НАДЕЖНОСТЬ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

Содержание

1. Основные понятия

Надежность программного обеспечения - способность программного продукта безотказно выполнять определенные функции при заданных условиях в течение заданного периода времени с достаточно большой вероятностью.

Степень надежности характеризуется вероятностью работы программного продукта без отказа в течение определенного периода времени.

Существует *4* основные составляющие функциональной надежности программных систем:

– *безотказность* - свойство программы выполнять свои функции во время эксплуатации;

– *работоспособность* - свойство программы корректно (так как ожидает пользователь) работать весь заданный период эксплуатации;

– *безопасность* - свойство программы быть не опасной для людей и окружающих систем;

– *защищенность* - свойство программы противостоять случайным или умышленным вторжениям в нее.

1.1. Модель анализа надежности программных средств

1.2. Факторы, влияющие на надежность ПО

К числу основных факторов, влияющих на надежность ПО отнесены:

– взаимодействие ПО с внешней средой (программно-аппаратная средства, трансляторы, ОС). Этот фактор вносит наименьший вклад в надежность ПО при современном уровне надежности аппаратуры, ОС и компиляторов;

– взаимодействие с человеком (разработчиком и пользователем) (см. например метрику Холстеда);

– организация ПО (проектирование, постановка задачи и способы их достижения и реализации) и качество его разработки. Этот фактор вносит наибольший вклад в надежность;

– тестирование.

1.3. Ошибки ПО

В борьбе со сложностью ПО используются две концепции**:**

Иерархическая структура. Иерархия позволяет разбить систему по уровням понимания (абстракции, управления). Концепция уровней позволяет анализировать систему, скрывая несущественные для данного уровня детали реализации других уровней. Иерархия позволяет понимать, проектировать и описывать сложные системы.

Независимость. В соответствии с этой концепцией, для минимизации сложности, необходимо максимально усилить независимость элементов системы. Это означает такую декомпозицию системы, чтобы её высокочастотная динамика была заключена в отдельных компонентах, а межкомпонентные взаимодействия (связи) описывали только низкочастотную динамику системы.

Методы обнаружения ошибок, которые базируются на введении в ПО системы различных видов избыточности:

Временная избыточность. Использование части производительности ЭВМ для контроля исполнения и восстановления работоспособности ПО после сбоя.

Информационная избыточность. Дублирование части данных информационной системы для обеспечения надёжности и контроля достоверности данных.

Программная избыточность включает в себя: взаимное недоверие - компоненты системы проектируются, исходя из предположения, что другие компоненты и исходные данные содержат ошибки, и должны пытаться их обнаружить; немедленное обнаружение и регистрацию ошибок; выполнение одинаковых функций разными модулями системы и сопоставление результатов обработки; контроль и восстановление данных с использованием других видов избыточности.

Задача обеспечения ПО устойчивости к ошибкам направлены на применение методов минимизации ущерба, вызванного появлением ошибок, и включают в себя:

- обработку сбоев аппаратуры;
- повторное выполнение операций;
- динамическое изменение конфигурации;
- сокращенное обслуживание в случае отказа отдельных функций системы;
- копирование и восстановление данных;
- изоляцию ошибок.

Дается *4* группы принципов обеспечения надежности:

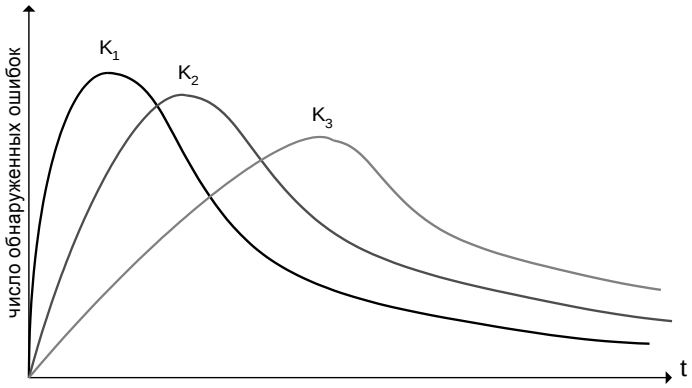
– предупреждение ошибок;
– обнаружение ошибок;
– исправление ошибок;
– обеспечение устойчивости к ошибкам.

Действия, направленные на минимизацию ошибок и сбоев:

– предотвращение ошибок за счет структурного программирования;
– сокрытие информации или дозированный доступ к данным со стороны программных средств и объектов в объектно-ориентированном программировании;
– отладка;
– устойчивость к сбоям;
– обработка исключительных ситуаций (перехват ошибок, например, деление на ноль) и локализация ошибок и сбоев;
– восстановление программы после сбоя

В соответствии с ГОСТ 19.004-80 различают следующие *виды работ, направленные на устранение ошибок в ПО*: проверка, отладка и испытание программы.

Чем интенсивнее использование программного изделия (ПИ), тем быстрее выявляются в нем ошибки. На рисунке приведена зависимость числа обнаруженных ошибок от числа использующих ПИ пользователей:



где K – число пользователей, $K_1 > K_2 > K_3$.

Рис. 1 Интенсивность обнаружения ошибок от интенсивности использования

Процентные частоты появления ошибок в ПО по типам ошибок представлены в таблице 1.

Таблица 1 – Процентные частоты появления ошибок в ПО

ТИП ОШИБКИ	ЧАСТОТА ПОЯВЛЕНИЯ, %
Не полная или ошибочная спецификация	28
Отклонение от спецификации	12
Пренебрежение правилами программирования	10
Ошибочная выборка данных	10
Ошибочная логика или последовательность операций	12
Ошибочные арифметические операции	9
Нехватка времени для решения	4
Ошибка обработки прерываний	4
Ошибка в исходных данных	3
Неточная запись	8

Как видно из таблицы 1 основное количество ошибок делается из-за неверной спецификации или ТЗ. Эти ошибки, в свою очередь, могут быть разделены на следующие категории:

Таблица 2 – Категории ошибок в ПО

ПРИЧИНА ОШИБКИ	ЧАСТОТА ПОЯВЛЕНИЯ, %
Ошибки в числовых значениях	12
Недостаточные требования к точности	4
Ошибочные символы или знаки	2
Ошибки оформления	15
Неправильное описание или требование к аппаратуре	2
Исходные данные для разработки неполные, неточные или ошибочные	52
Двусмысленность требований	13

Из этих таблиц, кстати, следует, на что нужно обращать особое внимание при проведении валидации и верификации ПО (верификация отвечает на вопрос, правильно ли и качественно ли создана программа, а валидация (или аттестация) - на вопрос правильно ли работает программа).

1.4. Средства и способы повышения надёжности ПО

На основании методов обнаружения ошибок были разработаны следующие *средства* повышения надёжности ПО.

Средства, использующие временную избыточность: авторизация доступа пользователей к системе, анализ доступных пользователю ресурсов, выделение ресурсов согласно ролям и уровням подготовки пользователей, разграничение прав доступа пользователей к отдельным задачам, функциям управления, записям и полям баз данных.

Средства обеспечения надёжности, использующие информационную избыточность: ссылочная целостность баз данных обеспечивается за счёт системы внутренних уникальных ключей для всех информационных записей системы, открытая система кодирования, позволяющая пользователю в любой момент изменять коды любых объектов классификации, обеспечивает стыковку системы классификации АИС делопроизводство с ПО других разработчиков, механизмы проверки значений контрольных сумм записей системы, обеспечивают выявление всех несанкционированных модификаций (ошибок, сбоев) информации.

Способы обеспечения и повышения надежности ПО:

- усовершенствование технологии программирования (например, формальное описание этапов программирования при помощи языка UML);
- выбор алгоритмов, не чувствительных к различного рода нарушениям вычислительного процесса (использование алгоритмической избыточности);
- резервирование программ - *N*-версионное программирование;
- верификация и валидация программ с последующей коррекцией.

1.5. Проблемы исследования надежности ПО

К основным проблемам исследований надежности ПО относятся:

- прежде всего - разработка методов оценки и прогнозирования надежности ПО;
- определение основных факторов, влияющих на надежность ПО;
- разработка методов, обеспечивающих достижение заданного уровня надежности ПО;
- совершенствование методов повышения надежности ПО в процессе проектирования и эксплуатации.

1.6. Тестирование ПО

Важным этапом жизненного цикла ПО, определяющим качество и надёжность системы, является **тестирование**. *Тестирование* - процесс выполнения программ с намерением найти ошибки и включает в себя следующие этапы:

- автономное тестирование;
- тестирование сопряжений;
- тестирование функций;
- комплексное тестирование;
- тестирование полноты и корректности документации;
- тестирование конфигураций.

Надежность ПО повышается также с помощью применения различных *методов тестирования*. Полное тестирование ПО невозможно. Обычно применяют следующие виды тестирования:

– тестирование ветвей;

– математическое доказательство правильности алгоритма решения задачи (в некоторых работах именно в этом смысле употребляется слово верификация).

– символическое тестирование (или с помощью специально подобранных тестовых наборов), еще называется статическим тестированием. Удобно при локализации ошибки, проявление которой выявлено при конкретном узком или строго заданном диапазоне входных значений;

– динамическое тестирование (с помощью динамически генерируемых входных данных), что удобно при быстром тестировании во всем широком диапазоне входных параметров;

– тестирование путей выполнения программы;

– функциональное тестирование;

– проверки по времени выполнения программы;

– проверка по использованию ресурсов и стрессовое тестирование.

2. Классификация показателей качества ПО

- По количеству характеризуемых свойств различают единичные и комплексные показатели. Единичные показатели качества характеризуют одно из свойств ПС, комплексный – несколько. Комплексные показатели могут быть групповыми, обобщенными или интегральными.
- В зависимости от места применения в процедуре оценки уровня качества ПС различают базовые и относительные показатели. *Базовым значением показателя качества продукции* называют значение показателя, принятое за основу при сравнительной оценке качества продукции. *Относительное значение показателя качества продукции* представляет собой отношение фактического значения показателя качества оцениваемой продукции к базовому значению этого показателя.
- По стадии определения значений показателей качества различают *прогнозируемые, проектные, производственные и эксплуатационные показатели*. Прогнозируемыми показателями оперируют на стадиях выполнения научно-исследовательских работ и составления ТЗ на разработку ПС, т. е. на тех стадиях, когда нет еще ни детального проекта ПС, ни, тем более, самого ПС. Значения прогнозируемых показателей в основном определяют на основе интуиции и опыта аналогичных разработок, поэтому эти показатели носят субъективный характер.

Значения проектных показателей определяют на основе анализа проектов ПС (эскизного, технического, рабочего), а также путем испытания опытного образца ПС. Эти показатели носят более объективный характер. Степень их достоверности зависит от эффективности используемых инструментальных средств анализа и испытания.

Производственные показатели мало отличаются от проектных, особенно если изготовление ПС сводится к простому копированию. Если же копированию предшествуют операции сборки или генерации ПС, то производственные показатели качества таких ПС могут существенно отличаться от проектных.

Значения эксплуатационных показателей определяют по результатам промышленной эксплуатации ПС. При соблюдении определенных правил сбора и обработки данных о качестве ПС в процессе эксплуатации эксплуатационные показатели дают наиболее объективную и достоверную оценку. Только по этим показателям можно произвести действительную оценку научно-технического уровня и качества ПС.

Около 50 % частных показателей можно определить автоматически с помощью ЭВМ, 25 % —с помощью компаратора. Таким образом, оценка около 75 % показателей может быть формализована. Оценка 20 % показателей может быть произведена только квалифицированным специалистом. Большинство показателей устанавливают путем статического анализа программ и лишь около 5 % — в процессе динамических испытаний (Данные соответствуют положению в этой области в 80-е годы).

2.2. Основные показатели надежности ПО

1. *Вероятность безотказной работы* $P(t_з)$ – это вероятность того, что в пределах заданной наработки отказ системы не возникает.

2. *Вероятность отказа* – вероятность того, что в пределах заданной наработки отказ системы возникает.

Это показатель, обратный предыдущему.

$$Q(t_з) = 1 - P(t_з) \tag{2.1}$$

где $t_з$ – заданная наработка, ч.;

$Q(t_з)$ – вероятность отказа.

3. *Интенсивность отказов системы* – это условная плотность вероятности возникновения отказа ПИ в определенный момент времени при условии, что до этого времени отказ не возник.

$$\chi(t) = \frac{f(t)}{P(t)} \tag{2.2}$$

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

$$f(t) = \frac{d}{dt}Q(t) = \frac{d}{dt}(1 - P(t)) = \frac{d}{dt}P(t) \tag{2.3}$$

Существует следующая связь между интенсивностью отказов системы и вероятностью безотказной работы

$$P(t) = \exp\left(-\int_0^t \chi(t)dt\right) \tag{2.4}$$

В частном случае, при

$$P(t) = \exp(-\chi t) \tag{2.5}$$

$$\chi(t) = const$$

Если в процессе тестирования фиксируется число отказов за определённый временной интервал, то интенсивность отказов системы есть число отказов в единицу времени.

4. *Средняя наработка на отказ* T_i - математическое ожидание времени работы ПИ до очередного отказа:

$$T_i = \int_0^t t \cdot f(t)dt \tag{2.6}$$

Иначе среднюю наработку на отказ T_i можно представить:

i=1;

$$T_i = \frac{t_1 + t_2 + t_3 ... t_n}{n} = \left(\frac{i}{n}\right) \cdot \sum_{i=1}^{n} t_i \qquad (2.7)$$

где t - время работы ПИ между отказами, с.

n – количество отказов.

5. *Среднее время восстановления T* - математическое ожидание времени восстановления - *t*; времени, затраченного на обнаружение и локализацию отказа - *t* ; времени устранения отказа - *t* ; времени пропускной проверки работоспособности - *t : t = t + t + t ,*

где *t* - время восстановления после i-го отказа.

n

T = i/nt ,

i=1

где *n* - количество отказов.

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

6. *Коэффициент готовности K* - вероятность того, что ПИ ожидается в работоспособном состоянии в произвольный момент времени его использования по назначению:

K = T / (T + T).

Необходимо стремиться повышать уровень надежности ПИ, но достижение 100%-ной надежности лежит за пределами возможного. Количественные показатели надежности могут использоваться для оценки достигнутого уровня технологии программирования, для выбора метода проектирования будущего программного средства.

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 программных средств параметров.

7. Все приведенные показатели надежности ПО характеризуют наличие ошибок программы (производственных дефектов), но ни один из них не характеризует характер этих ошибок и возможные их последствия. Поэтому предлагается ввести новый показатель надежности ПО – *средняя тяжесть ошибок* (СТО):

$$B=1/Q\ СУММ(bi*pi*zi) \qquad ,(2.8)$$

где *Q* – вероятность сбоя ПО;

СУММ – оператор суммирования по переменной i;

bi – функция принадлежности тяжести последствий ошибки, возникшей при i-ом наборе входных данных, к максимально тяжелым последствиям;

pi – вероятность ввода i-го набора входных данных при эксплуатации ПО;

zi – дихотомическая переменная, равная 1, если при i-ом наборе входных данных был зафиксирован сбой, и 0 в противном случае;

m – общее число наборов входных данных.

Значение показателя надежности СТО лежит на интервале [0;1]. Чем ближе значение СТО к единице, тем тяжелее последствия ошибок ПО, и тем менее надежна программа. Близость СТО к нулю показывает незначительность последствий ошибок программы.

Введение нового показателя надежности ПО позволило различать по надежности программные продукты, вероятности сбоя которых имеют один и тот же порядок. К тому же, говоря о надежности ПО, пользователь желает получить не столько безошибочное ПО, сколько безопасное. А именно безопасность ПО характеризует СТО. Значение этого показателя субъективно и может быть различным для одного и того же программного продукта в зависимости от области его применения. Это объясняется тем, что при использовании конкретного ПО, например, для выполнения студенческих расчетов и для выполнения конструкторских расчетов в космической промышленности последствия ошибок программы – несопоставимы. В ряде случаев, если к ПО предъявляются жесткие требования, лучше оценивать максимальную тяжесть ошибок ПО.

Таким образом, оценивая вероятность сбоя ПО и СТО ПО, получаем многостороннюю оценку надежности ПО.

3. Модели надежности ПО

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

Аналитические модели дают возможность рассчитывать 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 надёжности от числа межмодульных связей, количества циклов в модулях и т.д. Часто эмпирические модели не дают конечных результатов показателей надёжности, однако они включены в классификационную схему, так как развитие этих моделей позволяет выявлять взаимосвязь между сложностью АСОД и его надежно-

стью. Эти модели можно использовать на этапе проектирования ПО, когда осуществляется разбивка на модули и известна его структура.

Аналитические модели представлены двумя группами: динамические модели и статические. В *динамических* поведение ПС (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. Если фиксируются интервалы каждого отказа, то получается непрерывная картина появления отказов во времени (группа динамических моделей с непрерывным временем). С другой стороны, может фиксироваться только число отказов за произвольный интервал времени.

3.1. Динамические модели надежности

3.1.1. Модель Шумана

Исходными данными для модели Шумана, которая относится к динамическим моделям дискретного времени, собираются в процессе тестирования АСОД в течение фиксированных или случайных временных интервалов. Каждый энтервал - это стадия, на котором выполняется последовательность тестов и фиксируется некоторое число ошибок.

Модель Шумана может быть использована при определенном образе организованной процедуре тестирования. Использование модели Шумана предполагает, что тестирование поводиться в несколько этапов. Каждый этап представляет собой выполнение на полном комплексе разработанных тестовых данных. Выявление ошибки регистрируется, но не исправляются. По завершении этапа на основе собранных данных о поведении ПО на очередном этапе тестирования может быть использована модель Шумана для расчета количественных показателей надежности. При использовании модели Шумана предполагается, что исходное количество ошибок в программе постоянно, и в процессе тестирования может уменьшаться по мере того, как ошибки выявляются и исправляются.

Предполагается, что до начала тестирования в ПО имеется E_t ошибок. В течении времени тестирования *t1* в системе обнаруживается E_c ошибок в расчете на команду в машинном языке.

Таким образом, удельное число ошибок на одну машинную команду, оставшуюся в системе после *t1* времени тестирования, равно:

$$E(t)_r = \frac{E_t}{I_t} E(t)_c \qquad (3.1)$$

где I_t – общее число машинных команд, которое предполагается в рамках этапа тестирования.

Предполагаем, что значение функции частоты отказов *Z(t)* пропорционально числу ошибок, оставшихся в ПП после израсходованного на тестирование времени t.

$$Z(t) = C \cdot E(\tau) \tag{3.2}$$

где *C*- некоторая константа,

t – время работы ПП без отказа, ч.

Тогда, если время работы ПП без отказа t отсчитывается от точки *t = 0*, а *t1* остается фиксированным, функция надежности, или вероятность безотказной работы на интервале времени от 0 до t, равна:

$$R(t,t1) = \exp\left[C\left(\frac{E_t}{I_t} - E(t1)_C \right) \cdot t \right] \tag{3.3}$$

$$t_{cp} = \frac{1}{\left[C \cdot \left(\frac{E_t}{I_t} - E(t1)_C \right) \cdot t \right]} \tag{3.4}$$

Из величин, входящих в формулы (4.2) и (4.3) ,не известны начальное значение ошибок в ПП (E_t) и коэффициент пропорциональности – С. Для их определения прибегают к следующим рассуждениям. В процессе тестирования собирается информация о времени и количестве ошибок на каждом прогоне, т.е общее время тестирования *t1* складывается из времени каждого прогона:

$$t1 = t_1 + t_2 + t_3 + ... + t_n \tag{3.5}$$

Предполагая, что интенсивность появления ошибок постоянна и равна χ, можно вычислить её как число ошибок в единицу времени:

где A_i – количество ошибок на i-м прогоне.

$$\chi = \frac{\sum_{i=1}^{k} A_i}{t1} \tag{3.6}$$

$$t_{cp} = \frac{t1}{\sum_{i=1}^{k} A_i}$$

Имея данные для двух различных моментов тестирования t_A и t_b , которые выбираются произвольно с учетом требования, чтобы $E_c(t_b) > E_c(t_A)$, можно сопоставить уравнения (3.4) и (3.6) при t_A и t_b :

$$\frac{1}{\chi(t1_A)} = \frac{1}{\left[C \cdot \left(\frac{E_t}{I_t} - E(t1_A)_C \right) \right]} \tag{3.7}$$

$$\frac{1}{\chi(t1_b)} = \frac{1}{\left[C \cdot \left(\frac{E_t}{I_t} - E(t1_b)_C \right) \right]} \tag{3.8}$$

Вычисляя отношения (3.7) и (3.8), получим

$$E_T = \frac{E_T \cdot \left[\frac{\chi(t1_b)}{\chi(t1_A)} \cdot E(t1_A)_C - E(t1_b)_C \right]}{\left[\frac{\chi(t1_b)}{\chi(t1_A)} \right]^{-1}} \qquad (3.9)$$

Подставив полученную оценку параметров E_t в выражение (3.7), получим оценку для второго неизвестного параметра:

$$C = \frac{\chi(t1_A)}{\frac{E_t}{I_t} - E(t1_A)_C} \qquad (3.10)$$

Получив неизвестные E_t и C, можно рассчитать надежность программы по формуле (3.3).

Достоинство этой модели заключается в том, что можно исправлять ошибки, внося изменения в текст программы в ходе тестирования, не разбивая процесс на этапы, чтобы удовлетворить требованию постоянства числа машинных инструкции.

3.1.2. Модель La Padula

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

Надёжность ПП в течений *i* –го этапа:

$$R(t) = R(\infty) - \frac{A}{(i)} \qquad (3.11)$$

где *i = 1,2, ... n,*

A – параметр роста;

Предельная надежность ПП:

$$R(\infty) = \lim_{i \to \infty} R(i) \qquad (3.12)$$

Эти неизвестные величины можно найти, решив следующие уравнения:

$$\begin{gathered} \left[\sum_{i=1}^{m} \left(\frac{S_i - m_i}{S_i} - R(\infty) \right) + \frac{A}{i} \right] = 0 \\ \left[\sum_{i=1}^{m} \left(\frac{S_i - m_i}{S_i} - R(\infty) \right) + \frac{A}{i} \right] = 0 \end{gathered} \qquad (3.13)$$

где S_i – число тестов;

m_i – число отказов во время i-го этапа;

m – число этапов;

$i = 1,2 \ldots m.$

Определяемый по этой модели показатель есть надежность АСОД на i-м этапе.

$$R(i) = R(\infty) - \frac{A}{i} \qquad (3.14)$$

где $i = m+1, m+2 \ldots$

Преимущество данной 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 её выполнения.

3.2. Статические модели надежности

Статические модели принципиально отличаются от динамических прежде всего тем, что в них не учитывается время появления ошибок в процессе тестирования и не используется никаких предположений о поведении функции риска. Эти модели строятся на твердом статическом фундаменте.

3.2.1. Модель Миллса

Использование этой модели предполагает необходимость перед началом тестирования искусственно вносить в программу (засорять) некоторое количество известных ошибок. Ошибки вносятся случайным образом и фиксируются в протоколе искусственных ошибок. Специалист, проводящий тестирование, не знает ни количества ошибок, 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, собирается статистика об ошибках. В момент оценки надежности по протоколу искусственных ошибок все ошибки делятся на собственные и искусственные. Соотношение:

$$N = \frac{S \cdot n}{V} \qquad (3.15)$$

дает возможность оценить N – первоначальное количество ошибок в программе. В данном соотношении, которое называется формулой Миллса, S – количество искусственно внесенных ошибок, n – число найденных собственных ошибок, V – число обнаруженных к моменту оценки искусственных ошибок.

Вторая часть модели связанна с проверкой гипотезы от N. Предположим, что в программе имеется K собственных ошибок, и внесем в нее еще S оши-

бок. В процессе тестирования были обнаружены все *S* внесенных ошибок и n собственных ошибок.

Тогда по формуле Миллса мы предполагаем, что первоначально в программе было $N = n$ ошибок. Вероятность, с которой можно высказать такое предположение, возможно рассчитать по следующему соотношению:

$$C = 1, \quad \text{если} \quad n > K$$
$$C = \frac{S}{S + K + 1}, \quad \text{если} \quad n <= K \tag{3.16}$$

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 (5.2) для расчета C не может быть в случае, когда не обнаружены все искусственно рассеяние ошибки. Для этого случая, когда оценка надежности производиться до момента обнаружения всех S рассеянных ошибок, величина C рассчитывается по модифицируемой формуле

$$C = 1, \quad \text{если} \quad n > K$$
$$C = \frac{\frac{s}{V-1}}{\left[\frac{S+K+1}{V+K}\right]}, \text{если} \quad n <= K \tag{3.17}$$

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

$$\frac{a}{b} = \frac{a!}{b!(a-b)!} \tag{3.18}$$

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:

$$C = \frac{\frac{10}{4}}{\frac{11}{5}} = \frac{10!\cdot 5!\cdot 6!}{4!\cdot 6!\cdot 11!} = 0.45 \tag{3.19}$$

Если при тех же исходных условиях оценка надежности производится в момент, когда обнаруженны 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 отме-

чать 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.

3.2.2. Модель Липова

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

$$Q(n,V) = \frac{m}{n+V} \cdot q^{n+V} \cdot (1-q)^{m-n-\frac{V \cdot \frac{N}{n} \cdot \frac{S}{V}}{\left(\frac{N+S}{n+V}\right)}} \qquad (3.20)$$

где m – количество тестов, используемых при тестировании;

q - вероятность обнаружения ошибки в каждом из m тестов , рассчитывается по формуле :

$$q = \frac{n+V}{n} \qquad (3.21)$$

S – общее количество искусственно внесенных ошибок;

N – количество собственных ошибок, имеюшихся в ПС до начала тестирования.

Для использования модели Липова должны выполняться следующие условия:

$$\begin{aligned} &N \geq n \geq 0 \\ &S \geq V \geq 0 \\ &m \geq n+V \geq 0 \end{aligned} \qquad (3.22)$$

Модель Липова дополняет модель Миллса, дав возможность оценить вероятность обнаружения определенного количества ошибок к моменту оценки.