УДК 681.3.068

Н.В.Василенко, В.А.Макаров

МОДЕЛИ ОЦЕНКИ НАДЕЖНОСТИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

14 of the most famous models of software safety evaluation are reviewed. According to the review, there is no absolutely safe programme, as the absolutely safety degree can′t be theoretically proved and therefore it can not be achieved.

1. Введение

В международном стандарте ISO 9126:1991 [1] надежность выделена как одна из основных характеристик качества программного обеспечения (ПО). Стандартный словарь терминов программного инжиниринга [2] определяет надежность программного обеспечения как способность системы или компонента выполнять требуемые функции в заданных условиях на протяжении указанного периода времени.

Сама проблема надежности программного обеспечения имеет, по крайней мере, два аспекта: обеспечение и оценка (измерение) надежности [3]. Практически вся имеющаяся литература посвящена первому аспекту, а вопрос оценки надежности компьютерных программ недостаточно проработан. Вместе с тем очевидно, что надежность программы гораздо важнее таких традиционных ее характеристик, как время исполнения или требуемый объем оперативной памяти, однако никакой общепринятой количественной меры надежности программ до сих пор не существует.

Модели надежности программных средств подразделяются на *аналитические* и *эмпирические* [4]. Аналитические модели дают возможность рассчитать количественные показатели надежности, основываясь на данных о поведении программы в процессе тестирования. Эмпирические модели базируются на анализе структурных особенностей программ.

Аналитические модели представлены двумя группами: *динамические* и *статические*. В динамических моделях поведение ПО (появление отказов) рассматривается во времени. Если фиксируются интервалы каждого отказа, то получается непрерывная картина появления отказов во времени (модели с непрерывным временем). Может фиксироваться только число отказов за произвольный интервал времени. В этом случае поведение ПО может быть представлено только в дискретных точках (модели с дискретным временем).

В статических моделях появление отказов не связывают со временем, а учитывают зависимость количества ошибок либо от числа тестовых прогонов (модели по области ошибок), либо от характеристики входных данных (модели по области данных).

2. Аналитические динамические модели

2.1. Модель Шумана

Исходные данные для модели Шумана собираются в процессе тестирования ПО в течение фиксированных или случайных временных интервалов. Использование модели Шумана предполагает, что тестирование проводится в несколько этапов. Каждый этап представляет собой выполнение программы на полном комплексе разработанных тестовых данных. Выявленные ошибки регистрируются, но не исправляются. По завершении этапа на основе собранных данных модель Шумана может быть использована для расчета количественных показателей надежности. После этого исправляются ошибки, обнаруженные на предыдущем этапе, при необходимости корректируются тестовые наборы, и проводится новый этап тестирования. При использовании модели Шумана предполагается, что исходное количество ошибок в

программе постоянно и в процессе тестирования может уменьшаться по мере того, как ошибки выявляются и исправляются (новые ошибки не вносятся).

Базовые понятия модели заимствованы из теории надежности аппаратных средств [5].

$R(t)$ — функция надежности — вероятность того, что ни одна ошибка не появится на интервале от 0 до t.

$F(t)$ — функция отказов — вероятность того, что ошибка появится на интервале от 0 до t.

$$F(t) = 1 - R(t)$$

$f(t)$ — плотность вероятности для $F(t)$, $f(t) = \dfrac{dF(t)}{dt} = -\dfrac{dR(t)}{dt}$.

$z(t)$ — функция риска — условная вероятность того, что ошибка появится на интервале от 0 до $t + \Delta t$, при условии, что до t ошибок не было. Если T — время появления ошибки, то

$$z(t)\Delta t = P\{t < T < t + \Delta t | T > t\} = \frac{P(t < T < t + \Delta t)}{P(T < t)} = \frac{F(t + \Delta t) - F(t)}{R(t)}.$$

Поделив обе части на Δt и устремив Δt к 0, переходим к пределу

$$z(t) = \frac{f(t)}{R(t)} = -\frac{1}{R(t)} \cdot \frac{dR(t)}{dt}.$$

Решая это дифференциальное уравнение относительно $R(t)$ при начальном условии $R(0) = 1$, получим

$$R(t) = \exp\left(-\int_0^t z(x)dx \right).$$

$t_{\text{ср}}$ — среднее время между отказами, $t_{\text{ср}} = \int_0^\infty R(t)dt$.

В рассматриваемой модели автор предполагает, что значение функции $z(t)$ пропорционально числу ошибок, оставшихся в ПО после израсходованного на тестирование времени τ: $z(t) = C \cdot \varepsilon_r(\tau)$, где C — некоторая константа; t — время работы ПО без отказа; $\varepsilon_r(\tau)$ — удельное число ошибок на одну машинную команду, оставшихся в системе после времени тестирования τ: $\varepsilon_r(\tau) = \dfrac{E_T}{I_T} - \varepsilon_c(\tau)$, где E_T — число ошибок до начала тестирования; I_T — общее число машинных команд, которое предполагается постоянным в рамках этапа тестирования; $\varepsilon_c(\tau)$ — удельное количество обнаруженных ошибок на одну машинную команду в течение времени τ. Тогда

$$R(t, \tau) = \exp\left[C \cdot \left(\frac{E_T}{I_T} - \varepsilon_c(\tau) \right) \cdot t \right], \quad (1)$$

$$t_{\text{ср}} = \frac{1}{C \cdot \left(\dfrac{E_T}{I_T} - \varepsilon_c(\tau) \right)}. \quad (2)$$

Из величин, входящих в формулы (1) и (2), неизвестны начальное значение ошибок в ПО (E_T) и коэффициент пропорциональности C. Для их определения прибегают к следующим рассуждениям. В процессе тестирования собирается информация о времени и количестве ошибок на каждом прогоне, т.е. общее время тестирования τ складывается из времени каждого прогона. Полагая, что интенсивность появления ошибок постоянна и равна λ, а A_i — количество ошибок на i-м прогоне, можно вычислить

$$t_{\text{ср}} = \frac{\tau}{\sum_{i=1}^{k} A_i}. \quad (3)$$

Имея данные для двух различных моментов тестирования τ_a и τ_b, которые выбираются произвольно с учетом требования $\varepsilon_c(\tau_b) > \varepsilon_c(\tau_a)$, можно сопоставить уравнения (2) и (3) при τ_a и τ_b и получить следующие выражения:

$$E_T = \frac{I_T(\lambda_{\tau_a} \cdot \varepsilon_c(\tau_a) - \lambda_{\tau_b} \cdot \varepsilon_c(\tau_b))}{\lambda_{\tau_a} - \lambda_{\tau_b}}, \quad C = \frac{\lambda_{\tau_a}}{\frac{E_T}{I_T} - \varepsilon_c(\tau_a)},$$

с помощью которых рассчитывается надежность программы по формуле (1).

2.2. Модель Ла Падула

По этой модели выполнение последовательности тестов производится в m этапов. Каждый этап заканчивается внесением изменений (исправлений) в ПО. Возрастающая функция надежности базируется на числе ошибок, обнаруженных в ходе каждого тестового прогона.

Надежность ПО в течение i-го этапа

$$R(t) = R(\infty) - \frac{A}{i}, \; i = 1,2...,$$

где A — параметр роста; $R(\infty) = \lim_{i \to \infty} R(i)$ — предельная надежность ПО. Эти неизвестные величины автор предлагает вычислить, решив следующие уравнения:

$$\sum_{i=1}^{m} \left[\frac{S_i - m_i}{S_i} - R(\infty) + \frac{A_i}{i} \right] = 0,$$

$$\sum_{i=1}^{m} \left[\left(\frac{S_i - m_i}{S_i} - R(\infty) + \frac{A_i}{i} \right) \cdot \frac{1}{i} \right] = 0,$$

где S_i — число тестов на i-м этапе; m_i — число отказов во время i-го этапа; $i = 1,2,\ldots m$.

Определяемый по этой модели показатель есть надежность ПО на i-м этапе:

$$R(t) = R(\infty) - \frac{A}{i}, \; i = m+1, m+2,...$$

2.3. Модель Джелинского — Моранды

Основное положение, на котором базируется модель, заключается в том, что в процессе тестирования ПО значение интервалов времени тестирования между обнаружением двух ошибок имеет экспоненциальное распределение с интенсивностью отказов, пропорциональной числу еще не выявленных ошибок. Каждая обнаруженная ошибка устраняется, число оставшихся ошибок уменьшается на единицу.

Функция плотности распределения времени обнаружения i-й ошибки, отсчитываемого от момента выявления $(i-1)$-й ошибки, имеет вид

$$P(t_i) = \lambda_i \cdot \exp(-\lambda_i \cdot t_i), \tag{4}$$

где λ_i — интенсивность отказов, которая пропорциональна числу еще не выявленных ошибок в программе:

$$\lambda_i = C \cdot (N - i + 1), \tag{5}$$

где N — число ошибок, первоначально присутствующих в программе; C — коэффициент пропорциональности.

Наиболее вероятные значения величин $\overline{N}$ и $\overline{C}$ определяются на основе данных, полученных при тестировании. Для этого фиксируют время выполнения программы до очередного отказа $t_1,t_2,t_3,\ldots,t_k$. Значения $\overline{N}$ и $\overline{C}$ можно получить, решив систему уравнений

$$\sum_{i=1}^{k} (\overline{N} - i + 1)^{-1} = \frac{K}{\overline{N} + 1 - Q \cdot K},$$

$$\overline{C} = \frac{K}{A\left(\overline{N} + 1 - Q \cdot K\right)},$$

где $Q = \frac{B}{AK}; A = \sum_{i=1}^{k} t_i; B = \sum_{i=1}^{k} i \cdot t_i.$

Чтобы получить числовые значения λ_i, нужно подставить вместо N и C их возможные значения $\overline{N}$ и $\overline{C}$. Рассчитав K значений по формуле (5) и подставив их в выражение (4), можно определить вероятность безотказной работы на различных временных интервалах.

2.4. Модель Шика — Волвертона

Это модификация модели Джелинского — Моранды для случая возникновения на рассматриваемом интервале более одной ошибки. При этом считается, что исправление ошибок производится лишь после истечения интервала времени, на котором они возникли. В основе модели Шика — Волвертона лежит предположение, согласно которому частота ошибок пропорциональна не только количеству ошибок в программах, но и времени тестирования, т.е. вероятность обнаружения ошибок с течением времени возрастает. Интенсивность обнаружения ошибок λ_i предполагается постоянной в течение интервала времени t_i и пропорциональна числу ошибок, оставшихся в программе по истечении $(i-1)$-го интервала; но она пропорциональна также и суммарному времени, уже затраченному на тестирование:

$$\lambda_i = C(N - n_{i-1})(T_{i-1} + t_i / 2),$$

где C — коэффициент пропорциональности; N — число ошибок, первоначально присутствующих в программе; n_{i-1} — число ошибок, оставшихся в программе по истечении $(i-1)$-го интервала; T_{i-1} — суммарное время, затраченное на тестирование в течение $(i-1)$ этапов; t_i — среднее время выполнения программы в текущем интервале.

Остальные расчеты аналогичны расчетам модели Джелинского — Моранды.

2.5. Модель Муса

Модель предполагает, что в процессе тестирования фиксируется время выполнения программы до очередного отказа. Но считается, что не всякая ошибка ПО может вызвать отказ, поэтому допускается обнаружение более одной ошибки при выполнении программы до возникновения очередного отказа.

Считается, что на протяжении всего жизненного цикла ПО может произойти M_0 отказов и при этом будут выявлены все N_0 ошибки, которые присутствовали в ПО до начала тестирования. Общее число отказов M_0 связано с первоначальным числом ошибок N_0 соотношением $N_0 = B \cdot M_0$, где B — коэффициент уменьшения числа ошибок. После тестирования, за время которого зафиксировано m отказов и выявлено n ошибок, можно определить коэффициент B: $B = n / m$.

Один из основных показателей надежности, который рассчитывается по модели Муса, — средняя наработка на отказ. Этот показатель определяется как математическое ожидание временного интервала между последовательными отказами.

2.6. Модель переходных вероятностей

Процесс тестирования ПО рассматривается как марковский процесс.

Пусть в начальный момент тестирования ($t = 0$) в ПО было n ошибок. Предполагается, что в процессе тестирования выявляется по одной ошибке. Тогда последовательность состояний системы $\{n, n-1, n-2, n-3, \ldots\}$ соответствует периодам времени, когда предыдущая ошибка уже исправлена, а новая еще не обнаружена. Последовательность состояний $\{m, m-1, m-2, m-3, \ldots\}$ соответствует периодам времени, когда ошибки исправляются.

Любое состояние модели определяется рядом переходных вероятностей (P_{ij}), где P_{ij} означает вероятность перехода из состояния i в состояние j и не зависит от предшествую-

щих и последующих состояний системы, кроме состояний i и j. Вероятность перехода из состояния $(n-k)$ к состоянию $(m-k)$ есть $\lambda_{n-k}\cdot\Delta t$ для $k = 0,1,2\ldots$, где λ — интенсивность проявления ошибок.

Надежность ПО в данной модели определяется выражением

$$R_k(x) = \exp\left(-\int_0^{\tau}\lambda(k,x)dx\right).$$

3. Аналитические статические модели

3.1. Модель Миллса

Использование этой модели предполагает необходимость перед началом тестирования искусственно «засорять» программу, т.е. вносить в нее некоторое количество известных ошибок. Ошибки вносятся случайным образом и фиксируются в протоколе искусственных ошибок. Специалист, проводящий тестирование, не знает ни количества, ни характера внесенных ошибок до момента оценки показателей надежности по модели Миллса. Предполагается, что все ошибки (как естественные, так и искусственно внесенные) имеют равную вероятность быть найденными в процессе тестирования.

Тестируя программу в течение некоторого времени, собирают статистику об ошибках. В момент оценки надежности по протоколу искусственных ошибок все ошибки делятся на собственные и искусственные. Соотношение, называемое формулой Миллса,

$$N = \frac{S\cdot n}{V}$$

дает возможность оценить первоначальное число ошибок в программе N. Здесь S — количество искусственно внесенных ошибок; n — число найденных собственных ошибок; V — число обнаруженных к моменту оценки искусственных ошибок.

3.2. Модель Липова

Липов модифицировал модель Миллса, рассмотрев вероятность обнаружения ошибки при использовании различного числа тестов. Если сделать то же предположение, что и в модели Миллса, т.е. что собственные и искусственные ошибки имеют равную вероятность быть найденными, то вероятность обнаружения n собственных и V внесенных ошибок

$$Q(n,V) = \left(\frac{m}{n+V}\right)\cdot q^{n+V}\cdot(1-q)^{m-n-V}\cdot\frac{\frac{N}{n}\cdot\frac{S}{V}}{\frac{N+S}{n+V}},$$

где m — количество используемых тестов, q — вероятность обнаружения ошибки в каждом из m тестов, рассчитанная по формуле $q = \frac{n+V}{n}$; S — общее количество искусственно внесенных ошибок; N — количество собственных ошибок, имеющихся в ПО до начала тестирования.

3.3. Простая интуитивная модель

Использование этой модели предполагает проведение тестирования двумя группами программистов, использующими независимые тестовые наборы, независимо одна от другой. В процессе тестирования каждая из групп фиксирует все найденные ею ошибки. При оценке числа оставшихся в программе ошибок результаты тестирования обеих групп собираются и сравниваются. Положим, первая группа обнаружила N_1 ошибок, вторая — N_2, а N_{12} — это ошибки, обнаруженные дважды (обеими группами). Если обозначить через N неизвестное количество ошибок, присутствовавших в программе до начала тестирования, то можно эффективность тестирования каждой из групп определить как

$$E_1 = \frac{N_1}{N};\ E_2 = \frac{N_2}{N}.$$

Предполагая, что возможность обнаружения всех ошибок одинакова для обеих групп, можно допустить, что если первая группа обнаружила определенное количество всех ошибок, она могла бы определить то же количество любого случайным образом выбранного подмножества. В частности, можно допустить

$$E_1 = \frac{N_1}{N} = \frac{N_{12}}{N_2}.$$

Тогда

$$N = \frac{N_1 \cdot N_2}{N_{12}}.$$

3.4. Модель Коркорэна

В этой модели не используются параметры времени тестирования и учитывается только результат N испытаний, в которых выявлено N_i ошибок i-го типа. Модель использует изменяющиеся вероятности отказов для различных типов ошибок.

В отличие от двух рассмотренных выше статических моделей, по модели Коркорэна оценивается вероятность безотказного выполнения программы на момент оценки:

$$R = \frac{N_0}{N} + \sum_{i=1}^{K} \frac{Y_i \cdot (N_i - 1)}{N},$$

где N_0 — число безотказных выполнений программы; N — общее число прогонов; K — априори известное число типов ошибок; $Y_i = \begin{cases} a_i, \text{если } N_i > 0, \\ 0, \text{если } N_i = 0; \end{cases}$ a_i — вероятность выявления при тестировании ошибки i-го типа.

В этой модели вероятность a_i должна оцениваться на основе априорной информации или данных предшествующего периода функционирования однотипных программных средств.

3.5. Модель последовательности испытаний Бернулли

Пространство элементарных событий в классической модели последовательности испытаний Бернулли содержит $2n$ точек, где n — число испытаний (в данном случае под испытанием подразумевается запуск программы). Каждый запуск программы имеет два исхода: правильный и неправильный. Обозначим вероятность неправильного исхода p, а вероятность правильного — $(1 - p)$. Вероятность того, что из n запусков k приведут к неправильному результату, выражается формулой биномиального распределения:

$$B(p, n, k) = C(n, k) \cdot p^k \cdot (1 - p)^{n-k},$$

где $C(n,k)$ — число сочетаний. Вероятность p априори неизвестна, но по результатам запусков известны n и k. Величина B как функция p имеет максимум при $p = k / n$. В качестве меры надежности программы принимается величина

$$R = 1 - k / n = (n - k) / n.$$

3.6. Модель Нельсона

Данная модель при расчете надежности ПО учитывает вероятность выбора определенного тестового набора для очередного выполнения программы. Предполагается, что область данных, необходимых для выполнения тестирования программного средства, разделяется на k взаимоисключающих подобластей Z_i, i = 1,2,…,k. Пусть P_i — вероятность того, что набор данных Z_i будет выбран для очередного выполнения программы. Если к моменту оценки надежности было выполнено N_i прогонов программы на Z_i наборе данных и из них n_i прогонов закончились отказом, то надежность ПО определяется равенством

$$R = 1 - \sum_{i=1}^{k} \frac{n_i}{N_i} P_i.$$

4. Эмпирические модели

4.1. Модель сложности

Сложность ПО характеризуется его размером (количеством программных модулей), количеством и сложностью межмодульных интерфейсов. Под программным модулем в данном случае следует понимать программную единицу, выполняющую определенную функцию и взаимосвязанную с другими модулями ПО.

Существует несколько разновидностей модели сложности. В каждой из них дается некая оценка сложности программы, которая считается пропорциональной ее надежности.

4.2. Модель, определяющая время доводки программ

Анализ модульных связей ПО строится на том, что каждая пара модулей имеет конечную (возможно, нулевую) вероятность того, что изменения в одном модуле вызовут изменения в другом. Данная модель позволяет на этапе тестирования, а точнее при тестовой сборке системы, определять возможное число необходимых исправлений и время, необходимое для доведения ПО до рабочего состояния.

5. Вывод

Следует признать, что абсолютно надежных программ не существует, так как абсолютная степень надежности не может быть теоретически доказана и, следовательно, недостижима. Однако важно знать, насколько надежно конкретное ПО. Описанные модели представляют теоретический подход и, как правило, имеют ограниченное применение. До сих пор не предложено ни одного надежного количественного метода оценки надежности ПО, не содержащего чрезмерного количества ограничений.

Практика разработки ПО предполагает приоритет задачи обеспечения надежности над задачей ее оценки. Ситуация выглядит парадоксально: совершенно очевидно, что прежде чем обеспечивать надежность, следует научиться ее измерять. Но для этого нужно иметь практически приемлемую единицу измерения надежности ПО и модель ее расчета.

1. ISO 9126:1991. Информационная технология. Оценка программного продукта. Характеристики качества и руководство по их применению. 186 с.
2. IEEE Std 610.12-1990, IEEE Standard Glossary of Software Engineering Technology (ANSI). 1283 с.
3. Романюк С.Г. // Открытые системы. 1994. №4 — www.osp.ru/os/1994/04/68.htm
4. Благодатских В.А., Волнин В.А., Поскакалов К.Ф. Стандартизация разработки программных средств: Учеб. пособие под ред. О.С.Разумова. М.: Финансы и статистика, 2003. 284 с.
5. Майерс Г. Надежность программного обеспечения. М.: Мир, 1980. 360 с.