения имеется N отказов, желательно обнаружить все отказы программных систем, то есть в идеальной ситуации; количество отказов в тестируемых программных системах равно нулю. Состояние с N отказами рассматривается как начальное состояние модели цепи Маркова, а состояние без отказов рассматривается как завершение состояния. Под влиянием тестовых случаев определяется таблица переходов состояний модели цепи Маркова. Как только таблица переходов состояний будет определена, впоследствии будет определена цепочка Маркова.

Существующие стратегии адаптивного тестирования в основном основаны на теории марковских решений, а их структуры управляются тестовыми моделями цепи Маркова. При определенных условиях [7-11] эти модели делают слишком идеалистическую обработку, поэтому есть некоторые недостатки, в том числе:

1. Предполагается, что количество отказов, содержащихся в тестируемом программном обеспечении, является определенным. Однако в ходе фактического испытания количество отказов неизвестно. В ходе тестового мероприятия невозможно обнаружить все отказы. Необходимо регулярно оптимизировать тест, а затем проверять еще раз. Повторная оптимизация операции обнаружения тестовых случаев может повторяться много раз. Хотя каждый раз количество обнаруженных отказов изменчиво и ограничено, существует желаемое значение. Желаемое значение может быть использовано для измерения эффекта теста. Итак, как найти способ получить желаемое количество отказов – это ключ к исследованию.

2. Обычно предполагают, что обнаружена равная вероятность различных отказов программного обеспечения. Но это не так. Скорость обнаружения отказов связана не только с выбранными решениями, но и со свойствами самого отказа. Поэтому необходимо найти разумный способ количественной оценки и расчета вероятностей различных обнаруженных отказов.

3. Предполагается, что при обнаружении отказа он немедленно удаляется без внесения новых отказов. В ходе фактического тестирования установлено, что преждевременное устранение обнаруженных отказов может не только привести к появлению новых отказов, но и может привести к сбою программного обеспечения, что

повлияет на результаты оценки модели надежности программного обеспечения [1].

4. Предполагается, что решение об устранении отказов не потребляет системных ресурсов (т. е. затрат).

5. Предполагается, что затраты равны для устранения различных отказов. Однако возможная оплата любого решения не имеет ничего общего с состоянием программного обеспечения. Это связано только с самим соответствующим сбоем программного обеспечения. Обнаружение серьезного сбоя программного обеспечения явно более полезно, чем обнаружение незначительного сбоя программного обеспечения. Поэтому необходимо проанализировать особенности неисправностей отказов программного обеспечения, обнаруженных в процессе тестирования программного обеспечения [13]. В соответствии с серьезностью отказы классифицируются (такие как фатальные, серьезные, общие, вторичные, предложения или подсказки для тестирования и т. д.), и вес всех типов отказов разумно распределен. Чем серьезнее недостатки, тем больше вес, и соответствующая скидка больше.

Исследования, указанные выше, в основном описываются упрощенной моделью цепи Маркова. Модели основаны на предположении, что отказы независимы, игнорируя влияние корреляционного отказа на процесс тестирования, поэтому оценка надежности программного обеспечения приводит к искажениям.

Управляемая марковская модель, связанная с отказами при тестировании программного обеспечения

Как показано на Рисунке 1, в традиционной модели тестирования управляемой цепи Маркова (CMC) состояние тестовой модели [14] определяется следующим образом:

$S = \{s_t\} = \{M, M-1,..., 1,0\}$ $(t \geq 0)$,

где s_t относится к оставшемуся состоянию отказа программного обеспечения в течение определенного времени t.

На Рисунке 1 a_t относится к моменту времени t, когда выбирается действие тестового случая. В этой модели рассматривается только взаимосвязь между скоростью передачи состояния программного обеспечения и скоростью обнаружения отказов, но игнорируется корреляция между отказами. Чтобы воспользоваться информацией, связанной с отказами, для повышения эффективности тестирования, расширения модели условий обнаружения связанных отказов, вводятся многоцелевые веса на основе отказов, поэтому модифицированная тестовая модель управляемой цепи Маркова на основе связанных отказов (CDCMC) показана на Рисунке 2, где w_i является правильным весом для выполнения определенного действия, то есть на конкретный момент времени это вероятность выбора действия, предпринятого тестовыми случаями.

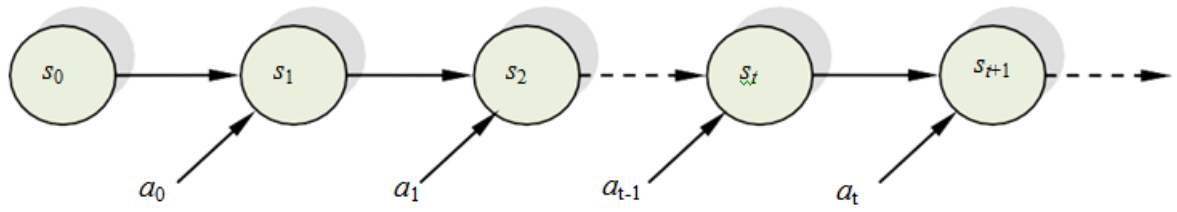

Рисунок 1 – Классическая модель управляемой цепи Маркова
Figure 1 – Classical model of a controlled Markov chain

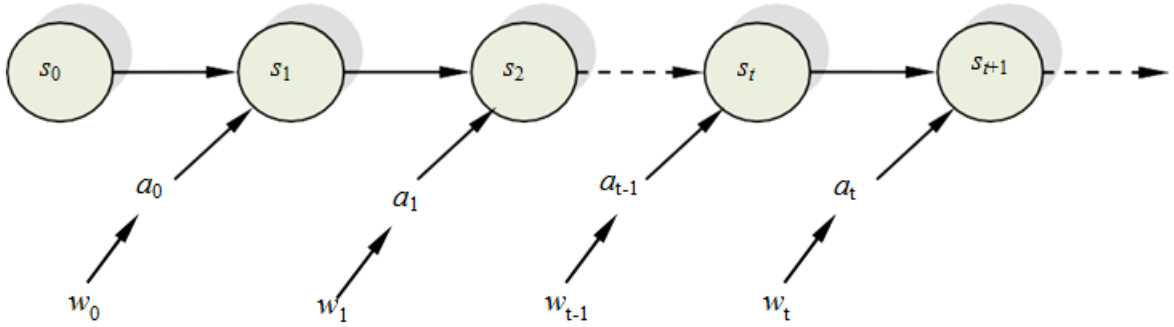

Рисунок 2 – Модифицированная модель управляемой цепи Маркова
Figure 2 – Modified model of a controlled Markov chain

Для построения многоцелевых весов в тестовой модели и улучшения условий перехода состояния программного обеспечения функция степени связи Связь (x, y, тип) на основе взвешенного модуля предназначена для создания формальных описаний связи между модулями, параметры x и y которых указывают код модуля, а тип указывает способ связи. Функция анализирует связь между модулями, чтобы вывести взаимосвязь между отказами, которые могут существовать в этих модулях, из которых возвращаемое значение функции рассматривается как вес связи между модулями. Рассматривая теорему Парето применительно к тестированию программного обеспечения [15], установлено, что 80 % ошибок, вероятно, были вызваны долей 20 % модулей в тестах. Поэтому при x=y функция возвращает значение 0,5, то есть, когда в определенном модуле обнаруживаются отказы, в следующий раз вероятность продолжения тестирования в модуле составляет 50 %, чтобы обеспечить решения по тестированию стандартного перехода состояния программного обеспечения. Таким образом, взаимосвязь между модулями может быть представлена в виде матрицы R:

$$R = \begin{bmatrix} r_{11} & r_{21} & \dots & r_{1i} & \dots & r_{1n} \\ r_{21} & r_{22} & \dots & r_{2i} & \dots & r_{2n} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ r_{i1} & r_{i2} & \dots & r_{ii} & \dots & r_{in} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ r_{n1} & r_{n2} & \dots & r_{ni} & \dots & r_{nn} \end{bmatrix} \quad (1)$$

Здесь n – количество модулей, содержащихся в тестируемом программном обеспечении, а r_{ij} – вес связи между i-м модулем и j-м модулем. Очевидно, что в соответствии с предыдущими соображениями, когда i равно j, $r_{ij}=0.5$, т.е. $r_{11} = r_{22} =... = r_{nn} = 0.5$.

Для построения многоцелевых весов W сначала строится весовая функция:

$$weight(r_{ij}, \rho_i^j, \theta_i^j) \quad (2)$$

Здесь r_{ij} – вес связи между i-м модулем и j-м модулем; ρ_i^j – расходы, связанные с тем, что после обнаружения отказа в i-м модуле обнаруживается отказ в j-м модуле; θ_i^j – коэффициент обнаружения отказов, который после обнаружения отказа в i-м модуле обнаруживает отказ в j-м модуле. Возвращаемое значение w_j функции – вероятность

обнаружения отказа в j-м модуле для выполнения соответствующих тестовых случаев. В соответствии с возвращаемым значением можно скорректировать весовую матрицу W большего количества целей для коррекции стратегии тестирования:

$$W= [w_0,w_1,...,w_j,...,w_t]$$

Для реализации стратегии необходимо собрать исторические данные, и они будут храниться в базе данных. Используя тестовую обратную связь для выполнения онлайн-оценки параметров и обновления матрицы веса нескольких целей W на основе функции связи взвешенного модуля, можем построить более точную и корректную модель (Рисунок 3).

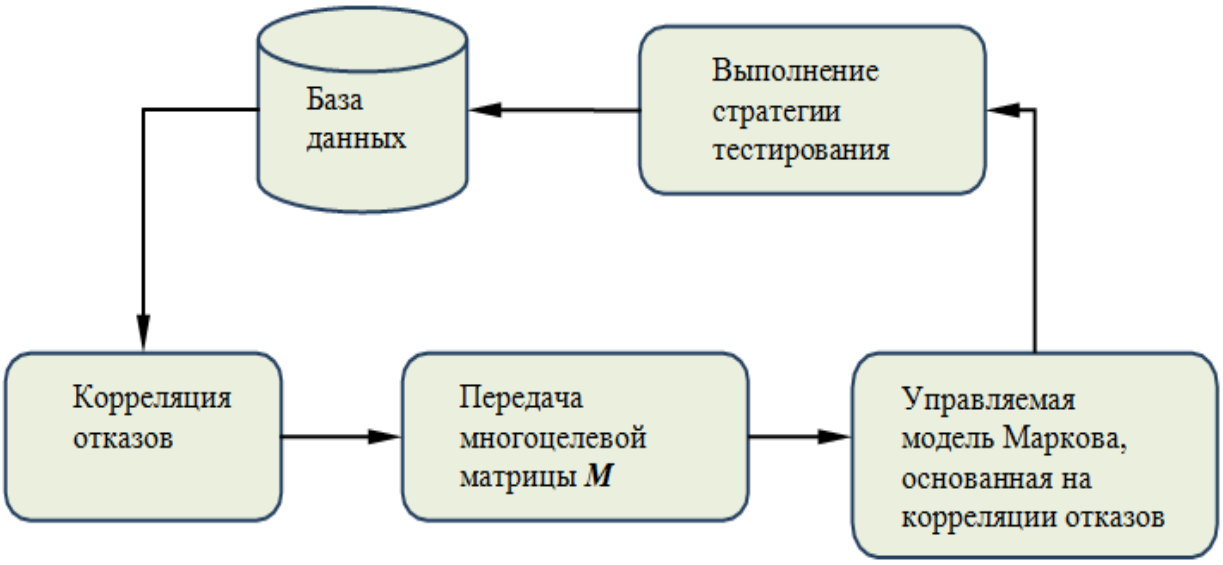

Рисунок 3 – Динамическая коррекция корреляционной связи отказов
Figure 3 – Dynamic correction of the correlation of failures

Экспериментальное исследование

Эксперименты были проведены в следующих условиях. Операционная система: Windows Ultimate, 64-разрядная версия. Процессор Intel ® Core ™ i5-2410M с частотой 2,30 ГГц. Объем памяти: 8,00 ГБ.

В интегрированной среде разработки Java Eclipse загружается CDT проекта с открытым исходным кодом, для реализации которого используется Java, а в среде Eclipse используются процедуры модульного тестирования с использованием JUNIT для разработки. Использованы инструменты тестирования программного обеспечения Eclipse с открытым исходным кодом plug EclEmma для проверки покрытия CDT, и тестовые данные будут динамически анализироваться в тесте. Различные тестовые случаи могут быть загружены отдельно для тестирования в соответствии с потребностями, и результаты нескольких тестов могут быть объединены [16].

В указанных средах, чтобы проверить эффективность модели тестирования программного обеспечения CDCMC, сравниваем модель со стратегией тестирования программного обеспечения CMC. Было проведено 50 тестов. Установлено, что оба метода обнаруживают одинаковое количество отказов в конкретном тесте, всего 39. Экспериментальные данные до и после эксперимента в общей сложности 10 раз сравниваются с двумя стратегиями тестирования, результаты показаны на Рисунке 4. Количество тестовых случаев было использовано в двух стратегиях, когда было обнаружено 39 отказов, как показано на Рисунке 5.

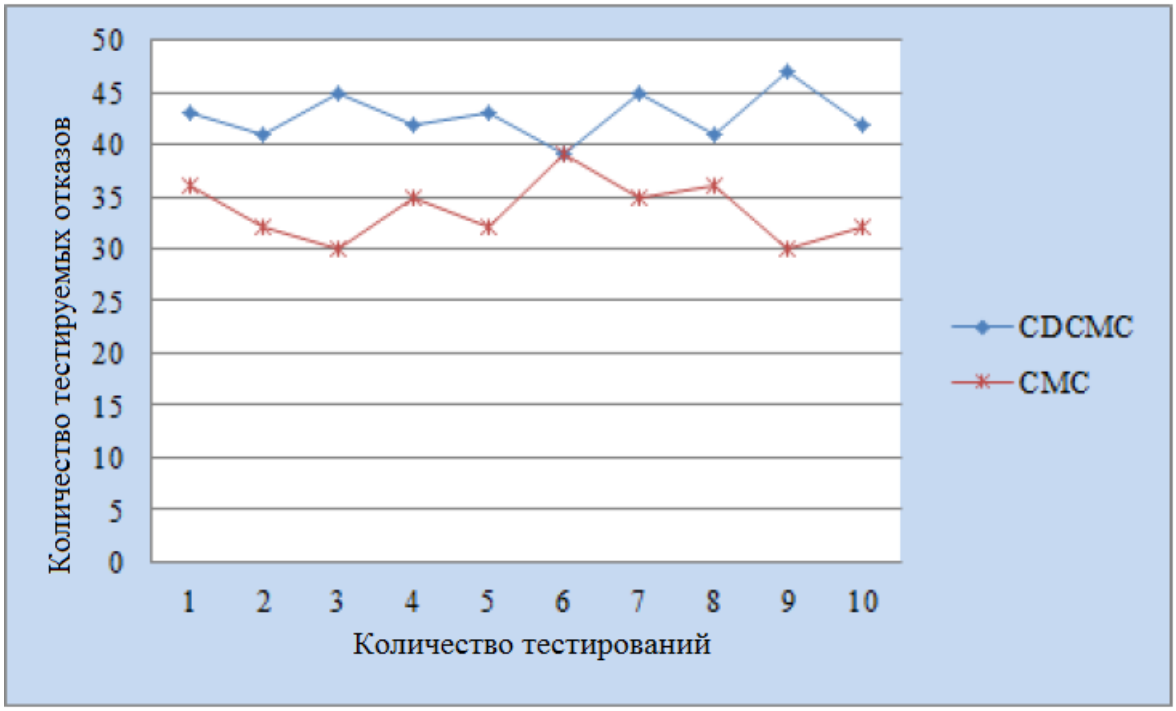

Рисунок 4 – Сравнение двух стратегий тестирования
Figure 4 – Comparison of two testing strategies

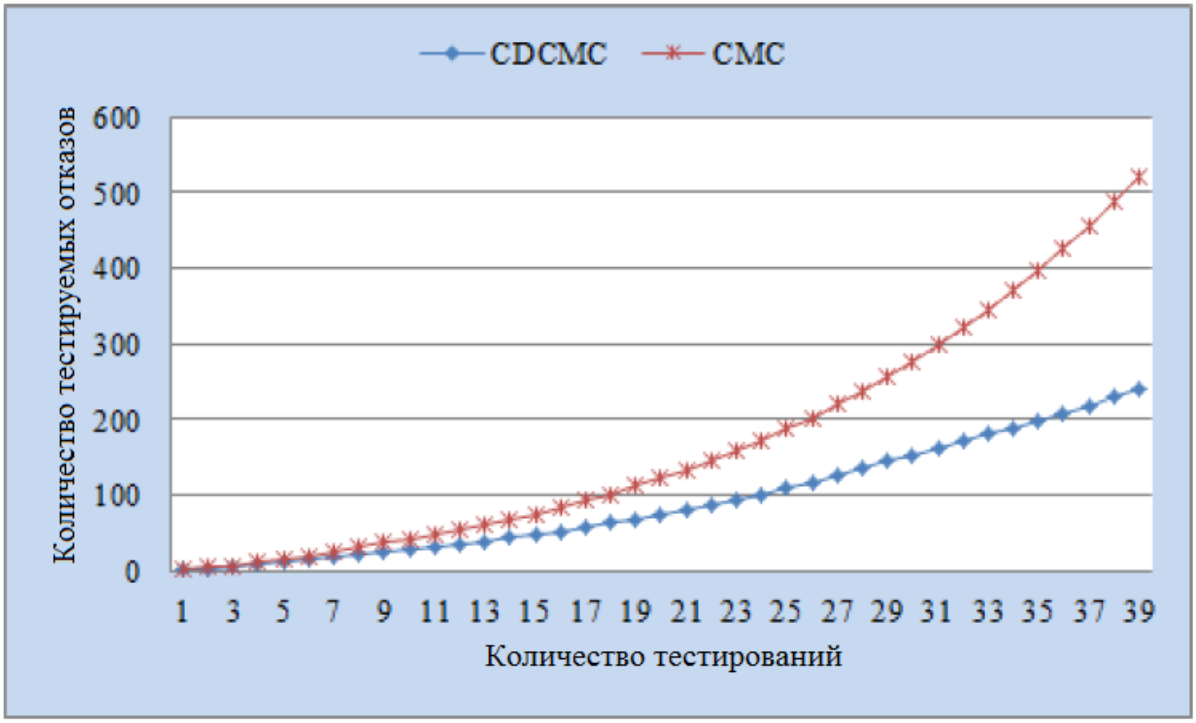

Рисунок 5 – Взаимосвязь между числом обнаруженных отказов и числом тестов
Figure 5 – Relationship between the number of detected failures and the number of tests

В стратегии тестирования программного обеспечения CMC с увеличением числа обнаруженных отказов количество тестовых случаев увеличивается очень быстро, почти экспоненциально. По сравнению с ней стратегия тестирования программного обеспечения CDCMC имеет тенденцию к более медленному росту, подробные данные

приведены в Таблице 1. Как видно из Таблицы 1, в начале теста нет существенных различий между количеством тестовых случаев, в которых нуждаются две стратегии, но с увеличением числа обнаруженных отказов разница в количестве тестовых случаев, в которых нуждаются две стратегии, растет. Как видно из соотношения количества тестовых случаев и количества обнаруженных отказов, стратегия тестирования программного обеспечения CDCMC составляет 6,19, в то время как стратегия тестирования программного обеспечения CMC составляет 13,38. Принимая во внимание 50 тестов, первый составляет 6,21, а второй – 13,51. Таким образом, частота обнаружения отказов в стратегии тестирования программного обеспечения CDCMC лучше, чем в стратегии тестирования программного обеспечения CMC.

Таблица 1 – Сравнение стратегий тестирования
Table 1 – Comparison of testing strategies

Число отказов	CDCMC	CMC	Число отказов	CDCMC	CMC	Число отказов	CDCMC	CMC
1	2	3	14	44	68	27	127	220
2	4	6	15	48	76	28	136	237
3	7	7	16	53	84	29	145	256
4	9	11	17	58	93	30	154	276
5	12	15	18	63	102	31	163	298
6	15	20	19	69	112	32	172	321
7	18	27	20	75	122	33	181	345
8	21	32	21	81	133	34	190	370
9	25	37	22	88	145	35	199	397
10	28	42	23	95	158	36	209	425
11	32	48	24	102	172	37	219	456
12	36	54	25	110	187	38	230	487
13	40	61	26	118	203	39	241	522
Число тестов/Число обнаруженных отказов						6.18	CMC	13.38

Заключение

В данной работе на основе модели тестирования управляемой цепи Маркова, основанной на корреляционных отказах, предложена модель тестирования управляемой цепи Маркова, в основном для решения проблемы тестирования программного обеспечения в ситуации взаимосвязи отказов программного обеспечения. Связь между программными модулями определяется количественно для расчета многоцелевой матрицы переноса и оценки взаимосвязи связанных отказов. Объединяя исторические данные для настройки многоцелевой матрицы передачи в режиме онлайн, матрица перехода используется в качестве стандарта передачи стратегий тестирования программного обеспечения. Анализируя результаты, можно видеть, что стратегия тестирования CDCMC более эффективна, чем стратегия тестирования CMC.

СПИСОК ИСТОЧНИКОВ

1. Kai-Yuan C., Zhao D., Ke L. On Several Issues in Software Reliability Testing. *Chinese Journalof Engineering Mathematics*. 2008;25(6):967–978.
2. Li Z., Hui G., Shou-Xin W. Software architecture evaluation. *Journal of Software*.

2008;19(6):1328−1339.
3. Jun-Hao H., Chun H., Zhu-Lin Z. Automatic System Testing Test Case Generation Based on UML. *Computer Systems & Applications*. 2011;20(2):178–181.
4. Bo Q., Chang-Hai N., Bao-Wen X. Test Case Prioritization Based on Test Suite Design Information. *Chinese Journal of Computer*. 2008;31(3):431–439.
5. Wei L., Zheng Z., Peng H. et al. Predicate Execution-Sequence Based Fault Localization Algorithm. *Chinese Journal of Computer*. 2013;36(12):2406–2419.
6. Gao-Chao X., Xin-Zhong L., Liang H., Xiao-Dong F., Yu-Shuang D. Software Reliability Assessment Models Incorporating Software Defect Correlation. *Journal of Software*. 2011;22(3):439–450.
7. Rothermel G., Untch R.H., Harrold M.T. Prioritizing test cases for regression testing. *IEEE Trans. Software Engineering*. 2001;27:929–948.
8. Jian Z., Hong-Yu Z., David L. Where should the Bugs be fixed. *Proc. of the International Conference on Software Engineering*. Zurich: IEEE Computer Society. 2012;14–24.
9. Katerina G.P., Trivedi K.S. Failure correlation in software reliability models. *IEEE Trans. on Reliability*. 2001;49(1):37–48.
10. Chen S., Mills S. A binary Markov process model for random testing. *IEEE Trans. on Software Engineering*. 1996;22(3):218–223.
11. Bishop P.G., Pullen F.D. PODS revisited-A study of software failure behaviour. *Proc. of the IEEE International Symposium On Fault Tolerant Computing*. 1998;2–8.
12. Tao J., Chang-Hai J., De-Bin H., Cheng-Gang B., Kai-Yuan C. An Approach for Detecting Correlated Software Defects. *Journal of Software*. 2005;18(1):17–28.
13. Chang-Ai S. A Constraint-Based Approach to Identifying and Analyzing Failure-Causing Regions. *Journal of Software*. 2012;23(7):1688−1701.
14. De-Ping Z., Chang-Hai N., Bao-Wen X. Cross-Entropy Method Based on Markov Decision Process for Optimal Software Testing. *Journal of Software*. 2008;19(10):2770–2779.
15. Логинова И.А., Дмитренко Ю.А., Будулуца А.Р., Мелихов С.А., Жалялетдинова Э.И., Шамукова Д.Р. Применение анализа Парето для обеспечения качества электронных средств. *Приднепровский научный вестник*. 2019;3(4):52–55.
16. Jin-Xia A., Guo-Qing W., Shu-Fang L., Ji-Hong Z. Dynamic Evaluation Method Based Multi-Dimensional Test Coverage for Software Testing. *Journal of Software*. 2010;21(9):2135–2147.

REFERENCES

1. Kai-Yuan C., Zhao D., Ke L. On Several Issues in Software Reliability Testing. *Chinese Journalof Engineering Mathematics*. 2008;25(6):967–978.
2. Li Z., Hui G., Shou-Xin W. Software architecture evaluation. *Journal of Software*. 2008;19(6):1328−1339.
3. Jun-Hao H., Chun H., Zhu-Lin Z. Automatic System Testing Test Case Generation Based on UML. *Computer Systems & Applications*. 2011;20(2):178–181.
4. Bo Q., Chang-Hai N., Bao-Wen X. Test Case Prioritization Based on Test Suite Design Information. *Chinese Journal of Computer*. 2008;31(3):431–439.
5. Wei L., Zheng Z., Peng H. et al. Predicate Execution-Sequence Based Fault Localization Algorithm. *Chinese Journal of Computer*. 2013;36(12):2406–2419.
6. Gao-Chao X., Xin-Zhong L., Liang H., Xiao-Dong F., Yu-Shuang D. Software Reliability Assessment Models Incorporating Software Defect Correlation. *Journal of Software*. 2011;22(3):439–450.
7. Rothermel G., Untch R.H., Harrold M.T. Prioritizing test cases for regression testing. *IEEE Trans. Software Engineering*. 2001; 27:929–948.

8. Jian Z., Hong-Yu Z., David L. Where should the Bugs be fixed. *Proc. of the International Conference on Software Engineering*. Zurich: IEEE Computer Society. 2012;14–24.
9. Katerina G.P., Trivedi K.S. Failure correlation in software reliability models. *IEEE Trans. on Reliability*. 2001;49(1):37–48.
10. Chen S., Mills S. A binary Markov process model for random testing. *IEEE Trans. on Software Engineering*. 1996;22(3):218–223.
11. Bishop P.G., Pullen F.D. PODS revisited-A study of software failure behaviour. *Proc. of the IEEE International Symposium On Fault Tolerant Computing*. 1998;2–8.
12. Tao J., Chang-Hai J., De-Bin H., Cheng-Gang B., Kai-Yuan C. An Approach for Detecting Correlated Software Defects. *Journal of Software*. 2005;18(1):17–28.
13. Chang-Ai S. A Constraint-Based Approach to Identifying and Analyzing Failure-Causing Regions. *Journal of Software*. 2012;23(7):1688−1701.
14. De-Ping Z., Chang-Hai N., Bao-Wen X. Cross-Entropy Method Based on Markov Decision Process for Optimal Software Testing. *Journal of Software*. 2008;19(10):2770–2779.
15. Loginova I.A., Dmitrenko Ju.A., Buduluca A.R., Melihov S.A., Zhaljaletdinova Je.I., Shamukova D.R. Primenenie analiza Pareto dlja obespechenija kachestva jelektronnyh sredstv. Pridneprovskij nauchnyj vestnik. 2019;3(4):52–55. (In Russ.)
16. Jin-Xia A., Guo-Qing W., Shu-Fang L., Ji-Hong Z. Dynamic Evaluation Method Based Multi-Dimensional Test Coverage for Software Testing. *Journal of Software*. 2010;21(9):2135–2147.

ИНФОРМАЦИЯ ОБ АВТОРАХ / INFORMATION ABOUT THE AUTHORS

Зозуля Михаил Михайлович, младший научный сотрудник НИЦ, Военный учебно-научный центр Военно-воздушных сил «Военно-воздушная академия имени профессора Н.Е. Жуковского и Ю.А. Гагарина», Воронеж, Российская Федерация.
e-mail: m.m.zozul@gmail.com

Zozulya Mikhail Mikhailovich, junior Researcher at the Research Center, Military Training And Research Center Of The Air Force Academy Named After Professor N.E. Zhukovsky And Y.A. Gagarin", Voronezh, Russian Federation.

Кравец Олег Яковлевич, д.т.н., Воронежский государственный технический университет, Воронеж, Российская Федерация.
e-mail: csit@bk.ru
ORCID: 0000-0003-0420-6877

Kravets Oleg Jakovlevich, Doctor Of Technical Sciences, Voronezh State Technical University, Voronezh, Russian Federation.

Статья поступила в редакцию 26.11.2021; одобрена после рецензирования 02.12.2021; принята к публикации 08.12.2021.

The article was submitted 26.11.2021; approved after reviewing 02.12.2021; accepted for publication 08.12.2021.