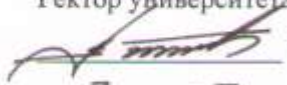




**МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ  
РОССИЙСКОЙ ФЕДЕРАЦИИ**  
Брянский государственный технический университет

Утверждаю

Ректор университета

 О.Н. Федонин  
« 17 » 07 2017г.

**МЕТРОЛОГИЯ  
И КАЧЕСТВО ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ**

**ОЦЕНКА НАДЕЖНОСТИ ПРОГРАММНЫХ СРЕДСТВ**

Методические указания  
к выполнению лабораторной работы №12, 13, 14, 15  
для студентов очной формы обучения  
по направлениям подготовки  
09.03.01 «Информатика и вычислительная техника»  
09.03.04 «Программная инженерия»  
02.03.03 «Математическое обеспечение и администрирование  
информационных систем»

Брянск 2017



---

**МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ  
РОССИЙСКОЙ ФЕДЕРАЦИИ**  
Брянский государственный технический университет

---

Утверждаю

Ректор университета

\_\_\_\_\_ О.Н. Федонин

« \_\_\_\_\_ » \_\_\_\_\_ 2017г.

**МЕТРОЛОГИЯ  
И КАЧЕСТВО ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ**

**ОЦЕНКА НАДЕЖНОСТИ ПРОГРАММНЫХ СРЕДСТВ**

**Методические указания  
к выполнению лабораторной работы №12, 13, 14, 15  
для студентов очной формы обучения  
по направлениям подготовки  
09.03.01 «Информатика и вычислительная техника»  
09.03.04 «Программная инженерия»  
02.03.03 «Математическое обеспечение и администрирование  
информационных систем»**

**Брянск 2017**

Метрология и качество программного обеспечения. Оценка надежности программных средств. [Электронный ресурс]: методические указания к выполнению лабораторной работы №12, 13, 14, 15 для студентов очной формы обучения по направлению подготовки 09.03.01 «Информатика и вычислительная техника»; 09.03.04 «Программная инженерия»; 02.03.03 «Математическое обеспечение и администрирование информационных систем» – Брянск, 2017. – 46 с.

Разработал

А.А. Азарченков,

канд. техн. наук, доц.

Рекомендовано кафедрой «Информатика и программное обеспечение» БГТУ (протокол № 7 от 01.06.2017г.)

**Методические издания публикуются в авторской редакции**

## ОЦЕНКА НАДЕЖНОСТИ ПРОГРАММНЫХ СРЕДСТВ

Одной из важнейших характеристик качества программного средства является надежность.

**Надежность** – свойство программного средства сохранять работоспособность в течение определенного периода времени, в определенных условиях эксплуатации с учетом последствий для пользователя каждого отказа.

Работоспособным называется такое состояние программного средства, при котором оно способно выполнять заданные функции с параметрами, установленными требованиями технического задания. С переходом в неработоспособное состояние связано событие отказа.

Причиной отказа программного средства является невозможность его полной проверки в процессе тестирования и испытаний. При эксплуатации программного средства в реальных условиях может возникнуть такая комбинация входных данных, которая вызовет отказ, следовательно, работоспособность программного средства зависит от входных данных, и чем меньше эта зависимость, тем выше уровень надежности.

Для оценки надежности используются три группы показателей: качественные, порядковые и количественные.

## КОЛИЧЕСТВЕННЫЕ ПОКАЗАТЕЛИ НАДЕЖНОСТИ ПС

1. Вероятность безотказной работы  $P(t_3)$  - эта вероятность того, что в пределах заданной наработки отказ системы не возникнет.

*Наработка* – продолжительность, или объем работы:

$$P(t_3) = P(t > t_3)$$

где  $t$  – случайное время работы ПС до отказа;  $t_3$  – заданная наработка.

2. Вероятность отказа – вероятность того, что в пределах заданной наработки отказ системы возникает. Этот показатель обратный предыдущему

$$Q(t_3) = 1 - P(t_3).$$

3. Интенсивность отказов системы  $\lambda(t)$  – это условная плотность вероятности возникновения отказа ПС в определенный момент времени при условии, что до этого времени отказ не возник.

$$\lambda(t) = \frac{f(t)}{P(t)},$$

где  $f(t)$  – плотность вероятности отказа в момент времени  $t$ .

Существует следующая связь между  $\lambda(t)$  и  $P(t)$ :

$$f(t) = \frac{dQ(t)}{dt} = \frac{d}{dt}[1 - P(t)] = -\frac{d}{dt}P(t),$$

в частном случае при  $\lambda = \text{const}$   $P(t) = \exp(-\lambda(t))$ .

Если в процессе тестирования фиксируется число отказов за определенный временной интервал, то  $\lambda(t)$  – число отказов в единицу времени.

4. Средняя наработка до отказа  $T_i$  – математическое ожидание времени работы ПС до очередного отказа.

$$T_i = \int_0^{\infty} t f(t) dt,$$

где  $t$  – время работы ПС от  $(k - 1)$  до  $k$ -го отказа

Иначе среднюю наработку на отказ  $T_i$  можно представить:

$$T_i = (t_1 + t_2 + \dots + t_n) / n = \left(\frac{1}{n}\right) \sum_{i=1}^n t_i,$$

где  $t_1$  – время работы ПС между отказами;  $n$  – количество отказов.

5. Среднее время восстановления ТВ – математическое ожидание времени восстановления - времени, затраченного на обнаружение и локализацию отказа  $t_{oli}$ , времени устранения отказа  $t_{yi}$ , времени пропускной проверки работоспособности –  $t_{ppi}$ :

$$t_{\epsilon i} = t_{oli} + t_{yi} + t_{ppi},$$

где  $t_{\epsilon i}$  – время восстановления после  $i$ -го отказа.

$$T_B = \frac{\sum_{i=1}^n t_{\epsilon i}}{n},$$

где  $n$  – количество отказов.

Для этого показателя термин «время» означает время, затраченное специалистом по тестированию на перечисленные виды работ.

6. Коэффициент готовности  $K_{\Gamma}$  – вероятность того, что ПС ожидается в работоспособном состоянии в произвольный момент времени его использования по назначению:

$$K_{\Gamma} = \frac{T_i}{T_i + T_B}.$$

## ПРОГНОЗИРОВАНИЕ НАДЕЖНОСТИ ПС

При проектировании сложных программных средств не всегда удастся провести исчерпывающее тестирование так как количество независимых маршрутов может быть очень велико. Например, если цикл в программе выполняется  $k$  раз, а внутри цикла имеется  $n$  ветвлений, то количество маршрутов вычисляется по формуле

$$m = \sum_{i=1}^k n^i.$$

При  $n = 5$  и  $k = 20$  количество маршрутов  $m = 10^{14}$ . Примем, что на разработку, выполнение и оценку теста по одному маршруту расходуется 1 мс. Тогда при работе 24 часа в сутки 365 дней в году на тестирование уйдет 3170 лет.

Поэтому основным средством определения количественных показателей надежности являются модели надежности, под которыми понимают математическую модель, построенную для оценки зависимости надежности от заранее известных или оцененных в ходе создания программного средства параметров. В связи с этим определение показателей надежности принято рассматривать в единстве трех процессов - предсказание, измерение, оценивание.

**Предсказание** - это определение количественных показателей надежности исходя из характеристик будущего программного средства.

**Измерение** - это определение количественных показателей надежности, основанное на анализе данных об интервалах между отказами, полученных при выполнении программ в условиях тестовых испытаний.

**Оценивание** - это определение количественных показателей надежности, основанное на данных об интервалах между отказами,

полученными при испытании программного средства в реальных условиях функционирования.

Все модели надежности можно классифицировать по тому, какой из перечисленных процессов они поддерживают (предсказывающие, прогнозные, оценивающие, измеряющие) Нужно отметить, что модели надежности, которые в качестве исходной информации используют данные об интервалах между отказами, можно отнести к измеряющим, и к оценивающим в равной степени. Некоторые модели, основанные на информации, полученной в ходе тестирования программного средства дают возможность делать прогнозы поведения программного средства в процессе эксплуатации.

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 особенностей программ. Они рассматривают зависимость показателей надежности от числа межмодульных связей, количества циклов в модулях, отношения количества прямолинейных участков к количеству точек ветвления и тому подобное. Нужно отметить, что часто эмпирические модели не дают конечных результатов показателей надежности.

Аналитическое моделирование надежности программного средства включает четыре шага:

1. определение предложений, связанных с процедурой тестирования программного средства;
2. разработка или выбор аналитической модели, базирующейся на предположениях о процедуре тестирования;
3. выбор параметров моделей с использованием полученных данных;
4. применение модели - расчет количественных показателей надежности по модели.

Аналитические модели представлены двумя группами: динамические и статические модели. В динамических моделях надежности программного средства поведение программы (появление отказов) рассматривается во времени. В статических моделях появление отказов не связывают со временем, а учитывают только зависимость количества ошибок от числа тестовых прогонов (по области ошибок) или зависимость количества ошибок от характеристики входных данных (по области данных). Для использования динамических моделей необходимо иметь данные о

появлении отказов во времени. Статические модели принципиально отличаются от динамических тем, что в них не учитывается время появления ошибок в процессе тестирования и не используется никаких предположений о поведении функции риска  $\lambda(t)$ . Эти модели строятся на твердом статистическом фундаменте.

## МОДЕЛЬ ДЖЕЛИНСКИ - МОРАНДЫ

### Теоретические сведения

Модель Джелински - Моранды (Z. Jclinski, P. Moranda) предложена в 1996 г. Иначе данная модель называется моделью роста надежности. Она основана на предположении об экспоненциальной зависимости плотности вероятности интервалов времени между проявлением ошибок от интенсивности ошибок. Кроме того, в модели полагается, что интенсивность ошибок на каждом случайном интервале времени линейно зависит от количества оставшихся в программе ошибок.

Если допустить, что ошибка после се каждого проявления устраняется и при этом в программный модуль не вносятся новые, то интенсивность ошибок  $\alpha(t_i)$  на интервале  $t_i$ , определяется следующим соотношением:

$$\lambda(t_i) = (N - i + 1) \cdot k,$$

где  $N$  - количество ошибок до начала тестирования;  $i = 1 \dots m$  ( $m$  - число ошибок, обнаруженных во время тестирования);  $k$  - коэффициент пропорциональности.

Для плотности вероятности ошибки  $p(t_i)$  на случайном интервале  $t_i$  справедливо выражение:

$$p(t_i) = k \cdot (N - i + 1) \cdot e^{-(N - i + 1) \cdot k \cdot t_i}.$$

Применяя для двух неизвестных этого уравнения (ими являются величины  $k$  и  $N$ ) метод максимального правдоподобия, авторы этого метода оценки надежности программных средств получили следующую систему из двух уравнений:

$$\begin{cases} \sum_{i=1}^n \frac{1}{(N - i + 1)} = k \cdot \sum_{i=1}^n t_i \\ k = \frac{n}{N \cdot \sum_{i=1}^n t_i - \sum_{i=1}^n (i-1) \cdot t_i} \end{cases}$$

где  $n$  - количество обнаруженных в процессе тестирования ошибок.



Уравнения, образующие систему, представляют модель Джелински — Моранды, позволяющей оценить количество ошибок в программе до начала тестирования  $N$  по количеству обнаруженных ошибок  $n$ . Для упрощения расчетов считают, что каждый тест может обнаружить только одну ошибку. Продолжительность интервала тестирования  $t_i$  измеряется не в единицах времени, а количеством тестов, которое потребовалось для обнаружения очередной ошибки. Таким образом, если очередная ошибка обнаруживается одним тестом, то интервал времени равен единице. Если ошибка обнаруживается  $m$  тестами (т. е. тест с номером  $m - 1$  не обнаружил ошибки, в тесте  $m$  ошибка была обнаружена), то интервал времени равен  $m$  (рис.1).

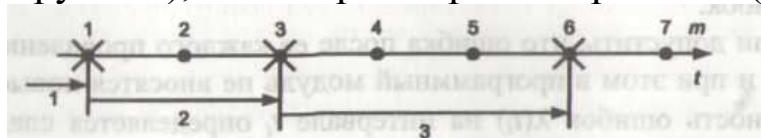


Рис. 1

Важным условием применимости на практике модели Джелински - Моранды является соответствие результатов тестирования допущению об уменьшении интенсивности ошибок после устранения очередной ошибки. Подтверждением этого соответствия должно быть увеличение интервалов времени (количества тестов) для обнаружения каждой последующей ошибки. Таким образом, модель Джелински - Моранды основывается на соблюдении следующих условий:

- экспоненциальная зависимость плотности вероятности интервалов времени между появлением ошибок;
- интенсивность ошибок линейно зависит от количества оставшихся ошибок на любом случайном интервале;
- каждый тест находит только одну ошибку;
- после каждого появления ошибка устраняется и не вносится новая ошибка.

В силу того, что с течением времени интенсивность ошибок уменьшается и растет интервал между проявлением ошибок, модель Джелински - Моранды в некоторых источниках называют моделью роста надежности.

### Задачи по применению модели Джелински - Моранды

#### Определение количества ошибок до начала тестирования

В результате тестирования программы серией из четырех случайно выбранных из набора тестов обнаружено 2 ошибки. Ошибки

обнаружены первым и третьим тестами. Требуется определить количество ошибок  $n$  в программе до начала тестирования.

### Решение задачи

Модель надежности Джелински Моранды представляет собой систему уравнений. Важнейшим условием применимости этой модели на практике является соответствие результатов тестирования принятому допущению об уменьшении интенсивности ошибок после устранения очередной ошибки.

Свидетельством подтверждения этого соответствия должен быть факт увеличения интервалов времени (количества тестов) для обнаружения каждой последующей ошибки.

Проанализируем исходные данные поставленной задачи:

- общее количество обнаруженных ошибок  $n = 2$ ;
- интервал продолжительности обнаружения первой ошибки  $t_1 = 1$ , так как ошибка обнаружена при проведении одного (причем первого) теста;
- интервал продолжительности обнаружения второй ошибки  $t_2 = 2$  (ошибка обнаружена при проведении третьего теста);
- интервал обнаружения второй ошибки больше интервала обнаружения первой ошибки ( $t_2 > t_1$ ), что не противоречит условию применимости модели Джелински - Моранды.

Таким образом, можно записать:

$$\begin{cases} \sum_{i=1}^n \frac{1}{N-i+1} = k \cdot \sum_{i=1}^n t_i \\ k = \frac{n}{N \cdot \sum_{i=1}^n t_i - \sum_{i=1}^n (i-1) \cdot t_i} \end{cases} \Rightarrow \begin{cases} \frac{1}{N-1+1} + \frac{1}{N-2+1} = k \cdot (1+2) \\ k = \frac{2}{N \cdot (1+2) - ((1-1) \cdot 1 + (2-1) \cdot 2)} \end{cases} \Rightarrow$$

$$\Rightarrow \begin{cases} \frac{1}{N} + \frac{1}{N-1} = k \cdot 3 \\ k = \frac{2}{N \cdot 3 - 2} \end{cases} \Rightarrow \frac{1}{N} + \frac{1}{N-1} = \frac{2}{3 \cdot N - 2} \cdot 3 \Rightarrow \frac{1}{N} + \frac{1}{N-1} = \frac{6}{3 \cdot N - 2}.$$

Полученное уравнение необходимо решить относительно переменной  $N$ .

В результате математических преобразований полученное уравнение приобретает следующий вид:

$$\frac{2-N}{N \cdot (N-1) \cdot (3 \cdot N-2)} = 0$$

из чего следует  $N=2$ .

Таким образом, в соответствии с моделью Джелински - Моранды до начала тестирования в программе содержалось две ошибки.

## Определение количества ошибок в программе, не устраненных после проведения тестирования

В результате тестирования программы серией из четырех случайно выбранных из набора тестов обнаружено 2 ошибки. Ошибки обнаружены первым и четвертым тестами. Все ошибки исправлены сразу после обнаружения. В предположении, что исправление ошибок не повлекло появление новых ошибок, требуется оценить количество оставшихся в программе ошибок. Результаты расчетов округлять в большую или меньшую сторону по стандартным правилам (например, если округлить число 2,3, то получим 2, а если округлить 2,5 или 2,6, то после округления получим 3).

### Решение задачи

Проанализируем исходные данные поставленной задачи в соответствии с моделью Джелински - Моранды:

- общее количество обнаруженных ошибок  $n = 2$ ;
- интервал продолжительности обнаружения первой ошибки  $t_1 = 1$ , так как ошибка обнаружена при проведении одного (причем первого) теста;
- интервал продолжительности обнаружения второй ошибки  $t_2 = 3$ , так как ошибка обнаружена при проведении четвертого теста;
- интервал обнаружения второй ошибки больше интервала обнаружения первой ошибки ( $t_1 > t_2$ ) что не противоречит условию применимости модели Джелински - Моранды.

Таким образом:

$$\begin{aligned} \left\{ \begin{aligned} \sum_{i=1}^n \frac{1}{N-i+1} &= k \cdot \sum_{i=1}^n t_i \\ k &= \frac{n}{N \cdot \sum_{i=1}^n t_i - \sum_{i=1}^n (i-1) \cdot t_i} \end{aligned} \right. \Rightarrow \left\{ \begin{aligned} \frac{1}{N-1+1} + \frac{1}{N-2+1} &= k \cdot (1+3) \\ k &= \frac{2}{N \cdot (1+3) - ((1-1) \cdot 1 + (2-1) \cdot 3)} \end{aligned} \right. \Rightarrow \\ \Rightarrow \left\{ \begin{aligned} \frac{1}{N} + \frac{1}{N-1} &= k \cdot 4 \\ k &= \frac{2}{N \cdot 4 - 3} \end{aligned} \right. \Rightarrow \frac{1}{N} + \frac{1}{N-1} = \frac{2}{4 \cdot N - 3} \cdot 4 \Rightarrow \frac{1}{N} + \frac{1}{N-1} = \frac{8}{4 \cdot N - 3} \end{aligned}$$

Полученное уравнение необходимо решить относительно переменной  $N$ .

В результате математических преобразований полученное уравнение приобретает следующий вид:

$$\frac{3 - 2N}{N(N-1)(4N-3)} = 0$$

из чего следует  $N = 1.5 \approx 2$ .

Таким образом, в соответствии с моделью Джелински - Моранды до начала тестирования в программе содержалось две ошибки, и две ошибки было обнаружено в процессе тестирования. Следовательно, в программе осталось  $N - n = 0$  необнаруженных ошибок.

### Задачи для самостоятельного решения

Задача 1. В результате тестирования программы серией из шести случайно выбранных из набора тестов обнаружено 2 ошибки. Ошибки обнаружены вторым и шестым тестами. Требуется определить количество ошибок  $N$  в программе до начала тестирования.

Задача 2. В результате тестирования программы серией из 10 случайно выбранных из набора тестов обнаружено 2 ошибки. Ошибки обнаружены первым и восьмым тестами. Требуется определить количество ошибок  $N$  в программе до начала тестирования.

Задача 3. В результате тестирования программы серией из 10 случайно выбранных из набора тестов обнаружено 3 ошибки. Ошибки обнаружены первым, третьим и десятым тестами. Требуется определить количество ошибок  $N$  в программе до начала тестирования.

Задача 4. В результате тестирования программы серией из восьми случайно выбранных из набора тестов обнаружено 2 ошибки. Ошибки обнаружены третьим и восьмым тестами. Требуется определить количество ошибок  $N$  в программе до начала тестирования.

Задача 5. В результате тестирования программы серией из 25 случайно выбранных из набора тестов обнаружено 3 ошибки. Ошибки обнаружены четвертым, десятым и двадцать вторым тестами. Требуется определить количество ошибок  $N$  в программе до начала тестирования.

Задача 6. В результате тестирования программы серией из 30 случайно выбранных из набора тестов обнаружено 3 ошибки. Ошибки обнаружены шестым, четырнадцатым и двадцать восьмым тестами. Требуется определить количество ошибок  $N$  в программе до начала тестирования.

Задача 7. В результате тестирования программы серией из 11 случайно выбранных из набора тестов обнаружено 2 ошибки. Ошибки обнаружены четвертым и одиннадцатым тестами. Все ошибки исправлены сразу после обнаружения. В предположении, что исправление ошибок не повлекло появления новых ошибок, требуется оценить количество оставшихся в программе ошибок.

Задача 8. В результате тестирования программы серией из 14 случайно выбранных из набора тестов обнаружено 2 ошибки. Ошибки обнаружены первым, пятым и тринадцатым тестами. Все ошибки

исправлены сразу после обнаружения. В предположении, что исправление ошибок не повлекло появление новых ошибок, требуется оценить количество оставшихся в программе ошибок.

Задача 9. В результате тестирования программы серией из 35 случайно выбранных из набора тестов обнаружено 3 ошибки. Ошибки обнаружены четвертым, десятым и двадцать вторым тестами. Все ошибки исправлены сразу после обнаружения. В предположении, что исправление ошибок не повлекло появление новых ошибок, требуется оценить количество оставшихся в программе ошибок.

Задача 10. В результате тестирования программы серией из 11 случайно выбранных из набора тестов обнаружено 2 ошибки. Ошибки обнаружены третьим и десятым тестами. Все ошибки исправлены сразу после обнаружения. В предположении, что исправление ошибок не повлекло появление новых ошибок, требуется оценить количество оставшихся в программе ошибок.

## СТАТИСТИЧЕСКАЯ МОДЕЛЬ МИЛЛСА

### Теоретические сведения

Статистическая модель Миллса позволяет оценить не только количество ошибок до начала тестирования, но и степень отлаженности программ. Для применения модели до начала тестирования в программу преднамеренно вносят ошибки. Далее считают, что обнаружение преднамеренно внесенных и так называемых собственных ошибок программы равновероятно.

Для оценки количества ошибок в программе до начала тестирования используется выражение

$$N = \frac{W \cdot S}{V},$$

где  $w$  - количество преднамеренно внесенных в программу ошибок до начала тестирования;  $v$  - количество обнаруженных в процессе тестирования ошибок из числа преднамеренно внесенных;  $s$  - количество «собственных» ошибок программы, обнаруженных в процессе тестирования.

Если продолжать тестирование до тех пор, пока все ошибки из числа преднамеренно внесенных не будут обнаружены, степень отлаженности программы  $C$  можно оценить с помощью выражения

$$C = \begin{cases} 1, & \text{если } S > r \\ \frac{W}{W + r + 1}, & \text{если } S \leq r \end{cases},$$



где  $S$  и  $W = V$  (равенство значений  $W$  и  $V$  в данном случае имеет место, поскольку считается, что все преднамеренно внесенные ошибки обнаружены) имеют тот же смысл, что и в предыдущем выражении, а  $r$  означает верхний предел (максимум) предполагаемого количества «собственных» ошибок в программе.

Приведенные выражения представляют собой статистическую модель Миллса. Необходимо заметить, что если тестирование будет закончено преждевременно (т. е. раньше, чем будут обнаружены все преднамеренно внесенные ошибки), то вместо первого выражения следует использовать более сложное второе комбинаторное выражение. Если обнаружено только  $V$  ошибок из  $W$  преднамеренно внесенных, используется выражение

где в круглых скобках записаны обозначения для числа сочетаний из  $S$  элементов по  $V - 1$  элементов в каждой комбинации и числа сочетаний из  $S + r + 1$  элементов по  $r + V$  элементов в каждой комбинации.

$$C = \begin{cases} 1, \text{ если } S > r \\ \frac{C_V^{S-1}}{C_{r+V+1}^{S+r}}, \text{ если } S \leq r, V = S \\ \frac{C_{V-1}^S}{C_{r+V}^{S+r+1}}, \text{ если } S \leq r, V - S > 0 \end{cases}$$

## Задачи по применению модели Миллса

### Задача 1

В программу преднамеренно внесли (посеяли) 10 ошибок. В результате тестирования обнаружено 12 ошибок, из которых 10 ошибок были внесены преднамеренно. Все обнаруженные ошибки исправлены. До начала тестирования предполагалось, что программа содержит не более 4 ошибок. Требуется оценить количество ошибок до начала тестирования и степень отлаженности программы.

Нам известно:

- количество внесенных в программу ошибок  $w = 10$ ;
  - количество обнаруженных ошибок из числа внесенных  $v = 10$ ;
  - количество «собственных» ошибок в программе  $s = 12 - 10 = 2$ .
- Подставив указанные значения в формулу, получим оценку количества ошибок:

$$N = (W \cdot S) / V = (10 \cdot 2) / 10 = 2.$$

Таким образом, из результатов тестирования следует, что до начала тестирования в программе имелось 2 ошибки.

Для оценки отлаженности программы используем уравнение. Нам известно:

- количество обнаруженных «собственных» ошибок в программе  $S = 2$ ;
- количество предполагаемых ошибок в программе  $r = 4$ ;
- количество преднамеренно внесенных и обнаруженных ошибок  $W = 10$ .

Очевидно, что обнаружено меньшее число «собственных» ошибок, чем количество предполагаемых ошибок в программе ( $5 < 2$ ). Для оценки отлаженности программы используем уравнение

$$C = W / (W + r + 1) = 10 / (10 + 4 + 1) = 0,67.$$

Степень отлаженности программы равна 0,67, что составляет 67%.

## Задача 2

В программу было преднамеренно внесено (посеяно) 7 ошибок. В результате тестирования обнаружено 11 ошибок, из которых 7 ошибок были внесены преднамеренно. Все обнаруженные ошибки исправлены. До начала тестирования предполагалось, что программа содержит не более 5 ошибок. Требуется оценить количество ошибок до начала тестирования и степень отлаженности программы.

Нам известно:

- количество внесенных в программу ошибок  $W = 7$ ;
- количество обнаруженных ошибок из числа внесенных  $V = 7$ ;
- количество «собственных» ошибок в программе  $S = 11 - 7 = 4$ .

Подставив указанные значения в формулу, получим оценку:

$$N = (W \cdot S) / V = (7 \cdot 4) / 7 = 4.$$

Таким образом, из результатов тестирования следует, что до начала тестирования в программе имелось 4 ошибки.

Для оценки отлаженности программы нам известно:

- количество обнаруженных «собственных» ошибок в программе  $S = 4$ ;
- количество предполагаемых ошибок в программе  $r = 5$ ;
- количество преднамеренно внесенных и обнаруженных ошибок  $W = 7$ .

Очевидно, что обнаружено меньшее число «собственных» ошибок, чем количество предполагаемых ошибок в программе ( $S < r$ ). Для оценки отлаженности программы используем уравнение

$$C = W / (W + r + 1) = 7 / (7 + 5 + 1) = 0,54.$$

Степень отлаженности программы равна 0,54, что составляет 54%.

### Задача 3

В программу было преднамеренно внесено (посеяно) 14 ошибок. В результате тестирования обнаружено 9 ошибок, из которых 6 ошибок были внесены преднамеренно. Все обнаруженные ошибки исправлены. До начала тестирования предполагалось, что программа содержит не более 4 ошибок. Требуется оценить степень отлаженности программы на момент завершения тестирования.

Если тестирование закончено раньше, чем обнаружены все преднамеренно внесенные ошибки, то следует применять сложное комбинаторное выражение.

Из условия задачи нам известно:

- количество внесенных в программу ошибок  $W = 14$ ;
- количество обнаруженных ошибок из числа внесенных  $V = 6$ ;
- количество «собственных» ошибок в программе  $S = 9 - 6 = 3$ ;
- количество предполагаемых ошибок  $r = 4$ .

Поскольку  $S \leq r$  (а именно  $3 < 4$ ), подставим указанные значения и получим оценку:

$$C = \frac{C_{V-1}^S}{C_{r+V}^{S+r+1}} = \frac{C_{6-1}^3}{C_{4+6}^{3+4+1}} = \frac{C_5^3}{C_{10}^8} = \frac{\frac{5!}{3! \cdot (5-3)!}}{\frac{10!}{8! \cdot (10-8)!}} = \frac{5! \cdot 8! \cdot 2!}{10! \cdot 3! \cdot 2!} = \frac{4 \cdot 5}{9 \cdot 10} = 0,22.$$

Таким образом, отлаженность программы составляет 22%.

### Задача 4

В программу было преднамеренно внесено (посеяно) 9 ошибок. В результате тестирования обнаружено 8 ошибок, из которых 6 ошибок были внесены преднамеренно. Все обнаруженные ошибки исправлены. До начала тестирования предполагалось, что программа содержит не более 8 ошибок. Требуется оценить степень отлаженности программы на момент обнаружения указанных выше ошибок.



Если тестирование закончено раньше, чем обнаружены все преднамеренно внесенные ошибки, то следует применять более сложное комбинаторное выражение.

Из условия задачи нам известно:

- количество внесенных в программу ошибок  $W=9$ ;
- количество обнаруженных ошибок из числа внесенных  $V=6$ ;
- количество «собственных» ошибок в программе  $S = 8-6 = 2$ ;
- количество предполагаемых ошибок  $r = 8$ .

Так как  $S < r$  (а именно  $2 < 8$ ), подставим указанные значения и получим оценку:

$$C = \frac{C_{r-1}^S}{C_{r+V}^{S+r+1}} = \frac{C_{8-1}^2}{C_{8+6}^{2+8+1}} = \frac{C_7^2}{C_{14}^{11}} = \frac{\frac{7!}{2! \cdot (5-2)!}}{\frac{14!}{11! \cdot (14-11)!}} = \frac{7! \cdot 11! \cdot 3!}{14! \cdot 2! \cdot 3!} = \frac{3 \cdot 4 \cdot 5}{12 \cdot 13 \cdot 14} = 0,027.$$

Таким образом, отлаженность программы составляет 2,7%.

### Задача 5

В программу было преднамеренно внесено (посеяно) 14 ошибок. Предположим, что в программе перед началом тестирования было 14 ошибок. В процессе четырех тестовых прогонов было выявлено следующее количество ошибок.

Номер прогона	1	2	3	4
$V$	6	4	2	2
$S$	4	2	1	1

Необходимо оценить количество ошибок перед каждым тестовым прогоном. Оценить степень отлаженности программы после последнего прогона. Построить диаграмму зависимости возможного числа ошибок в данной программе от номера тестового прогона.

Количество ошибок перед каждым прогоном будем оценивать в соответствии с простым выражением. Перед каждой последующей оценкой количества ошибок и степени отлаженности программы необходимо корректировать значения внесенных  $W$  и предполагаемых  $r$  ошибок с учетом выявленных и устраненных после каждого прогона тестов. Степень отлаженности программы на всех прогонах, кроме последнего, рассчитывается по комбинаторной формуле.

Определяя показатели программы по результатам первого прогона, необходимо учитывать, что  $W_1 = 14$ ;  $S_1 = 4$ ;  $V_1 = 6$ , тогда

$$N_1 = (W_1 \cdot S_1) / V_1 = (14 \cdot 4) / 6 = 9.$$

По результатам второго прогона корректируем исходные данные для оценки параметров:  $r_2 = 14 - 4 = 10$ ;  $W_2 = 14 - 6 = 8$ ;  $S_2 = 2$ ;  $V_2 = 4$ , следовательно,

$$N_2 = (W_2 \cdot S_2) / V_2 = (8 \cdot 2) / 4 = 4.$$

Корректировка исходных данных после третьего прогона дает следующие данные:  $r_3 = 10 - 2 = 8$ ,  $W_3 = 8 - 4 = 4$ ;  $S_3 = 1$ ;  $V_3 = 2$ , откуда количество ошибок определится следующим образом:

$$N_3 = \frac{W_3 \cdot S_3}{V_3} = \frac{4 \cdot 1}{2} = 2$$

После четвертого прогона программы получим следующие исходные данные:  $r_4 = 8 - 1 = 7$ ,  $W_4 = 4 - 2 = 2$ ;  $S_4 = 1$ ;  $V_4 = 2$ , тогда

$$N_4 = (W_4 \cdot S_4) / V_4 = (2 \cdot 1) / 2 = 1.$$

Поскольку после четвертого прогона все «посеянные» ошибки выявлены и устранены, то для оценки отлаженности программы можно воспользоваться упрощенной формулой

$$C = W / (W + r + 1) = 2 / (2 + 7 + 1) = 0,2.$$

Таким образом, в предположении, что до начала четвертого прогона в программе оставалось 7 ошибок, степень отлаженности программы составляет 20%.

Результат по количеству ошибок в программе до начала каждого прогона приведен ниже.

Номер прогона	1	2	3	4
$N$	9	4	2	1

Графически динамика количества ошибок по результатам тестирования программы показана на рис .2.

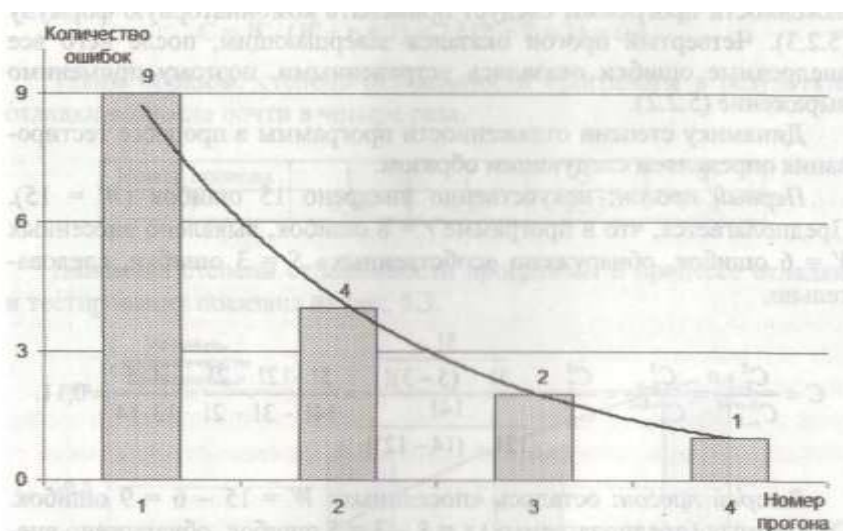


Рис. 5.2. Динамика количества ошибок при тестировании программы

Рис. 2 Динамика количества ошибок при тестировании программы

На приведенном графике дополнительно отображена линия тренда, характеризующая тенденцию снижения количества ошибок в программе по результатам тестовых прогонов.

### Задача 6

В программу было преднамеренно внесено (посеяно) 15 ошибок. В процессе четырех тестовых прогонов было выявлено следующее количество ошибок.

Номер прогона	1	2	3	4
$V$	6	4	3	2
$S$	3	2	1	1

Все обнаруженные ошибки исправлены. До начала тестирования предполагалось, что программа содержит не более 8 ошибок. Требуется оценить динамику степени отлаженности программы в процессе тестирования.

В процессе тестирования на первых трех этапах выявлены и устранены не все «посеянные» ошибки, поэтому для оценки степени отлаженности программы следует применять комбинаторную формулу. Четвертый прогон оказался завершающим, после него все внедренные ошибки оказались устраненными, поэтому применимо простое выражение.

Динамику степени отлаженности программы в процессе тестирования определяем следующим образом.

Первый прогон: искусственно внедрено 15 ошибок ( $W = 15$ ). Предполагается, что в программе  $r = 8$  ошибок, выявлено внесенных  $V = 6$  ошибок, обнаружено «собственных»  $S = 3$  ошибки, следовательно,

$$C = \frac{C_{V-1}^S}{C_{r+V}^{S+r+1}} = \frac{C_{6-1}^3}{C_{8+6}^{3+8+1}} = \frac{C_5^3}{C_{14}^{12}} = \frac{\frac{5!}{3! \cdot (5-3)!}}{\frac{14!}{12! \cdot (14-12)!}} = \frac{5! \cdot 12! \cdot 2!}{14! \cdot 3! \cdot 2!} = \frac{4 \cdot 5}{13 \cdot 14} = 0,11.$$

Второй прогон, осталось «посеянных»  $W = 15 - 6 = 9$  ошибок. Ожидаемых (предполагаемых)  $r = 8 - 3 = 5$  ошибок, обнаружено внесенных  $V = 4$ , обнаружено «собственных»  $S = 2$  ошибки. Тогда степень отлаженности программы после второго прогона будет:

$$C = \frac{C_{V-1}^S}{C_{r+V}^{S+r+1}} = \frac{C_{4-1}^2}{C_{5+4}^{2+5+1}} = \frac{C_3^2}{C_9^8} = \frac{\frac{3!}{2! \cdot (3-2)!}}{\frac{9!}{8! \cdot (9-8)!}} = \frac{3! \cdot 8! \cdot 1!}{9! \cdot 2! \cdot 1!} = \frac{3}{9} = 0,33.$$

Третий прогон: осталось посеянных  $W = 9 - 4 = 5$  ошибок. Осталось предполагаемых  $r = 5 - 2 = 3$  ошибки, обнаружено внесенных  $V = 3$  ошибки, обнаружено «собственных»  $S = 1$ , поэтому

$$C = \frac{C_{V-1}^S}{C_{r+V}^{S+r+1}} = \frac{C_{3-1}^1}{C_{3+3}^{1+3+1}} = \frac{C_2^1}{C_6^5} = \frac{\frac{2!}{1! \cdot (2-1)!}}{\frac{6!}{5! \cdot (6-5)!}} = \frac{2! \cdot 5! \cdot 1!}{6! \cdot 1! \cdot 1!} = \frac{2}{6} = 0,33.$$

После четвертого прогона осталось «посеянных»  $W = 5 - 3 = 2$  ошибки. Предполагается ошибок в программе  $r = 3 - 1 = 2$ , обнаружено внесенных  $V = 2$ , а «собственных»  $S = 1$ . Поскольку все внесенные ошибки в последнем прогоне обнаружены и устранены, возможно применение упрощенной формулы для оценки степени отлаженности программы:

$$C = W / (W + r + 1) = 2 / (2 + 2 + 1) = 0,4.$$

Таким образом, степень отлаженности программы в результате отладки возросла почти в четыре раза.

Номер прогона	1	2	3	4
$V$	0,11	0,33	0,33	0,4

Динамика степени отлаженности программы в процессе отладки и тестирования показана на рис. 3.

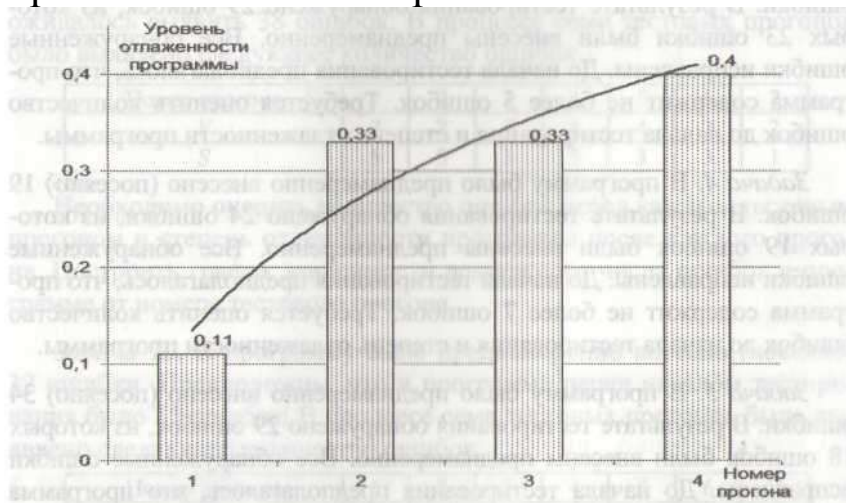


Рис. 3. Динамика уровня отлаженности при тестировании

На приведенном графике дополнительно отображена линия тренда, характеризующая тенденцию уровня отлаженности программы в процессе проведения тестовых прогонов.

### Задачи для самостоятельного решения

Задача 1. В программу было преднамеренно внесено (посеяно) 11 ошибок. В результате тестирования обнаружено 16 ошибок, из которых 11 ошибок были внесены преднамеренно. Все обнаруженные ошибки исправлены. До начала тестирования предполагалось, что программа содержит не более 7 ошибок. Требуется оценить

количество ошибок до начала тестирования и степень отлаженности программы.

Задача 2. В программу было преднамеренно внесено (посеяно) 25 ошибок. В результате тестирования обнаружено 29 ошибок, из которых 25 ошибок были внесены преднамеренно. Все обнаруженные ошибки исправлены. До начала тестирования предполагалось, что программа содержит не более 8 ошибок. Требуется оценить количество ошибок до начала тестирования и степень отлаженности программы.

Задача 3. В программу было преднамеренно внесено (посеяно) 23 ошибки. В результате тестирования обнаружено 29 ошибок, из которых 23 ошибки были внесены преднамеренно. Все обнаруженные ошибки исправлены. До начала тестирования предполагалось, что программа содержит не более 5 ошибок. Требуется оценить количество ошибок до начала тестирования и степень отлаженности программы.

Задача 4. В программу было преднамеренно внесено (посеяно) 19 ошибок. В результате тестирования обнаружено 24 ошибки, из которых 19 ошибок были внесены преднамеренно. Все обнаруженные ошибки исправлены. До начала тестирования предполагалось, что программа содержит не более 7 ошибок. Требуется оценить количество ошибок до начала тестирования и степень отлаженности программы.

Задача 5. В программу было преднамеренно внесено (посеяно) 34 ошибки. В результате тестирования обнаружено 29 ошибок, из которых 18 ошибок были внесены преднамеренно. Все обнаруженные ошибки исправлены. До начала тестирования предполагалось, что программа содержит не более 15 ошибок. Требуется оценить степень отлаженности программы на момент обнаружения указанных выше ошибок.

Задача 6. В программу было преднамеренно внесено (посеяно) 17 ошибок. В результате тестирования обнаружено 10 ошибок, из которых

ошибок были внесены преднамеренно. Все обнаруженные ошибки исправлены. До начала тестирования предполагалось, что программа содержит не более 7 ошибок. Требуется оценить степень отлаженности программы на момент обнаружения указанных выше ошибок.

Задача 7. В программу было преднамеренно внесено (посеяно) 39 ошибок. В результате тестирования обнаружено 23 ошибки, из которых



ошибок были внесены преднамеренно. Все обнаруженные ошибки исправлены. До начала тестирования предполагалось, что программа содержит не более 9 ошибок. Требуется оценить степень отлаженности программы на момент обнаружения указанных выше ошибок.

Задача 8. В программу было преднамеренно внесено (посеяно) 42 ошибки. В результате тестирования обнаружено 29 ошибок, из которых 25 ошибок были внесены преднамеренно. Нее обнаруженные ошибки исправлены. До начала тестирования предполагалось, что программа содержит не более 9 ошибок. Требуется оцепить степень отлаженности программы на момент обнаружения указанных выше ошибок.

Задача 9. В программу было преднамеренно внесено (посеяно) 32 ошибки. Предположим, что в программе перед началом тестирования ожидалось выявить 38 ошибок. И процессе семи тестовых прогонов было выявлено следующее количество ошибок.

Номер прогона	1	2	3	4	5	6	7
<i>V</i>	7	7	5	5	4	2	2
<i>S</i>	5	5	3	5	3	3	1

Необходимо оценить количество ошибок перед каждым тестовым прогоном и степень отлаженности программы после каждого прогона. Построить график зависимости возможного числа ошибок в программе от номера тестового прог она.

Задача 10. В программу было преднамеренно внесено (посеяно) 32 ошибки. Предположим, что в прог рамме перед началом тестирования было 35 ошибок. В процессе семи тестовых прогонов было выявлено следующее количество ошибок.

Номер прогона	1	2	3	4	5	6	7
<i>V</i>	8	6	7	4	3	2	2
<i>S</i>	5	3	3	4	3	1	1

Необходимо оценить количество ошибок перед каждым тестовым прогоном и степень отлаженности программы после каждого прогона. Построить график зависимости возможного числа ошибок в программе от номера тестового прогона. Проанализировать динамику отлаженности программы, для чего построить диаграмму и оценить тенденцию изменения этого показателя.

## ЭВРИСТИЧЕСКАЯ МОДЕЛЬ

### Теоретические сведения

Эвристическая модель оценки надежности программных средств позволяет оценить количество ошибок  $N$  до начала тестирования по результатам тестирования программы двумя независимыми группами. Для этого применяется следующее выражение:

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

где  $N_1$  - количество ошибок, обнаруженных первой группой тестирующих;  $N_2$  - количество ошибок, обнаруженных второй группой тестирующих;  $N_{12}$  – количество ошибок, которые обнаружила и первая, и вторая группа (общие обнаруженные ошибки).

Эвристическая модель хорошо работает при «перекрестном» тестировании программ несколькими группами тестировщиков, поскольку обеспечивает достаточно легкую обработку получаемых результатов.

### Задачи по применению эвристической модели

#### Задача 1

Программа тестируется двумя независимыми группами тестировщиков, которые силами групп выявили в программе 40 и 20 ошибок соответственно. При этом оказалось, что 10 ошибок - общие, их нашли обе группы. Требуется оценить общее количество ошибок в программе до начала тестирования и сделать вывод о необходимости продолжения тестирования или возможности его завершения.

Из условия задачи нам известны следующие исходные данные:

- количество ошибок, обнаруженных первой независимой группой тестировщиков ( $N_1 = 40$ );
- количество ошибок, обнаруженных второй независимой группой тестировщиков ( $N_2 = 20$ );
- количество ошибок, обнаруженных как первой, так и второй группой тестировщиков ( $N_{12} = 10$ ).

Согласно формуле определения общего числа ошибок  $N$  получим:

$$N = \frac{N_1 \cdot N_2}{N_{12}} = \frac{40 \cdot 20}{10} = 80.$$

Таким образом, можно считать, что в программе  $N = 80$  ошибок и из них не обнаружено  $80 - 40 - 20 + 10 = 30$  ошибок. Следовательно,

отладку программы и ее тестирование необходимо продолжать.

## Задача 2

Две независимые группы тестировщиков проводили тестирование программного средства. Первая группа обнаружила 15 ошибок, а вторая - 20. На основании результатов юстирования было определено, что до начала тестирования в программе содержалось 42 ошибки.

Необходимо определить, сколько ошибок было обнаружено как первой, так и второй группой.

Таким образом, нам известны следующие исходные данные:

- количество ошибок, обнаруженных первой независимой группой тестировщиков ( $N_1 = 15$ );
- количество ошибок, обнаруженных второй независимой группой тестировщиков ( $N_2 = 20$ );
- общее количество ошибок ( $N = 42$ ).

Из соотношения можно непосредственно получить искомый результат:

$$N_{1,2} = \frac{N_1 \cdot N_2}{N} = \frac{15 \cdot 20}{42} = 7.$$

Следовательно, первой и второй группами тестировщиков найдено 7 общих ошибок.

## Задача 3

В результате тестирования программы двумя независимыми группами: первой группой обнаружено 25 ошибок, а второй группой - 20 ошибок. Десять ошибок, обнаруженных первой группой, совпадает с ошибками, обнаруженными второй группой. Обнаруженные ошибки устранены. Требуется оценить количество ошибок, неустраненных и оставшихся в программе.

Из условия задачи нам известно:

- количество ошибок, обнаруженных первой группой тестирующих  $N_1 = 25$ ;
- количество ошибок, обнаруженных второй группой тестирующих  $N_2 = 20$ ;
- количество ошибок, которые обнаружила и первая, и вторая группа  $N_{12} = 10$ .

Подставляя исходные данные в формулу, получим:

$$N = (N_1 \cdot N_2) / N_{12} = (40 \cdot 20) / 10 = 80.$$

Определим количество обнаруженных и исправленных ошибок:

$$n = N_1 + N_2 - N_{12} = 25 + 20 + 10 = 35.$$



Количество оставшихся в программе ошибок  $n$  определим по следующему соотношению:

$$n1 = N - n = 80 - 35 = 45.$$

Таким образом, после тестирования и устранения выявленных ошибок в программе остается 45 ошибок.

#### Задача 4

По результатам тестирования программы двумя независимыми группами известно следующее:

- первой группой обнаружено 16 ошибок;
- количество ошибок до начала тестирования составляло 35;
- общее количество обнаруженных ошибок двумя группами 7.

Требуется установить вклад второй группы в процесс тестирования, определив количество ошибок, обнаруженных второй группой.

Из условия задачи нам известно:

- количество ошибок, обнаруженных первой группой тестирующих  $N1 = 16$ ;
- количество ошибок до начала тестирования  $N = 35$ ;
- количество общих ошибок, которые обнаружила и первая, и вторая группа  $N12 = 7$ .

Количество ошибок, обнаруженных второй группой, можно определить из выражения:

$$N2 = \frac{N * N12}{N1} = \frac{35 * 7}{16} = 15$$

Таким образом, вторая группа выявила 15 ошибок.

#### Задачи для самостоятельного решения

Задача 1. Две независимые группы проводили тестирование программного средства. Первая группа обнаружила 23 ошибки, а вторая - 31. На основании результатов тестирования было определено, что до начала тестирования в программе содержалось 53 ошибки. Определить, какое количество общих ошибок обнаружено обеими группами.

Задача 2. Две независимые группы проводили тестирование программного средства. Первая группа обнаружила 24 ошибки, а вторая - 19. На основании результатов тестирования было определено, что до начала тестирования в программе содержалось 32 ошибки. Определить, какое количество общих ошибок обнаружено обеими группами.

Задача 3. Две независимые группы проводили тестирование программного средства. Первая группа обнаружила 27 ошибок, а вторая

— 33. На основании результатов тестирования было определено, что до начала тестирования в программе содержалось 54 ошибки. Определить, какое количество общих ошибок обнаружено обеими группами.

Задача 4. Две независимые группы проводили тестирование программного средства. Первая группа обнаружила 38 ошибок, а вторая — 29. На основании результатов тестирования было определено, что до начала тестирования в программе содержалось 54 ошибки. Определить, какое количество общих ошибок обнаружено обеими группами.

Задача 5. Две независимые группы проводили тестирование программного средства. Первая группа обнаружила 35 ошибок, а вторая - 21. На основании результатов тестирования было определено, что до начала тестирования в программе содержалось 49 ошибок. Определить, какое количество общих ошибок обнаружено обеими группами.

Задача 6. В результате тестирования программы двумя независимыми группами: первой группой обнаружено 17 ошибок, а второй группой - 26 ошибок. 14 ошибок, обнаруженных первой группой, совпадают с ошибками, обнаруженными второй группой. Обнаруженные ошибки устранены. Требуется оценить количество неустраненных ошибок, которые остались в программе после тестирования.

Задача 7. В результате тестирования программы двумя независимыми группами: первой группой обнаружено 39 ошибок, а второй группой - 28 ошибок. 16 ошибок, обнаруженных первой группой, совпадает с ошибками, обнаруженными второй группой. Обнаруженные ошибки устранены. Требуется оценить количество неустраненных ошибок, которые остались в программе после тестирования.

Задача 8. В результате тестирования программы двумя независимыми группами: первой группой обнаружено 37 ошибок, а второй группой - 22 ошибки. 13 ошибок, обнаруженных первой группой, совпадает с ошибками, обнаруженными второй группой. Обнаруженные ошибки устранены. Требуется оценить количество неустраненных ошибок, которые остались в программе после тестирования.

Задача 9. В результате тестирования программы двумя независимыми группами: первой группой обнаружена 41 ошибка, а второй группой - 37 ошибок. 24 ошибки, обнаруженные первой группой, совпадает с ошибками, обнаруженными второй группой. Обнаруженные ошибки устранены. Требуется оценить количество неустраненных ошибок, которые остались в программе после тестирования.

Задача 10. В результате тестирования программы двумя независимыми группами: первой группой обнаружено 16 ошибок, а второй

группой - 19 ошибок. 10 ошибок, обнаруженных первой группой, совпадает с ошибками, обнаруженными второй группой. Обнаруженные ошибки устранены. Требуется оценить количество неустраненных ошибок, которые остались в программе после тестирования.

Задача 11. По результатам тестирования программы двумя независимыми группами известно, что первой группой обнаружено 19 ошибок, количество ошибок до начала тестирования 43, общее количество обнаруженных ошибок двумя группами 14. Требуется определить количество ошибок, обнаруженных второй группой.

Задача 12. По результатам тестирования программы двумя независимыми группами известно, что первой группой обнаружено 28 ошибок, количество ошибок до начала тестирования 44, общее количество обнаруженных ошибок двумя группами 22. Требуется определить количество ошибок, обнаруженных второй группой.

Задача 13. По результатам тестирования программы двумя независимыми группами известно, что первой группой обнаружено 19 ошибок, количество ошибок до начала тестирования 25, общее количество обнаруженных ошибок двумя группами 17. Требуется определить количество ошибок, обнаруженных второй группой.

Задача 14. По результатам тестирования программы двумя независимыми группами известно, что первой группой обнаружено 23 ошибки, количество ошибок до начала тестирования 22, общее количество обнаруженных ошибок двумя группами 9. Требуется определить количество ошибок, обнаруженных второй группой.

Задача 15. По результатам тестирования программы двумя независимыми группами известно, что первой группой обнаружено 22 ошибки, количество ошибок до начала тестирования 26, общее количество обнаруженных ошибок двумя группами 10. Требуется определить количество ошибок, обнаруженных второй группой.

## МОДЕЛЬ НЕЛЬСОНА

### Теоретические сведения

Классической измерительной моделью надежности является известная модель Нельсона, предложенная в 1978 г. Модель основана на выделении областей исходных данных  $E_i$  покрывающих все множество вариантов их использования в программе  $E$ :

$$E = \{E_i\}$$

Множество результатов выполнения программы  $F(E)$  определяется композицией результатов по всем вариантам данных  $F(E_i)$ :

$$F(E) = \bigcup F(E_i).$$

Выражение позволяет определить программу как описание некоторой вычисляемой функции  $F$  на множестве всех значений наборов входных данных  $E$ .

Если обозначить  $F_\phi(E_i)$  множество фактических результатов выполнения программы на наборе  $E_i$  то множество правильных значений будет определяться условием

$$\forall i |F(E_i) - F_\phi(E_i)| \leq \varepsilon_i,$$

где  $\varepsilon_i$  - допустимое расхождение между правильным (эталонным) и фактическим результатом выполнения программы на наборе  $E_i$ .

Рабочим отказом считается ситуация, при которой выполняется условие

$$\forall i |F(E_i) - F_\phi(E_i)| > \varepsilon_i.$$

или программа «зацикливается» (выполняется бесконечное продолжение работы программы, при котором программа не может закончиться). При этом предполагается, что выбор любого набора исходных данных  $E_i \subset E$  равновероятен.

Мощность всего множества наборов исходных данных  $E$  обозначим символом  $N$ . Множество, состоящее из всех наборов  $E_i$  для которых получены неудовлетворительные результаты, обозначим  $E_0$ . Мощность множества  $E_0$  обозначим  $N_0$ . Совокупность действий, включающих ввод  $E_i$  и выполнение программы, которое заканчивается получением результата  $F_\phi(E_i)$  или рабочим отказом, называется прогоном программы. Следует заметить, что входные данные, образующие набор  $E_i$  не обязательно должны подаваться на вход одновременно.

При этих допущениях справедливо утверждать следующее: вероятность  $P$  того, что прогон программы приведет к рабочему отказу, равна вероятности того, что набор данных  $E$  который использовался в прогоне, принадлежит множеству  $E_0$ . Тогда вероятность появления ошибки при прогоне программы на входном наборе, случайно выбранном из числа равновероятных, определяется следующим выражением:

$$P = N_0 / N.$$

Вероятность  $R$  того, что прогон программы на наборе входных данных  $E_i$ , случайно выбранном из  $E$  среди равновероятных наборов, приведет к приемлемому результату, определяется выражением

$$R = 1 - P = 1 - \frac{N_0}{N}.$$

Если выбор набора данных из множества  $E$  не равновероятен, а возможны какие-либо приоритеты выбора набора данных, то для оценки надежности программы следует использовать другое выражение:

$$R = 1 - \sum_{i=1}^N p_i \cdot y_i,$$

где  $p_i$  — вероятность (частота) использования  $i$ -го набора исходных данных,  $y_i$  — динамическая переменная, которая принимает нулевое значение, если прогон заканчивается приемлемым результатом, и значение 1, если прогон заканчивается рабочим отказом.

Вероятность  $R(n)$  успешного выполнения  $n$  прогонов программы при независимом для каждого прогона выборе исходных данных определяется выражением

$$R(n) = R^n = (1 - P)^n.$$

Эту вероятность  $R(n)$  можно представить в следующем виде:

$$R(n) = e^{\sum_{j=1}^n \ln(1 - p_j)},$$

где  $p_j$  — вероятность отказа для  $j$ -го прогона.

Дальнейшие преобразования показывают значительное сходство с технологией определения безотказности, принятой в теории надежности технических устройств. Модель Нельсона в наибольшей степени отражает традиционный подход, принятый для определения надежности технических устройств, для измерения надежности программ.

## Задачи по применению модели Нельсона

### Задача 1

Для испытания программы использовалось 20 наборов исходных данных, которые равновероятно выбирались для прогона 20 тестов. При этом 10 тестов обнаружили дефекты программного обеспечения. Требуется провести расчет надежности программного обеспечения по результатам испытаний.

В соответствии с моделью Нельсона надежность программного обеспечения по результатам испытаний определяется вероятностью  $R$  того, что прогон программы на наборе входных данных  $E_h$  случайно выбранном из  $E_i$  среди равновероятных, приведет к приемлемому результату.

Из условия задачи общее количество тестов  $N = 20$ , количество тестов с обнаружением дефектов программы  $N_0 = 10$ . Подставим исходные данные в расчетную формулу

$$R = 1 - \frac{N_0}{N} = 1 - \frac{10}{20} = 1 - 0.5 = 0.5$$

Таким образом, вероятность  $R$  события, при котором прогон программы на заданном наборе исходных данных не приведет к рабочему отказу, равна 0,5.

## Задача 2

Для испытания программы использовалось 30 наборов исходных данных, которые выбирались в соответствии с функцией распределения частот, значения которой представлены ниже.

№ теста	Частота выбора теста	Исход прогона теста	№ теста	Частота выбора теста	Исход прогона теста
1	0,04	1	16	0,01	0
2	0,01	0	17	0,02	1
3	0,03	0	18	0,01	0
4	0,05	0	19	0,03	1
5	0,02	1	20	0,19	0
6	0,03	0	21	0,03	1
7	0,05	1	22	0,02	0
8	0,01	0	23	0,04	1
9	0,04	0	24	0,01	1
10	0,01	0	25	0,02	1
11	0,02	1	26	0,01	1
12	0,07	0	27	0,03	1
13	0,01	0	28	0,06	1
14	0,02	1	29	0,02	1
15	0,05	1	30	0,04	1

В 17 тестах были обнаружены ошибки. Все исходы прогонов, закончившиеся отказом, в таблице обозначены единицами. Определить надежность программы по результатам испытаний.

Если набор данных для тестирования программы не равновероятен, то для оценки надежности программы используем выражение

$$R = 1 - (0,04 \cdot 1 + 0,02 \cdot 1 + 0,05 \cdot 1 + 0,02 \cdot 1 + 0,02 \cdot 1 + 0,05 \cdot 1 + 0,02 \cdot 1 + 0,03 \cdot 1 + 0,03 \cdot 1 + 0,04 \cdot 1 + 0,01 \cdot 1 + 0,02 \cdot 1 + 0,01 \cdot 1 + 0,03 \cdot 1 + 0,06 \cdot 1 + 0,02 \cdot 1 + 0,04 \cdot 1) = 1 - 0,51 = 0,49.$$

Таким образом, вероятность события  $R$ , что прогон программы, на заданном наборе исходных данных не приведет к рабочему отказу, равна 0,49.



### Задача 3

При испытании программы провели 20 прогонов тестов. По результатам испытаний было установлено, что вероятность отказа программы при равновероятном выборе набора данных составляет. Определить количество дефектов, выявленных в процессе испытаний программы.

В соответствии с моделью Нельсона надежность программного обеспечения по результатам испытаний при использовании равновероятно выбираемого набора данных определяется вероятностью  $R$

Тогда количество дефектов  $N_0$ , которые обнаружены при тестировании, определяем следующим образом

$$N_0 = (1 - R) * N = (1 - 0.3) * 20 = 0.7 * 20 = 14$$

Таким образом, при испытании программы выявлено 14 дефектов.

### Задача 4

При испытании программы было выявлено 8 дефектов. При этом вероятность отказа программы составила 0,6. Определить количество наборов исходных данных, используемых для тестирования программы, если наборы данных выбираются равновероятно.

В соответствии с моделью Нельсона надежность программного обеспечения по результатам испытаний при использовании равновероятно выбираемого набора данных определяется вероятностью  $R$ .

Тогда количество наборов исходных данных  $N$  для испытания можно определить следующим образом:

$$N = \frac{N_0}{1 - R} = \frac{8}{1 - 0.6} = 20$$

Таким образом, программа тестировалась на основе 20 наборов исходных данных.

### Задачи для самостоятельного решения

Задача 1. Для испытания программы использовалось 10 наборов исходных данных, которые равновероятно выбирались для прогона 10 тестов. При этом в 8 тестах обнаружены дефекты программного обеспечения. Требуется провести расчет надежности программного обеспечения по результатам испытаний.

Задача 2. Для испытания программы использовалось 15 наборов исходных данных, которые равновероятно выбирались для прогона 15 тестов. При этом в 5 тестах обнаружены дефекты программного

обеспечения. Требуется провести расчет надежности программного обеспечения по результатам испытаний.

Задача 3. Для испытания программы использовалось 32 набора исходных данных, которые равновероятно выбирались для прогона 32 тестов. При этом в 27 тестах обнаружены дефекты программного обеспечения. Требуется провести расчет надежности программного обеспечения по результатам испытаний.

Задача 4. Для испытания программы использовалось 17 наборов исходных данных, которые равновероятно выбирались для прогона тестов. При этом в 6 тестах обнаружены дефекты программного обеспечения. Требуется провести расчет надежности программного обеспечения по результатам испытаний.

Задача 5. Для испытания программы использовалось 24 набора исходных данных, которые равновероятно выбирались для протона 24 тестов. При этом в 12 тестах обнаружены дефекты программного обеспечения. Требуется провести расчет надежности программного обеспечения по результатам испытаний.

Задача 6. Для испытания программы использовалось 16 наборов исходных данных, которые выбирались в соответствии с функцией распределения частот, представленной ниже.

№ теста	Частота выбора теста	Исход прогона теста	№ теста	Частота выбора теста	Исход прогона теста
1	0,05	0	9	0,07	0
2	0,03	1	10	0,04	0
3	0,07	1	11	0,08	1
4	0,06	0	12	0,09	0
5	0,08	1	13	0,04	1
6	0,1	1	14	0,07	0
7	0,06	1	15	0,06	0
8	0,06	0	16	0,04	0

В 7 тестах были обнаружены ошибки. Итоги прогонов, закончившиеся отказом, обозначены единицами. Определить надежность программы по результатам испытаний.

Задача 7. Для испытания программы использовалось 18 наборов исходных данных, которые выбирались в соответствии с функцией распределения частот, представленной ниже.

№ теста	Частота выбора теста	Исход прогона теста	№ теста	Частота выбора теста	Исход прогона теста
1	0,06	0	10	0,05	0
2	0,03	0	11	0,06	0
3	0,07	1	12	0,05	0
4	0,06	0	13	0,04	0
5	0,07	1	14	0,07	0
6	0,05	1	15	0,06	1
7	0,06	1	16	0,03	0
8	0,07	1	17	0,07	0
9	0,04	1	18	0,06	1



В 8 тестах были обнаружены ошибки. Итоходы прогонов, закончившиеся отказом, обозначены единицами. Определить надежность программы по результатам испытаний.

Задача 8. Для испытания программы использовалось 18 наборов исходных данных, которые выбирались в соответствии с функцией распределения частот, представленной ниже.

№ теста	Частота выбора теста	Итоход прогона теста	№ теста	Частота выбора теста	Итоход прогона теста
1	0,09	1	10	0,08	0
2	0,02	0	11	0,09	0
3	0,06	1	12	0,05	1
4	0,05	0	13	0,04	1
5	0,07	1	14	0,07	0
6	0,05	1	15	0,06	1
7	0,03	1	16	0,05	0
8	0,02	1	17	0,1	0
9	0,04	1	18	0,03	1

В 11 тестах были обнаружены ошибки. Итоходы прогонов, закончившиеся отказом, обозначены единицами. Определить надежность программы по результатам испытаний.

Задача 9. Для испытания программы использовалось 22 набора исходных данных, которые выбирались в соответствии с функцией распределения частот, представленной ниже.

№ теста	Частота выбора теста	Итоход прогона теста	№ теста	Частота выбора теста	Итоход прогона теста
1	0,05	0	12	0,05	0
2	0,03	0	13	0,07	1
3	0,03	1	14	0,07	0
4	0,05	0	15	0,06	0
5	0,06	1	16	0,06	0
6	0,05	1	17	0,09	1
7	0,03	1	18	0,03	1
8	0,02	0	19	0,01	0
9	0,04	0	20	0,03	1
10	0,06	1	21	0,02	0
11	0,07	0	22	0,02	0

В 9 тестах были обнаружены ошибки. Итоходы прогонов, закончившиеся отказом, обозначены единицами. Определить надежность программы по результатам испытаний.

Задача 10. Для испытания программы использовалось 22 набора исходных данных, которые выбирались в соответствии с функцией распределения частот, представленной ниже.

№ теста	Частота выбора теста	Исход прогона теста	№ теста	Частота выбора теста	Исход прогона теста
1	0,09	1	12	0,05	1
2	0,06	0	13	0,04	1
3	0,07	1	14	0,03	0
4	0,05	0	15	0,06	1
5	0,06	0	16	0,05	0
6	0,05	1	17	0,05	1
7	0,03	0	18	0,03	1
8	0,05	1	19	0,01	0
9	0,04	0	20	0,03	1
10	0,05	1	21	0,02	1
11	0,07	0	22	0,01	0

В 12 тестах были обнаружены ошибки. Исходы прогонов, закончившиеся отказом, обозначены единицами. Определить надежность программы по результатам испытаний.

Задача 11. Для испытания программы использовалось 24 набора исходных данных, которые выбирались в соответствии с функцией распределения частот, представленной ниже.

№ теста	Частота выбора теста	Исход прогона теста	№ теста	Частота выбора теста	Исход прогона теста
1	0,07	1	13	0,04	1
2	0,04	0	14	0,03	1
3	0,06	1	15	0,05	0
4	0,05	0	16	0,04	1
5	0,05	0	17	0,05	1
6	0,05	1	18	0,03	0
7	0,03	0	19	0,01	1
8	0,05	1	20	0,03	1
9	0,04	1	21	0,03	0
10	0,05	0	22	0,02	1
11	0,05	1	23	0,04	1
12	0,05	0	24	0,04	1

В 15 тестах были обнаружены ошибки. Исходы прогонов, закончившиеся отказом, обозначены единицами. Определить надежность программы по результатам испытаний.

Задача 12. При испытании программы провели 24 прогона тестов. По результатам испытаний было установлено, что вероятность отказа программы при заданном наборе данных составляет 0,7. Определить количество дефектов, выявленных в процессе испытания программы.

Задача 13. При испытании программы провели 15 прогонов тестов. По результатам испытаний было установлено, что вероятность отказа программы при заданном наборе данных составляет 0,25. Определить количество дефектов, выявленных в процессе испытания программы.

Задача 14. При испытании программы было выявлено 16 дефектов, при этом вероятность отказа программы составила 0,2. Определить количество наборов исходных данных, используемых для тестирования программы.

Задача 15. При испытании программы было выявлено 16 дефектов, при этом вероятность отказа программы составила 0,4. Определить количество наборов исходных данных, используемых для тестирования программы.

## МОДЕЛЬ КОРКОРЭНА

### Теоретические сведения

Применение модели предполагает знание следующих ее показателей:

- модель содержит изменяющуюся вероятность отказов для различных источников ошибок и соответственно разную вероятность их исправления;
- в модели используются такие параметры, как результат только  $N$  испытаний, в которых наблюдается  $N_i$  ошибок  $i$ -го типа;
- выявление в ходе  $N$  испытаний ошибки  $i$ -го типа появляется с вероятностью  $a_i$ .

Показатель уровня надежности  $R$  вычисляют по следующей формуле:

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

где  $N_0$  - число безотказных (или безуспешных) испытаний, выполненных в серии из  $N$  испытаний,  $k$  - известное число типов ошибок,  $Y_i$  - вероятность появления ошибок, при  $N_i > 0$ ,  $Y_i = a_i$ , при  $N_i = 0$ ,  $Y_i = 0$ .

### Задачи по применению модели Коркорэна

1. Оттестировать и оценить надёжность по модели Коркорэна. Было проведено 75 испытаний программы. 15 из 75 испытаний прошли безуспешно, а в остальных случаях получились следующие данные (табл. 1):

#### Результаты тестирования

Таблица 1

Тип ошибки	Вероятность появления $Y_i$	Вероятность появления ошибки. При исп. $N_i$
1. Ошибки вычисления	0.09	5

2. Логические ошибки	0,26	25
3. Ошибки ввода/вывода	0,16	3
4. Ошибки манипулирования данными	0,18	- -
5. Ошибки сопряжения	0,17	11
6. Ошибки определения данных	0,08	3
7. Ошибки в БД	0,06	4

Вероятность безотказного выполнения программы (R) вычисляют по формуле:

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

Тогда

$$R = \frac{15}{75} + \frac{0,09 \cdot (5 - 1)}{75} + \frac{0,26 \cdot (25 - 1)}{75} + \frac{0,16 \cdot (3 - 1)}{75} + \frac{0 \cdot (0 - 1)}{75} + \frac{0,17 \cdot (11 - 1)}{75} + \frac{0,08 \cdot (3 - 1)}{75} + \frac{0,06 \cdot (4 - 1)}{75} = 0,32$$

### Задачи для самостоятельного решения

1. Оттестировать и оценить надёжность по модели Коркорэна. Было проведено 100 испытаний программы. 20 из 100 испытаний прошли безуспешно, а в остальных случаях получились следующие данные (табл. 1):

#### Результаты тестирования

Таблица 1

Тип ошибки	Вероятность появления $Y_i$	Вероятность появления ошибки при испытании. $N_i$
1. Ошибки вычисления	0,09	5
2. Логические ошибки	0,26	25
3. Ошибки ввода/вывода	0,16	3
4. Ошибки манипулирования данными	0,18	- -
5. Ошибки сопряжения	0,17	11
6. Ошибки определения данных	0,08	3

Тип ошибки	Вероятность появления $Y_i$	Вероятность появления ошибки при испытании. $N_i$
7. Ошибки в БД	0,06	4

2. Оттестировать и оценить надёжность по модели Коркорэна. Было проведено 100 испытаний программы. 20 из 100 испытаний прошли безуспешно, а в остальных случаях получились следующие данные (табл. 2):

#### Результаты тестирования

Таблица 2

Тип ошибки	Вероятность появления $Y_i$	Вероятность появления ошибки при испытании. $N_i$
1. Ошибки вычисления	0,26	5
2. Логические ошибки	0,9	- -
3. Ошибки ввода/вывода	0,8	4
4. Ошибки манипулирования данными	0,2	25
5. Ошибки сопряжения	0,17	11
6. Ошибки определения данных	0,08	3
7. Ошибки в БД	0,16	3

3. Оттестировать и оценить надёжность по модели Коркорэна. Было проведено 100 испытаний программы. 20 из 100 испытаний прошли безуспешно, а в остальных случаях получились следующие данные (табл. 3):

#### Результаты тестирования

Таблица 3

Тип ошибки	Вероятность появления $Y_i$	Вероятность появления ошибки при испытании. $N_i$
1. Ошибки вычисления	0,09	8
2. Логические ошибки	0,26	--
3. Ошибки ввода/вывода	0,17	4
4. Ошибки манипулирования данными	0,2	25
5. Ошибки сопряжения	0,8	25

Тип ошибки	Вероятность появления $Y_i$	Вероятность появления ошибки при испытании. $N_i$
6. Ошибки определения данных	0,08	3
7. Ошибки в БД	0,16	5

4. Оттестировать и оценить надёжность по модели Коркорэна. Было проведено 100 испытаний программы. 20 из 100 испытаний прошли безуспешно, а в остальных случаях получились следующие данные (табл. 3):

#### Результаты тестирования

Таблица 3

Тип ошибки	Вероятность появления $Y_i$	Вероятность появления ошибки. При исп. $N_i$
1. Ошибки вычисления	0,2	4
2. Логические ошибки	0,26	3
3. Ошибки ввода/вывода	0,17	11
4. Ошибки манипулирования данными	0,9	--
5. Ошибки сопряжения	0,08	3
6. Ошибки определения данных	0,8	5
7. Ошибки в БД	0,16	25

## МОДЕЛЬ ШУМАНА

### Теоретические сведения

Модель Шумана относится к динамическим моделям дискретного времени, данные для которой собираются в процессе тестирования программного обеспечения в течение фиксированных или случайных интервалов времени. Модель 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. В конце этапа рассчитываются количественные показатели надёжности, исправляются найденные ошибки,

корректируются тестовые наборы и проводится следующий этап тестирования. В модели Шумана предполагается, что число ошибок в программе постоянно и в процессе корректировки новые ошибки не вносятся. Скорость обнаружения ошибок пропорциональна числу оставшихся ошибок.

Предполагается, что до начала тестирования имеется  $Et$  ошибок. В течение времени тестирования  $\tau$  обнаруживается  $\varepsilon_c$  ошибок в расчете на одну команду в машинном языке.

Таким образом, удельное число ошибок на одну машинную команду, оставшихся в системе после  $\tau$  времени тестирования, равно:

$$\varepsilon_c(\tau) = \frac{Et}{It} \cdot \varepsilon_c(\tau),$$

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

Предполагается, что значение функции частоты отказов  $Z(t)$  пропорционально числу ошибок, оставшихся в программе после израсходованного на тестирование времени  $\tau$ .

$$Z(t) = C \cdot \varepsilon_r(\tau),$$

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

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

$$R(t, \tau) = \exp \left\{ -C \cdot \left[ \frac{Et}{It} - \varepsilon_c(\tau) \right] \cdot t \right\},$$

$$t_{cp} = \frac{1}{C \cdot \left[ \frac{Et}{It} - \varepsilon_c(\tau) \right]}.$$

Нам необходимо найти начальное значение ошибок  $Et$  и коэффициент пропорциональности -  $C$ . В процессе тестирования собирается информация о времени и количестве ошибок на каждом прогоне, т.е. общее время тестирования  $\tau$  складывается из времени каждого прогона

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

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

$$\lambda = \frac{\sum_{i=1}^i A_i}{\tau},$$

где  $A_i$  - количество ошибок на  $i$  - ом прогоне.

Имея данные для двух различных моментов тестирования  $\tau_a$  и  $\tau_b$ , которые выбираются с учетом того, чтобы  $\varepsilon(\tau_a) < \varepsilon(\tau_b)$ , можно получить что:

$$Et = \frac{It \left[ \frac{\lambda t_b}{\lambda t_a} \cdot \varepsilon_c(\tau_b) - \varepsilon_c(\tau_a) \right]}{\frac{\lambda t_b}{\lambda t_a} - 1}$$

$$C = \frac{\lambda \tau_a}{\frac{Et}{It} - \varepsilon_c(\tau_a)}$$

Получив неизвестные  $Et$  и  $C$ , определяется надежность  $R(t, \tau)$  программы.

### Задачи по применению модели Шумана

Например, в программе имеется  $It = 4381$  оператор. В процессе последовательных тестовых прогонов были получены следующие данные (табл. 1):

#### Исходные данные

Таблица 1.

N прогона	1	2 A	3	4	5	6	7	8B	9	10
Кол-во ошибок	1	2	1	1	1	1	1	2	1	1
Время (м)	5	8	2	1	5	1	1	2	5	5

Выберем две точки, исходя из требования, чтобы число ошибок, найденных на интервале  $A \div B$  было больше, чем на интервале  $0 \div A$ . За точку A возьмем 2 прогон, а за точку B – 8 прогон. Тогда ошибки, найденные на этапах тестирования на интервалах  $0 \div A$  и  $A \div B$ , будут равны соответственно:



$$\varepsilon_c(\tau_A) = \frac{41}{4381} = 0.0007$$

$$\varepsilon_c(\tau_B) = \frac{7}{4381} = 0.0015$$

Время тестирования на интервалах равно:

$$\tau_A = 13$$

$$\tau_B = 12$$

Рассчитаем интенсивности появления ошибок на двух интервалах:

$$\lambda_A = \frac{3}{13} = 0.23$$

$$\lambda_B = \frac{7}{12} = 0.58$$

Тогда число имеющихся до начала тестирования ошибок равно:

$$Et = \frac{4381 \left( \frac{0.58}{0.23} \cdot 0.0007 - 0.0015 \right)}{\frac{0.58}{0.23} - 1} = 0.763 \approx 1 \text{ ошибка}$$

$$C = \frac{0.23}{\frac{0.76}{4381} - 0.0007} = -437$$

Удельное число ошибок за все время испытаний:

$$\varepsilon_c(\tau_{35}) = \frac{12}{4381} = 0.0027.$$

Из технических требований известно, что минимальное время безотказной работы составляет 60 мин, тогда:

$$R(60,35) = \exp \left\{ 437 \cdot \left[ \frac{0.763}{4381} - 0.0027 \right] \cdot 60 \right\} = 0.9$$

Таким образом, надежность безотказной работы достаточно велика и вероятность сбоев и возникновения ошибок небольшая.

## Задачи для самостоятельного решения по модели Шумана

### 1. Оценить надёжность по модели Шумана.

Дано:

Общее число операторов ( $It$ ): 10000

Оценка осуществляется после 10 прогонов.

Исходные данные:

Номер прогона	1	2	3	4	5	6	7	8	9	10
Кол-во ошибок	2	0	5	3	4	1	3	2	0	1
Время	0.5	0.4	0.5	0.75	0.2	0.5	0.3	0.3	0.1	0.4

### 2. Оценить надёжность по модели Шумана.

Дано:

Общее число операторов ( $It$ ): 9000

Оценка осуществляется после 10 прогонов.

Исходные данные:

Номер прогона	1	2	3	4	5	6	7	8	9	10
Кол-во ошибок	2	5	0	3	4	1	2	1	1	0
Время	0.4	0.5	0.5	0.2	0.75	0.3	0.5	0.3	0.1	0.4

### 3. Оценить надёжность по модели Шумана.

Дано:

Общее число операторов ( $It$ ): 10000

Оценка осуществляется после 9 прогонов.

Исходные данные:

Номер прогона	1	2	3	4	5	6	7	8	9
Кол-во ошибок	1	5	4	1	0	1	2	3	2
Время	0.5	0.1	0.3	0.2	0.75	0.3	0.4	0.5	0.5

### 4. Оценить надёжность по модели Шумана.

Дано:

Общее число операторов ( $It$ ): 10000

Оценка осуществляется после 8 прогонов.

Исходные данные:

Номер прогона	1	2	3	4	5	6	7	8
Кол-во ошибок	0	5	1	3	1	2	1	2
Время	0.5	0.1	0.75	0.5	0.3	0.4	0.2	0.5

## МОДЕЛЬ LA PADULA

### Теоретические сведения

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

Надежность ПС в течение  $i$ -го этапа:

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

где  $A$  — параметр роста,  $R(\infty) = \lim_{i \rightarrow \infty} R(i)$  — предельная надежность ПС.

Эти неизвестные величины автор предлагает вычислить, решив следующие уравнения:

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

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

где  $S_i$  — число тестов,  $m_i$  — число отказов во время  $i$ -го этапа,  $m$  — число этапов,  $i = 1, 2, \dots, m$ .

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

$$R(t) = R(\infty) - \frac{A}{i}.$$

Преимущество модели заключается в том, что она является прогнозной и, основываясь на данных, полученных в ходе тестирования, дает возможность предсказать вероятность безотказной работы программы на последующих этапах ее выполнения.

### Задачи по применению модели La Padula

Произведено тестирование ПС в три этапа по 10 тестов на каждом. На первом этапе было обнаружено 10 ошибок, на втором – 5 и на третьем – 0.

Пример реализации количественной оценки надежности в MathCad.

Число отказов на  $i$ -ом этапе:

$$m_i := \begin{vmatrix} 10 \\ 5 \\ 0 \end{vmatrix}.$$

Количество тестов на этапе:

$$S := \begin{vmatrix} 10 \\ 10 \\ 10 \end{vmatrix}.$$

$m := 3$  количество этапов;

$A := 1$  параметр роста;

$Rf := 1$  предельное значение надежности.

Основные разрешающие уравнения:

Given

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

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

$$\begin{vmatrix} Rf \\ A \end{vmatrix} := Find(Rf, A)$$

$$Rf = 1.946$$

$$A = 1.385$$

Надежность системы в течении  $i$ -го этапа определяется формулой:

$$R(i) := Rf - \frac{A}{i}$$

$$i := 1..100$$

График изменения надежности представлен на рис. 1.

Видно, что в процессе проведенных тестов прогнозируемое значение надежности приближается к 1. Тестирование может быть прекращено при достижении требуемого показателя надежности.

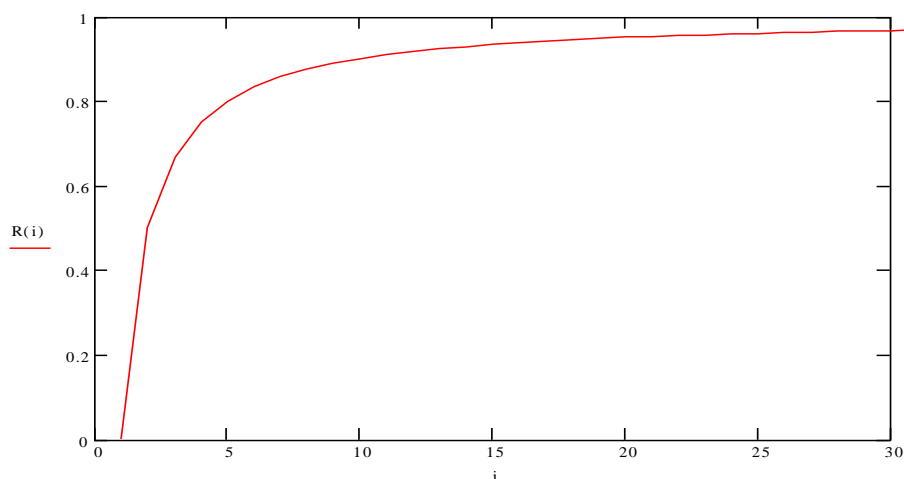


Рис.1. График изменения надежности

### Задачи для самостоятельного решения по модели La Padula

1. Произведено тестирование ПС в три этапа по 8 тестов на каждом. На первом этапе было обнаружено 12 ошибок, на втором – 8 и на третьем – 3.

2. Произведено тестирование ПС в пять этапов по 10 тестов на каждом. На первом этапе было обнаружено 12 ошибок, на втором – 8, на третьем – 3, на четвертом – 4 и на пятом - 1.

3. Произведено тестирование ПС в семь этапов по 8 тестов на каждом. На первом этапе было обнаружено 20 ошибок, на втором – 16, на третьем – 12, на четвертом – 10, на пятом – 7, на шестом – 3 и на седьмом - 0.

4. Произведено тестирование ПС в четыре этапа по 16 тестов на каждом. На первом этапе было обнаружено 32 ошибки, на втором – 16, на третьем – 8 и на четвертом – 4.

## СПИСОК РЕКОМЕНДУЕМО ЛИТЕРАТУРЫ

1. Черников Б.В. Управление качеством программного обеспечения:/ Б.В. Черников. – М.: ИД «ФОРУМ»: ИНФРА-М, 2012. – 240 с.:
2. Черников Б.В. Оценка качества программного обеспечения: Практикум: учебное пособие / Б.В. Черненко. Б.Е. Поклонов/ Под ред. Б.В. Чернякова. – М.: ИД «Форум»: ИНФРА-М, 2012. – 400 с.

Метрология и качество программного обеспечения. Оценка надежности программных средств: методические указания к выполнению лабораторной работы №12, 13, 14, 15 для студентов очной формы обучения по направлению подготовки 09.03.01 «Информатика и вычислительная техника»; 09.03.04 «Программная инженерия»; 02.03.03 «Математическое обеспечение и администрирование информационных систем»

АЗАРЧЕНКОВ АНДРЕЙ АНАТОЛЬЕВИЧ

Научный редактор Д.А. Коростелев

Компьютерный набор А.А. Азарченков

Иллюстрации А.А. Азарченков

---

Подписано в печать 07.07.2017г. Формат 60х84 1/16 Бумага офсетная. Офсетная печать. Усл.печ.л. 2,8 Уч.-изд.л. 2,8 Тираж 1 экз.

---

Брянский государственный технический университет

Кафедра «Информатика и программное обеспечение», тел. 56-09-84

241035, Брянск, бульвар 50 лет Октября, 7 БГТУ