НАДЕЖНОСТЬ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ЭВМ

- Проблема надежности программного обеспечения (ПО) приобретает все большее значение в связи с постоянным усложнением разрабатываемых систем, расширением круга задач, возлагаемых на них, а следовательно, и значительным увеличением объемов и сложности ПО.

- Реальная надежность ПО нередко оказывается ниже, чем надежность самой аппаратуры.

- Надежность программного обеспечения обуславливается наличием в программах разного рода ошибок, внесенных в нее, как правило, при разработке или в процессе эксплуатации.

- Под ошибкой понимают всякое невыполнение программой заданных функций.

- Проявление ошибки является отказом программы.

Вследствие изложенного проявление ошибки в ПО, является случайным событием, хотя сама ошибка не является случайной.

Факторами, определяющими надежность ПО, являются:

1. подготовка инженерного персонала по технологии использования ЭВМ;
2. контроль выдачи и изменения программ;
3. постоянная связь разработчика и заказчика ПО;
4. применение методов контроля процесса разработки программ и документации;
5. внедрение стандартов, регламентирующих работы по проектированию ПО.

Повышению качества разработки сложных программ способствуют прогрессивные приемы структурного программирования и принцип модульности ПО.

В основу структурного программирования должны быть положены определенные правила:

1. программа должна составляться мелкими шагами;
2. сложная задача должна разбиваться на простые с одним входом и выходом;
3. логика программы должна содержать минимум простых базовых структур;

- **Принцип модульности** заключается в разбиении сложной программы на отдельные подпрограммы - модули, характеризующиеся функциональной законченностью, автономностью и независимостью в разработке и оформлении. Рекомендуется объем модулей в 100-500 команд.
- **Запрет** на применение потенциально ненадежных программных конструкций.
- Для создания надежного ПО используют также **принцип структурирования массивов данных**, что позволяет снизить вероятность появления ошибок из-за их неправильного использования.

Процесс отладки ПО делят на этапы

- **Структурный контроль** соответствия ПО формализованным требованиям применяют на нижних уровнях иерархической структуры ПО - модулях, подпрограммах, отдельных блоках программы.
- **Детерминированное тестирование** предусматривает задание конкретных исходных данных и маршрутов исполнения программы.
- **Восходящее тестирование** начинается с автономного тестирования программных модулей самого нижнего уровня, а **нисходящее тестирование** с автономного тестирования головной программы.

Показатели надежности программного обеспечения

Наиболее распространенными показателями надежности ПО являются следующие:

- ✓ начальное число ошибок $N0$ в ПО после сборки программы и перед ее отладкой;
- ✓ число ошибок n в ПО, обнаруженных и оставшихся после каждого этапа отладки;
- ✓ наработка на отказ T на этапе отладки;
- ✓ необходимое время этапа отладки $\Delta\tau к.о$ на заданное качество (на число оставшихся ошибок);
- ✓ надежность $P0$ ПО при отладке на отсутствие единичной ошибки;
- ✓ эксплуатационная надежность P или коэффициент готовности $КГ$.

Анализ публикаций по надежности ПО дает следующие возможные оценки начального числа ошибок:

- ✓ до отладки ПО, начальное число N_0 составляет 2% от объема ПО в словах;
- ✓ до отладки ПО, начальное число N_0 =1 ошибка на 750 бит текста ассемблера ПО;
- ✓ до отладки ПО, для объема V ПО в битах

$$N_0 = \frac{V}{3000} 2 \log\left(\frac{V}{1300}\right);$$

- ✓ до комплексной отладки, с точностью 20% для ассемблера с объемом V в словах;

$$N_0 = 5 \quad V / 1024;$$

По аналогии с аппаратурной надежностью возможно применение экспоненциальной модели надежности ПО в виде $P(t) = \exp\ (-\Delta t / T)$,

где Δt - время работы;

$$T = \frac{1}{kN_0}\exp(k\tau_{к.о})$$

экспоненциальная модель среднего времени наработки на отказ;

- k - коэффициент темпа выборки ошибок, определяемый по формуле

$$n = N_0[1 - \exp(-k\tau_{к.о})],$$

где n - число обнаруженных и исправленных ошибок ПО; $\tau_{к.о}$ – *время* комплекной отладки программы.

Оценка надежности ПО по ГОСТ 28195-89 рассчитывается по весьма упрощенной методике, констатирующей фактическую надежность по опыту эксплуатации программного комплекса

$$P(t) = 1 - n/N,$$

где n - число отказов при испытаниях ПО;

N - число экспериментов при испытаниях.

Главное отличие между надежностью аппаратуры и ПО: Оборудование отказывает из-за ошибок в проектировании, производстве, эксплуатации, из-за старения. Причины отказа ПО – свойства программы. Причинами отказа могут быть ошибки, вызванные внутренними свойствами ПО или реакцией ПО на изменение внешней среды функционирования. Это означает, что даже при самом тщательном тестировании, если допустить, что удалось избавиться от всех внутренних ошибок, с полной уверенностью нельзя сказать, что при эксплуатации ПО не возникнет отказ.

Основным средством определения количественных характеристик надежности ПО являются *модели надежности*.

Модель надежности – это математическая модель, построенная для оценки зависимости надежности от заранее известных или оцененных в ходе испытания ПО параметров.

В связи с этим определение параметров надежности принято рассматривать в единстве трех процессов: ***предсказание, измерение, оценивание***.

Предсказание – это определение количественных характеристик надежности исходя из характеристик будущего ПО.

Измерение - это определение количественных характеристик надежности, основанное на анализе данных об интервале времени между отказами. Эти данные собираются в ходе тестирования ПО.

Оценивание – это определение количественных характеристик надежности, основанное на данных об интервалах времени между отказами, полученными при испытание ПО в реальных условиях функционирования.

Все модели надежности можно классифицировать по тому, какой из перечисленных процессов они поддерживают.

Модели надежности ПО подразделяют на ***аналитические*** и ***эмпирические.***

Аналитические модели дают возможность рассчитать количественные показатели надежности, основываясь на данных о поведении программы в процессе тестирования (измеряющие и оценивающие модели).

Эмпирические модели базируются на анализе структурных особенностях программы.

Они рассматривают зависимости показателей надежности от числа межмодульных связей, количества циклов в модулях, отношения числа прямолинейных участков программы к количеству точек ветвления и т.д.
Эти модели можно использовать на этапе проектирования
Часто они не дают конечных результатов показателей надежности.
Аналитические модели подразделяются на ***динамические и статические.***
В ***динамических*** моделях поведение ПО рассматривается во времени. А в ***статических*** моделях учитывают только зависимость количества ошибок от числа тестовых прогонов (по области ошибок) или зависимость количества ошибок от характеристики входных данных (по области данных).

При использовании динамических моделей необходимо иметь данные о появлении отказов во времени.
Если фиксируются интервалы каждого отказа, то получается непрерывная картина появления отказов во времени – ***динамические модели с непрерывным временем.***
Если фиксируется число отказов за произвольный интервал времени ***динамические модели с дискретным временем.***

Модель Шумана

(*динамическая модель с дискретным временем*)

Предполагается, что тестирование проводится в несколько этапов. Каждый этап представляет собой выполнение программы на полном комплексе разработанных тестовых данных.

Модель Шумана строится на основе нескольких критериев:

- общее число команд в программе на машинном языке постоянно;
- в начале компоновочных испытаний число ошибок равно некоторой постоянной величине, и по мере исправления ошибок их становится меньше. В ходе испытаний программы новые ошибки не вносятся;
- ошибки изначально различимы, по суммарному числу исправленных ошибок можно судить об оставшихся;
- интенсивность отказов программы пропорциональна числу остаточных ошибок.

Предполагается, что до начала тестирования (т.е. в момент $\tau = 0$) имеется M ошибок. В течение времени тестирования τ обнаруживается $\varepsilon_1(\tau)$ ошибок в расчете на одну команду в машинном языке.

Тогда удельное число ошибок на одну машинную команду, оставшихся в системе после времени тестирования τ, равно:

$$\varepsilon_2(\tau) = \frac{M}{I} - \varepsilon_1(\tau) \qquad (1)$$

где I - общее число машинных команд, которое предполагается постоянным в рамках этапа тестирования.

Предполагается, что значение функции количества ошибок $Z(t)$ пропорционально числу ошибок, оставшихся в программе после израсходованного на тестирование времени τ.

$$Z(t) = C * \varepsilon_2(\tau),$$

где C - некоторая постоянная, t - время работы программы без отказов.

Тогда, если время работы программы без отказа t отсчитывается от точки $t = 0$, а τ остается фиксированным, функция надежности, или вероятность безотказной работы на интервале от 0 до t, равна

$$P(t,\tau) = \exp\left(-C\left(\frac{M}{I} - \varepsilon_1(\tau)\right)\cdot t\right) \qquad (2)$$

$$t_{ср} = \frac{1}{C\cdot\left(\frac{M}{I} - \varepsilon_1(\tau)\right)} \qquad (3)$$

Нам необходимо найти начальное значение ошибок M и коэффициент пропорциональности C. Эти неизвестные оцениваются путем пропуска функционального теста в двух точках переменной оси отладки τ_a и $\tau_в$, выбранных так, что $\varepsilon_1(\tau_a) < \varepsilon_1(\tau_в)$.

В процессе тестирования собирается информация о времени и количестве ошибок на каждом прогоне, т.е. общее время тестирования τ складывается из времени каждого прогона:

$$\tau = \tau_1 + \tau_2 + \tau_3 + \ldots + \tau_n.$$

Предполагая, что интенсивность появления ошибок постоянна и равна λ, можно вычислить ее как число ошибок в единицу времени,

$$\lambda = \frac{\sum_{i=1}^{n} A_i}{\tau} \quad (4)$$

где A_i - количество ошибок на i - ом прогоне.

Тогда

$$t_{cp} = \frac{\tau}{\sum_{i=1}^{n} A_i} \quad (5)$$

Имея данные для двух различных моментов тестирования τ_a и τ_b, можно сопоставить уравнения (3) при τ_a и τ_b:

$$\frac{1}{\lambda_a} = \frac{1}{C \cdot \left(\frac{M}{I} - \varepsilon_1(\tau_a) \right)} \quad (6) \qquad \frac{1}{\lambda_b} = \frac{1}{C \cdot \left(\frac{M}{I} - \varepsilon_1(\tau_b) \right)} \quad (7)$$

Из соотношений (6) и (7) найдем неизвестный параметр C и M:

$$M^* = I \cdot \frac{\frac{\lambda_b}{\lambda_a} \cdot \varepsilon_1(\tau_a) - \varepsilon_1(\tau_b)}{\frac{\lambda_b}{\lambda_a} - 1} \quad (8)$$

$$C^* = \frac{\lambda_{\tau_a}}{\frac{M^*}{I} - \varepsilon_1(\tau_a)} \quad (9)$$

Получив неизвестные M^* и C^*, можно рассчитать надежность программы по формуле (2).

Статическая 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,

$$N = \frac{S * n}{V},$$

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.

Вторая часть модели связана с выдвижением и проверкой гипотез об N.

Пусть в программе имеется не более k собственных ошибок, и внесем в нее еще S ошибок. Программа тестируется, пока не будут найдены все внесенные ошибки (собственные ошибки фиксируются, пусть их n). Уровень значимости C

$$C = \begin{cases} 1, & n > k \\ \dfrac{S}{S+k+1}, & n \le k \end{cases}$$

Таким образом, величина *C* является мерой доверия к модели и показывает вероятность того, насколько правильно найдено значение *N*. Эти два связанных между собой по смыслу соотношения образуют полезную модель ошибок : первое предсказывает возможное первоначально имевшихся в программе ошибок , а второе используется для установления доверительного уровня прогноза. Однако формула для расчета *C* не может быть в случае, когда не обнаружены все искусственно внесенные ошибки. Для этого случая, когда оценка надежности производиться до момента обнаружения всех *S* внесенных ошибок, величина *C* рассчитывается по модифицируемой формуле

$$\begin{cases} C := 1 \ , \ \text{если } n > к \\ C := \dfrac{\dfrac{S}{v - 1}}{\left[\dfrac{S + к + 1}{(v + к)}\right]} \ , & \text{если } n <= к \end{cases}$$

где числитель и знаменатель формулы при $n <= k$ являются биноминальными коэффициентами вида

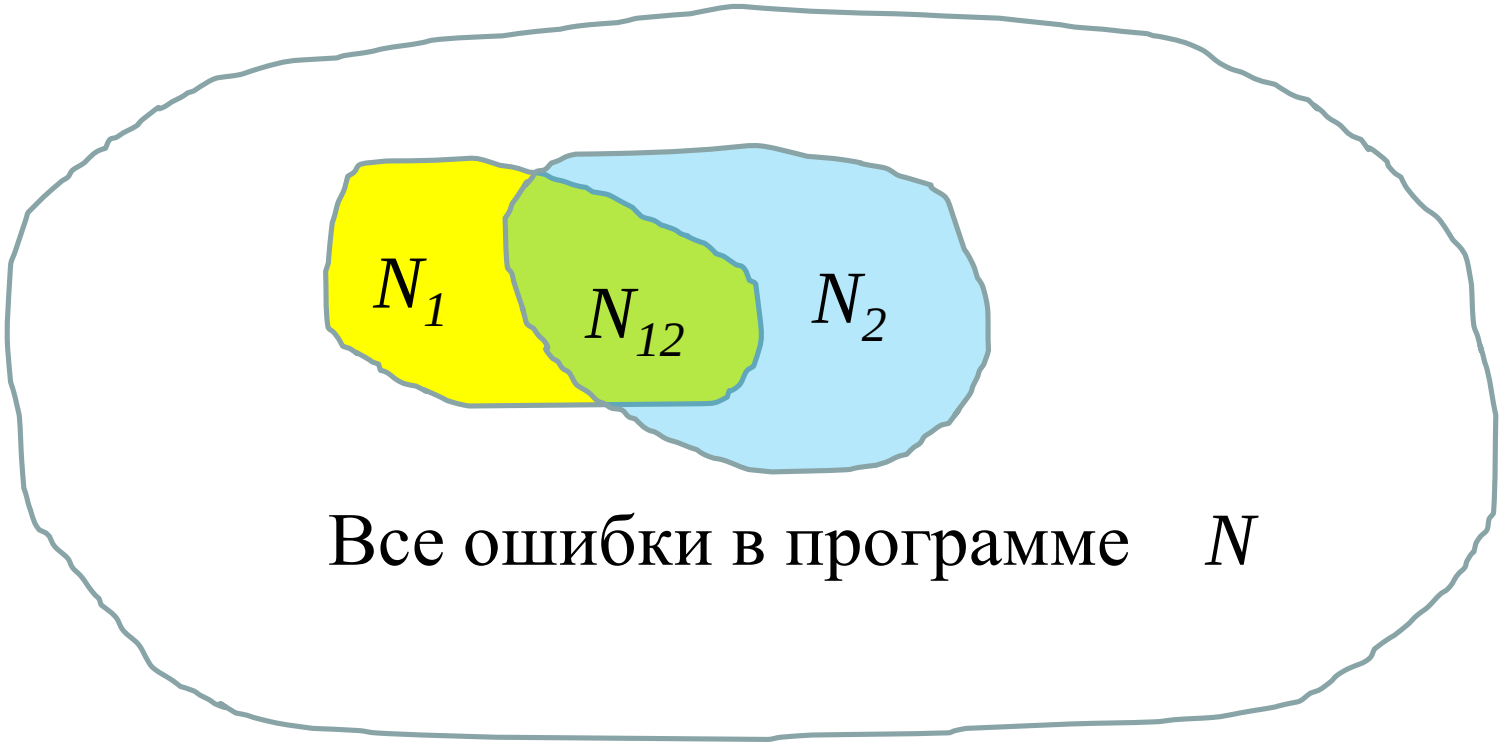
$$\frac{a}{b} := \frac{a!}{b! \cdot (a - b)!} \ .$$

Например, если утверждается, что в программе нет ошибок, а к моменту оценки надежности обнаруженно 5 из 10 рассеянных ошибок и не обнаружено ни одной собственной ошибки, то вероятность того, что в программе действительно нет ошибок, будет равна:

$$C := \frac{\frac{10}{4}}{\frac{11}{5}} = \frac{10! \cdot 5! \cdot 6!}{4! \cdot 6! \cdot 11!} = 0.45 .$$

Если при тех же исходных условиях оценка надежности производится в момент, когда обнаружены 8 из 10 искусственных ошибок, то вероятность того, что в программе не было ошибок, увеличивается до 0.73. В действительности модель Миллса можно использовать для оценки N после каждой найденной ошибки. Предлагается во время всего периода тестирования отмечать на графике число найденных ошибок и текущее значение для N. Достоинством модели является простота применения математического аппарата, наглядность и возможность использования в процессе тестирования.

Однако она не лишена и ряда недостатков, самые существенные из которых – это необходимость внесения искусственных ошибок (этот процесс плохо формализуется) и достаточно вольное допущения величины k, которое основывается исключительно на интуиции и опыте человека, проводящего оценку, т.е. допускается большое влияние субъективного фактора.


N_1
N_{12}
N_2
Все ошибки в программе N

Простая интуитивная модель

Использование этой модели предполагает проведение тестирования двумя группами программистов независимо друг от друга. Они должны использовать независимые тесовые наборы. 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. Получается, что 1-я группа обнаружила N_1 ошибок, 2-я – N_2, а N_{12}– это ошибки, обнаруженные обеими группами. Если обозначить через N неизвестное количество ошибок, присутствовавших в программе до начала тестирования, то можно эффективность тестирования каждой из групп определить как

$$E_1 = N_1 / N; \quad E_2 = N_2 / N .$$

Предположив, что возможность обнаружения для всех ошибок одинаковая, 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, что

$E_1 = N_1 / N = N_{12} / N_2$.

$N_2 = E_2 N$, тогда $N = N_{12} / (E_1 E_2)$ и $N = N_1 N_2 / N_{12}$.

Тип ошибки	Вероятность появления ошибки
Ошибки вычислений	0,09
Логические ошибки	0,26
Ошибки ввода-вывода	0,16
Ошибки манипулирования данными	0,18
Ошибки сопряжения	0,17
Ошибки определения данных	0,08
Ошибки в БД	0,06