УДК (651. 34)

Кассем Халифе, А.Е. Горюшкина, В.Н. Змиевская

Национальный технический университет «ХПИ», Харьков

GERT-МОДЕЛЬ ПРОЦЕССА БЕЗОПАСНОГО ТЕСТИРОВАНИЯ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

Проведен анализ жизненного цикла программного обеспечения. Определено, что одним из основных этапов, влияющих на безопасность программного обеспечения, является его тестирование. Разработана GERT-модель процесса безопасного тестирования программного обеспечения. В ходе моделирования разработаны и математически формализованы типовая и упрощенная GERT-сети алгоритма тестирования программного обеспечения. Найдены эквивалентная W-функция и плотность распределения вероятностей времени тестирования программного обеспечения. Модель может быть использована для исследования основных этапов тестирования ПО с целью снижения его уязвимости и повышения безопасности IT-проекта в целом, а также при разработке новых методов, алгоритмов и способов управления IT-проектами.

Ключевые слова: *безопасное тестирование программного обеспечения, GERT-модель, жизненный цикл программного обеспечения, жизненный цикл бага.*

Введение

Постановка проблемы. Современные методики разработки программного обеспечения представляют собой сложный многоитерационный процесс, в котором задействован ряд специалистов, выполняющих специальные функции. Важность каждого из этапов этого процесса показателями, в целом характеризующими конечный положительный эффект разработки. Одним из наиболее важных показателей эффективности разработки является безопасность программного обеспечения (ПО) [2, 3], характеризующая степень его уязвимости к различного рода угрозам злоумышленных хакерских вторжений.

Анализ литературы [2, 3] показал, что с точки зрения безопасности одними из наиболее важных этапов являются этапы подготовки соответствующей документации и проведения тестирования ПО. Часто именно из-за ошибок в планировании тестовой документации и самих процедур тестирования IT-фирмы выпускают заведомо уязвимые к кибератакам программные продукты. Проведенные исследования показали, что в настоящее время существует множество методик тестирования ПО. Их эффективность в основном определяется практическим опытом различных фирм-разработчиков ПО. В то же время единой теоретической базы, позволяющей аргументировано давать рекомендации использования методик тестирования ПО, в настоящее время не существует. Поэтому актуальными становятся решения задач математической формализации процесса тестирования ПО для снижения его уязвимостей (безопасного тестирования ПО).

Основная часть

Процесс разработки программного обеспечения можно рассматривать как сетевую GERT-структуру [4, 6], входными данными которой является поток задач, требующих решения, а конечной целью становится поток выполненных задач. При этом следует учитывать, что любая исходная задача может быть декомпозирована на более мелкие подзадачи, что в целом ускоряет процесс упрощающих преобразований GERT-сети. В конечном итоге такая декомпозиция позволит описать совокупность нескольких единичных задач, например, реализующих отдельные методы (единичной задачей считается та, которую один специалист может выполнить за один рабочий день). Далее формируется очередь единичных задач, которая в соответствии с различными методологиями разработки ПО разбивается либо на несколько итераций (в соответствии с гибкими методологиями типа «agile»), либо на более сложные комплексные итерационные структуры (в соответствии со «спиральной» методологией) [3].

Проведенные исследования показали, что критерием разбиения на итерации чаще всего служит или отведенное на итерацию время, или количество составляющих задач. Поэтому использование подобного разбиения при математической формализации «водопадной» модели не представляется целесообразным.

GERT-сеть, описывающая процесс реализации поставленных задач командой разработчиков программного обеспечения, содержит N-ветвей, соответствующих N членам коллектива. Поскольку, описываемый моделью процесс представляет собой поитерационную деятельность, зависящую от времени и вероятности выполнения предыдущих итераций, моделируемый процесс целесообразно считать полумарковским. Кроме этого, при математической формализации процесса разработки программного обеспечения необходимо учитывать возможность одновременного завершения выполнения нескольких задач и необходимость выполнения командой разработчиков мини-

мального количества задач на итерацию. В ходе исследования процесса выполнения командой разработчиков программного обеспечения задач, в которых были определены типичные характеристики обслуживания в зависимости от свойств входного потока данных, параметров и структуры системы и дисциплин обслуживания. В работе в качестве основной характеристики системы принята вероятность успешного выполнения командой разработчиков всех поставленных задач за отведенное на итерацию время.

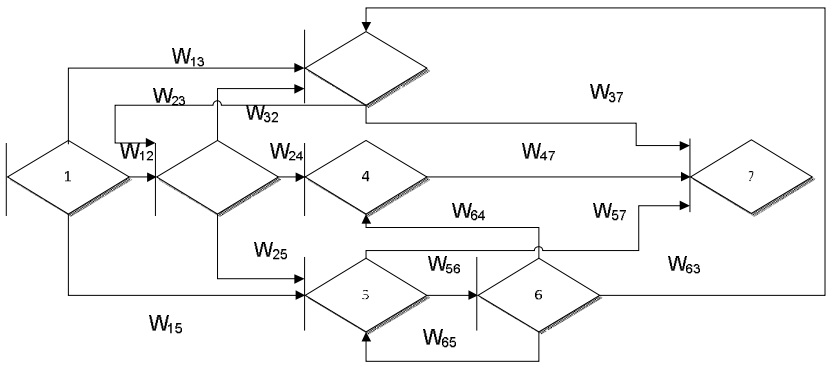
Проведенный анализ методик разработки программного обеспечения показал, что одним из ключевых этапов этого процесса, обеспечивающих отсутствие критических ошибок и, соответственно, безопасность продукта, является тестирование.

Разработаем GERT-модель процесса тестирования программного обеспечения. Типовая GERT-модель алгоритма тестирования программного обеспечения с учетом поиска его уязвимостей в виде GERT-сети представлена на рис. 1, а. Эта модель может интерпретироваться следующим образом. Состояние 1 соответствует статусу «Новый». Этот статус присваивается автоматически после внесения баг-репорта. Состояние 2 соответствует статусу «Открыт». Такой статус баг получает после того, как была проведена его валидация руководителем команды и данная ошибка действительно должна быть исправлена. Состояние 3 соответствует статусу «Отсрочен» (формируется, если баг не нужно исправлять в данной итерации). Состояние 4 соответствует статусу «Исправлен», который присваивается специалистом разработки после того, как ошибка (баг) была устранена. Состояние 5 соответствует статусу «Отклонён». Данный статус присваивается также после анализа нового бага руководителем команды в случае, если описанная ошибка уже ранее была внесена в систему (дубликат) или же по каким-то причинам не требуется её исправление. Состояние 6 соответствует статусу «Повторно открыт», который формируется при повторном возникновении ошибки после её предварительного исправления. Состояние 7 соответствует статусу «Закрыт». Данное состояние формируется после окончательного исправления бага и проведения дополнительной проверки. Для такой GERT-сети эквивалентная W-функция времени тестирования программного обеспечения равна

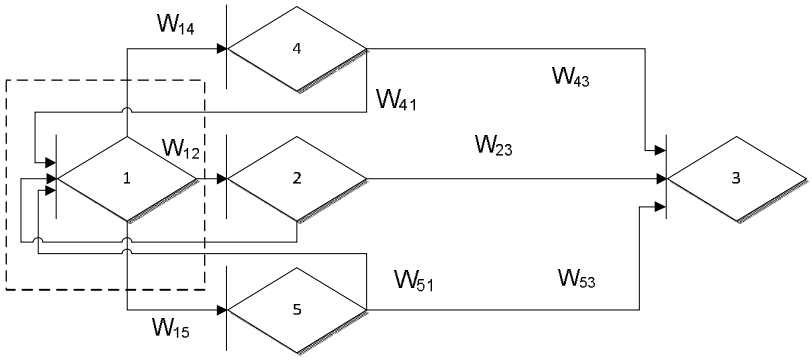
$$W_E(s) = \frac{\left(\begin{array}{c} W_{12}W_{24}W_{47} + W_{15}W_{57} + W_{13}W_{37} + W_{12}W_{23}W_{37} \\ +W_{12}W_{25}W_{57} + W_{15}W_{56}W_{64}W_{47} + \\ +W_{12}W_{25}W_{56}W_{64}W_{47} + W_{15}W_{56}W_{63}W_{37} + \\ +W_{12}W_{25}W_{56}W_{64}W_{47} \end{array}\right)}{1 - W_{12}W_{25}W_{56}W_{65} - W_{12}W_{23}W_{32}} \quad (1)$$

Сложность (1) после подстановки производящих функций моментов значительно снижает практическую значимость разрабатываемой математической модели тестирования программного обеспечения. В то же время проведение эквивалентных упрощающих преобразований позволит снизить вычислительную сложность модели, практически не уменьшив точность ее результатов. Для решения этой задачи воспользуемся общей методикой упрощающих эквивалентных преобразований GERT-сети, описанной в работах [4]. Тогда с помощью эквивалентных упрощающих преобразований сформируем упрощенную GERT-сеть тестирования программного обеспечения в виде рис. 1, а.

В представленной упрощенной модели состояние 1 соответствует статусам «Новый – Открыт – Повторно открыт». Состояние 2 соответствует статусу «Исправлен». Состояние 3 соответствует статусу «Закрыт». Состояние 4 – статус «Отклонён». Состояние 5 соответствует статусу «Отсрочен». Соответствующие ветви модели интерпретируются математической формализацией переходов из со-



а – типовая



б – упрощенная

Рис. 1. GERT-сеть тестирования программного обеспечения

стояния в состояние. Характеристики ветвей модели представлены в табл. 1.

Таблица 1

Характеристики ветвей упрощенной GERT-модели тестирования программного обеспечения

№ п/п	Ветвь	W-функция	Вероятность	Производящая ф-ия моментов
1.	(1,2)	W_{11}	p1	$\lambda_1/(\lambda_1-s)$
2.	(1,4)	W_{14}	p2	$\lambda_2/(\lambda_2-s)$
3.	(1,5)	W_{15}	1 - p1 - p2	$\lambda_2/(\lambda_2-s)$
4.	(2,3)	W_{23}	p3	$\lambda_3/(\lambda_3-s)$
5.	(2,1)	W_{21}	1 - p3	$\lambda_4/(\lambda_4-s)$
6.	(4,3)	W_{43}	p4	$\lambda_5/(\lambda_5-s)$
7.	(4,1)	W_{41}	1 - p4	$\lambda_4/(\lambda_4-s)$
8.	(5,3)	W_{53}	p5	$\lambda_5/(\lambda_5-s)$
9.	(5,1)	W_{51}	1 - p5	$\lambda_4/(\lambda_4-s)$

Эквивалентная W-функция времени тестирования программного обеспечения равна

$$W_E(s)=\frac{W_{12}W_{23}+W_{14}W_{43}+W_{15}W_{53}}{1-W_{15}W_{51}-W_{14}W_{41}-W_{12}W_{21}}= \\ =\left(\begin{array}{c}\left(p_1\lambda_1/(\lambda_1-s)\times p_3\lambda_3/(\lambda_3-s)\right)+\\ +\left(p_2\lambda_2/(\lambda_2-s)\times p_4\lambda_5/(\lambda_5-s)\right)+\\ +\left(p_5\lambda_5/(\lambda_5-s)\times q_1\lambda_2/(\lambda_2-s)\right)\end{array}\right)/\left(\begin{array}{c}1-\left(q_1\lambda_2/(\lambda_2-s)\times q_4\lambda_4/(\lambda_4-s)\right)-\\ -\left(p_2\lambda_2/(\lambda_2-s)\times q_3\lambda_4/(\lambda_4-s)\right)-\\ -\left(p_1\lambda_1/(\lambda_1-s)\times q_2\lambda_4/(\lambda_4-s)\right)\end{array}\right), \quad (2)$$

где $q_1=1-p_1-p_2$, $q_2=1-p_3$, $q_3=1-p_4$, $q_4=1-p_5$. Проведя несложные математические действия, получим выражение

$$W_E(s)=\frac{\frac{(\lambda_4-s)}{(\lambda_3-s)}\times\left(\begin{array}{c}\lambda_2\lambda_5q_5(\lambda_1-s)(\lambda_3-s)+\\ +p_1p_3\lambda_1\lambda_3(\lambda_2-s)(\lambda_5-s)\end{array}\right)}{(\lambda_5-s)\times\left(\begin{array}{c}\left((\lambda_4-s)(\lambda_1-s)-p_1q_2\lambda_1\lambda_4\right)\times\\ \times(\lambda_2-s)-\lambda_2\lambda_4q_6(\lambda_1-s)\end{array}\right)}, \quad (3)$$

где $q_5=(p_2p_4+q_1p_5)$, $q_6=(q_1q_4-q_3p_2)$.

В сложных GERT-сетях с возможными циклами отсутствуют простые методы нахождения особых точек функции $\Phi_E(z)$ замены действительных переменных $(z=-i\varsigma)$, где ς – действительная переменная. Связано это с тем, что для нахождения особых точек необходимо решать нелинейные уравнения, и чем сложнее структура GERT-сети, тем сложнее и исходное уравнение [4]. Поэтому в ходе моделирования, выполняя комплексное преобразование, получим:

$$\Phi(z)=\frac{uz^3+kz^2+wz+h}{(\lambda_3+z)(\lambda_5+z)\times(z^3+vz^2+rz+c)}, \quad (4)$$

где $u=q_5\lambda_5\lambda_2+p_1p_3\lambda_1\lambda_3$, $h=\lambda_1\lambda_2\lambda_3\lambda_4\lambda_5(q_5+p_1p_3)$,

$k=q_5\lambda_5\lambda_2(\lambda_1+\lambda_3+\lambda_4)+p_1\lambda_1p_3\lambda_3(\lambda_4+\lambda_2+\lambda_5)$,

$w=q_5\lambda_5\lambda_2(\lambda_3\lambda_4+\lambda_1\lambda_4+\lambda_1\lambda_3)+p_1\lambda_1p_3\lambda_3\times$
$\times(\lambda_2\lambda_4+\lambda_5\lambda_4+\lambda_2\lambda_5)$, $v=\lambda_1+\lambda_4+\lambda_2$;

$r=\lambda_2(\lambda_4+\lambda_1+\lambda_4(q_1q_4\lambda_1-q_2q_3))+\lambda_1\lambda_4(1-p_1q_2)$,

$c=\lambda_1\lambda_4\lambda_2(1-p_1q_2-q_1q_4+p_2q_3)$.

Плотность распределения вероятностей времени тестирования программного обеспечения

$$\varphi(x)=\frac{1}{2\pi i}\int_{-i\infty}^{i\infty}\frac{\exp(zx)\cdot(uz^3+kz^2+wz+h)}{(\lambda_3+z)(\lambda_5+z)\times(z^3+vz^2+rz+c)}dz, \quad (5)$$

где интегрирование выполняется по контуру Бромвича [1]. Способ интегрирования зависит от того, имеет ли функция Φ(z) только простые полюсы, или полюсы некоторого порядка. Если функция Φ(z) имеет только простые полюсы, то

$$e^{zx}\Phi(z)=\frac{\exp(zx)\cdot(uz^3+kz^2+wz+h)}{z^5+g_4z^4+g_3z^3+g_2z^2+g_1z+g_0}=\frac{\alpha(z)}{\beta(z)}, \quad (6)$$

где $g_4=\lambda_5+\lambda_3+v$, $g_3=\lambda_5\lambda_3+v(\lambda_5+\lambda_3)+r$,
$g_2=\lambda_5\lambda_3v+r(\lambda_5+\lambda_3)+c$, $g_1=\lambda_5\lambda_3+c(\lambda_5+\lambda_3)$,
$g_0=\lambda_5\lambda_3c$. Тогда плотность распределения времени тестирования программного обеспечения

$$\varphi(x)=\sum_{n=1}^{5}\mathrm{Res}\left[e^{zx}\Phi(z)\right]=\sum_{n=1}^{5}\alpha(z_n)/\beta(z_n)= \\ =\sum_{n=1}^{5}\frac{e^{z_nx}\left(z_n^3u+z_n^2k+z_nw+h\right)}{\left(5z_n^4+4g_4z_n^3+3g_3z_n^2+2g_2z_n+g_1\right)}. \quad (7)$$

Функция Φ(z) кроме простых полюсов, определяемых корнями уравнения $z^3+vz^2+rz+c=0$, может иметь и полюсы второго или третьего порядка. Это возможно в тех случаях, когда значения λ_3 и λ_5 или совпадают между собой, или равны значениям корней z_3, z_4, z_5. В этих случаях плотность распределения времени тестирования программного обеспечения $\varphi(x)$ находится по формуле нахождения вычетов γ_{-1} от полюсов z_n порядка m:

$$\gamma_{-1}=\frac{1}{(m-1)!}\lim_{z\to z_n}d^{m-1}\left[(z-z_n)^m e^{zx}\Phi(z)\right]\Big/dz^{m-1}.$$

Выражение (4) можно представить как дробно-рациональную функцию относительно z со степенью знаменателя большей, чем степень числителя [4], поэтому для него выполняются условия леммы Жордана. Функция Φ(z) имеет полюсы в точках $z_1=-\lambda_3$, $z_2=-\lambda_5$. Многочлен z^3+vz^2+rz+c порождает еще три полюса. Решение уравнения

$$z^3+vz^2+rz+c=0 \quad (8)$$

может быть найдено любым численным методом. Тогда получим еще три особые точки z_3, z_4, z_5.

Коэффициенты уравнения (8) больше нуля. Поэтому, если это уравнение имеет действительные корни, то все они отрицательны. Если уравнение (8)

имеет отрицательные действительные корни, то они обязательно не второй или третьей кратности. Если предположить, что многочлен z^3+vz^2+rz+c порождает полюс третьего порядка в точке $z=-\alpha$, то $(z+\alpha)^3=z^3+3\alpha z^2+3\alpha^2 z+\alpha^3$. Так как $\alpha<0$, то и $\alpha^3<0$, что противоречит условию $\alpha^3=c=$ $=\lambda_1\lambda_4\lambda_2(1-p_1q_2-q_1q_4+p_2q_3)>0$. Следовательно, многочлен z^3+vz^2+rz+c не может порождать полюсов третьего порядка. Можно предположить, что многочлен z^3+vz^2+rz+c порождает полюс второго порядка в точке $z=-\alpha_1$ и полюс первого порядка в точке $z=-\alpha_2$, то должно выполняться соотношение $(z+\alpha_1)^2(z+\alpha_2)=z^3+(2\alpha_1+\alpha_2)z^2+$ $+(2\alpha_1^2+2\alpha_1\alpha_2)z+\alpha_1^2\alpha_2$.. Так как $\alpha_2<0$, и $\alpha_1^2>0$, то естественно $\alpha_1^2\alpha_2<0$. В то же время имеем несоответствие, так как $\alpha_1^2\alpha_2=c=$ $=\lambda_1\lambda_4\lambda_2(1-p_1q_2-q_1q_4+p_2q_3)>0$. Следовательно, выражение (4) не может порождать полюсов второго порядка.

В соответствии с основной теоремой алгебры [5] всякая целая рациональная функция n-й степени имеет n нулей. Следовательно, многочлен z^3+vz^2+rz+c может иметь или три разных отрицательных действительных нуля, или один отрицательный действительный нуль и два комплексных сопряженных нуля.

Выводы

В статье предложена математическая GERT-модель процесса тестирования ПО. Предложенная математическая модель отличается от известных учетом основных этапов жизненного цикла бага (дефекта ПО) в процессе математической формализации GERT-сети. Модель может быть использована для исследования основных этапов тестирования ПО с целью снижения его уязвимости и повышения безопасности IT-проекта в целом, а так же при разработке новых методов, алгоритмов и способов управления IT-проектами. Применение GERT-сетей в ходе математического моделирования даст возможность использования результатов, полученных в аналитическом виде (функции, плотности распределения) для проведения сравнительного анализа и исследований, более сложных комплексных этапов жизненного цикла ПО в условиях возможных внешних злоумышленных воздействий.

Список литературы

1. Гмурман В.Е. Теория вероятностей и математическая статистика / В.Е. Гмурман. – М.: Высшая школа, 2003. – 479 с.

2. Демарко Т. Вальсируя с медведями / Т. Демарко, Т. Листер. – Компания p.m.Office, 2005. – 321 с.

3. Савин Р. Тестирование Дот Ком, или пособие по жестокому обращению с багами в интернет-стартапах / Р. Савин. – М.: Дело, 2007. – 312 с.

4. Семенов С.Г. Методика математического моделирования защищенной ИТС на основе многослойной GERT-сети / С.Г. Семенов // Вісник Національного технічного університету «Харківський політехнічний інститут». – Х.:НТУ «ХПІ». – 2012. –№62 (968). – С. 173-181.

5. Тихомиров В.М. Десять доказательств основной теоремы алгебры / В.М. Тихомиров, В.В. Успенский // Матем. просвещение. – МЦНМО, 1997. – № 1. – С. 50-70.

6. Pritsker A.A. Modeling and analysis using Q-GERT networks / A.A. Pritsker. – New York: Wiley: Distributed by Halsted Press, 1979.

Поступила в редколлегию 24.02.2016

Рецензент: д-р техн. наук, ст. научн. сотр. С.Г. Семенов, Национальный технический университет «ХПИ», Харьков.

GERT-МОДЕЛЬ ПРОЦЕСУ БЕЗПЕЧНОГО ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Кассем Халіфе, А.Е. Горюшкіна, В.М. Зміївська

Проведений аналіз життєвого циклу програмного забезпечення. Визначено, що одним з основних етапів, що впливають на безпеку програмного забезпечення, є його тестування. Розроблена GERT-модель процесу безпечного тестування програмного забезпечення. В ході моделювання розроблені і математично формалізовані типова і спрощена GERT-мережі алгоритму тестування програмного забезпечення. Знайдені еквівалентна W-функция і щільність розподілу вірогідності часу тестування програмного забезпечення. Модель може бути використана для дослідження основних етапів тестування ПО з метою зниження його уразливості і підвищення безпеки IT-проекта в цілому, а так само при розробці нових методів, алгоритмів і способів управління IT-проектами.

Ключові слова: *безпечне тестування програмного забезпечення, GERT-модель, життєвий цикл програмного забезпечення, життєвий цикл бага.*

GERT-MODEL OF PROCESS OF SAFE TESTING OF SOFTWARE

Kassem Khalife, A.E. Gorushkina, V.N. Zmeevskaya

The analysis of life cycle of software is conducted. It is certain that one of the basic stages, influencing on safety of software, there is his testing. The GERT-model of process of the safe testing of software is developed. During a design developed and the model and simplified GERT-network of algorithm of testing of software formalize mathematically. An equivalent W-функция and closeness of distribution of probabilities of time of testing of software is found. A model can be utillized for research of the basic stages of testing SOFTWARE with the purpose of decline of his vulnerability and increase of safety of IT-проекта on the whole, and similarly at development of new methods, algorithms and methods of management IT-проектами.

Keywords: *safe testing of software, GERT-model, life cycle of software, bag's life cycle.*