15. Поспелов Г. С. Искусственный интеллект – основа новой информационной технологии / Поспелов Г. С. – М. : Наука, 1988. – 280 с.

16. Свирин И. Некоторые аспекты автоматического распознавания автомобильных номеров / И. Свирин, А. Ханин // Алгоритм безопасности. – 2010. – № 3.

17. Тлебалдинова А. С. Исследование методов и алгоритмов для задачи распознавания номерных знаков транспортных средств / А. С. Тлебалдинова, Н. Ф. Денисова // Вестник КазНТУ. – 2014. – № 1 (101).

18. Дорожній транспорт. Знаки номерні транспортних засобів. Загальні вимоги. Правила застосування : ДСТУ 4278:2006.

✧ ✧ ✧ ✧ ✧ ✧ ✧

УДК 621.396.6.019.3

О. В. Иванченко, кандидат технических наук,
доцент кафедры информационных систем
и технологий Академии таможенной службы Украины

МОДЕЛЬ ОТКАЗОУСТОЙЧИВОСТИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ КРИТИЧЕСКОГО ПРИЛОЖЕНИЯ

Обеспечение требуемого уровня функциональной безопасности критических инфраструктур (КИ) в значительной степени определяется параметрами отказоустойчивости программного модуля, используемого в контурах управления КИ (т. е. рассматривается программное обеспечение (ПО) критического приложения). Однако результаты анализа последствий наиболее крупных аварий и катастроф с участием различных инфраструктурных образований свидетельствуют о недооценивании влияния сбоев, отказов ПО критического приложения на ухудшение общего уровня надежности. Для устранения указанной проблемы предлагается использовать комплексный подход к оценке отказоустойчивости ПО, методологической основой которого является аналитико-стохастическое описание процессов функционирования и тестового контроля соответствующего программного модуля.

Ключевые слова: *отказоустойчивость; программное обеспечение критического приложения; марковская модель; модель Гоела–Окумото.*

Ensuring the required level of critical infrastructures (CI's) functional safety is largely determined by the parameters of fault tolerance software used in the control loops CI (that is, to considered critical software applications). However, the results of the analysis of the effects of most major accidents and disasters involving various infrastructure entities indicate underestimating the contribution of failures, failures of critical applications in the deterioration of the general level of reliability. We propose to resolve this problem, using an integrated approach to the assessment of fault tolerance software methodology which is based on analytical and stochastic description of the functioning and the corresponding test control software module.

Key words: *fault tolerance; software of critical applications; Markov model; Goel–Okumoto model.*

Постановка проблемы. Одним из важнейших факторов обеспечения эффективного применения критических инфраструктур по назначению является широкое внедрение информационных технологий в контур управления КИ. Это улучшает быстродействие различных инфраструктурных образований, расширяет их возможности по решению широкого спектра задач критического компьютинга [1]. В значительной степени этому способствует использование программного обеспечения, рассматриваемого как критическое приложение конкретной исследуемой КИ.

Однако анализ последствий наиболее крупных аварий и катастроф с участием критических инфраструктур [2] свидетельствует о существенном влиянии отказоустойчивости программных модулей, входящих в состав систем управления, на общий уровень функциональной безопасности КИ. Исходя из этого, в статье рассмотрен комплексный подход к оценке показателей надежности ПО критического приложения на основе реализуемого механизма отказоустойчивости, применения аппарата аналитико-стохастического моделирования процессов функционирования и тестового контроля соответствующего программного модуля.

Цель статьи. В качестве исследуемого программного модуля выбрано критическое приложение, написанное на языке программирования С, используемое в интересах Европейского космического агентства (ЕКА). Указанный программный модуль информационно обеспечивает пользовательский интерфейс центра управления космическими аппаратами ЕКА и применяется для формирования файла данных в соответствии с заранее определенным пользовательским форматом, отвечающим конкретным характеристикам конфигурации массива антенных систем (АС). Это ПО включает около 10 000 линий кода С [3]. Программный модуль состоит из трех основных подмодулей:

1) подмодуль синтаксического анализа (ПМСА);

2) вычислительный подмодуль (ПМВТ);

3) подмодуль форматирования (ПМФТ).

Исходя из указанного, требуется оценить уровень надежности рассматриваемого программного модуля на основе аналитико-стохастического моделирования процессов функционирования и тестового контроля соответствующего ПО критического приложения. В качестве опорной модели предлагается использовать дискретную марковскую цепь, с помощью которой будут определены временные характеристики исследуемого инструментария. На следующем этапе на основе использования известных стохастических зависимостей для случая ограничения предельного роста интенсивности отказов программного обеспечения планируется построить соответствующую оценочную модель.

Изложение основного материала. На рис. 1 в соответствии с [3; 4] представлена архитектура исследуемого ПО, которая фактически трансформируется в граф состояний приводимой дискретной марковской цепи (ДМЦ). Из рисунка видно, что ДМЦ строится с учетом взаимодействия трех указанных подмодулей С и операционной системы (ОС). Для реализации аналитико-стохастического моделирования рассмотрим следующие состояния, описывающие процесс функционирования программного модуля критического приложения: С1 – состояние функционирования ПМСА; С2 – состояние функционирования ПМВТ; С3 – состояние функционирования ПМФТ; OS – состояние функционирования ОС и системного обмена с подмодулями С; End – состояние завершения вычислительного процесса по управлению АС.

Значения переходных вероятностей для приводимой ДМЦ в соответствии с [4] представлены в табл. 1. Моделирование выполняется для начальных условий в момент времени $t = 0$, когда $P_{OS}(0) = 1$, $P_{C1}(0) = 0$, $P_{C2}(0) = 0$, $P_{C3}(0) = 0$, $P_{End}(0) = 0$.

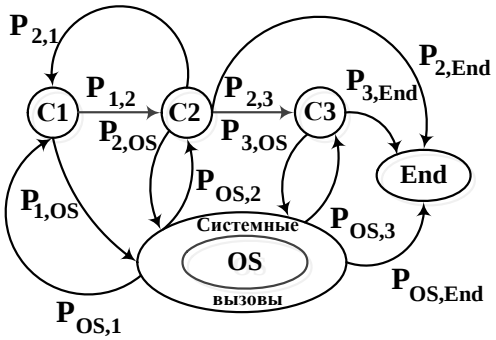

Рис. 1. Архитектура исследуемого программного обеспечения

Таблица 1

Матрица переходных вероятностей

p_{ij}	C1	C2	C3	OS	End
C1	0,236	3E–4	0	0,77	7,8E–5
C2	0	0,632	0,029	0,34	5,0E–4
C3	0	0	9,9E–4	0,99	9,1E–4
OS	0,405	0,014	0,581	0	0
End	0	0	0	0	1

В общем виде матрица переходных вероятностей $P = [p_{ij}]$ может быть записана следующим образом:

$$P=\begin{pmatrix} Q & C \\ 0 & I \end{pmatrix}, \tag{1}$$

где Q – субстохастическая матрица размерностью $(n - \ell)$ на $(n - \ell)$; I – единичная матрица размерностью l = 1 (т. е. размерность матрицы соответствует количеству поглощающих состояний); 0 – матрица размерностью ℓ на $(n - \ell)$, все элементы которой равны нулю; C – матрица размерностью $(n - \ell)$ на ℓ.

Тогда фундаментальная матрица M для k-го количества переходов из состояния s_i в состояние s_j может быть представлена в виде [5]

$$M=(I-Q)^{-1}=I+Q+Q^2+...+Q^k=\sum_{k=0}^{\infty}Q^k\ . \tag{2}$$

После несложных преобразований фундаментальная матрица (2) приводится к виду M = 1 + MQ. Обозначим переходы из состояния s_i в состояние s_j через $x_{i,j}$. Будем полагать, что ожидаемое (прогнозируемое) количество переходов соответствует значению математического ожидания $n_{i,j} = E[X_{i,j}]$ или $m_{i,j}$. После чего ожидаемое количество переходов из

начального состояния i = 1 в состояние j запишем как $v_{1,j} = m_{1,j}$. Далее в фундаментальной матрице можно выстроить диагональ из элементов [6]

$$M_D = \begin{cases} m_{i,j}, \ \forall i=j, \\ 0, \ i \neq j, \end{cases} \tag{3}$$

и определить

$$\sigma^2 = M(2M_D - I) - M_2, \tag{4}$$

где $M_2 = [m_{i,j}^2]$, $D[X_{i,j}] = \sigma_{i,j}^2$.

Используя архитектурно ориентированный подход, изложенный в [3], мы можем определить вероятность правильного функционирования ПО критического приложения как

$$E[R] = \prod_i^n E[R_i^{X_{1,i}}] = \left(\prod_i^{n-1} R_i^{E[X_{1,i}]} \right) R_n, \tag{5}$$

где $R_i^{E[X_{1,i}]}$ – вероятностный показатель функционирования i-й компоненты ПО за количество посещений $E[X_{1,i}]$; R_n – вероятностный показатель функционирования n-го компонента ПО (т. е. вероятностный показатель завершения функционирования программного модуля).

Разложив выражение (5) в ряд Тейлора, получим [6]

$$E[R] = \left[\prod_i^{n-1} \left(R_i^{m_{1,i}} + \frac{1}{2} (R_i^{m_{1,i}})(\log R_i)^2 s_{1,i}^2 \right) \right] R_n, \tag{6}$$

где $m_{1,i} = E[X_{1,i}]$, $D[X_{1,i}] = \sigma_{1,i}^2$.

При рассмотрении (6) необходимо учитывать, если $X_{1,n} = 1$, то $m_{1,n} = 1$, $\sigma_{1,n}^2 = 0$.

Для завершения архитектурного описания модели добавим состояние, учитывающее функционирование ОС, которая взаимодействует с другими компонентами программного модуля С через интерфейс системных вызовов. Предположим, что вероятностный показатель W правильного функционирования ОС известен. Тогда по аналогии с (6) будет справедливо следующее выражение:

$$W' = E[W^{X_{1,OS}}] = W^{m_{1,OS}} + \frac{1}{2} W^{m_{1,OS}} \log W^2 \sigma_{1,OS}^2, \tag{7}$$

где $m_{1,OS}$, $\sigma_{1,OS}^2$ – математическое ожидание и дисперсия количества посещений ОС, соответственно.

Аналогично запишем выражение для остальных компонентов ПО

$$R' = E[R_i^{X_{1,i}}] = R_i^{m_{1,i}} + \frac{1}{2} R_i^{m_{1,i}} \log R_i^2 \sigma_{1,i}^2, \tag{8}$$

а выражение (6) представим в упрощенном виде как

$$E[R] = \left[\prod_i^{n-1} R_i' \right] R_n W'. \tag{9}$$

При дальнейших вычислениях ОС рассматривается как абсолютно надежная, правильно функционирующая система. Поэтому можно принять $W \approx 1$.

С помощью предложенной опорной модели определяются основные временные и системные характеристики функционирования исследуемого программного модуля критического приложения. В качестве исследуемых временных характеристик особый интерес вызывает фактическая продолжительность выполнения пользовательских функций при обращении к подмодулям С1, С2, С3 и системных вызовов при обращении к ОС. При оценке пользовательских и системных параметров необходимо определить среднее количество обращений к модулю С и среднее количество системных вызовов при обращении к ОС. Использование указанных характеристик предоставляет возможность найти продолжительность полного рабочего цикла ПО. Результаты моделирования, полученные с использованием специализированной программной утилиты SREPT (Software Reliability Estimation and Prediction Tool), представлены в табл. 2.

В качестве основной оценочной модели безотказности рассматривается экспоненциальная модель, для которой интенсивность отказов компонентов ПО, т. е. подмодулей С1, С2, С3 и ОС на этапе эксплуатации ПО, является постоянной величиной. При таком допущении вероятность их безотказной работы (ВБР) определяется с помощью известного соотношения

$$P(\tau_i)=\exp\left\{-\int_0^{\tau_i}\lambda_i(t)dt\right\}=\exp\ -\lambda_i\tau_i\ , \qquad (10)$$

где τ_i – время реализации пользовательских функций для i-го подмодуля С.

Таблица 2

Основные временные и системные характеристики ПО

Исследуемый параметр		Величина
1. Временные характеристики, ч		
Продолжительность реализации пользовательских функций (для подмодулей С)		
С1	С2	С3
0,01 128	0,00 248	0,0 001 251
Продолжительность реализации системных вызовов (для ОС)		0,10 589
2. Пользовательские и системные характеристики		
Среднее количество пользовательских функций (для модуля С)		407,4
Среднее количество системных вызовов (для ОС)		5442

В соотношении (10) значения τ_i определяются с использованием данных, представленных в табл. 2. Для адекватного оценивания изменения уровня надежности каждого i-го компонента ПО предлагается использовать модель Гоела–Окумото [7], в соответствии с которой интенсивность отказов определяется согласно выражению

$$\lambda(T)=E_E e^{-\beta T}, \qquad (11)$$

где $E_E = kN$ – ожидаемое значение интенсивности отказов (сбоев) ПО за конечное время его выполнения Т; β – скорость снижения интенсивности отказов (сбоев) ПО за счет частичной реализации механизма отказоустойчивости; k – коэффициент пропорциональности, устанавливающий связь между количеством отказов и временем выполнения ПО.

Основу механизма отказоустойчивости программных модулей (ПО) критического приложения составляют перезапускающиеся компоненты, т. е. приложения с повторным выполнением и резервированием. Обычно программные модули С включают в свой состав основную и резервную копию. Исходя из этого, механизм отказоустойчивости реализуется в соответствии со схемой, представленной на рис. 2, как последовательность следующих событий.

Событие 1 – обнаружить отказ (сбой) после его возникновения не удалось.

Событие 2 – перезагрузить программный модуль после возникновения и необнаружения отказа (сбоя) не удалось.

Событие 3 – невозможно выполнить повторный запуск приложения после того, как произошел отказ (сбой), его обнаружение и выполнение перезагрузки программного модуля не удалось.

Событие 4 – невозможно выполнить переход на резервный программный модуль после того, как произошел отказ (сбой), и обнаружить его; соответственно, перезагрузить программный модуль, осуществить повторный запуск приложения не удалось.

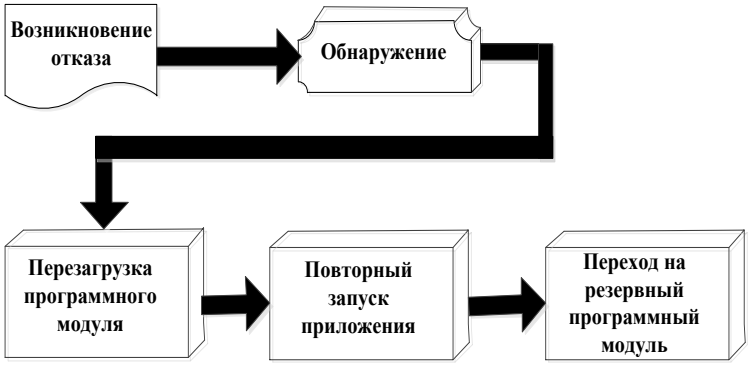

Рис. 2. Схема реализации механизма отказоустойчивости ПО критического приложения

Согласно соотношению (10) значение вероятности отказа i-й компоненты программного модуля определяется в виде $F_i = 1 - R_i = 1 - \exp\ -\lambda_i t_i$, т. е. наработка между отказами (TTF) распределена по экспоненциальному закону. Соответственно, когда происходит отказ (сбой) одного из компонентов программного модуля С, имеют место следующие группы событий (обозначим их как E1, E2, E3).

1. Произошел отказ (сбой), который успешно обнаружен или не обнаружен, но операции перезагрузки, повторного запуска основного программного модуля и перехода на резервный программный модуль потерпели неудачу. Для этого события справедливо соотношение

$$P_{E1} = P_{\text{Событие 1}} + (1 - P_{\text{Событие 1}})(P_{\text{Событие 2}} P_{\text{Событие 3}} P_{\text{Событие 4}}),$$

а условное распределение TTF модуля С как распределение основного программного модуля подчиняется экспоненциальному закону, т. е.

$$F(t) = EXP(\lambda) = 1 - e^{-\lambda t}. \qquad (12)$$

2. Обнаружение отказа (сбоя) и перезагрузка успешны или повторный запуск приложения успешен после срыва перезагрузки программного модуля. Это событие описывается соотношением

$$P_{E2} = (1 - P_{\text{Событие 1}})[(1 - P_{\text{Событие 2}}) + P_{\text{Событие 2}}(1 - P_{\text{Событие 3}})].$$

В этом случае условное распределение TTF модуля С соответствует закону Эрланга 2-го порядка, т. е.

$$F(t) = \text{ERLANG}(\lambda, 2) = 1 - (e^{-\lambda t}(1+\lambda t)). \quad (13)$$

3. Обнаружение отказа (сбоя) успешно, но операции перезагрузки и повтора неуспешны. Однако переход на резервный программный модуль был выполнен успешно, т. е.

$$P_{E3} = (1 - P_{\text{Событие 1}})[P_{\text{Событие 2}} P_{\text{Событие 3}}(1 - P_{\text{Событие 4}})].$$

Если резервная копия является точной копией основной версии, то условное распределение TTF описывается соотношением (13). Если используется другая, отличающаяся от основной, версия, то условное распределение TTF задается последовательностью из двух независимых показательных распределений (описывающих TTF основного и резервного программных модулей). Это распределение известно как гиперэкспоненциальное распределение. Располагая λ_1, λ_2, соответственно, как интенсивностями отказов основной и резервной версий, условное распределение TTF запишем в виде

$$F(t) = \text{HYPO}(\lambda_1, \lambda_2) = 1 - \frac{\lambda_2}{\lambda_2 - \lambda_1} e^{-\lambda_1 t} + \frac{\lambda_1}{\lambda_1 - \lambda_2} e^{-\lambda_2 t}. \quad (14)$$

Дальнейшие расчеты связаны с вычислением параметров модели (11), которые определяются в два этапа [7]:

– на этапе предварительного тестирования или перед клиентским использованием ПО по результатам моделирования или аналитического прогнозирования;

– на этапе клиентского использования по результатам статистического анализа, обработки данных о размере и сложности программного модуля; информации об отказах, сбоях и дефектах ПО.

График функциональной зависимости $\lambda(T)$ представлен на рис. 3.

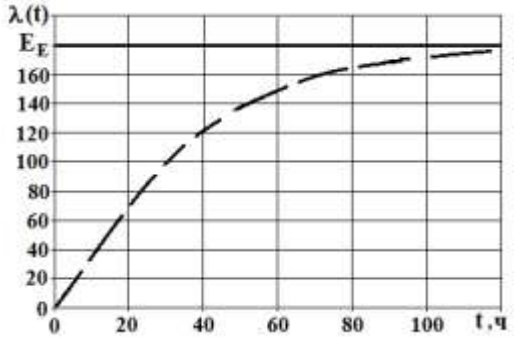

Рис. 3. Зависимость $\lambda(t)$ для модели Гоела–Окумото

В табл. 3 представлены результаты статистического оценивания параметров модели Гоела–Окумото, полученные в [3] на этапе клиентского использования ПО, архитектура которой представлена на рис. 1.

Для вычисления значений ВБР исследуемого программного модуля С рекомендуется использовать следующее соотношение:

$$P(\tau)=\prod_{s=1}^{\xi} P_s(\tau_s), \qquad (15)$$

где $P_s(\tau_s)$ – вероятность безотказной работы s-й составляющей программного модуля С за время реализации τ_s пользовательских функций s-м подмодулем; ξ – количество подмодулей, содержащихся в программном модуле С.

Таблица 3

Статистические оценки параметров модели Гоела–Окумото

Подмодуль синтаксического анализа (С1)		Вычислительный подмодуль (С2)		Подмодуль форматирования (С3)	
N	k	N	k	N	k
13	0,0546	11	0,0946	5	0,0534

Тогда, подставляя в формулу (12) в качестве исходных данных как временные характеристики программного модуля (табл. 2), так и статистические оценки параметров модели Гоела–Окумото (табл. 3), мы можем определить значения ВБР для случая частичной реализации механизма отказоустойчивости.

Общее выражение для определения значения показателя надежности R_C программного модуля с полностью реализуемым механизмом отказоустойчи-вости (рис. 2) записывается следующим образом [3]:

$$R_C=1-\ P_{E1}\text{ЧEXP}(\lambda)+P_{E2}\text{ЧERLANG}(\lambda,\ 2)+P_{E3}\text{Ч}\ \gamma\text{ЧERLANG}(\lambda,2)+\gamma\text{ЧHYPO}(\lambda_1,\lambda_2))\ , \qquad (16)$$

где $\gamma = 1$, если резервная версия совпадает с основной, иначе $\gamma = 0$.

Заменив в (8) R_i на R_{C_i} и подставив в соотношение (9), получим следующее выражение:

$$E[R] = \left[\prod_{i}^{n-1} R'_{C_i}\right] R_{C_n} W' . \qquad (17)$$

Далее, используя в качестве исходных данных как временные характеристики программного модуля (табл. 2), так и статистические оценки параметров модели Гоела–Окумото (табл. 3), определяем результирующее значение вероятностного показателя правильного функционирования ПО критического приложения с учетом реализуемого механизма отказоустойчивости E[R]. График зависимости $E[R,T_0,t]$ для указанных исходных данных представлен на рис. 4.

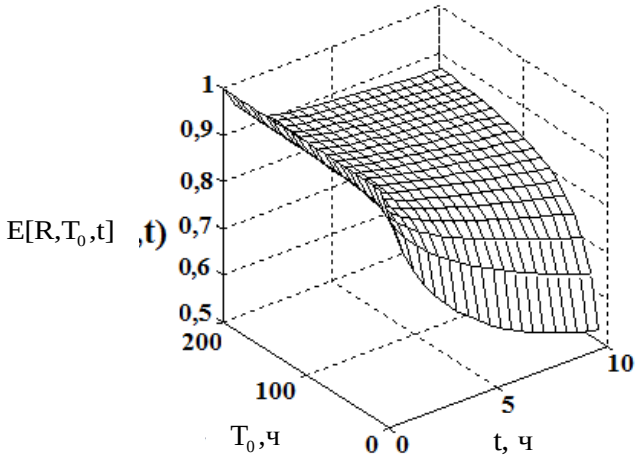

Рис. 4. Зависимость $E[R,T_0,t]$

Результаты моделирования (рис. 4), полученные с использованием соотношений (11–17), могут быть использованы для обоснования предельных величин показателей надежности и отказоустойчивости ПО критического приложения.

Выводы из данного исследования и перспективы дальнейших разведок в данном направлении. Дальнейшие перспективы применения предложенного модельного ряда связаны с оптимизацией механизма отказоустойчивости и оценки гарантоспособности программных модулей, используемых в контурах управления критических инфраструктур.

Список использованных источников:

1. Безопасность критических инфраструктур: математические и инженерные методы анализа и обеспечения / под ред. В. С. Харченко. – Харьков : Национальный аэрокосмический университет им. Н. Е. Жуковского “ХАИ”, 2011. – 641 с.

2. Информационные технологии для критических инфраструктур / под ред. А. В. Скаткова. – Севастополь : Севастопольский национальный технический университет, 2012. – 306 с.

3. Pietrantuono R. Software Reliability and Testing Time Allocation: An Architecture-Based Approach / R. Pietrantuono, S. Russo, K. Trivedi // IEEE Transactions on Software Engineering. – 2010. – Vol. 36. – № 3 – P. 323–337.

4. Trivedi K. S. Probability and Statistics with Reliability, Queuing and Computer Science Applications / Trivedi K. S. – John Wiley and Sons, 2001. – 356 p.

5. Chin-Yu Huang An Assessment of Testing-Effort Dependent Software Reliability Growth Models / Chin-Yu Huang, Sy-Yen Kuo, Michael R. Lyu // IEEE Trans. On Reliability. – 2007. – Vol. 56. – № 2.

6. Sharma V. S. Quantifying software performance, reliability and security: an architecture-based approach / V. S. Sharma, K. S. Trivedi // The journal of systems and software, 2007. – Vol. 80. – Issue 4. – P. 493–509.

7. Application of Goel–Okumoto Model in Software Reliability Measurement [Электронный ресурс] // Special Issue of International Journal of Computer Applications (0975–8887) on Issues and Challenges in Networking, Intelligence and Computing Technologies. – ICNICT 2012, November 2012. – Режим доступа : http://www.research.ijcaonline.org/icnict/number5/icnict1007.pdf