

**МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ**  
**Федеральное государственное бюджетное образовательное учреждение**  
**высшего профессионального образования**  
**«УЛЬЯНОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»**  
**Факультет математики и информационных технологий**  
**Кафедра телекоммуникационных технологий и сетей**

*Е. Г. Чекал, А. А. Чичев*

# **НАДЕЖНОСТЬ ИНФОРМАЦИОННЫХ СИСТЕМ**

**Учебное пособие**

**Часть 1**

**Ульяновск 2012**

УДК 004.052  
ББК 32.973  
Ч-37

*Печатается по решению Ученого совета  
факультета математики и информационных технологий  
Ульяновского государственного университета*

**Рецензенты:**

доцент кафедры «Самолетостроение» ИАТУ УлГТУ, к.т.н. *Н. А. Попов*;  
доцент кафедры инженерной физики УлГУ, к.т.н. *П. В. Дубровский*

**Чекал Е. Г.**

**Ч-37 Надежность информационных систем** : учебное пособие :  
в 2 ч. Ч. 1 / Е. Г. Чекал, А. А. Чичев. – Ульяновск : УлГУ, 2012. –  
118 с.

В пособии рассматриваются основные понятия надежности информационных систем, подробно описаны модели надежности программного обеспечения и технических средств информационных систем, опирающиеся на теорию случайных событий и случайных величин.

Предназначено для студентов специальности 230400 «Информационные системы и технологии» различных форм обучения. Может использоваться студентами родственных специальностей.

УДК 004.052  
ББК 32.973

## О Г Л А В Л Е Н И Е

<b>Введение</b> .....	5
<b>1. Основные понятия надежности информационной системы</b> .....	6
1.1. Состояния информационной системы .....	7
1.2. События информационной системы .....	9
1.3. Примитивы надежности информационной системы .....	12
1.4. Типы ИС в аспекте надежности .....	13
1.5. Оценка надежности информационной системы .....	14
1.6. Обеспечение надежности информационной системы .....	16
<b>2. Элементы теории вероятностей и математической статистики</b> .....	18
2.1. Случайные события .....	18
2.2. Случайные величины .....	22
2.3. Числовые характеристики случайной величины .....	25
2.4. Закон больших чисел .....	28
2.5. Статистические характеристики случайной величины .....	29
2.6. Погрешность оценки параметров распределения .....	32
2.7. Проверка статистических гипотез .....	33
<b>3. Анализ надежности технических средств информационной системы</b> .....	36
3.1. Показатели надежности технических средств .....	36
3.2. Модели надежности технических средств .....	40
3.2.1. Экспоненциальный закон надёжности .....	40
3.2.2. Усеченный нормальный закон надёжности .....	41
3.2.3. Закон распределения Релея .....	42
3.2.4. Закон распределения Вейбулла .....	43
3.2.5. Гамма-распределение .....	44
3.2.6. Закон Пуассона .....	45
3.2.7. Экспериментальная оценка надежности .....	45
3.3. Структурная надежность технических средств .....	47
3.3.1. Структурно-логический анализ технических средств .....	47
3.3.2. Расчет структурной надежности на основе вероятностного метода .....	48
3.3.3. Расчет надежности передачи информации на основе логики-вероятностного метода .....	56
3.4. Повышение надежности технических средств .....	62
3.4.1. Резервирование технических средств .....	62
3.4.2. Расчет надежности систем с резервированием .....	65

<b>4. Надежность программного обеспечения информационной системы.....</b>	<b>68</b>
4.1. Основные понятия.....	68
4.2. Оценка надежности программных средств .....	69
4.2.1. Аналитические динамические модели.....	77
4.2.2. Аналитические статические модели .....	83
4.2.3. Эмпирические модели .....	86
4.3. Обеспечение надежности программных средств.....	87
4.3.1. Общие принципы обеспечения надежности .....	87
4.3.2. Предупреждение ошибок .....	88
4.3.3. Обеспечение примитивов надежности программного средства.....	90
4.3.4. Оценки стиля программирования.....	99
4.3.5. Тестирование программного обеспечения .....	107
Использованная и рекомендуемая литература.....	115

## ВВЕДЕНИЕ

Современное общество является информационным. Под информационным обществом понимают такую ступень в развитии цивилизации, которая характеризуется возрастанием роли информации во всех областях общественной жизни; созданием и развитием рынка информационных услуг, новых форм социальной и экономической деятельности; созданием глобального информационного пространства, обеспечивающего доступ к мировым информационным ресурсам; превращением информационных ресурсов общества в реальные ресурсы социально-экономического развития. Поэтому в настоящее время проблема надёжности является ключевой по отношению к современным информационным системам. От ее решения во многом зависят темпы их развития. Отказ в работе (в том числе и неправильное функционирование) информационных систем может привести к катастрофическим последствиям глобального масштаба. Вместе с тем следует отметить, что надёжность программного обеспечения стала играть более важную роль по сравнению с надёжностью аппаратной составляющей информационных систем. И связано это с возрастающей технологичностью проектирования и производства вычислительных средств, тогда как разработка программного обеспечения по-прежнему остается в большинстве своем искусством программирования.

Данное пособие состоит из четырех глав. В первой главе даны основные понятия надёжности информационной системы. Изложение опирается на ГОСТ 24.701-86 «Единая система стандартов автоматизированных систем управления. Надёжность автоматизированных систем управления. Основные положения».

Во второй главе приводятся необходимые сведения из теории вероятностей и математической статистики.

Третья глава посвящена надёжности технических средств информационных систем. Описаны основные показатели надёжности, приведены модели надёжности. Показано, как проводить экспериментальную оценку надёжности.

Поскольку пособие ориентировано на студентов ИТ-специальностей, много внимания уделено вопросам надёжности программного обеспечения, обычно недостаточно подробно освещаемым в подобной литературе. Рассмотрены различные классы моделей надёжности ПО, методы обеспечения примитивов надёжности ПО. Уделено особое внимание таким методам обеспечения надёжности ПО, как формирование правильного стиля программирования и тестирование ПО.

## 1. ОСНОВНЫЕ ПОНЯТИЯ НАДЕЖНОСТИ ИНФОРМАЦИОННОЙ СИСТЕМЫ

В настоящее время под *информационной системой* (ИС) понимается, прежде всего, автоматизированная система, которая представляет собой совокупность комплекса средств автоматизации, организационно-методических и технологических документов и специалистов, использующих их в процессе своей профессиональной деятельности, предназначенная для сбора, передачи, обработки, хранения и выдачи информации потребителям и состоящая из следующих основных компонентов:

- программное обеспечение (ПО);
- информационное обеспечение;
- технические средства (ТС);
- обслуживающий персонал.

Согласно ГОСТ 24.701-86 под *надежностью ИС* понимается свойство системы сохранять во времени в установленных пределах значения всех параметров, характеризующих способность системы выполнять требуемые функции в заданных режимах и условиях эксплуатации.

При решении вопросов, связанных с обеспечением требуемого уровня надежности ИС, необходимо учитывать следующие ее особенности:

- каждая ИС является многофункциональной системой, функции которой имеют различную значимость и, соответственно, характеризуются разным уровнем требований к надежности их выполнения;
- возможно возникновение некоторых исключительных (аварийных, критических) ситуаций, представляющих сочетание отказов или ошибок функционирования системы и способных привести к значительным нарушениям функционирования автоматизируемого объекта;
- в функционировании ИС участвуют различные виды ее обеспечения и персонал, которые могут в той или иной степени влиять на уровень ее надежности;
- в состав каждой ИС входит большое количество разнородных элементов: технических, программных, информационных, эргатических и др., при этом в выполнении одной функции ИС обычно участвует несколько различных элементов, а один и тот же элемент может участвовать в выполнении нескольких функций системы.

Перечень функций и видов их отказов, по которым задаются требования к надежности, а также критерии этих отказов прописываются в техническом задании на ИС. В выполнении функции, как правило, участвуют технические, программные средства, информационное обеспечение и персонал ИС, выделяемые в функциональную подсистему ИС. Поэтому при решении вопросов надежности ИС количественное описание, анализ, оценку и обеспечение проводят по каждой функциональной подсистеме ИС отдельно.

Уровень надежности ИС зависит от следующих основных факторов:

- состава и уровня надежности технических средств, их взаимосвязи в надежностной структуре комплекса технических средств ИС;
- состава и уровня надежности используемого программного обеспечения ИС;
- уровня квалификации персонала, организации работы и уровня надежности действий персонала ИС;
- режимов, параметров и организационных форм технической эксплуатации технических средств;
- степени использования различных видов резервирования (структурного, информационного, временного, алгоритмического, функционального);
- степени использования методов и средств технической диагностики;
- реальных условий функционирования ИС.

## **1.1. Состояния информационной системы**

Информационная система может находиться в состоянии: исправном, неисправном, работоспособном, неработоспособном, предельном.

*Исправное состояние* (исправность) – состояние объекта, при котором он соответствует всем требованиям нормативно-технической и/или проектно-конструкторской документации.

*Неисправное состояние* (неисправность) – состояние объекта, при котором он не соответствует хотя бы одному из требований нормативно-технической и/или проектно-конструкторской документации.

*Работоспособное состояние* (работоспособность) – состояние объекта, при котором значения всех параметров, характеризующих способ-

ность выполнять заданные функции, соответствуют требованиям нормативно-технической и/или проектно-конструкторской документации.

*Неработоспособное состояние* (неработоспособность) – состояние объекта, при котором значение хотя бы одного параметра, характеризующего способность выполнять заданные функции, не соответствует требованиям нормативно-технической и/или проектно-конструкторской документации.

*Предельное состояние* – состояние объекта, при котором его дальнейшее применение по назначению недопустимо или экономически нецелесообразно либо восстановление его исправного и работоспособного состояния технически невозможно или экономически нецелесообразно.

Каждое из рассмотренных состояний характеризуется совокупностью значений численных параметров или качественных признаков, для которых не применяют количественные оценки. Номенклатуру этих параметров и признаков, а также пределы их допустимых изменений устанавливают в нормативно-технической и/или проектно-конструкторской документации на объект.

Работоспособный объект в отличие от исправного должен удовлетворять лишь тем требованиям нормативно-технической и/или проектно-конструкторской документации, выполнение которых обеспечивает нормальное применение объекта по назначению.

Очевидно, что работоспособный объект может быть неисправным, например не удовлетворять эстетическим требованиям, если ухудшение внешнего вида не препятствует применению объекта по назначению.

Переход объекта из исправного состояния в неисправное и из работоспособного в неработоспособное обычно происходит вследствие повреждения или отказа.

Если объект переходит в неисправное, но работоспособное состояние, это событие называют *повреждением*. Если объект переходит в неработоспособное состояние, это событие называют *отказом*.

Переход объекта в предельное состояние влечет за собой временное или окончательное прекращение его применения по назначению.

Различают два вида предельного состояния. Первый совпадает с неработоспособным состоянием объекта. Второй обусловлен тем обстоятельством, что, начиная с некоторого момента времени, дальнейшее применение по назначению пока еще работоспособного объекта согласно опреде-



ленным критериям оказывается недопустимым в связи с опасностью, вредностью для здоровья людей или экономической нецелесообразностью этого использования. Переход объекта в предельное состояние второго вида происходит раньше, чем возникает отказ. Система может перейти в предельное состояние, оставаясь работоспособной, если ее дальнейшее применение по назначению станет недопустимым по требованиям безопасности, экономичности, эффективности и безвредности. Возможен и другой случай, при котором система, перешедшая в неработоспособное состояние, может не достигнуть предельного состояния, если восстановление работоспособного состояния целесообразно и/или допустимо.

## **1.2. События информационной системы**

*Отказ ИС* – событие, заключающееся в нарушении работоспособного состояния объекта (отказ функции). При классификации отказов ИС применимы следующие термины и определения (см. ГОСТ 27.002-83).

*Независимый отказ* – отказ системы, не обусловленный отказом другой системы (подсистемы), входящей в структурный состав данной.

Классификация по признаку зависимости осуществляется при наличии (отсутствии) зависимости отказов определенного элемента данной системы от отказов других элементов. В таких случаях рассматриваемой системой является элемент, отказы которого зависят (или не зависят) от отказов элементов этой же системы, рассматриваемых также в качестве самостоятельных систем (объектов) в экономических ИС (ЭИС).

*Внезапным отказом* является отказ, характеризующийся скачкообразным изменением значений одного или нескольких основных (заданных) параметров системы.

*Постепенный отказ* – отказ, характеризующийся постепенным изменением значений одного или нескольких основных (заданных) параметров системы.

Заметим, что деление отказов на внезапные и постепенные условно в том смысле, что при эксплуатации системы всякому скачкообразному изменению параметра предшествует процесс постепенного изменения каких-либо других физических величин.

Внезапному отказу не предшествует направленное изменение какого-нибудь из наблюдаемых эксплуатационных параметров системы, в связи с чем прогнозирование момента возникновения внезапного отказа практически невозможно. Понятие внезапного отказа относительно в том смысле, что при более глубоком проникновении в сущность процессов, связанных с возникновением отказа, может появиться возможность обнаружения таких постепенных изменений в системе или же ее элементах, которые закономерно предшествуют возникновению данного отказа, ранее относившегося к внезапным.

*Постепенный отказ* характеризуется наличием, по меньшей мере, тенденции или закономерности изменения заданного эксплуатационного параметра системы за время, предшествующее моменту возникновения отказа. Это обычно позволяет с заданной высокой вероятностью прогнозировать достаточно небольшой интервал времени, на котором следует ожидать возникновения постепенного отказа.

Следует отметить, что некоторые системы обладают способностью после некоторых отказов самовосстанавливать работоспособность. Отказы такого типа называют *самоустраняющимися* (или *сбоями*).

Отказ одного и того же характера может возникать в системе несколько раз через относительно короткие интервалы времени. До того как причина этих отказов будет обнаружена, возникает несколько однородных отказов, определяемых как перемежающийся отказ.

*Перемежающийся отказ* (сбой) характеризуется как многократно возникающий самоустраняющийся отказ системы одного и того же характера.

Если отказ возникает в результате несовершенства или нарушения установленных правил и/или норм конструирования (проектирования) системы, то такой отказ называют *конструкционным*.

Отказ, возникший в результате несовершенства или нарушения установленного процесса создания системы или ремонта ее технических средств, выполнявшегося ремонтным органом, относится к *производственному* отказу.

Если же отказ возникает в результате нарушения установленных правил и/или условий эксплуатации системы, он считается *эксплуатационным*.

Заметим, что отказы относят к конструкционным, производственным или эксплуатационным с целью установления, на каком этапе жизненного цикла системы следует провести мероприятия, предупреждающие и устраняющие причины отказов.

Применительно к отказу рассматривают критерий, причину и последствия отказа функции ИС.

Под *критерием* отказа понимается признак или совокупность признаков, установленных в нормативно-технической и/или проектно-конструкторской документации и позволяющих определить наличие отказа в выполнении некоторой функции ЭИС.

*Признаком* отказа в выполнении некоторой функции системы является недопустимое изменение значений показателей работоспособности, т.е. выход значений параметров за пределы допуска, нарушение признаков нормальной работы (своевременности и достоверности решения задач) и т.д.

*Причина* отказа в общем случае может определяться как совокупность явлений, процессов, событий и состояний, обусловивших возникновение отказа системы или какого-либо из ее элементов.

Причинами отказа могут быть дефекты, допущенные при проектировании, изготовлении и ремонте, нарушение правил и норм эксплуатации, различного рода повреждения, а также естественные процессы изнашивания и старения. Например, ошибочная установка оператором органов управления вычислительной системой может привести к несрабатыванию в нужный момент определенных устройств, т.е. произойдет эксплуатационный отказ этих устройств, не связанный с их повреждениями.

К *последствиям* отказа относятся явления, процессы, события и состояния, возникшие при отказе и находящиеся в непосредственной причинной связи с ним, например отказ ЭВМ и, как следствие, несоблюдение сроков решения задачи (комплекса задач).

Отказы в ИС целесообразно подразделять на аппаратные и программные.

*Аппаратным отказом* принято считать событие, при котором изделие утрачивает работоспособность и для его восстановления требуется проведение ремонта аппаратуры или замена отказавшего изделия на исправное.

*Программным отказом* считается событие, при котором объект утрачивает работоспособность по причине несовершенства программы (не-

совершенство алгоритма решения задачи, отсутствие программной защиты от сбоев, отсутствие программного контроля за состоянием изделия, ошибки в представлении программы на физическом носителе и т.д.). Характерным признаком программного отказа является то, что устраняется он путём исправления программы.

### **1.3. Примитивы надежности информационной системы**

Надежность ИС включает следующие примитивы: безотказность, ремонтпригодность, долговечность, сохранность.

Под *безотказностью* понимается свойство системы непрерывно сохранять работоспособное состояние в течение некоторого времени или некоторой наработки. Безотказность системы свойственна системе в любом из возможных режимов эксплуатации.

Под *ремонтпригодностью* понимается свойство системы, заключающееся в приспособленности к предупреждению и обнаружению причин возникновения отказов, поддержанию и восстановлению работоспособного состояния путем проведения технического обслуживания и ремонтов. Затраты времени определяются в заданных условиях выполнения операций технического обслуживания и ремонта, а также в части организации технологии ремонта, материально-технического обеспечения, квалификации персонала и т.д.

*Долговечность* можно определить как свойство системы сохранять работоспособное состояние до наступления предельного состояния при установленном режиме технического обслуживания и ремонта.

*Сохраняемость* – свойство системы непрерывно сохранять установленные значения показателей безотказности, долговечности и ремонтпригодности в течение и после хранения и/или транспортирования.

Сохраняемость системы характеризуется ее способностью противостоять отрицательному влиянию условий и продолжительности хранения и транспортирования на ее безотказность, ремонтпригодность и/или долговечность. Сохраняемость представляют в виде двух составляющих: одна из них проявляется во время хранения или транспортирования, другая – во время применения объекта после хранения и/или транспортирования.

Очевидно, что продолжительное хранение и транспортирование в необходимых условиях для многих объектов может отрицательно влиять на их поведение не только во время хранения или транспортирования, но и при последующем применении объекта, поскольку показатели сохранности могут оказаться значительно ниже, чем аналогичные показатели однотипных объектов, не находящихся на хранении и не подвергавшихся транспортированию.

#### 1.4. Типы ИС в аспекте надежности

При разработке ИС предусматривают выполнение или невыполнение технического обслуживания входящего в нее комплекса технических средств на протяжении всего срока их службы. Таким образом, все системы делят на *технически обслуживаемые* и *технически необслуживаемые*.

К *обслуживаемым* относятся такие системы, для которых проведение технического обслуживания предусмотрено в нормативно-технической и/или проектно-конструкторской документации, что, соответственно, не предусмотрено для необслуживаемых систем.

Переход системы из неработоспособного состояния в работоспособное осуществляется с помощью операций *восстановления* или *ремонта*. К операциям *восстановления* в основном относят такие, с помощью которых осуществляются операции идентификации отказа (определение его места и характера), замены, регулирования и контроля технического состояния элементов системы и заключительные операции контроля работоспособности ЭИС в целом, предусмотренные в нормативно-технической и/или проектно-конструкторской документации.

Таким образом, *восстанавливаемой системой* можно считать такую, для которой в рассматриваемой ситуации проведение восстановления работоспособного состояния предусмотрено в нормативно-технической и/или проектно-конструкторской документации. Соответственно, для *невосстанавливаемой системы* проведение восстановления работоспособного состояния по техническим, экономическим или другим соображениям не предусмотрено в нормативно-технической и/или проектно-конструкторской документации.

Переход системы из предельного состояния в работоспособное осуществляется с помощью ремонта, при котором происходит восстановление ее ресурса. Поэтому системы делятся на *ремонтируемые и неремонтируемые*.

К *ремонтируемым* относятся такие системы, для которых проведение ремонтов предусмотрено в нормативно-технической и/или проектно-конструкторской документации. Соответственно, *неремонтируемыми* являются системы, для которых проведение ремонтов не предусмотрено в нормативно-технической и/или проектно-конструкторской документации.

Такое деление систем связано с возможностью восстановления их ресурса путем ремонта, что полностью обуславливается конструкцией систем, т.е. предусматривается и обеспечивается при их разработке и вводе в эксплуатацию.

## **1.5. Оценка надежности информационной системы**

Оценку надежности ИС проводят на различных стадиях создания и эксплуатации. При разработке ИС проводят проектную (априорную) оценку надежности системы. При опытной и промышленной эксплуатации проводят экспериментальную (апостериорную) оценку надежности системы.

Оценку надежности системы производят, в общем случае, с учетом надежных свойств участвующих технических, программных средств, информационного обеспечения и персонала ИС. В предположении, что успешное функционирование каждого из технических, программных средств, информационного обеспечения и персонала ИС при выполнении *i*-й функции является независимым событием, *вероятностные показатели надежности информационной системы определяют как произведение соответствующих вероятностей этих независимых событий*.

В качестве показателей надежности ИС используют показатели, характеризующие:

- надежность реализации функций системы;
- опасность возникновения в системе аварийных ситуаций.

Описание надежности ИС по функциональным подсистемам осуществляют по отдельным составляющим надежности (единичным показате-

лям) и/или по нескольким составляющим надежности совместно (комплексным показателям надежности).

Основными *единичными показателями безотказности* являются:

- средняя наработка системы на отказ при выполнении  $i$ -й функции;
- вероятность безотказной работы системы при выполнении  $i$ -й функции.

Допускается использовать следующие показатели:

- средняя наработка системы до отказа при выполнении  $i$ -й функции;
- интенсивность отказов системы при выполнении  $i$ -й функции;
- параметр потока отказов системы при выполнении  $i$ -й функции.

Основными *единичными показателями ремонтпригодности* являются:

- среднее время восстановления способности системы к выполнению  $i$ -й функции;
- вероятность восстановления в течение заданного времени способности системы к выполнению  $i$ -й функции.

В ряде случаев показатели ремонтпригодности в нормативно-технической документации относятся к регламентируемым условиям ремонта.

*Комплексными показателями безотказности и ремонтпригодности* являются:

- коэффициент готовности системы к выполнению  $i$ -й функции;
- коэффициент технического использования системы по  $i$ -й функции;
- коэффициент сохранения эффективности системы по  $i$ -й функции.

К *показателям долговечности* можно отнести:

- средний ресурс (математическое ожидание ресурса);
- гамма-процентный ресурс (суммарная наработка, в течение которой объект не достигнет предельного состояния с вероятностью  $\gamma$ , выраженной в процентах);
- средний срок службы (математическое ожидание срока службы);
- гамма-процентный срок службы (календарная продолжительность эксплуатации, в течение которой устройство не достигнет предельного состояния с вероятностью  $\gamma$ , выраженной в процентах).

Здесь под ресурсом понимается суммарная наработка устройства от начала эксплуатации или ее возобновления после ремонта до перехода в предельное состояние, а под сроком службы – календарная продолжительность эксплуатации от начала эксплуатации устройства или ее возобновления после ремонта до перехода в предельное состояние.

К показателям *сохраняемости* можно отнести:

- средний срок сохраняемости (математическое ожидание срока сохраняемости);
- гамма-процентный срок сохраняемости (срок сохраняемости, достигаемый устройством с заданной вероятностью  $\gamma$ , выраженной в процентах).

Под сроком сохраняемости понимается календарная продолжительность хранения и/или транспортирования устройства, в течение которой сохраняются в заданных пределах значения параметров, характеризующих способность устройства выполнять заданные функции.

Описание надежности ИС по аварийным ситуациям осуществляют с помощью комплексных показателей надежности.

## **1.6. Обеспечение надежности информационной системы**

Необходимый уровень надежности ИС обеспечивают специальным комплексом работ, проводимых на всех стадиях создания и эксплуатации системы.

К обязательным работам по обеспечению надежности ИС, которые следует выполнять в процессе создания и эксплуатации ИС, относят:

- анализ состава и содержания функций разрабатываемой ИС, определение конкретного содержания понятия «отказ» и критериев отказа по каждому виду отказов для всех функций системы;
- выбор состава показателей надежности по всем функциям и аварийным ситуациям ИС, указанным в техническом задании (ТЗ) на ИС;
- анализ аварийных (исключительных) ситуаций в ИС, определение конкретного содержания понятия «аварийная ситуация» для данной ИС и критериев аварийной ситуации;
- выбор методов оценки надежности ИС на различных стадиях ее создания и функционирования;



- проведение проектной оценки надежности ИС при разработке технического проекта системы;
- определение режимов и параметров технической эксплуатации ИС.

Состав, содержание и последовательность выполнения работ по обеспечению надежности системы устанавливаются в программе обеспечения надежности ИС, которую составляют для каждой вновь разрабатываемой или модернизируемой ИС с учетом специфики системы и условий ее функционирования, важности выполняемых ею функций, требуемого уровня надежности, общего объема затрат на создание, а также особенностей ее создания.

Программу обеспечения надежности ИС составляют при разработке ТЗ на систему и оформляют отдельным документом как приложение к ТЗ. В ней указывают стадии создания ИС, на которых выполняются работы по обеспечению надежности системы, перечень этих работ, форму представления результатов работы, сроки выполнения и исполнителей.

Программу обеспечения надежности ИС составляет организация – головной исполнитель работ и согласует со всеми организациями, участвующими в реализации программы.

Для специальных ИС на всех стадиях создания и эксплуатации системы результаты реализации программы обеспечения надежности оформляют в виде отчета, предоставляемого заказчику. На основании отчета, при необходимости, в эту программу вносятся изменения.

## 2. ЭЛЕМЕНТЫ ТЕОРИИ ВЕРОЯТНОСТЕЙ И МАТЕМАТИЧЕСКОЙ СТАТИСТИКИ

Возникновение отказов в информационных системах зависит от разных факторов и носит случайный характер, что обуславливает применение в процессе исследования надежности систем методов теории вероятностей и математической статистики. Рассмотрим кратко их основные положения.

### 2.1. Случайные события

#### События и вероятность

Событие называется *случайным*, если при осуществлении определенной совокупности условий (испытаний) оно может произойти либо не произойти.

События называются *несовместными*, если появление одного из них исключает появление других событий при одних и тех же условиях, иначе события называются *совместными*.

Несколько событий образуют *полную группу*, если в результате испытаний появится хотя бы одно из них. Другими словами, появление хотя бы одного из событий полной группы есть достоверное событие. В частности, если события, образующие полную группу, попарно несовместны, то в результате испытания появится одно и только одно из этих событий.

*Противоположными* называют два единственно возможных события, образующих полную группу.

Два события называются *независимыми*, если появление одного из них не изменяет вероятности другого, иначе они называются *зависимыми*.

Несколько событий называют *попарно независимыми*, если каждые два из них независимы.

*Исходом* называют каждый из возможных результатов испытания.

*Вероятностью* (безусловной вероятностью) события  $A$  называют отношение числа благоприятствующих этому событию исходов к общему числу всех равновозможных несовместных исходов, образующих полную группу:

$$P(A) = m/n,$$

где  $n$  – общее число равновозможных элементарных исходов испытаний;  
 $m$  – число благоприятных исходов, в которых появляется событие  $A$ .

Очевидно, что вероятность *достоверного* события равна единице, а *невозможного* события – нулю.

### **Сумма событий**

*Суммой*  $A+B$  двух событий  $A$  и  $B$  называется событие, состоящее в появлении события  $A$ , или события  $B$ , или обоих этих событий.

В частности, если события  $A$  и  $B$  несовместные, то  $(A+B)$  – событие, состоящее в появлении одного из этих событий, безразлично какого.

*Суммой нескольких событий* называют событие, которое состоит в появлении хотя бы одного из этих событий.

Справедливы следующие теоремы.

**Теорема.** Вероятность появления одного из двух несовместных событий, безразлично какого, равна сумме вероятностей этих событий:

$$P(A+B) = P(A) + P(B)$$

**Теорема.** Сумма вероятностей событий  $A_1, A_2, \dots, A_n$ , образующих полную группу, равна единице:

$$P(A_1) + P(A_2) + \dots + P(A_n) = 1.$$

**Теорема.** Сумма вероятностей противоположных событий равна единице:

$$P(A) + P(\bar{A}) = 1.$$

### **Произведение событий и условная вероятность**

*Произведением* двух событий  $A$  и  $B$  называется событие  $AB$ , состоящее в совместном появлении этих событий.

*Произведением нескольких событий* называется событие, состоящее в совместном появлении всех этих событий.

*Условной вероятностью* события  $B$  при условии наступления события  $A$  называется вероятность события  $B$ , вычисленная в предположении, что событие  $A$  уже наступило:

$$P(B|A) = \frac{P(AB)}{P(A)}$$

**Теорема.** Вероятность совместного появления двух событий равна произведению вероятности одного из них и условной вероятности другого, вычисленной в предположении, что первое событие уже наступило:

~~$$P(A \cap B) = P(A)P(B|A)$$~~

**Теорема.** Вероятность совместного появления двух независимых событий равна произведению вероятностей этих событий:

~~$$P(A \cap B) = P(A)P(B)$$~~

**Теорема.** Вероятность появления хотя бы одного из двух совместных событий равна сумме вероятностей этих событий без вероятности их совместного появления:

~~$$P(A \cup B) = P(A) + P(B) - P(A \cap B)$$~~

### Формула полной вероятности

**Теорема.** Вероятность события  $A$ , которое может наступить лишь при условии появления одного из несовместных событий  $B_1, B_2, \dots, B_n$ , образующих полную группу, равна сумме произведений вероятностей каждого из этих событий и соответствующих условных вероятностей события  $A$ :

~~$$P(A) = P(B_1)P(A|B_1) + P(B_2)P(A|B_2) + \dots + P(B_n)P(A|B_n)$$~~

### Формулы Бейеса

Пусть событие  $A$  может наступить при условии появления одного из несовместных событий  $B_1, B_2, \dots, B_n$ , образующих полную группу.

Формулы Бейеса позволяют переоценить вероятность событий  $B_1, B_2, \dots, B_n$  после того, как становится известным результат испытания, в итоге которого появилось событие  $A$ :

$$P(B_i) = \frac{P(B_i)P(A|B_i)}{P(B_1)P(A|B_1) + P(B_2)P(A|B_2) + \dots + P(B_n)P(A|B_n)}, P(B_i) \neq 0, i = 1, 2, \dots, n.$$

### Схема Бернулли

Испытания называются *независимыми относительно события  $A$* , если вероятность события  $A$  в каждом испытании не зависит от исходов других испытаний.

Пусть производится  $n$  независимых испытаний, в каждом из которых вероятность появления события  $A$  постоянна и равна  $p$  и, соответственно, вероятность ненаступления события равна  $q = 1 - p$ .

Тогда вероятность того, что при  $n$  независимых испытаниях событие  $A$  осуществится ровно  $k$  раз и, следовательно, не осуществится  $(n - k)$  раз, будет равна

$$P_n(k) = \binom{n}{k} p^k q^{n-k}$$

Эта формула называется *формулой Бернулли*.

Если  $P_n(k)$  рассматривать как функцию произвольного допустимого аргумента  $k$ , то значения аргумента, при которых функция принимает максимальное значение, называются *точками наивероятнейшего аргумента*. При дробном значении  $np - q$  такой точкой будет одна –

$$m_1 = [np - q],$$

при целом значении  $np - q$  таких точек будет две:

$$m_1 = [np - q], m_2 = [np - q] + 1$$

Справедливы следующие формулы, позволяющие вычислить  $P_n(k)$  с заданной точностью.

**Теорема Муавра-Лапласа.** *Если вероятность  $p$  появления события  $A$  в каждом испытании постоянна и отлична от нуля и единицы, то вероятность  $P_n(k)$  того, что событие  $A$  появится в  $n$  испытаниях ровно  $k$  раз, приближенно равна (тем точнее, чем больше  $n$ )*

$$P_n(k) \approx \frac{1}{\sqrt{npq}} \varphi\left(\frac{k - np}{\sqrt{npq}}\right)$$

Функция  $\varphi(x)$ , равная

$$\varphi(x) = \frac{1}{\sqrt{2\pi}} e^{-x^2/2},$$

четна и табулирована. Следует иметь в виду, что приведенная формула дает хорошее приближение лишь при достаточно больших значениях  $n$ .

**Интегральная теорема Лапласа.** *Если вероятность  $p$  появления события  $A$  в каждом испытании постоянна и отлична от нуля и единицы, то вероятность  $P_n(k_1, k_2)$  того, что событие  $A$  появится в  $n$  испытаниях от  $k_1$  до  $k_2$  раз, приближенно равна*



где

$$x_1 = (k_1 - np) / \sqrt{npq}, \quad x_2 = (k_2 - np) / \sqrt{npq},$$

$$\Phi = \frac{1}{\sqrt{2\pi}} \int_0^x e^{-t^2/2} dt$$

Функция Лапласа  $\Phi(x)$  нечетна, табулирована.

**Теорема Пуассона.** Если вероятность  $p$  появления события  $A$  в каждом испытании мала и среднее число появлений события  $A$  остается неизменным при различных  $n$ , то вероятность  $P_n(k)$  того, что событие  $A$  появится при очень большом числе испытаний ровно  $k$  раз, стремится к

$$P_n(k) \rightarrow \frac{\lambda^k}{k!} e^{-\lambda}, \quad 0 < \lambda < \infty \text{ и } \delta \epsilon \quad p \rightarrow 0, \quad np \rightarrow \lambda.$$

## 2.2. Случайные величины

Случайной величиной  $X$  называется величина, значения которой зависят от случая.

Случайные величины могут быть непрерывными и дискретными. Случайная величина может быть задана плотностью распределения вероятностей либо законом распределения.

### Дискретная случайная величина

Дискретной называют случайную величину  $X$ , которая принимает конечное или счетное число значений  $\{x_1, x_2, \dots, x_n, \dots\}$  с определенной вероятностью  $\{p_1, p_2, \dots, p_n, \dots\}$ . Здесь  $p_i = P(X = x_i)$  – вероятность того, что случайная величина  $X$  равна  $x_i$ , причем

$$\sum_{k=1}^{\infty} p_k = 1.$$

Дискретная величина определяется законом распределения – совокупностью возможных значений случайной величины и соответствующих вероятностей.

*Примеры дискретных распределений:*

1) биномиальное распределение

$$x_k = k; p_k = P(X = k) = C_n^k p^k (1-p)^{n-k}, \quad 0 < p < 1, \quad k = 0, 1, 2, \dots, n.$$

Биномиальное распределение имеет, например, такая случайная величина, как число наступлений некоторого события в независимых испытаниях, имеющего неизменную вероятность наступления во всех испытаниях;

2) распределение Пуассона

$$x_k = k; p_k = P(X = k) = \frac{a^k}{k!} e^{-a}, \quad a > 0, \quad k = 0, 1, 2, \dots, n.$$

Эта формула выражает закон распределения массовых и маловероятных событий с постоянной вероятностью наступления в независимых испытаниях;

3) геометрическое распределение

$$x_k = k; p_k = P(X = k) = p(1-p)^{k-1}, \quad 0 < p < 1, \quad k = 0, 1, 2, \dots, n.$$

Геометрическим распределением описывается число испытаний, которые нужно провести до первого появления события в независимых испытаниях, имеющего неизменную вероятность наступления во всех испытаниях;

4) гипергеометрическое распределение

$$x_k = k; p_k = P(X = k) = \frac{C_m^k C_{n-m}^{r-k}}{C_n^r}, \quad 0 < p < 1, \quad k = 0, 1, 2, \dots, n.$$

Если из множества  $n$  элементов, имеющего  $m$  ( $m < n$ ) элементов стандартного типа, берется наугад  $r$  элементов и не возвращается, то число  $k$  элементов стандартного типа среди  $r$  отобранных будет иметь гипергеометрическое распределение.

## **Непрерывная случайная величина**

*Непрерывной* называют случайную величину  $X$ , которая может принимать все значения из некоторого конечного или бесконечного промежутка.

Непрерывная случайная величина  $X$  определяется *плотностью распределения*  $f(x)$  для каждого значения  $x$ , т.е. пределом отношения вероятности того, что случайная величина примет значение из интервала, к длине

этого интервала при устремлении ее к нулю, для любого значения из некоторого промежутка:

$$f(x) = \lim_{\Delta x \rightarrow 0} \frac{p(x < X < x + \Delta x)}{\Delta x}.$$

Плотность распределения обладает дополнительным свойством:

$$\int_{-\infty}^{+\infty} f(z) dz = 1$$

либо

$$\int_a^b f(z) dz = 1,$$

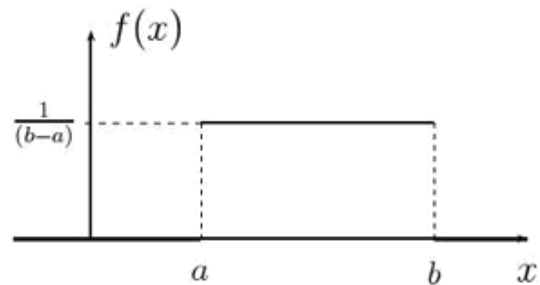
если все возможные значения случайной величины принадлежат интервалу  $(a, b)$ .

*Примеры непрерывных распределений:*

1) равномерное распределение (рис. 1):

$$f(x) = \begin{cases} \frac{1}{b-a}, & x \in [a, b], \\ 0, & x \notin [a, b], \end{cases}$$

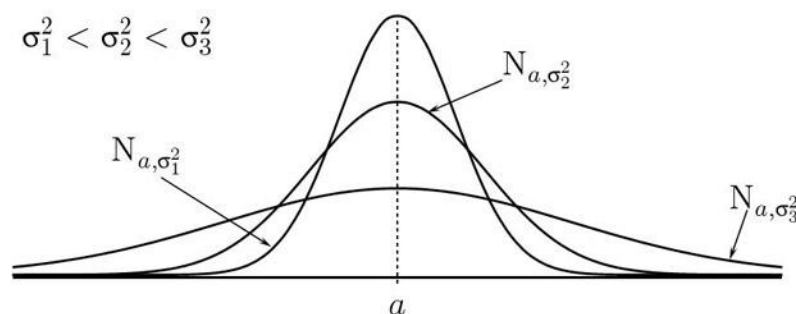
$-\infty < a < b < +\infty;$



**Рис. 1. График плотности равномерного распределения**

2) нормальное распределение (рис. 2):

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(x-a)^2}{2\sigma^2}\right), \quad x \in (-\infty, +\infty), \quad -\infty < a < +\infty, \quad \sigma > 0;$$



**Рис. 2. График плотности нормального распределения при различных  $\sigma$**



3) показательное (экспоненциальное) распределение (рис. 3):

$$f(x) = \begin{cases} \lambda e^{-\lambda x}, & x > 0, \lambda > 0, \\ 0, & x \leq 0. \end{cases}$$

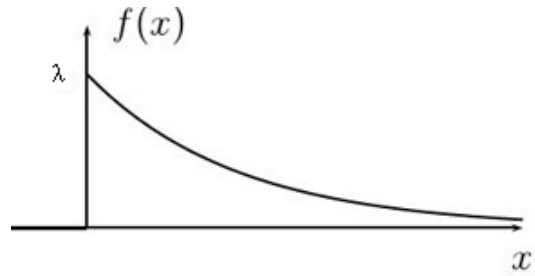


Рис. 3. График плотности показательного распределения

## 2.3. Числовые характеристики случайной величины

### Функция распределения

Функцией распределения случайной величины  $X$  называется функция  $F$  от произвольного вещественного числа  $x$ , определяемая как вероятность того, что случайная величина  $X$  примет значение меньше, чем  $x$ :

~~$$F(x) = P(X \leq x)$$~~

Некоторые свойства функции распределения:

1) значения функции распределения принадлежат отрезку  $[0, 1]$ :

$$0 \leq F(x) \leq 1;$$

2) функция распределения – неубывающая функция, т.е. если  $x_1 < x_2$ , то

$$F(x_1) \leq F(x_2);$$

3) вероятность того, что случайная величина примет значение, заключенное в интервале  $[a, b)$ , равна приращению функции распределения на этом интервале:

$$P(a \leq X < b) = F(b) - F(a).$$

Отсюда следует, что плотность распределения случайной величины равна

$$f(x) = \lim_{\Delta x \rightarrow 0} \frac{P(x < X < x + \Delta x)}{\Delta x} = \lim_{\Delta x \rightarrow 0} \frac{F(x + \Delta x) - F(x)}{\Delta x} = F'(x);$$

4) существуют пределы

$$\lim_{x \rightarrow -\infty} F(x) = 0, \quad \lim_{x \rightarrow +\infty} F(x) = 1;$$

5) для дискретной случайной величины функция распределения определяется по закону распределения (и наоборот):

$$F(x) = P(X < x) = \sum_{k: x_k < x} P(X = x_k);$$

6) для непрерывной случайной величины функция распределения определяется через плотность распределения вероятностей (и наоборот):

$$F(x) = P(X < x) = \int_{-\infty}^x f(z) dz;$$

7) для непрерывной случайной величины вероятность того, что случайная величина примет значение, ограниченное величинами  $a, b$ , равна

$$P(a < X < b) = P(a \leq X < b) = P(a < X \leq b) = P(a \leq X \leq b) = \int_a^b f(z) dz.$$

### Математическое ожидание

*Математическим ожиданием дискретной случайной величины  $X$  называется сумма абсолютно сходящегося ряда, составленного из произведений всех ее возможных значений и их вероятностей:*

$$MX = \sum_{k=1}^{\infty} x_k p_k.$$

*Математическим ожиданием непрерывной случайной величины  $X$  называется значение абсолютно сходящегося интеграла:*

$$MX = \int_{-\infty}^{+\infty} zf(z) dz.$$

*Свойства математического ожидания:*

1) математическое ожидание постоянной величины равно самой постоянной:

$$MC = C;$$

2) постоянный множитель можно выносить за знак математического ожидания:

$$M(CX) = C \cdot MX;$$

3) математическое ожидание произведения двух независимых случайных величин равно произведению их математических ожиданий:

$$M(XY) = MX \cdot MY;$$

4) математическое ожидание суммы двух случайных величин равно сумме их математических ожиданий:

$$M(X + Y) = MX + MY.$$

## Дисперсия

*Дисперсией (рассеиванием) случайной величины  $X$*  называют математическое ожидание квадрата отклонения случайной величины от ее математического ожидания:

$$DX = M(X - MX)^2.$$

*Среднеквадратичным отклонением случайной величины  $X$*  называют число

$$\sigma = \sqrt{DX}.$$

*Свойства дисперсии:*

1) дисперсия равна разности между математическим ожиданием квадрата случайной величины и квадратом ее математического ожидания:

$$DX = M(X^2) - (MX)^2;$$

2) дисперсия постоянной величины равна нулю:

$$DC = 0;$$

3) постоянный множитель можно выносить за знак дисперсии, возводя его в квадрат:

$$D(CX) = C^2 \cdot DX;$$

4) дисперсия суммы (разности) двух независимых случайных величин равна сумме их дисперсий:

$$D(X + Y) = DX + DY, \quad D(X - Y) = DX + DY;$$

5) дисперсия дискретной случайной величины равна

$$DX = \sum_{k=1}^{\infty} (x_k - MX)^2 P(X = x_k);$$

6) дисперсия непрерывной случайной величины равна

$$DX = \int_{-\infty}^{+\infty} (z - MX)^2 f(z) dz;$$

7) среднеквадратичное отклонение суммы конечного числа взаимно независимых случайных величин равно квадратному корню из суммы квадратов средних квадратичных отклонений этих величин:

$$\sigma(X_1 + X_2 + \dots + X_n) = \sqrt{\sigma^2(X_1) + \sigma^2(X_2) + \dots + \sigma^2(X_n)}.$$

## 2.4. Закон больших чисел

При очень большом количестве случайных явлений средний их результат практически перестает быть случайным и может быть предсказан с большой степенью определенности.

В узком смысле слова под законом больших чисел понимается ряд теорем, в каждой из которых для тех или иных условий устанавливается факт приближения средних характеристик большого числа испытаний к определенным неслучайным величинам.

**Неравенство Чебышева.** Для вероятности того, что отклонение случайной величины  $X$  от ее математического ожидания по абсолютной величине меньше положительного числа  $\varepsilon$ , справедливо неравенство

$$P(|X - MX| < \varepsilon) \geq 1 - \frac{DX}{\varepsilon^2}.$$

**Следствие из неравенства Чебышева (правило трех сигм).** Вероятность того, что случайная величина  $X$  с любым законом распределения находится в окрестности  $3\sigma$  от своего математического ожидания, не меньше  $8/9$ :



**Теорема Чебышева 1.** Среднее арифметическое наблюдаемых значений случайной величины  $X$  в  $n$  независимых испытаниях сходится по вероятности к ее математическому ожиданию:

$$\lim_{n \rightarrow \infty} P(|y_n - MX| < \varepsilon) = 1, \quad y_n = \frac{1}{n} \sum_{k=1}^n x_k.$$

**Теорема Чебышева 2.** Если  $X_1, X_2, \dots, X_n, \dots$  – попарно независимые случайные величины, причем дисперсии их не превышают постоянного числа  $C$ , то среднее арифметическое наблюдаемых значений случайных величин сходится по вероятности к среднему арифметическому их математических ожиданий:

$$\lim_{n \rightarrow \infty} P\left(\left|y_n - \frac{1}{n} \sum_{k=1}^n MX_k\right| < \varepsilon\right) = 1, \quad y_n = \frac{1}{n} \sum_{k=1}^n x_k.$$

**Теорема Бернулли.** При неограниченном возрастании числа опытов, в каждом из которых событие  $A$  появляется с одной и той же вероятностью  $p$ , частота события сходится по вероятности к его вероятности  $p$ :

$$\lim_{n \rightarrow \infty} P\left(\left|\frac{m}{n} - p\right| < \varepsilon\right) = 1.$$

**Теорема Пуассона.** При неограниченном возрастании числа опытов, в каждом из которых событие  $A$  появляется с разной вероятностью  $p_i$ , частота события сходится по вероятности к среднему арифметическому его вероятностей:

$$\lim_{n \rightarrow \infty} P\left(\left|\frac{m}{n} - \frac{1}{n} \sum_{k=1}^n p_i\right| < \varepsilon\right) = 1.$$

**Теорема Маркова.** Если  $X_1, X_2, \dots, X_n$  – случайные величины с математическими ожиданиями  $MX_1, MX_2, \dots, MX_n$  и дисперсиями  $DX_1, DX_2, \dots, DX_n$  такими, что  $\frac{1}{n^2} D\left(\frac{1}{n} \sum_{k=1}^n X_k\right) \rightarrow 0$  и  $n \rightarrow \infty$ , то их среднее арифметическое сходится по вероятности к среднему арифметическому их математических ожиданий:

$$\lim_{n \rightarrow \infty} P\left(\left|\frac{1}{n} \sum_{k=1}^n X_k - \frac{1}{n} \sum_{k=1}^n MX_k\right| < \varepsilon\right) = 1.$$

**Центральная предельная теорема.** Если  $X_1, X_2, \dots, X_n, \dots$  – попарно независимые случайные величины, имеющие одно и то же распределение с  $MX_i = a$ ,  $DX_i = \sigma^2$  для  $i = 1, 2, \dots, n$ , то при возрастании  $n$  закон распределения суммы случайных величин  $X = \sum_{k=1}^n X_k$  неограниченно приближается к нормальному.

## 2.5. Статистические характеристики случайной величины

В результате наблюдения над случайными явлениями выявляются случайные величины, характеризующие эти явления.

*Генеральной* совокупностью случайной величины называют множество всех значений, которые может принять случайная величина.

*Выборочной* совокупностью (выборкой) случайной величины называют совокупность независимых наблюдений этой величины. Количество значений выборки называется *объемом* выборки.

Наблюдаемые значения случайной величины могут называться *вариантами*, а последовательность вариантов, записанная в возрастающем порядке, – *вариационным рядом*.

Отношение числа повторов значений случайной величины в выборке к объему выборки дает *частоту* повторяемости значений случайной величины.

*Статистическим распределением* (статистическим законом распределения) случайной величины называют перечень ее наблюдаемых значений и их частот.

## Статистические параметры распределения

*Частота* случайной величины  $X$  (*частота встречаемости* значения  $x_i$ ) определяется как отношение числа повторов значения  $x_i$  случайной величины в выборке к объему выборки для каждого значения.

*Статистической функцией распределения* случайной величины  $X$  называется функция, определяющая для каждого ее наблюдаемого значения  $x_i$  частоту события  $X < x_i$ .

*Выборочной средней* называют среднее арифметическое значение выборочных значений случайной величины:

$$\bar{x} = \frac{1}{n} \sum_{k=1}^n x_k.$$

*Выборочной улучшенной дисперсией* называют

$$(s^*)^2 = \frac{1}{n-1} \sum_{k=1}^n (x_k - \bar{x})^2.$$

Очевидно, что в случае повторяемости значений

$$\bar{x} = \frac{1}{n} \sum_{k=1}^n n_k x_k, \quad (s^*)^2 = \frac{1}{n-1} \sum_{k=1}^n n_k (x_k - \bar{x})^2,$$

где  $n_k$  – частота.

## Типы оценок параметров распределения

Оценкой неизвестного параметра  $\theta$  распределения случайной величины  $X$  называется произвольная функция  $\theta^*$ , зависящая от наблюдаемых значений случайной величины:

$$\theta^* = \theta^*(x_1, x_2, \dots, x_n) \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}.$$

Оценка  $\theta^*$  параметра  $\theta$  называется *несмещенной*, если  $M\theta^* = \theta$ , иначе *смещенной*.

Оценка  $\theta^*$  параметра  $\theta$  называется *состоятельной*, если

$$P(|\theta^* - \theta| < \varepsilon) \rightarrow 1 \text{ ĩ ðè } n \rightarrow \infty,$$

т.е. состоятельная оценка сходится по вероятности к параметру при увеличении числа наблюдаемых значений.

Оценка  $\theta^*_1$  параметра  $\theta$  называется *лучшей*, чем  $\theta^*_2$ , если

$$M(\theta^*_1 - \theta)^2 < M(\theta^*_2 - \theta)^2,$$

т.е. разброс значений оценки  $\theta^*_1$  вокруг параметра  $\theta$  меньше, чем разброс значений оценки  $\theta^*_2$  вокруг параметра  $\theta$ .

Минимальная оценка среди лучших называется *эффективной*.

Справедливы следующие утверждения.

**Лемма 1.** Частота случайной величины  $X$  является несмещенной состоятельной оценкой вероятности случайной величины  $X$ .

**Лемма 2.** Выборочная средняя случайной величины  $X$  является несмещенной состоятельной оценкой математического ожидания случайной величины.

**Лемма 3.** Выборочная улучшенная дисперсия случайной величины  $X$  является несмещенной состоятельной оценкой дисперсии случайной величины.

## 2.6. Погрешность оценки параметров распределения

### Погрешность и точность оценки

Абсолютной погрешностью (точностью) оценки  $\theta^*$  параметра  $\theta$  случайной величины называется абсолютная разность  $|\theta^* - \theta|$ .

Вероятность того, что абсолютная погрешность не превзойдет некоторого малого положительного  $\varepsilon$ :

$$P(|\theta^* - \theta| < \varepsilon),$$

называется *надежностью* оценки  $\theta^*$ .

Следующие леммы дают возможность определить надежность выборочного среднего, выборочной улучшенной дисперсии и частоты как оценок соответственно математического ожидания, дисперсии и вероятности наблюдения значения из выборочной совокупности случайной величины.

**Лемма 1.** Надежность выборочного среднего случайной величины  $X$  равна

$$P(|\bar{x} - MX| < \varepsilon) = 2\Phi\left(\frac{\varepsilon}{\sqrt{\frac{(s^*)^2}{n}}}\right).$$

**Лемма 2.** Надежность выборочной улучшенной дисперсии случайной величины  $X$  равна

$$P\left(\frac{(s^*)^2}{n} - D \leq \varepsilon\right) = 2\Phi\left(\frac{\varepsilon}{\sqrt{\frac{2(s^*)^2}{n}}}\right).$$

**Лемма 3.** Надежность частоты  $m$  принятия случайной величиной  $X$  некоторого значения из выборки объемом  $n$  равна

$$P\left(\frac{m}{n} - p < \varepsilon\right) = 2\Phi\left(\frac{\varepsilon n}{\sqrt{\frac{m}{n}\left(1 - \frac{m}{n}\right)}}\right).$$



## Доверительные интервалы

*Доверительным интервалом* оцениваемого параметра  $\theta$  случайной величины называется интервал  $(a,b)$  такой, что вероятность попадания  $\theta$  в этот интервал равно числу  $\rho$ , называемому *доверительным уровнем* (коэффициентом доверия). Границы интервала  $a$  и  $b$  называются *доверительными пределами*.

Очевидно, что  $0 < \rho < 1$ . Чем ближе  $\rho$  к 1, тем выше уровень доверия к доверительному интервалу.

Значения доверительного уровня и доверительного интервала для математического ожидания, дисперсии и вероятности наблюдения значения из выборочной совокупности случайной величины легко получаются из вышеприведенных лемм.

## 2.7. Проверка статистических гипотез

### Статистические гипотеза и критерий

*Статистической гипотезой* называют любое утверждение о виде или свойствах распределения наблюдаемых случайных величин.

Процедура обоснованного сопоставления высказанной гипотезы с имеющимися выборочными данными, сопровождаемая количественной оценкой степени достоверности получаемого вывода, осуществляется с помощью того или иного статистического критерия и называется *статистической проверкой гипотезы*.

Правило, согласно которому проверяемая гипотеза принимается или отвергается, называется *статистическим критерием* или критерием проверки гипотезы.

При принятии гипотезы возможны ошибки первого и второго рода.

*Ошибка первого рода*: в некой малой доле случаев гипотеза не подтверждается, хотя на самом деле она верна.

*Ошибка второго рода*: в небольшой доле случаев гипотеза подтверждается, хотя на самом деле она ошибочна.

Уровень значимости  $\alpha$  критерия – это вероятность ошибочного отклонения проверяемой гипотезы. Чем меньше уровень значимости, тем лучше. Уровень значимости связан с уровнем доверия как  $\alpha = 1 - \rho$ .

Мощность критерия равна  $1 - \beta$ , где  $\beta$  – вероятность ошибочного принятия гипотезы.

### Критическая статистика

Критической статистикой называется функция, зависящая от результатов наблюдения:

$$\gamma_n = \gamma_n(x_1, x_2, \dots, x_n)$$

и выбираемая исходя из предположения справедливости выдвинутой гипотезы.

Функция  $\gamma_n$  является случайной величиной как функция случайных величин, для которой задан закон распределения.

В зависимости от уровня значимости  $\alpha$  функция  $\gamma_n$  имеет одну или две критические точки, разделяющие всю область ее значений на две или три части в предположении справедливости гипотезы – области правдоподобных и неправдоподобных значений функции  $\gamma_n$  (рис. 4).

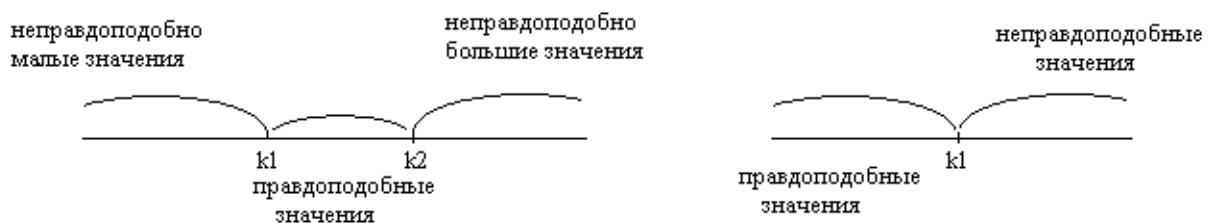


Рис. 4. Области правдоподобных и неправдоподобных значений функции

Гипотеза принимается, если на наблюдаемых значениях  $x_i$  функция  $\gamma_n$  принимает значения из области правдоподобных значений.

Если задача проверки статистической гипотезы такова, что выдвигается одна гипотеза и проверяется согласованность наблюдаемых данных с данными теоретического распределения гипотезы, то статистические критерии называются *критериями согласия*. На практике чаще всего применяются критерии согласия Колмогорова, Пирсона и др.

## Статистические критерии

### **Критерий Колмогорова для непрерывной случайной величины.**

*Выдвигаемая гипотеза принимается, если соответствующая ей статистическая функция  $q^*(X)$  такова, что критическая статистика Колмогорова*

$$\gamma_n = \sqrt{n} \max_X |q^*(X) - q(X)|$$

*с теоретической функцией  $q(X)$  для заданного уровня значимости  $\alpha$  на наблюдаемых значениях случайной величины  $X$  принимает значения из правдоподобной области, иначе гипотеза отвергается.*

**Критерий Пирсона для непрерывной или дискретной случайной величины.** *Выдвигаемая гипотеза принимается, если критическая статистика Пирсона*

$$\chi^2 = \sum_{i=1}^r \frac{(v_i - np_i)^2}{np_i}$$

*не превосходит критической точки для заданного уровня значимости  $\alpha$  на наблюдаемых значениях случайной величины  $X$ . Здесь  $v_i$  – количество наблюдаемых значений в  $r$ -интервалах;  $p_i$  – теоретические вероятности выбранного распределения;  $r$  – число интервалов;  $n$  – число наблюдений;  $s$  – число степеней свободы, равное  $r - 1 - \text{par}$ ;  $\text{par}$  – число параметров распределения, вычисляемых по выборке.*

### 3. АНАЛИЗ НАДЕЖНОСТИ ТЕХНИЧЕСКИХ СРЕДСТВ ИНФОРМАЦИОННОЙ СИСТЕМЫ

#### 3.1. Показатели надежности технических средств

Основными техническими средствами информационных систем являются:

- ЭВМ (состоящая из системного блока, монитора, клавиатуры и мыши);
- периферийные устройства (принтеры, сканеры);
- сетевая инфраструктура (пассивное оборудование – кабельная система, активное оборудование – модемы, коммутаторы, маршрутизаторы);
- система электропитания;
- система охлаждения;
- система вентиляции;
- специальное оборудование и т.п.

Некоторые из указанных устройств в случае выхода из строя не ремонтируются, а заменяются на исправные, т.е. являются невосстанавливаемыми. Другие устройства частично ремонтируются, например, замена неисправной материнской платы в системном блоке ЭВМ приводит к восстановлению ЭВМ. В целом, информационная система относится к восстанавливаемым системам.

Рассмотрим с точки зрения надежности невосстанавливаемые устройства информационных систем. Надежность восстанавливаемых устройств будет рассмотрена во второй части пособия.

Пусть  $T$  – это наработка (продолжительность работы или время безотказной работы) технического средства до отказа, т.е. интервал времени от начала работы до первого отказа. Тогда к основным показателям надежности технических устройств ИС можно отнести:

- вероятность безотказной работы;
- вероятность отказа;
- плотность распределения времени безотказной работы (частоту отказов);
- интенсивность отказов;

- среднюю наработку до первого отказа;
- дисперсию времени безотказной работы.

Указанные показатели надежности позволяют достаточно полно оценить надежность невосстанавливаемых технических средств, а также надежность восстанавливаемых технических средств до первого отказа.

Наиболее полно надежность ТС характеризуется частотой отказов. Это объясняется тем, что частота отказов является плотностью распределения, а поэтому содержит в себе всю информацию о случайном явлении – времени безотказной работы.

Средняя наработка до первого отказа является достаточно наглядной характеристикой надежности, однако применение этого показателя ограничено в тех случаях, когда интенсивность отказов непостоянна или время работы отдельных частей ТС разное.

Интенсивность отказов – наиболее удобная характеристика надежности, так как она позволяет вычислить остальные показатели.

Наиболее практичным показателем надежности ТС является вероятность безотказной работы, поскольку может быть сравнительно просто оценена в процессе эксплуатации.

**Вероятность безотказной работы** технического средства – это вероятность того, что при заданных условиях эксплуатации в течение интервала времени  $(0, t)$  не возникнет ни одного отказа, т.е. средство будет работоспособно:

$$P(t) = P(T \geq t).$$

**Вероятность отказа** технического средства – это вероятность того, что при заданных условиях эксплуатации в течение интервала времени  $(0, t)$  произойдет хотя бы один отказ, т.е. вероятность того, что время безотказной работы  $T$  меньше произвольного  $t$ :

$$Q(t) = P(T < t) = 1 - P(t).$$

**Плотность распределения времени безотказной работы (частота отказов)** технического средства  $f(t)$  для каждого значения  $t$  – предел отношения вероятности того, что время безотказной работы примет значение из интервала  $(t, t + \Delta t)$ , к длине этого интервала при устремлении ее к нулю для любого значения из некоторого промежутка  $(0, t)$ :

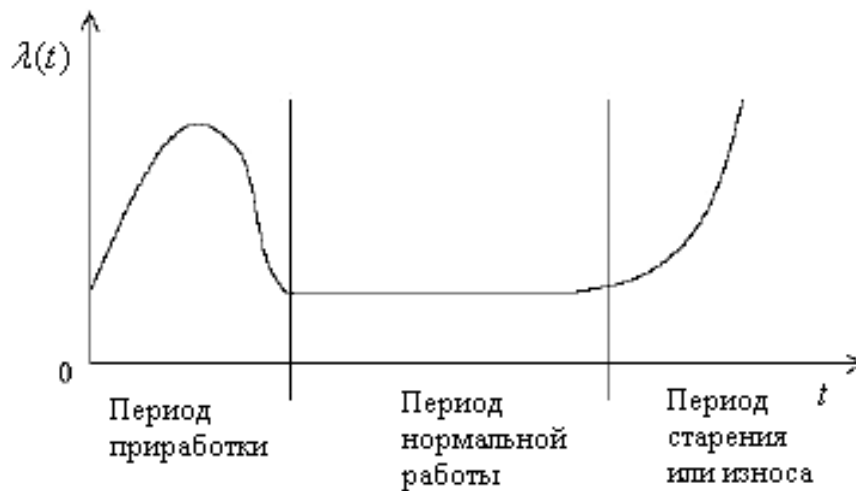
$$f(t) = \lim_{\Delta t \rightarrow 0} \frac{P(t < T < t + \Delta t)}{\Delta t} = Q'(t) = -P'(t).$$

**Интенсивность отказов** представляет собой предел отношения вероятности отказа на интервале  $(t, t + \Delta t)$  к длине этого интервала при устремлении  $\Delta t$  к нулю:

$$\lambda(t) = \lim_{\Delta t \rightarrow 0} \frac{Q(t, t + \Delta t)}{\Delta t} = -\frac{P'(t)}{P(t)} = \frac{f(t)}{P(t)}.$$

Размерность интенсивности отказов – 1/ч.

Для многих важных случаев интенсивность отказов имеет вид, приведенный на рис. 5.



**Рис. 5. Функция интенсивности отказов**

**Средняя наработка до первого отказа** технического средства есть математическое ожидание времени его безотказной работы  $T$ :

$$T_{ср} = MT.$$

Средняя наработка до первого отказа измеряется в часах.

**Дисперсия  $DT$  времени безотказной работы  $T$**  технического средства характеризует степень разброса значений времени его безотказной работы  $T$  относительно средней наработки; измеряется в часах.

### **Взаимосвязь показателей надежности**

Через плотность распределения времени безотказной работы и интенсивность отказов могут быть выражены остальные показатели надеж-

ности технических средств. Такая зависимость, в частности, позволяет вычислить основные показатели надежности по плотности распределения времени безотказной работы, которую сравнительно просто получить по накопленным наблюдениям за функционированием информационной системы.

$$P(t) = \exp\left(-\int_0^t \lambda(u) du\right) = 1 - \int_0^t f(u) du,$$

$$Q(t) = 1 - P(t),$$

$$\lambda(t) = -\frac{P'(t)}{P(t)} = \frac{f(t)}{P(t)} = \frac{f(t)}{1 - \int_0^t f(u) du},$$

$$f(t) = Q'(t) = -P'(t) = \lambda(t) \cdot \exp\left(-\int_0^t \lambda(u) du\right),$$

$$T_{cp} = \int_0^{\infty} t \cdot f(t) dt = \int_0^{\infty} P(t) dt,$$

$$DT = 2 \int_0^{\infty} t^2 \cdot f(t) dt - T_{cp}^2 = 2 \int_0^{\infty} t \cdot P(t) dt - T_{cp}^2.$$

### Статистические оценки показателей надежности

По статистическим данным об отказах показатели надежности технических средств могут быть оценены следующими выражениями:

$$P^*(t) = \frac{N_0 - n(t)}{N_0},$$

$$\lambda^*(t) = \frac{n(\Delta t)}{N_{cp} \cdot \Delta t}, \quad N_{cp} = \frac{N_1 + N_2}{2},$$

$$Q^*(t) = \frac{n(t)}{N},$$

$$T_{cp}^* = \frac{1}{N} \left( \sum_{i=1}^N t_i \right),$$

$$f^*(t) = \frac{n(\Delta t)}{N \cdot \Delta t},$$

$$D^*T = \frac{1}{N-1} \sum_{i=1}^N (t_i - T_{cp}^*)^2,$$

где  $N$  – число ТС в начале испытаний;  $n(t)$  – число отказавших ТС за время  $(0, t)$ ;  $n(\Delta t)$  – число отказавших ТС за время  $(t - \Delta t/2, t + \Delta t/2)$ ;  $t_i$  – время безотказной работы  $i$ -го ТС;  $N_1$  – число исправно работающих ТС в начале интервала  $\Delta t$ ;  $N_2$  – число исправно работающих ТС в конце интервала  $\Delta t$ .

При большом количестве устройств статистические оценки практически совпадают с теоретическими значениями.

## 3.2. Модели надежности технических средств

Рассмотрим типичные распределения случайной величины  $T$  – времени безотказной работы технического средства до первого отказа.

### 3.2.1. Экспоненциальный закон надёжности

При экспоненциальном законе распределения вероятности времени безотказной работы  $T$  интенсивность отказов является постоянной, т.е.

$$\lambda(t) = \lambda = \text{const.}$$

Ниже приведены формулы, по которым определяются количественные характеристики экспоненциального закона надёжности. Графики плотности распределения времени безотказной работы  $f(t)$  и вероятности безотказной работы  $P(t)$  представлены соответственно на рис. 6, 7.

$$f(t) = \begin{cases} \lambda \cdot \exp(-\lambda t), & t > 0; \\ 0, & t \leq 0; \end{cases}$$

$$P(t) = \exp(-\lambda t);$$

$$Q(t) = 1 - \exp(-\lambda t);$$

$$T_{cp} = \frac{1}{\lambda};$$

$$DT = \frac{1}{\lambda^2}.$$

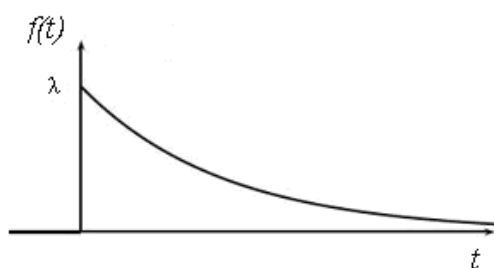


Рис. 6. Функция  $f(t)$

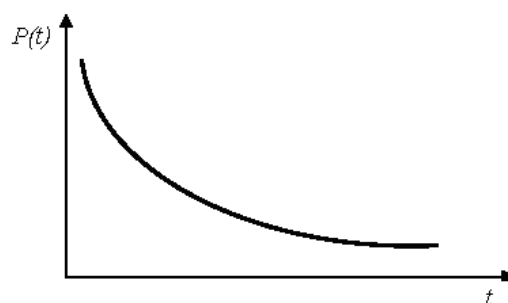


Рис. 7. Функция  $P(t)$

Экспоненциальный закон надёжности справедлив, когда отказы ТС независимы и носят внезапный характер. Для большинства ТС данный закон проявляется при установившемся режиме эксплуатации (период нормальной работы).

Для экспоненциального закона вероятность безотказной работы на любом интервале времени  $\Delta t$  не зависит от прошедшего времени, а зависит только от  $\Delta t$ .



### 3.2.2. Усеченный нормальный закон надёжности

При нормальном распределении случайной величины она может принимать значения от  $-\infty$  до  $+\infty$ . Поскольку время безотказной работы устройства может быть только положительным, ее распределение может быть только усеченным.

Усеченный нормальный закон надёжности характеризует вероятность отказа при длительном и постепенном изменении характеристик технического средства (старение, износ).

Ниже приведены формулы, по которым определяются количественные характеристики усеченного нормального закона надёжности. Графики плотности распределения времени безотказной работы и интенсивности отказов представлены на рис. 8.

$$f(t) = \begin{cases} \frac{c}{\sigma\sqrt{2\pi}} \cdot \exp\left(-\frac{(t-m)^2}{2 \cdot \sigma^2}\right), & t > 0; \\ 0, & t \leq 0; \end{cases}$$

$$P(t) = c \cdot (0,5 - \hat{O}(u));$$

$$Q(t) = c \cdot (0,5 + \hat{O}(u));$$

$$\lambda(t) = \frac{1}{\sigma} \cdot \frac{\phi(u)}{0,5 - \hat{O}(u)};$$

$$T_{cp} = m + \sigma \cdot t \cdot k_0;$$

$$DT = \sigma^2 \left(1 - k^2 - k \frac{m}{\sigma}\right).$$

Здесь  $\hat{O}(u) = \frac{1}{\sqrt{2\pi}} \int_0^u \exp(-x^2/2) dx$  – нормированная функция Лапласа;

$$\phi(u) = \frac{1}{\sqrt{2\pi}} \exp(-u^2/2), \quad u = \frac{t-m}{\sigma}, \quad c = \frac{1}{0,5 + \hat{O}\left(\frac{m}{\sigma}\right)}, \quad k = \frac{c}{\sqrt{2\pi}} \exp\left(-\frac{1}{2}\left(\frac{m}{\sigma}\right)^2\right).$$

Показатели  $m$ ,  $\sigma$  являются параметрами усеченного нормального распределения.

На практике усеченный нормальный закон почти совпадает с нормальным законом, поскольку  $\frac{m}{\sigma} > 2$ .

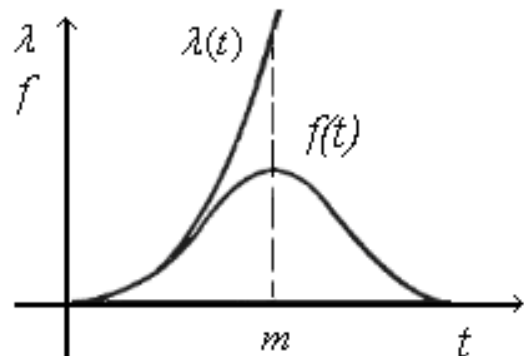


Рис. 8. Функции  $f(t)$ ,  $\lambda(t)$

Для данного закона надежности справедливы следующие утверждения.

**Лемма 1.** Вероятность безотказной работы устройства на интервале  $(t_1, t_2)$  равна

$$P(t_1 \leq T \leq t_2) = \hat{O}\left(\frac{t_2 - T_{cp}}{\sqrt{DT}}\right) - \hat{O}\left(\frac{t_1 - T_{cp}}{\sqrt{DT}}\right).$$

**Лемма 2.** Вероятность безотказной работы устройства на интервале  $(t, t + \Delta t)$  при условии, что на интервале  $(0, t)$  устройство работало исправно, равна

$$P(t, t + \Delta t) = \frac{0,5 - \hat{O}\left(\frac{t + \Delta t - T_{cp}}{\sqrt{DT}}\right)}{0,5 - \hat{O}\left(\frac{t - T_{cp}}{\sqrt{DT}}\right)}.$$

### 3.2.3. Закон распределения Релея

Распределение Релея имеет место при отказах системы, возникающих в результате износа и старения элементов, а также при отказах системы, состоящей из последовательно соединенных повторяющихся устройств.

Ниже приведены формулы, по которым определяются количественные характеристики надёжности распределения Релея времени безотказной работы. График вероятности безотказной работы представлен на рис. 9, графики плотности распределения времени безотказной работы и интенсивности отказов – на рис. 10.

$$f(t) = \begin{cases} \frac{1}{MODT^2} \cdot \exp\left(-\frac{t^2}{2 \cdot MODT^2}\right), & t > 0; \\ 0, & t \leq 0; \end{cases}$$

$$P(t) = \exp\left(\frac{-t^2}{2 \cdot MODT^2}\right);$$

$$Q(t) = 1 - P(t);$$

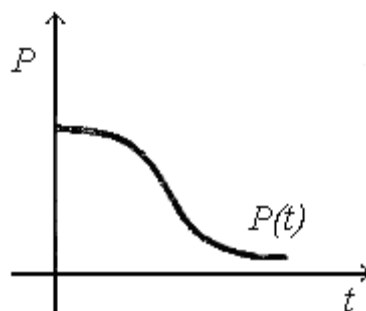


Рис. 9. Функции  $P(t)$

$$\lambda(t) = \frac{t}{MODT^2};$$

$$T_{cp} = MODT \sqrt{\frac{\pi}{2}};$$

$$DT = \left(2 - \frac{\pi}{2}\right) \cdot MODT^2.$$

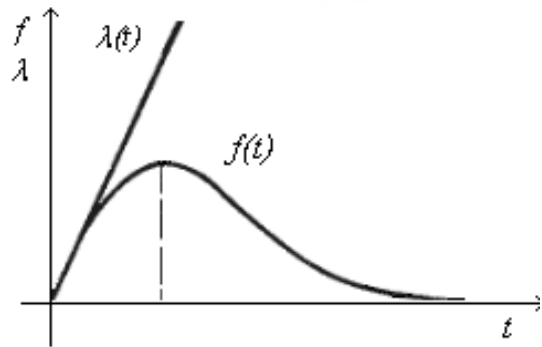


Рис. 10. Функции  $f(t)$ ,  $\lambda(t)$

Показатель  $MODT$  является параметром (модой) данного распределения.

### 3.2.4. Закон распределения Вейбулла

Закон распределения Вейбулла также имеет место при отказах системы, возникающих в результате износа и старения элементов, и при отказах системы, состоящей из последовательно соединенных повторяющихся устройств.

Ниже приведены формулы, по которым определяются количественные характеристики надёжности распределения Вейбулла времени безотказной работы. Графики вероятности безотказной работы и плотности распределения времени безотказной работы представлены соответственно на рис. 11, 12.

$$f(t) = \begin{cases} a \cdot k \cdot t^{k-1} \cdot \exp(-at^k), & t > 0; \\ 0, & t \leq 0; \end{cases}$$

$$P(t) = \exp(-at^k);$$

$$Q(t) = 1 - P(t);$$

$$\lambda(t) = a \cdot k \cdot t^{k-1};$$

$$T_{cp} = \frac{\tilde{A} \left( \frac{1}{k} + 1 \right)}{a^{\frac{1}{k}}};$$

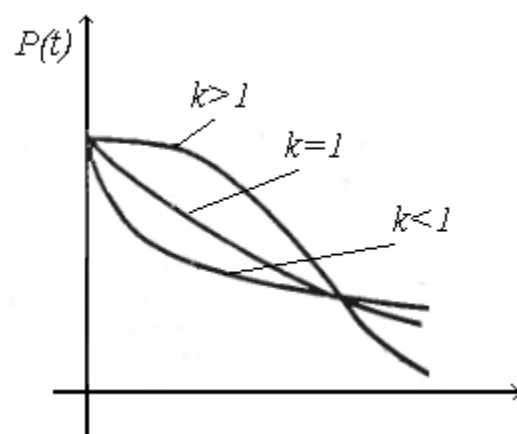


Рис. 11. Функция  $P(t)$

$$DT = \frac{1}{a^k} \left\{ \tilde{A} \left( \frac{2}{k} + 1 \right) - \left[ \frac{1}{k} \tilde{A} \left( \frac{1}{k} \right) \right]^2 \right\}.$$

Здесь  $\tilde{A} \left( \frac{1}{k} + 1 \right) = \int_0^{\infty} e^{-at} \cdot t^{\frac{1}{k}} dt$  – гамма-функция;  $a, k$  – параметры распределения Вейбулла.

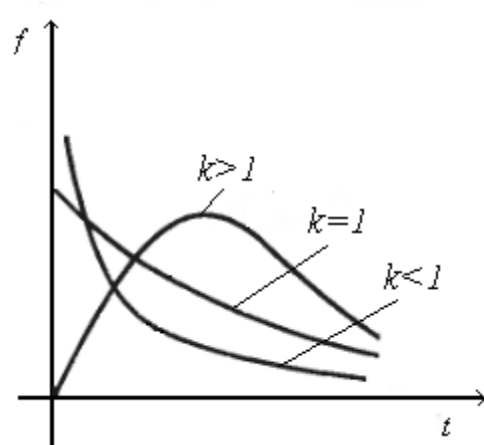


Рис. 12. Функция  $f(t)$

При  $a < 1$  интенсивность отказов будет монотонно убывающей функцией, при  $a > 1$  – монотонно возрастающей. Это обстоятельство дает возможность подбирать для опытных данных наиболее подходящие значения параметров  $a$  и  $k$ , с тем чтобы функция распределения наилучшим образом совпадала со значениями опытных данных.

При  $a = \lambda, k = 1$  закон Вейбулла совпадает с экспоненциальным законом.

### 3.2.5. Гамма-распределение

Гамма-распределение ( $\gamma$ -распределение) – это распределение случайной величины с вероятностью  $\gamma$ , выраженной в процентах. Гамма-распределению подчиняется время безотказной работы системы резервированных устройств.

Если отказ системы возникает тогда, когда произойдет не менее  $k$  отказов ее устройств, а отказы устройств подчинены экспоненциальному закону с параметром  $\lambda_0$ , то плотность вероятности отказа системы за время  $t$  равна

$$f(t) = \frac{\lambda_0^k \cdot t^{k-1}}{\tilde{A}(k)} \exp(-\lambda_0 t),$$

где  $\tilde{A}(k) = \int_0^{\infty} e^{-u} \cdot u^{k-1} du$  – гамма-функция;  $\lambda_0$  – исходная интенсивность отказов устройств системы, отказ которой вызывается отказом  $k$  ее элементов.

Вероятность не менее  $k$  отказов устройств системы равна

$$F(t) = 1 - \sum_{i=0}^{k-1} \frac{(\lambda_0 t)^i}{i!} \exp(-\lambda_0 t).$$

При  $k = 1$   $\gamma$ -распределение совпадает с экспоненциальным распределением. При больших значениях  $k$  гамма-распределение сходится к нормальному распределению.

Гамма-распределение иногда называют распределением Эрланга.

### 3.2.6. Закон Пуассона

Закон Пуассона принимается, когда отказы системы (или устройства) независимы между собой.

Вероятность возникновения  $k$  отказов за время  $t$  равна

$$P(t) = \frac{(\lambda t)^k}{k!} \exp(-\lambda t),$$

где  $\lambda$  – интенсивность отказов, причем постоянная.

Закон Пуассона используется тогда, когда необходимо определить вероятность того, что в системе за данное время произойдет один, два, три и т.д. отказов.

### 3.2.7. Экспериментальная оценка надежности

Применение описанных выше типичных законов распределения времени безотказной работы технических устройств информационной системы сопряжено с определенными трудностями:

во-первых, необходимо знание параметров этих распределений;

во-вторых, реальные значения показателей надежности могут существенно отличаться от теоретических;

в-третьих, для того чтобы подобрать тот или иной закон распределения, достаточно близко описывающий реальную ситуацию, необходимо сначала оценить временную зависимость некоторых показателей надежности.

Как правило, в ходе испытаний и/или эксплуатации технического средства фиксируется время его безотказной работы. По сути, это является выборкой случайной величины  $T$  – времени безотказной работы устройства.

При большом количестве наблюдений рекомендуется эти значения сгруппировать. Для этого определяется диапазон значений случайной ве-

личины  $T$  как разность между максимальным и минимальным значениями. Далее диапазон разбивается на 10 (20) интервалов.

Отношение количества значений случайной величины, попавших в каждый интервал, к объему выборки позволяет определить *частоту случайной величины  $T$* , т.е. оценку вероятности принятия случайной величиной  $T$  наблюдаемого значения из каждого интервала.

Отношение частоты к интервалам времени позволяет вычислить *плотность частоты*, т.е. оценить плотность распределения времени безотказной работы.

На практике число наблюдаемых значений ограничено, и статистический закон распределения является каким-то приближением к истинному закону распределения случайной величины  $T$ . Поэтому следует подбирать аналитическую формулу таким образом, чтобы она отражала существенные черты статистического закона распределения.

Вид плотности распределения подбирают из существа задачи либо по внешнему виду статистического закона распределения. Для этого строят гистограмму частот следующим образом: по временной оси откладывают интервалы времени и над ними строят прямоугольники высотой, равной плотности частоты.

Оценивая визуально гистограмму, выдвигают гипотезу о том или ином распределении. Затем так подбирают параметры, чтобы наилучшим образом (например, по методу наименьших квадратов) согласовать аппроксимирующее распределение со статистическим.

Выдвинутая гипотеза нуждается в проверке с помощью статистического критерия. В случае если гипотеза принимается (напомним, что таким образом получается статистическая оценка плотности распределения времени безотказной работы устройства), она становится основным показателем надежности устройства, на основании которого вычисляются остальные показатели.

### **3.3. Структурная надежность технических средств**

#### **3.3.1. Структурно-логический анализ технических средств**

Конечной целью расчета надежности технических средств информационных систем является оптимизация конструктивных решений и параметров, режимов эксплуатации, организация технического обслуживания и ремонтов. Если учесть, что, как правило, на технические средства, входящие в информационную систему, имеется техническая документация, содержащая, в частности, характеристику надежности, и сами технические средства объединены вычислительной сетью в структурированную кабельную систему (СКС), то оценка надежности технических средств информационной системы сводится к оценке надежности ее СКС. Решение этих задач возможно после предварительного структурно-логического анализа структурированной кабельной системы.

Расчленение СКС на элементы достаточно условно и зависит от постановки задачи расчета надежности. При определении структуры СКС в первую очередь необходимо оценить влияние каждого элемента и его работоспособности на работоспособность системы в целом. С этой точки зрения целесообразно разделить все элементы на четыре группы:

1. Элементы, отказ которых практически не влияет на работоспособность системы.
2. Элементы, работоспособность которых за время эксплуатации практически не изменяется и вероятность безотказной работы близка к единице.
3. Элементы, ремонт или регулировка которых возможны при работе изделия или во время планового технического обслуживания.
4. Элементы, отказ которых сам по себе или в сочетании с отказами других элементов приводит к отказу системы.

Очевидно, при анализе надежности структурированной кабельной системы следует рассматривать только элементы последней группы.

Для расчетов параметров надежности удобно использовать структурно-логические схемы надежности СКС, которые графически отображают взаимосвязь элементов и их влияние на работоспособность системы в целом.

*Структурно-логическая схема* представляет собой совокупность ранее выделенных элементов, соединенных друг с другом последовательно или параллельно. Критерием для определения вида соединения элементов при построении схемы является влияние их отказа на работоспособность ТС.

Анализ структурной надежности СКС, как правило, включает следующие операции:

1. Анализируются устройства и выполняемые системой и ее составными частями функции, а также взаимосвязь составных частей.
2. Формируется содержание понятия «безотказная работа» для данной конкретной системы.
3. Определяются возможные отказы составных частей системы, их причины и возможные последствия.
4. Оценивается влияние отказов составных частей системы на ее работоспособность.
5. Система разделяется на элементы, показатели надежности которых известны.
6. Составляется структурно-логическая схема надежности технической системы, которая является моделью ее безотказной работы.
7. Составляются расчетные зависимости для определения показателей надежности ТС с использованием данных по надежности ее элементов и с учетом структурной схемы.

В зависимости от поставленной задачи на основании результатов расчета характеристик надежности СКС делаются выводы и принимаются решения о необходимости резервирования отдельных элементов или узлов, об установлении определенного режима профилактического обслуживания, о номенклатуре и количестве запасных элементов для ремонта и т.д.

### **3.3.2. Расчет структурной надежности на основе вероятностного метода**

Расчеты показателей безотказности СКС обычно проводятся в предположении, что как вся система, так и любой ее элемент могут находиться только в одном из двух возможных состояний: работоспособном и неработоспособном – и отказы элементов не зависят друг от друга. Состояние системы (работоспособное или неработоспособное) определяется состоянием элементов и их сочетанием. Поэтому теоретически возможен расчет



безотказности любой ТС свести к перебору всех возможных комбинаций состояний элементов, определению вероятности каждого из них и сложению вероятностей работоспособных состояний системы.

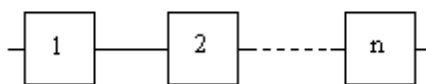
Такой метод (метод прямого перебора) практически универсален и может использоваться при расчете любых СКС. Однако при большом количестве элементов системы  $n$  такой путь становится нереальным из-за большого объема вычислений (например, при  $n = 10$  число возможных состояний системы составляет  $2^n = 1024$ , при  $n = 20$  – превышает  $10^6$ , при  $n = 30$  – более  $10^9$ ). Поэтому на практике используют более эффективные и экономичные методы расчета, не связанные с большим объемом вычислений. Возможность применения таких методов связана со структурой СКС.

Сущность методов заключается в том, что узел сложной конфигурации структурой схемы надежности СКС заменяется на узел более простой конфигурации с сохранением показателей надёжности преобразуемого узла. При этом СКС с преобразованным узлом упрощается и сводится к последовательному или параллельному соединению или их комбинации.

### **Система с последовательным соединением элементов**

*Последовательным* (с точки зрения надежности) считается соединение, при котором отказ любого элемента приводит к отказу всей системы (рис. 13).

Системой с *последовательным соединением элементов* называется система, в которой отказ любого элемента приводит к отказу всей системы.



**Рис. 13. Последовательное соединение элементов**

Такое соединение элементов в технике встречается наиболее часто, поэтому его называют основным соединением. В системе с последовательным соединением для безотказной работы в течение некоторого времени необходимо и достаточно, чтобы каждый из ее  $n$  элементов работал безотказно в течение этого периода времени.

Считая отказы элементов независимыми, вероятность одновременной безотказной работы  $n$  элементов определяется по теореме умножения

вероятностей: вероятность совместного появления независимых событий равна произведению вероятностей этих событий:

$$P(t) = P_1(t)P_2(t)\dots P_n(t) = \prod_{i=1}^n P_i(t) = \prod_{i=1}^n (1 - Q_i(t)).$$

Из формулы видно, что вероятность безотказной работы СКС при последовательном соединении не может быть выше вероятности безотказной работы самого ненадежного из ее элементов (принцип «хуже худшего»), и из малонадежных элементов нельзя создать высоконадежной СКС с последовательным соединением.

Вероятность отказа СКС будет равна

$$Q(t) = 1 - P(t) = 1 - \prod_{i=1}^n P_i(t) = 1 - \prod_{i=1}^n (1 - Q_i(t)).$$

Интенсивность отказов СКС выразится как

$$\lambda(t) = \sum_{i=1}^n \lambda_i(t),$$

т.е. интенсивность отказов СКС при последовательном соединении элементов равна сумме интенсивностей отказов элементов.

Среднее время безотказной работы и дисперсия вычисляются по формулам, указанным ранее:

$$T_{cp} = \int_0^{\infty} P(t) dt,$$

$$DT = 2 \int_0^{\infty} t \cdot P(t) dt - T_{cp}^2.$$

Справедливы следующие утверждения.

**Лемма 1.** Если СКС состоит из последовательно соединенных равнонадежных элементов  $P_i(t) = P_1(t)$  и  $\lambda_i(t) = \lambda_1(t)$ ,  $i=1, \dots, n$ , то

$$P(t) = P_1^n(t), \quad Q(t) = 1 - (1 - Q_1(t))^n, \quad \lambda(t) = n\lambda_1(t), \quad T_{cp} = \frac{T_{cp1}}{n}.$$

Из формул видно, что даже при высокой надежности элементов надежность системы при последовательном соединении оказывается тем более низкой, чем больше число элементов (например, при  $p = 0,95$  и  $n = 10$  имеем  $P = 0,60$ , при  $n = 15$   $P = 0,46$ , а при  $n = 20$   $P = 0,36$ ). А интенсивность отказов СКС в  $n$  раз больше, средняя наработка – в  $n$  раз меньше, чем у отдельного элемента.

**Лемма 2.** Если все последовательно соединенные элементы СКС работают в периоде нормальной эксплуатации, время безотказной работы элементов и СКС подчиняется экспоненциальному распределению, то

$$P(t) = \prod_{i=1}^n \exp(-\lambda_i t) = \exp\left[-\left(\sum_{i=1}^n \lambda_i\right)t\right] = \exp(-\lambda t),$$

$$Q(t) = 1 - \exp(-\lambda t),$$

$$f(t) = \lambda \cdot \exp(-\lambda t),$$

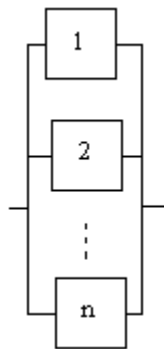
$$T_{cp} = \frac{1}{\lambda},$$

$$DT = \frac{1}{\lambda^2},$$

где  $\lambda = \lambda_1 + \lambda_2 + \dots + \lambda_n = \sum_{i=1}^n \lambda_i = \text{const}$  есть интенсивность отказов системы.

### Система с параллельным соединением элементов

Параллельным (с точки зрения надежности) считается соединение, при котором отказ любого элемента не приводит к отказу системы, пока не откажут все соединенные элементы (рис. 14).



**Рис. 14.** Параллельное соединение элементов

Системой с параллельным соединением элементов называется система, отказ которой происходит только в случае отказа всех ее элементов.

Для отказа системы с параллельным соединением элементов в течение наработки необходимо и достаточно, чтобы все ее элементы отказали в течение этой наработки.

Следовательно, вероятность отказа СКС (при допущении независимости отказов) может быть найдена по теореме умножения вероятностей как произведение вероятностей отказов элементов:

$$Q(t) = Q_1(t) \cdot Q_2(t) \cdot \dots \cdot Q_n(t) = \prod_{i=1}^n Q_i(t) = \prod_{i=1}^n (1 - P_i(t)).$$

Соответственно, вероятность безотказной работы СКС равна

$$P(t) = 1 - Q(t) = 1 - \prod_{i=1}^n Q_i(t) = 1 - \prod_{i=1}^n (1 - P_i(t)).$$

Из формул видно, что вероятность отказа СКС не может быть выше вероятности самого надежного ее элемента («лучше лучшего»), и даже из сравнительно ненадежных элементов возможно построение вполне надежной системы. Такие схемы надежности характерны для СКС, в которых элементы дублируются или резервируются, т.е. параллельное соединение используется как метод повышения надежности.

Справедливы следующие утверждения.

**Лемма 1.** Для СКС, состоящей из параллельно соединенных равнонадежных элементов  $P_i(t) = P_1(t)$ ,

$$Q(t) = Q_1^n(t),$$

$$P(t) = 1 - (1 - P_1(t))^n,$$

т.е. надежность системы с параллельным соединением повышается при увеличении числа элементов (например, при  $p = 0,9$  и  $n = 2$   $P = 0,99$ , а при  $n = 3$   $P = 0,999$ ).

**Лемма 2.** При экспоненциальном законе надежности работы элементов и СКС вероятность времени безотказной работы СКС, состоящей из параллельно соединенных равнонадежных элементов, равна

$$P(t) = 1 - (1 - \exp(-\lambda t))^n.$$

### Система с мостиковой схемой соединения элементов

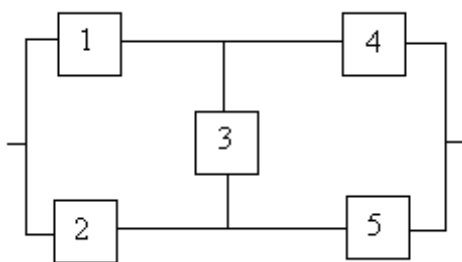


Рис. 15. Мостиковая система

Мостиковая структура (см. рис. 15) не сводится к параллельному или последовательному типу соединения элементов, а представляет собой параллельное соединение последовательных цепочек элементов с диагональными элементами, включенными между узлами различных параллельных ветвей (элемент 3 на рис. 15).

Работоспособность такой системы определяется не только количеством отказавших элементов, но и их положением в структурной схеме. Например, работоспособность системы, схема которой приведена на рис. 15, будет утрачена при одновременном отказе элементов 1 и 2, или 4 и 5, или 2,

3 и 4, или 1, 3 и 5. В то же время отказ элементов 1 и 5, или 2 и 4, или 1, 3 и 4, или 2, 3 и 5 к отказу системы не приводит.

Для анализа надежности систем, структурные схемы которых не сводятся к параллельному или последовательному типу, можно преобразовать исходную схему в эквивалентную ей параллельного или последовательного типа.

Для преобразования структурной схемы используются два метода: *минимальных путей* и *минимальных сечений*.

**Метод минимальных путей.** *Минимальным путем* называется такой набор элементов системы, который обеспечивает ее работоспособность при работоспособности всех его элементов, а отказ любого из них приводит к ее отказу.

Минимальных путей в системе может быть один или несколько. Очевидно, система с последовательным соединением элементов имеет только один минимальный путь, включающий все элементы. В системе с параллельным соединением число минимальных путей совпадает с числом элементов, и каждый путь включает один из них.

Для мостиковой системы из пяти элементов (рис. 15) минимальных путей четыре: (элементы 1 и 4), (2 и 5), (1, 3 и 5), (2, 3 и 5). Эквивалентная схема такой системы (рис. 16), с точки зрения надежности, составляется таким образом, чтобы все элементы каждого минимального пути были соединены друг с другом последовательно, а все минимальные пути – параллельно.

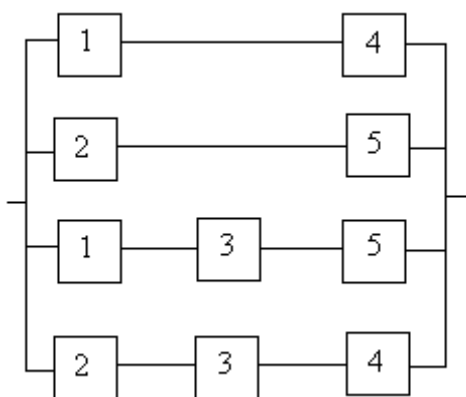


Рис. 16. Структурная схема по методу минимальных путей

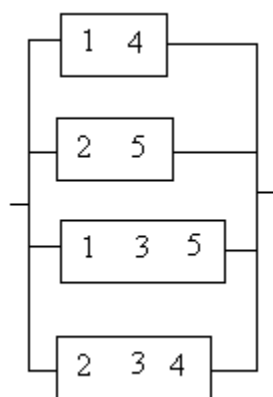


Рис. 17. Структурная схема с укрупненными блоками элементов

Надежность полученной структурной схемы легко может быть найдена по приведенным выше формулам для последовательного и параллельного соединений.

Для облегчения вычисления рекомендуется выделять укрупненные блоки элементов (см., например, рис. 17). Для такой схемы производится расчет надежности как для схемы с параллельно соединенными элементами. Затем рассчитываются надежности каждого блока как схем с последовательным соединением элементов.

Метод минимальных путей дает точное значение только для сравнительно простых систем с небольшим числом элементов. Для более сложных систем формула расчета становится громоздкой, а ее приближенное вычисление является нижней границей вероятности безотказной работы.

**Метод минимальных сечений.** *Минимальным сечением* называется набор элементов, совместный отказ которых приводит к отказу системы, а восстановление работоспособности любого из них – к восстановлению работоспособности системы.

Как и минимальных путей, минимальных сечений может быть несколько. Очевидно, система с параллельным соединением элементов имеет только одно минимальное сечение, включающее все ее элементы (восстановление любого восстановит работоспособность системы). В системе с последовательным соединением элементов число минимальных путей совпадает с числом элементов, и каждое сечение включает один из них.

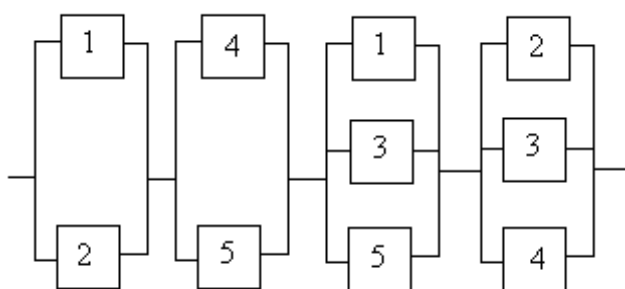


Рис. 18. Структурная схема по методу минимальных сечений

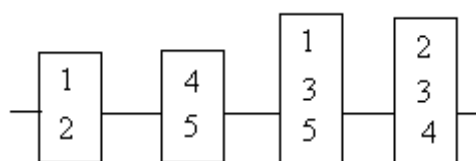


Рис. 19. Структурная схема с укрупненными блоками элементов

В мостиковой системе (рис. 15) минимальных сечений четыре (элементы 1 и 2), (4 и 5), (1, 3 и 5), (2, 3 и 4). Схема системы (см. рис. 18, 19) составляется таким образом, чтобы все элементы каждого минимального

сечения были соединены друг с другом параллельно, а все минимальные сечения – последовательно.

Аналогично методу минимальных путей, надежность полученной структурной схемы легко может быть найдена по приведенным выше формулам для последовательного и параллельного соединения.

Метод минимальных сечений позволяет определить верхнюю границу вероятности безотказной работы СКС мостиковой структуры.

### Системы со схемами соединения элементов типа «треугольник» и «звезда»

Система со схемами соединения элементов типа «треугольник» и «звезда» представлена на рис. 20. Рассмотрим преобразование одного вида соединения в другой.

**Преобразование «треугольника» в «звезду».** Пусть, например, требуется заменить узел в виде «треугольника» элементов 12, 23 и 13 на рис. 20 на узел в виде «звезды» элементов 1, 2 и 3 при условии, что вероятность отказа элемента 12 равна  $Q_{12}$ , элемента 23 –  $Q_{23}$  и элемента 13 –  $Q_{13}$ .

Переход к соединению элементов «звездой» не должен изменять надежность цепей 1-2, 2-3 и 3-1.

Условия сохранения показателей надежности рассматриваемых цепей математически выразятся следующими равенствами, далее рассматриваемыми как уравнения:

$$\begin{cases} Q_1 + Q_2 - Q_1 Q_2 = Q_{12} (Q_{23} + Q_{13} - Q_{23} Q_{13}), \\ Q_2 + Q_3 - Q_2 Q_3 = Q_{23} (Q_{13} + Q_{12} - Q_{13} Q_{12}), \\ Q_3 + Q_1 - Q_3 Q_1 = Q_{13} (Q_{12} + Q_{23} - Q_{12} Q_{23}). \end{cases}$$

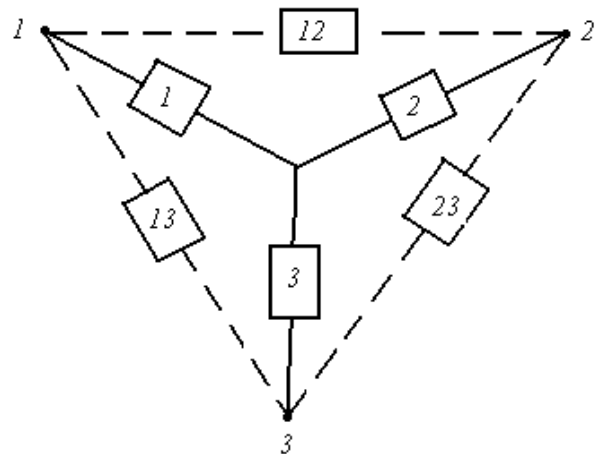


Рис. 20. Схемы соединения элементов типа «треугольник» и «звезда»

Тогда, производя преобразования системы уравнений и пренебрегая малыми членами, решение системы относительно  $Q_1$ ,  $Q_2$ ,  $Q_3$  будет следующим:

$$\begin{cases} Q_1 = Q_{12} \cdot Q_{13}, \\ Q_2 = Q_{23} \cdot Q_{12}, \\ Q_3 = Q_{13} \cdot Q_{23}. \end{cases}$$

**Преобразование «звезды» в «треугольник».** При обратном преобразовании «звезды» элементов в «треугольник» необходимо найти решение системы уравнений относительно  $Q_{12}$ ,  $Q_{23}$ ,  $Q_{13}$ . Путем несложных преобразований можно получить следующие окончательные выражения:

$$\begin{cases} Q_{12} = \sqrt{\frac{Q_1 Q_2}{Q_3}}, \\ Q_{23} = \sqrt{\frac{Q_2 Q_3}{Q_1}}, \\ Q_{13} = \sqrt{\frac{Q_1 Q_3}{Q_2}}. \end{cases}$$

### 3.3.3. Расчет надежности передачи информации на основе логико-вероятностного метода

Для передачи данных в информационных системах наиболее часто применяются следующие сетевые структуры:

- с обходными каналами (ОВК);
- иерархическая несимметричная 3-уровневая, состоящая из 5 каналов передачи информации (IER.5)
- иерархическая симметричная, состоящая из 6 каналов передачи информации (IER.6);
- сетевая структура, состоящая из 5 каналов (SET.5);
- сетевая структура, состоящая из 6 каналов (SET.6), причем один из каналов двунаправленный
- и их обобщения на  $n$ -узлов.

При использовании логико-вероятностного метода расчета надежности структурных схем применяют законы и теоремы алгебры логики, в том числе теорему разложения, согласно которой булева функция может быть представлена следующим образом:

$$f(a_1, a_2, \dots, a_n) = a_1 \cdot f(1, a_2, \dots, a_n) \vee \bar{a}_1 \cdot f(0, a_2, \dots, a_n)$$

или

$$f(a_1, a_2, \dots, a_n) = [a_1 \vee f(0, a_2, \dots, a_n)][\bar{a}_1 \cdot f(1, a_2, \dots, a_n)].$$

Здесь  $a_i$  – логические переменные.

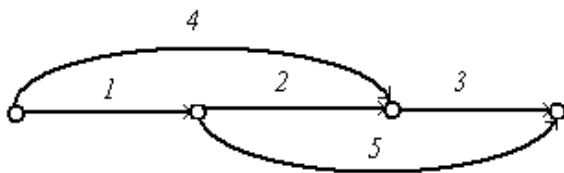


Сущность этого метода состоит в следующем:

- состояние каждого элемента кодируется нулем (если элемент в состоянии отказа) или единицей (если элемент в исправном состоянии);
- в булевой функции состояния элементов представляются как булева переменная (если код 1) или ее отрицание (если код 0);
- записывается условие работоспособности схемы через работоспособность ее элементов в виде логической функции;
- далее логическая функция упрощается по законам логики;
- затем вместо булевых переменных подставляются вероятности безотказной работы, а вместо отрицаний булевых функций – вероятности отказов;
- полученное выражение и будет вероятностью безотказной работы схемы элементов.

Условие работоспособности структурной схемы может быть различным, примеры его приведены при рассмотрении конкретных типов схем.

**Система со схемой соединения элементов типа «обходные каналы» (ОВК)** показана на рис. 21.



**Рис. 21. Соединение элементов типа ОВК**

Обозначим через  $A_i$  событие, заключающееся в том, что  $i$ -й канал работоспособен. Тогда логическая функция работоспособности структуры запишется так:

$$f(a_1, a_2, a_3, a_4, a_5) = a_1 a_2 a_3 \vee a_1 a_5 \vee a_3 a_4.$$

Переход к вероятностям в функции потребует дополнительных трудоемких преобразований. Поэтому предварительно логическая функция упрощается:

$$f(a_1, a_2, a_3, a_4, a_5) = a_1 (a_2 a_3 \vee a_5 \vee a_3 a_4) \vee \bar{a}_1 a_3 a_4 = a_1 (a_3 (a_2 \vee a_4) \vee a_5) \vee \bar{a}_1 a_3 a_4.$$

Подставим вероятности безотказной работы или вероятности отказов вместо логических переменных и их отрицаний и учтем формулу суммы вероятностей. Получим

$$P = P_1 (P_3 (P_2 + P_4 - P_2 P_4) + P_5) + (1 - P_1) P_3 P_4.$$

Система со схемой соединения элементов типа «иерархическая несимметричная структура» (IER.5) показана на рис. 22.

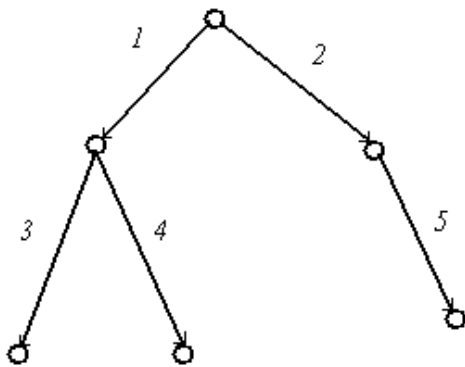


Рис. 22. Соединение элементов типа IER.5

Обозначим через  $A_i$  событие, заключающееся в том, что  $i$ -й канал работоспособен. Тогда логическая функция того, что информация дойдет до всех трех нижних получателей, запишется так:

$$f(a_1, a_2, a_3, a_4, a_5) = a_1 a_2 a_3 a_4 a_5.$$

Подставим вероятности безотказной работы вместо логических переменных. Получим

$$P = P_1 P_2 P_3 P_4 P_5.$$

Система со схемой соединения элементов типа «иерархическая симметричная структура» (IER.6) показана на рис. 23.

Обозначим через  $A_i$  событие, заключающееся в том, что  $i$ -й канал работоспособен. Тогда логическая функция того, что информация дойдет до трех нижних получателей из четырех, запишется так:

$$f(a_1, a_2, a_3, a_4, a_5, a_6) = a_1 a_2 a_3 a_4 a_5 \bar{a}_6 \vee a_1 a_2 a_3 a_4 \bar{a}_5 a_6 \vee a_1 a_2 a_3 \bar{a}_4 a_5 a_6 \vee a_1 a_2 \bar{a}_3 a_4 a_5 a_6.$$

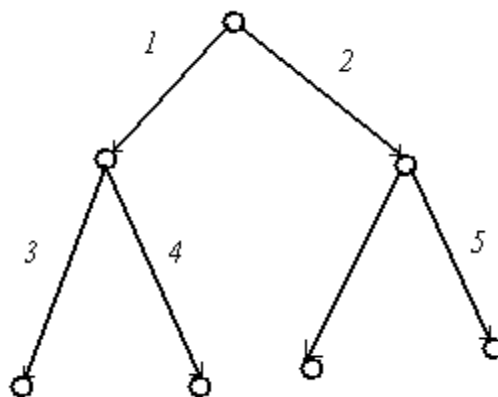


Рис. 23. Соединение элементов типа IER.6

Подставим вероятности безотказной работы или вероятности отказов вместо логических переменных. Получим

$$P = P_1 P_2 \left( P_3 \left( P_4 \left( P_5 \bar{P}_6 + \bar{P}_5 P_6 \right) + \bar{P}_4 P_5 P_6 \right) + \bar{P}_3 P_4 P_5 P_6 \right).$$

Системы со схемами соединения элементов типа «сетевые структуры» с пятью каналами (SET.5) и с шестью каналами (SET.6) показаны на рис. 24, 25.

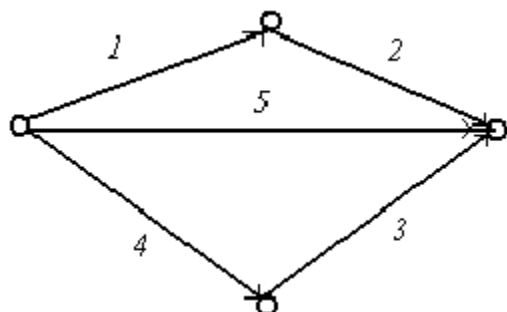


Рис. 24. Соединение элементов типа SET.5

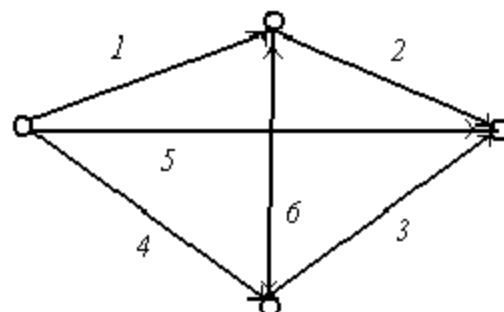


Рис. 25. Соединение элементов типа SET.6

Расчет надежности передачи данных по разным условиям работоспособности сетевых структур производится аналогично предыдущим рассмотренным схемам.

Соответствие архитектур информационных систем структурным схемам надежности приведено в табл. 1-3.

Таблица 1

Структурная схема надежности распределенной архитектуры ИС

<i>Топология сети</i>	<i>Тип структурной схемы надежности</i>
<p>Комбинированная</p>	<p>Клиенты соединены параллельно, серверы – параллельно (при условии их реализации на структурах: кластер, grid или «облако»)</p>

Таблица 2

Структурная схема надежности файл-серверной архитектуры ИС

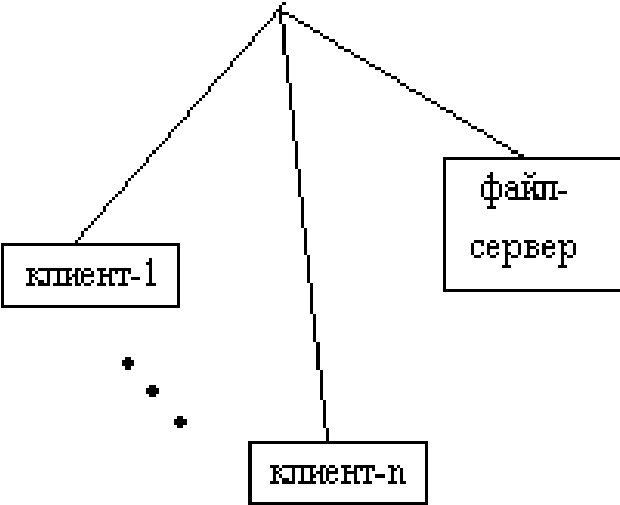
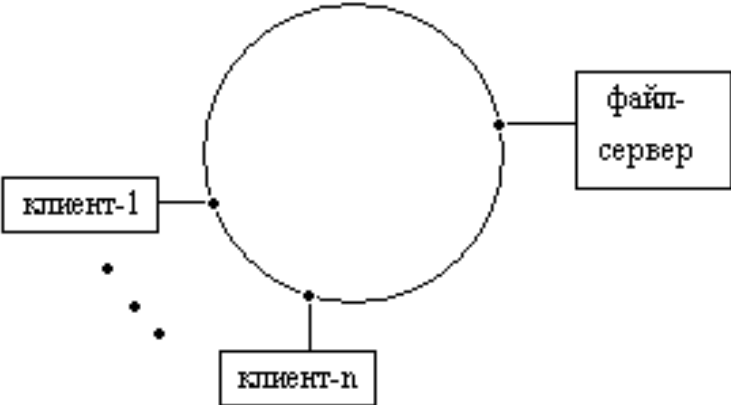
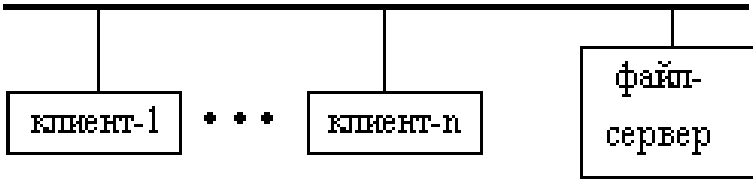
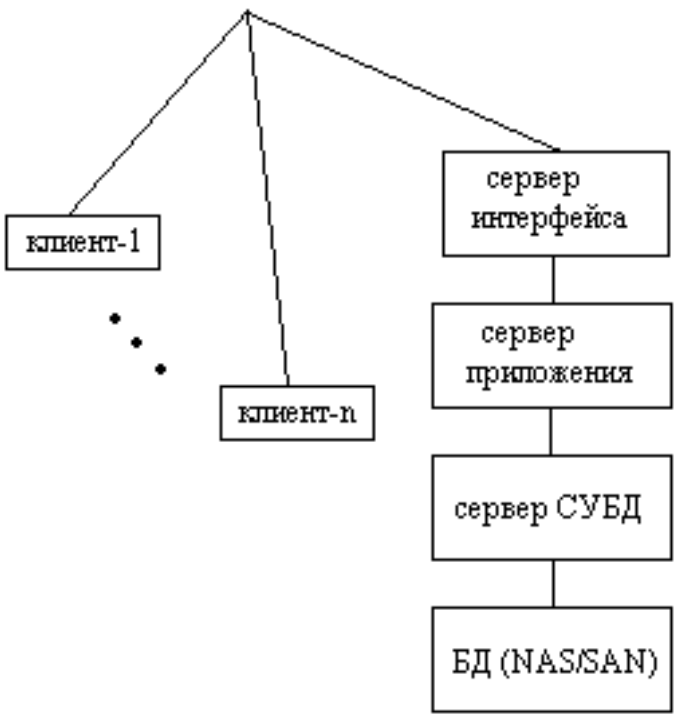
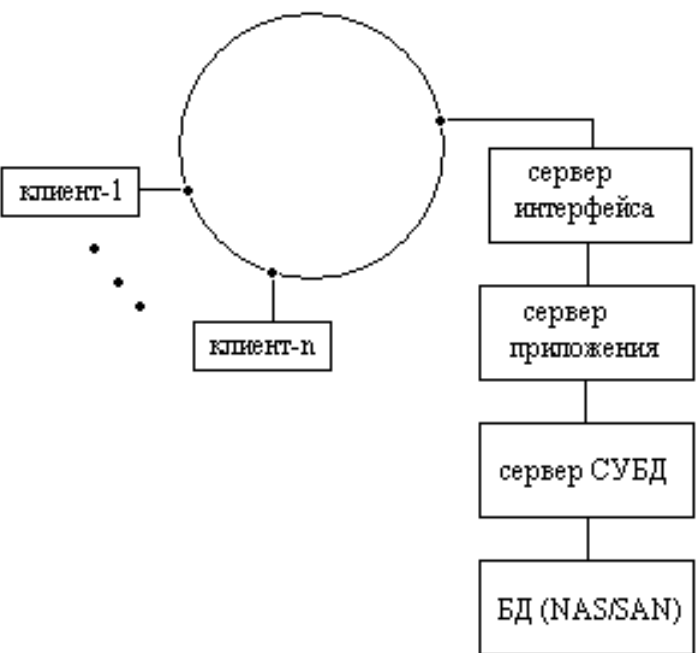
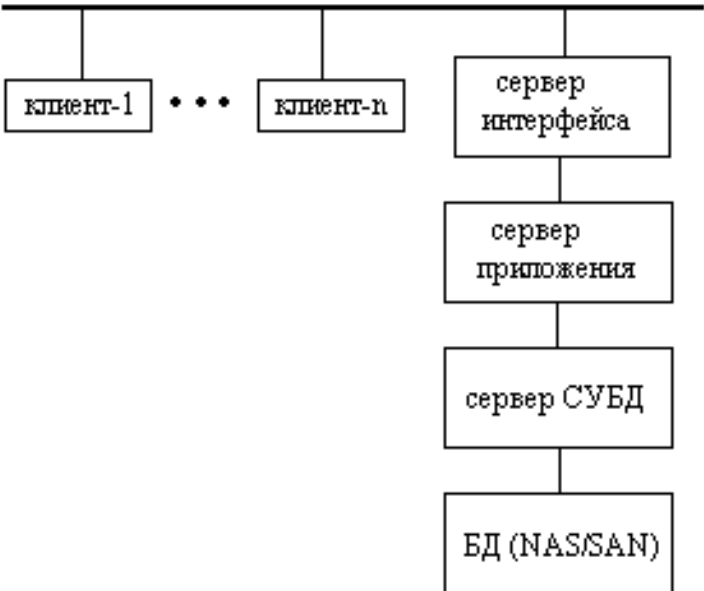
Топология сети	Тип структурной схемы надежности
<p>Звезда</p>  <p>The diagram shows a central point at the top with three lines radiating downwards. The leftmost line connects to a box labeled 'клиент-1'. The middle line connects to a box labeled 'клиент-n'. The rightmost line connects to a box labeled 'файл-сервер'. Three small dots are placed between 'клиент-1' and 'клиент-n' to indicate intermediate clients.</p>	
<p>Кольцо</p>  <p>The diagram shows a large circle representing a ring. On the left side, a box labeled 'клиент-1' is connected to the circle. At the bottom, a box labeled 'клиент-n' is connected to the circle. On the right side, a box labeled 'файл-сервер' is connected to the circle. Three small dots are placed between 'клиент-1' and 'клиент-n' to indicate intermediate nodes.</p>	<p>Клиенты соединены параллельно, серверы – последовательно</p>
<p>Общая шина</p>  <p>The diagram shows a thick horizontal line representing a bus. Three vertical lines connect this bus to three boxes below it. From left to right, the boxes are labeled 'клиент-1', 'клиент-n', and 'файл-сервер'. Three small dots are placed between 'клиент-1' and 'клиент-n' to indicate intermediate clients.</p>	

Таблица 3

Структурная схема надежности клиент-серверной архитектуры ИС

Топология сети	Тип структурной схемы надежности
<p>Звезда</p> 	<p>Клиенты соединены параллельно,</p>
<p>Кольцо</p> 	<p>серверы – по схеме IER-n</p>

<i>Топология сети</i>	<i>Тип структурной схемы надежности</i>
<p>Общая шина</p>  <pre> graph TD     Bus[Общая шина] --- C1[клиент-1]     Bus --- Dots[...]     Bus --- Cn[клиент-n]     Bus --- SI[сервер интерфейса]     SI --- SA[сервер приложения]     SA --- SUD[сервер СУБД]     SUD --- BDN[БД (NAS/SAN)] </pre>	

### 3.4. Повышение надежности технических средств

#### 3.4.1. Резервирование технических средств

Расчетные зависимости для определения основных характеристик надежности структурированной кабельной системы ИС показывают, что надежность системы зависит от ее структуры (структурно-логической схемы) и надежности элементов. Поэтому для сложных систем возможны два пути повышения надежности:

- повышение надежности элементов,
- изменение структурной схемы.

Рассмотрение методов обеспечения надежности элементов является предметом специальных технологических и физико-химических дисциплин и выходит за рамки теории надежности.

Изменение структуры системы с целью повышения надежности означает перестройку конструктивной/функциональной схемы СКС (структуры связей между составными элементами) или введение в систему до-

полнительных, избыточных элементов, включающихся в работу при отказе основных.

Применение дополнительных средств и возможностей с целью сохранения работоспособного состояния структурированной кабельной системы ИС при отказе одного или нескольких ее элементов называется *резервированием*.

Принцип резервирования подобен рассмотренному ранее параллельному соединению элементов, где за счет избыточности возможно обеспечение более высокой надежности системы, чем ее элементов.

Выделяют несколько видов резервирования: временное, информационное, функциональное и др. Для анализа структурной надежности ТС интерес представляет *структурное резервирование* – введение в структуру объекта дополнительных элементов, выполняющих функции основных элементов в случае их отказа.

Классификация различных способов структурного резервирования осуществляется по различным признакам:

***по схеме включения резерва:***

- *общее резервирование*, при котором резервируется объект в целом;
- *раздельное резервирование*, при котором резервируются отдельные элементы или их группы;
- *смешанное резервирование*, при котором различные виды резервирования сочетаются в одном объекте;

***по способу включения резерва:***

- *постоянное резервирование*, без перестройки структуры объекта при возникновении отказа его элемента;
- *динамическое резервирование*, при котором при отказе элемента происходит перестройка структуры схемы.

В свою очередь динамическое резервирование подразделяется на:

- *резервирование замещением*, при котором функции основного элемента передаются резервному только после отказа основного;
- *скользящее резервирование*, при котором несколько основных элементов резервируется одним или несколькими резервными, каждый из которых может заменить любой основной (т.е. группы основных и резервных элементов идентичны);

**по состоянию резерва:**

- *нагруженное резервирование*, при котором резервные элементы (или один из них) находятся в режиме основного элемента;
- *облегченное резервирование*, при котором резервные элементы (по крайней мере один из них) находятся в менее нагруженном режиме по сравнению с основными;
- *ненагруженное резервирование*, при котором резервные элементы до начала выполнения ими функций находятся в ненагруженном режиме.

Основной характеристикой структурного резервирования является *кратность резервирования* – отношение числа резервных элементов к числу резервируемых ими основных элементов, выраженное несокращаемой дробью (типа 2:3; 4:2 и т.д.).

Резервирование одного основного элемента одним резервным (т.е. с кратностью 1:1) называется *дублированием*.

Количественно повышение надежности системы в результате резервирования или применения высоконадежных элементов можно оценить по *коэффициенту выигрыша надежности*, определяемому как отношение показателя надежности до и после преобразования системы. Например, для системы из  $n$  последовательно соединенных элементов после резервирования одного из элементов ( $k$ -го с вероятностью  $P_k$ ) аналогичным по надежности элементом коэффициент выигрыша надежности по вероятности безотказной работы составит

$$G_p = \frac{P_{\text{до резервирования}}}{P_{\text{после резервирования}}} = 2 - P_k.$$

Из формулы следует, что эффективность резервирования (или другого приема повышения надежности) тем больше, чем меньше надежность резервируемого элемента (при  $P_k = 0,9$   $G_p = 1,1$ ; при  $P_k = 0,5$   $G_p = 1,5$ ). Следовательно, при структурном резервировании максимального эффекта можно добиться при резервировании самых ненадежных элементов (или групп элементов).

В общем случае при выборе элемента (или группы элементов) для повышения надежности или резервирования необходимо исходить из условия обеспечения при этом максимального эффекта.



### 3.4.2. Расчет надежности систем с резервированием

Расчет показателей надежности систем с резервированием отдельных элементов или групп элементов во многом определяется видом резервирования. Ниже рассматриваются схемы расчетов для самых распространенных случаев простого резервирования, к которым путем преобразований может быть приведена и структура смешанного резервирования. При этом расчетные зависимости получены без учета надежности переключающих устройств, обеспечивающих перераспределение нагрузки между основными и резервными элементами (т.е. для «идеальных» переключателей). В реальных условиях введение переключателей в структурную схему необходимо учитывать и в расчете надежности систем.

#### Расчет надежности систем с нагруженным резервированием

Расчет систем с *нагруженным резервированием* осуществляется по формулам последовательного и параллельного соединения элементов. При этом считается, что резервные элементы работают в режиме основных как до, так и после их отказа, поэтому надежность резервных элементов не зависит от момента их перехода из резервного состояния в основное и равна надежности основных элементов.

Для системы с последовательным соединением  $n$  элементов при общем резервировании с кратностью  $k$  (рис. 26) вероятность безотказной работы будет равна

$$P(t) = 1 - (1 - P)^{k+1} = 1 - \left(1 - \prod_{i=1}^n P_i(t)\right)^{k+1}.$$

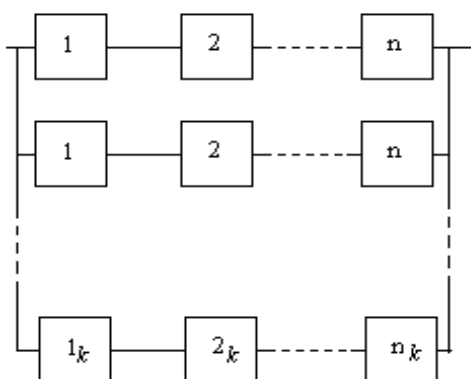


Рис. 26. Система с последовательным соединением  $n$  элементов при общем нагруженном резервировании с кратностью  $k$

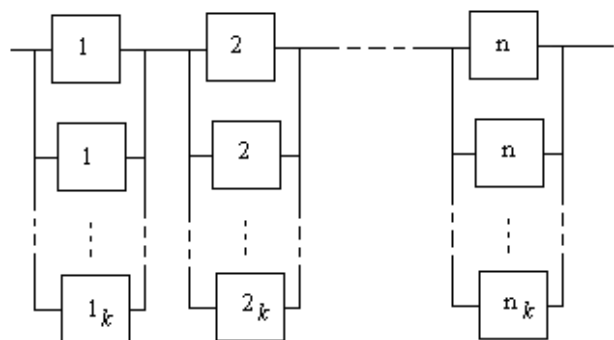


Рис. 27. Система с последовательным соединением  $n$  элементов при раздельном нагруженном резервировании с кратностью  $k$

При отдельном резервировании (рис. 27)

$$P(t) = \prod_{i=1}^n (1 - (1 - P_i(t))^{k+1}).$$

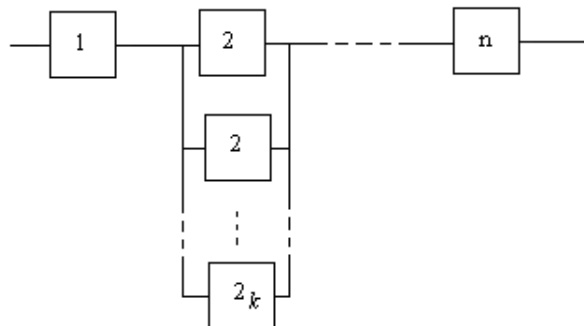
### Расчет надежности систем с ненагруженным резервированием

При *ненагруженном резервировании* резервные элементы последовательно включаются в работу при отказе основного, затем первого резервного и т.д. (рис. 28), поэтому надежность резервных элементов зависит от момента их перехода в основное состояние. Такое резервирование в различных ТС встречается наиболее часто, так как оно по сути аналогично замене отказавших элементов и узлов на запасные.

Если резервные элементы до их включения абсолютно надежны, то для системы с ненагруженным резервированием кратности  $k$  (всего элементов  $k + 1$ )

$$P(t) = 1 - \frac{1}{(k + 1)!} \prod_{i=1}^{k+1} (1 - P_i(t)),$$

т.е. вероятность отказа в  $(k+1)!$  раз меньше, чем при нагруженном.



**Рис. 28.** Система с последовательным соединением  $n$  элементов при ненагруженном резервировании с кратностью  $k$

При ненагруженном резервировании средняя наработка на отказ равна

$$T_{cp} = \sum_{i=1}^{k+1} T_{cp_i}.$$

### Расчет надежности систем с облегченным резервированием

*Облегченное резервирование* используется при большой инерционности переходных процессов, происходящих в элементе при его переходе из



## 4. НАДЕЖНОСТЬ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ИНФОРМАЦИОННОЙ СИСТЕМЫ

### 4.1. Основные понятия

*Надежность (reliability)* программного обеспечения (ПО) – это его способность безотказно выполнять определенные функции при заданных условиях в течение заданного периода времени с достаточно большой вероятностью. При этом под *отказом* в ПО понимают проявление в нем ошибки. В программном средстве имеется *ошибка (software error)*, если оно не выполняет того, что разумно ожидать от него пользователю. Таким образом, надежное ПО не исключает наличия в нем ошибок – важно лишь, чтобы эти ошибки при практическом применении этого ПО в заданных условиях проявлялись достаточно редко. Убедиться, что ПО обладает таким свойством, можно при его испытании путем тестирования, а также при практическом применении.

Примитивами надежности ПО являются: завершенность, точность, автономность, устойчивость, защищенность.

*Завершенность (completeness)* – свойство, характеризующее степень обладания ПО всеми необходимыми частями и чертами, требующимися для выполнения своих явных и неявных функций.

*Точность (accuracy)* – мера, характеризующая приемлемость величины погрешности в выдаваемых программами ПО результатах с точки зрения предполагаемого их использования.

*Автономность (self-containedness)* – свойство, характеризующее способность ПО выполнять предписанные функции без помощи или поддержки других компонент программного обеспечения.

*Устойчивость (robustness)* – свойство, характеризующее способность ПО продолжать корректное функционирование, несмотря на задание неправильных (ошибочных) входных данных.

*Защищенность (defensiveness)* – свойство, характеризующее способность ПО противостоять преднамеренным или нечаянным деструктивным (разрушающим) действиям пользователя.

*Живучесть* – свойство, характеризующее способность ПО продолжать корректное функционирование, несмотря на сбой отдельных частей ПО.

Программное средство имеет ряд специфических особенностей:

– оно представляет собой некоторую совокупность текстов (т.е. *статических* объектов), смысл же (семантика) этих текстов выражается процессами обработки данных и действиями пользователей, запускающих эти процессы (т.е. является *динамическим*);

– ПО при своем использовании (эксплуатации) не расходуется и не расходует используемых ресурсов.

ПО может обладать различной степенью надежности. Так же как в технике, степень надежности можно характеризовать вероятностью работы ПО без отказа в течение определенного периода времени. В силу специфических особенностей программного средства определение вероятности работы ПО без отказа в течение определенного периода времени наталкивается на ряд трудностей по сравнению с решением этой задачи в технике.

При оценке степени надежности ПО следует также учитывать последствия каждого отказа. Некоторые ошибки в ПО могут вызывать лишь некоторые неудобства при его применении, тогда как другие ошибки могут иметь катастрофические последствия, например угрожать человеческой жизни. Поэтому для оценки надежности ПО иногда используют дополнительные показатели, учитывающие стоимость (вред) для пользователя каждого отказа.

## **4.2. Оценка надежности программных средств**

### **Базовые показатели надежности ПО**

На оценку надежности ПС, заключающуюся в определении вероятности безотказной работы ПС в заданных условиях в течение определенного периода времени, существенное влияние оказывают два фактора:

– программные методы и средства технологии программирования, применяемые в процессах разработки и способствующие достижению требуемой надежности;

– тестирование и проверка функционирования созданного ПС со сбором данных о результатах обнаружения ошибок и интенсивности отказов в интервалах времени функционирования.

Методы оценки надежности базируются на аналогичных методах и в теории надежности технических средств и отличаются тем, что ошибки в

ПС устраняются, улучшая их качество. Отказы в технических средствах, как правило, являются следствием износа и меньше всего зависят от ошибок проектирования (аппаратные средства логически проще, чем программные). Отказы при функционировании технических средств могут привести к непригодности или физическому износу этих средств.

Базовые понятия надежности программного обеспечения заимствовались из теории надежности аппаратных средств.

$P(t)$  – вероятность того, что ни одна ошибка не появится на интервале от 0 до  $t$  («функция надежности»):

$$P(t) = P(T \geq t).$$

$Q(t)$  – вероятность того, что ошибка появится на интервале от 0 до  $t$  («функция отказов»):

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

$f(t)$  – плотность распределения времени безотказной работы ПО:

$$f(t) = Q'(t) = -P'(t).$$

$\lambda(t)$  – интенсивность отказов, или условная вероятность того, что ошибка появится на интервале от 0 до  $t + \Delta t$  при условии, что на интервале от 0 до  $t$  ошибок не было (иногда называется «функцией риска»):

$$\lambda(t) = -\frac{P'(t)}{P(t)} = \frac{f(t)}{P(t)}, \quad P(t) = \exp\left(-\int_0^t \lambda(u) du\right)$$

$T_{cp}$  – среднее время между отказами:

$$T_{cp} = \int_0^{\infty} P(u) du = \int_0^{\infty} u \cdot f(u) du.$$

## Классификация Хетча

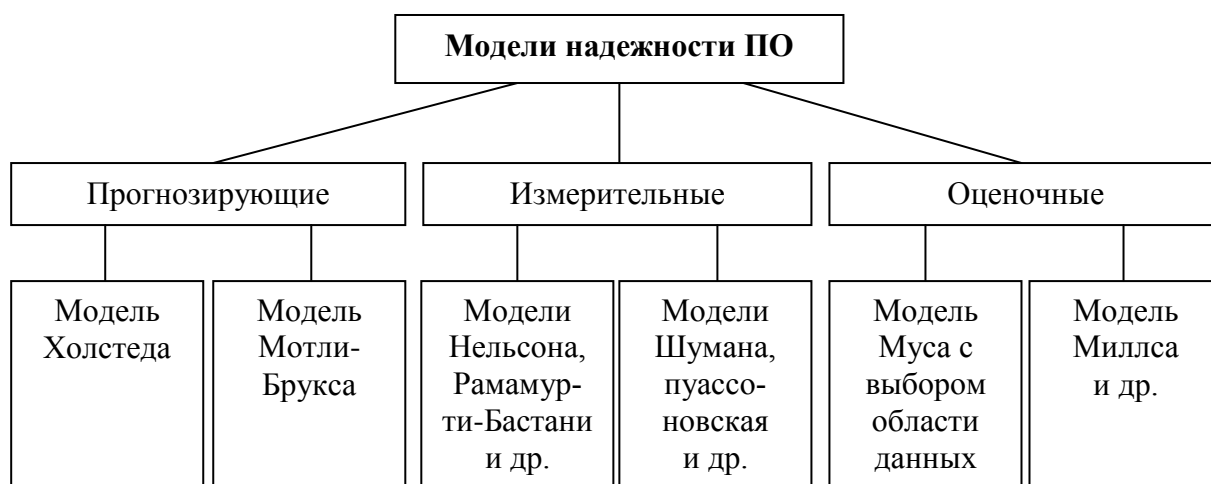
Исследования программных средств на надежность интенсивно проводились по основным направлениям создания трех типов программных систем.

К первому типу систем относятся *программы решения инженерных и научно-технических задач*. Они характеризуются неполным жизненным циклом, небольшим объемом и при эксплуатации не требуют всех ресурсов ЭВМ, носят эпизодический и кратковременный характер.

Второй тип представляет собой класс *программных систем для информационно-справочных и автоматизированных систем обработки информации, функционирующих вне реального времени.*

К третьему типу относятся *программы и комплексы, входящие в контур управления и функционирующие в реальном времени,* используя все ресурсы ЭВМ (память, быстродействие и др.).

В соответствии с классификацией Хетча в классе этих типов систем сформировались три типа моделей надежности (рис. 30).



**Рис. 30. Типы моделей надежности ПО по Хетчу**

*Прогнозирование надежности* – это утверждение, основанное на количественных характеристиках создаваемых программ (длина программы, объем, число переменных и др.). Примером таких моделей является модель надежности Холстеда, которая применяется для прогнозирования надежности на ранних процессах разработки (программирование и сборка).

*Оценочные модели надежности* основаны на результатах тестирования, получаемых в тестовой среде при прогонах ПО на тестах и используемых при определении вероятности отказов ПО в реальной операционной среде функционирования. Данные модели исходят из частоты проявления и устранения причины, вызвавшей отказ ПО, поэтому их называют моделями интенсивности (сходство с моделями оценки надежности технических средств).

*Измерительные модели* предназначены для измерения надежности программного обеспечения, работающего с заданной внешней средой и имеющего следующие ограничения:

- программное обеспечение не модифицируется во время периода измерений свойств надежности;
- обнаруженные ошибки не исправляются;
- измерение надежности проводится для зафиксированной конфигурации программного обеспечения.

Типичными примерами таких моделей являются модели Нельсона и Рамамурти-Бастани.

Оценочные модели основываются на серии тестовых прогонов и проводятся на процессах тестирования ПО. В тестовой среде определяется вероятность отказа программы при ее выполнении или тестировании.

Эти типы моделей могут применяться на процессах жизненного цикла программного обеспечения. Кроме того, результаты прогнозирующих моделей могут использоваться как входные данные для оценочной модели. Имеются модели (например, модель Муса), которые можно рассматривать как оценочные и в то же время как измерительные модели.

### **Классификация Гоэла**

Другой вид классификации моделей предложил А.Л. Гоэл. Согласно этой классификации модели надежности базируются на отказах и разбиваются на четыре класса моделей (рис. 31):

- без подсчета ошибок,
- с подсчетом отказов,
- с подсевом ошибок,
- модели с выбором области входных значений.

*Модели без подсчета ошибок* основаны на измерении интервала времени между отказами и позволяют спрогнозировать количество ошибок, оставшихся в программе. После каждого отказа оценивается надежность и определяется среднее время до следующего отказа. К таким моделям относятся модели Джелински-Моранды, Шика-Вулвертона и Литвуда-Вералла.

*Модели с подсчетом отказов* базируются на количестве ошибок, обнаруженных на заданных интервалах времени. Возникновение отказов в зависимости от времени является стохастическим процессом с непрерывной интенсивностью, а количество отказов является случайной величиной. Обнаруженные ошибки устраняются, и поэтому количество ошибок в еди-



ницу времени уменьшается. К этому классу моделей относятся модели Шумана, Шика-Вулвертона, пуассоновская модель и др.



Рис. 31. Типы моделей надежности ПО по Гоэлу

*Модели с подсевом ошибок* основаны на количестве устраненных ошибок и подсеве, внесенном в программу искусственных ошибок, тип и количество которых заранее известны. Затем определяется соотношение числа оставшихся прогнозируемых ошибок и числа искусственных ошибок, которое сравнивается с соотношением числа обнаруженных действительных ошибок и числа обнаруженных искусственных ошибок. Результат сравнения используется для оценки надежности и качества программы. При внесении изменений в программу проводится повторное тестирование и оценка надежности. Этот подход к организации тестирования отличается громоздкостью и редко используется из-за дополнительного объема работ, связанных с подбором, выполнением и устранением искусственных ошибок. Примером может служить модель Миллса.

*Модели с выбором области входных значений* основываются на генерации множества тестовых выборок из входного распределения. Оценка надежности проводится по полученным отказам на основе тестовых выборок из входной области. К этому типу моделей относится, например, модель Нельсона.

Таким образом, классификация моделей роста надежности относительно процесса выявления отказов фактически делит модели на две группы:

– модели, рассматривающие количество отказов как марковский процесс;

– модели, которые рассматривают интенсивность отказов как пуассоновский процесс. Фактор распределения интенсивности отказов разделяет модели на экспоненциальные, логарифмические, геометрические, байесовские и др.

Марковский процесс характеризуется дискретным временем и конечным множеством состояний. Временной параметр пробегает неотрицательные числовые значения, а процесс (цепочка) определяется набором вероятностей перехода  $p_{ij}(n)$ , т.е. вероятностью на  $n$ -шаге перейти из состояния  $i$ -го в состояние  $j$ -е. Процесс называется однородным, если он не зависит от  $n$ .

В моделях, базирующихся на процессе Маркова, предполагается, что количество дефектов, обнаруженных в ПС, в любой момент времени зависит от поведения системы и представляется в виде стационарной цепи Маркова. При этом количество дефектов конечное, но является неизвестной величиной, которая задается для модели в виде константы. Интенсивность отказов в ПО, или скорость прохода по цепи, зависит лишь от количества дефектов, которые остались в ПО. К этой группе моделей относятся модели Джелински-Моранды, Шика-Вулвертона, Шантикумера.

Модели надежности пуассоновского типа базируются на выявлении отказов и моделируются неоднородным процессом, который задает  $\{M(t), t \geq 0\}$  – неоднородный пуассоновский процесс с функцией интенсивности  $\lambda(t)$ , что соответствует общему количеству отказов ПО за время его использования  $t$ . Примером таких моделей является модель Гоэла-Окумото.

### **Ранее принятая классификация**

Модели надежности программных средств можно также подразделить на аналитические и эмпирические (см. рис. 32). *Аналитические* модели дают возможность рассчитать количественные показатели надежности, основываясь на данных о поведении программы в процессе тестирования. *Эмпирические* модели базируются на анализе структурных особенностей программ.

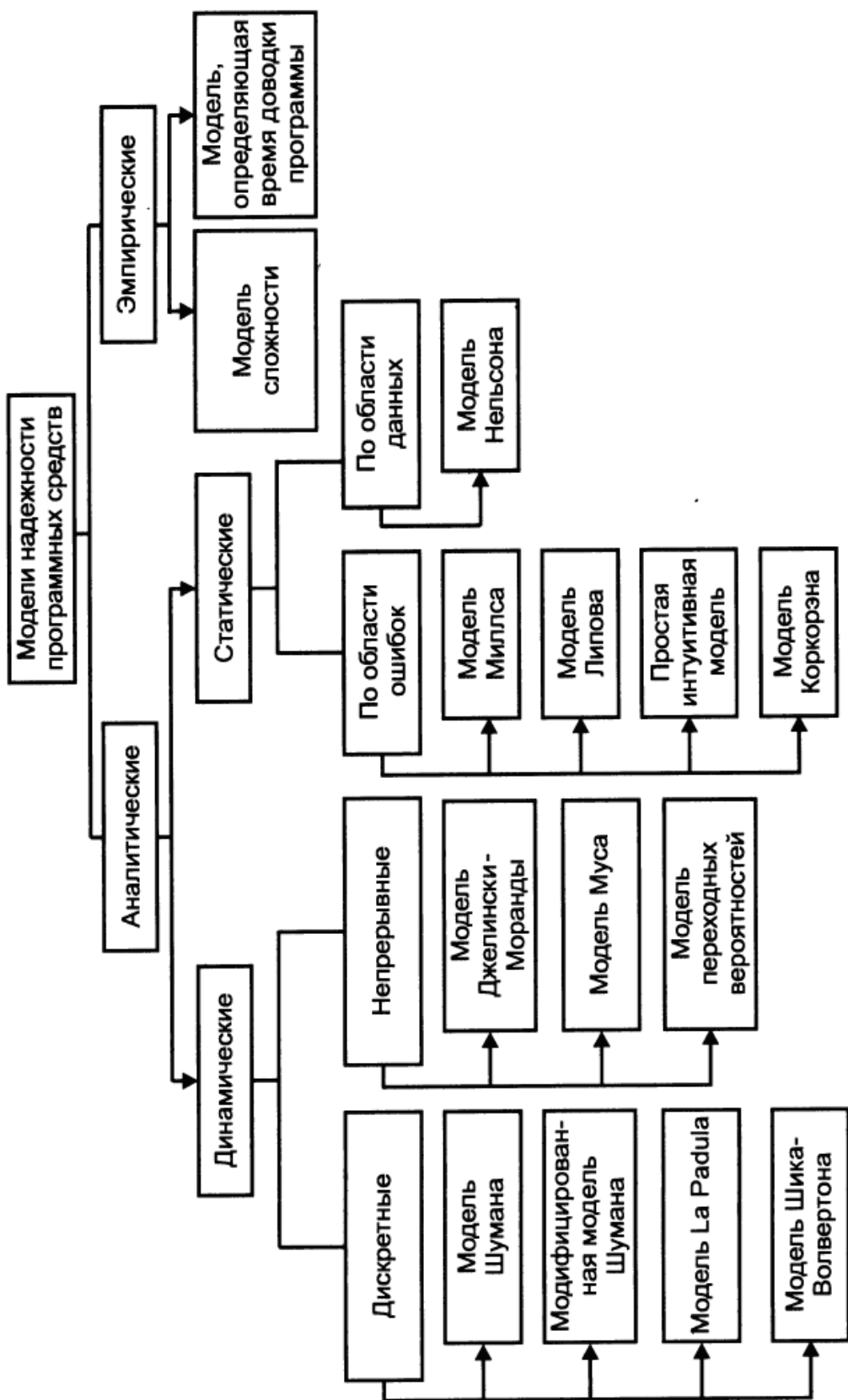


Рис. 32. Классификационная схема моделей надежности ПО

Аналитические модели представлены двумя группами:

- динамические,
- статические.

В динамических моделях поведение ПО (появление отказов) рассматривается во времени. Если фиксируются интервалы каждого отказа, то получается непрерывная картина появления отказов во времени (модели с непрерывным временем). Может фиксироваться только число отказов за произвольный интервал времени. В этом случае поведение ПО может быть представлено только в дискретных точках (модели с дискретным временем).

В статических моделях появление отказов не связывают со временем, а учитывают зависимость количества ошибок либо от числа тестовых прогонов (модели по области ошибок), либо от характеристики входных данных (модели по области данных).

В настоящее время в распоряжении специалистов имеется достаточное количество эмпирических и аналитических моделей, обеспечивающих с той или иной степенью точности расчет числовых оценок показателей надежности программного обеспечения на различных стадиях его жизненного цикла.

Большинство моделей надежности программного обеспечения определяет надежность программного обеспечения на начальных стадиях жизненного цикла. Применение этих моделей для оценки завершающих стадий жизненного цикла программного обеспечения ограничено по следующим причинам:

- на фазах производства и использования программного обеспечения информация о процессе отладки, обнаружении и устранении ошибок, как правило, недоступна;
- отказы при приемо-сдаточных испытаниях малоинтенсивны или отсутствуют.

Поэтому для определения надежности программного обеспечения на всех стадиях его жизненного цикла целесообразно применять как минимум две модели надежности программного обеспечения.

Модель надежности программного обеспечения для фазы разработки выбирается для каждой конкретной программы. Для этого нужно собрать данные об ошибках, на основании имеющихся данных выбрать модель надежности, а затем выполнить тесты, показывающие, насколько эта модель подходит.

Для определения надежности программного обеспечения на завершающих стадиях наиболее эффективно применять модели надежности с системно-независимым аргументом, например модель Нельсона.

Далее будут подробно рассмотрены некоторые из них.

#### 4.2.1. Аналитические динамические модели

##### Модель Шумана

Исходные данные для модели Шумана собираются в процессе тестирования ПО в течение фиксированных или случайных временных интервалов. Использование модели Шумана 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.

По завершении этапа на основе собранных данных модель Шумана может быть использована для расчета количественных показателей надежности. После этого исправляются ошибки, обнаруженные на предыдущем этапе, при необходимости корректируются тестовые наборы, и проводится новый этап тестирования.

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 (т.е. новые ошибки не вносятся).

В рассматриваемой модели предполагается, что значение функции  $z(t)$  пропорционально числу ошибок, оставшихся в ПО после израсходованного на тестирование времени  $\tau$ :

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

где  $C$  – коэффициент пропорциональности, константа;  $t$  – время работы ПО без отказа;  $\varepsilon_r(\tau)$  – удельное число ошибок на одну машинную команду, оставшихся в системе после времени тестирования  $\tau$ :

$$\varepsilon_r(\tau) = \frac{E_T}{I_T} - \varepsilon_c(\tau);$$

$E_T$  – начальное число ошибок до начала тестирования;  $I_T$  – общее число машинных команд, которое предполагается постоянным в рамках этапа

тестирования;  $\varepsilon_c(\tau)$  – удельное количество обнаруженных ошибок на одну машинную команду в течение времени  $\tau$ .

Общее время тестирования  $\tau$  равно сумме времен прогонов:

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

Тогда

$$P(t) = \exp\left(C \cdot \left(\frac{E_T}{I_T} - \varepsilon_c(\tau)\right) \cdot t\right),$$

$$T_{cp} = \frac{1}{C \cdot \left(\frac{E_T}{I_T} - \varepsilon_c(\tau)\right)}.$$

Для определения начального количества ошибок полагают интенсивность появления ошибок постоянной и равной  $\lambda$ , которую можно найти как отношение суммы числа ошибок на каждом прогоне к общему времени тестирования в какой-то момент времени  $\tau_k$ :

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

Тогда

$$E_T = \frac{I_T \left( \lambda_{\tau_a} \cdot \varepsilon_c(\tau_a) - \lambda_{\tau_b} \cdot \varepsilon_c(\tau_b) \right)}{\lambda_{\tau_a} - \lambda_{\tau_b}},$$

$$C = \frac{\lambda_{\tau_a}}{\frac{E_T}{I_T} - \varepsilon_c(\tau_a)}.$$

Здесь  $\tau_a, \tau_b$  – два различных момента тестирования, которые выбираются произвольно с учетом требования

$$\varepsilon_{\bar{n}}(\tau_b) > \varepsilon_{\bar{n}}(\tau_a).$$

К преимуществам модели можно отнести то, что по ней можно определить все неизвестные параметры, т.е. нет необходимости обращаться к другим моделям, что сокращает время расчета надежности.

К недостаткам относится предположение, что при корректировке не вносятся новые ошибки, а это не всегда имеет место в реальных программах. Кроме того, в процессе тестирования необходимо регистрировать большое количество данных, необходимых для расчета по формулам этой модели.

## Модель Ла Падула

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

Надежность ПО в течение  $i$ -го этапа

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

где  $A$  – параметр роста;  $P(\infty) = \lim_{i \rightarrow \infty} P(i)$  – предельная надежность ПО. Эти неизвестные величины в данной модели предлагается найти из решения системы уравнений

$$\begin{cases} \sum_{i=1}^m \left( \frac{S_i - m_i}{S_i} - P(\infty) + \frac{A_i}{i} \right) = 0; \\ \sum_{i=1}^m \left( \frac{S_i - m_i}{S_i} - P(\infty) + \frac{A_i}{i} \right) \cdot \frac{1}{i} = 0, \end{cases}$$

где  $S_i$  – число тестов на  $i$ -м этапе;  $m_i$  – число отказов во время  $i$ -го этапа,  $i = 1, 2, \dots$

Модель является прогнозной и на основании данных тестирования позволяет предсказать вероятность безотказной работы программы на последующих этапах ее выполнения.

## Модель Джелински-Моранды

Основное положение, на котором базируется модель, заключается в том, что в процессе тестирования ПО значение интервалов времени тестирования между обнаружением двух ошибок имеет экспоненциальное распределение с интенсивностью отказов, пропорциональной числу еще не выявленных ошибок. Каждая обнаруженная ошибка устраняется, число оставшихся ошибок уменьшается на единицу.

Функция плотности распределения времени обнаружения  $i$ -й ошибки, отсчитываемого от момента выявления  $(i - 1)$ -й ошибки, имеет вид

$$f(t_i) = \lambda_i \cdot \exp(-\lambda_i t_i).$$

Согласно допущениям вероятность безотказной работы ПО

$$P(t_i) = \exp(-\lambda_i t_i),$$

где интенсивность отказов, пропорциональная числу еще не выявленных ошибок в программе, равна

$$\lambda_i = C \cdot (N - i + 1).$$

Здесь  $C$  – коэффициент пропорциональности;  $N$  – первоначальное количество ошибок программы.

Наиболее вероятные значения параметров  $C$ ,  $N$  определяются на основе данных, полученных при тестировании. Для этого фиксируют время выполнения программы до очередного отказа  $t_1, t_2, \dots, t_k$ . Затем решают систему уравнений

$$\begin{cases} \sum_{i=1}^k \frac{1}{(N - i + 1)} = \frac{K}{N + 1 - Q \cdot K}; \\ C = \frac{K}{A(N + 1 - Q \cdot K)}, \end{cases}$$

где  $Q = \frac{B}{A \cdot K}$ ,  $A = \sum_{i=1}^k t_i$ ,  $B = \sum_{i=1}^k i \cdot t_i$ .

Основным преимуществом модели является простота расчетов.

Недостаток этой модели состоит в том, что при неточном определении величины  $N$  интенсивность отказов программы может стать отрицательной, что приводит к бессмысленному результату. Кроме того, предполагается, что при исправлении обнаруженных ошибок не вносятся новые ошибки, что тоже не всегда выполняется.

### Модель Шика-Волвертона

Это модификация модели Джелински-Моранды для случая возникновения на рассматриваемом интервале более одной ошибки. При этом считается, что исправление ошибок производится лишь после истечения интервала времени, на котором они возникли.

В основе модели Шика-Волвертона лежит предположение, согласно которому частота ошибок пропорциональна не только количеству ошибок в программах, но и времени тестирования, т.е. вероятность обнаружения ошибок с течением времени возрастает. Интенсивность обнаружения ошибок  $\lambda_i$  предполагается постоянной в течение интервала времени  $t_i$  и пропорциональной числу ошибок, оставшихся в программе по истечении  $(i - 1)$ -го интервала; но она пропорциональна также и суммарному времени, уже затраченному на тестирование:



$$\lambda_i(t_i) = C(N - (i - 1))(T_{i-1} + t_i / 2),$$

где  $C$  – коэффициент пропорциональности;  $N$  – число ошибок, первоначально присутствующих в программе;  $i$  – число ошибок, обнаруженных в программе к моменту  $t_i$ ;  $T_{i-1}$  – суммарное время, затраченное на тестирование в течение  $(i - 1)$  этапов;  $t_i$  – среднее время работы программы в текущем интервале.

Остальные расчеты аналогичны расчетам модели Джелински-Моранды.

Для крупномасштабных разработок программ или проектов с продолжительным периодом отладки наилучший прогноз числа остаточных ошибок дает модель Шика-Волвертона.

### Модель Муса

Модель предполагает, что в процессе тестирования фиксируется время выполнения программы до очередного отказа. Но считается, что не всякая ошибка ПО может вызвать отказ, поэтому допускается обнаружение более одной ошибки при выполнении программы до возникновения очередного отказа.

Считается, что на протяжении всего жизненного цикла ПО может произойти  $M_0$  отказов и при этом будут выявлены все  $N_0$  ошибки, которые присутствовали в ПО до начала тестирования. Общее число отказов  $M_0$  связано с первоначальным числом ошибок  $N_0$  соотношением

$$N_0 = B \cdot M_0,$$

где  $B$  – коэффициент уменьшения числа ошибок.

После тестирования, за время которого зафиксировано  $m$  отказов и выявлено  $n$  ошибок, можно определить

$$B = \frac{n}{m}.$$

В модели различают два вида времени:

- суммарное время функционирования, которое учитывает чистое время тестирования до контрольного момента, когда проводится оценка надежности;

- оперативное время выполнения программы, планируемое от контрольного момента и далее при условии, что дальнейшего устранения ошибок не будет (время безотказной работы на этапе эксплуатации).

Для суммарного времени функционирования предполагается, что интенсивность отказов пропорциональна числу неустранимых ошибок и скорость изменения числа устранимых ошибок, измеряемая относительно суммарного времени, пропорциональна интенсивности отказов.

Один из основных показателей надежности, который рассчитывается по модели Муса, – средняя наработка на отказ. Этот показатель определяется как математическое ожидание временного интервала между последовательными отказами и связан с надежностью следующими формулами:

$$T_{ср} = \int_0^{\infty} P(t) dt,$$

но с учетом допущений модели

$$T_{ср} = T_0 \cdot \exp\left(\frac{c \cdot \tau}{M_0 T_0}\right).$$

Здесь  $\tau$  – время функционирования;  $T_0$  – начальная наработка на отказ;  $c$  – коэффициент сжатия тестов (равен времени испытаний).

Дополнительно устанавливается среднее число отказов в зависимости от времени функционирования  $\tau$ :

$$m = M_0 \left(1 - \exp\left(\frac{c \cdot \tau}{M_0 T_0}\right)\right).$$

Если интенсивность отказов постоянна (т.е. когда длительность интервалов между последовательными отказами имеет экспоненциальное распределение), то средняя наработка на отказ обратно пропорциональна интенсивности отказов.

К преимуществам модели можно отнести то, что нет необходимости фиксировать моменты отказов. В случае появления отказов ошибки регистрируются, а исправляются лишь по завершении этапа тестирования.

К недостаткам модели относится то, что для определения первоначального числа ошибок в программном обеспечении необходимо вести расчеты по другой модели, что приводит к дополнительным затратам времени.

## 4.2.2. Аналитические статические модели

### Модель Миллса

Использование этой модели предполагает необходимость перед началом тестирования искусственно вносить в программу некоторое количество известных ошибок. Ошибки вносятся случайным образом и фиксируются в протоколе искусственных ошибок. Специалист, проводящий тестирование, не знает ни количества, ни характера внесенных ошибок. Предполагается, что все ошибки (как естественные, так и искусственные) имеют равную вероятность быть найденными в процессе тестирования.

Программа тестируется в течение некоторого времени, и собирается статистика об обнаруженных ошибках.

Пусть после тестирования обнаружено  $n$  собственных ошибок программы и  $v$  искусственно внесенных ошибок. Тогда первоначальное число ошибок в программе  $N$  можно оценить по формуле Миллса

$$N = n \frac{S}{v},$$

где  $S$  – количество искусственно внесенных ошибок.

Вторая часть модели связана с проверкой гипотезы об  $N_0$ . Допустим, мы считаем, что в программе первоначально  $N_0$  ошибок. Вносим искусственно в программу  $S$  ошибок и тестируем ее до тех пор, пока все искусственно внесенные ошибки не будут обнаружены. Пусть при этом обнаружено  $n$  собственных ошибок программы. Вероятность, что в программе первоначально было  $N_0$  ошибок, можно рассчитать по соотношению

$$P = \begin{cases} 0, & \text{если } n > N_0; \\ \frac{S}{S + N_0 + 1}, & \text{если } n \leq N_0. \end{cases}$$

Эту формулу можно использовать только в случае, если обнаружены все  $S$  искусственно внесенных ошибок. Если же обнаружено только  $v$  искусственно внесенных ошибок, то применяют формулу

$$P = \begin{cases} 0, & \text{если } n > N_0; \\ \frac{C_S^{v-1}}{C_{S+N_0+1}^{N_0+v}}, & \text{если } n \leq N_0, \end{cases}$$

где  $C_n^m = \frac{n!}{m!(n-m)!}$  – число сочетаний из  $n$  элементов по  $m$ .

Достоинством модели Миллса является простота применяемого математического аппарата и наглядность. Применение этой модели для оценки надежности оказывает положительное психологическое воздействие на лиц, выполняющих тестирование, уже только тем, что они знают: в программу внесены ошибки.

Однако есть недостатки:

1) необходимость внесения искусственных ошибок (этот процесс плохо формализуем);

2) достаточно вольное допущение величины  $K$ , которое основывается исключительно на интуиции и опыте человека, производящего оценку, т.е. допускает большое влияние субъективного фактора.

### **Модель Нельсона**

Модель была разработана с учетом основных свойств машинных программ и практически не использует методы теории вероятности. Все приближения, принятые в этой модели, четко определены, и границы их применимости известны. Поскольку в основу модели Нельсона положены свойства программного обеспечения, она допускает развитие за счет более детального описания других аспектов надежности и может использоваться для расчета надежности программного обеспечения на всех этапах его жизненного цикла.

В модели предполагается, что область, которой могут принадлежать входные данные программы, разделена на  $k$  непересекающихся областей  $Z_i$ ,  $i = 1, 2, \dots, k$ . Пусть  $p_i$  – вероятность того, что для очередного выполнения программы будет выбран набор данных из области  $Z_i$ . Значения  $p_i$  определяются по статистике входных данных в реальных условиях работы программного обеспечения.

Пусть к моменту оценки надежности было выполнено  $M_i$  прогонов программного обеспечения на наборах данных из области  $Z_i$ , и  $m_i$  из этих прогонов закончились отказом.

Тогда надежность программного обеспечения оценивается по формуле

$$P = 1 - \sum_{i=1}^k \frac{m_i}{M_i} p_i.$$

Основным преимуществом этой модели является то, что она была специально создана для определения надежности программного обеспечения, а не исходила из теории надежности аппаратуры, как другие модели (кроме модели Миллса), поэтому может использоваться для расчета надежности программного обеспечения на всех этапах его жизненного цикла.

Но на ранних стадиях использовать эту модель не очень удобно, так как для объективной оценки надежности требуется большое число прогонов ПО.

### Модель Коркорэна

Модель предполагает наличие в программном обеспечении многих источников программных отказов, связанных с различными типами ошибок, и разную вероятность их появления. Аргументом модели является число прогонов программы  $M$ ,  $n_i$  ошибок  $i$ -го типа. При этом оценка надежности программного обеспечения имеет вид

$$P = \frac{M_0}{M} + \sum_{i=1}^k \frac{\delta_i (n_i - 1)}{M},$$

где  $M_0$  – число успешных прогонов программного обеспечения;  $M$  – общее число прогонов;  $k$  – априори известное число типов ошибок;  $\delta_i$  – коэффициент, определяемый следующим образом:

$$\delta_i = \begin{cases} p_i, & \text{если } n_i > 0, \\ 0, & \text{если } n_i = 0; \end{cases}$$

$p_i$  – вероятность выявления при тестировании ошибки  $i$ -го типа.

К преимуществам модели можно отнести то, что она учитывает существование в программном обеспечении нескольких источников ошибок, а также то, что расчет надежности с математической точки зрения проще, чем в других моделях.

К недостаткам можно отнести необходимость определения статистическим методом вероятности того, что для очередного прогона программы будет выбран набор данных из предполагаемой области, что затрудняет расчеты.

### 4.2.3. Эмпирические модели

Эмпирические модели основаны на анализе накопленной информации о функционировании ранее разработанных программ.

Наиболее простая эмпирическая модель связывает число ошибок в программном обеспечении с его объемом. Опытные данные свидетельствуют, что к началу системного тестирования в программном обеспечении на каждые 1000 операторов приходится примерно 10 ошибок. Уровень надежности программного обеспечения считается приемлемым для начала эксплуатации, если тому же объему операторов будет соответствовать одна ошибка.

Преимущество эмпирических моделей состоит в том, что они не содержат сложных формул и вычисления по ним просты.

К недостаткам эмпирических моделей относится то, что они очень грубы, весьма приблизительны. Кроме того, они не отражают динамики вычислительного процесса при эксплуатации программ.

#### Модель фирмы IBM

Фирма IBM использует эмпирическую модель, которая оценивает число ошибок в различных редакциях операционной системы:

$$N_0 = 23M_{10} + 2M_1,$$

где  $M_{10}$  – число модулей, потребовавших 10 и более исправлений;  $M_1$  – число модулей, в которых обнаружено меньше 10 ошибок.

Применяется также эмпирическая формула для оценки средней наработки программного обеспечения на отказ:

$$T_{cp} = \alpha \frac{V_{op}}{N_0},$$

где  $T_{cp}$  – средняя наработка программного обеспечения на отказ в часах;  $V_{op}$  – объем программы в операторах;  $N_0$  – число ошибок в программном обеспечении, оцененное по одной из приведенных выше моделей;  $\alpha$  – коэффициент, лежащий в диапазоне от 100 до 1000.

#### Модель Холстеда

Модель оценивает количество оставшихся в программе ошибок после окончания ее разработки как

$$N_0 = K_{HO} V_{ii} \log_2(\eta_1 + \eta_2),$$

где  $N_0$  – число ошибок в программе;  $K_{HO}$  – коэффициент пропорциональности;  $V_{OP}$  – число операторов в программе;  $\eta_1$  – число операторов в программном средстве;  $\eta_2$  – число операндов в программном средстве.

### 4.3. Обеспечение надежности программных средств

#### 4.3.1. Общие принципы обеспечения надежности

Рассмотрим теперь общие принципы обеспечения надежности ПО. В технике известны четыре подхода к обеспечению надежности:

- предупреждение ошибок;
- самообнаружение ошибок;
- самоисправление ошибок;
- обеспечение устойчивости к ошибкам.

Подходы самообнаружения, самоисправления ошибок и обеспечения устойчивости к ошибкам связаны с организацией самих продуктов технологии, в нашем случае – программ. Они учитывают возможность ошибки в программах.

*Самообнаружение* ошибки в программе означает, что программа содержит средства обнаружения отказа в процессе ее выполнения.

*Самоисправление* ошибки в программе означает не только обнаружение отказа в процессе ее выполнения, но и исправление последствий этого отказа, для чего в программе должны иметься соответствующие средства.

Обеспечение *устойчивости* программы к ошибкам означает, что в программе содержатся средства, позволяющие локализовать область влияния отказа программы либо уменьшить его неприятные последствия, а иногда предотвратить катастрофические последствия отказа.

Однако эти подходы используются весьма редко (чаще используется обеспечение устойчивости к ошибкам). Связано это, во-первых, с тем, что многие простые методы, используемые в технике в рамках этих подходов, неприменимы в программировании, например дублирование отдельных блоков и устройств (выполнение двух копий одной и той же программы всегда будет приводить к одинаковому эффекту – правильному или непра-

вильному). Во-вторых, добавление в программу дополнительных фрагментов приводит к ее усложнению (иногда значительному), что в какой-то мере мешает методам предупреждения ошибок.

Рассмотрим более подробно первый подход – предупреждение ошибок.

#### **4.3.2. Предупреждение ошибок**

Цель подхода *предупреждения ошибок* – не допустить ошибок в готовых продуктах, в нашем случае – в ПО. Рассмотрение природы ошибок при разработке ПО позволяет для достижения этой цели сконцентрировать внимание на следующих вопросах:

- борьба со сложностью;
- обеспечение точности перевода;
- преодоление барьера между пользователем и разработчиком;
- обеспечение контроля принимаемых решений.

Этот подход связан с организацией процессов разработки ПО, т.е. с технологией программирования. И хотя гарантировать отсутствие ошибок в ПО невозможно, но в рамках этого подхода можно достигнуть приемлемого уровня надежности ПО.

##### **Методы борьбы со сложностью**

Известны два общих метода борьбы со сложностью систем:

- обеспечение независимости компонентов системы;
- использование в системах иерархических структур.

Обеспечение независимости компонентов означает разбиение системы на такие части, между которыми должно остаться по возможности меньше связей. Одним из воплощений этого метода является модульное программирование.

Использование в системах иерархических структур позволяет локализовать связи между компонентами, допуская их лишь между компонентами, принадлежащими смежным уровням иерархии. Этот метод, по существу, означает разбиение большой системы на подсистемы, образующие малые системы. Здесь существенно используется способность человека к абстрагированию.



## **Обеспечение точности перевода**

Обеспечение точности перевода направлено на достижение однозначности интерпретации документов различными разработчиками, а также пользователями программного средства. Для этого необходимо при переводе придерживаться определенной дисциплины. Рекомендуется придерживаться общей дисциплины решения задач, рассматривая перевод как решение задачи. В соответствии с этим весь процесс перевода можно разбить на следующие этапы:

- поймите задачу;
- составьте план (включая цели и методы решения);
- выполните план (проверяя правильность каждого шага);
- проанализируйте полученное решение.

## **Преодоление барьера между пользователем и разработчиком**

ПО должно выполнять то, что пользователю разумно ожидать от него. Для этого разработчикам необходимо правильно понять, во-первых, чего хочет пользователь, во-вторых, его уровень подготовки и окружающую его обстановку. Ясное описание соответствующей сферы деятельности пользователя или интересующей его проблемной области во многом облегчает достижение разработчиками этой цели. При разработке ПО следует привлекать пользователя для участия в процессах принятия решений, а также тщательно освоить особенности его работы (лучше всего – побывать в его «шкуре»).

## **Контроль принимаемых решений**

Обязательным шагом в каждом процессе (этапе) разработки ПО должна быть проверка правильности принятых решений. Это позволит обнаруживать и исправлять ошибку на самой ранней стадии после ее возникновения, что, во-первых, существенно снижает стоимость ее исправления и, во-вторых, повышает вероятность правильного ее устранения.

С учетом специфики разработки ПС необходимо везде, где это возможно, применять:

- смежный контроль,
- сочетание как статических, так и динамических методов контроля.

Смежный контроль означает проверку полученного документа лицами, не участвующими в его разработке, с двух сторон: во-первых, со стороны автора исходного для контролируемого процесса документа и, во-вторых, лицами, которые будут использовать полученный документ в качестве исходного в последующих технологических процессах. Такой контроль позволяет обеспечивать однозначность интерпретации полученного документа.

Сочетание статических и динамических методов контроля означает, что нужно не только контролировать документ как таковой, но и проверять, какой процесс обработки данных он описывает. Это отражает одну из специфических особенностей ПС (статическая форма, динамическое содержание).

### **4.3.3. Обеспечение примитивов надежности программного средства**

#### **Обеспечение завершенности**

Завершенность ПО является общим примитивом качества ПО для выражения и функциональности, и надежности ПО, причем для функциональности она является единственным примитивом.

Функциональность ПО определяется его функциональной спецификацией. Завершенность ПС как примитив его качества является мерой того, в какой степени эта спецификация реализована в разрабатываемом ПО. Обеспечение этого примитива в полном объеме означает реализацию каждой из функций, определенной в функциональной спецификации, со всеми указанными там деталями и особенностями.

Однако в спецификации качества ПО может быть определено несколько уровней реализации функциональности ПО:

- может быть определена некоторая упрощенная (начальная или стартовая) версия, которая должна быть реализована в первую очередь;
- могут быть также определены и несколько промежуточных версий;
- может быть полная реализация функциональной спецификации.

В первых двух случаях возникает дополнительная технологическая задача: организация наращивания функциональности ПО.

Здесь важно отметить, что разработка упрощенной версии ПО не есть разработка его *прототипа*. Прототип разрабатывается для того, чтобы лучше понять условия применения будущего ПС, уточнить его внешнее описание. Прототип рассчитан на избранных пользователей и поэтому может сильно отличаться от требуемого ПС не только выполняемыми функциями, но и особенностями пользовательского интерфейса. Упрощенная же версия разрабатываемого ПО должна быть рассчитана на *практически полезное* применение любыми пользователями, для которых предназначено это ПО.

Поэтому главный принцип обеспечения функциональности такого ПО заключается в том, чтобы с самого начала разрабатывать ПС таким образом, как будто требуется ПО в полном объеме, до тех пор, когда разработчики будут иметь дело непосредственно с теми частями или деталями ПО, реализацию которых можно отложить в соответствии со спецификацией его качества. Тем самым и внешнее описание, и описание архитектуры ПО должны быть разработаны в полном объеме. Можно отложить лишь реализацию тех программных подсистем (определенных в архитектуре разрабатываемого ПО), функционирование которых не требуется в начальной версии этого ПО.

Реализацию же самих программных подсистем лучше всего производить методом целенаправленной конструктивной реализации, оставляя в начальной версии ПС подходящие имитаторы тех программных модулей, функционирование которых в этой версии не требуется. Допустима также упрощенная реализация некоторых программных модулей, опускающая реализацию некоторых деталей соответствующих функций. Однако такие модули с технологической точки зрения лучше рассматривать как своеобразные их имитаторы (хотя и далеко продвинутые).

Достигнутый при обеспечении функциональности ПО уровень его завершенности на самом деле может быть не таким, как ожидалось, из-за ошибок, оставшихся в этом ПО. Можно лишь говорить, что требуемая завершенность достигнута с некоторой вероятностью, определяемой объемом и качеством проведенного тестирования. Для того чтобы повысить эту вероятность, необходимо продолжить тестирование и отладку ПО.

## **Обеспечение точности**

Обеспечение этого примитива качества связано с действиями над значениями вещественных типов (точнее говоря, над значениями, представляемыми с некоторой погрешностью). Обеспечить требуемую точность при вычислении значения той или иной функции – значит получить это значение с погрешностью, не выходящей за рамки заданных границ.

Видами погрешности, методами их оценки и методами достижения требуемой точности (так называемыми *приближенными вычислениями*) занимается вычислительная математика.

Здесь лишь обратим внимание на некоторую структуру погрешности: погрешность вычисленного значения (*полная погрешность*) зависит от:

- *погрешности используемого метода* вычисления (в которую мы включаем и неточность используемой модели);
- *погрешности представления используемых данных* (так называемой неустранимой погрешности);
- *погрешности округления* (неточности выполнения используемых в методе операций).

## **Обеспечение автономности**

Вопрос об автономности программного средства решается путем принятия решения о возможности использования в разрабатываемом ПО какого-либо подходящего базового программного обеспечения.

Надежность имеющегося в распоряжении разработчиков базового программного обеспечения для целевой информационной системы может не отвечать требованиям к надежности разрабатываемого ПО. Поэтому от использования такого программного обеспечения приходится отказываться, а его функции в требуемом объеме приходится реализовывать в рамках разрабатываемого ПО.

Такое решение может быть принято уже в процессе разработки технического задания, иногда – на этапе рабочего проекта ИС.

## **Обеспечение устойчивости**

Этот примитив качества ПО обеспечивается с помощью так называемого *защитного программирования*. Вообще говоря, защитное про-

граммирование применяется при программировании модуля для повышения надежности ПО в более широком смысле.

Как утверждает Майерс, «защитное программирование основано на важной предпосылке: худшее, что может сделать модуль, – это принять неправильные входные данные и затем вернуть неверный, но правдоподобный результат». Для того чтобы этого избежать, в текст модуля включают проверки его входных и выходных данных на их корректность в соответствии со спецификацией этого модуля, в частности должны быть проверены выполнение ограничений на входные и выходные данные и соотношения между ними, указанные в спецификации модуля. В случае отрицательного результата проверки возбуждается соответствующая исключительная ситуация.

Для обработки исключительных ситуаций в конец этого модуля включаются фрагменты второго рода – обработчики соответствующих исключительных ситуаций. Эти обработчики, помимо выдачи необходимой диагностической информации, могут принять меры либо по исключению ошибки в данных (например, потребовать их повторного ввода), либо по ослаблению влияния ошибки (например, во избежание поломки устройств, управляемых с помощью данного ПО, при аварийном прекращении выполнения программы осуществляют мягкую их остановку).

Применение защитного программирования модулей приводит к снижению эффективности ПО как по времени, так и по памяти. Поэтому необходимо разумно регулировать степень применения защитного программирования в зависимости от требований к надежности и эффективности ПО, сформулированных в спецификации качества разрабатываемого ПО.

Входные данные разрабатываемого модуля могут поступать как непосредственно от пользователя, так и от других модулей. Наиболее употребительным случаем применения защитного программирования является применение его для первой группы данных, что и означает реализацию устойчивости ПС. Это нужно делать всегда, когда в спецификации качества ПО имеется требование об обеспечении устойчивости ПО.

Применение защитного программирования для второй группы входных данных означает попытку обнаружить ошибку в других модулях во время выполнения разрабатываемого модуля, а для выходных данных разрабатываемого модуля – попытку обнаружить ошибку в самом этом моду-

ле во время его выполнения. По существу, это означает частичное воплощение подхода самообнаружения ошибок для обеспечения надежности ПС. Этот случай защитного программирования применяется крайне редко – только в том случае, когда требования к надежности ПС чрезвычайно высоки.

### **Обеспечение защищенности**

Различают следующие виды защиты ПС от искажения информации:

- защита от сбоев аппаратуры;
- защита от влияния «чужой» программы;
- защита от отказов «своей» программы;
- защита от ошибок оператора (пользователя);
- защита от несанкционированного доступа;
- защита от защиты.

**Защита от сбоев аппаратуры.** В настоящее время этот вид защиты является не очень злободневной задачей (с учетом уровня достигнутой надежности компьютеров). Но все же полезно знать ее решение. Данный вид защиты обеспечивается организацией так называемых двойных или тройных просчетов.

Для этого весь процесс обработки данных, определяемый ПО, разбивается по времени на интервалы так называемыми опорными точками. Длина интервала не должна превосходить половины среднего времени безотказной работы компьютера. В начале каждого такого интервала во вторичную память записывается с некоторой контрольной суммой копия состояния изменяемой в этом процессе памяти (опорная точка).

Для того чтобы убедиться, что обработка данных от одной опорной точки до следующей (т.е. один просчет) произведена правильно (без сбоев компьютера), производится два таких просчета.

После первого просчета вычисляется и запоминается указанная контрольная сумма, а затем восстанавливается состояние памяти по опорной точке и делается второй просчет.

После второго просчета снова вычисляется указанная контрольная сумма, которая затем сравнивается с контрольной суммой первого просчета.

Если эти две контрольные суммы совпадают, второй просчет считается правильным, в противном случае контрольная сумма второго просче-

та также запоминается и производится третий просчет (с предварительным восстановлением состояния памяти по опорной точке). Если контрольная сумма третьего просчета совпадет с контрольной суммой одного из первых двух просчетов, то третий просчет считается правильным, в противном случае требуется инженерная проверка компьютера.

**Защита от влияния «чужой» программы.** При появлении мультипрограммного режима работы компьютера в его памяти может одновременно находиться в стадии выполнения несколько программ, попеременно получающих управление в результате возникающих прерываний (так называемое квазипараллельное выполнение программ). Одна из таких программ (обычно – операционная система) занимается обработкой прерываний и управлением мультипрограммным режимом. Здесь под «чужой» программой понимается программа (или какой-либо программный фрагмент), выполняемая параллельно (или квазипараллельно) по отношению к защищаемой программе (или ее фрагменту). Этот вид защиты должен обеспечить независимость эффекта выполнения защищаемой программы от программ, выполняемых параллельно с ней, и относится, прежде всего, к функциям операционных систем.

Различают две разновидности этой защиты:

- защита от отказов «чужой» программы,
- защита от злонамеренного влияния «чужой» программы.

*Защита от отказов «чужой» программы* означает, что на выполнение функций защищаемой программой не будут влиять отказы (проявления ошибок), возникающие в параллельно выполняемых программах.

Для того чтобы управляющая программа (операционная система) могла обеспечить защиту себя и других программ от такого влияния, аппаратура компьютера должна реализовывать следующие возможности:

- защиту памяти,
- два режима функционирования компьютера: привилегированный и рабочий (пользовательский),
- два вида операций: привилегированные и обычные,
- корректную реализацию прерываний и начального включения компьютера,
- временное прерывание.

Защита памяти означает возможность программным путем задавать для каждой программы недоступные для нее участки памяти. В привилегированном режиме могут выполняться любые операции (как ординарные, так и привилегированные), а в рабочем режиме – только ординарные. Попытка выполнить привилегированную операцию, а также обратиться к защищенной памяти в рабочем режиме вызывает соответствующее прерывание.

К привилегированным операциям относятся операции изменения защиты памяти и режима функционирования, а также доступа к внешней информационной среде.

Корректная реализация прерываний и начального включения компьютера означает обязательную установку привилегированного режима и отмену защиты памяти. В этих условиях управляющая программа (операционная система) может полностью защитить себя от влияния отказов других программ. Для этого достаточно, чтобы:

- все точки передачи управления при начальном включении компьютера и при прерываниях принадлежали этой программе;
- она не позволяла никакой другой программе работать в привилегированном режиме (при передаче управления любой другой программе должен включаться только рабочий режим);
- она полностью защищала свою память (содержащую, в частности, всю ее управляющую информацию, включая так называемые вектора прерываний) от других программ.

Тогда никто не помешает ей выполнять любые реализованные в ней функции защиты других программ (в том числе и доступа к внешней информационной среде). Для облегчения решения этой задачи часть такой программы помещается в постоянную память.

Наличие временного прерывания позволяет управляющей программе защититься от заикливания в других программах (без такого прерывания она могла бы просто лишиться возможности управлять).

*Защита от злонамеренного влияния «чужих» программ* означает, что изменение внешней информационной среды, предоставленной защищаемой программе, со стороны другой, параллельно выполняемой программы, будет невозможно или сильно затруднено без ведома защищаемой программы. Для этого операционная система должна обеспечить подходящий контроль доступа к внешней информационной среде. Необходимым условием обеспечения такого контроля является обеспечение защиты от злона-



меренного влияния «чужих» программ хотя бы самой операционной системы. В противном случае такой контроль можно было бы обойти путем изменения операционной системы со стороны «злонамеренной» программы.

Этот вид защиты включает, в частности, и защиту от компьютерных вирусов, под которыми понимают фрагменты программ, способные в процессе своего выполнения внедряться (копироваться) в другие программы (или в отдельные программные фрагменты). Компьютерные вирусы, обладая способностью к размножению (к внедрению в другие программы), при определенных условиях вызывают изменение эффекта выполнения «зараженной» программы, что может привести к серьезным деструктивным изменениям ее внешней информационной среды. Операционная система, будучи защищенной от влияния «чужих» программ, может ограничить доступ к программным фрагментам, хранящимся во внешней информационной среде.

**Защита от ошибок пользователя.** Здесь идет речь не об ошибочных данных, поступающих от пользователя ПО, – защита от них связана с обеспечением устойчивости ПО, – а о действиях пользователя, приводящих к деструктивному изменению состояния внешней информационной среды ПС, несмотря на корректность используемых при этом данных.

Защита от таких действий частично обеспечивается выдачей предупредительных сообщений о попытках изменить состояние внешней информационной среды ПО с требованием подтверждения этих действий.

Для случаев же, когда такие ошибки совершаются, может быть предусмотрена возможность восстановления состояния отдельных компонентов внешней информационной среды ПС на определенные моменты времени. Такая возможность базируется на ведении (формировании) архива состояний (или изменений состояния) внешней информационной среды.

**Защита от несанкционированного доступа.** Каждому пользователю ПО предоставляются определенные информационные и процедурные ресурсы (услуги), причем у разных пользователей ПО предоставленные им ресурсы могут отличаться, иногда очень существенно. Этот вид защиты должен обеспечить, чтобы каждый пользователь ПО мог использовать только то, что ему предоставлено (санкционировано). Для этого ПО в своей внешней информационной среде может хранить информацию о своих пользователях и предоставленных им правах использования ресурсов, а

также предоставлять пользователям определенные возможности формирования этой информации.

Защита от несанкционированного доступа к ресурсам ПС осуществляется с помощью *паролей*. При этом предполагается, что каждый пользователь знает только свой пароль, зарегистрированный в ПО этим пользователем. Для доступа к выделенным ему ресурсам он должен предъявить ПО свой пароль.

Различают две разновидности такой защиты:

- простая защита от несанкционированного доступа,
- защита от взлома защиты.

*Простая защита* от несанкционированного доступа обеспечивает защиту от использования ресурсов ПС пользователем, которому не предоставлены соответствующие права доступа или который указал неправильный пароль. При этом предполагается, что пользователь, получив отказ в доступе к интересующим ему ресурсам, не будет предпринимать попыток каким-либо несанкционированным образом обойти или преодолеть эту защиту. Поэтому этот вид защиты может применяться и в ПС, которые базируются на операционной системе, не обеспечивающей полную защиту от влияния «чужих» программ.

*Защита от взлома защиты* – такая разновидность защиты от несанкционированного доступа, которая существенно затрудняет преодоление этой защиты. Это связано с тем, что в отдельных случаях могут быть предприняты настойчивые попытки взломать защиту от несанкционированного доступа, если защищаемые ресурсы представляют для кого-то чрезвычайную ценность. Для такого случая приходится предпринимать дополнительные меры защиты. Во-первых, необходимо обеспечить, чтобы такую защиту нельзя было обойти, т.е. должна действовать защита от влияния «чужих» программ. Во-вторых, необходимо усилить простую защиту от несанкционированного доступа использованием в ПС специальных программистских приемов, в достаточной степени затрудняющих подбор подходящего пароля или его вычисление по информации, хранящейся во внешней информационной среде ПС.

*Защита от защиты*. Защита от несанкционированного доступа может создать нежелательную ситуацию для самого владельца прав доступа к ресурсам ПО – он не сможет воспользоваться этими правами, если за-

будет (или потеряет) свой пароль («ключ»). Для защиты интересов пользователя в таких ситуациях и предназначена защита от защиты. Для обеспечения такой защиты ПО должно иметь привилегированного пользователя, называемого администратором ПО. Администратор ПО должен, в частности, отвечать за функционирование защиты ПО: именно он должен формировать контингент пользователей данного ПО, предоставляя каждому из этих пользователей определенные права доступа к ресурсам ПО. В ПО должна быть привилегированная операция (для администратора), позволяющая временно снимать защиту от несанкционированного доступа для пользователя с целью фиксации требуемого пароля («замка»).

**Защита от отказов «своей» программы.** Обеспечивается технологией программирования, описанию которой посвящено достаточно много источников. В данном пособии будут рассмотрены два аспекта технологии программирования: требования к стилю программирования и тестирование ПО, представляющие, с точки зрения авторов, достаточно важные составляющие обеспечения защиты от программных отказов и в целом надежности ПО.

#### 4.3.4. Оценки стиля программирования

Стиль программирования – это «отпечаток», налагаемый на программу программистом со свойственными ему методами работы.

В языке программирования<sup>1</sup>, так же как в обычном языке, на котором человек общается с себе подобными в жизни, определены алфавит, лексика, синтаксис и т.д., т.е. определены некоторые правила построения фраз (операторов), используя которые, человек может общаться на этом языке с другими людьми (и даже с ЭВМ). Психология и лингвистика констатируют, что каждый человек, знающий некоторый язык (умеющий читать на этом языке и писать на этом языке), также имеет некоторый стиль, который является характерным для данного человека и который проявляется, когда человек пишет на этом языке. Стиль зависит от языка, является атрибутом (свойством) конкретного человека и характеризует конкретного человека.

---

<sup>1</sup> Язык программирования – формальная знаковая система, служащая общению человека с ЭВМ и являющаяся важным средством профессионального общения людей.

Всякая программа имеет свой стиль. Каждому программисту свойственны стилевые особенности. Если программист получает программу, которая не соответствует его стилю, возникает стилевой конфликт. В результате программист отвергает программу, не желая приспособливаться к ее стилю. При этом он начинает «изобретать велосипед», т.е. переписывать программу заново, заявляя, что «легче новую написать, чем в этой разобратся».

Для оценки эффективности стиля программирования  $\varepsilon$  может быть использована, например, мера, вычисляемая как отношение количества машинных команд  $k$ , в которые транслируется программа (процедура, функция), к количеству  $n$  исполняемых операторов языка высокого уровня, на котором написана программа:

$$\varepsilon = k / n.$$

Количество машинных команд, в которые транслируется программа, можно просто получить в тех системах программирования, которые позволяют видеть ассемблерный листинг программы. Естественно, модули из подключаемых библиотек не должны при этом учитываться.

В максимальной степени значение качества стиля программирования проявляется в процессе эксплуатации программного обеспечения на этапе сопровождения, когда возникает необходимость доработок и исправления ошибок. Легкий, красивый стиль программирования способствует увеличению времени жизни программ. Это становится особенно важно в последние годы в связи с принижением значения серии ГОСТов ЕСПД, регламентирующих программную документацию.

Далее изложены методы оценки мер модульности, структурности кодирования и читабельности программ, определяющих качество стиля программирования.

Далее везде, где написано «процедура», следует читать «процедура и/или функция».

### **Оценка структурированности программы**

Оценка структурированности программы – это оценка того, насколько программа соответствует требованиям структурного программирования.

Определение (из классики): «Структурное программирование сосредотачивается на одном из наиболее подверженных ошибкам факторов

программирования – логике программы – и включает три главные составляющие:

Проектирование сверху вниз.

Модульное программирование.

Структурное кодирование».

Поэтому естественно оценку структурированности программы определять с помощью оценок модульности программы и структурности кодирования. К сожалению, «проектирование сверху вниз» в силу неформализованности процесса в настоящее время не может быть каким-либо образом измерено.

### **Оценка модульности программы**

В языках программирования (в частности, в языках, используемых в учебном процессе: Паскале, Бейсике, Фортране, Си и т.д.) широко используется аппарат процедур и функций. Причем процедуры и функции могут быть как внешними (относительно главного модуля программы или относительно какой-либо процедуры), так и внутренними (вложенными).

Модульность программы подразумевает, что программа разбита на процедуры разумной длины, каждая из которых выполняет некоторую функцию, определяемую алгоритмом программы.

Вообще говоря, модульность программы имеет две стороны (формальную и смысловую):

– разбиение программы на модули (процедуры и функции) разумной длины;

– разбиение программы на модули должно иметь смысл.

К сожалению, измерять смысловую сторону модульности человечество пока ещё не научилось. Поэтому ограничимся измерением формальной стороны.

При определении длины процедуры будем учитывать только операторы самой процедуры, в терминах Паскаля – блок Begin-End. Внутренние (вложенные) процедуры будем считать как отдельные процедуры.

Определим «нормальную длину процедуры». Положим, что «нормальная длина процедуры» составляет 50 исполняемых операторов, т.е. не учитываем строки комментария, операторы объявлений констант, типов и переменных. Это число 50, с одной стороны, зависит от языка программи-

рования, с другой – определяется естественным стремлением человека к «обозримости объекта», в частности к расположению процедуры на одном листе бумаги формата А4. Предлагается считать константу «нормальной длины процедуры» равной:

- для Паскаля – 40-60;
- для Фортрана – 30-50;
- для Си, С++ – 15-30;
- для Бейсика – 30-50;
- для VB и VBA – 10-20;
- для Delphi – 20-30.

Оценим среднюю длину процедуры  $l_{cp}$  как отношение суммы длин всех процедур  $l_i$ , включая блок main, к количеству  $N$  всех процедур, включая блок main:

$$l_{\bar{n}\bar{d}} = \frac{\sum_{i=1}^N l_i}{N}.$$

Однако одной этой величины для оценки структурированности программы недостаточно.

Контрпример: пусть в программе  $m$  логических процедур по 100 операторов каждая и  $k$  вычислительных процедур длиной по 5 операторов. Тогда:

- а) при  $m = 2$ ,  $k = 98$  имеем  $l_{cp} = 6,9$  оператора;
- б) при  $m = 10$ ,  $k = 10$  имеем  $l_{cp} = 52,2$  оператора;
- в) пусть в программе 10 процедур по 5 операторов и одна процедура длиной 500 операторов; тогда имеем  $l_{cp} = 50$  операторов.

То есть приходим к неверным суждениям о структурированности программы. И если случаи а) и б) ещё можно «понять», то случай в) чреват последствиями.

Поэтому введем меру модульности  $\mu$  программы как разность между 1 и суммой двух отношений, где первое есть отношение абсолютной разницы средней длины процедуры  $l_{cp}$  и константы «нормальной длины процедуры» к 100, а второе представляет собой отношение длины  $l_{pr.max}$  самой большой процедуры к удвоенному произведению общей длины  $l_{общ}$  программы (общему количеству исполняемых операторов в программе):

$$\mu = 1 - \left( \frac{|l_{\bar{n}\bar{d}} - 50|}{100} + \frac{l_{pr.max}}{2 l_{\bar{a}\bar{u}}} \right),$$

где  $l_{cp}$  определяется ранее.

При отрицательных значениях  $\mu$  разбиение программы на модули неудовлетворительно.

### **Оценка структурности кодирования**

Будем учитывать требования структурного кодирования суммированием штрафных баллов по каждому случаю нарушения:

1) оператор GOTO используется для выхода из процедуры – 5 баллов на каждый случай;

2) оператор GOTO используется для перехода назад по программе – 2 балла;

3) другие случаи использования оператора GOTO (выход из цикла вперед по программе, использование в условном операторе) – 1 балл;

4) наличие в строке более одного оператора – 0,25 балла;

5) процедуры не отделены друг от друга строками комментария (минимум две строки комментария на процедуру) – 1 балл;

6) текст программы не сформатирован, т.е. не используются сдвиги вправо для обозначения блоков программы – 5 баллов;

7) отсутствуют начальные блоки комментариев в процедурах – 3 балла;

8) не комментированы циклы, условные операторы, операторы END (операторы «;» для Си), операторы перехода – 0,25 балла;

9) метки GOTO (а также метки оператора format для Фортрана) определены не в возрастающем порядке – 1 балл;

10) перекрытие GOTO-диапазонов – 3 балла;

11) метки оператора format (для Фортрана) расположены в диапазоне GOTO – 1 балл;

12) глубина вложенности условных операторов и операторов циклов не превышает 10; каждый уровень вложенности сверх указанного – 0,5 балла;

13) читатель может добавить дополнительные требования.

Оценку структурности кодирования  $\beta$  определим следующим образом. Предположим, что читатель новых требований не добавил. Пусть размерность вектора «нарушений»  $X$  равна  $L$  (в данном случае  $L = 12$ ). Тогда оценка  $\beta$  равна частному от деления суммы координат вектора  $X$  на удвоенную норму вектора  $X$ :

$$\beta = \frac{\sum_{i=1}^L x_i}{2 \| X \|}.$$

Если  $\beta > 1$ , тогда оценка структурности кодирования неудовлетворительная.

### **Оценка структурированности программы**

Пусть известны  $\mu$  – мера модульности программы и  $\beta$  – мера структурности кодирования программы. Тогда оценку структурированности программы определим как произведение меры модульности программы и меры структурности кодирования программы:

$$\alpha = \mu \beta.$$

Если программа имеет оценку  $\alpha \ll 1$ , то её структурированность хорошая. Если  $\alpha < 1$ , то её структурированность удовлетворительная.

### **Оценка читабельности программы**

Определим требования, которым должен удовлетворять исходный текст читабельной программы, и будем начислять штрафные баллы за каждый случай нарушения этих требований:

1. Структура программы выдержана (т.е. явно выделены части программы):

Начальный блок комментария.

Объявления:

- констант,
- типов,
- переменных,
- внешних процедур,
- внутренних процедур.

Внутренние процедуры.

Блок begin-end.

Штраф – 1 балл за каждый случай нарушения.

Пример нарушения. Константы объявлены внутри исполняемого оператора:  
`RESET(F,'DK:MNOZXI.PAS','PAS',I);`

2. Наличие начального блока комментария, описывающего наименование программы, принадлежность (в состав чего входит), функциональ-



ное назначение, метод решения (источник алгоритма), входные и выходные данные, ограничения и условия применения, побочные эффекты, содержащего дату разработки, автора, дату последней корректировки, версию. Для процедур и функций дополнительно описывается использование глобальных переменных. За отсутствие каждого перечисленного пункта – штраф 0,5 балла.

3. Константы, типы и переменные описаны в комментариях. За каждый неописанный случай – штраф 0,1 балла.

4. Текст отформатирован: логические блоки выделены сдвигами операторов вправо.

Каждое нарушение форматирования – 0,05 балла, но в сумме не более 3 баллов.

Грубые нарушения данного пункта:

- все операторы набиты с первой колонки – 3 балла;
- «книжный текст»: сверхплотное расположение операторов (несколько на каждой строке) – 3 балла.

5. Процедуры отделены друг от друга двумя строками комментария – 0 баллов, одной строкой – 1 балл, комментарий отсутствует – 2 балла.

6. Один оператор на строке; за каждый случай нарушения – 0,1 балла.

7. Все логические блоки (операторы if, case, switch, goto, exit, операторы циклов и т.д.) прокомментированы. Невыполнение требования – 0,25 балла за каждый случай.

8. Прокомментированы операторы end /\* конец чего? \*/ (для языка Си – ";" ). Невыполнение требования – 0,1 балла.

Поясняющий пример: не прокомментированы два оператора end; – это два случая.

9. Комментарии располагаются сбоку (справа) от исполняемых (комментируемых операторов), а не вперемежку с ними. Невыполнение требований пункта – до 0,5 балла.

10. Программа удовлетворяет требованию структурированности (см. выше). Оценка выполнения требования в баллах равна трёхкратной оценке структурированности (3β).

11. Константы, типы, переменные, имена процедур (функций) имеют осмысленные имена. Оценка невыполнения – в пределах 0-0,5 балла.

12. Если переменные используются в программе для нескольких целей – 1 балл. Если случай «перенацеливания» не комментирован – дополнительно 0,5 балла за каждый случай.

13. Использование булевских переменных (типа BOOLEAN): true – включено, false – выключено; если переключатель моделируется переменной другого типа, то аналогично: 1 – включено, а 0 – выключено. За каждое нарушение – 1 балл.

14. Не «изобретается велосипед»: используются стандартные функции и подпрограммы из библиотек. Невыполнение – 1 балл.

15. Если алгоритм программы формирует во входной области программы особые точки, то это прокомментировано. Невыполнение требования – 3 балла.

Поясняющий пример: входная область 0 – 65535. Входные данные в диапазоне от 0 до 100 обрабатываются программой как действительные числа, остальные данные из этой входной области обрабатываются как целые числа. Точка  $x = 100$  – особая точка входной области.

16. Имеются комментарии, позволяющие оценить влияние на данный модуль изменений, вносимых в другие модули (например, прокомментировано каждое использование глобальных переменных). Невыполнение требования – до 1 балла.

17. Если предполагается использование программы в существенно различных условиях, требующих изменений в тексте, например раскомментировать (закомментировать) что-то, то необходимые изменения и их место – комментированы. Невыполнение требования – до 2 баллов.

18. Читатель может добавить дополнительные требования.

Оценку читабельности программы определим следующим образом. Пункты 1-10 достаточно просто могут быть реализованы программно. Пусть  $Y$  есть  $K$ -мерный (в данном случае  $K = 10$ ) вектор «читабельности». Тогда оценка  $\gamma$  читабельности исходного текста программы равна частному от деления суммы координат вектора  $X$  на удвоенную норму вектора  $X$ :

$$\gamma = \frac{\sum_{i=1}^K x_i}{2 \|X\|}.$$

Если  $\gamma > 1$ , тогда оценка читабельности исходного текста программы неудовлетворительная.

### 4.3.5. Тестирование программного обеспечения

#### Основные понятия

Тестирование является одним из наиболее устоявшихся способов обеспечения качества разработки программного обеспечения и входит в набор эффективных средств современной системы обеспечения качества программного продукта.

С технической точки зрения тестирование заключается в выполнении приложения на некотором множестве исходных данных и сверке получаемых результатов с заранее известными (эталонными) с целью установить соответствие различных свойств и характеристик приложения заказанным свойствам. Как одна из основных фаз процесса разработки программного продукта (Дизайн приложения – Разработка кода – Тестирование) тестирование характеризуется достаточно большим вкладом в суммарную трудоемкость разработки продукта. Широко известна оценка распределения трудоемкости между фазами создания программного продукта: 40–20–40 % (Разработка спецификации – Кодирование – Тестирование).

*Тестирование* – это:

- процесс выполнения ПО системы или компонента в условиях анализа или записи получаемых результатов с целью проверки (оценки) некоторых свойств тестируемого объекта;
- процесс анализа пункта требований к ПО с целью фиксации различий между существующим состоянием ПО и требуемым при экспериментальной проверке соответствующего пункта требований;
- контролируемое выполнение программы на конечном множестве тестовых данных и анализ результатов этого выполнения для поиска ошибок.

Последовательность действий, необходимых для тестирования:

- спецификация программы;
- разработка тестов;
- анализ тестовых случаев;
- выполнение тестовых случаев;
- оценка результатов выполнения программы на тестах.

Реализация тестирования разделяется на три этапа:

- создание тестового набора (test suite) путем ручной разработки или автоматической генерации для конкретной среды тестирования;
- прогон программы на тестах, управляемый тестовым монитором (test monitor, test driver), с получением протокола результатов тестирования (test log);
- оценка результатов выполнения программы на наборе тестов с целью принятия решения о продолжении или остановке тестирования.

Тестирование разделяют на статическое и динамическое.

*Статическое* тестирование выявляет формальными методами анализа без выполнения тестируемой программы неверные конструкции или неверные отношения объектов программы (ошибки формального задания) с помощью специальных инструментов контроля кода.

*Динамическое* тестирование (собственно тестирование) осуществляет выявление ошибок только на выполняющейся программе с помощью специальных инструментов автоматизации тестирования.

*Отладка* (debug, debugging) – процесс поиска, локализации и исправления ошибок в программе.

Термин «отладка» в зарубежной литературе используется двояко: 1) для обозначения активности по поиску ошибок (собственно тестирование), по нахождению причин их появления и исправлению; 2) для обозначения активности по локализации и исправлению ошибок.

*Тестирование* обеспечивает выявление (констатацию наличия) фактов расхождений с требованиями (ошибок).

Как правило, на фазе тестирования осуществляется и исправление идентифицированных ошибок, включающее локализацию ошибок, нахождение причин ошибок и соответствующую корректировку программы тестируемого приложения.

Если программа не содержит синтаксических ошибок (прошла трансляцию) и может быть выполнена на компьютере, она обязательно вычисляет какую-либо функцию, осуществляющую отображение входных данных в выходные. Это означает, что компьютер на своих ресурсах доопределяет частично определенную программой функцию до тотальной определенности. Следовательно, судить о правильности или неправильности результатов выполнения программы можно, только сравнивая специфика-

цию желаемой функции с результатами ее вычисления, что и осуществляется в процессе тестирования.

*Путь* программы – последовательность вершин (операторов программы) и дуг (управлений, соединяющих операторы-вершины).

Пути, различающиеся хотя бы числом прохождений цикла, – разные пути, поэтому число путей в программе может быть не ограничено.

*Ветвь* – линейный участок программы, содержащий первым условный оператор или первый оператор программы, затем безусловные операторы и заканчивающийся условным оператором либо оператором выхода из программы. Ветвью программы – конечное число.

Существуют реализуемые и нереализуемые пути в программе, в нереализуемые пути в обычных условиях попасть нельзя.

### **Критерии выбора тестов**

Основная проблема тестирования – определение достаточности множества тестов для истинности вывода о правильности реализации программы, а также нахождение множества тестов, обладающего этим свойством. Поскольку тестирование программы на всех входных значениях невозможно и невозможно тестирование на всех путях – задача о выборе конечного набора тестов для проверки программы в общем случае неразрешима. Поэтому для решения практических задач применяют частные случаи решения этой задачи.

Требования к идеальному критерию:

– **критерий должен быть достаточным**, т.е. показывать, когда некоторое конечное множество тестов достаточно для тестирования данной программы;

– **критерий должен быть полным**, т.е. в случае ошибки должен существовать тест из множества тестов, удовлетворяющих критерию, который раскрывает ошибку;

– **критерий должен быть надежным**, т.е. любые два множества тестов, удовлетворяющих ему, одновременно должны раскрывать или не раскрывать ошибки программы;

– **критерий должен быть легко проверяемым**, например вычисляемым на тестах.

Для нетривиальных классов программ в общем случае не существует полного и надежного критерия, зависящего от программ или спецификаций.

Существуют следующие классы критериев:

**класс I.** Структурные критерии используют информацию о структуре программы (критерии так называемого «белого ящика»);

**класс II.** Функциональные критерии формулируются в описании требований к программному изделию (критерии так называемого «черного ящика»);

**класс III.** Критерии стохастического тестирования формулируются в терминах проверки наличия заданных свойств у тестируемого приложения средствами проверки некоторой статистической гипотезы;

**класс IV.** Мутационные критерии ориентированы на проверку свойств программного изделия на основе подхода Монте-Карло.

**Структурные критерии (класс I).** Структурные критерии используют модель программы в виде «белого ящика», что предполагает знание исходного текста программы или спецификации программы в виде потокового графа управления. Структурная информация понятна и доступна разработчикам подсистем и модулей приложения, поэтому данный класс критериев часто используется на этапах модульного и интеграционного тестирования.

Структурные критерии базируются на операторах, ветвях и путях. Наиболее важные среди них:

- **Тестирование команд** (критерий C0): набор тестов в совокупности должен обеспечить прохождение каждой команды не менее одного раза. Это слабый критерий, он, как правило, используется в больших программных системах, где другие критерии применить невозможно.

- **Тестирование ветвей** (критерий C1): набор тестов в совокупности должен обеспечить прохождение каждой ветви не менее одного раза. Это достаточно сильный и при этом экономичный критерий, поскольку множество ветвей в тестируемом приложении конечно и не так уж велико. Данный критерий часто используется в системах автоматизации тестирования.

- **Тестирование путей** (критерий C2): набор тестов в совокупности должен обеспечить прохождение каждого пути не менее 1 раза. Если про-

грамма содержит цикл (в особенности с неявно заданным числом итераций), то число итераций ограничивается константой (часто – 2, или числом классов выходных путей).

Структурные критерии не проверяют соответствие спецификации, если оно не отражено в структуре программы. Поэтому при успешном тестировании программы по критерию С2 мы можем не заметить ошибку, связанную с невыполнением некоторых условий спецификации требований.

**Функциональные критерии (класс II).** Функциональный критерий – важнейший для программной индустрии критерий тестирования. Он обеспечивает, прежде всего, контроль степени выполнения требований заказчика в программном продукте. Поскольку требования формулируются к продукту в целом, они отражают взаимодействие тестируемого приложения с окружением. При функциональном тестировании преимущественно используется модель «черного ящика». Проблема функционального тестирования – это, прежде всего, трудоемкость; дело в том, что документы, фиксирующие требования к программному изделию, как правило, достаточно объемны, тем не менее, соответствующая проверка должна быть всеобъемлющей.

Ниже приведены частные виды функциональных критериев.

- **Тестирование пунктов спецификации:** набор тестов в совокупности должен обеспечить проверку каждого тестируемого пункта не менее одного раза.

Спецификация требований может содержать сотни и тысячи пунктов требований к программному продукту, и каждое из этих требований при тестировании должно быть проверено в соответствии с критерием не менее чем одним тестом.

- **Тестирование классов входных данных:** набор тестов в совокупности должен обеспечить проверку представителя каждого класса входных данных не менее одного раза.

При создании тестов классы входных данных сопоставляются с режимами использования тестируемого компонента или подсистемы приложения, что заметно сокращает варианты перебора, учитываемые при разработке тестовых наборов. Следует заметить, что перебирая в соответствии с критерием величины входных переменных (например, различные файлы – источники входных данных), мы вынуждены применять мощные

тестовые наборы. Действительно, наряду с ограничениями на величины входных данных существуют ограничения на величины входных данных во всевозможных комбинациях, в том числе проверка реакций системы на появление ошибок в значениях или структурах входных данных. Учет этого многообразия – процесс трудоемкий, что создает сложности для применения критерия.

- **Тестирование правил:** набор тестов в совокупности должен обеспечить проверку каждого правила, если входные и выходные значения описываются набором правил некоторой грамматики.

Следует заметить, что грамматика должна быть достаточно простой, чтобы трудоемкость разработки соответствующего набора тестов была реальной (вписывалась в сроки и штат специалистов, выделенных для реализации фазы тестирования).

- **Тестирование классов выходных данных:** набор тестов в совокупности должен обеспечить проверку представителя каждого выходного класса, при условии что выходные результаты заранее расклассифицированы, причем отдельные классы результатов учитывают в том числе ограничения на ресурсы или на время (time out).

При создании тестов классы выходных данных сопоставляются с режимами использования тестируемого компонента или подсистемы, что заметно сокращает варианты перебора, учитываемые при разработке тестовых наборов.

- **Тестирование функций:** набор тестов в совокупности должен обеспечить проверку каждого действия, реализуемого тестируемым модулем, не менее одного раза.

Очень популярный на практике критерий, который, однако, не обеспечивает покрытия части функциональности тестируемого компонента, связанной со структурными и поведенческими свойствами, описание которых не сосредоточено в отдельных функциях (т.е. описание рассредоточено по компоненту).

Критерий тестирования функций объединяет отчасти особенности структурных и функциональных критериев. Он базируется на модели «полупрозрачного ящика», где явно указаны не только входы и выходы тестируемого компонента, но также состав и структура используемых методов (функций, процедур) и классов.



- **Комбинированные критерии для программ и спецификаций:** набор тестов в совокупности должен обеспечить проверку всех комбинаций непротиворечивых условий программ и спецификаций не менее одного раза.

При этом все комбинации непротиворечивых условий надо подтвердить, а условия противоречий следует обнаружить и ликвидировать.

**Стохастические критерии (класс III).** Стохастическое тестирование применяется при тестировании сложных программных комплексов – когда набор детерминированных тестов имеет громадную мощность.

В случаях, когда подобный набор невозможно разработать и исполнить на фазе тестирования, можно применить следующую методику:

- Разработать программы – имитаторы случайных последовательностей входных сигналов.

- Вычислить независимым способом значения выходных сигналов для соответствующих входных сигналов и получить тестовый набор.

- Протестировать приложение на тестовом наборе, используя два способа контроля результатов:

*детерминированный контроль* – проверка соответствия вычисленного значения значению, полученному в результате прогона теста на наборе – случайной последовательности входных сигналов, сгенерированной имитатором;

*стохастический контроль* – проверка соответствия множества значений, полученного в результате прогона тестов на наборе входных значений, заранее известному распределению результатов. В этом случае множество выходных сигналов неизвестно (его вычисление невозможно), но известен закон распределения данного множества.

Критерии стохастического тестирования:

- **Статистические методы** окончания тестирования – стохастические методы принятия решений о совпадении гипотез о распределении случайных величин. К ним принадлежат широко известные метод Стьюдента ( $St$ ), метод Хи-квадрат ( $\chi^2$ ) и т.п.

- **Метод оценки скорости выявления ошибок** основан на модели скорости выявления ошибок, согласно которой тестирование прекращается, если оцененный интервал времени между текущей ошибкой и следующей слишком велик для фазы тестирования приложения.

**Мутационный критерий (класс IV).** Предлагается подход, позволяющий на основе мелких ошибок оценить общее число ошибок, оставшихся в программе.

Подход базируется на следующих понятиях:

*мутации* – мелкие ошибки в программе;

*мутанты* – программы, отличающиеся друг от друга мутациями.

**Метод мутационного тестирования:** в разрабатываемую программу Р вносят мутации, т.е. искусственно создают программы-мутанты Р1, Р2,... Затем программа Р и ее мутанты тестируются на одном и том же наборе тестов.

Если на наборе тестов подтверждается правильность программы Р и, кроме того, выявляются все внесенные в программы-мутанты ошибки, то **набор тестов соответствует** мутационному критерию, а тестируемая программа объявляется **правильной**.

Если некоторые мутанты не выявили всех мутаций, то надо расширить набор тестов и продолжать тестирование.

### *Использованная и рекомендуемая литература*

1. Благодатских В. А. Стандартизация разработки программных средств / В. А. Благодатских, В. А. Волнин, К. Ф. Посакалов. – М. : Финансы и статистика, 2005. – 288 с.
2. Борзов Ю. В. Выбор путей программы для построения тестов / Ю. В. Борзов, Г. Б. Уртанс, В. А. Шимаров // УСиМ. – 1989. – № 6. – С. 29–36.
3. Брукс Ф. Мифический человеко-месяц, или Как создаются программные системы / Ф. Брукс. – СПб. : Символ-Плюс, 1999. – 304 с.
4. Ван Тассел Д. Стиль, разработка, эффективность, отладка и испытания программ : пер. с англ. / Д. Ван Тассел. – 2-е изд., испр. – М. : Мир, 1985.
5. Василенко Н. В. Модели оценки надежности программного обеспечения / Н. В. Василенко, В. А. Макаров // Вестник Новгородского гос. ун-та. – 2004. – № 28. – С. 126–132.
6. Вентцель Е. С. Теория вероятностей и ее инженерные приложения / Е. С. Вентцель, Л. А. Овчаров. – М. : Наука, ГРФМЛ, 1988. – 480 с.
7. Гласс Р. Сопровождение программного обеспечения / Р. Гласс, Р. Нуазо. – М. : Мир, 1983.
8. Гнеденко Б. В. Курс теории вероятностей / Б. В. Гнеденко. – М. : Наука, ГРФМЛ, 1988. – 448 с.
9. ГОСТ 24.701-86. Единая система стандартов автоматизированных систем управления. Надежность автоматизированных систем управления. Основные положения.
10. ГОСТ 27.002-89. Надежность в технике. Основные понятия. Термины и определения.
11. ГОСТ 34.003-90. Информационная технология. Комплекс стандартов и руководящих документов на автоматизированные системы. Термины и определения.
12. ГОСТ Р ИСО/МЭК 9126-93. Информационная технология. Оценка программной продукции. Характеристика качества и руководство по их применению.

13. Дружинин Г. В. Надежность автоматизированных систем / Г. В. Дружинин. – М. : Энергия, 1977. – 536 с.
14. Жоголев Е. А. Введение в технологию программирования : конспект лекций / Е. А. Жоголев. – М. : ДИАЛОГ-МГУ, 1994.
15. Канер С. Тестирование программного обеспечения / С. Канер, Дж. Фолк, Нгуен Енг. – Киев : ДиаСофт, 2000. – 544 с.
16. Котляров В. П. Основы тестирования программного обеспечения : учеб. пособие / В. П. Котляров, Т. В. Коликова. – М. : БИНОМ. Лаб. знаний : Интернет-Ун-т информ. технологий, 2006. – 285 с.
17. Лаврищева Е. М. Сборочное программирование. Основы индустрии программных продуктов / Е. М. Лаврищева, В. Н. Грищенко. – Киев : Наукова Думка, 2009. – 372 с.
18. Леонтьев Е. А. Надежность экономических информационных систем / Е. А. Леонтьев. – Тамбов : Тамбовский гос. тех. ун-т, 2002. – 128 с.
19. Липаев В. В. Надежность программных средств / В. В. Липаев. – М. : СИНТЕГ, 1998. – 232 с.
20. Липаев В. В. Тестирование программ / В. В. Липаев. – М. : Радио и связь, 1986. – 296 с.
21. Майерс Г. Искусство тестирования программ / Г. Майерс. – М. : Финансы и статистика, 1982. – 176 с.
22. Майерс Г. Надежность программного обеспечения : пер. с англ. / Г. Майерс. – М. : Мир, 1980. – 356 с.
23. Математическая энциклопедия / гл. ред. И. М. Виноградов. – М. : Советская энциклопедия, 1985. – Т. 4.
24. Надежность информационных систем / Ю. Ю. Громов и др. – Тамбов : ГОУ ВПО ТГТУ, 2010. – 160 с.
25. Половко А. М. Основы теории надежности / А. М. Половко, С. В. Гуров. – СПб. : БХВ-Петербург, 2006. – 704 с.
26. Характеристики качества программного обеспечения / Б. Боэм и др. – М. : Мир, 1981.
27. Холстед М. Х. Начала науки о программах / М. Х. Холстед. – М. : Финансы и статистика, 1981. – 128 с.
28. Чичев А. А. Отчет по НИР «Разработка принципов построения системы автоматизированного проектирования ПО перспективных ИУС»

(шифр «Колос»). Основные принципы выбора языка программирования. РК № У-98328. – 1984.

29. Чичев А. А. Отчёт по НИР «Разработка принципов построения системы автоматизированного проектирования ПО перспективных ИУС» (шифр «Колос»). Основные принципы тестирования ПО ИУС. РК № У-98328 / А. А. Чичев, И. И. Караган. – 1984.

30. Чичев А. А. Отчёт по НИР «Разработка системы технологической поддержки процесса проектирования ПО изделий» (шифр «Калина»). Кн. 1. Принципы технологической поддержки. РК № У-10818 / А. А. Чичев, В. А. Беспалов. – 1984.

31. Чичев А. А. Стилиевые конфликты в обучении программированию / А. А. Чичев, Е. Г. Чекал // Сб. тр. XI Международной конференции «Информационные технологии в образовании». Ч. 2. Москва, 2001. – М. : МИФИ, 2001. – С. 151–154.

32. Шимаров В. А. Тестирование программ: цели и особенности инструментальной поддержки // Программное обеспечение ЭВМ / АН БССР. Институт математики. – Минск, 1994. – Вып. 100. – С. 19–43

33. Шураков В. В. Надежность программного обеспечения систем обработки данных : учебник для вузов / В. В. Шураков. – М. : Статистика, 1981. – 216 с.

Учебное издание

*Чекал Елена Георгиевна*  
*Чичев Александр Алексеевич*

## **НАДЕЖНОСТЬ ИНФОРМАЦИОННЫХ СИСТЕМ**

**Учебное пособие**

Директор Издательского центра *Т. В. Филиппова*  
Редактирование и подготовка оригинал-макета *Е. Г. Туженковой*  
Оформление обложки *Н. В. Пеньковой*

Подписано в печать 25.12.12.  
Формат 60x84/16. Усл. печ. л. 7,0. Уч.-изд. л. 5,9.  
Тираж 100 экз. Заказ 237 /

Оригинал-макет подготовлен  
в Издательском центре  
Ульяновского государственного университета

Отпечатано в Издательском центре  
Ульяновского государственного университета  
432017, г. Ульяновск, ул. Л. Толстого, 42