***Слайд 1***

**Лекция 4**

**МОДЕЛИ НАДЕЖНОСТИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ**

***Слайд 2***

**Учебные вопросы:**

**1. Основные показатели надежности ПО.**

**2. Модели надежности ПО.**

**2.1. Аналитические динамические модели надежности программных средств.**

**2.2.1. Модель Шумана.**

**2.2.2. Модель Джелинского-Моранды.**

**2.2. Аналитические статические модели надежности программных средств.**

**2.2.1. Модель Миллса.**

**2.2.2. Модель Нельсона.**

**2.2.3. Модель Липова.**

***Слайд 3***

## 1. Основные показатели надежности ПО

Термин модель надежности программного обеспечения, как правило, относится к математической модели, построенной для оценки зависимости надежности программного обеспечения от некоторых определенных параметров. Значения таких параметров либо предполагаются известными, либо могут быть измерены в ходе наблюдений или экспериментального исследования процесса функционирования программного обеспечения.

Данный термин может быть использован так же применительно к математической зависимости между определенными параметрами, которые хотя и имеют отношение к оценке надежности программного обеспечения, но тем не менее не содержат ее характеристик в явном виде. *Например, поведение некоторой ветви программы на подмножестве наборов входных данных, с помощью которых эта ветвь контролируется, существенным образом связано с надежностью программы, однако характеристики этого поведения могут быть оценены независимо от оценки самой надежности. Другим таким параметром является частота ошибок, которая позволяет оценить именно качество систем реального времени, функционирующих в непрерывном режиме, и в то же время получать только косвенную информацию относительно надежности программного обеспечения (например, в предположении экспоненциального распределения времени между отказами).*

Одним из видов модели надежности программного обеспечения, которая заслуживает особого внимания, является так называемая феноменологическая, или эмпирическая, модель. При разработке моделей такого типа предполагается, что связь между надежностью и другими параметрами является статической. С помощью подобного подхода пытаются количественно оценить те характеристики программного обеспечения, которые свидетельствуют либо о высокой, либо о низкой его надежности. *Так, например, параметр* **сложность** *программы характеризует степень уменьшения уровня ее надежности, поскольку усложнение программы всегда приводит к нежелательным последствиям и в том числе к неизбежным ошибкам программистов при составлении программ и трудности их обнаружения и устранения.* Иначе говоря, при разработке *феноменологической модели* надежности программного обеспечения стремятся иметь дело с такими параметрами, соответствующее изменение значений которых должно приводить к повышению надежности программного обеспечения.

***Слайд 4***

В общем случае систему программного обеспечения можно рассматривать как подсистему некоторой вычислительной системы.

Преимущества подобного представления с точки зрения *обеспечения заданных характеристик системы, гарантирующих правильное функционирование программ при наличии отказов вычислительной системы*, заключаются в том, что отказы системы программного обеспечения определяются как «отклонения от правильного хода выполнения программы вследствие ошибок, допущенных в процессе преобразования исходного алгоритма в действующую программу».

Вычислительную систему можно сделать устойчивой против отказов путем введения избыточных программных и аппаратных средств. *Программное обеспечение может быть построено и таким образом, что частота возникновения отказов определенных классов, характерных для данной системы может быть существенно уменьшена или практически сведена к нулю.*

Используя приведенное определение **отказа** программного обеспечения, можно определить надежность системы программного обеспечения как **вероятность того, что отказ программного обеспечения, вызывающий отклонение получаемого выхода от требуемого за допустимые пределы, не произойдет при определенных условиях внешней среды в течение заданного периода наблюдения.**

*Обсудим это определение более подробно.*

**Во-первых,** следует иметь в виду, что не все отказы приводят к уменьшению надежности программного обеспечения, а только те, которые вызывают отклонение выхода от требуемого за допустимые пределы.

**Во-вторых,** под определенными условиями внешней среды следует понимать описание входных данных и состояние вычислительной системы в процессе выполнения программы. *Состояние вычислительной системы часто описывается объемом выделенной памяти с быстрым доступом (т. е. оперативной памяти) и зависит от требований к программному обеспечению в части его* способности *нормально функционировать при наличии отказов. Под способностью подразумеваются свойства программного обеспечения, которые закладываются при его* ***проектировании*** *(например, возможность смены программ в памяти; возможность возобновления работ с некоторых контрольных точек и т. д.). В общем случае работа в условиях внешней среды, не предусмотренных техническим заданием и проектом программного обеспечения, приведет к снижению надежности последнего.*

Что касается заданного периода наблюдений, то он, как правило, представляет собой время, необходимое для выполнения поставленной задачи.

*Выделение определенного интервала времени наблюдения для оценки качества программного обеспечения, по-видимому, целесообразно в случае* ***систем реального времени****, в которых непредсказуемыми являются число прогонов любой из действующих программ, состояние базы данных и моменты начала выполнения той или иной программы. В условиях так называемого* **пакетного режима работы**, *т. е. когда состояние системы, включая базу данных, достоверно известно, в качестве периода наблюдений следует выбрать* **рабочий цикл,** *или* **прогон**. *В общем случае каждый раз перед повторным выполнением программы необходимо либо восстанавливать состояние памяти, либо осуществлять серию последовательных прогонов программы, при которых определенным образом последовательно изменяется состояние базы данных.*

***Слайд 5***

*Рассмотрим некоторые модели надежности программного обеспечения, но сначала введем некоторые понятия.*

1. *Вероятность безотказной работы P(tз)* – 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)* (4.1)

где *t з* – заданная наработка, ч.;

*Q(t з)* – вероятность отказа.

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

(4.2)

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

Существует следующая связь между интенсивностью отказов системы и вероятностью безотказной работы

(4.3)

В частном случае, при

(4.4)

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

***Слайд 6***

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

Иначе среднюю наработку на отказ *Тi* можно представить:

*i=1..n;*

где *t* - время работы ПО между отказами, (дн., ч., мин., сек.).

*n* – количество отказов.

5. *Среднее время восстановления Tв* - математическое ожидание времени восстановления работоспособного состояния ПО после отказа

*n* – количество восстановлений, равное числу отказов.

- время, затраченное на восстановление (обнаружение, поиск причины и устранение отказа).

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

***Слайд 7***

6. *Коэффициент готовности KГ*

*Процесс функционирования восстанавливаемого ПО можно представить как последовательность чередующихся интервалов работоспособности и восстановления (простоя).*

*Коэффициент готовности KГ* - вероятность того, что ПО ожидается в работоспособном состоянии в произвольный момент времени его использования по назначению. Этот показатель одновременно оценивает свойства работоспособности и возможности отладки ПО.

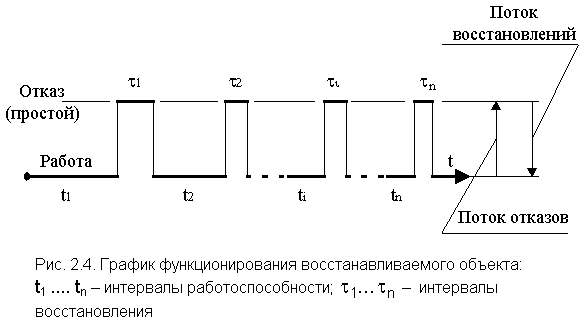


Рис. 1 График функционирования восстанавливаемого объекта

- интервалы работоспособности; – интервалы восстановления.

Для одного восстанавливаемого ПО коэффициент готовности определяется

*Из выражения (4.8) видно, что коэффициент готовности ПО может быть повышен за счет увеличения наработки на отказ и уменьшения среднего времени восстановления. Для определения коэффициента готовности необходим достаточно длительный календарный срок функционирования объекта.*

*Необходимо стремиться повышать уровень надежности ПО, 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.*

***Слайд 8***

7. *Все приведенные показатели надежности ПО характеризуют наличие ошибок программы (производственных дефектов), но* ***ни один из них не определяет характер этих ошибок и возможные их последствия****.*

Поэтому вводят иной показатель надежности ПО – *средняя тяжесть ошибок* (СТО):

где *Q* – вероятность сбоя ПО;

*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 (математическая переменная, которая может принимать только два значения.), равная 1, если при i-ом наборе входных данных был зафиксирован сбой, и 0 в противном случае;

*n* – 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.*

***Слайд 9***

2. 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 надежностью. 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* ***(группа динамических моделей с дискретным временем)****.*

***Слайд 10***

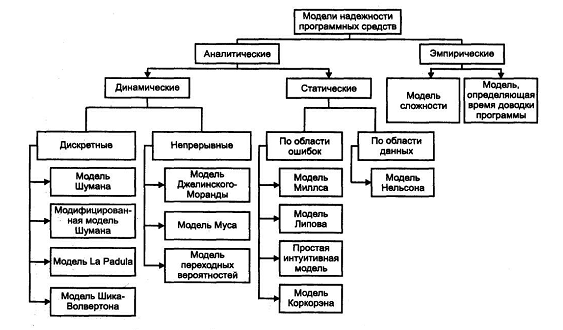


Рис. 2 Основные модели надежности программных средств

***Слайд 11***

2.1. Аналитические динамические модели надежности программных средств

### 2.1.1. Модель Шумана

Целесообразность применения этой модели для оценки надежности программного обеспечения зависит от принятых допущений и условий, наиболее существенным из которых является условие существования «программы исследователя системы» - она должна снабжать испытуемые программные средства входными данными, отражающими реальные условия функционирования. Как правило, эти данные называются функциональным разрезом и определяются через распределение вероятностей значений входных переменных.

*Остальные допущения и условия носят статический характер и не связаны с какими-либо специфическими свойствами программного обеспечения. По существу они сводятся к следующему.*

**А1.** Общее число команд в программе на машинном языке постоянно.

**А2.** Предполагается, что в начальный момент компоновки программных средств в систему программного обеспечения в них имеется *ЕТ* ошибок. С этого момента начинается отсчет времени отладки , *которое включает затраты времени на выявление ошибок с помощью тестов, на контрольные проверки и т. п.,* при этом время исправного функционирования системы не учитывается.

В течение времени отладки устраняется ошибок в расчете на одну команду в машинном языке. Таким образом, удельное число ошибок на одну машинную команду, остающихся в системе после месяцев отладки, равно

, (4.11)

где — общее число машинных команд, которое предполагается постоянным.

***Слайд 12***

**А2.** Предполагается, что значение функции количества ошибок или **интенсивности отказов** *Z(t)* пропорционально числу ошибок, оставшихся в программе после израсходованного на тестирование времени *τ*.

, (4.12)

где *С* - некоторая постоянная (коэффициент пропорциональности), *t* - время работы программы без отказов.

***Слайд 13***

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

; (4.13)

(4.14)

В уравнении присутствуют две неизвестные – начальное значение ошибок 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 складывается из времени каждого прогона:

(4.15)

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

где *Аi* – количество ошибок на i-м прогоне.

***Слайд 14***

Эти неизвестные оцениваются путем пропуска функционального теста в двух точках переменной оси отладки *τa* и *τв*, выбранных так, что *εc(τa)<εc(τd)*.

Имея данные для двух различных моментов тестирования *τa* и *τв*, можно сопоставить с использованием уравнения (4.14) при *τa*и *τb*:

(4.16)

(4.17)

Из соотношений (4.16) и (4.17) найдем неизвестный параметр С и :

(4.18)

(4.19)

*Получив неизвестные: начальное значение ошибок и коэффициент пропорциональности С, можно рассчитать надежность программы по формуле (4.13).*

***Слайд 15***

### 2.1.2 Модель Джелинского *—* Моранды

*Данная модель, предназначенная для прогнозирования надежности программного обеспечения, представляет собой частный случай модели Шумана.* При разработке этой модели предполагалось, что значения интервалов времени отладки (*в шумановском смысле*) между обнаружением двух ошибок имеют экспоненциальное распределение с частотой ошибок (или интенсивностью отказов), пропорциональной числу не выявленных дефектов. Каждая обнаруженная в программе ошибка немедленно устраняется, и число оставшихся ошибок уменьшается на единицу. Таким образом, функция плотности распределения времени обнаружения *i*-й ошибки, отсчитываемого от момента выявления (*i-*1) -й ошибки, имеет вид

(4.20)

где ; k- коэффициент пропорциональности; N - число ошибок, первоначально присутствующих в программах.

Применяя для двух неизвестных этого уравнения (ими в выражении (4.20) являются величины *к* и*N*метод максимального правдоподобия, авторы этого метода оценки надежности программных средств получили следующую систему из двух уравнений:

(4.20)

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

*Уравнения, образующие систему (4.20), представляют модель Джелинского - Моранды, позволяющей оценить количество ошибок в программе до начала тестирования* ***N*** *по количеству обнаруженных ошибок n.*

***Слайд 16***

Продолжительность интервала тестирования *ti*, 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 т**тестами (т. е. тест с номером т**- 1 не обнаружил ошибки, в тесте т**ошибка была обнаружена), то интервал времени равен т**(рисунок 3).*



Рисунок 3 - Учет интервалов времени при выявлении ошибок

Важным условием применимости на практике модели Джелинского-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.

***Слайд 17***

Таким образом, модель Джелинского - 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 **Джелинского - Моранды** в некоторых источниках называют **моделью роста надежности**.

Модификация модели Джелинского — Моранды была предложена Волвертоном и Шиком. В основе модели Шика - Волвертона лежит предположение, согласно которому частота ошибок пропорциональна не только количеству ошибок в программах, но также и времени отладки, т. е. вероятность обнаружения ошибок с течением времени возрастает.

***Слайд 18***

2.2. Аналитические статические модели надежности программных средств.

### 2.2.1. Модель Миллса.

*Использование этой модели предполагает внесение в программу некоторого количества известных ошибок. Они вносятся случайным образом и фиксируются в протоколе искусственных ошибок. Специалист, проводящий тестирование, не знает ни количества, ни характера внесенных ошибок до момента оценки показателей надежности по модели Миллса. Предполагается, что все ошибки (как имеющиеся в исследуемой программе, так и искусственно внесенные) имеют равную вероятность быть найденными в процессе тестирования. Тестируя программу в течение некоторого времени, собирают статистику об ошибках. В момент оценки надежности по протоколу искусственных ошибок все ошибки делятся на ошибки собственно программы и искусственные.*

Статистическая модель Миллса позволяет оценить не только количество ошибок до начала тестирования, но и степень отлаженности программ.

Для оценки количества ошибок в программе до начала тестирования используется выражение:

(4.21)

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.

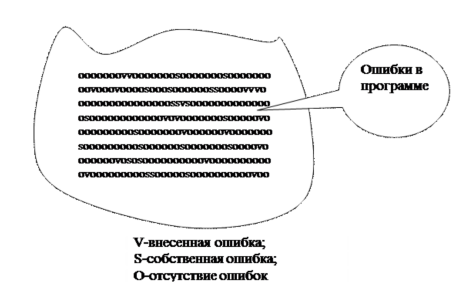


Рис. 4 Ошибки в программе.

***Слайд 19***

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

(4.22)

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 (4.21),

*r* - верхний предел (максимум) предполагаемого количества «собственных» ошибок в программе.

Выражения (4.21) и (4.22) 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 выражения (4.22) следует использовать более сложное комбинаторное выражение (4.23). Если обнаружено только V ошибок из W преднамеренно внесенных, используется выражение:

(4.23).

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 *r*, которое основывается исключительно на интуиции и опыте человека, проводящего оценку, т.е. допускается большое влияние субъективного фактора.

***Слайд 20***

### 2.2.2. Модель Нельсона.

Данная модель была создана в аэрокосмической компании TRW в рамках разработки математической теории надежности программного обеспечения (МТНПО). *Эта теория включает точные математические определения основных элементов надежности программного обеспечения, математическое описание связи между элементами надежности, математические операции над элементами с целью изучения новых аспектов надежности программного обеспечения.*

**Общее описание модели Нельсона.**

Интуитивное определение надежности программного обеспечения, приведенное ранее, может быть уточнено в статистическом смысле на основе следующих простых соображений:

* машинная программа *р* может быть определена как описание некоторой вычисляемой функции *F* на множестве *Е* всех значений наборов входных данных, таких, что каждый элемент *Еi* множества *Е* представляет собой набор значений данных, необходимый для выполнения прогона программы: ;
* выполнение программы *р* приводит к получению для каждого *Еi* определенного значения функции *F(Еi)*;
* множество *Е* определяет все возможные вычисления в программе *р*,   
  т. е. каждому набору входных данных *Еi*, соответствует некоторый прогон *р*, и наоборот, каждому прогону соответствует некоторый набор входных данных *Еi*;

***Слайд 21***

* наличие дефектов в программе *р* приводит в тому, что ей на самом деле соответствует функция *F'*, отличная от заданной функции *F*;
* для некоторого *Еi* отклонение выхода *F'(Еi)*, полученного в результате выполнения программы, от желаемого значения *F(Еi)* находится в допустимых пределах , т. е.

— для всех остальных *Еi*, образующих подмножество *Ее* множества *Е*, выполнение программы *р* не обеспечивает приемлемого результата, т. е.

,

либо работа программы прекращается преждевременно или выполнение программы не может закончиться (программа зацикливается). **Все такие случаи называются «рабочими отказами».**

*Каждое Еi представляет* ***возможную комбинацию значений****, которые могут быть приписаны* ***входным переменным*** *(переменным, значения которых должны быть зафиксированы в качестве входных данных программы р, для того чтобы сделать возможным ее выполнение). Число* ***N*** *возможных* ***Еi*** *очень велико, но оно конечно, так как с помощью машинного слова фиксированной длины возможно представить лишь конечное число различных значений переменной.*

Совокупность действий, включающая ввод *Еi*, выполнение программы *р*, которое оканчивается получением результата , или рабочим отказом, называется прогоном программы *р*.

***Слайд 22***

**Оценка надежности программного обеспечения**

***Заметим****, что значения входных переменных, образующие Еi,, не должны все одновременно подаваться на вход программы р.* Таким образом, вероятность *Р* того что прогон программы *р* приведет к рабочему отказу, равна вероятности, что набор входных данных *Еi*,, использованный в данном прогоне, принадлежит множеству *Ее*. Если обозначить через пе число различных наборов значений входных данных, содержащихся в *Ее*, то

есть вероятность того, что прогон программы на наборе входных данных *Еi*,, случайно выбранном из *Е* среди равновероятных, закончится рабочим отказом.

При этом

есть вероятность того, что прогон программы *р* на наборе входных данных *Еi*, случайно выбранном из *Е* среди априори равновероятных, приведет к получению приемлемого результата.

*Однако в процессе* ***реального функционирования программы*** *выбор входных данных из Е обычно осуществляется не с одинаковыми априорными вероятностями, а диктуется определенными условиями работы.* Эти условия могут быть охарактеризованы некоторым распределением вероятностей , того, что будет сделан выбор именно входных данных *Еi*. Множество вероятностей называется ***функциональным разрезом***.

***Слайд 23***

Распределение *Р* может быть определено через с помощью «динамической переменной» , которая принимает значение 0, если прогон программы на наборе *Еi* заканчивается вычислением приемлемого значения функции, и значение 1, если этот прогон заканчивается рабочим отказом. Тогда

есть вероятность того, что прогон программы *р* на входных данных *Еi*, выбранных в соответствии с распределением вероятностей закончится отказом, и

есть вероятность того, что прогон программы с выходными данными *Еi*, выбранными в соответствии с распределением вероятностей , приведет к правильному выполнению программы.

***Слайд 24***

Так как *R* — вероятность того, что единичный прогон программы не закончится отказом на наборе входных данных, выбранных в соответствии с распределением , то вероятность успешного выполнения п прогонов этой программы при независимом для каждого прогона выборе входных данных в соответствии с распределением *Р* будет равна

Таким образом, можно дать следующее математическое определение надежности машинной программы: **надежность программы — это вероятность безотказного выполнения п прогонов программы. Поэтому прогон является единичным испытанием программы.**

На практике обычно выбор входных данных для каждого прогона **нельзя считать независимым.** *Исключение составляют лишь такие последовательности прогонов, которые определяются возрастающими значениями некоторой входной переменной или некоторым порядком установленных процедур, как в случае программ, работающих в реальном масштабе времени.*

С учетом введенного определения функциональный разрез должен быть переопределен в терминах вероятностей выбора *Еi* в качестве входных данных при *j*-м прогоне из некоторой последовательности прогонов. Тогда вероятность того, что *j*-й прогон закончится отказом, может быть записана в виде

***Слайд 25***

Надежность *R(n)* программы *р* равна вероятности того, что в последовательности из п прогонов ни один из них не закончится отказом:

Эта, формула может быть представлена в виде

.

*Рассмотренный подход к оценке надежности программного обеспечения был проверен на двух программах, написанных различными специалистами на основании одного и того же технического задания. Разрез* pi*определялся таким образом, чтобы он приближенно отражал реальные условия функционирования программы. Выборка из 1000 наборов входных данных формировалась в соответствии с этим разрезом с помощью датчика случайных чисел. Было сделано 1000 прогонов каждой программы. Результаты каждого прогона анализировались, и отмечались те прогоны, которые заканчивались отказом. Для одной программы было зафиксировано три отказа, а для другой — 35. Рассчитанная оценка надежности R для первой программы равнялась 0,997, а для второй — 0,965. Программа, которой соответствовала более высокая надежность, имела более простую структуру, и этот факт подтверждает установившееся мнение, что усложнение структуры программ ведет к снижению их надежности.*

***Слайд 26***

**Сравнение модели Нельсона с другими моделями.**

В моделях Шумана и Джелинского — Моранды в качестве независимой переменной фигурирует время и надежность вычисляется по формуле .

Значение интенсивности отказов h предполагается постоянным в течение всего периода функционирования системы и изменяется только при обнаружении и устранении ошибок, после чего время t снова отсчитывается от нуля.

*Поскольку эта формула может быть получена как частный случай модели Нельсона при некоторых допущениях, целесообразно определить условия, при которых верны модели Шумана и Джелинского — Моранды.*

*Эти условия таковы:*

* время t должно интерпретироваться как суммарное время работы программы относительно некоторого определенного начального момента времени;
* время t должно быть больше средней длительности выполнения одного прогона программы ;
* наборы входных данных для последовательных прогонов программы должны выбираться случайным образом в соответствии с законом распределения, приближенно отражающим реальные условия функционирования, относительно которых производится оценка надежности.

*Авторы и той и другой модели попытались расширить их в предположении, что интенсивность отказов пропорциональна числу оставшихся в программе ошибок, а затем применить эти модели к тестированию программ.*

*Вейсом и Коркорэном были предложены модели для определения степени повышения надежности программы в процессе тестирования, причем они попытались учесть эффект существования в программе нескольких источников ошибок.*

*Перечисленные модели нашли свое применение, но в них слабо учитывались свойства программ, режимы функционирования и стратегии испытаний. Эти модели основывались главным образом на таких общих закономерностях теории вероятностей, как экспоненциальная зависимость надежности от числа испытаний, и на весьма упрощенных представлениях относительно последствий программных ошибок. Хотя в этих моделях экспоненциальный характер зависимости надежности установлен достаточно точно, другие свойства программного обеспечения в них учитываются в лучшем случае весьма приближенно.*

*Модель Нельсона была разработана с учетом основных свойств машинных программ и использует методы теории вероятностей лишь в тех случаях, когда невозможно получение полной информации о том или ином факторе, например при ответе на вопрос, какой набор входных данных следует взять при следующем прогоне программы. Все приближения, принятые в модели, четко определены, и известны границы их применимости. Поскольку в основу модели Нельсона положены свойства программного обеспечения, она допускает развитие за счет более детального описания других аспектов надежности.*

*Вследствие отмеченных особенностей модели ее следует рассматривать как математическую теорию надежности программного обеспечения, а не как простую модель надежности.*

Литература:

1. Орлов С.А. Технологии разработки программного обеспечения : учеб. для вузов. - 3-е изд СПб.: Питер, 2004
2. Аллен Э. Типичные ошибки проектирования : [Пер. с англ.] СПб.: Питер, 2003
3. Фатрелл Р.Т. Управление программными проектами: достижение оптимального качества при минимуме затрат : [Пер. с англ.] М.: Вильямс, 2003
4. Липаев В.В. Надежность программных средств./ В.В. Липаев — М.: СИНТЕГ,1998.
5. Майерс Г. Надежность программного обеспечения: Пер. с англ/ Г. Майерс.. - М.: Мир, 1980.

Контрольные вопросы

1. Как классифицируются основные модели надежности программных средств.
2. На каких допущениях основана модель Шумана?
3. Как рассчитывается вероятность безотказной работы программы в модели Шумана?
4. Статистическая модель Миллса: суть, порядок расчета показателей?
5. Как оценить степень отлаженности программ, используя статистическую модель Миллса?
6. Как классифицируются основные модели надежности программных средств.
7. На каких допущениях основана модель Джелинского – Моранды?
8. Почему модель Джелинского – Моранды называют «модель роста надежности»?
9. Модель Нельсона: суть, порядок расчета показателей?
10. Что называется «прогоном программы»?
11. Как оценить надежность программы при не равновероятных выборах исходных данных и зависимых испытаниях?