Марков А.С. Модели оценки и планирования испытаний программных средств по требованиям безопасности информации // Вестник МГТУ им. Н.Э. Баумана. Сер. «Приборостроение», 2011. Специальный выпуск «Технические средства и системы защиты информации». С. 90-103. ISSN 0236-3933.

УДК 681.3.06

А.С.Марков

МОДЕЛИ ОЦЕНКИ И ПЛАНИРОВАНИЯ ИСПЫТАНИЙ ПРОГРАММНЫХ СРЕДСТВ ПО ТРЕБОВАНИЯМ БЕЗОПАСНОСТИ ИНФОРМАЦИИ

Представлен обзор и классификация моделей планирования испытаний программ по требованиям безопасности информации. Даны рекомендации по выбору моделей.

Несмотря на усилия ведущих мировых разработчиков программного обеспечения (ПО), задача снижения числа уязвимостей в программных системах не получила реального решения [7]. Объективно это обусловлено чрезвычайно высокой структурной сложностью программных систем, динамичностью версий и технологий. Одним из путей повышения уровня безопасности ПО является использование на этапах тестирования и испытаний ПО моделей, позволяющих получить гарантированные оценки показателей безопасности ПО и эффективности технологии его разработки. Большинство таких моделей заимствованы из теории надежности технических систем, поэтому в литературе их часто называют моделями надежности ПО [8,9]. С точки зрения испытаний программных средств по требованиям безопасности информации понятие «надежность функционирования ПО» эквивалентно понятию «технологической безопасности ПО», где под ошибкой понимается *уязвимость* (некорректность программирования, дефект, недекларированная возможность), потенциально влияющая на безопасность системы и инфраструктуры [4].

Опыт испытательных лабораторий показывает, что применение математических моделей не должно отвлекать экспертов от реального трудоемкого и ответственного процесса исследования ПО и должно, главным образом, способствовать принятию правильных решений. Поэтому модели целесообразно классифицировать по имеющимся входным статистикам, получаемым на различных этапах жизненного цикла ПО:

1. Модели полноты тестирования, позволяющие получить оценки показателей доверия к процессу оценки соответствия ПО;

2. Модели сложности ПО, позволяющие оценить метрики сложности ПО и связанные с ними показатели качества и безопасности ПО;

3. Временные модели роста надежности (reliability growth model), позволяющие оценить показатели технологической безопасности ПО в зависимости от времени испытаний;

4. Отладочные модели, позволяющие оценить показатели технологической безопасности ПО в зависимости от прогонов на заданных областях входных данных и последующих доработок ПО.

1. Модели полноты тестирования

Модели оценки полноты тестирования ПО основаны на методах независимого внесения и выявления тестовых ошибок и методах проведения независимых экспертиз.

1.1. Модель учета внесенных ошибок

Модель учета внесенных ошибок, известная также как решение задачи теории вероятности «меченых рыб» или как модель Миллса (Mills), предполагает внесение в текст программы тестовых ошибок. В процессе тестирования собирается статистика о выявленных ошибках - внесенных и реальных.

Предполагается, что тестовые ошибки вносятся случайным образом, выявление всех ошибок (внесенных и собственных) равновероятно. В этом случае, используя метод максимального правдоподобия, можно получить оценку числа первоначальных ошибок ПО:

$$N = \frac{Sn}{v},$$

где: S - число внесенных ошибок, v - число найденных внесенных ошибок, n - число найденных реальных ошибок.

Модель можно использовать для получения оценок безошибочности программы. Пусть тестирование выполняется до тех пор, пока не будут найдены все внесенные ошибки S=v. Тогда, достоверность утверждения, что в программе k ошибок, имеет следующий вид:

$$R(k,S) = \begin{cases} 1, при\ n > k, \\ \frac{S}{S+k+1}, при\ n \le k. \end{cases}$$

В случае, если в процессе испытаний выявлены не все внесенные тестовые ошибки, следует использовать следующую формулу:

$$R(k,v) = \begin{cases} 1, n > k, \\ \frac{C_S^{v-1} n}{C_{S+k+1}^{k+v}}, n \le k, \end{cases}$$

где v - число обнаруженных тестовых ошибок.

На практике данная модель применяется для контроля эффективности работы экспертов, проводящих аудит безопасности кода. К недостаткам модели относят проблему случайного внесения уязвимостей.

1.2. Модель учета внесения ошибок в разные модули

В данной модели ПО разделяется на две части. Считается, что 1-ая часть ПО содержит N_1 оставшихся ошибок, вторая часть - N_2, общее число оставшихся ошибок ПО $N = N_1 + N_2$. Предполагается, что выявление оставшихся ошибок равновероятно, обнаруживается одна ошибка в заданный интервал времени и исправляется после обнаружения.

Можно показать, что:

$$\begin{cases} N_1(i) = N_1 - i + \sum_{j=0}^{i} \chi_j, \\ N_2(i) = N_2 - \sum_{j=0}^{i} \chi_j, \end{cases}$$

где $\chi_j -$ характеристическая функция, такая что:

$$\chi_j = \begin{cases} 0, \text{если j} - \text{ая ошибка найдена в } 1 - \text{ой части ПО;} \\ 1, \text{если j} - \text{ая ошибка найдена во } 2 - \text{ой части ПО.} \end{cases}$$

В этом случае можно рассчитать вероятности обнаружения ошибки на заданном интервале времени тестирования $(t_i, t_{i+1}]$:

$$\begin{cases} p_1(i) = \frac{N_1(i)}{N_1(i)+N_2(i)} = \frac{N_1 - i + \sum_{j=0}^{i} \chi_j}{N_1 - i + N_2}, \\ p_2(i) = \frac{N_2(i)}{N_1(i)+N_2(i)} = \frac{N_2 - \sum_{j=0}^{i} \chi_j}{N_1 - i + N_2}. \end{cases}$$

Параметры N_1 и N_2 можно найти, используя метод максимального правдоподобия.

1.3. Модель контроля функциональных объектов

Данная модель используется аккредитованной лабораторией «Эшелон» [7] при проведении испытаний на отсутствие недекларированных возможностей. Согласно руководящему документу Гостехкомиссии России, в процессе динамического анализа ПО контролируется p заданный процент от $M_{фо}$ идентифицированных функциональных объектов. В процессе статического анализа произвольно выбираются $m_{фо}$ функциональных объектов, в которые вносятся тестовые ошибки. При тестировании фиксируются найденные тестовые ошибки - s и найденные реальные ошибки - n. По аналогии с подходом Басина [15], используя метод максимального правдоподобия, можно получить оценку числа ошибок в ПО:

$$N = n \frac{M_{фо} - m_{фо} + 1}{\frac{p}{100} M_{фо} - s}.$$

1.4. Модель испытаний независимыми группами

Групповая модель испытания предполагает проведение тестирования 2-я независимыми группами тестирования или экспертами.

В процессе тестирования подсчитывается количество обнаруженных ошибок обеими группами - N_1 и N_2, а также число обнаруженных обеими группами совпавших ошибок -N_{12}.

Обозначив N как число первоначальных ошибок, можно определить эффективность тестирования каждой группы: $E_1 = N_1/N$ и $E_2 = N_2/N$. Гипотетически предполагая одинаковую эффективность тестирования 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. Это позволяет получить следующее равенство:

$N_1/N=N_{12}/N_2$.

В этом случае интуитивная модель оценки числа первоначальных ошибок в ПО будет иметь следующий вид:

$N = N_1N_2/N_{12}$.

Данная модель полезна на практике, когда тестирование параллельно проводит группа экспертов, имеющих собственные АРМ тестирования, что часто бывает при выездных испытаниях при ограничениях по времени работы.

2. Модели сложности программного обеспечения

Модели сложности ПО основаны на гипотезе о том, что уровень безошибочности ПО может быть предсказан с помощью показателей (метрик) сложности ПО [19]. Это справедливо для непреднамеренных уязвимостей, так как, чем сложнее и больше программа, тем выше вероятность того, что программист ошибется при ее написании и модификации.

В качестве аргументов моделей, как правило, используются метрики сложности ПО, а сами модели сложности можно разделить на априорные и статистические.

2.1. Метрическая модель ошибок Холстеда

Наиболее известной априорной моделью сложности ПО является модель Холстеда (Halstead). В основу разработки модели положены две базовые характеристики ПО: η словарь операторов и операндов языка программирования и N число использования операторов и операндов в программных реализациях, а также гипотеза, что частота использования операторов и операндов в программе пропорциональна двоичному логарифму количества их типов (по аналогии с теорией информации).

Полагаясь на общие статистические физиологические характеристики человека при реализации программы, описанной данными характеристиками, получен ряд эмпирических моделей оценки показателей качества ПО.

Сложность программы предложено рассматривать как совокупность интеллектуальных усилий (решения элементарных задач человеком до возникновения ошибки) при кодировании текста на определенном языке программирования:

$$E = \widetilde{N}\log_2(\eta/L) = \frac{\eta_1 N_2 N \log_2 \eta}{2\eta_2},$$

где $\widetilde{N} = \eta_1 \log_2 \eta_1 + \eta_2 \log_2 \eta_2$ – теоретическая длина программы, $\eta = \eta_1 + \eta_2$ - число уникальных операторов и операндов языка программирования, $L = 2\eta_2/(\eta_1 N_2)$ - качество программирования, $N = N_1 + N_2$ – число обращений к операторам и операндам в ПО.

Для расчета D_0 первоначального числа ошибок в ПО предложено использовать выражение:

$$D_0 = \frac{V}{3000},$$

где $V = N\log_2(\eta_1 + \eta_2)$ - объем программы (в бит информации).

Известна также статистическая модель Холстеда для расчета числа ошибок на этапах отладки и испытаний в ПО:

$$D_m = \kappa_m E^{\frac{2}{3}},$$

где κ_m - коэффициент пропорциональности (например, для майнфреймов $\kappa_m = \frac{1}{3200}$).

2.2. Многофакторная модель сложности

К простым статистическим моделям сложности можно отнести феноменологическую модель фирмы TRW [5]. Феноменологическая модель представляет собой линейную модель оценки показателя сложности ПО по 5-ти эмпирическим характеристикам программ, а именно: логической сложности L_{tot}, сложности взаимосвязей C_{inf}, сложности вычислений C_c, сложности ввода вывода C_{io} и понятности U_{read}:

$$C = L_{tot} + 0.1C_{inf} + 0.2C_c + 0.4C_{io} + (-0.1)U_{read}.$$

Для расчета числа ошибок N предлагается использовать следующую многофакторную модель:

$$N = L_{tot}\kappa_1 + 0.1C_{inf}\kappa_2 + 0.2C_c\kappa_3 + 0.4C_{io}\kappa_4 + (-0.1)U_{read}\kappa_5,$$

где: κ_i- коэффициент корреляции числа ошибок с i-ым показателем сложности.

Значения коэффициентов κ_i легко найти методом наименьших квадратов.

К недостаткам многофакторных моделей относят возможность получения эффекта «проклятья размерности» при большом количестве факторов или нелинейном виде аппроксимирующего полинома.

3. Модели роста надежности от времени

Модели данного класса относят к вероятностным динамическим моделям дискретных систем с непрерывным или дискретным временем. Большинство популярных моделей данного класса можно свести к Марковским однородным, неоднородным или полумарковским моделям массового обслуживания.

В однородных Марковских моделях полагается, что общее число ошибок ПО – неизвестная конечная постоянная величина. Число ошибок, оставшихся в ПО в процессе

тестирования и отладки, описывается экспоненциальным законом распределения. Интенсивность ошибок $\lambda(i)$ зависит от текущего i −состояния системы и не зависит от прошлых состояний.

По сравнению с однородными Марковскими моделями в полумарковских моделях полагается, что $\lambda(t_i)$ интенсивность ошибок зависит не только от числа оставшихся ошибок в ПО, но и от t_i времени в этом состоянии.

В настоящее время все более популярным классом временных моделей становятся неоднородные Марковские модели. В данных моделях общее число ошибок ПО является случайной величиной, описываемой Пуассоновским законом распределения, причем интенсивность потока ошибок не является линейной функцией от времени. По этой причине модели часто называют Пуассоновскими (Non-Homogeneous Poisson Process model, NHPP-модели). В зависимости от вида функции интенсивности Пуассоновского потока NHPP-модели разделяют на выпуклые (например, с распределением Вейбулла и Парето), S-образные (например, с распределением Гомпертца и Рэлея), бесконечные (например, логарифмическая модель).

Существуют модификации моделей надежности ПО путем применения Байесовского подхода, которые иногда выделяют в отдельный класс моделей [20, 28].

Следует сказать, что для расчета параметров динамических моделей традиционно используется метод максимального правдоподобия, в редких случаях - методы линейной регрессии и кросс-энтропии.

Приведем примеры самых популярных временных моделей роста надежности каждого вида.

3.1. Экспоненциальная модель роста надежности программ

Экспоненциальная модель роста надежности, получившая название модели Елинского-Моранды (Jelinski-Moranda, JM-модель), основана на допущениях, что в процессе тестирования ПО длительность интервалов времени между обнаружением двух ошибок имеет экспоненциальное распределение с интенсивностью отказов, пропорциональной числу необнаруженных ошибок. Все ошибки равновероятны. Каждая обнаруженная ошибка мгновенно устраняется, число оставшихся ошибок уменьшается на единицу.

Функция плотности распределения времени обнаружения *i*-й ошибки, отсчитываемого от момента выявления (i – 1)-ой ошибки, имеет вид:

$$p(t_i) = \lambda_i e^{-\lambda_i t_i},$$

где $\lambda_i = \phi(N - (i-1))$ - интенсивность ошибок, которая пропорциональна числу еще не выявленных ошибок в программе, N - число ошибок, первоначально присутствующих в программе; ϕ - коэффициент пропорциональности, интерпретируемый как интенсивность выявления ошибок, t_i - интервал между *(i-1)*-ой и *i*-ой ошибками.

Полученное выражение позволяет легко найти формулы для вероятности безошибочной работы, вероятности устранения всех ошибок за заданное время, средней наработки на ошибку, среднего времени устранения всех ошибок и др.

Для расчета параметров N и ϕ используется метод максимального правдоподобия.

В табл. 1 представлены примеры популярных Марковских моделей, причем, параметр N интерпретируется в качестве числа первоначальных ошибок в ПО.

Таблица 1.

Марковские модели роста надежности программ

Название модели	Интенсивность ошибок, λ_i
JM-модель [18]	$\phi(N-(i-1))$
Модель Липова [26]	$\phi(N-\sum_{j=1}^{i-1} N_j)$
Xui-модель [30]	$\phi(e^{-k(N-i+1)}-1)$
Shanthikumar-модель [30]	$\phi(N-(i-1))^k$
Bucchianico-модель [12]	$1-\phi^{(N-(i-1))}$

3.2. Рэлеевская модель роста надежности программ

Развитием JM-модели является модель Шика-Волвертона (Shick-Wolverton, SW-модель), в которой полагается, что интенсивность ошибок пропорциональна не только количеству необнаруженных ошибок в ПО, но и интервалу времени отладки:

$\lambda_i = \phi(N-(i-1))t_i$,

где: N - число ошибок, первоначально присутствующих в программе, i-число обнаруженных ошибок, ϕ — коэффициент пропорциональности, интерпретируемый как интенсивность выявления ошибок, t_i - интервал времени между *(i-1)*-ой и i-ой ошибками.

Отсюда выводится распределение Рэлея со следующей функцией плотности:

$p(t_i) = \phi(N-(i-1))t_i e^{-\phi(N-(i-1))\frac{t_i^2}{2}}$,

Для расчета значений параметров N и ϕ модели используется метод максимального правдоподобия.

Примеры популярных полумарковских моделей представлены в таблице 2, причем параметр *N* также интерпретируется как число первоначальных ошибок в ПО.

Таблица 2.

Полумарковские модели роста надежности программ

Название модели	Интенсивность ошибок, λ_i
SW-модель [28]	$\phi(N-(i-1))\,t_i$
Гиперболическая модель [15]	$\phi\big(N-(i-1)\big)(-a\,t_i^2+bt_i+c)$
Sukert-модель [30]	$\phi(N-(i-N_i))\,t_i$
Модифицированная модель Липова [26]	$\phi\left(N-\sum_{j=1}^{i-1}N_j\right)\left(\frac{t_i}{2}+\sum_{j=1}^{i-1}t_i\right)$

3.3. S-образная NHPP-модель роста надежности программ

В настоящее время одной из самых популярных моделей роста надежности является S-образная NHPP-модель Ямады (Yamada). В модели полагается, что количество ошибок, проявляющихся в единицу времени, - независимые случайные величины, распределенные по закону Пуассона с интенсивностью потока, пропорциональной ожидаемому числу остающихся в программе ошибок на заданный момент времени.

В отличие от JM- и SW-подобных выпуклых моделей, здесь делается дополнительное предположение о *S*-образной зависимости числа ошибок от времени тестирования. Понятийно *S*-образная зависимость числа обнаруженных ошибок от времени объясняется тем, что в начальной стадии тестирования имеется фаза изучения экспертом ПО.

Функция количества ошибок задается следующей формулой:

$$m(t)=a(1-(1+gt)e^{-gt}),$$

где a - коэффициент, характеризуемый число ожидаемых ошибок в ПО, g – коэффициент интенсивности выявления ошибок.

Соответственно интенсивность возникновения ошибки определяется следующим образом:

$$\lambda(t)=ag^2te^{-gt}.$$

Полученные выражения позволяют легко найти формулы для вероятностей того, что за заданное время будет выявлено и локализовано (или нет) то или иное количество ошибок.

Параметры модели a и g можно найти с помощью метода максимального правдоподобия.

Примеры популярных NHPP-моделей представлены в таблице 3.

Таблица 3.

Неоднородные Марковские модели роста надежности программ

Название NHPP-модели	Функция количества ошибок, $m(t)$
Duane-модель [18]	at^g
Модель Гомпертца [14]	ag^{c^t}
Goel-Okumoto - модель [30]	$a(1-e^{-gt})$
Schneidewind-модель [18]	$a/g(1-e^{-gt})$
Модель Вейбулла [16]	$a(1-e^{-gt^c})$
Yamada-экспоненциальная модель [30]	$a(1-e^{-r\,c(1-e^{(-gt)})})$

S-образная модель Релея [27]	$a(1-e^{-r\,c(1-e^{(-\frac{gt^2}{2})})})$
S-образная модель с задержкой [18]	$a(1-(1+gt)e^{-gt})$
S-образная модель с точкой перегиба [20]	$\frac{a(1-e^{-gt})}{1+ce^{-gt}}$
Параметризованная S-изогнутая модель [15]	$a\frac{1-e^{-gt}}{1+\psi(r)e^{-gt}}$
Dohiya – модель [16]	$a(1-e^{-gt})/(1+e^{-gt})$
Модель Парето [15]	$a(1-(1+t/c)^{1-g})$
Гиперэкспоненциальная модель [20]	$a\left(1-\sum_{i=1}^{2} b_i\, e^{-g_i t}\right)$
Littlewood -модель [15]	$ac^g(\frac{1}{c^g}-\frac{1}{(c+t)^g})$
Параболическая модель [15]	$a(1-e^{-((\frac{l}{3})t^3+(\frac{m}{2})t^2+nt)})$
Логистическая модель [17]	$\frac{a}{1+ke^{-gt}}$
Pham - модель [27]	$a-ae^{-gt}(1+(g+d)t+gdt^2$
Zhang - модель [14]	$\frac{a}{p-\beta}\left((1-\frac{(1+\alpha)e^{-gt}}{1+\alpha e^{-gt}})^{\frac{c}{g}(p-\beta)}\right)$
Xie-логарифмическая модель [15]	$a\,ln^g(1+t)$
Musa-Okumoto-логарифмическая модель [18]	$1/a\,ln(agt+1)$

4. Отладочные модели программ

В основе отладочных моделей лежит утверждение, что свойство безошибочности программы меняется только при ее доработках, а измеряется путем прогона программы на заданных входных данных. Такие модели часто называют моделями надежности, основанными на областях входных данных (data-domain models). Условно отладочные модели можно разделить на ориентированные на покрытие входных данных и ориентированные на величину доработки.

4.1. Структурная модель Нельсона

Модель, получившая название модели Нельсона (Nelson) [5], является биноминальной моделью Бернулли с наложенными правилами по использованию входных данных.

В частности, область входных данных ПО задается в виде k непересекающихся областей - $\{Z_i\}$, которым однозначно соответствует множество вероятностей $\{p_i\}$ того, что соответствующий набор данных будет выбран для очередного прогона ПО. Таким образом, если при выполнении N_i прогонов программы (на Z_i наборе входных данных) n_i из них закончились отказом, то степень надежности функционирования ПО определяется выражением:

$$P = 1 - \sum_{i=1}^{k} \frac{n_i}{N_i} p_i.$$

Модель позволяет рассчитать вероятность P_u безотказного выполнения программы n-прогонов программы:

$$P_u = \prod_{j=1}^{u}(1 - Q_j) = e^{\left(\sum_{j=1}^{u} \ln(1-Q_j)\right)},$$

где: $Q_j = \sum_{i=1}^{k} p_{ji}\chi_i$, χ_i - характеристическая функция отказа на i-ом наборе данных; p_{ji} - вероятность появления i-го набора в j-ом прогоне.

Заметим, в структурной модификации модели Нельсона предлагается для нахождения p_{ji} использовать анализ графа ПО. К сожалению, проведение анализа структурно-сложного модифицируемого ПО для решения указанных задач на практике не представляется эффективным. К недостаткам модели также относят требования по большому количеству испытаний для получения точных оценок. Однако данная категория трудностей решается путем применения статистического метода Вальда [8].

Можно продемонстрировать переход от моделей отладки к временным моделям. Полагая, что Δt_j- время выполнения j-го прогона, можно получить следующее расчетное выражение:

$$P_u = e^{(\sum_{j=1}^{u} \lambda(t_j)\Delta t_j)},$$

где: $\lambda(t_j) = \frac{-\ln(1-Q_j)}{\Delta t_j}$ - интенсивность отказов, $t_j = \sum_{i=1}^{j} \Delta t_i)$ - суммарное время выполнения j-прогонов ПО.

Полагая, что Δt_i становится относительно малой величиной с ростом u-числа испытаний, имеем:

$$P(t) = e^{-\int_0^t \lambda(z)dz}.$$

4.2. Немонотонная модель отладки и обновлений программного обеспечения

В основе модели положено свойство ПО изменения надежности только в моменты доработки ПО, причем степень надежности может как повышаться, так и понижаться:

$$P_u = P_0 + \sum_{j=1}^{u} \Delta P_j,$$

где u - число проведенных доработок ПО, ΔP_j - приращение степени надежности после j-ой доработки.

Для учета эффективности доработки вводится k_{ij} метрика величины измененного кода, например, при исправлении ошибок или обновлении ПО. Это позволяет получить основное расчетное выражение для показателя надежности ПО [3]:

$$P_u = P_\infty - (P_\infty - P_0)\prod_{j=0}^{u}(1 - \sum_{i=1}^{2} a_i k_{ij} / P_\infty),$$

где a_i - коэффициент эффективности доработки ПО с целью исправления ошибки или обновления, P_0 - начальная степень надежности, P_∞ - предельная степень надежности.

Данная модель зависит от 4-х параметров (P_0, P_∞, a_1, a_2) , расчет которых удобно осуществить с помощью метода максимального правдоподобия.

Следует указать, что модель позволяет вывести формулы для планирования испытаний, например, число оставшихся ошибок после u-ой доработки можно рассчитать следующим образом:

$$N_u = ceil(\frac{\ln(\frac{1-P_0}{1-P_u})}{\ln(1-a)}),$$

где a – усредненный коэффициент эффективности доработки ПО.

5. Выбор модели оценки и планирования испытаний

Отметим, что не существует универсальной модели оценки и планирования испытаний ПО. Более того, кроме рассмотренных классов моделей в литературе можно встретить имитационные модели, структурные, нечеткие, интервальные модели, модели динамической сложности программ, модели программно-аппаратных комплексов, а также нейронные сети, получившие применение для решения отдельных научных задач. Для выбора подходящих моделей можно предложить ряд качественных и количественных критериев.

Качественными критериями можно назвать следующие:

1. Простота использования. В первую очередь касается степени адекватности модели системе сбора статистики, т.е. используемые входные данные могут быть легко получены, они должны быть представительны, входные и выходные данные понятны экспертам.

2. Достоверность, т.е. модель должна обладать разумной точностью, необходимой для решения задач анализа или синтеза в области безопасности ПО. Положительным свойством модели является возможность использования априорной информации и комплексирования данных других моделей с целью сокращения входной выборки.

3. Применимость для решения различных задач. Некоторые модели позволяют получить оценки широко спектра показателей, необходимых экспертам на различных этапах жизненного цикла ПО, например, показатели надежности, ожидаемое число ошибок различных типов, прогнозируемые временные и стоимостные затраты, квалификацию специалистов, качество тестов, показатели точности, показатели покрытия ПО и др.

4. Простота реализации, в том числе, возможность автоматизируемости процесса оценки на базе известных математических пакетов и библиотек, переобучения модели после доработок, учета случаев неполных или некорректных входных статистик, учета других ограничений моделей.

В качестве количественных критериев используют следующие:

- показатели точности оценки;

- показатели качества прогнозирующих моделей (сходимость, устойчивость к шуму, точность предсказания, согласованность);

- информационные критерии качества прогнозирующих моделей (размерность, критерии BIC/AIC) [20].

- комбинированные и интегральные показатели, например:

$IC = \max \sum_{i=1}^{K} k_i \chi_i$,

где k_i– весовой коэффициент i -го свойства рассматриваемой модели, выбираемой экспертом; χ_i.- характеристическая функция i-го свойства.

Выводы

Исследование показало, что существует весьма большое количество математических моделей, позволяющих получить оценки показателей технологической безопасности ПО на различных этапах жизненного цикла, что важно при планировании затрат на информационную безопасность. Рассмотренная классификация моделей позволяет на практике сориентироваться при выборе и комплексировании моделей в зависимости от имеющейся статистики.

Надо понимать, что в виду динамичности, сложности и разнородности современных программных проектов, к указанным моделям не могут предъявляться высокие требования по точности, и они носят зачастую интуитивный характер при принятии решений по планированию тестирования ПО на всем множестве входных данных. Несмотря на это, результаты применения моделей удобно использовать как при обосновании трудоемкости испытаний, так и при предоставлении отчетных материалов, что повышает уверенность заказчика в результатах выполненных работ.

Литература

1. Василенко Н.В., Макаров В.А. Модели оценки надежности программного обеспечения. // Вестник НГУ. 2008. № 28. С.126-132.

2. Волканов Д.Ю., Зорин Д.А. Исследование применимости моделей оценки надёжности для разработки программного обеспечения с открытым исходным кодом. // Прикладная информатика. 2011. №2. С.26-32.

3. Марков А.С., Годердзишвили Г.М. Модель оценки степени отлаженности программного обеспечения по результатам испытаний // Методы и средства управления и контроля. Л.: МО СССР, 1987. С. 51-52.

4. Марков А.С., Миронов С.В., Цирлов В.Л. Выявление уязвимостей в программном коде. // Открытые системы. СУБД, 2005. № 12. С.64-69. URL: http://www.osp.ru/os/2005/12/380655/

5. Надежность программного обеспечения. / Т. Тейер, М. Липов, Э. Нельсон. М.: Мир, 1981. 326 с.

6. Одарущенко О.Н., Руденко А.А., Харченко В.С. Учет вторичных дефектов в моделях надежности программных средств // Математические машины и системы, 2010, № 1. С.205-217.

7. Сводный отчёт по безопасности программного обеспечения в России и мире за 2010 год / Фадин А.А. и др.; Под.ред. А.С.Маркова и В.Л.Цирлова. М.: НПО «Эшелон». 2011. 34 с.

8. Смагин В.А. Основы теории надежности программного обеспечения. - СПб.: ВКА им.А.Ф.Можайского, 2009. 336 с.

9. Черкесов Г. Н. Надежность аппаратно-программных комплексов. СПб.: «Питер», 2005. 479 с.

10. Andersson B, Persson M. Software Reliability Prediction – An Evaluation of a Novel Technique. SEBIT, 2004. 32 p.

11. Bubnov V., Tyrva A., Khomonenko A. Model of reliability of the software with Coxian distribution of length of intervals between the moments of detection of errors // Proceedings of 34th Annual IEEE Computer Software and Applications Conference COMPSAC-2010, 2010. P. 238-243.

12. Bucchianico A.D., Groote J.F., van Hee K.M., Kruidhof. R. Statistical Certification of Software Systems. // Communications in Statistics - Simulation and Computation. 2008. Vol. 37, № 2. P.346-359.

13. Cai K.Y. Towards a conceptual framework of software run reliability modeling. // Information Sciences. 2000. Vol. 126, N 1-4. P. 137-163.

14. Garg R.P., Sharma K., Kumar R., Garg R.K. Performance Analysis of Software Reliability Models using Matrix Method. // World Academy of Science, Engineering and Technology. 2010. V. 71. P. 31-38.

15. Gokhale S.S., Marinos P.N., Trivedi K.S. Important Milestones in Software Reliability Modeling // Proc. of Software Engineering and Knowledge Engineering (SEKE 96). Lake Tahoe. 1996. P. 345-352.

16. Hosain Md. S. Software Reliability Using Markov Chain Usage Model. DCSEEBUET, 2005. 90 p.

17. Huang C.-Y., Kuo S.-Y., Lyu M.R. An Assessment of Testing-Effort Dependent Software Reliability Growth Models. // IEEE Transactions on Reliability. 2007. Vol. 56, № 2. P.198-211.

18. IEEE Std. 1633-2008 IEEE Recommended Practice in Software Reliability, 2008. 72 p.

19. IEEE Std. 1061-1998 IEEE Computer Society: Standard for Software Quality Metrics Methodology, 1998. 20 p.

20. Karanta I. Methods and problems of software reliability estimation. VTT WP 63, 2006. 57 p.

21. Kharchenko V.S., Tarasyuk O.M., Sklyar V.V., Dubnitsky V.Yu. The method of software reliability growth models choice using assumptions matrix. // COMPSAC ’02: Proceedings of the 26th International Computer Software and Applications Conference on Prolonging Software Life: Development and Redevelopment. 2002. P. 541–546.

22. Kostogryzov A., Nistratov G., Kleshchev N. Mathematical Models and Software Tools to Support an Assessment of Standard System Processes. // Proceedings of the 6th International SPICE Conference on Process Assessment and Improvement (SPICE-2006), Luxembourg, 2006. P. 63-68.

23. Lipaev V.V. Problems of the development and quality control of large software systems // Programming and computer software. 2005. Т. 31. № 1. С. 47-49.

24. Lyu M. R.-T. Software Reliability Theory. John Wiley & Sons, 2002. 43 p.

25. Musa J. D.. More Reliable Software Faster and Cheaper. 2nd Edition. TATA McGraw-Hill. 2004. 632 p.

26. Neufelder A.M. Ensuring Software Reliability. Marcel Dekker Inc., 1993. 242 p.

27. Pham H. System Software Reliability. Springer Series in Reliability Engineering. Springer, 2006. 440 p.

28. Ramos I.C. Statistical Procedures for Certification of Software Systems. TSIM, 2009. 195 p.

29. Utkin L.V., Zatenko S.I., Coolen F.P.A. New interval Bayesian models for software reliability based on non-homogeneous Poisson processes //Automation and Remote Control. 2010. Т. 71. № 5. P. 935-944.

30. Xie M., Dai Y.-S., Poh K.-L. Computing Systems Reliability. Models and Analysis. Kluwer, 2004. 293 p.

Alexey Markov. Software Testing Models Against Information Security Requirements // «Vestnik of the Bauman Moscow State Technical University» Journal. Series «Instrument», 2011. Special Issue on "Hardware and Information Security Systems". P. 90-103. (In Russian) ISSN 0236-3933.

Марков А.С. Модели оценки и планирования испытаний программных средств по требованиям безопасности информации // Вестник МГТУ им. Н.Э. Баумана. Сер. «Приборостроение», 2011. Специальный выпуск «Технические средства и системы защиты информации». С. 90-103. ISSN 0236-3933.