ФЕДЕРАЛЬНОЕ АГЕНТСТВО ПО ОБРАЗОВАНИЮ

МОСКОВСКИЙ ИНЖЕНЕРНО-ФИЗИЧЕСКИЙ ИНСТИТУТ
(ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ)

В.О. Чуканов

Надежность программного обеспечения и аппаратных средств систем передачи данных атомных электростанций

Рекомендовано УМО «Ядерные физика и технологии» в качестве учебного пособия для студентов высших учебных заведений

Москва 2008

УДК 004.4.052Ж621.311.25(075)
ББК 32.973.202-018.2я7
Ч-88

Чуканов В.О. Надежность программного обеспечения и аппаратных средств систем передачи данных атомных электростанций: Учебное пособие. – М.: МИФИ, 2008. – 168 с.

Данное учебное пособие предназначено студентам старших курсов, аспирантам и молодым специалистам, а также инженерам, бакалаврам и магистрам, которые занимаются разработкой и проектированием высоконадёжных вычислительных систем с ориентацией на атомные электростанции.

Система передачи данных атомных электростанций является основным звеном в передаче данных в случае критических ситуаций.

Подготовлено в рамках Инновационной образовательной программы МИФИ.

Рецензент д-р техн. наук, проф. Ю.Ю. Шумилов.

ISBN 978-5-7262-1076-6

© Московский инженерно-физический институт (государственный университет), 2008

ОГЛАВЛЕНИЕ

ПРЕДИСЛОВИЕ 5
ГЛАВА 1. ОСНОВНЫЕ НАПРАВЛЕНИЯ ИССЛЕДОВАНИЙ В ОБЛАСТИ НАДЕЖНОСТИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ..... 6
1.1. Проблемы надежности ПО 7
1.2. Основные этапы разработки ПО 12
1.3. Количественные характеристики надежности 16
1.4. Классификация моделей надежности ПО 31
1.5. Тестирование ПО. Испытания ПО.....37
Список литературы к главе 1 42
ВВЕДЕНИЕ К ГЛАВАМ 2 И 3 43
ГЛАВА 2. МЕТОДЫ РЕЗЕРВИРОВАНИЯ АППАРАТНЫХ СРЕДСТВ СИСТЕМЫ ПЕРЕДАЧИ ДАННЫХ АЭС.....47
2.1. Аналитический обзор высоконадежных систем передачи данных 47
2.2. Выбор критериев надежности восстанавливаемых систем передачи данных.....60
2.3. Разработка логико-аналитической модели надежности системы передачи данных.....65
Список литературы к главе 2 71
ГЛАВА 3. МОДЕЛИ НАДЕЖНОСТИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ТИПОВОГО НАПРАВЛЕНИЯ СИСТЕМ ПЕРЕДАЧИ ДАННЫХ АЭС.....75
3.1. Нормативные определения надежности программного обеспечения и процессов оценивания 75
3.2. Особенности программного обеспечения как восстанавливаемой системы.....85
3.3. Модель надежности программного обеспечения типового направления системы передачи данных между Кризисным центром и атомными станциями 99
3.4. Программная реализация обобщенной модели надежности программного обеспечения типового направления системы передачи данных.....108

3.5. Разработка рекомендаций по построению комплексной модели надежности типового направления системы передачи данных с учетом параметров ее аппаратного и программного обеспечения 118
Список литературы к главе 3 120
ВВЕДЕНИЕ К ГЛАВАМ 4 И 5 121
ГЛАВА 4. РАСЧЕТ ПОКАЗАТЕЛЕЙ НАДЁЖНОСТИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ВЫЧИСЛИТЕЛЬНЫХ СРЕДСТВ СИСТЕМ ПЕРЕДАЧИ ДАННЫХ АЭС 123
4.1. Особенности оценки надежности программного обеспечения высоконадёжных вычислительных систем.....123
4.2. Структура имитационных моделей надёжности программного обеспечения 125
4.3. Расчет показателей надежности программного обеспечения, использующегося в составе вычислительных средств систем передачи данных 132
Список литературы к главе 4 146
ГЛАВА 5. МОДЕЛИ НАДЕЖНОСТИ АППАРАТНЫХ СРЕДСТВ СИСТЕМ ПЕРЕДАЧИ ДАННЫХ АЭС 147
5.1. Обеспечение надежности современных информационно-вычислительных систем 147
5.2. Модели надежности сервера типа NT-AKPB-LEN 150
5.3. Модели надежности систем передачи данных 159
5.4. Расчет параметров одиночного и группового ЗИП.....163
5.5. Программа расчета параметров ЗИП.....164
Список литературы к главе 5 166

ПРЕДИСЛОВИЕ

В настоящее время интерес к проектированию высоконадёжных систем специального назначения постоянно возрастает. Особенно актуальна эта проблема для компьютерного обеспечения работы атомных электростанций. В России для мониторинга работы атомных станций был создан Кризисный центр для фиксации критических ситуаций. Основным звеном информационного обеспечения работы атомных электростанций являются системы передачи данных, в том числе между станциями и Кризисным центром.

В данном учебном пособии представлены основные проблемы надежностного проектирования систем передачи данных, даны обзоры методов и подходов к повышению надежности программных и аппаратных средств.

Произведена классификация моделей надёжности программного обеспечения и аппаратуры. Освещен целый ряд вопросов проектирования систем передачи данных. Представлена конкретная методика надежностного проектирования систем передачи данных. Даны практические примеры проектирования систем с точки зрения надежности. В примерах учитываются современные характеристики программного обеспечения и аппаратных средств.

Пособие будет полезно студентам, аспирантам и научным работникам, которые занимаются надежностным проектированием.

Материалы к главам 3 и 4 любезно предоставлены доцентом Гуровым В.В., а к главе 5 доцентом Александровичем А.Е., за что автор выражает им искреннюю признательность.

ГЛАВА 1. ОСНОВНЫЕ НАПРАВЛЕНИЯ ИССЛЕДОВАНИЙ В ОБЛАСТИ НАДЕЖНОСТИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

На сегодня самая серьезная проблема в области обработки данных – это проблема ПО. Внешне она проявляется в постоянных жалобах на то, что ПО, во-первых, дорого, во-вторых, ненадежно. Большинство специалистов считает первый из недостатков во многом проявлением второго. Поскольку современное ПО по самой своей природе ненадежно, его тестирование и обслуживание требуют существенных расходов.

Интересно отметить, что проблема надежности ПО (с позиций, весьма близких к современным) рассматривалась еще на заре применения ВМ.

Это наблюдение было опубликовано тремя английскими математиками в 1952 г. «Те, кто регулярно программируют для быстродействующих электронных машин, знают на собственном опыте, что солидная доля подготовительного этапа работы на ЭВМ уходит на устранение ошибок, сделанных при составлении программ. С помощью здравого смысла и отладочных программ большинство ошибок удается найти и исправить достаточно быстро. Однако некоторые из них настолько неуловимы, что не поддаются обнаружению удивительно долгое время».

Хотя ошибки в ПО встречались и до 1952 г., это, кажется, первое признание проблемы надежности, т.е. того факта, что тестирование требует значительного времени, и даже после его завершения некоторые ошибки в ПО остаются необнаруженными.

В профессиональных журналах по обработке данных часто публикуются веселые (для всех, кроме их участников) истории о классических ошибках в ПО.

1.1. Проблемы надежности ПО

Рассмотрим наиболее важные проблемы надежности ПО.

1. Оценка надежности ПО.
2. Определение факторов, влияющих на достижение заданной надежности.
3. Совершенствование методов повышения надежности в процессе проектирования и в процессе эксплуатации разработанного ПО.

Направления исследований в надежности ПО

1. Обоснование интуитивного представления о надежности ПО.
2. Разработка методов, обеспечивающих достижение заданного уровня надежности.

Основные типы комплексов программ

1. Программы для решения инженерных и научно-исследовательских задач.

Характеристики:

- относительно небольшой жизненный цикл;
- неполное использование ресурсов вычислительной системы;
- небольшая длительность разработки;
- их эксплуатация носит эпизодический и кратковременный характер;
- отсутствие жестких ограничений на допустимую длительность ожидания результатов;
- практически всегда имеется возможность достаточно строго проконтролировать выходные данные и при необходимости поставить контрольные эксперименты.

К этому типу программ практически не применимы основные понятия теории надежности.

2. Сложные комплексы программ для информационно–справочных систем, которые функционируют вне реального времени. Это системы организационного типа.

Характеристики:

- период их эксплуатации значительно превышает длительность разработки;
- в ходе эксплуатации они могут развиваться и обновляться, что влечет за собой изменение характеристик и сопровождающей документации.

Программы этого типа можно классифицировать как системы и применять к ним теорию надежности. Для них могут быть определены функции, характеристики, а также промежуток времени, на котором должны сохраняться заданные показатели, в соответствии с определениями теории надежности и требованиями технической документации.

- изменение комплекса программ в процессе развития и модернизации системы приводят к тому, что содержание показателей надежности оказываются нестабильными.

3. Комплексы программ автоматического или автоматизированного управления, непосредственно входящие в контур управления и функционирующие в реальном масштабе времени.

Характеристики:

- обычно практически полностью используют ресурсы вычислительной системы;
- снабжаются подробной документацией;
- эксплуатируются долгое время.

Такие программы обладают всеми чертами промышленных изделий. К ним в наибольшей степени применимы понятия теории надежности.

Факторы, позволяющие анализировать показатели надежности программ 2-го и 3-го типов

1. Стабильность длительной эксплуатации.
2. Наличие технической документации.
3. Возможность существования дефектов в комплексах программ, вызывающих сбои и отказы при функционировании.
4. Взаимосвязь надежностных характеристик ПО и аппаратуры.

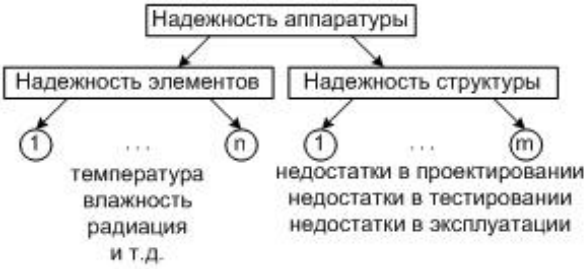


Рис 1.1. Факторы надёжности

Факторы надежности аппаратных средств

- Надежность элементов.
- Ошибки в конструировании, допущенные при проектировании или изготовлении.

Относительно невысокая надежность компонентов, глубокая взаимозависимость, способность к старению и разрушению или снижению надежности в процессе эксплуатации привели к тому, что этот фактор в ряде случаев оказывается превалирующим для надежности большинства компонентов аппаратуры.

С другой стороны надежность программных компонентов определяется теми же двумя факторами, однако, степень их влияния иная.

Хранение программ на магнитном носителе при отсутствии внешних вмешательств характеризуется высокой надежностью. Содержание программ на бумаге практически не подвержено разрушению в разумные сроки. Превалирующим для надежности ПО является 2-й фактор, т.е. ошибки проектирования.

Отмеченные факторы различаются по своей природе, проявлению и методам устранения. Отказы компонента аппаратуры, обусловленные старением или разрушением элементов, в большинстве случаев требуют их замены. Восстановительные работы и особенно замена компонентов аппаратуры трудно автоматизируются и сохраняют преимущественно ручной характер. Это определяет длительность восстановления (минуты, часы).

Проявление ошибок проектирования ПО обусловлено конкретными ситуациями и сочетанием данных. Возникающий в результа-

те отказ, как правило, не связан с физическим разрушением аппаратуры и не требует проведения ремонтных работ. Такие отказы в программных комплексах, как, например, зацикливание или искажение массивов данных могут быть устранены программными методами. Поэтому процессы оперативного восстановления при отказах реализуются преимущественно автоматизированными методами. Проблема восстановления при отказах переходит в проблему преобразования отказов в автоматически- или автоматизированно-устраняемые сбои. При этом не устраняется причина отказа и в такой же ситуации отказ должен повторится. Однако повторение данных, приводящих к отказу, маловероятно и наработка на отказ может быть удовлетворительной, тем более что его проявление сведено к кратковременному сбою.

Проблемы эталонов

Накопление сведений о сбоях и отказах в ПО позволяет их устранить в период профилактических работ. Понятие ошибок проектирования программ трудно сформулировать без учета выполнения программ и учета результата их функционирования.

Во многих случаях имеются эталонные изделия, с которыми сравниваются вновь изготовленные. Такое сравнение позволяет в статике выявить ошибки тиражирования. При этом ошибки проектирования сохраняются соответственно эталонному изделию.

Тиражирование программ может проводится очень точно, а их физическое разрушение маловероятно и легко устранимо. В процессе отладки и испытания программ устраняются многие ошибки, и программы приближаются к идеальным. Однако степень их приближения остается неизвестной, и копии программ содержат ошибки эталонной программы.

Программы любой сложности при фиксированных исходных данных и надежной аппаратуре исполняются по заданному маршруту и дают на выходе определенный результат. Однако комбинаторный характер исходных данных и множество условных переходов создают огромное число различных маршрутов выполнения программы, которые на несколько порядков превышают число команд в программе. Такое число вариантов исполнения программы не может быть проверено полностью при отладке и приемочных, и

сертификационных испытаниях. Некоторые варианты выполнения программы могут быть маловероятны.

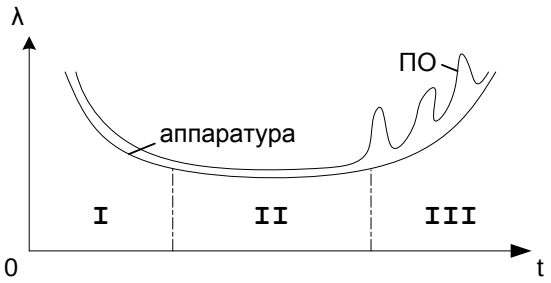


Рис. 1.2. График зависимости интенсивности λ от времени t, где λ – интенсивность проявления отказов или ошибок

Отказы из-за ошибок в программах сначала уменьшаются вследствие их обнаружения и устранения в процессе отладки. В процессе устаревания ПО модифицируется, что приводит к возникновению новых ошибок.

Затраты на обеспечение надежности должны быть сопоставимы с ущербом вследствие отказа системы.

1.2. Основные этапы разработки ПО

Единственно важная причина ошибок в ПО – это неправильный перевод из одного представления в другое [1].

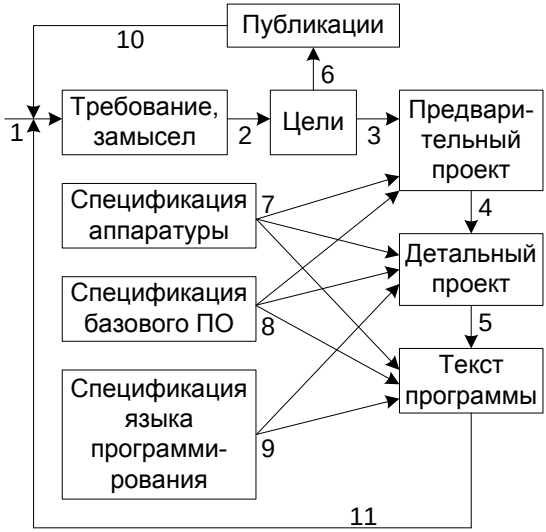


Рис. 1.3. Этапы разработки ПО

На рис. 1.3 представлены основные этапы разработки ПО.

1. Разработка описания реальной задачи в виде перечня требований пользователя (который в некоторых случаях пользователь составляет сам). Здесь имеются обширные возможности для появления ошибок. Например, пользователь не сумеет адекватно выразить свои потребности, они могут быть неверно поняты либо учтены не в полном объеме. Ошибки этого уровня обходятся чрезвычайно дорого.

2. Перевод требований пользователя в цели программы. Ошибки на этом этапе возникают, когда неверно определяются требования.

3. Шаг связан с преобразованием цели программы в ее внешние спецификации, т.е. поведение всей системы с точки зрения пользо-

вателей. По объему перевода это самый значительный этап, он больше всего подвержен ошибкам.

4. Этот этап представляет собой несколько переводов от внешнего описания готового продукта до получения готового проекта, описывающего многие составляющие проекта-предложения, выполнение которых должно обеспечить поведение системы, соответствующее внешним спецификациям:

- перевод внешнего описания в структуру программы (например, модуль);
- перевод каждой из этих компонент в описание процедурных шагов (например, блок-схему).

Так как приходится иметь дело с все большим объемом информации, шансы внесения ошибок очень высоки.

5. Перевод описания логики программы в предложения языка программирования. На этом этапе делается много ошибок, но они легко обнаруживаются и корректируются. На этом этапе есть еще один перевод – перевод текста программы на языке программирования в объектный код (выполняется компиляторами или интерпретаторами).

6. В результате работы над программным проектом возникает как само ПО, так и документы, описывающие его использование в виде руководств в бумажном или электронном виде. Если возникают ошибки при подготовке документации, то она не будет точно описывать поведение программы (если только на шагах 4 и 6 не сделаны идентичные ошибки). Прочитав руководство, пользователь начнет работать с программой и обнаружит, что она ведет себя не так, как он ожидал – это и является, по определению, ошибкой в программе.

7. Например, разработчик операционной системы опирается на описание процессора (набор команд, прерывания, средства защиты и т.д.) и периферийного оборудования. Разработчики должны знать характеристики линий связи и т.д. Неправильное толкование этих данных может привести к ошибкам в программе.

8. Прикладные программы взаимодействуют с базовым ПО (таким, например, как ОС). Неправильное понимание документации по базовому ПО – еще один источник ошибок.

9. Готовая программа состоит из предложений хотя бы одного языка программирования. Непонимание синтаксиса и семантики языка – также причина ошибок.

10. Есть две формы связи между пользователем и готовой программой – это руководство, описывающее использование и непосредственно работа с ним. Этот шаг представляет собой изучение пользователем руководств и перевод их содержания в его понимание того, как он желает применять программу.

Модель перевода входной информации в выходную

Рассмотрим обобщенную модель перевода информации из одного вида в другой. В модели отражены основные этапы перевода с точки зрения разработчика. Модель дает возможность представить перевод информации из одного вида в другой (рис. 1.4).

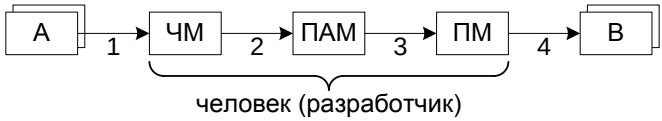


Рис. 1.4. Модель перевода информации

А – исходная информация.

В – результирующая информация.

ЧМ – читающий механизм (области мозга, управляющие зрением и слухом).

ПАМ – память.

ПМ – пишущий механизм (области мозга, управляющие речью и движением рук).

1. Человек получает информацию с помощью ЧМ.
2. Человек запоминает информацию в своей памяти.
3. Человек выбирает из памяти эту информацию, а также информацию, описывающую процесс перевода, выполняет перевод и посылает результат своему ПМ.
4. Информация физически распространяется с помощью печати на терминале или с помощью речи.

Ошибки перевода состоят:

1. Одна из способностей человека – понимать входную информацию, сопоставляя ее с образами, созданными образованием и жизненным опытом. Этот принцип основывается на способности человека «читать между строк», понимать грамматически неправильные предложения и т.д. Эти способности могут вызывать неточности в переводе информации. Это возникает тогда, когда человек читает документ А и видит то, что хочет, а не то, что написано, пытается восстановить недостающие факты или не понимает информацию. Ошибки могут присутствовать в самом документе А.

2. В большинстве случаев, чтобы правильно запомнить информацию, надо ее понять. Ошибки появляются в результате неправильной интерпретации или полного непонимания входной информации. Информация может быть слишком сложной или двусмысленной, образовательный уровень человека может оказаться недостаточно высоким.

3. Основной источник ошибок на этом этапе – забывчивость. Человек может забыть входную информацию А, либо точные правила выполнения перевода. Слабость других умственных способностей, таких как четкость мышления, также способствует появлению ошибок.

4. Многие не умеют ясно писать или выражать свои мысли, это затуманивает смысл их сообщений. Если количество выходной информации велико, человека начинает раздражать разница между скоростью мышления и письма. Чтобы справиться с этим, он использует сокращения, либо предполагает, что факты будут интуитивно очевидны. Это увеличивает вероятность того, что следующий участник процесса разработки при переводе совершит ошибки.

1.3. Количественные характеристики надежности

Программа тестируется на всех компьютерах, выявляются ошибки. Через промежуток времени Δt фиксируются появления ошибок, собирается статистика.

P(t) – вероятность безотказной работы, т.е. вероятность того, что случайная величина τ (время проявления ошибки) будет больше заданного времени *t*

$P(t) = \mathrm{P}(\tau > t)$.

$Q(t)$ – вероятность отказа

$Q(t) = \mathrm{P}(\tau < t)$.

$P^*(t)$ – статистическое определение вероятности безотказной работы

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

где N_0 – количество прогонов;

$n(t)$ – количество прогонов, закончившихся ошибками за время *t*.

$a(t) = Q'(t)$ – частота отказов.

a(t)* – статистическое определение частоты отказов

$$a^*(t) = \frac{n(\Delta t_i)}{N_0 \cdot \Delta t_i}.$$

Типовая задача оценки надежности ПО

Проводятся прогоны ПО на нескольких компьютерах параллельно.

Все прогоны начались в момент времени $t = 0$.

Количество отказов фиксировалось с интервалом Δt = 500 часов, $n(\Delta t_i)$ – количество отказов ПО на *i*-м интервале.

Требуется определить вероятность безотказной работы $P^*(t)$ и частоту отказов $a^*(t)$.

Исходные данные и результаты расчетов для N_0 = 1000 прогонов ПО приведены в таблице.

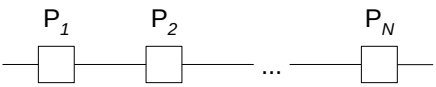
Δt_i, ч.	$n(\Delta t_i)$	$P^*(t)$	$a^*(t)$
0 – 500	145	0,855	0,00029
500 – 1000	86	0,769	0,000172
1000 – 1500	77	0,692	0,000154
1500 – 2000	69	0,623	0,000138
2000 – 2500	62	0,561	0,000124
2500 – 3000	56	0,505	0,000112
3000 – 3500	51	0,454	0,00102
3500 – 4000	45	0,409	0,00009
4000 – 4500	41	0,368	0,000082
4500 – 5000	37	0,331	0,000074
5000 – 5500	33	0,298	0,000066
5500 – 6000	35	0,263	0,00007
6000 – 6500	60	0,203	0,00015
6500 – 7000	75	0,128	0,00015
7000 – 7500	62	0,066	0,000124
7500 – 8000	42	0,024	0,000084
8000 – 8500	16	0,008	0,000032

Структурно-логические модели надежности ПО

Классификация этого типа моделей приведена на рис. 1.5.

Рис. 1.5. Структурно-логические модели надёжности ПО

Последовательная



Отказ одной подсистемы (программы) приводит к отказу всей системы.

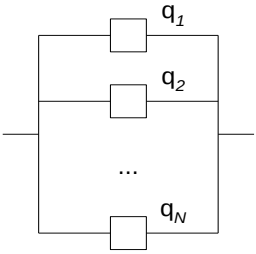
$P_C = \prod_{i=1}^{N} P_i$ – вероятность безотказной работы всей системы.

$Q_C = 1 - P_C$ – вероятность отказа всей системы.

$q_i = 1 - P_i, \quad i \in \overline{1, N}$ – вероятность отказа одной подсистемы (программы).

Это модель неизбыточных систем.

Параллельная



При отказе одной подсистемы (программы) ее функции берет на себя одна из оставшихся. Пример: RAID-массив.

N – кратность резервирования (уровень избыточности).

q_i, $i \in \overline{1, N}$ – вероятность отказа одной подсистемы (программы).

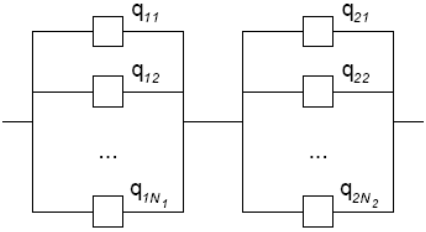
$Q_C = \prod_{i=1}^{N} q_i$ – вероятность отказа всей системы.

$P_C = 1 - Q_C$ – вероятность безотказной работы всей системы.

$P_i = 1 - q_i, \quad i \in \overline{1, N}$ – вероятность безотказной работы одной подсистемы.

Параллельно-последовательная

Схема содержит две избыточные подсистемы.

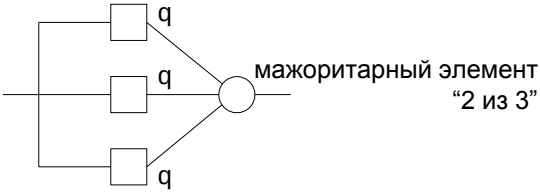


$$\overline{q_1} = \prod_{i=1}^{N_1} q_{1i} \qquad \overline{q_2} = \prod_{i=1}^{N_2} q_{2i}$$

$P_C = \left(1 - \overline{q_1}\right)\left(1 - \overline{q_2}\right)$ – вероятность безотказной работы всей системы.

Мажоритарная

Система работает с одной отказавшей подсистемой.



$P_C = 3P^2 - 2P^3$ – вероятность безотказной работы всей системы.

$P = 1 - q$ – вероятность безотказной работы одной подсистемы.

$Q_C \approx 3q^2, \quad если\, q \le 0.1$ – вероятность отказа всей системы.

Экспоненциальный закон распределения

$q_i = 1 - e^{-\lambda_i t}$ – вероятность отказа одного элемента (подсистемы).

$P_i = e^{-\lambda_i t}$ – вероятность безотказной работы одного элемента.

λ_i – интенсивность отказов *i*-го элемента.

t – время работы *i*-го элемента.

Основные определения и понятия теории надежности

Под системой в теории надежности принято понимать совокупность подсистем, функционально объединенных в соответствии с некоторым алгоритмом взаимодействия в процессе применения по назначению [3].

Программная ошибка имеет место тогда, когда программа работает не так, как предполагает пользователь, т.е. программная ошибка не является неотъемлемым свойством программного обеспечения. Наличие ошибки в программе – это функция от самой программы и от того, чего ждет от нее пользователь.

Применительно к сложным системам, надежность ПО определяется как свойство системы выполнять заданные функции, сохраняя во времени значения установленных эксплуатационных показателей в заданных пределах в соответствующих режимах и условиях обслуживания, ремонта, хранения. Свойства надежности проявляются в том, что система выполняет поставленные перед ней задачи.

Надежность ПО – это вероятность того, что программа какой-то период времени будет работать без сбоев, с учетом степени их влияния на выходные характеристики. Надежность ПО есть функция от ущерба, наносимого ошибкой пользователя.

Ошибки ПО являются функцией от входной информации и состояния системы. Ошибки несут систематический характер.

Надежность ПО базируется на понятиях корректности и устойчивости.

Программа считается корректной, если она выполняет запланированные действия и не имеет побочных эффектов. Корректность – узкое понятие, т.к. ее полная проверка для большинства программных систем практически невыполнима. Корректность базируется на тщательной спецификации требований пользователя. Возможна ситуация, когда программа корректна с точки зрения разработчика, но некорректна с точки зрения пользователя. Если ПО удовлетворяет своей спецификации, то оно корректно. Систему можно считать надежной, если велика вероятность того, что при обращении к ней можно получить требуемую услугу. Программа рассматривается как ненадежная, если она допускает сбои в особых ситуациях, даже если эти ситуации имеют небольшой процент от времени пользования системой.

Под устойчивостью программы понимается ее способность выполнять правильное действие при наличии отказов в работе аппаратуры и ошибках в исходных данных. При оценке устойчивости должны быть заданы параметры окружающей среды , по отношению к которым программа должна быть устойчивой. Обычно разработчик ПО не располагает средствами эффективного контроля за окружающей средой и его задача сводится к локализации нарушений или изменения характеристик окружающей среды и описанию способов их устранения.

Под надежностью ПО понимается свойство ПО выполнять предписанные функции в соответствии с требованиями заказчика при определенных условиях функционирования в течение заданного времени и обусловленное корректностью и устойчивостью.

Критерии надежности

Критерии надежности – это показатели, позволяющие оценивать предпочтительность тех или иных решений при создании и эксплуатации систем по степени достижения основных целей и с учетом затрат, при которых эти цели достигаются [4].

При исследовании надежности основная цель состоит в разработке эффективных методов и обеспечении длительной работоспособности систем с заданными функциональными характеристиками.

Основной задачей теории надежности остается оценка технических решений, позволяющих создать систему с заданными показателями надежности и заданными совокупными затратами.

Критерии:

- Детерминированные – оценка количества ошибок в программе на том или ином этапе работы.
- Вероятностные – вероятностная оценка свойств ПО.

Все ошибки носят детерминированный характер, но вероятностным оказывается процесс внесения ошибок. Проявление ошибок также носит вероятностный характер.

Примеры критериев:

- Корректность ПО.
- Число серьезных текущих ошибок в программе и время, необходимое для их устранения.
- Обслуживаемость системы – степень влияния ошибок ПО на обслуживаемость системы.
- Безопасность системы.
- Частота отказов.
- Вероятность безотказной работы за время t при условии времени отладки τ .
- Средняя наработка на программный отказ при условии исправления или неисправления обнаруженных отказов.

Верификация программ – процесс формального доказательства правильности программы, т.е. корректности.

Верификация:

- Статическая.
- Динамическая (частный случай – тестирование).

С учетом сложившейся практики выбора критериев оценки надежности необходимо принимать во внимание следующее:

1. Разработанное ПО в начальной стадии эксплуатации может потребовать менее жестких критериев и большего времени для его совершенствования.

2. После выпуска новой версии некоторое время потребуются также менее строгие критерии качества ПО.

3. Имеют место разбросы, вызываемые различием в условиях применения и использования ПО.

4. Эффективность работ по исправлению ошибок, проводимых разработчиками ПО, зависит от частоты проявления ошибок, что, в свою очередь, зависит от информации, поступающей от пользователей.

5. Необходимость соблюдения ограничений по быстродействию и производительности, если таковые диктуются пользователями.

Характеристики ПО

1. Количественные характеристики (оцениваются числом):
   - объем программы;
   - количество сопряжений;
   - количество ветвлений;
   - точки входа/выхода;
   - количество процедур;
   - уровень вложения;
   - количество комментариев;
   - количество страниц документации;
   - требуемое машинное время.

2. Качественные характеристики (оцениваются числом):
   - трудности проектирования;
   - трудности в эксплуатации из-за ошибок;
   - тип программ;
   - данные о персонале (количество, коэффициент загруженности).

3. Качественные характеристики как объективное суждение.

Испытания ПО

1. Стендовые (многократные проверки в реальных условиях).
2. Приемосдаточные (подтверждение требуемых характеристик и передача в эксплуатацию).
3. Системные (оценивают правильность взаимодействия с другими системами).
4. Демонстрация в реальных условиях.
5. Сертификационные испытания.

Основные параметры персонала

1. Данные, характеризующие программиста.
2. Данные, характеризующие выполнение конкретной работы.

В результате формируются критерии, учитывающие конкретного программиста в конкретной работе.

	Параметры программиста	**Оценка**	**Оцениваемые факторы**
А	Уровень знаний	1 – 5	1) ОП=А+Б+В+Г 2) ОПКР=(20-Д)*ОП ОП – оценка программиста ОПКР – оценка программиста и конкретной работы.
Б	Уровень способностей	1 – 5	
В	Стиль работы	1 – 5	
Г	Степень ответственности	1 – 5	
Д	Параметры конкретной работы	0 – 10	

Цель анализа программных ошибок при сертификации и оценке надежности ПО

Определение структурных свойств программ и характеристик процесса их разработки, влияющих на надежность.

Организация методов сбора данных о надежности.

Категоризация возможных программных ошибок по их причинам и проявлениям.

Выработка рекомендаций по совершенствованию бездефектных программ.

Разработка моделей надежности ПО.

Извещения об ошибке

Исходная информация об ошибках может быть представлена в извещении об ошибке. В нем указывается:

1. Объект затруднения (подсистема, БД, ОС и т.д.).
2. Дата и время ошибки.
3. Пример или задача, на которой зафиксирована ошибка.
4. Конфигурация активной структуры ПО.
5. Содержание ошибки.
6. В соответствие ставится извещение о закрытии ошибки, в котором содержится:
   - информация о закрытии ошибки;
   - генерация новой конфигурации;
   - правильность распознавания объекта затруднения;
   - существо ошибки.

Основные задачи в области надежности ПО (рис. 1.6)

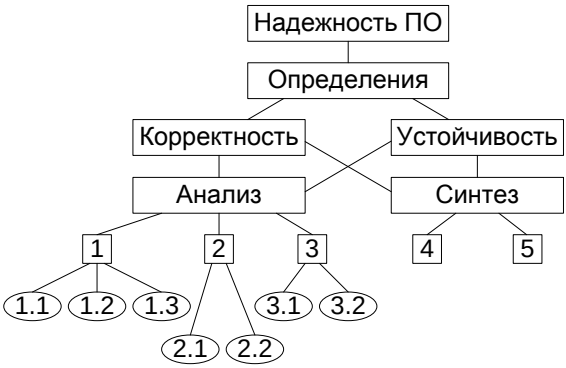


Рис. 1.6. Основные задачи в области надёжности ПО

1. Классификация ошибок.
   1.1. Организация систем сбора данных.
   1.2. Рекомендации по совершенствованию.
   1.3. Построение модели.
2. Верификация программ.
   2.1. Статическая верификация.
   2.2. Динамическая верификация.
3. Тестирование.
   3.1. Выбор тестов.
   3.2. Управление тестированием.
4. Защита информации.
5. Защита вычислительного процесса.

Количественные вероятностные характеристики надежности ПО

$P(t)$ – вероятность безошибочной работы.

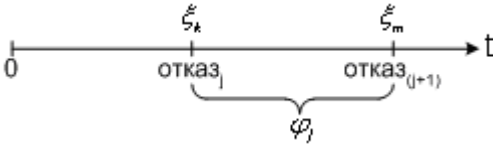
$Q(t)$ – вероятность появления ошибок.

$a(t) = Q'(t)$ – частота появления ошибок.

$\lambda(t)$ – интенсивность появления ошибок.

T_0 – среднее время между ошибками.

Рассмотрим методику вычисления основных параметров. Программа испытывается на одном компьютере.



ξ_k и ξ_m – время k-го и m-го прогонов, окончившихся отказами.
$N_{отк}$ – количество прогонов, окончившихся отказом.

$j \in \overline{1, N_{отк}}$ $\qquad \varphi_j = \sum_{l=k+1}^{m} \xi_l$, ξ_l – время l-го успешного прогона.

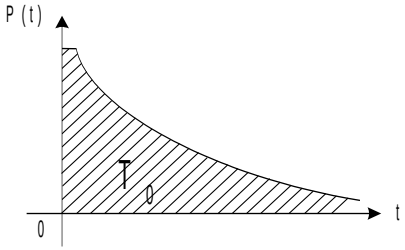
φ_j – время между j-м и (j+1)-м отказами.

$$T_0^* = \frac{\sum_{j=1}^{N_{отк}} \varphi_j}{N}$$ – статистическое среднее время между двумя ошибками, N – общее количество прогонов.

$$T_0 = \int_0^\infty P(t)dt$$ – среднее время между двумя ошибками.

$$P(t) = e^{-\lambda t}, \quad \lambda = const$$ – экспоненциальный закон надежности.

$$P(t) = e^{-\int_0^t \lambda(t)dt}$$ – обобщенный закон надежности ($\lambda \neq const$)



Классификация ошибок ПО

Основные проблемы классификации.

1. Где произошла ошибка?

1.1. Персонал.

Информация по категориям персонала включает структуру и процедуры. Это операционные процедуры, правила кодирования и проверки, а также стандарты документирования.

1.1.1. Структура.

- Технический – для конкретного модуля определяется имя, квалификация разработчика, а именно опыт программирования, знание языков и ответственность.
- Административный – определяет административную информацию.

1.1.2. Процедура.

- Операционные процедуры включают информацию о рабочей среде, т.е. пакетный или интерактивный режим работы, свободный или ограниченный доступ.
- Правила кодирования и проверки. Они содержат информацию о степени использования, например, структурного программирования.
- Стандарты документации включают форматы и процедуры документирования данного модуля.

1.2. Оборудование.

1.2.1. Компьютер.

Перечень оборудования и интерфейсов.

1.2.2. Связь.

Содержит информацию о внешнем оборудовании в комплекте ПК, включая линии связи с терминалами.

1.2.3. Сопровождающее обеспечение.

Информация обо всем оборудовании, для подготовки модуля к работе.

1.3. ПО.

1.3.1. Внутреннее ПО.

Языковой процессор, загрузчик, редактор связей, утилиты.

1.3.2. Применение.

Это размеры, смежные модули, область применения. Каждый из этих разделов обеспечивает идентификацию относящихся к ним программ по имени, номеру версии и т.д.

2. На что похожа ошибка?

2.1. ПО.

2.1.1. Внутреннее ПО.

ОС, редактор связей, загрузчик, утилиты.

2.1.2. Применение.

Каждый ошибочный элемент характеризуется именем, номером версии и будет специфицировать любой ошибочный код и любое диагностическое сообщение, являющееся следствием данной ошибки элемента.

2.2. Функции.

Категория функций идентифицирует с помощью имени точку, в которой зафиксирована ошибка. Эта точка может относиться к процедурам ввода/вывода или использования ресурсов.

2.2.1. Процедура.

В процедурах ввода/вывода подразумевается наличие неправильных значений данных.

2.2.2. Использование ресурса.

При использовании ресурсов наиболее критичными ошибками являются:

- неправильное использование терминальных устройств;
- ошибки синхронизации;
- ошибки в описании форматов вводимой и выводимой информации.

2.3. Ресурсы.

2.3.1. Имя.

2.3.2. Использование ресурса.

2.4. Область.

2.4.1. Структура программы.

2.4.2. Приложение.

3. Как была сделана ошибка?
   3.1. Данные.
      3.1.1. Входные.
      3.1.2. Внутренние.
   3.2. Процедуры.
      3.2.1. Вычисление.
      3.2.2. Контроль.
      3.2.3. Интерфейс.

4. Когда была сделана ошибка?
   4.1. Начальная разработка.
   4.2. Внедрение.
   4.3. Функционирование.

5. Почему произошла ошибка?
   5.1. Механические причины.
      5.1.1. Подстановка.
      5.1.2. Путаница.
      5.1.3. Пропуск.
   5.2. Умственные причины.
      5.2.1. Концептуализация.
      5.2.2. Реализация.
   5.3. Коммуникационные причины.
      5.3.1. Персонал.
      5.3.2. Документация.

Целью задачи классификации ошибок ПО является создание методов упорядочивания информации о программных ошибках с тем, чтобы на основе этой информации выяснить причины, породившие ненадежность и о путях их предупреждения.

1.4. Классификация моделей надежности ПО

Классификация моделей надёжности представлена на рис. 1.7.

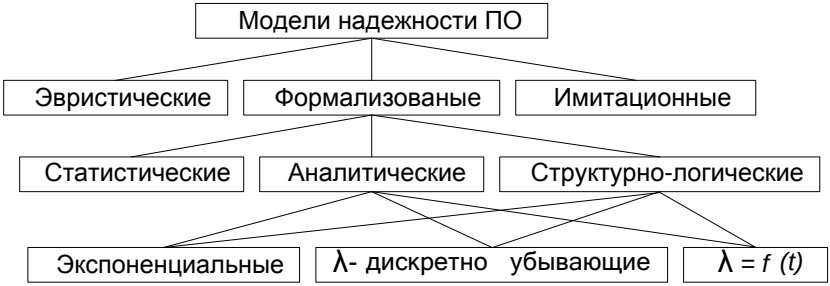


Рис. 1.7. Классификация моделей надёжности

Экспоненциальная модель (модель Шумана)

Вводится ряд допущений и условий, основным из которых является условие существования программы «исследователя системы». Остальные допущения и условия не связаны с какими-то специфическими свойствами ПО.

Условия сводятся к следующему:

1. Предполагается, что в начальный момент компоновки программных средств системы в них имеются ошибки (E – количество ошибок). С этого времени отсчитывается время отладки τ, которое включает затраты времени на выявление ошибок с помощью тестов, на контрольные проверки и т.д. При этом время исправного функционирования системы не учитывается. В течение времени τ выявляется $\varepsilon_0(\tau)$ ошибок в расчете на одну команду машинного языка. Таким образом, удельное число ошибок на одну машинную команду, остающихся в системе после времени τ работы, равно:

$$\varepsilon_\tau(\tau) = \frac{E}{I} - \varepsilon_0(\tau),$$

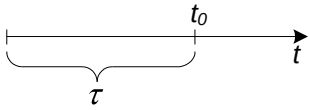
где I – общее число машинных команд.

2. Предполагается, что значение функции частоты или интенсивности отказов $z(t)$ пропорционально числу ошибок, оставшихся в ПО после израсходования на отладку времени τ, то есть:

$$z(t) = C\,\varepsilon_\tau(\tau),$$

где C – коэффициент пропорциональности.

Тогда, если время работы системы t отсчитывается от момента времени t_0, а τ остается фиксированным (τ =const), то функция надежности или вероятность безотказной работы на интервале времени от 0 до t есть



$$R(t,\tau) = \exp\left\{-C\left[\frac{E}{I} - \varepsilon_0(\tau)\right]t\right\}.$$

Для нахождения C и E используются принцип максимального правдоподобия (пропорция).

Модель Джелинского-Моранды

Модель с дискретным убыванием интенсивности отказов. В этой модели предполагается, что интенсивность ошибок описывается кусочно-постоянной функцией, пропорциональной числу неустраненных ошибок. Т.е. предполагается, что интенсивность отказов $\lambda(t)$ постоянна до обнаружения и исправления ошибки, после чего она опять становится постоянной, но с другим, меньшим, значением. При этом предполагается, что между $\lambda(t)$ и числом оставшихся в программе ошибок существует прямая зависимость

$$\lambda(t) = k(M - i) = \lambda_i,$$

где M – неизвестное первоначальное число ошибок;

i – число обнаруженных ошибок, зависящих от времени t;

k – константа.

Значения неизвестных параметров k и M может быть оценено на основе последовательности наблюдений интервалов между моментами обнаружения ошибок по методу максимального правдоподобия.

Статистическая модель Миллса

Эту модель можно использовать для сертификации программных средств.

В модели не используются предположения о поведении функции риска $\lambda(t)$. Эта модель строится на твердом статистическом фундаменте.

Сначала программа «засоряется» некоторым количеством известных ошибок. Эти ошибки вносятся в программу случайным образом, а затем делается предположение, что для ее собственных и внесенных ошибок вероятность обнаружения одинакова и зависит только от их количества. Тестируя программу в течение некоторого времени и отсортировывая собственные и внесенные ошибки можно оценить N – первоначальное число ошибок в программе.

Предположим, что в программу было внесено S ошибок. Пусть при тестировании обнаружено $(n+V)$ ошибок,
где n – число собственных ошибок,
V – число внесенных ошибок.

Тогда оценка для N по методу максимального правдоподобия будет следующей:

$$N = \frac{S \cdot n}{V}.$$

В действительности N можно оценивать после каждой ошибки. Миллс предлагает во время всего периода тестирования отмечать на графике число найденных ошибок и текущие оценки для N.

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

Примем, что в программе имеется не более k собственных ошибок и внесем в нее еще S ошибок. Теперь программа тестируется, пока не будут обнаружены все внесенные ошибки. Причем подсчитывается число обнаруженных собственных ошибок n. Уровень значимости C (мера доверия к модели) – вероятность того, что модель будет правильно отклонять ложные предположения и вычисляется по формуле

$$C = \begin{cases} 1, \textit{при } \mathrm{n} > \mathrm{k} \\ \dfrac{\mathrm{S}}{\mathrm{S}+\mathrm{k}+1}, \textit{при } \mathrm{n} \le \mathrm{k} \end{cases}.$$

Формулы для N и C образуют полезную модель ошибок.

Недостатки модели

Меру доверия C нельзя предсказать до тех пор, пока не будут обнаружены все внесенные ошибки, это может не произойти до самого конца этапа тестирования.

Модификация формулы для C, если не все ошибки обнаружены:

$$C = \begin{cases} 1, & \textit{при} \quad n > k \\ f(s, j, k), & \textit{при} \quad n \le k \end{cases},$$

где j – число найденных внесенных ошибок, j < S.

Еще один график, который полезно строить во время тестирования, – это текущее значение верхней границы k для некого доверительного уровня.

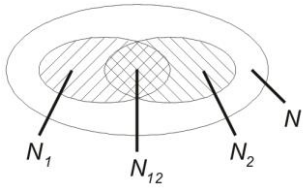
Модель математически проста и интуитивно привлекательна. Легко представить программу внесения ошибок, которая случайным образом выбирает модуль и вносит логические ошибки, изменяя или убирая операторы. Природа внесения ошибок должна оставаться в тайне, но все их следует регистрировать с целью последующего деления на собственные и несобственные.

Процесс внесения ошибок является самым слабым местом модели, поскольку предполагается, что для собственных и внесенных ошибок вероятность обнаружения одинакова, но неизвестна. Отсюда следует, что внесенные ошибки должны быть типичными, но на сегодня непонятно, какими именно они должны быть. Однако, по сравнению с проблемами других моделей, эта проблема кажется не очень сложной и разрешимой.

Простейшие интуитивные (эвристические) модели

Эти модели особенно эффективны для целей сертификации. Было разработано несколько моделей для оценки числа ошибок. Они основаны на более слабых предположениях, чем сложные модели.

Предполагается начать тестирование двумя независимыми группами. В течение некоторого времени производится параллельное тестирование системы, затем результаты собираются и сравниваются.



где N_1, N_2 – число обнаруженных каждой группой ошибок,

N_{12} – число ошибок, обнаруженных дважды (обеими группами),

N – число ошибок в программе (неизвестное),

E_1 и E_2 – эффективность тестирования каждой группой

$$E_1 = \frac{N_1}{N} \qquad E_2 = \frac{N_2}{N} \ .$$

Предполагаем, что возможность обнаружения ошибок одинакова. Это серьезное предположение, не лишенное смысла. Можно рассматривать каждое подмножество N, как аппроксимацию всего пространства.

Например, если первая группа обнаружила 10% ошибок, то она должна была найти примерно 10% всякого случайным образом выбранного подмножества, например подмножества N_2. Можно сказать, что:

$$E_1 = \frac{N_{12}}{N_2} \, .$$

Если выполнить подстановку для N_2 получим

$$N = \frac{N_{12}}{E_1 \cdot E_2} \, .$$

Очевидно, самый простой способ оценки числа ошибок – сравнить оценки, основанные на исторических данных, в частности на среднем числе ошибок, приходящихся на 1 оператор в предыдущих проектах. В литературе есть сведения о частоте ошибок, но они не очень обширны. Имеющиеся данные ориентированы по отраслям и

берутся как среднее значение по некоторому количеству операторов (например, 10 ошибок на 1000 операторов).

Пример: данные IBM для OS/360, OS/VS1, OS/VS2:

$$z = 23x + 2y,$$

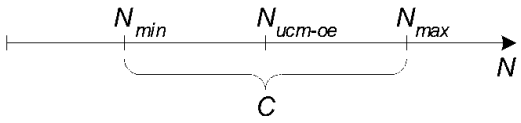
где x – многократно исправляемые модули или число модулей, которые потребовали 10 и более исправлений;

y – число модулей, потребовавших 1 или несколько исправлений;

z – полное число исправлений в модулях.

Из-за неопределенностей, пока самый разумный подход – воспользоваться несколькими моделями и объединить их результаты. Например, данные по прежним проектам можно использовать для грубой оценки. Далее можно использовать модель с двумя параллельными группами. Далее – тестирование с искусственным внесением ошибок, определить достоверность C по модели Миллса, а также использовать другие модели. Объединение результатов нескольких тестов:

1. Средняя величина N, $P(t)$.
2. Минимальные оценки.
3. Максимальные оценки.
4. Интервальные оценки.



Имитационные модели

Такие модели, которые имитируют процессы появления ошибок, процесс обнаружения ошибок, процесс исправления ошибок с точки зрения надежности ПО.

Часто программу представляют как последовательность узлов, дуг и петель ориентированного графа.

Узлы – точки, в которых части программы объединяются или разъединяются.

Дуги – последовательности линейных участков. В них размещаются команды.

1.5. Тестирование ПО. Испытания ПО

Принципы и задачи детерминированного тестирования
Детерминированное тестирование при проектировании

Основной задачей детерминированного тестирования является установление факта работоспособности программ и соответствие их техническому заданию, а также выявление и устранение ошибок и доведение характеристик программ до уровня требований, заданных заказчиком [7].

Диагностирование производится автоматическими человеко-машинными системами, в которых:

1. на долю человека приходится анализирующая роль в обнаружении ошибок, их анализ и принятие решений по их корректировке;
2. вычислительные системы обеспечивают исполнение программы, управление заданиями и тестами, селективное информирование о ходе тестирования.

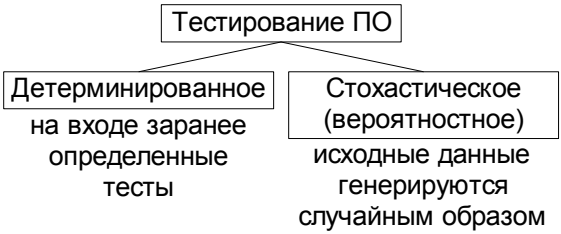


Рис 1.8. Классификация методов тестирования

При детерминированном тестировании (рис. 1.8) получаются результаты при фиксированном наборе исходных данных, а также сравнение этих значений с эталонными. Диапазон варьирования исходных данных и число вариантов сочетания переменных определяют достоверность отладки. Сравнение результатов исполнения с эталонами, как правило, происходит автоматически. Если результат отличается от эталона, определяется место и тип ошибки.

В зависимости от используемой при тестировании информации различают два метода:

1. Метод проверки по исходным данным и результатам.

Программа рассматривается как чёрный ящик, и после установления факта неработоспособности, на каком-то наборе исходных данных, используется информация о структуре программы, т.е. проверка от общего к частному с надеждой, что ошибки нет.

2. Метод с учётом промежуточных результатов.

Анализируются логические маршруты исполнения программы и промежуточные результаты в точках маршрута. Проверка программы осуществляется по ходу её исполнения от частного к общему с уверенностью, что ошибка есть. Метод – «белый ящик».

«Серый ящик» – частично структура известна, а частично – нет.

Метод белого ящика более прост при локализации ошибок, но требует больше времени, чем метод проверки по окончательным результатам.

Процесс отладки программы при детерминированном тестировании подразделяется на следующие этапы:

1) планирование отладки;
2) составление тестов и задач на отладку и исполнение программы;
3) информирование о результатах по задачам теста;
4) анализ результатов, обнаружение и локализация ошибок;
5) устранение ошибок и корректировка исходного текста программы.

Детерминированное тестирование включает:

1) выбор последовательности контрольных точек, входы и выходы из программы;
2) выбор наборов значений исходных данных;
3) выбираются промежуточные точки контроля и перечень переменных, подлежащих контролю в этих точках, маршруты исполнения программы.

Порядок тестирования может быть:

1) безусловный, т.е. независимый от результатов исполнения предыдущих наборов;
2) условный.

Методы тестирования

Рассмотрим основные методы, представленные на рис. 1.9.

Рис. 1.9. Методы тестирования

Восходящее тестирование

Программа собирается и тестируется снизу вверх, только модули самого нижнего уровня тестируются изолированно или автономно. После их тестирования вызов их должен быть также надёжен, как вызов встроенной функции языка. Затем тестируются модули, вызывающие уже проверенные. Эти модули тестируются не автономно, а вместе с уже протестированными.

При восходящем тестировании для каждого модуля нужен драйвер. Нужно подавать тесты в соответствии с сопряжением модуля. Одно из возможных решений – написать для каждого модуля небольшую ведущую программу, тестовые данные представляются как встроенные в эту программу переменные. Другое решение – использовать программу тестирования модулей, тесты написать на специальном языке и избавиться от необходимости писать драйвер.

Нисходящее тестирование (нисходящая разработка)

Изолированно тестируется только головной модуль. После завершения тестирования этого модуля с ним соединяются модули, непосредственно вызываемые им, и тестируется полученная комбинация. Процесс повторяется до тех пор, пока не будет собраны все модули.

Возникают две проблемы:

1. Что делать, когда тестируемый модуль вызывает модуль более низкого уровня, которого в данный момент еще не существует?

Для имитации функций недостающих модулей программируются модули-заглушки, которые моделируют функции отсутствующих модулей.

2. Как подаются тестовые данные?

Преимущества метода

1. Метод совмещает тестирование модулей, тестирование сопряжений и частично тестирование внешних функций.

2. Когда модули ввода/вывода уже подключены, тесты можно готовить в удобном виде.

Недостатки метода

1. Модуль редко тестируется досконально сразу после его подключения. Это связано с тем, что тестирование некоторых модулей требуют крайне сложных заглушек и пишутся простые заглушки, проверяя часть работоспособности модуля.

2. Этот подход может породить веру в возможность начать программирование и тестирование верхнего уровня программы до того, как вся программа будет полностью спроектирована.

Нисходящий подход выгоден в тех случаях, когда есть сомнения в осуществлении программы в целом.

Нормальный стиль проектирования структуры программы предполагает при окончании проектирования нижних уровней вернуться назад и поправить верхний уровень, внося в него усовершенствования и исправляя ошибки. Если головная часть программы запрограммирована и оттестирована, разработчик с трудом идет на его модификацию.

Модифицированный нисходящий метод

Еще один недостаток нисходящего метода – полнота тестирования. Также сложно проверять исключительные ситуации.

Модифицированный метод требует, чтобы каждый модуль прошел автономное тестирование перед подключением к програм-

ме. Этот метод решает перечисленные проблемы, но все равно нужны как драйверы, так и заглушки.

Метод большого скачка

Это самый распространенный подход в интеграции модулей. Каждый модуль тестируется автономно. По окончании тестирования модулей они интегрируются в систему все сразу.

Метод имеет несколько недостатков и мало достоинств. Для каждого модуля необходима заглушка и драйверы, модули не интегрируются до самого последнего момента, а это означает, что в течение долгого времени серьезные ошибки в сопряжениях могут оставаться необнаруженными.

Метод «сэндвича»

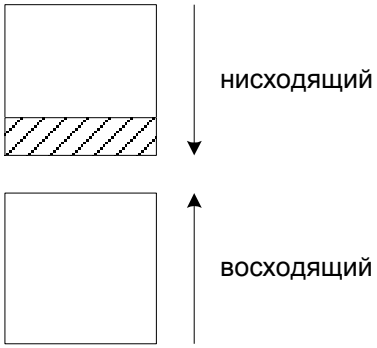
Это компромисс между восходящим и нисходящим тестированиями. При использовании этого метода одновременно начинают восходящее и нисходящее тестирования, собирая программу как сверху, так и снизу, встречаясь где-то посередине. Точка встречи зависит от конкретной программы и должна быть определена заранее.

Метод сохраняет достоинства восходящего и нисходящего подходов – начало интеграции системы на самом раннем этапе. Т.к. вершина программы вступает в строй рано как в нисходящем методе, на самом раннем этапе получается работающий каркас программы. Нижние уровни программы создаются восходящим методом, при этом снимаются те проблемы нисходящего метода, которые были связаны с невозможностью тестирования в глубине программы.

Модифицированный метод «сэндвича»

При тестировании методом «сэндвича» возникает также проблема, что и при нисходящем подходе, хотя здесь она стоит не так остро. Проблема в том, что невозможно досконально тестировать отдельные модули. Восходящий этап тестирования по методу «сэндвича» решает эту проблему для нижних уровней, но она может остаться для нижней половины верхней части программы. В модифицированном методе «сэндвича» нижний уровень тестирует-

ся строго снизу вверх, а модули верхних уровней тестируются изолированно, а затем собираются нисходящим методом. Т.о. модифицированный метод «сэндвича» также представляет собой компромисс между нисходящим и восходящим подходами.



Список литературы к главе 1

1. Майерс Г. Надежность ПО, М.: Мир, 1981.
2. Тейер Т., Липов М., Нельсон Э. Надежность ПО. М.: Мир, 1981.
3. Липаев В.В. Надежность ПО АСУ. М.: Энергоатомиздат, 1981.
4. Шураков В.В Надежность ПО систем обработки данных: Учебник. М.: Финансы и статистика, 1987.
5. Мамиконов А.Г., Кульба В.В., Шелков А.Б. Достоверность, защита и резервирование информации в АСУ. М.: Энергоатомиздат, 1986.
6. Шураков В.В. Обеспечение сохранности информации в системах обработки данных (по данным зарубежной печати): Учебное пособие. М.: Финансы и статистика, 1985.
7. Липаев В.В. Тестирование программ. М.: Радио и связь, 1986.

ВВЕДЕНИЕ К ГЛАВАМ 2 И 3

Для разработки моделей надежности систем необходимо провести обзор реальных разработок высоконадежных информационно-вычислительных и коммуникационных систем и рассмотреть основные критерии их надежности. Приводится аналитический обзор методов обеспечения надежности, используемых в реальных разработках высоконадежных информационно-вычислительных систем (ИВС) и систем передачи данных.

В настоящее время для повышения надежности информационно-вычислительных систем (ИВС) на практике широко используется резервирование замещением (РЗ) и постоянное резервирование (ПР). Для РЗ характерно обнаружение и отключение отказавших и подключение работоспособных резервных блоков. РЗ применяется тогда, когда в работе системы допустимы перерывы для обнаружения отказавших и подключения резервных блоков. Из известных примеров можно отметить многопроцессорную систему ILLIAC-IV [4], которая содержит 64 рабочих процессора и один резервный, а также проекты SERF [5] и MECRA [6], где применяется скользящее резервирование на мелком уровне.

ПР характеризуется тем, что передача и прием информации ведутся параллельно по нескольким каналам сразу, а исправление ошибок, возникающих из-за отказов и сбоев, осуществляется за счет внутренних свойств системы, без отключения отказавших блоков. При этом исправление ошибок происходит автоматически, без перерывов в работе системы с помощью специальных восстанавливающих органов (например, мажоритарных элементов) или за счет организации связей между избыточными элементами (как, например в ДОТ-логике). Здесь невозможно выделить основные и резервные блоки. Метод ПР широко используется в тех системах, где недопустимы даже кратковременные перерывы в работе, например в бортовых ЭВМ на самолетах и космических аппаратах. Примерами могут служить бортовой вычислитель на космических кораблях для исследования планет солнечной системы SATURN [7] и система управления подачей топлива в самолете Boeing [8].

В настоящее время в связи с увеличением степени интеграции и уменьшением габаритов компонентов ИВС появилась возможность использования кратности резервирования 4 и более. При этом остро встает вопрос о рациональном построении таких устройств. Кроме того, появились ИВС с новыми специфическими требованиями, для которых использование либо РЗ, либо ПР во многих случаях не позволяет удовлетворить требования к надежности, а именно – высокую долговечность, конструктивные ограничения на минимальный уровень резервирования, непрерывность функционирования, простоту контроля и диагностики. Это вызвано теми недостатками, которыми обладает каждый из методов структурной избыточности. Для РЗ характерно наличие перерывов в работе для отключения отказавшего и подключения резервного блока и, как правило, сложность и трудоемкость разработки методов контроля и диагностики. К недостаткам ПР можно отнести неэффективность использования избыточного оборудования и невысокие характеристики долговечности.

Возникшее противоречие привело к появлению новых решений в области структурной избыточности, одним из которых является комбинированное резервирование (КР) [9, 10]. КР – это метод повышения надежности за счет совместного использования методов ПР и РЗ. В системах КР появляется возможность сочетания преимуществ технологии РЗ (гибкость схем резервирования) и технологии ПР (быстродействие процессов восстановления информации в случае возникновения ошибок).

Системы с РЗ могут быть реализованы в форме фиксированного резервирования, в рамках которого резервные блоки могут замещать только определенные рабочие блоки системы. В рамках скользящего резервирования любой резервный блок может заменять любой отказавший рабочий блок системы с РЗ.

Системы с ПР могут иметь восстанавливающий орган, например мажоритарный элемент, реализующий голосование информации, поступающей на его входы. Системы с ПР без восстанавливающего органа встречаются на практике реже. К ним относятся, например, системы, построенные на логике с переплетениями [16].

Системы с КР (появившиеся сравнительно недавно) могут быть реализованы по гибридному принципу, в рамках которого мажори-

тарное ядро системы сочетается с резервными блоками по скользящей технологии. Итеративные системы с КР основаны на том, что резервированными компонентами системы являются уже зарезервированные конфигурации. Общеизвестным примером такого подхода является контроллер технологических процессов фирмы Siemens, состоящий из дублированной пары двух дублированных контроллерных плат [22]. Системы с КР, построенные по смешанному принципу, включают в себя различные схемы введения структурной избыточности, то есть их блоки резервируются различными способами.

Рассмотрены основные методы введения избыточности в рамках резервирования замещением, постоянного резервирования и комбинированного резервирования. Для проведения обзора по высоконадежным системным разработкам использовались источники [4-9, 12-32, 34-38].

При оценке надежности программного обеспечения важной задачей является выбор и обоснование показателей качества ПО, которым должно быть уделено особое внимание. Выбор этих показателей должен проводиться на основе требований, выдвигаемых действующими нормативными документами, а также специфики области использования программных средств. Поэтому в первую очередь должен быть проведен анализ этих вопросов.

Решение проблемы обеспечения заданного уровня надежности является решением задачи синтеза и аналогично этой задаче для аппаратной части информационной системы. Оно призвано обеспечить заданный уровень надежности ПО на этапе его разработки. Вторая часть проблемы касается надежности уже существующего программного обеспечения, установленного на реально функционирующем комплексе. Эта задача имеет место в случае, когда требуется оценить надежность комплекса в целом, выделяя из общих характеристик вклад аппаратной и программной составляющих. Данная задача является задачей анализа данных, полученных в реальной системе на протяжении определенного периода ее работы с целью оценить существующие ее параметры надежности и предсказать поведение системы в будущем.

Используя термин “надежность ПО” аналогично термину “надежность аппаратуры”, необходимо помнить, что “отказы ПО” как

результат проявления ошибок имеют совершенно другую физическую природу, чем отказы техники. Это вызывается тем, что отказы аппаратуры вызываются физическими особенностями реализации конкретного модуля, в то время как главный источник программных отказов – это ошибки проектирования, которые проявляются в каждом экземпляре программы. Это вызывает необходимость учета специфики программного обеспечения при использовании методов надежностного проектирования и анализа. Программные средства, с точки зрения теории надежности, относятся к восстанавливаемым изделиям. Соответственно, должны быть получены и проанализированы основные показатели надежности восстанавливаемых систем, исходя из их применения к программным средствам.

В пособии представлены модели надежности программного обеспечения типового направления системы передачи данных Кризисного центра и проведения анализа надежности на основе имеющейся статистической информации об отказах и восстановления программных средств. Определен состав программных модулей, составляющих систему программного обеспечения, определены модули, которые можно рассматривать в качестве дублирующих друг друга, и определены режимы их использования. Эти вопросы существенно сказываются на выборе необходимого аппарата расчета показателей надежности. При этом следует учесть специфику работы программных средств Кризисного центра, особенно их высокую надежность, которая не позволяет собрать большую статистику по отказам.

При разработке модели надежности программного обеспечения, так же как и для аппаратных средств, должны быть определены допущения, в пределах которых она строится, и ограничения на её использование.

Необходимо также наметить основные направления развития и уточнения полученной модели.

Приводятся некоторые рекомендации по созданию обобщенной аппаратно-программной модели надежности типового направления системы передачи данных Кризисного центра.

ГЛАВА 2. МЕТОДЫ РЕЗЕРВИРОВАНИЯ АППАРАТНЫХ СРЕДСТВ СИСТЕМЫ ПЕРЕДАЧИ ДАННЫХ АЭС

2.1. Аналитический обзор высоконадежных систем передачи данных

На рис. 2.1 представлена классификация методов резервирования устройств систем передачи данных (СПД). Рассмотрим основные методы введения избыточности.

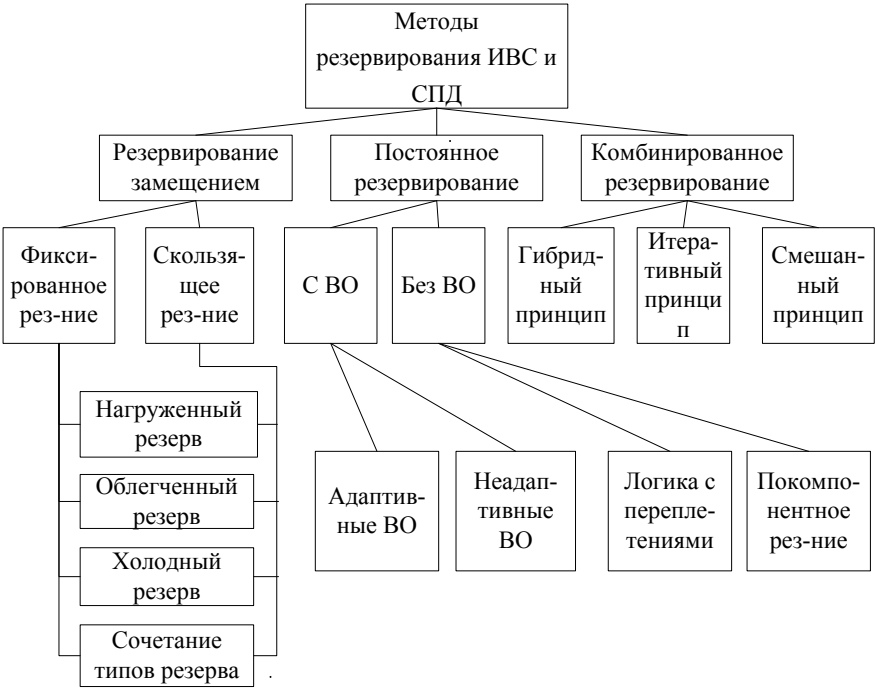


Рис. 2.1. Методы резервирования

Фиксированное резервирование

Для фиксированного резервирования характерно однозначное соответствие каждого резервного блока только одному рабочему блоку, который резервный блок может заместить в случае отказа рабочего. Данный способ широко используется на практике, например, в вычислительном комплексе МАРШРУТ-1, для повышения надежности которого уже в начале 70-х годов применялось фиксированное резервирование кратностью 3 [1]. Другие примеры фиксированного резервирования можно найти в [2] и [39].

Скользящее резервирование

Скользящее резервирование (СР) можно определить как способ резервирования группы основных блоков группой резервных, когда любой из резервных блоков может быть подключен вместо любого отказавшего из основных блоков. Интерес к СР в последнее время возрастает, поскольку этот метод позволяет наиболее эффективно использовать избыточное оборудование [10]. Это вызвано появлением интегральных схем с повышенной степенью интеграции (в частности, СБИС), а также ЭВМ с развитым программным обеспечением, что позволяет использовать программные методы локализации отказавших блоков и подключения резервных.

В [5] описывается ЭВМ повышенной надежности SERF, разработанная в США. Одно из требований к данной ЭВМ заключалось в том, чтобы в течение пяти лет вероятность полного сохранения ее работоспособности составляла не менее 95%. В лабораториях Франции разработана экспериментальная микропрограммная ЭВМ MECRA [6]. В [4] рассматриваются различные стратегии подключения резервного процессора в многопроцессорной системе ILLIAC-IV. В [13-15] описаны конкретные реализации СР.

Постоянное резервирование с восстанавливающим органом

Восстанавливающие органы (ВО) могут быть двух типов – адаптивные и неадаптивные. Для первых имеет место изменение функции восстановления по мере накопления отказов в избыточной структуре. Для вторых функция восстановления остается неизмен-

ной в течение всего времени работы устройства [16]. Вопросы построения избыточных устройств с ВО впервые были исследованы Дж. Нейманом [17]. Наибольшую известность и распространение получил мажоритарный метод, который, в частности, используется для повышения надежности микропроцессоров [18]. В [19] рассматриваются мажоритарные ЭВМ с шинной организацией. ВО в таких системах могут работать по принципу большинства, а также передавать информацию от любого неизбыточного канала.

Постоянное резервирование без восстанавливающего органа

ПР без ВО обычно в литературе делят на покомпонентное резервирование и логику с переплетениями. Покомпонентное резервирование используется для повышения надежности устройств на уровне логических элементов. Этот тип резервирования впервые был описан в работах Э. Мура и К. Шеннона [40]. К этому классу можно отнести ДОТ-логику [20], а также радиальную логику. В методах логики с переплетениями исправление ошибок осуществляется теми же логическими элементами, которые выполняют логические функции устройства. Резервирование данного типа было предложено Дж. Трайоном [21]. На практике широко распространена кратность резервирования, равная 4, поэтому метод получил название “учетверенная логика”. У. Пирс распространил эту концепцию на произвольные типы логических элементов [16].

Методы ПР широко используются на практике, однако для реализации на их базе долговечных устройств необходимо уменьшать уровень их резервирования, что составляет проблему для систем с модульным принципом построения. В таких системах трудно сделать уровень резервирования меньше модуля без существенных затрат на проектирование.

Определение 2. Гибридным принципом называется принцип, который основан на сочетании двух методов резервирования в одном избыточном блоке [9]. Комбинация ПР с ВО и фиксированного резервирования известна под названием гибридного резервирования [22].

Принцип работы устройства с гибридным резервированием (ГР) представлен на рис. 2.3, б. Нечетное число блоков устройства образует ядро устройства с ГР. Устройство имеет некоторое число резервных блоков, на практике 1-3, каждый из которых в общем случае может замещать любой из блоков ядра. Блоки, образующие ядро, работают на восстанавливающий орган (ВО на рис. 2.3 обозначен В). Для работы устройства с ГР необходим детектор рассогласования (ДР), который сравнивает сигнал с ВО и сигналы с каждого из блоков ядра. Если эти значения не совпадают, то ДР выдает сигнал в схему переключения (П), которая отключает отказавший и подключает резервный блок. Отключение отказавшего и подключение резервного блоков происходит без перерыва в работе вычислительного устройства.

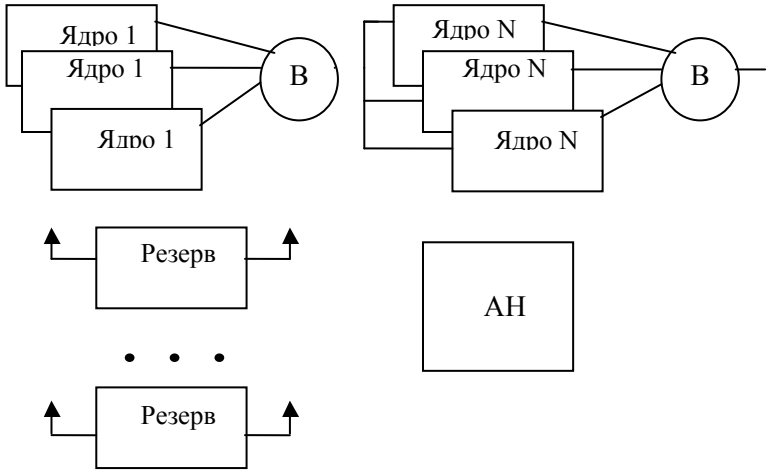
Метод резервирования, основанный на комбинации ПР с ВО и СР, известен как параллельно-гибридное резервирование (рис. 2.3,а). Неизбыточное устройство содержит N однотипных блоков, в избыточном устройстве имеется N ядер (для предотвращения перерывов в работе при отказе любого из блоков), а также резервные блоки, которые по скользящему принципу могут замещать любой из блоков каждого из N ядер. В этом методе наиболее эффективно используются резервные блоки. Устройства с КР на основе гибридного принципа различаются следующими факторами:

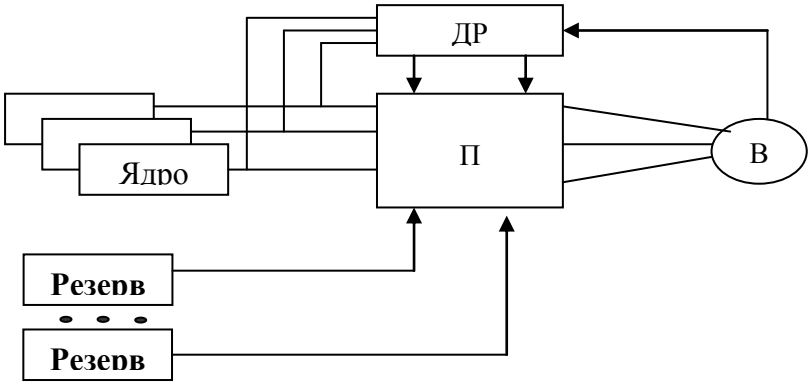
1) тип ВО,
2) стратегии подключения резервных блоков,
3) метод борьбы со сбоями,
4) метод резервирования автомата надежности.

При работе реальных устройств для борьбы со сбоями используются следующие решения.

1. Выход каждого элемента ДР подключается к специальному счетчику, который подсчитывает количество сигналов несовпадения, выдаваемых детектором. Если количество сигналов несовпадения за заданный цикл контроля превысит порог (на практике

а)





б)

Рис.2.3. Принципы работы устройства с гибридным резервированием

Определение 2. Гибридным принципом называется принцип, который основан на сочетании двух методов резервирования в одном избыточном блоке [9]. Комбинация ПР с ВО и фиксированного резервирования известна под названием гибридного резервирования [22].

Принцип работы устройства с гибридным резервированием (ГР) представлен на рис. 2.3, б. Нечетное число блоков устройства образует ядро устройства с ГР. Устройство имеет некоторое число резервных блоков, на практике 1-3, каждый из которых в общем случае может замещать любой из блоков ядра. Блоки, образующие ядро, работают на восстанавливающий орган (ВО на рис. 2.3 обозначен В). Для работы устройства с ГР необходим детектор рассогласования (ДР), который сравнивает сигнал с ВО и сигналы с каждого из блоков ядра. Если эти значения не совпадают, то ДР выдает сигнал в схему переключения (П), которая отключает отказавший и подключает резервный блок. Отключение отказавшего и подключение резервного блоков происходит без перерыва в работе вычислительного устройства.

Метод резервирования, основанный на комбинации ПР с ВО и СР, известен как параллельно-гибридное резервирование (рис. 2.3,а). Неизбыточное устройство содержит N однотипных блоков, в избыточном устройстве имеется N ядер (для предотвращения перерывов в работе при отказе любого из блоков), а также резервные блоки, которые по скользящему принципу могут замещать любой из блоков каждого из N ядер. В этом методе наиболее эффективно используются резервные блоки. Устройства с КР на основе гибридного принципа различаются следующими факторами:

1) тип ВО,
2) стратегии подключения резервных блоков,
3) метод борьбы со сбоями,
4) метод резервирования автомата надежности.

При работе реальных устройств для борьбы со сбоями используются следующие решения.

1. Выход каждого элемента ДР подключается к специальному счетчику, который подсчитывает количество сигналов несовпадения, выдаваемых детектором. Если количество сигналов несовпадения за заданный цикл контроля превысит порог (на практике

примерно 4-8), то счетчик выдает сигнал на подключение резервного блока вместо отказавшего.

2. Схема переключения строится таким образом, что она подключает резервный блок при первом сигнале ДР, однако после того, как все резервные блоки исчерпаны, и приходит сигнал от ДР, схема подключает блок, отказавший первым и т.д. Таким образом, если в рабочем блоке произошел сбой, он снова будет включен в работу.

3. Возможны модификации этих способов [23].

4. Наиболее слабым местом, с точки зрения надежности, в устройствах на основе гибридного принципа является автомат надежности: схемы ДР и П. Для того, чтобы повысить надежность ДР и П предлагается использовать введение избыточности: дублирование, мажорирование и специальные методы [24].

В последние десятилетия гибридное резервирование используется на практике при реализации систем ответственного целевого назначения. Так, в лаборатории JPL в США создана специализированная ЭВМ STAR со средствами самопроверки и последующего самовосстановления, в которой используется ГР [25]. Весьма существенным требованием при ее разработке являлся длительный период (порядка 10 и более лет) безотказной работы ЭВМ в процессе исследования планет солнечной системы. В ЭВМ имеется специальный процессор проверки и восстановления. Три экземпляра этого процессора работают на мажоритарный элемент, а еще два находятся в ненагруженном резерве. В [39] описывается специализированная ЭВМ для космического самолета SPACE SHUTTLE. ЭВМ обеспечивает сбор, обработку, хранение, визуальное отображение и распределение информации между подсистемами космического корабля. Основная концепция состояла в том, что в контуре управления полетом необходимо было использовать стандартные ЭВМ. Во время критических фаз полета четыре из пяти однотипных ЭВМ системы осуществляют идентичные вычисления задач наведения, навигации и управления с голосованием результатов, образуя, таким образом, избыточную структуру, в которой обеспечивается эффективное обнаружение первых двух отказавших ЭВМ. В [26] представлен мультипроцессор с ГР, управление конфигурацией которого осуществляется специальными резервированными моду-

лями. В [27] рассматривается вычислительное устройство с ГР, в котором основное оборудование учетверено. В [28] описан мультипроцессор FTMP с параллельно-гибридным резервированием.

Определение 3. Итеративным принципом называется принцип, основанный на резервировании уже зарезервированного блока различными методами, вложенными друг в друга [6].

КР на основе итеративного принципа может характеризоваться:

1) различными методами, которые используются при резервировании блока,
2) различными способами подключения резервного оборудования.

Один из примеров устройства с КР на основе итеративного принципа из четырех контроллеров приведен на рис. 2.4. Данная структура используется для управления технологическими процессами на предприятиях SIEMENS.

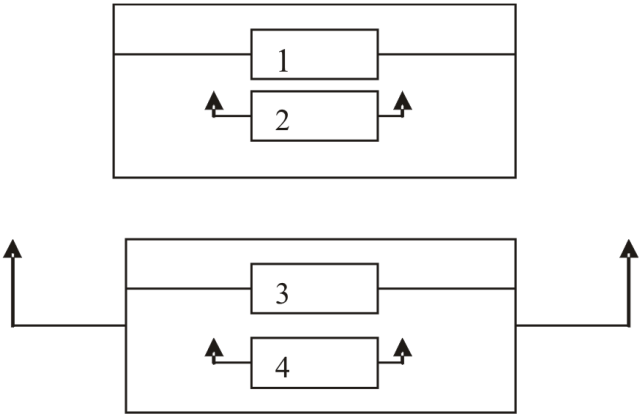


Рис.2.4. Примеры устройства с КР на основе итеративного принципа из четырех контроллеров

Для повышения надежности блока используется метод замещения дублированием (фиксированное резервирование), а сам дублированный блок, в свою очередь, дублируется с помощью такого же дублированного блока. В режиме долговечной работы такая конфигурация работает точно так же, как и учетверенная схема с за-

мещением, преодолевая последствия отказа трех неизбыточных блоков из четырех. В режиме быстрого самоконтроля дублированные блоки используются для быстрого выявления ошибок на рабочих шинах и переключения на резервное оборудование в условиях жестких временных ограничений на процедуру приостановки технологического процесса. Ряд известных реализаций итеративного принципа резервирования представлены также в [23, 30].

Определение 4. Смешанным принципом называется принцип, основанный на резервировании разных блоков системы различными методами [6].

КР на основе смешанного принципа характеризуется:

1) различными методами, которые используются при резервировании различных блоков. В настоящее время, в связи с повышением требований к надежности ЭВМ, все большее распространение получает сочетание нескольких методов,

2) различными способами подключения резервного оборудования – фиксированный, скользящий и т.д.

При проектировании ЭВМ с КР на основе смешанного принципа необходимо учитывать блоки сопряжения между избыточными устройствами, которые преобразуют избыточные сигналы одного избыточного устройства во входные сигналы соседнего с ним. Необходимость учета блоков сопряжения при проектировании избыточных СПД, особенно на этапе выбора их надежностной конфигурации, является одной из причин выделения такого принципа построения избыточных структур в отдельный принцип. Этот принцип широко используется на практике. Так, в цифровой системе стабилизации орбитальной космической станции SKYLAB [30, 39] основное оборудование удвоено, а для повышения надежности регистра, содержащего критические параметры системы, используется резервирование кратности 3. Уже рассмотренные специализированные ЭВМ, как STAR и SHUTTLE, также могут быть отнесены к системам, в которых используется КР на основе смешанного принципа.

Восстанавливаемые системы передачи данных

Использование резервирования информационно-вычислительных систем (ИВС или СПД) и устройств на различных уровнях

подразумевает восстановление систем после ошибок, вызванных аппаратными неисправностями (сбоями, отказами) с точки зрения восстановления, как ошибочной информации, так и неисправных аппаратных средств. В различных работах и публикациях предложены или подразумеваются различные определения восстановления системы.

В ряде работ (например, [33, 34]) под восстановлением ИВС понимается исправление только системной информации (обрабатываемой, передаваемой, выводимой, управляющей), и в этом смысле восстановление относится только к устранению последствий неисправностей и эквивалентно восстановлению системы после сбоев.

В некоторых публикациях (например, в [36]) оба типа восстановления входят в качестве отдельных этапов в общую процедуру послеаварийного восстановления системы.

Для специализированных резервированных ИВС оказывается удобным четко разделить эти два понятия, поскольку избыточная ИВС может быть восстанавливаемой на общем системном уровне (посредством подключения резервных блоков или реконфигурации) и не является восстанавливаемой на уровне отдельных ее блоков или элементов.

Исходя из этого, предложена классификация восстанавливаемых ИВС по степени и форме проводимого восстановления (ремонта) аппаратных технических средств, представленная на рис. 2.5. Классификация построена на разделении восстанавливаемых ИВС на полностью ремонтируемые, частично ремонтируемые, с несколькими режимами работы и неремонтируемые.

Полностью ремонтируемые ИВС представляют собой, главным образом, восстанавливаемые системы наземного типа с достаточным запасом времени на ремонт и подключение резервного оборудования. Системы реального времени могут представлять собой полностью ремонтируемые ИВС в режиме хранения, подготовки к работе или ожидания.

Частично ремонтируемые системы представляют собой класс восстанавливаемых ИВС, подвергаемых профилактическому контролю, в процессе которого происходит не полное восстановление оборудования, а его частичная настройка (юстировка параметров),

превентивная замена компонентов, устранение загрязнения и других потенциальных источников отказов. К частично восстанавливаемым системам относятся также системы, восстанавливаемые (ремонтируемые) на уровне не целых функциональных блоков, а отдельных элементов [41].

Рис.2.5. Классификация ИВС с восстановлением

Системы со смешанным режимом работы построены на сочетании двух режимов работы: ремонтируемого и неремонтируемого. Сочетание может осуществляться по времени (например, при чередовании интервалов наличия и отсутствия средств или возможностей ремонта аппаратуры), либо по уровню (например, при восстановлении системы на уровне отдельных блоков и отсутствии восстановления на общесистемном уровне). Последний тип систем характерен для систем с катастрофическими последствиями отказов, то есть для систем, отказ которых недопустим, так как связан с огромными материальными потерями или человеческими жертвами. К таким системам относятся системы управления ядерными

реакторами, большими промышленными технологическими комплексами и системами вооружений.

Неремонтируемые восстанавливаемые ИВС - это, главным образом, бортовые цифровые вычислители, предназначенные для управления или информационного обеспечения полета авиационных и космических средств, а также системы управления ответственными объектами в реальном времени. Восстановление таких систем происходит только за счет переключения резервных блоков, реконфигурации и специальных программных средств.

Из программных средств восстановления наиболее известен метод восстановления с откатом (к контрольным точкам программы или процесса), описанный, например, в [34].

Контроль и диагностика систем передачи данных

Для реализации преимуществ резервирования и восстановления ВС необходима достаточно эффективная система контроля и диагностики неисправностей. Различают несколько видов контроля ВС [32, 33]: оперативный и тестовый, аппаратный и программный, диагностический и профилактический и т.д. Качество организации контроля определяет, в конечном счете, эффективность использования избыточной аппаратуры ИВС, ее живучесть и способность отвечать требованиям к быстродействию, особенно для систем реального времени.

В большинстве резервированных ИВС контроль организован за счет использования их избыточности [10, 16, 28], то есть при помощи сравнения одноименных выходов рабочих и резервных системных блоков. Однако для повышения живучести ИВС и эффективности используемого избыточного оборудования в последнее время все более широко применяются методы, основанные на самоконтроле и самотестировании вычислительных устройств [42, 43]. В информационных системах и больших базах данных при этом широко используются методы, основанные на сочетании копирования информационных файлов с организацией кворумов.

В процессе надежностного проектирования ИВС немаловажное место занимает проектирование системы ее контроля и диагностики. Как показывает практика, разработка системы контроля и диагностики обычно включает в себя решение следующих задач: выбор

метода контроля, выбор структуры системы контроля (СК), определение контролируемых параметров или выходов проектируемого устройства, определение оптимального периода контроля (для периодического контроля).

Решение двух первых задач носит достаточно общий характер, решается на ранних этапах эскизного проектирования изделия ВТ и обусловлено общими (зачастую качественными) критериями достоверности работы устройства. На более поздних этапах проектирования выбранные варианты построения СК могут анализироваться с учетом конкретной структуры и функционально-логической схемы устройства и с применением конкретных количественных критериев надежности и достоверности обработки информации [44-46].

Решение двух последних задач производится на более поздних этапах проектирования устройства и определения режима его эксплуатации. В [47-49] рассмотрена задача выбора контрольных точек применительно к математическим объектам (выделение внутренних переменных), к многоэлементным объектам управления (выбор контролируемых параметров) и к программным комплексам реального времени.

В [32, 35, 50, 51] рассмотрено решение задачи выбора оптимальных параметров периодического контроля, реализуемого в форме проверок, предупредительных замен, переключения резерва или профилактического технического обслуживания.

Общеизвестно, что введение средств аппаратного контроля в неизбыточную систему, с одной стороны, снижает характеристики ее аппаратной безотказности (поскольку связано с добавлением избыточного оборудования, например, дополнительных контрольных разрядов), а с другой стороны, повышает достоверность обработки информации при наличии сбоев или отказов в системе [10, 46]. В случае проектирования избыточных резервированных систем введение средств контроля позволяет повысить характеристики функциональной безотказности системы [45], поскольку обеспечивают эффективную и оперативную информацию о состоянии ее отдельных блоков и компонентов, а, следовательно, влияют на скорость восстановления системы в случае сбоев и отказов.

2.2. Выбор критериев надежности восстанавливаемых систем передачи данных

Приведем расширенный обзор критериев, которые могут быть использованы при проектировании СПД ответственного целевого назначения.

Коэффициент готовности. Коэффициент готовности *Кг* является самым универсальным показателем надежности восстанавливаемых систем. Он характеризует соотношение времени работоспособного состояния систем и времени ее простоев (отказовых состояний). Если за некоторый промежуток времени *Т* система находилась в рабочем состоянии время *Тр*, а режиме восстановления (простоя) – *Тв*, то коэффициент готовности определится как отношение

$$Кг = \frac{Тр}{Тр + Тв}. \qquad (2.1)$$

Если использовать приближенный метод усреднения параметров, то под значениями *Тр* и *Тв* можно понимать средние значения данных параметров, то есть *Тр* – среднее время безотказной работы системы, а *Тв* – среднее время ее восстановления после отказа.

Заметим, что данный критерий целесообразно использовать для анализа систем, имеющих в течение времени *Т* непрерывный режим работы.

Довольно распространенным в настоящее время является тип систем, работающих в режиме готовности к решению целевой задачи. Такие системы в течение времени *Т* ожидают приход задачи (запроса), а при появлении задачи занимаются ее оперативным решением, как правило, в реальном режиме времени. Для анализа таких систем целесообразно использовать оперативный коэффициент готовности.

Коэффициент простоя (неготовности). Значение данного критерия определяется как *Кнг=1 – Кг*.

Оперативный коэффициент готовности. Оперативный коэффициент готовности *Кго* используется для анализа надежности систем с нерегулярным режимом работы. Его значение можно записать в виде:

$$Кго = Кг\ P(t)\ , \qquad (2.2)$$

где *Кг* – коэффициент готовности системы, рассчитанный по формуле 2.1, а *P(t)* – вероятность решения целевой задачи за время *t*.

Вероятность безотказной работы системы за время *t*. Для восстанавливаемых систем интервал времени отсчитывается от времени последнего восстановления. Если после отказов система восстанавливается полностью, то интервал времени *t* отсчитывается от времени любого восстановления системы и, в принципе, данный критерий аналогичен вероятности безотказной работы, характеризующей невосстанавливаемые объекты.

Наработка на отказ. Наработка на отказ определяется как среднее время между восстановлением системы и ее следующим отказом. Если в процессе восстановления система восстанавливается полностью, то данная характеристика аналогична среднему времени безотказной работы для невосстанавливаемых объектов.

Среднее время восстановления системы. Данный критерий характеризует усредненное значение времени восстановления системы после каждого ее отказа.

Другие критерии. Кроме этих, выделяется ряд специальных показателей, позволяющих оценивать надежность систем специфического целевого назначения.

Вероятность функционально безотказной работы. Для автоматизированных систем управления, работающих в реальном времени в [10], предлагается специфический показатель – вероятность функционально безотказной работы системы за заданное время *t*, позволяющий оценить качество информации о текущем состоянии системы. Значение критерия в данном случае вычисляется как сумма вероятности безотказной работы системы *P(t)* и произведения вероятности отказа *(1-P(t))* на коэффициент качества контроля *K(t)*.

При определенных допущениях [33, 38] коэффициент качества контроля можно интерпретировать как условную вероятность того, что в случае отказа системы соответствующие ошибки будут обнаружены (и/или устранены) на заданное время t_0.

Вероятность безотказной коммуникации. В соответствии с данным подходом аналогичный критерий можно предложить и для

коммуникационных систем типа систем передачи (СПД) ответственного целевого назначения. Данный критерий во многом аналогичен коэффициенту оперативной готовности *Кго* (формула 2.2) и формулируется следующим образом: вероятность того, что в произвольный момент времени оборудование системы находится в работоспособном состоянии (состоянии готовности к передаче срочной информации), а информация будет передана за заданное время t_0. Математически критерий, обозначенный как $P_Ф$, (при отсутствии корреляции процессов отказа и передачи данных по линии связи) запишется в виде:

$$P_Ф = Кг P(t_0) \ , \qquad (2.3)$$

где *Кг* - коэффициент готовности системы,

$P(t_0)$ - вероятность правильной (безошибочной) передачи информации за время t_0.

Очевидно, что значение $P(t_0)$ необходимо рассчитывать, исходя из параметров надежности коммуникационного оборудования, а также безошибочности и быстродействия используемой линии связи.

При проектировании СПД ответственного целевого назначения, предназначенных для оперативной коммуникации, часто используют обобщенное (расширенное) определение $P(t_0)$, которое учитывает специфические требования к качеству информационных передач. При таком подходе вероятность правильной передачи информации определится как вероятность ее передачи за заданное время t_0 и с заданным уровнем качества Q_0. Заданный уровень качества передачи может определяться как предельное число (частота) битовых или байтовых ошибок в передаваемой информации, предельный уровень шума, предельный уровень искажений при приеме видео или голосовой информации и т.д. Математически это можно записать как:

$$P_{(t_0, Q_0)} = P\{t \le t_0, Q \le Q_0\} \qquad . \qquad (2.4)$$

Для иллюстрации предлагаемого подхода рассмотрим пример расчета.

Пример. Пусть μ - интенсивность битовых ошибок, тогда можно показать, что вероятность безошибочной передачи n-битового сообщения определится по формуле:

$$P = e^{-n\mu} \quad . \qquad (2.5)$$

Заметим, что выражение (2.5) справедливо лишь при предположении о том, что битовые ошибки взаимонезависимы, что на практике не всегда справедливо. Но при таком предположении, интенсивность битовых ошибок можно трактовать как их частоту.

При $n>>1$ вероятность ошибки в информационной передаче становится существенной. На практике длинное сообщение разбивается на m блоков, длина каждого блока равна n/m. Тогда вероятность передачи n-битового сообщения из m блоков можно записать как

$$P = \prod_{i=1}^{m} e^{n\mu/m} \quad .$$

Кроме того, вводится режим контроля передачи информационного сообщения. Обычно контроль реализуется посредством суммирования разрядов сообщения по четности или по нечетности. Для повышения полноты контроля, как правило, используется метод продольно-поперечного суммирования по четности. В этом случае передаваемый информационный блок разбивается на k полос. Для каждой из полос производится контрольное суммирование по четности. Кроме того, производится суммирование одноименных разрядов всех полос, входящих в блок. Полнота контроля при таком продольно-поперечном суммировании, практически, близка к 100%.

Если система контроля при передаче блока обнаруживает ошибку, то происходит запрос на его повторную передачу.

Для простой математической иллюстрации рассмотренного подхода возьмем значения $k=1$ и $m=1$. Таким образом, информация передается в виде одного сообщения, которое контролируется на основе продольно-поперечного суммирования. Пусть $t_к$ – время передачи n-битового сообщения, тогда $c=[t_0/t_к]$ – допустимое число повторений передачи, означает целую часть выражения.

Вероятность передачи сообщения за один раз определится по формуле (2.5). Тогда, предположив, что c – кратность резервирования информационной передачи и воспользовавшись методикой расчета для резервного оборудования [3], можно определить вероятность правильной передачи с c-кратным повторением:

$$P = 1 - (1 - e^{-n\mu})^c \quad . \qquad (2.6)$$

Формула (2.6) означает, что информационная передача считается правильной, если хотя бы одно из ее c-кратных повторений закончилось успешно.

При $m \neq 1$ можно воспользоваться методом оценки “по наихудшему случаю”, распределив запас времени на передачу информации t_0 равномерно по всем m блокам сообщения. Тогда:

$$c = \left[\frac{t_0}{t_к m} \right] \qquad P = 1 - (1 - e^{-cn\mu/m})^m.$$

Заметим далее, что на практике часто встречается случай, когда информационная передача считается успешной (достоверной) не только тогда, когда все биты переданы безошибочно, но и в том случае, когда число искаженных битов не превышает некоторого допустимого порога.

2.3. Разработка логико-аналитической модели надежности системы передачи данных

Для построения логико-аналитической модели надежности любой достаточно сложной и многокомпонентной системы необходимо выполнить следующие этапы:

1. Построение приближенной логической схемы надежности системы.
2. Выбор критерия (или критериев) надежности системы.
3. Построение аналитической модели надежности в терминах математических выражений выбранных критериев.

Построенная таким образом логико-аналитическая (или просто аналитическая) модель надежности используется в дальнейшем для расчета основных показателей надежности при наличии достоверных исходных данных. Достоверные исходные данные для расчета надежности системы могут быть получены, главным образом, из двух источников: из технической документации на базовые аппаратные системные компоненты и на основе обработки статистики по отказам и сбоям в системе (если система уже начала эксплуатироваться или проходит рабочие испытания).

Первый этап рассмотренной методики обычно реализуется посредством анализа логики функционирования системы. На основе данного анализа составляется логическая конфигурация, состоящая из укрупненных системных блоков. В каждый такой укрупненный системный блок объединяются реальные системные компоненты, которые проявляют общий поведенческий характер с точки зрения:

1) последствий отказа,
2) стратегии восстановления,
3) стратегии реконфигурации (переключения отказавшего оборудования и подключения резервного оборудования).

На рис. 2.6 приведена такая логическая конфигурация исследуемой СПД, показанная для одного информационного тракта “Центр – Атомная станция”. В представленную приближенную логическую схему вошли два неизбыточных (нерезервированных) компонента, названных серверами коммуникации. Такой сервер коммуникации может включать в себя другие неизбыточные ком-

поненты системы, например, сервер качества, сервер локальной БД и т.д., если их надежность оказывает существенное влияние на надежность всего информационного тракта. Пунктирные линии на рис. 2.6 обозначают возможность раздельного резервирования локальных маршрутизаторов и каналов связи. На аппаратном уровне анализа системы это означает возможность использования резервного маршрутизатора при отказе основного блока с целью маршрутизации передаваемого сообщения по любому из каналов связи.

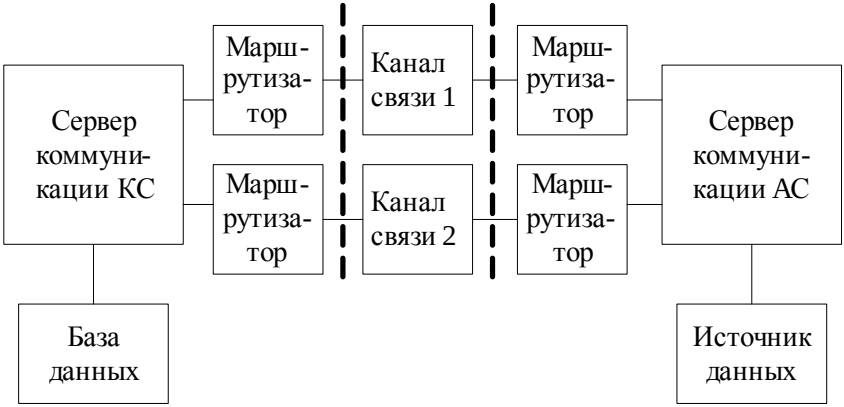


Рис.2.6. Принципиальная схема одного звена связи между КЦ и АС

В конечном счете, как видно из рис. 2.6, анализ информационного тракта СПД сводится к построению двух категорий моделей надежности: модели надежности нерезервированного системного блока с восстановлением и модели надежности оперативно дублированного системного блока. При анализе надежности каждого блока системы необходимо учитывать два типа восстановления: быстрого локального восстановления (на основе локального ЗИП силами местного обслуживающего персонала) и задержанного (на основе дистанционной доставки необходимого оборудования или вызова специалистов-экспертов по возникшей проблеме). Для простоты назовем эти два режима восстановления как режим локального восстановления и режим дистанционного восстановления.

В качестве основного критерия надежности выберем стационарный коэффициент готовности, который наиболее целесообразен для оценки надежности сложных и ответственных систем непрерывного действия [3].

Тогда для анализа готовности системных блоков первой категории (неизбыточных компонентов с двумя типами восстановления) можно использовать две типа системных графов, приведенные на рис. 2.7.

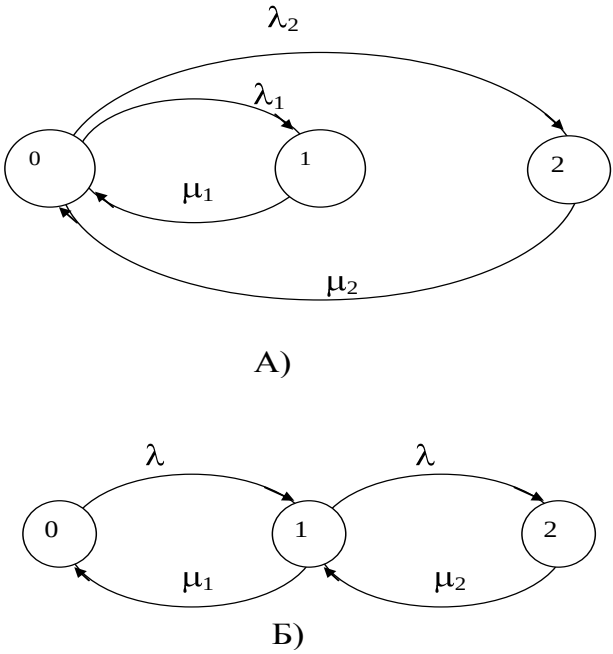


Рис.2.7. Марковская цепь неизбыточного системного компонента с двумя типами восстановления и одной ремонтной бригадой:

а) с немедленным дистанционным восстановлением,

б) с отложенным дистанционным восстановлением

На рис. 2.7, а приведен граф системы, построенный при предположении о том, что процесс и локального, и дистанционного восста-

новления начинается сразу после обнаружения отказа в системе и отнесения его к категории сложных (требующих дистанционного вызова оборудования или специалистов). Граф на рис. 2.7, б построен при предположении о том, что процесс дистанционного восстановления начинается только при отказе компонента запасного комплекта оборудования из ЗИП, установленного вместо ранее отказавшего основного оборудования (т.н. режим отложенного дистанционного восстановления).

На рис. 2.7 приняты следующие обозначения:

λ – общая интенсивность отказов неизбыточного компонента системы.

λ_1 – интенсивность отказов неизбыточного компонента системы, соответствующих состоянию “несложного” отказа.

λ_2 – интенсивность отказов неизбыточного компонента системы, соответствующих состоянию “сложного” отказа.

μ_1 – интенсивность локального восстановления системного компонента.

μ_2 – интенсивность дистанционного восстановления системного компонента. Очевидно, что $\lambda=\lambda_1+\lambda_2$, а $\mu_1 >\mu_2$.

Для дублированного системного компонента с различными типами восстановления можно предложить упрощенный граф состояний, приведенный на рис. 2.8.

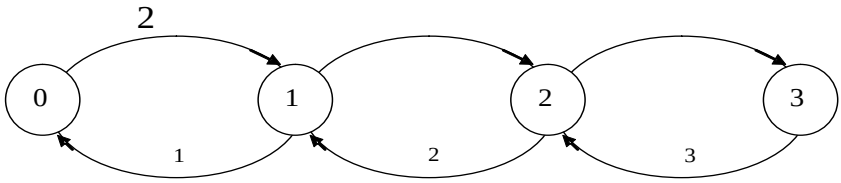


Рис.2.8. Граф состояний дублированного системного компонента с тремя типами восстановления

Обозначения, принятые на рис. 2.8, аналогичны обозначениям на рис. 2.7. Однако здесь рассмотрены не два, а три типа (режима) восстановления системного компонента. В режиме 1 (режим оперативного восстановления) резервный блок, находящийся в горячем резерве, просто подключается вместо отказавшего рабочего блока с последующей его инициализацией. Режим 2 соответствует уже рассмотренному режиму локального восстановления, в котором, фактически, реализуется принцип холодного резервирования. Наконец, режим 3 соответствует уже рассмотренному режиму отложенного или дистанционного восстановления. Поэтому на данном рисунке обозначены:

μ_1 – интенсивность восстановления из горячего резерва;

μ_2 – интенсивность восстановления из холодного резерва;

μ_3 – интенсивность дистанционного восстановления.

Граф на рис. 2.8 построен при упрощающем предположении о том, что дистанционное восстановление системного компонента (с интенсивностью μ_3) начинается только после того, как откажут оба рабочих блока и один резервный (находившийся в ЗИП и введенный в рабочую конфигурацию после отказа двух рабочих). Кроме того, предполагается, что восстановление из холодного резерва (замена оборудования из ЗИП) начинается только после отказа второго из рабочих блоков.

Теперь составим уравнения Колмогорова, соответствующие графам состояний на рис. 2.7 и 2.8. При их решении получим следующие результаты:

1. для графа на рис. 2.7, а

$$K_Г = \frac{1}{1 + \frac{\lambda_1}{\mu_1} + \frac{\lambda_2}{\mu_2}};$$

2. для графа на рис. 2.7, б

$$K_Г = \frac{1}{1 + \frac{\lambda}{\mu_1} + \frac{\lambda^2}{\mu_1 \mu_2}};$$

3. для графа на рис. 2.8

$$K_Г = 1 - \frac{2\lambda^3}{\mu_1\mu_2\mu_3 + 2\lambda\mu_2\mu_3 + 2\lambda^2\mu_3} \approx 1 - \frac{2\lambda^3}{\mu_1\mu_2\mu_3}.$$

Приближенное значение для последней формулы соответствует соотношениям

$\lambda \ll \mu_1$, $\lambda \ll \mu_2$, $\lambda \ll \mu_3$, $\mu_2 > \mu_1$, $\mu_3 \gg \mu_1$.

Дальнейшие исследования в рамках данного направления, то есть в целях создания адекватной модели надежности СПД, которая была бы удобна и практична для ее проектирования, можно сформулировать следующим образом:

1. Необходимо построить граф переходов состояний для дублированной подсистемы СПД при условии, что состав и параметры оборудования в отдельных каналах системы различны. На практике это означает учет различия параметров надежности и восстановления канала наземной связи и канала высокочастотной (спутниковой) связи.

2. Необходимо построить модель надежности СПД (базовых компонентов СПД) в терминах вероятности безотказной работы за заданное время *t*.

3. Необходимо построить модель надежности СПД (базовых компонентов СПД) в терминах среднего времени восстановления. Данная задача может быть решена на основе известной методики, которая включает в себя:

а) расчет стационарного коэффициента готовности *Кг* на основе уже полученных или новых соотношений;

б) составление уравнения на основе общепринятого определения коэффициента готовности как отношения средней наработки на отказ к сумме средней наработки и среднего времени восстановления: $Кг = \frac{T_0}{T_0 + T_B}$. Это уравнение содержит два неизвестных: T_0 и T_B;

в) определение значения T_0 на основе анализа свойств соответствующего графа и его узлов;

г) решение составленного уравнения относительно значения T_B.

Список литературы к главе 2

1. Кучукян А.Т., Мелик-Парсаданян В.Х. Некоторые функциональные особенности вычислительного комплекса “Маршрут-1” // Вопросы радиоэлектроники. ЭВТ, 1974, № 9. С. 72-77.
2. Хьюз Дж. Повышение надежности путем функционального резервирования // Электроника, 1973, № 9. С. 72-77.
3. Козлов Б.А., Ушаков И.А. Справочник по расчету надежности аппаратуры радиоэлектроники и автоматики. М.: Сов. радио, 1975.
4. Baque I. Reliability Aspects of The ILLIAC-IV // Computer. Proc. 1976 Int. Conf. on Parallel Processing, 1976, P. 123-131.
5. Stiffer J.J. The SERF Fault-Tolerant Computer // Intern. Symp. on Fault-Tolerant Computing, 1973, Palo Alto, Calif., P. 23-26.
6. Maison F.P. The MECRA: A Self-Reconfigurable Computer for Highly Reliable Process // IEEE Trans. Comput., 1971. 20. № 11. P. 1382-1388.
7. Kutch R.E. Computer Redundancy: Design, Performance and Future // IEEE Trans. Reliab., 1969. 18. № 1, P.3-11.
8. Happer R.E., Alger L.S. A Design Approach for Ultrareliable Real-Time System // Computer, 1991. № 7. P.28-30.
9. Чернышев Ю.А., Чуканов В.О. Надежностное проектирование устройств СЦВМ с комбинированным резервированием // Надежность сложных технических систем. Материалы семинара МДНТП. М.: МДНТП, 1979. С. 35-40.
10. Надежность автоматизированных систем управления / Под ред. Я.А. Хетагурова. М.: Высшая школа, 1979.
11. Справочник по цифровой вычислительной технике (электронные вычислительные машины и системы) /Б.Н. Малиновский, В.Я. Александров, В.П. Боюн и др.; Под ред. Б.Н. Малиновского. – К.: Техника, 1980.
12. Школин В.П., Фогилев А.Н. Методы построения космических БЦВМ. // Зарубежная радиоэлектроника, 1978, № 3. С. 26-37.
13. Snuth T.B. A Damage-and-Fault-Tolerant Input, Output Network // IEEE Trans. Comput., 1975. 24. № 5. P. 505-512.

14. Lipovski G.J. On A Varistructured Array of Microprocessors // IEEE Trans. Comput., 1977. № 2. P. 125-138.

15. Baer J.-L. Multiprocessing Systems // IEEE Trans. Comp., 1976, 25. № 12. P. 1271-1277.

16. Пирс У. Построение надежных вычислительных машин. М.: Мир, 1968.

17. Нейман Дж. Вероятностная логика и синтез надежных организмов из ненадежных компонентов. М.: Иностранная литература, 1956. С. 114-134.

18. Уэйкерли Дж. Повышение надежности микроЭВМ путем тройного резервирования модулей // ТИИЭР, 1976, 64, № 6. С. 68-76.

19. IV Международный симпозиум по толерантности вычислительных систем и процессов // Зарубежная радиоэлектроника, 1978, №6. С. 114-134.

20. Freeman H.A., Metre G. Fault-Tolerant Computers Using “Dotted-Logic” Redundancy Techniques // IEEE Trans. Comput., 1972, 21, №8. P. 867-871.

21. Трайон Дж. Учетверенная логика // Методы введения избыточности для вычислительных систем. М.: Советское радио, 1966.

22. Avizienis A. Fault-Tolerant Systems // IEEE Trans. Comp., 1976, 25, №12. P. 1304-1312.

23. Вайрадян А.В., Петухов М.Н., Чеканов А.А. Адаптивный восстанавливающий орган на микропроцессоре для многоканальных вычислительных систем // Тезисы докладов Всесоюзного научно-технического совещания “Проблемы создания и использования высокопроизводительных машин”. Кишинев, 1979. С. 13-14.

24. Ingle A.D., Siewiorek D.P. A Reliability Model For Various Switch Designs In Hybrid Redundancy // IEEE Trans. Comput., 1976, 25, №2. P. 115-133.

25. Avizienis A., Gillay G., Marthur F.P. и др. The STAR (Self-Testing And Repairing) Computer: An Investigation Of The Theory And Practice Of Fault-Tolerant Computer Design // IEEE Trans. Comput., 1971, 20, №11. P. 1312-1321.

26. Hopkins A.L., Smith T.B. The Architectural Elements Of A Symmetric Fault-Tolerant Multiprocessor // IEEE Trans. Comput., 1975, 24, №5. P. 498-505.

27. Koszeva L.J. A Three-Failure-Tolerant Computer System // IEEE Trans. Comput., 1971, 20, №11. P. 1389-1393.

28. Хопкинс А.Л., Смит Т.Б., Лала Дж.Х. FTMP – высоконадежный отказоустойчивый мультипроцессор для управления самолетом // ТИИЭР, 1976, 66, №10. С. 142-165.

29. Ведешенков В.А., Волков Т.Б. Об одном способе оценки избыточности вычислительных систем с саморемонтом // Автоматика и телемеханика, 1977, №6. С. 158-166.

30. Каструччио О.П., Ирби С. Цифровая система стабилизации орбитальной космической станции “Скайлаб”// Вопросы ракетной техники, 1973, №11. С. 77-85.

31. Червоный А.А., Лукьященко В.И., Котин Л.В. Надежность сложных систем. М.: Машиностроение, 1975.

32. Сандлер Дж. Техника надежности систем. М.: Наука, 1978.

33. Ушакова Г.Н. Аппаратный контроль и надежность специализированных ЭВМ. М.: Советское радио, 1969.

34. A. Ranganathan, S. Upadhya. Performance Evaluation Of Roll-Back-Recovery Techniques In Computer Programs // IEEE Trans. On Reliab., 1993, 42, № 2.

35. Барлоу Р., Прошан Ф. Математическая теория надежности. М.: Советское радио, 1969.

36. R. Rohde, J. Haskett. Disaster Recovery Planning // Commun. ACM, 1990, 33, № 6. P. 652-657.

37. Chung W.H. Reliability Analysis Of A Series System With Repair // Microelectronics Reliability, 1991, 31, № 2/3. P. 363-365.

38. Александрович А.Е., Бородакий Ю.В., Чуканов В.О. Проектирование высоконадежных информационно-вычислительных систем. М.: Радио и связь, 2004.

39. Школин В.П., Фогилев А.Н. Методы построения космических БЦВМ Зарубежная радиоэлектроника, 1978, № 3. С. 26-37.

40. Мур В.Ф., Шеннон К.Э. Надежные реле из ненадежных реле // Кибернетический сборник. Вып.1. М: Советское радио, 1966.

41. Гнеденко Б.В., Беляев Ю.К., Соловьев А.Д. Математические методы в теории надежности. М.: Наука, 1965.

42. F. Gatterson, S. Jankley. Built-In Self-Test (BIST) Structure For Analog Circuit Fault Diagnosis // IEEE Trans. On Instrumentation And Measurement, 1990.39 № 3. P. 517-521.

43. S. Byun, S. Moon. Resilient Data Management For Replicated Mobile Database Systems // Data & Knowledge Engineering, 1999. № 29. P.43-55.

44. Староселец В.Г. Оценка эффективности автоматического поиска отказов // Техническая диагностика. Труды I Всесоюзного совещания по технической диагностике / под ред. П.П.Пархоменко.

45. Курочкин Ю.А., Смирнов А.С., Степанов В.А. Информационная и функциональная надежность дискретных устройств // Автоматика и телемеханика, 1983. № 6. С. 163-168.

46. Хетагуров Я.А., Малишевский В.В., Потураев О.С. Основы инженерного проектирования управляющих ЦВМ. М.: Советское радио, 1972.

47. Кузнецов В.В. Эффективный алгоритм выделения внутренних переменных ВП // АВТ, 1979, № 1. С. 54-60.

48. Морозов Е.В. Установка контрольных точек в системах реального времени // Вопросы радиоэлектроники. (Серия ЭВТ), 1982. № 10. С. 54-57.

49. Пашковский В.П. Оптимальный выбор контролируемых параметров для систем встроенного контроля // Известия АН СССР. Техническая кибернетика, 1980. № 1. С. 81-84.

50. Половко А.М. Основы теории надежности. М: Наука, 1964.

51. Гилевич А.И. Оптимизация частоты переключения резерва // АВТ, 1978. № 1. С. 38-40.

ГЛАВА 3. МОДЕЛИ НАДЕЖНОСТИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ТИПОВОГО НАПРАВЛЕНИЯ СИСТЕМ ПЕРЕДАЧИ ДАННЫХ АЭС

3.1. Нормативные определения надежности программного обеспечения и процессов оценивания

Надежность является одной из основных характеристик качества программных средств. ГОСТ 28806-90 "Качество программных средств. Термины и определения" [1] определяет надежность программного средства (reliability) как совокупность свойств, характеризующих способность программного средства сохранять заданный уровень пригодности в заданных условиях в течение заданного интервала времени.

К подхарактеристикам надежности программного средства этот ГОСТ относит

– *завершенность* программного средства: совокупность свойств программного средства, характеризующая частоту отказов, обусловленных дефектами программного средства;

– *отказоустойчивость* программного средства: совокупность свойств программного средства, характеризующая его способность поддерживать необходимый уровень пригодности при проявлении дефектов программного средства или нарушении установленных интерфейсов. Необходимый уровень пригодности включает в себя способность к безопасному функционированию при отказах, к минимизации возможных потерь данных и исключению опасных действий при внезапном нарушении условий функционирования;

– *восстанавливаемость* программного средства: совокупность свойств программного средства, характеризующая возможность осуществления, трудоемкость и продолжительность действий по восстановлению им своего уровня пригодности, а также непосредственно подвергшихся воздействию данных, в случае отказа.

Согласно ГОСТ 28195-89 "Оценка качества программных средств. Общие положения" [2] надежность программного средства (ПС) характеризуется показателями надежности ПС, под которыми понимается способность ПС в конкретных областях применения выполнять заданные функции в соответствии с программными документами в условиях возникновения отклонений в среде функционирования, вызванных сбоями технических средств, ошибками во входных данных, ошибками обслуживания и другими дестабилизирующими воздействиями. При этом к комплексным показателям надежности относятся:

– *устойчивость функционирования* – способность обеспечивать продолжение работы программы после возникновения отклонений, вызванных сбоями технических средств, ошибками во входных данных и ошибками 'обслуживания;

– *работоспособность* – способность программы функционировать в заданных режимах и объемах обрабатываемой информации в соответствии с программными документами при отсутствии сбоев технических средств.

Согласно ГОСТ Р ИСО/МЭК 9126-93 "Оценка программной продукции. Характеристики качества и руководства по их применению" [3] надежность определяется как набор атрибутов, относящихся к способности программного обеспечения сохранять свой уровень качества функционирования при установленных условиях за установленный период времени. По этому ГОСТу подхарактеристики надежности включают:

– *стабильность* – атрибуты программного обеспечения, относящиеся к частоте отказов при ошибках в программном обеспечении;

– *устойчивость к ошибке* – атрибуты программного обеспечения, относящиеся к его способности поддерживать определенный уровень качества функционирования в случаях программных ошибок или нарушения определенного интерфейса;

– *восстанавливаемость* – атрибуты программного обеспечения, относящиеся к его возможности восстанавливать уровень качества функционирования и восстанавливать данные, непосредственно поврежденные в случае отказа, а также ко времени и усилиям, необходимым для этого.

Таким образом, среди параметров, характеризующих надежность программного средства, явным образом выделяются показатели, характеризующие частоту отказов, обусловленных дефектами программного средства и ошибками его эксплуатации, а также атрибуты, связанные с возможностью восстановления программного средства и данных, непосредственно поврежденных в случае отказа.

Стандарт [3] применяется для установления требований к качеству программного обеспечения и оценивания (измерения, ранжирования и оценки) программных продуктов. При этом в нем описывается некоторая модель оценивания качества ПО, но в то же время, отмечается, что в целях описания и оценивания уровни ранжирования и критерии необходимо установить конкретно для данной организации или для данного применения, или для того и другого. Важность каждой характеристики качества также меняется в зависимости от принятых точек зрения (*представление пользователя, руководителя).*

Процесс оценивания качества программного обеспечения состоит из трех стадий: установление (определение) требований к качеству, подготовка к оцениванию и процедура оценивания. Данный процесс может применяться в любой подходящей фазе жизненного цикла для каждого компонента программной продукции.

Целью начальной стадии процесса оценивания является установление требований в терминах характеристик качества и возможных комплексных показателей (подхарактеристик). Требования выражают потребности внешнего окружения для рассматриваемой программной продукции. Требования для продукции в целом могут отличаться от требований для отдельных компонентов.

Вторая стадия включает в себя подготовку к оцениванию (выбор метрик (показателей) качества, определение уровней ранжирования, определение критерия оценки). Ее основная задача – подготовить материал для проведения третьей стадии – непосредственного оценивания. Применение критериев оценки качества программного средства призвано обеспечить получение на основе оценок отдельных характеристик качества программного средства обоснованного заключения о его качестве в целом.

В ходе третьей стадии проводится измерение выбранных метрик, их ранжирование, а также собственно оценка программного обеспечения, при которой обобщается множество установленных уровней. Результатом является заключение о качестве программной продукции.

Для обеспечения возможности получения интегральной оценки по группам показателей качества используют факторы качества (1-й уровень): надежность ПС, сопровождаемость, удобство применения, эффективность, универсальность (гибкость) и корректность.

Каждому фактору качества соответствует определенный набор критериев качества (комплексные показатели — 2-й уровень): устойчивость функционирования, работоспособность, структурность, простота конструкции, наглядность, повторяемость, легкость освоения, доступность эксплуатационных программных документов, удобство эксплуатации и обслуживания, уровень автоматизации, временная эффективность, ресурсоемкость, гибкость, мобильность, модифицируемость, полнота реализации, согласованность, логическая корректность, проверенность.

Критерии качества определяют одной или несколькими метриками (3-й уровень).

Метрики составляются из оценочных элементов (единичных показателей — 4-й уровень), определяющих заданное в метрике свойство. Число оценочных элементов, входящих в метрику не ограничено.

Выбор оценочных элементов в метрике зависит от функционального назначения оценочного элемента и определяется с учетом данных, полученных при проведении испытаний различных видов, а также по результатам эксплуатации ПС.

Оценка качества ПС проводится в определенной последовательности:

– на фазе анализа проводится выбор показателей и их базовых значений;
– для показателей качества на всех уровнях (факторы, критерии, метрики, оценочные элементы) принимается единая шкала оценки от 0 до 1;

– показатели качества на каждом вышестоящем уровне (кроме уровня оценочных элементов) определяются показателями качества нижестоящего уровня, то есть:
  - результаты оценки каждого фактора определяются результатами оценки соответствующих ему критериев;
  - результаты оценки каждого критерия определяются результатами оценки соответствующих ему метрик;
  - результаты оценки каждой метрики определяются результатами оценки определяющих ее оценочных элементов.

Для надежности программного средства используется следующий набор критериев и метрик на различных фазах жизненного цикла (табл. 3.1).

Оценочные элементы фактора "Надежность" представлены в табл. 3.2.

Как видно из этой таблицы, на нижнем уровне оценки качества надежность программного средства характеризуется 23-я оценочными элементами, из которых лишь 4 носят расчетный характер. Рассматривается прямая задача исследования надежности программного обеспечения СПД. Поэтому для анализа должны быть выбраны метрики, относящиеся именно к фазе обслуживания (сопровождения) жизненного цикла программного средства.

Среди этих элементов особое значение имеют элементы, связанные с вероятностью безотказной работы и средним временем восстановления.

Таблица 3.1

Фаза жизненного цикла	**Критерий**	**Метрика**	
		Номер	**Название**
Фаза анализа	Устойчивость функционирования	1	Средства восстановления при ошибках на входе
		2	Средства восстановления при сбоях оборудования
Фаза проектирования	Устойчивость функционирования	1	Средства восстановления при ошибках на входе
		2	Средства восстановления при сбоях оборудования
		3	Реализация управления средствами восстановления
Фазы реализации, тестирования, изготовления, обслуживания (сопровождения)	Устойчивость функционирования	1	Средства восстановления при ошибках на входе
		2	Средства восстановления при сбоях оборудования
		3	Реализация управления средствами восстановления
	Работоспособность	4	Функционирование в заданных режимах
		5	Обеспечение обработки заданного объема информации

Таблица 3.2

Код оценочного элемента	Наименование	Метод оценки	Оценка
Н0101	Наличие требований к программе по устойчивости функционирования при наличии ошибок во входных данных	Экспертный	0-1
Н0102	Возможность обработки ошибочных ситуаций	– " –	0-1
Н0103	Полнота обработки ошибочных ситуаций	– " –	0-1
Н0104	Наличие тестов для проверки допустимых значений входных данных	– " –	0-1
Н0105	Наличие системы контроля полноты входных данных	– " –	0-1
Н0106	Наличие средств контроля корректности входных данных	– " –	0-1
Н0107	Наличие средств контроля непротиворечивости входных данных	– " –	0-1
Н0108	Наличие проверки параметров и адресов по диапазону их значений	– " –	0-1
Н0109	Наличие обработки граничных результатов	– " –	0-1
Н0110	Наличие обработки неопределенностей (деление на 0, квадратный корень из отрицательного числа и т. д.)	– " –	0-1
Н02О1	Наличие требований к программе по восстановлению процесса выполнения в случае сбоя операционной системы, процессора, внешних устройств	– " –	0-1
Н0202	Наличие требований к программе по восстановлению результатов при отказах процессора, ОС	– " –	0-1
Н02О3	Наличие средств восстановления процесса в случае сбоев оборудования	– " –	0-1
Н0204	Наличие возможности разделения по времени выполнения отдельных функций программ	– " –	0-1

Код оценочного элемента	Наименование	Метод оценки	Оценка
Н0205	Наличие возможности повторного старта с точки останова	– " –	0-1
Н0301	Наличие централизованного управления процессами, конкурирующими из-за ресурсов	– " –	0-1
Н0302	Наличие возможности автоматически обходить ошибочные ситуации в процессе вычисления	– " –	0-1
Н0303	Наличие средств, обеспечивающих завершение процесса решения в случае помех	– " –	0-1
Н0304	Наличие средств, обеспечивающих выполнение программы в сокращенном объеме в случае ошибок или помех	– " –	0-1
Н0305	Показатель устойчивости к искажающим воздействиям	расчетный	P(Y)=1-D/K, где D–число экспериментов, в которых искажающие воздействия приводили к отказу, K–число экспериментов, в которых имитировались искажающие воздействия
Н0401	Вероятность безотказной работы	– " –	P=1-Q/N, где Q–число зарегистрированных отказов, N–число экспериментов

Код оценочного элемента	Наименование	Метод оценки	Оценка
Н0501	Оценка по среднему времени восстановления	– " –	$Q_{в} = \begin{cases} 1, \text{если } Т_{в} \le T_{в}^{доп} \\ \dfrac{T_{в}^{доп}}{Т_{в}}, если\, Т_{в} > T_{в}^{доп} \end{cases}$, где $T_{в}^{доп}$ – допустимое время восстановления; $Т_{в}$ – среднее время восстановления, которое определяется по формуле: $Т_{в} = \dfrac{1}{N}\sum_{i}^{N} T_{Bi}$, где $Т_{вi}$ – время восстановления после i-го отказа

Код оценочного элемента	Наименование	Метод оценки	Оценка
Н0502	Оценка по продолжительности преобразования входного набора данных в выходной	– " –	$Q_{в} = \begin{cases} 1, \text{если } T_{пi} \le T_{ni}^{доп} \\ \dfrac{T_{пi}^{доп}}{T_{пi}}, если T_{пi} > T \end{cases}$ где $T_{ni}^{доп}$ – допустимое время преобразования i-го входного набора данных, T_{ni} – фактическая продолжительность преобразования i-го входного набора данных

Примечание. Коды оценочных элементов составлены из 5 символов следующим образом:

1-й символ – буква русского алфавита – указывает на принадлежность элемента тому или иному фактору.(«Н» – надежности, «С» – сопровождаемости, «У» – удобству применения, «Э» – эффективности, «Г» – универсальности, «К» – корректности);

2-й и 3-й символы – номер метрики, которой принадлежит оценочный элемент;

4-й и 5-й символы–порядковый номер данного оценочного элемента в метрике.

Например, «Н 0501» означает, что это 1-й оценочный элемент из 5-й метрики фактора «надежность».

3.2. Особенности программного обеспечения как восстанавливаемой системы

Основные характеристики восстанавливаемых систем

Используя термин “надежность ПО” аналогично термину “надежность аппаратуры”, необходимо помнить, что “отказы ПО” как результат проявления ошибок имеют совершенно другую физическую природу, чем отказы техники. Однако это не является причиной невозможности использования некоторых терминов и показателей надежности техники при исследовании качества ПО. В частности, это оправдывается и необходимостью решения задачи распределения ресурсов или затрат между самими вычислительными системами и их программным обеспечением при достижении заданного значения показателя надежности.

При исследовании надежности функционирования программного комплекса (ПК) обычно необходимо решать две задачи. Первая – по заданной структуре ПК, состоящего из некоторой совокупности программных средств, имеющих определенные показатели надежности, найти показатель надежности ПК. Эту задачу традиционно называют *прямой.*

Наряду с первой может решаться вторая задача – достижение максимального (минимального) или заданного значения показателя надежности при ограничениях на ресурсы, в качестве которых выступают время, стоимость и др. Или может решаться задача минимизации величины ограничения при достижении требуемого (заданного) значения показателя надежности. Любую из этих задач принято называть *обратной*.

Система программного обеспечения типового направления системы передачи данных Кризисного центра обладает рядом особенностей, которые делают невозможным непосредственное применение моделей надежности ПО, описанной в литературе. К этим особенностям следует отнести:

– логически и физически распределенный характер средств, на которых установлено данное программное обеспечение;

– отсутствие показателей, учитывающих взаимосвязь отдельных модулей;

– недостаточное количество информации по характеристикам надежности отдельных программных модулей;

– в связи с тем, что ПО функционирует на ответственных объектах и, вследствие этого, его отказы чрезвычайно редки, обработка имеющейся статистики с приемлемой точностью представляет собой отдельную непростую задачу;

– большинство используемых моделей надежности программного обеспечения позволяют получить характеристики систем без учета времени восстановления после отказа. В рассматриваемом случае такой подход принципиально невозможен. После проявления программное ошибки программное обеспечение должно быть восстановлено и время этого восстановления, иногда достаточно существенное, должно быть учтено при определении показателей надежности.

В связи с этим рассмотрим вначале общие показатели надежности восстанавливаемых систем и методы их расчета.

Общие показатели надежности восстанавливаемых систем и методы их расчета

При анализе работоспособности системы требуется оценка надёжностных характеристик для следующих случаев:

1) невосстанавливаемый элемент;
2) восстанавливаемый элемент;
3) дублированная восстанавливаемая система;
4) последовательное включение элементов для случаев:
   а) при отказе любого элемента система выключается;
   б) исправные элементы системы продолжают работать при восстановлении любых отказавших элементов, причем восстановление может быть как неограниченным, так и ограниченным.

Показатели надежности невосстанавливаемого элемента общеизвестны, и в данном случае, рассматриваться не будут.

Рассмотрим <u>*основные показатели надежности восстанавливаемых систем*</u>.

Все состояния системы S можно разделить на подмножества:

$S_K \subset S$ – подмножество состояний $j = 1...K$, в которых система работоспособна;

$S_M \subset S$ – подмножество состояний $i = 1...M$, в которых система неработоспособна.

$$S = S_K \, U \, S_M ,$$
$$S_K \cap S_M = 0.$$

1. *Функция готовности G(t) системы* определяет вероятность нахождения системы в работоспособном состоянии в момент t:

$$G(t) = \sum_{j=1}^{K} P_j(t) = 1 - \sum_{z=1}^{M} P_z(t),$$

где $P_j(t)$ – вероятность нахождения системы в работоспособном j-м состоянии;

$P_z(t)$ – вероятность нахождения системы в неработоспособном z-м состоянии.

2. *Функция простоя П(t) системы:*

$$\Pi(t) = 1 - G(t) = \sum_{z=1}^{M} P_z(t).$$

3. *Коэффициент готовности $K_г$ системы* определяется при установившемся режиме эксплуатации (при $t \to \infty$). При $t \to \infty$ устанавливается *стационарный (предельный) режим*, в ходе которого система переходит из состояния в состояние, но вероятности нахождения системы в том или ином состоянии уже не меняются:

$$P_i = \lim_{t \to \infty} P_i(t), \quad i=1...n .$$

Коэффициент готовности есть предельное значение функции готовности при установившемся режиме ($t \to \infty$):

$$k_г = \lim_{t \to \infty} G(t).$$

4. *Параметр потока отказов* системы:

$$\omega(t) = \sum_{j=1}^{K} \sum_{z=1}^{M} \lambda_{jz} P_j(t),$$

где λ_{jz} – интенсивности переходов из работоспособного состояния в неработоспособное.

Рассмотрим показатели надежности для различных видов восстанавливаемых систем. Описание такой системы наиболее удобно проводить с привлечением аппарата теории массового обслуживания [4, 5].

Граф переходов между состояниями восстанавливаемой системы, состоящей из одного элемента, имеет следующий вид (рис. 3.1).

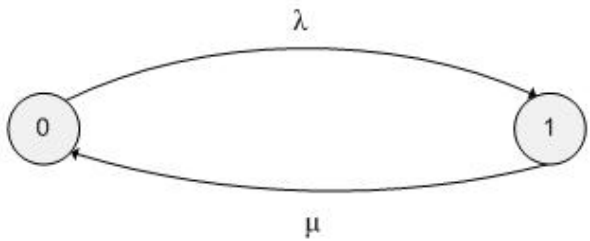


Рис 3.1. Граф переходов системы с восстановлением
S0 – работоспособное состояние
S1 – неработоспособное состояние

По графу состояний составляется система обыкновенных дифференциальных уравнений первого порядка (уравнений Колмогорова-Чепмена), имеющих вид:

$$\frac{dP_i(t)}{dt} = \sum_{l=1}^{\Theta} \lambda_{il} P_l(t) - P_i(t) \sum_{j=1}^{q} \lambda_{ij}, \; j = 1 \ldots q, \; l = 1 \ldots \Theta,$$

где λ_{ij} – интенсивность потока, переводящего систему из состояния j в состояние i,

q – количество состояний, из которых можно перейти в состояние i за один шаг,

Θ – количество состояний, в которые можно перейти из состояния i за один шаг.

В общем случае, интенсивности потоков λ_{ij} могут зависеть от времени *t*.

Чтобы решить систему дифференциальных уравнений для вероятностей состояний $P_1(t), ..., P_i(t), ... , Pn(t)$, необходимо задать начальное значение вероятностей

$$P_1(0) ,... , P_i(0), ... , Pn(0).$$

При этом должно быть обеспечено соблюдения условия:

$$\sum_{i=1}^{n} P_l(t) = 1 \ .$$

Для системы, представленной на рис. 3.1, система дифференциальных уравнений имеет вид:

$$dP_0(t)/dt = -\lambda P_0(t)+\mu P_1(t), \qquad (3.1)$$
$$dP_1(t)/dt = \lambda P_0(t)-\mu P_1(t) \ .$$

Считаем, что в начальный момент времени система работоспособна, т.е.

$$P_0(0) = 1 \text{ и } P_1(0) = 0.$$

Поскольку состояния S_0 и S_1 представляют полную группу событий, то

$$P_0(t) + P_1(t) = 1 \ .$$

Выражая $P_0(t) = 1 - P_1(t)$, и подставляя в (3.1), получаем одно дифференциальное уравнение относительно $P_1(t)$:

$$dP_1(t)/dt = \lambda (1 - P_1(t)) - \mu P_1(t).$$

Решение данного уравнения производится с использованием преобразования Лапласа.

Преобразование Лапласа для вероятностей состояния $P_i(t)$:

$$P_i(S) = \int_0^{\infty} P_i(t) \cdot e^{-St} dt \,,$$

где $P_i(S) = L\{P_i(t)\}$ – изображение вероятности $P_i(t)$.

Преобразование Лапласа для производной $dP_i(t)/dt$:

$$L\{dP_i(t)/dt\} = \int_0^{\infty} (dP_i(t)/dt) \cdot e^{-St} dt = -P_i(0) + SP_i(S) \ ,$$

для константы:

$$L\{\lambda\} = \lambda\, L\{1\} = \lambda/S\ .$$

После применения преобразования Лапласа к левой и правой частям уравнения, получим (при $P_1(0) = 0$):

$$SP_1(S) + P_1(S)(\lambda + \mu) = \lambda / S\ ;$$
$$P_1(S)(S + \lambda + \mu) = \lambda/S.$$

Откуда изображение вероятности нахождения объекта в неработоспособном состоянии:

$$P_1(S) = \frac{\lambda}{S(S + \lambda + \mu)}\ .$$

Разложение дроби на элементарные составляющие приводит к:

$$\frac{\lambda}{S(S + \lambda + \mu)} = \frac{a - \mu}{a}\left[\frac{1}{S} - \frac{1}{S + a}\right],$$

где $a = \lambda + \mu$.

Применяя обратное преобразование Лапласа, с учетом:

если $L\{f(t)\} = 1/S$, то $f(t) = 1$;

если $L\{f(t)\} = 1/(S + a)$, то $f(t) = e^{-at}$,

получим *вероятность нахождения объекта в неработоспособном состоянии*:

$$P_1(t) = \frac{a - \mu}{a}\left(1 - e^{-at}\right) = \frac{\lambda}{\lambda + \mu} \cdot \left(1 - e^{-(\lambda + \mu)t}\right).$$

Тогда *вероятность нахождения в работоспособном состоянии* $P_0(t) = 1 - P_1(t)$, равна

$$P_0(t) = \frac{\mu}{\lambda + \mu} \cdot \left(1 + \frac{\lambda}{\mu} e^{-(\lambda + \mu)t}\right).$$

С помощью полученных выражений можно рассчитать вероятность работоспособного состояния и отказа восстанавливаемого объекта в любой момент t и остальные показатели надёжности:

- *функция готовности G(t),*

$$G(t) = P_0\,(t);$$

- *коэффициент готовности системы $k_г$.* определяется при установившемся режиме $t \to \infty$:

$$k_г = \lim_{t \to \infty} P_0(t) = \frac{\mu}{\lambda + \mu}, \qquad (3.2)$$

- *функция простоя П(t)*

$$\Pi(t) = 1 - G(t) = P_1(t).$$

- *параметр потока отказов ω(t)*

$$\omega(t) = \lambda G(t) = \lambda P_0(t)$$

при $t \to \infty$ (стационарный установившийся режим восстановления):

$$\omega(t) = \omega(\infty) = \omega = \lambda P_0 = \lambda k_г,$$

- *средняя наработка между отказами* ($t \to \infty$):

$$t_0 = k_г / \omega = k_г / (\lambda k_г) = 1/\lambda.$$

Анализ изменения $P_0(t)$ позволяет сделать выводы:

1. При мгновенном (автоматическом) восстановлении работоспособности ($\mu = \infty$)

$$\lambda/\mu = 0 \text{ и } P_0(t) = 1.$$

2. При отсутствии восстановления ($\mu = 0$)

$$\lambda/\mu = \infty \text{ и } P_0(t) = e^{-\lambda t}$$

и вероятность работоспособного состояния объекта равна вероятности безотказной работы невосстанавливаемого элемента.

Для дублированной восстанавливаемой системы граф переходов между состояниями имеет вид, приведенный на рис. 3.2.

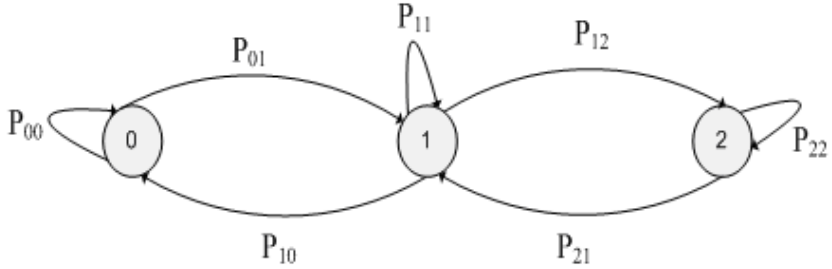


Рис. 3.2. Граф переходов дублированной восстанавливаемой системы

Вначале рассчитаем показатели надежности такой системы в предположении, что для обоих элементах параметры потоков отказов и потоков восстановлений одинаковы, т.е.

$$\lambda_1 = \lambda_2 = \lambda; \quad \mu_1 = \mu_2 = \mu.$$

Такая дублированная система может находиться в трех состояниях:

"0" – оба элемента работоспособны;

"1" – один элемент восстанавливается, другой работоспособен;

"2" – оба элемента восстанавливаются. С точки зрения выполнения функциональных задач, возложенных на систему, состояние "2" соответствует отказу.

Расчет такой системы проведем, исходя только из определений показателей надежности с использованием матрицы переходных вероятностей.

У этой системы возможны семь видов перехода из состояния в момент времени t в состояние в момент времени $t + \Delta t$:

00 – обе цепи сохраняют работоспособность;

01 – отказал один из двух элементов;

10 – один из элементов восстановлен (оба элемента стали работоспособными);

11 – восстановление одного из элементов продолжается;

12 – отказал второй элемент, а первый не восстановлен (отказ системы);

21 – один из двух отказавших элементов восстановлен;

22 – оба элемента продолжают восстанавливаться (пребывание системы в состоянии отказа).

Графу переходов соответствует матрица переходных вероятностей 3x3. Крайние элементы побочной диагонали матрицы имеют порядок $0(\Delta t)$, так как по исходному предположению поток отказов в системе простейший, и время восстановления распределено по экспоненциальному закону. Согласно простейшему потоку в первой строке матрицы исключается ситуация, когда за время Δt система может перейти из состояния "0" в состояние "2": $P_{02}(\Delta t) = 0$. Рассуждая аналогично, по третьей строке матрицы запишем $P_{20}(\Delta t) = 0$. При простейшем потоке система за время Δt может из состояния "0" с вероятностью $P_{01}(\Delta t)$ перейти в состояние "1" или с вероятностью $P_{00}(\Delta t)$ остаться в состоянии "0". Точно такая же картина соответствует состоянию "2". С вероятностью $P_{21}(\Delta t)$ система может перейти в состояние "1" (один элемент восстановится) или с вероятностью $P_{22}(\Delta t)$ останется пребывать в состоянии "2" (оба элемента неработоспособны – состояние отказа).

Элементы первой строки матрицы переходных вероятностей зависят от режима использования резервной цепи. Так при нагруженном резерве, работающих обеих цепях, интенсивность потока отказов равна 2λ , а при ненагруженном – λ. Поэтому

$$P_{00}(\Delta t) = e^{-(y+1)\cdot\lambda\cdot\Delta t},$$

где y - коэффициент, учитывающий состояние резерва (y = 0 при ненагруженном режиме и y=1 при нагруженном).

С учетом того, что для первой строки матрицы

$P_{00}(\Delta t) + P_{01}(\Delta t) = 1,$

получим

$P_{01}(\Delta t) = 1 - P_{00}(\Delta t) = 1 - e^{-(y+1)\lambda\Delta t}$.

Элементы второй строки матрицы переходных вероятностей соответственно запишутся так:

$P_{10}(\Delta t) + P_{11}(\Delta t) + P_{12}(\Delta t) = 1$

$P_{10}(\Delta t) = 1 - e^{-\mu\Delta t}$

$P_{12}(\Delta t) = 1 - e^{-\lambda\Delta t}$

$P_{11}(\Delta t) = 1 - [P_{10}(\Delta t) + P_{12}(\Delta t)] = 1 - e^{-\mu\Delta t} - e^{-\lambda\Delta t}$.

Элементы третьей строки анализируемой матрицы, с учетом количества ремонтных бригад и многократного восстановления отказавших цепей, соответственно определятся так:

$P_{21}(\Delta t) + P_{22}(\Delta t) = 1$

$$P_{22}(\Delta t) = e^{-r\mu\Delta t}$$

$P_{21}(\Delta t) = 1 - P_{22}(\Delta t) = 1 - e^{-r\mu\Delta t}$,

где r - число ремонтных бригад (r = 1 или r = 2).

При дублировании с восстановлением возможны 4 варианта задач анализа надежности такой системы:

1) многократно восстанавливаемая система с нагруженным резервом и одной ремонтной бригадой (y = 1, r = 1);

2) многократно восстанавливаемая система с нагруженным резервом и двумя ремонтными бригадами (y = 1, r = 2);

3) многократно восстанавливаемая система с ненагруженным резервом и двумя ремонтными бригадами (y = 0, r = 2);

4) многократно восстанавливаемая система с ненагруженным резервом и одной ремонтной бригадой (y = 0, r = 1).

Для определения $P_0(t)$, $P_1(t)$, $P_2(t)$ необходимо составить и решить систему трех дифференциальных уравнений:

$P'_0(t) = f_0(\lambda,\mu,r,y,t)$
$P'_1(t) = f_1(\lambda,\mu,r,y,t)$
$P'_2(t) = f_2(\lambda,\mu,r,y,t)$,
где λ, μ, r, y, t – постоянные коэффициенты.

Для этого на основе свойств столбцов матрицы необходимо записать выражения формул полных вероятностей $P_0(t + \Delta t)$, $P_1(t + \Delta t)$, $P_2(t + \Delta t)$, затем записать производные для выражений вероятностей нахождения системы в состояниях "0", "1", "2" и свести их в систему уравнений:

$$P_0^{'}(t) = \lim_{\Delta t \to 0} \frac{P_0(t+\Delta t) - P_0(t)}{\Delta t} \ ;$$

$$P_1^{'}(t) = \lim_{\Delta t \to 0} \frac{P_1(t+\Delta t) - P_1(t)}{\Delta t} \ ;$$

$$P_2^{'}(t) = \lim_{\Delta t \to 0} \frac{P_2(t+\Delta t) - P_2(t)}{\Delta t} \ .$$

Формулы полных вероятностей запишутся на основе составленной матрицы соответственно по первому столбцу:
$P_0(t+\Delta t) = P_0(t) * P_{00}(\Delta t) + P_1(t) * P_{10}(\Delta t)$,
по второму столбцу:
$P_1(t+\Delta t) = P_1(t) * P_{11}(\Delta t) + P_0(t) * P_{01}(\Delta t) + P_2(t) * P_{21}(\Delta t)$,
по третьему столбцу:
$P_2(t+\Delta t) = P_2(t) * P_{22}(\Delta t) + P_1(t) * P_{12}(\Delta t)$.

Подставив в эти выражения соответствующие значения переходных вероятностей, получим систему из трех дифференциальных уравнений с четырьмя постоянными коэффициентами λ, μ, r, y.

Определение искомых вероятностей пребывания системы в состояниях "0", "1" и "2" в момент времени t производится при следующих начальных условиях: $P_0(0) = 1$; $P_1(0) = P_2(0) = 0$, т.е. система первоначально включается в работу с обеими исправными цепями.

Решение системы уравнений такого типа подробно изложено в специальной литературе, Искомое выражение функции готовности анализируемой системы при найденных значениях $P_0(t)$, $P_1(t)$, $P_2(t)$ на основе известного свойства $P_0(t) + P_1(t) + P_2(t) = 1$ удобнее записать в виде: $G(t) = 1 - P_2(t)$.

Даже в нерезервированной восстанавливаемой системе при $\frac{\lambda}{\mu} \le 0{,}01$, G(t≥0,5года) ≥0,999, и значение этой функции быстро приближается к стационарному коэффициенту готовности. В связи со сказанным, оценку надежности ответственных систем, рассчитанных на длительный срок эксплуатации, целесообразно производить с помощью коэффициента готовности.

Для дублированной системы с многократным восстановлением с одной (r = 1) и двумя (r = 2) ремонтными бригадами коэффициенты готовности имеют вид:

$$К_{Г(r=1)} = \frac{(y+1)\lambda\mu + \mu^2}{(y+1)\lambda^2 + (y+1)\lambda\mu + \mu^2}. \qquad (3.3)$$

$$К_{Г(r=2)} = \frac{2(y+1)\lambda\mu + 2\mu^2}{(y+1)\lambda^2 + 2(y+1)\lambda\mu + 2\mu^2}. \qquad (3.4)$$

Таким образом, получили показатели надежности отдельного восстанавливаемого элемента и дублированной системы, состоящей из двух восстанавливаемых элементов.

Рассмотрим теперь показатели надежности системы, состоящей из n последовательно включенных элементов, каждый из которых представляет собой отдельный восстанавливаемый элемент или дублированную систему.

Граф переходов между состояниями такой системы зависит от того, в каком режиме находятся исправные элементы системы при восстановлении отказавшего элемента.

В режиме 1 при отказе любого одного из элементов остальные элементы выключаются, то есть интенсивность отказов исправных элементов на это время считается равной нулю. Это означает, что при восстановлении одного из элементов ни один из остальных элементов системы не может отказать. В таком случае для восстановления системы достаточно одной ремонтной бригады.

В режиме 2 при отказе любого одного из элементов остальные элементы остаются в рабочем режиме, то есть показатели надежности каждого из элементов не зависят от состояния других элементов.

Граф переходов между состояниями системы, состоящей из двух элементов и работающей в режиме 1, имеет вид, приведенный на рис. 3.3.

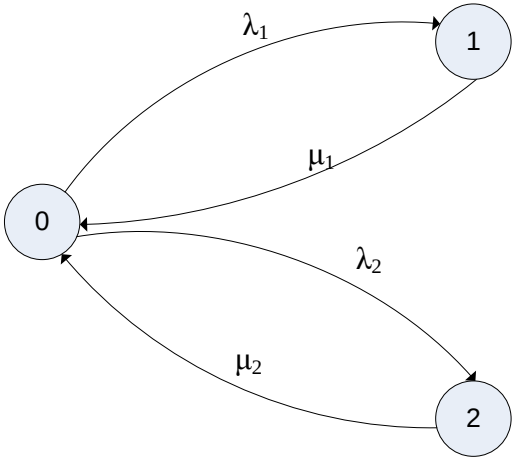


Рис.3.3. Граф переходов системы из двух элементов с восстановлением, работающей в режиме 1:
S_0 – работоспособное состояние
S_1 – отказ элемента 1 (неработоспособное состояние)
S_2 – отказ элемента 2 (неработоспособное состояние)

Расчет такой системы дает следующее значение коэффициента готовности:

$$К_Г(t) = К_Г + (1 - К_Г) \cdot e^{-t(\lambda_1 + \lambda_2 + \frac{1}{\tau})}, \qquad (3.5)$$

где $К_Г$ – стационарный коэффициент готовности, определяемый по формуле:

$$К_Г = \frac{1}{1 + \frac{\lambda_1}{\mu_1} + \frac{\lambda_2}{\mu_2}} = \frac{1}{1 + \sum_{i-1}^{2} (\frac{1}{К_{Г_i}} - 1)}, \qquad (3.6)$$

а τ определяется соотношением: $\tau = \frac{\mu_1 + \mu_2}{\lambda_1 + \lambda_2}$.

В общем случае, стационарный коэффициент готовности для системы из n элементов, включённых последовательно, в случае, когда при отказе система выключается, определяется выражением [10]:

$$К_{Г} = \frac{1}{1+\sum_{i-1}^{n}(\frac{1}{К_{Г_i}}-1)} \quad , \qquad (3.7)$$

где $К_{Г_i}$ – коэффициент готовности i-го элемента.

Граф переходов между состояниями системы, состоящей из двух элементов и работающей в режиме 2, когда при отказе система не выключается, имеет вид, приведенный на рис. 3.4.

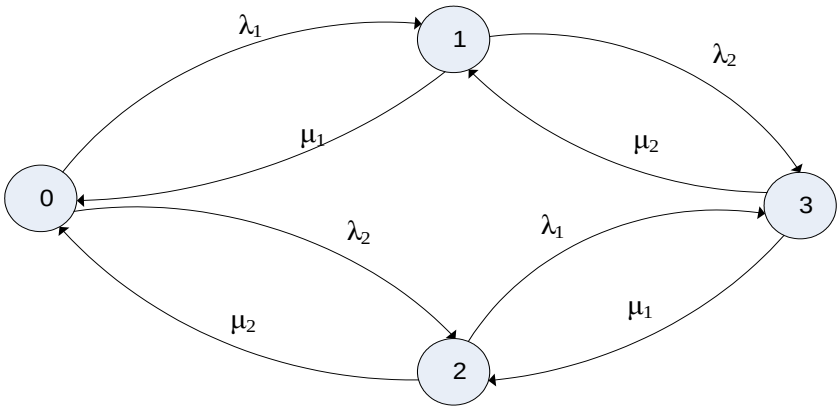


Рис.3.4. Граф переходов системы из двух элементов с восстановлением работающей в режиме 2
S_0 --- работоспособное состояние
S_1 --- отказ элемента 1 (неработоспособное состояние)
S_2 --- отказ элемента2 (неработоспособное состояние)
S_3 --- отказ двух элементов (неработоспособное состояние)

Расчет такой системы дает следующее значение коэффициента стационарной готовности:

$$К_{Г} = \frac{1}{(1+\frac{\lambda_1}{\mu_1})\cdot(1+\frac{\lambda_2}{\mu_2})} = \frac{1}{\prod_{i=1}^{2}(1+\frac{\lambda_i}{\mu_i})} \quad , \qquad (3.8)$$

В общем случае, стационарный коэффициент готовности для системы из n элементов, включённых последовательно, в случае, когда при отказе система не выключается, определяется выражением [10]:

$$К_{Г} = \frac{1}{\prod_{i=1}^{n}(1+\frac{\lambda_i}{\mu_i})} = \frac{1}{\prod_{i=1}^{n}\frac{1}{K_{Г_i}}} = \prod_{i=1}^{n} K_{Г_i} \quad , \qquad (3.9)$$

где $К_{Г_i}$ – коэффициент готовности i-го элемента.

3.3. Модель надежности программного обеспечения типового направления системы передачи данных между Кризисным центром и атомными станциями

Разработка логической схемы модели надежности программного обеспечения типового направления системы передачи данных между Кризисным центром и атомными станциями

Для оценки показателей надежности программного комплекса необходима следующая информация:

– состав модулей, образующих комплекс,

– структура комплекса с той точностью, которая позволит построить логическую схему надежности;

– выражения, задающие интенсивности переходов между модулями программного комплекса,

– показатели надежности каждого модуля,

– показатели восстановления каждого модуля.

Построение логической схемы надежности программного обеспечения требует учета ряда факторов, которые в значительной степени отличают в этом вопросе надежность программного обеспечения от надежности аппаратного обеспечения. Отметим среди них следующие:

– не всегда удается разбить всё ПО на модули с указанием связей между ними,

– трудно определить, какие модули могут выступать в качестве резервных по отношению друг к другу,

– для определения последовательности вызовов программных модулей комплекса требуются специальные инструментальные средства, которые должны быть встроены в программный комплекс на этапе его проектирования,

– каждая программа (как ПК в целом, так и составляющие его модули) представляет собой уникальное изделие, для которого, в отличие от аппаратуры, принципиально отсутствуют какие-либо заводские показатели надежности типа наработки на отказ или интенсивности отказов,

– формулировка отказа системы представляет собой нетривиальную задачу, так как согласно [3] надежность программного средства представляет собой набор атрибутов, относящихся к способности программного обеспечения сохранять свой уровень качества функционирования при установленных условиях за установленный период времени,

– большое влияние на оценку надежности каждого модуля и, следовательно, всего комплекса оказывают условия функционирования, в число которых входят помимо характеристик технических средств также и другие программные средства, установленные на данной машине. Эти средства, не участвуя непосредственно в решении данной задачи, могут существенно изменять процесс обработки информации на компьютере. Поэтому функционирование одного и того же программного модуля в различном программно-аппаратном окружении может привести к различным характеристикам надежности, и показатели, полученные, например, на пакете программ, установленных на Курской АЭС, могут существенно отличаться от показателей, полученных на Билибинской станции,

– программные средства практически всегда следует рассматривать как систему, имеющую время восстановления, отличное от нуля, так как, в отличие от аппаратуры, здесь нельзя заменить один отказавший модуль на другой, заведомо исправный, а требуется восстановление отказавшего модуля с восстановлением также контекста задачи на момент отказа или ближайший к нему момент контрольной точки;

– программы, обеспечивающие дублирование выполняемых функций, в отличие от дублирующих аппаратных модулей, нельзя рассматривать как два полностью идентичных блока, имеющие одинаковые показатели потоков отказов и восстановлений. Каждая из них является самостоятельным изделием, с присущими только ей ошибками проектирования, которое характеризуются собственными потоками отказов. Так же время восстановления будет зависеть от многих факторов, среди которых выделим следующие:

– является ли данное программное средство собственной разработкой или изделием, поставляемым сторонней фирмой,

– каковы условия обслуживания данного продукта (собственными силами или фирмой-разработчиком);

– какова интенсивность отказов данного программного продукта (зачастую, бо́льшая интенсивность отказов может соответствовать меньшему времени восстановления, так как обслуживающий персонал получает опыт работы с этим продуктом).

Структуру программных средств типового направления системы передачи данных (ТНСПД) между Кризисным центром (КЦ) и АС можно представить следующим образом (рис. 3.5):

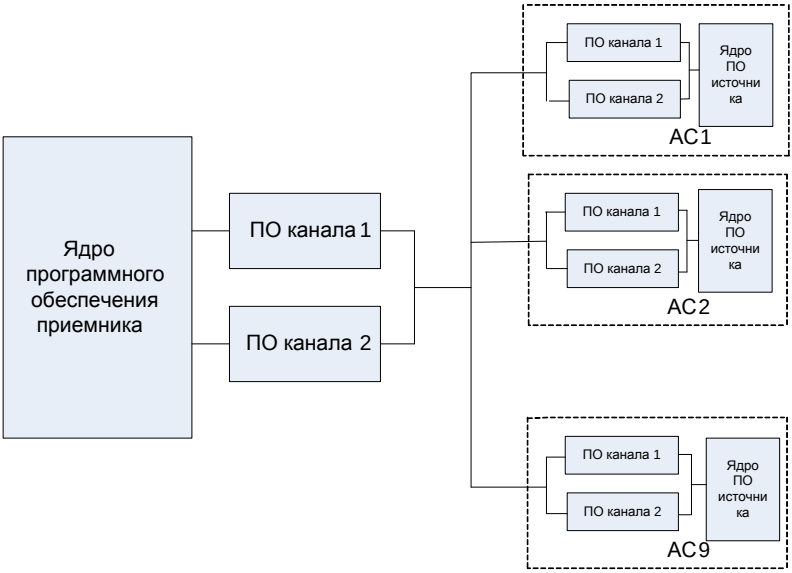


Рис.3.5. Обобщённая структура программного обеспечения

Здесь предполагается, что программный комплекс распадается на две группы программных средств: средства, расположенные собственно в КЦ, и средства, расположенные на каждой атомной станции. Часть этих средств функционирует на устройствах, дублирующих работу друг друга. К таким средствам следует отнести, например, программы, обеспечивающие передачу информации между АС и КЦ по параллельным каналам, а также автоматизированные системы контроля радиационной (АСКРО) обстановки на

АЭС, которые в настоящее время выполняются с использованием двух различных программно-аппаратных комплексов.

Такая структура может привести к различным моделям надежности в зависимости от определения условия возникновения отказа системы.

При составлении модели надежности программных средств системы передачи данных были использованы следующие положения:

– программные модули, обеспечивающие работу каждого из каналов передачи данных, как в Кризисном центре, так и на каждой станции дублируют работу друг друга;

– программные модули, обеспечивающие передачу информации по тому или иному каналу, в Кризисном центре и на каждой атомной станции работают независимо, то есть выход из строя модуля передачи информации по наземному каналу в Кризисном центре и модуля передачи информации по спутниковому каналу на станции сохраняют работоспособное состояние системы;

–программные модули автоматизированной системы контроля радиационной обстановки, принадлежащие к различным её подсистемам, также учитываются как дублирующие друг друга, хотя в этом случае точнее было бы говорить о снижении функциональных возможностей комплекса в случае выхода из строя одной из подсистем, что потребует определения критерия отказа АСКРО при определённом уровне такого снижения;

– программное обеспечение приемника и каждой из АС с точки зрения надежности представляют собой последовательное соединение, то есть потеря работоспособности ПО в центре или на любой из станций фиксируется как отказ всей системы.

Таким образом, логическая схема надежности программного обеспечения программного комплекса выглядит следующим образом (рис. 3.6).

Расчет надежности программного обеспечения типового направления системы передачи данных между Кризисным центром и атомными станциями

Согласно исходным данным, программное обеспечение Кризисного центра и каждой атомной станции имеет вид, представленный на рис. 3.7 и 3.8 соответственно.

Таким образом, каждая из таких структур представляет собой последовательное соединение программных модулей, отдельные из которых дублированы.

Система, состоящая из N последовательных восстанавливаемых элементов, отказывает, когда отказывает любой из элементов системы. Коэффициенты готовности отдельных модулей определяются выражением (3.2) для недублированного и (3.3), (3.4) – для дублированного модулей. Как показано выше, при заданных допущениях и известных значениях коэффициентов готовности каждого из последовательно включенных элементов $К_{Г_i}$, коэффициент готовности системы будет определяться режимом использования работоспособных модулей при отказе системы.

Соответственно, коэффициент готовности программного обеспечения в случае, когда при отказе система отключается, будет определяться следующим выражением:

$$К_{Г} = \frac{1}{1+\sum_{i=1}^{n}\frac{\lambda_i}{\mu_i}+\sum_{i=1}^{m}\frac{2\lambda_i^2}{2\lambda_i\mu_i+\mu_i^2}},$$

где n – количество недублированных программных модулей, m – количество модулей, имеющих дублирующие программные средства. Для дублированных систем предполагается одна ремонтная бригада и одинаковые интенсивности отказов и восстановлений для обоих модулей.

Для структуры программного обеспечения Кризисного центра, представленной на рис. 3.7, $n = 6$, $m = 1$ и, следовательно:

$$К_{Г_{КЦ}} = \frac{1}{1+\sum_{i=1}^{6}\frac{\lambda_i}{\mu_i}+\frac{2\lambda_i^2}{2\lambda_i\mu_i+\mu_i^2}},$$

где λ_i и μ_i – интенсивности потоков отказов и восстановлений.

Рис.3.6. Модель надежности программного обеспечения

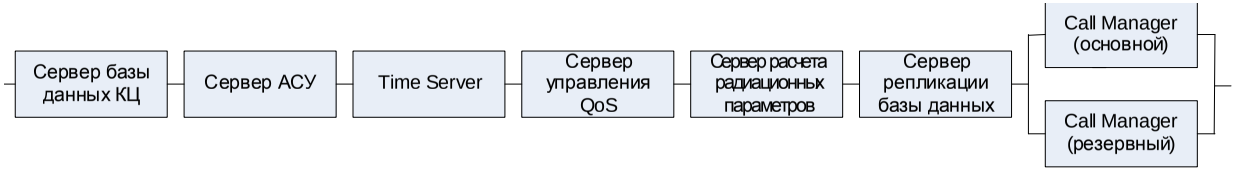


Рис.3.7 Структура программного обеспечения Кризисного центра

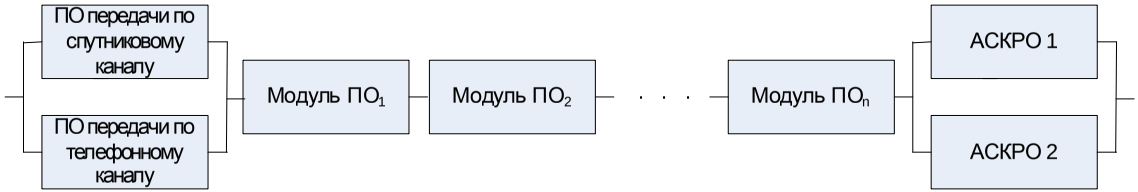


Рис.3.8. Структура программного обеспечения АС

Для структуры программного обеспечения атомной станции, представленной на рис. 3.8, m = 2 (ПО передачи по спутниковому или телефонному каналу связи и две параллельно работающие автоматизированные системы контроля радиационной обстановки), а значение n определяется количеством последовательно работающих программных модулей:

$$К_{Г_{АЭС}} = \frac{1}{1 + \sum_{i-1}^{n} \frac{\lambda_i}{\mu_i} + \sum_{i=1}^{2} \frac{2\lambda_i^2}{2\lambda_i \mu_i + \mu_i^2}} .$$

Вычислив коэффициенты готовности программного обеспечения Кризисного центра ($К_{Г_{КЦ}}$) и коэффициенты готовности ПО каждой атомной станции ($К_{Г_{АЭС}}$), можно рассчитать общий коэффициент готовности программного обеспечения.

В том случае, когда система работает в режиме, при котором исправные модули не отключаются при неработоспособности какого-либо модуля, имеем согласно (3.9):

$$К_{Г} = \prod_{i=1}^{n} K_{Г_i} ,$$

то есть коэффициенты стационарной готовности всех модулей, входящих, как в состав программного обеспечения Кризисного центра, так и в состав ПО всех атомных станций, необходимо перемножить.

Основные направления развития модели надежности

Представленный материал представляет собой общие подходы к построению модели надежности типового направления системы передачи данных между Кризисным центром и АС. Развитие представленной модели требует решения ряда вопросов.

Во-первых, дублирование программных средств в значительной степени отличается от дублирования аппаратных модулей. В случае программного обеспечения дублирующий программный модуль, в общем случае, представляет собой не второй экземпляр той же программы, а другое программное средство со своими ин-

тенсивностями потоков отказов и восстановлений. Граф переходов между состояниями такой системы представлен на рис. 3.9.

Вычисление показателей надежности такой системы представляет собой отдельную задачу, особенно с точки зрения определения параметров потоков отказов и восстановлений каждого модуля, вычленения их из общей статистики. Также требует отдельного рассмотрения вопрос взаимосвязи модулей, составляющих программный комплекс.

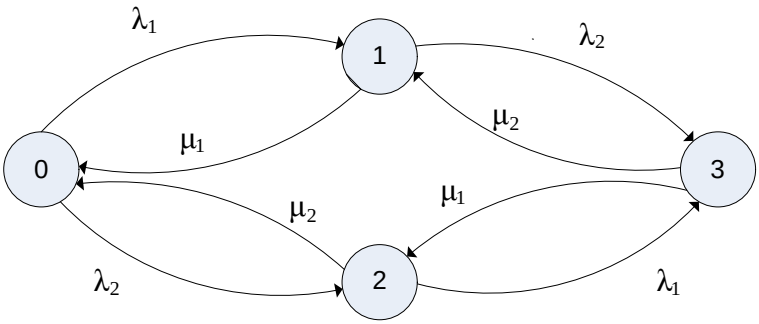


Рис.3.9. Граф переходов в случае дублированной восстанавливаемой системы с неоднородными элементами

Второй вопрос, который требует дополнительного исследования, связан с высокой надежностью применяемых средств. В случаях, когда имеется ограниченное число статистических данных о надежности используемых модулей, оказывается полезным оценить верхнюю и нижнюю границы того или иного показателя надёжности.

Малая статистика позволяет более или менее достоверно оценить лишь среднее время безотказной работы и иногда дисперсию распределения времени безотказной работы.

Для произвольных законов распределений случайных величин можно воспользоваться оценкой Чебышева:

$$p\{|\xi - M[\xi]| \geq \varepsilon\} \leq \sigma^2 / \varepsilon^2,$$

где ξ – случайная величина; $M[\xi]$ – математическое ожидание; σ^2 – дисперсия; ε – произвольное положительное число [10].

Эта оценка показывает, что при любом законе распределения вероятность отклонения случайной величины от её математического ожидания на значение, большее $k\sigma$, не превышает $1/k^2$. Однако эта оценка является достаточно грубой. Сузить границы оценки можно, если имеется дополнительная информация о законе распределения случайной величины.

В условиях малой информации существенное улучшение оценок можно получить для так называемых "стареющих" распределений. Старение означает ухудшение показателей надёжности с течением времени, например, увеличение со временем интенсивности отказов. В этом случае существуют методы, описанные, например, в [9], позволяющие получить двустороннюю оценку вероятности безотказной работы. Однако данные подходы не могут быть применены в случае оценки надежности программного обеспечения, интенсивность отказов которого со временем снижается. Поэтому необходима разработка отдельного математического аппарата для оценки надежности программного обеспечения по малой информации.

Третий вопрос, которыё требует дополнительного изучения, заключается в уточнении режимов работы каждого программного модуля и программного комплекса в целом. На этом этапе требуется специфицировать работу каждого программного модуля в случае отказов других модулей, входящих в состав типового направления системы передачи данных, а также определить уровень снижения функциональных возможностей ПО, при котором отказ не наступает.

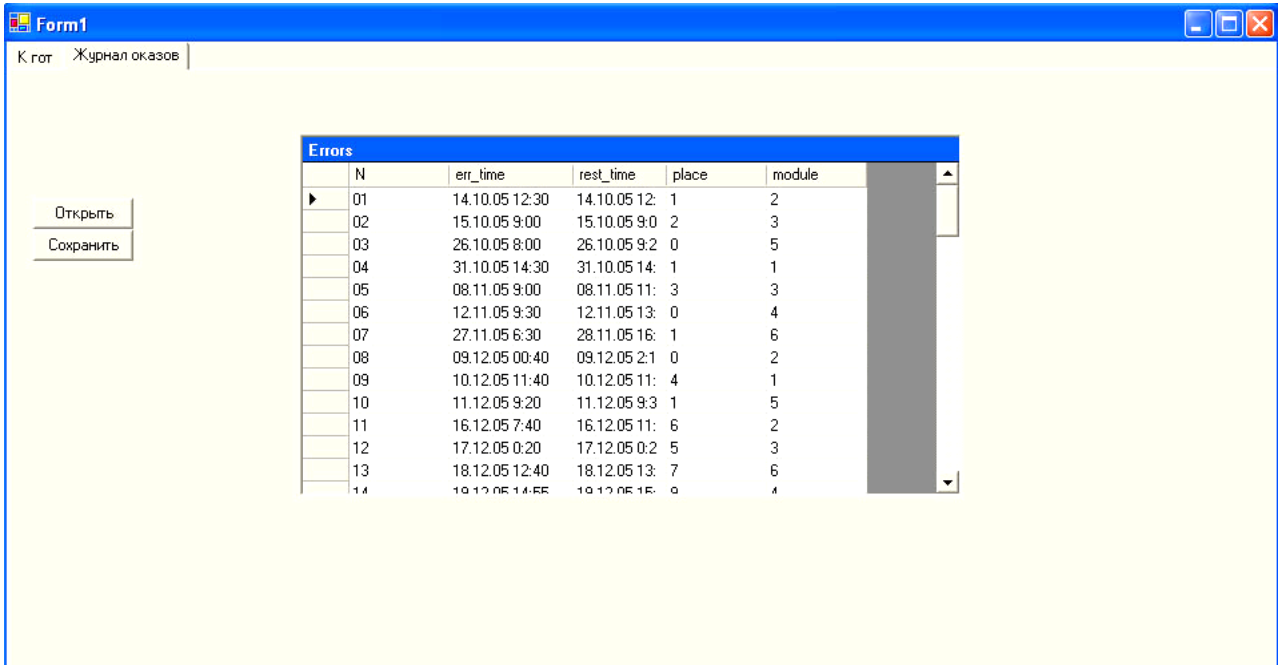
3.4. Программная реализация обобщенной модели надежности программного обеспечения типового направления системы передачи данных

Для расчета надежности программного обеспечения типового направления системы передачи данных разработана соответствующая программа. Входной информацией для неё служит журнал учета неисправностей, содержащий массив статистической информации об астрономическом времени отказа того или иного модуля и времени его включения в работу. Результатом работы являются коэффициенты готовности программного обеспечения Кризисного центра, а также каждой из атомных станций. Коэффициенты готовности АЭС вычисляются по двум различным схемам:

- по индивидуальной статистике программных модулей, относящихся к каждой станции,
- в случае, если набранный статистический материал будет признан недостаточным для получения достоверных оценок, коэффициенты готовности программных модулей АЭС рассчитываются, исходя из предположения, что программы, работающие на различных станциях, представляют собой разные копии одной и той же программы и, следовательно, обеспечивают общий статистический материал. В этом случае коэффициенты готовности ПО каждой станции будут одинаковыми.

Структура интерфейса

Интерфейс программы представляет собой оконную форму с двумя вкладками. В первой вкладке проходит обработка получаемых статистических данных. В результате выбора файла происходит его загрузка в таблицу на экране (рис. 3.10). Полученную таблицу можно исправлять или вносить в неё новые значения. Следующая вкладка "Кгот" (рис. 3.11) является основной частью программы. Вкладка предоставляет пользователю возможность просмотреть таблицу отказов, подсчитать коэффициент готовности для различного типа подключения.



N	err_time	rest_time	place	module
01	14.10.05 12:30	14.10.05 12:	1	2
02	15.10.05 9:00	15.10.05 9:0	2	3
03	26.10.05 8:00	26.10.05 9:2	0	5
04	31.10.05 14:30	31.10.05 14:	1	1
05	08.11.05 9:00	08.11.05 11:	3	3
06	12.11.05 9:30	12.11.05 13:	0	4
07	27.11.05 6:30	28.11.05 16:	1	6
08	09.12.05 00:40	09.12.05 2:1	0	2
09	10.12.05 11:40	10.12.05 11:	4	1
10	11.12.05 9:20	11.12.05 9:3	1	5
11	16.12.05 7:40	16.12.05 11:	6	2
12	17.12.05 0:20	17.12.05 0:2	5	3
13	18.12.05 12:40	18.12.05 13:	7	6
14	19.12.05 14:55	19.12.05 15:	9	4

Рис.3.10. Редактирование журнала отказов

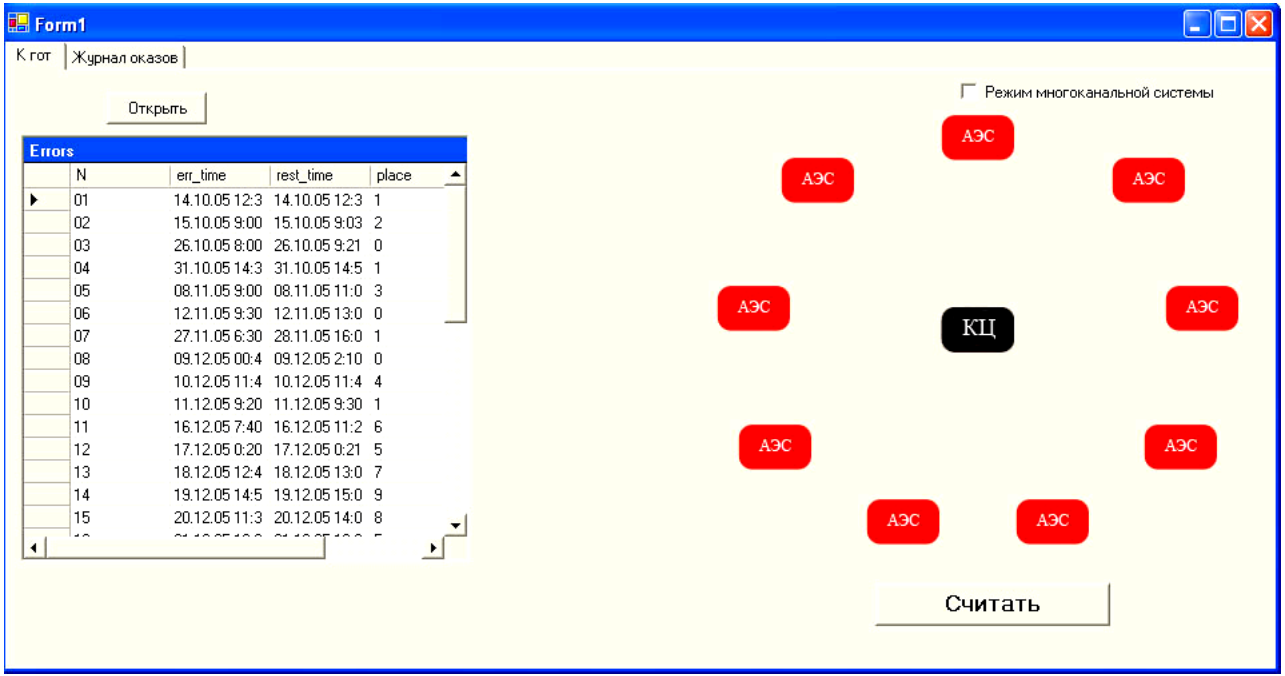


Рис. 3.11. Основной модуль программы

Определение коэффициента готовности для каждой АЭС производится нажатием на соответствующую пиктограмму. При этом появлении стрелки программа указывает на связь АЭС и Кризисного центра (рис. 3.12). Кроме стрелки возникает графическое описание структуры программного обеспечения этого объекта в виде отдельного окна (рис. 3.13).

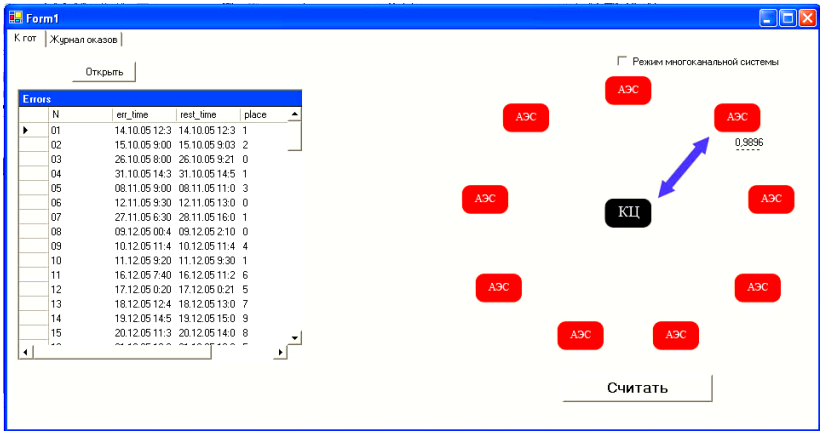


Рис. 3.12. Расчёт $K_{гот}$ для одноканальной системы

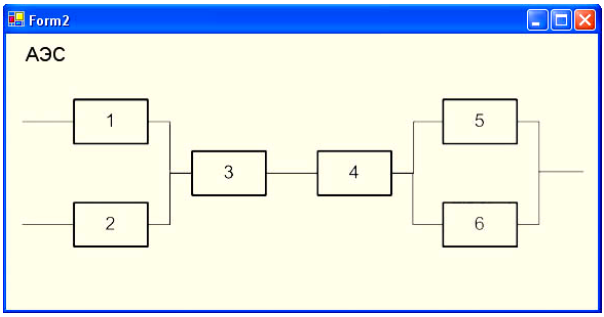


Рис. 3.13. Структура программного обеспечения АЭС

В режиме многоканальной системы производится расчет Кгот для всех АЭС с учётом последовательного подключения всех объектов. То есть, если один из объектов находится в состоянии отказа, то вся система неработоспособна.

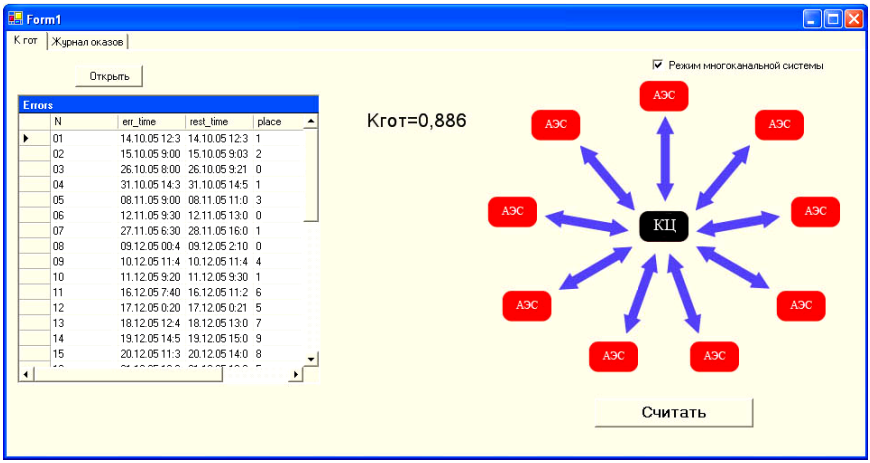


Рис. 3.14. Расчёт $К_{гот}$ для многоканальной системы

Алгоритм программы представлен на рис. 3.15.

Алгоритм работы процедуры многоканальной обработки представлен на рис. 3.16, а процедуры статистического анализа – на рис. 3.17.

Структура данных

Входные данные, используемые в программе, представляют собой файл формата XML: Errors.xml.

Файл содержит следующую информацию:

<N> – номер ошибки по порядку;

<err_time> – время отказа;

<rest_time> – время восстановления;

<place> – объект, на котором произошёл сбой;

<module> – модуль, в котором произошёл сбой.

Данная структура в максимальной степени соответствует структуре журнала неисправностей.

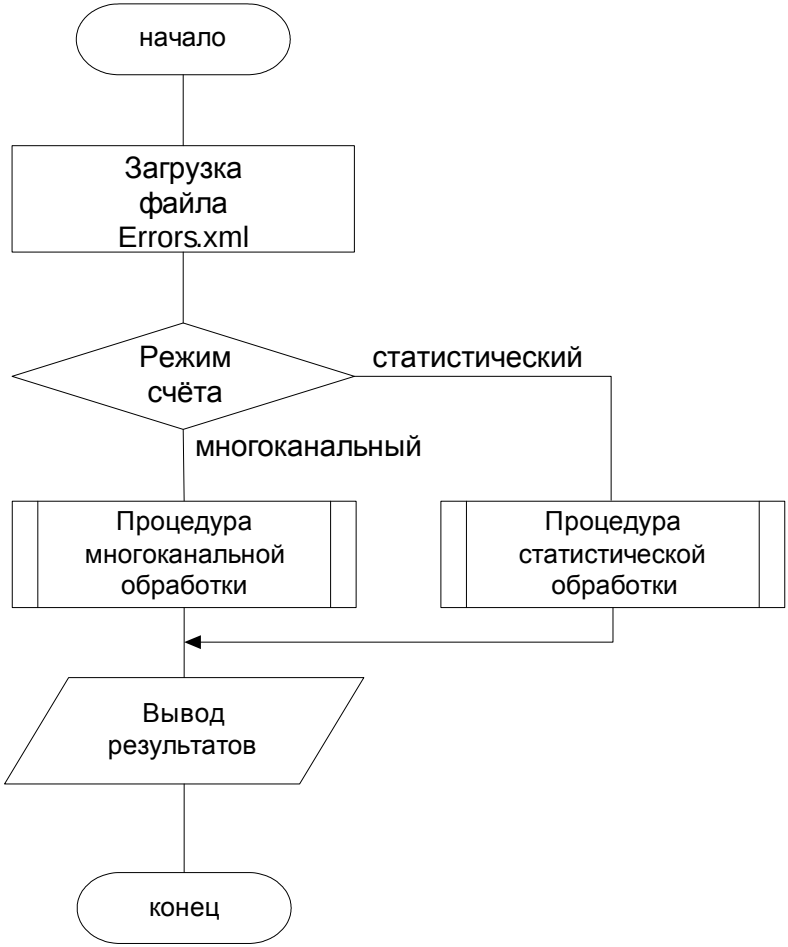


Рис. 3.15. Алгоритм программы

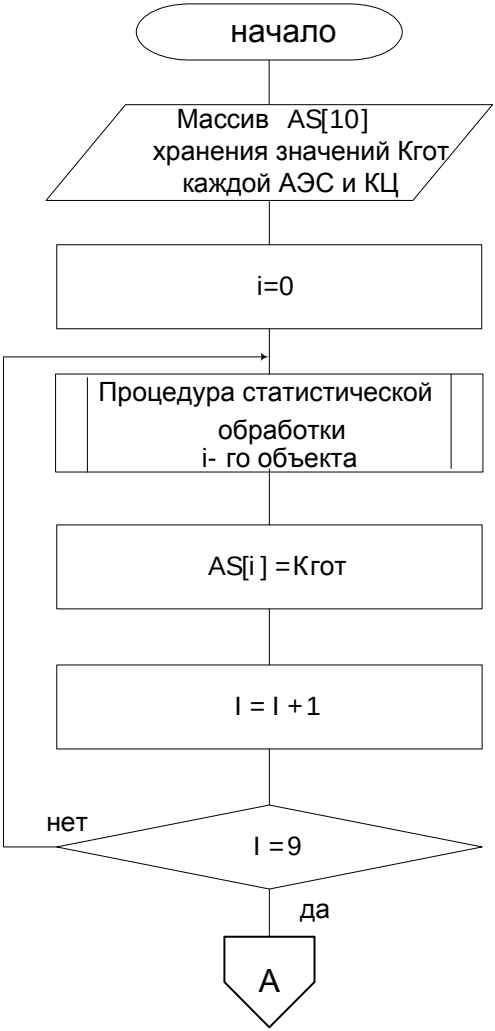


Рис. 3.16. Алгоритм процедуры многоканальной обработки

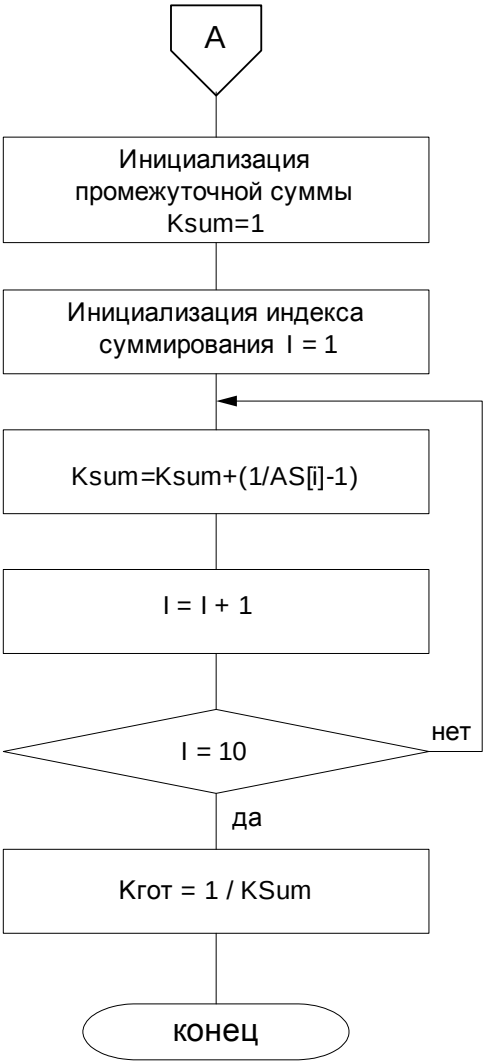


Рис. 3.16. Окончание

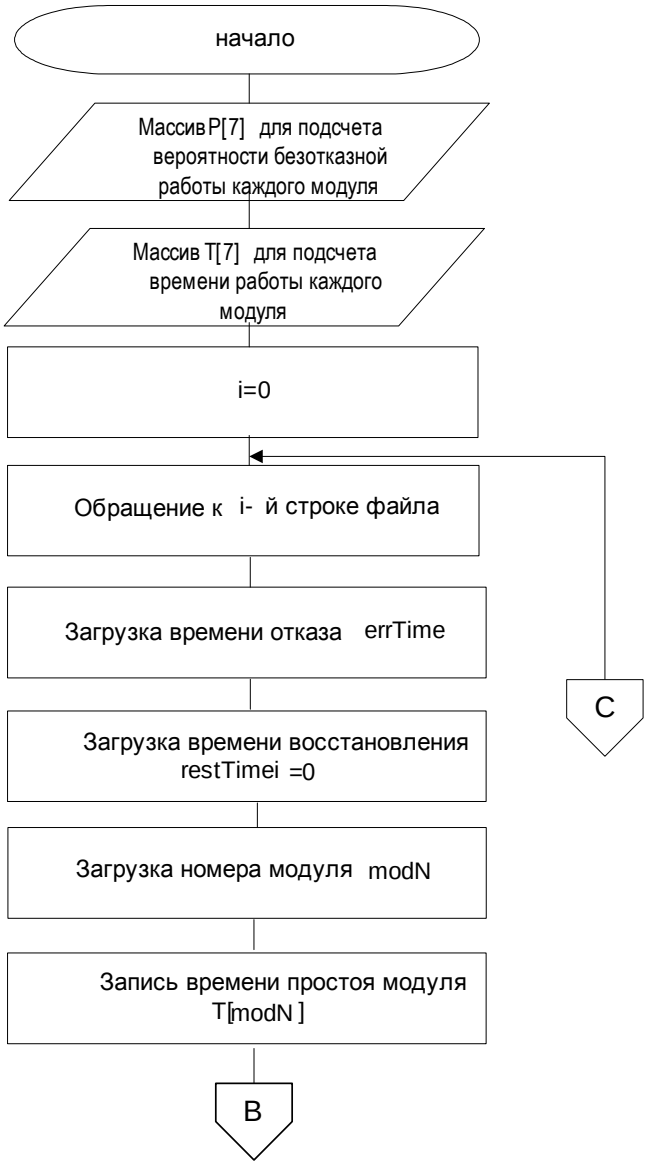


Рис. 3.17. Алгоритм процедуры статистического анализа

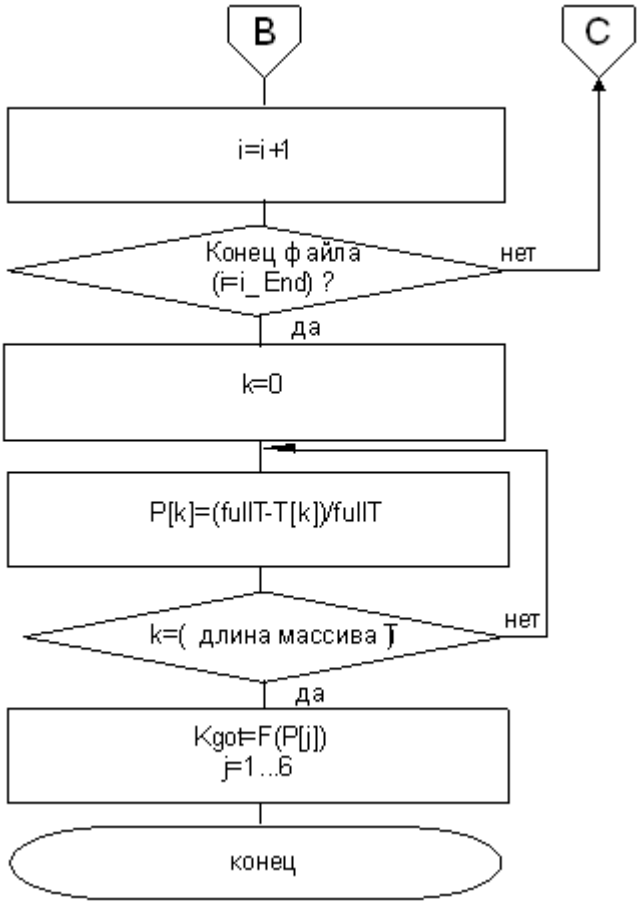


Рис. 3.17. Окончание

3.5. Разработка рекомендаций по построению комплексной модели надежности типового направления системы передачи данных с учетом параметров ее аппаратного и программного обеспечения

Задача построения комплексной модели надежности системы с учетом ее аппаратных и программных средств является весьма сложной задачей. Данная задача на практике решается, как правило, несколькими упрощенными способами.

1. Первый подход заключается в том, что для надежности аппаратной части системы и для надежности ПО строятся отдельные модели. Результаты расчета по каждой из полученных моделей надежности интерпретируются таким образом, как если бы система состояла из двух независимых комплексов: аппаратного и программного. Если результатом расчетов служит вероятностная характеристика системы (например, вероятность функционально безотказной работы или коэффициент готовности), то общая характеристика системы получается перемножением отдельных показателей для аппаратной и программной частей. Значительным упрощающим допущением в этом случае является предположение о независимости отказов программных и аппаратных компонентов системы, хотя на практике это условие далеко не всегда соблюдается.
2. Второй подход заключается в составлении общего графа переходов состояний системы с учетом как программных, так и аппаратных компонентов. Такой подход целесообразен и дает хорошие результаты, если каждому аппаратному компоненту системы соответствует свой программный компонент. В этом случае, если процессы проявления ошибок (отказов) ПО и их исправления могут быть аппроксимированы экспоненциальными законами, то для получения общих оценок надежности удобно использовать известный аппарат марковских процессов. Такой подход используется, например, в [11, 12].

3. Наиболее точный результат должен дать смешанный метод, при котором весь программно-аппаратный комплекс сначала разбивается на чисто аппаратные и программно-аппаратные модули, которые в логической модели надежности представляются последовательным соединением. Расчет показателей надежности аппаратных модулей проводится традиционными методами. Для программно-аппаратных модулей должен быть проведен более тщательный анализ, особенно с точки зрения получения необходимой статистики. Здесь требуется разграничить отказы этих модулей, вызванные аппаратурой, от отказов, обусловленных наличием ошибок в программном обеспечении. Для вычислительных систем, работающих на атомных станциях, эта работа может быть основана на том факте, что каждая вычислительная система представляет собой один и тот же образец программы, работающий на уникальных аппаратных средствах. Для вычислительных систем, работающих в Кризисном центре, условия для такого сравнительного анализа отсутствуют, что потребует более тщательного подхода к разделению причин отказа на аппаратные и программные.

Список литературы к главе 3

1. ГОСТ 28806-90. Качество программных средств. Термины и определения.

2. ГОСТ 28195-89. Оценка качества программных средств. Общие положения.

3. ГОСТ Р ИСО/МЭК 9126-93 Оценка программной продукции. Характеристики качества и руководства по их применению.

4. Клейнрок Л. Теория массового обслуживания. Пер. с англ. М.: Машиностроение, 1979. 432 с., ил.

5. Саати Т.Л. Элементы теории массового обслуживания и ее приложения. М.: Сов. радио. 1971. 520 с.

6. Лазарев Д.В. Качество и надежность программного обеспечения. "Новые информационные технологии". Тезисы докладов XIII Международной школы-семинара. М.: МГИЭМ, 2005. 361с.

7. Смагин В.А. Об одном методе исследования немарковских систем. АН СССР. Техническая кибернетика. М: 1983, № 6. С. 31–36.

8. Барлоу Р., Прошан Ф. Математическая теория надежности. М.: Сов. радио, 1969. 488 с.

9. Журавлев Ю.П., Котелюк Л.А., Циклинский Н.И. Надежность и контроль ЭВМ. М.: Сов. радио, 1978. 416с.

10. Козлов Б.А., Ушаков И.А. Справочник по расчету надежности аппаратуры радиоэлектроники и автоматики. М.: Сов. радио, 1975. 472 с.

ВВЕДЕНИЕ К ГЛАВАМ 4 И 5

В настоящее время в связи с повышением масштабов и сложности проектируемых информационно-вычислительных систем (ИВС) все более остро встает проблема обеспечения их надежности. Особенно это характерно для ИВС ответственного целевого назначения, например, для систем мониторинга критических технологических объектов (объектов повышенной степени опасности). К таким объектам, безусловно, относятся АЭС.

Расчёт показателей надёжности высоконадёжных вычислительных систем должен проводиться комплексно, с учётом всех факторов. Их можно разбить на две основные группы: факторы, связанные с надежностью установленного на ИВС программного обеспечения, и традиционные факторы, учитывающие надёжность аппаратных средств.

Учет надежности программного обеспечения весьма затруднён тем обстоятельством, что используемые в ответственных ИВС программы весьма надёжны, их отказы крайне редки, а протоколирование времени и причин таких отказов на стадии эксплуатации ПО и тем более на стадии его тестирования и отладки, как правило, недостаточны. Кроме того, здесь следует учитывать те обстоятельства, что, в отличие от аппаратных модулей, какие-либо надёжностные характеристики для используемых программных модулей при их поставке не приводятся. Также отсутствует какая-либо подробная информация по структуре программ. В связи с этим использование традиционных методов и моделей оценки надёжности программного обеспечения в данном случае представляется затруднительным.

Среди структурных методов обеспечения надежности аппаратных средств ИВС выделяются две группы методов: методы, основанные на резервировании их компонентов, и методы, предусматривающие реконфигурацию системы в случае отказовых ситуаций.

Методы резервирования систем предполагают неизменность базовой рабочей конфигурации системы в процессе ее работы. В случае отказов компонентов системы происходит простая замена отказавших блоков на исправные резервные блоки. Такая замена

может происходить либо в рамках оперативного переключения оборудования (в этом случае время простоя системы считается несущественным), либо в рамках процесса восстановления отказавших блоков (в этом случае при анализе надежности системы учитывается конечное время восстановления). Методы реконфигурации систем предполагают изменение рабочей конфигурации системы в процессе ее работы. При этом изменяются отдельные показатели ее производительности, например, объем памяти, быстродействие или параметры отказоустойчивости.

В рамках данного пособия основное внимание уделяется средствам восстановления системы и методам активного резервирования.

ГЛАВА 4. РАСЧЕТ ПОКАЗАТЕЛЕЙ НАДЁЖНОСТИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ВЫЧИСЛИТЕЛЬНЫХ СРЕДСТВ СИСТЕМ ПЕРЕДАЧИ ДАННЫХ АЭС

4.1. Особенности оценки надежности программного обеспечения высоконадёжных вычислительных систем

Разработка моделей надежности программного обеспечения вычислительных средств системы передачи данных должна учитывать тот факт, что в системе параллельно функционируют три группы программ, каждая из которых вносит свой вклад в показатели надежности: операционная система, системные программы, например СУБД, а также прикладные программы. Основными особенностями функционирования программно-аппаратных комплексов ответственного назначения являются высокие требования к их надежности и, как следствие, необходимость учёта всех факторов, определяющих её.

Учет показателей надежности системных программ весьма затруднен тем, что, в отличие от аппаратных модулей, какие-либо надёжностные характеристики для них не приводятся. Также отсутствует какая-либо подробная информация по структуре программ этого типа. Поэтому здесь приходится использовать оценочные показатели, связанные с длиной программы.

Для систем, прошедших тестирование, по различным оценкам содержится от пяти до ста ошибок на 1000 строк исполняемого кода [1, 2, 9]. По мере развития программного обеспечения в каждой его новой версии появляется все больше возможностей (и, соответственно, больший объем кода), и часто новая версия является менее надежной, чем предыдущая. В [1, 3] показано, что число ошибок на тысячу строк кода стремится к стабилизации по мере роста числа выпущенных версий, но асимптотически этот показатель отличается от нуля. Для ответственных применений, к которым можно отнести операционные системы и другое системное ПО, к моменту

поставки системы клиенту в нем может содержаться 0,04...0,15 ошибки на 1000 строк кода программы [3, 5, 6].

Однако незначительность этой величины не дает повода полностью отбросить влияние ошибок в этих программных средствах ввиду большого объема таких программ. В ядре ОС Windows XP содержится более 5 миллионов строк кода [2, 12]. Это означает, что даже при самых оптимистичных оценках (0,04 ошибки на 1000 строк программного кода, что можно считать предельным для высококачественных программных продуктов) к моменту начала поставок системы клиентам ядро Windows XP содержит более 200 ошибок.

Еще хуже то, что около 70% кода операционных систем занимает код драйверов устройств. Драйверы устройств обычно пишутся программистами, работающими на изготовителей периферийных устройств, и контроль качества их продукции обычно ниже, чем у поставщиков операционных систем. Например, в Linux частота появления ошибок в драйверах устройств от трех до семи раз выше, чем в других частях ядра. В Windows XP 85% всех аварийных отказов обуславливается наличием ошибок в коде драйверов.

Чтобы оценить вклад системных программ в общую надежность программного обеспечения, необходимо знать не только количество имеющихся в нём ошибок, но и законы распределения времени их обнаружения. Такая информация может быть получена на основе анализа протоколов тестирования программных средств, которые для данного вида программ недоступны пользователю. Оценка вида и параметров этих законов может быть проведена путём ретроспективного анализа на основе информации о времени подготовки к выходу новой версии ПО и её объёма.

Примем средний срок выпуска новой версии системного ПО в 3 года. Тогда получим, что интенсивность проявления ошибок составляет (для Windows):

– в начале этапа тестирования (5 ошибок на 1000 строк кода) число ошибок составит

$$2{,}5*10^6*2*5/10^3=25000;$$

– в конце этапа тестирования (0,04 ошибок на 1000 строк кода) число ошибок составит

$$5*10^6*0{,}04/10^3=200.$$

4.2. Структура имитационных моделей надёжности программного обеспечения

Вероятность безотказной работы комплекса при случайной длительности выполнения задачи составляет [7]:

$$P = \int_0^\infty p(t)dG(t) = \int_0^\infty p(t)g(t)dt, \qquad (4.1)$$

где $G(t)$ – вероятность того, что длительность решения задачи не превысит величины t (распределение длительности выполнения задач), $g(t)$ – плотность распределения $G(t)$, $p(t)$ – вероятность безотказной работы комплекса за время t.

Вероятность безотказной работы программного комплекса при случайной длительности выполнения задачи определяется временем работы программы при конкретных исходных данных, а также видом и параметрами функции надежности программных средств.

Для оценки показателей надежности программного комплекса необходима следующая информация:

– состав модулей, образующих комплекс,

– структура ПК с той точностью, которая позволит построить логическую схему надежности;

– выражения, задающие интенсивности переходов между модулями программного комплекса,

– показатели надежности каждого модуля.

Для оценки вероятности того, что за время решения задачи не произойдёт отказа программного обеспечения, необходимо определить вид и параметры подынтегральных функций в выражении (4.1).

Приведенная аналитическая модель оценки надежности программных средств не всегда обеспечивает получение выражения, которое позволит вычислить вероятность решения задачи при случайном времени её работы в условиях наличия ошибок в программе, не выявленных на этапах её тестирования и отладки. Это связано со следующими факторами:

– при произвольных законах распределения случайных вели-

чин, определяющих искомую вероятность, выражение (4.1) может не иметь аналитического решения;

– вычисление этого выражения затрудняется при конечных пределах интеграла, то есть в случае необходимости определить вероятность решения задачи за определённое время;

– при увеличении сложности программного комплекса существенно возрастают размерности величин, используемых для вычислений.

В силу этого оценка вероятности безотказной работы программного комплекса может быть выполнена методами и средствами имитационного моделирования на основе информации, полученной на этапе его тестирования и отладки, а также оценки возможного количества не обнаруженных на этих этапах ошибок.

При наличии информации только о количестве выявленных ошибок, времени тестирования и возможного количества необнаруженных ошибок наиболее целесообразным представляется, несмотря на ее определенные недостатки, модель Шумана роста надежности программ [7], которая позволяет по накапливаемой статистике появления программных ошибок предсказать время появления следующей ошибки.

Математические соотношения, описывающие эту модель, положены в основу разработанной имитационной модели. Она позволяет по полученным значениям интервалов времени между обнаружением на этапе тестирования программных ошибок определить время появления следующей ошибки:

$$\sum_{i=1}^{n} \frac{1}{N - i} = n/(N - \frac{1}{T} \cdot \sum_{i=1}^{n} (x[i] \cdot i)) = K \cdot T, \qquad (4.2)$$

где N – общее количество ошибок в программе до начала тестирования (в общем случае – неизвестная величина),

n – количество ошибок, обнаруженных на этапе тестирования,

T – период тестирования,

$x[i]$ – интервалы между обнаружениями ошибок,

K – некоторая неизвестная константа, на которую снижается функция риска после обнаружения очередной ошибки.

Одним из основных недостатков модели Шумана является то, что она не учитывает возможного изменения величины порога в функции риска по мере отладки программы. Для устранения этого недостатка в имитационной модели использован метод, позволяющий по первоначально накопленной статистике проявления программных ошибок на каждом очередном шаге пересчитывать все получаемые параметры, касающиеся отладки программы, и вновь проводить моделирование процесса отладки.

Чем больше накоплено первоначальной статистики (количество x[*i*]), тем точнее будут все последующие вычисления. Однако для того, чтобы в этом случае получить исходные значения K и N по имеющейся статистике, необходимо решить систему нелинейных уравнений n-го порядка. Как правило, существующие системы имитационного моделирования не располагают встроенными средствами для выполнения такой задачи и к тому же плохо согласуются с универсальными языками программирования и специализированными математическими пакетами (типа Mathcad).

Рассматриваемая модель лишена указанных недостатков.

Её структура приведена рис. 4.1.

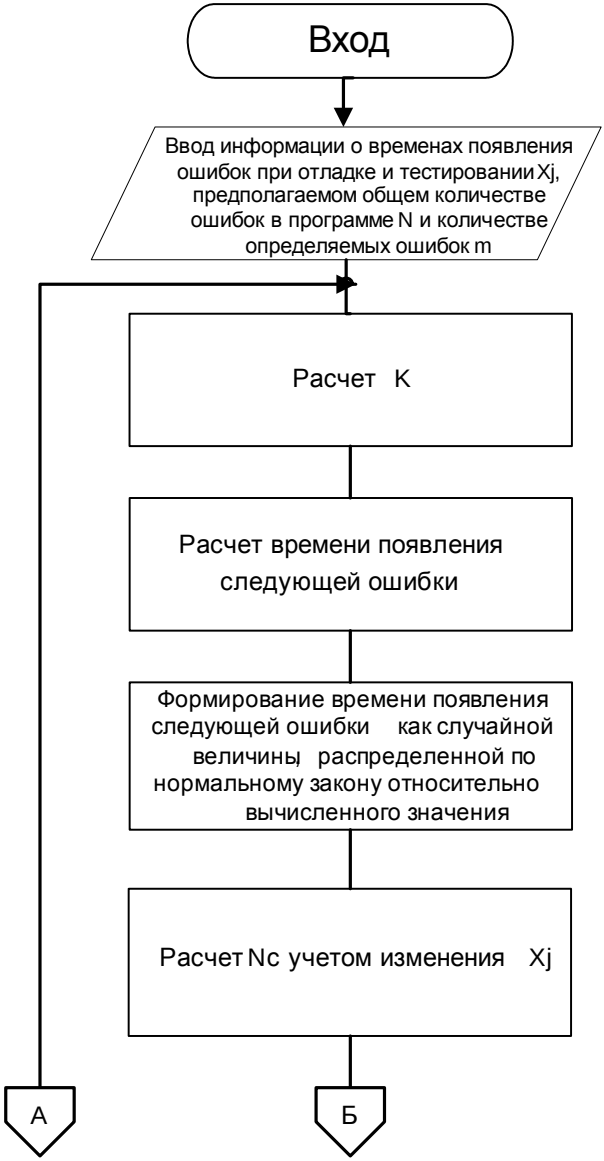


Рис. 4.1. Структура системы имитационного моделирования для анализа времени обнаружения очередной ошибки в программе

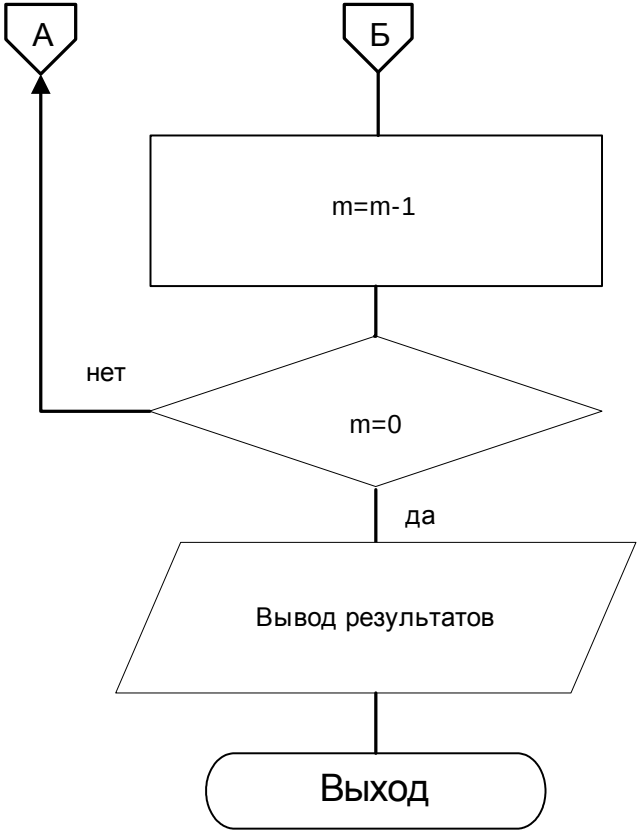


Рис. 4.1. Окончание

Основные этапы имитационного моделирования

1. По полученным экспериментально значениям $x[i]$ $(i=1...n)$ (времена между обнаружением ошибок в программе) вычисляем время выявления следующей ошибки $x[n+1]$ по соотношению, полученному из выражения (4.2):

$$x_{n+1} = \frac{1}{K} \bullet \sum_{i=1}^{n+1} \frac{1}{N-i} - \sum_{i=1}^{n} x_i \ . \qquad (4.3)$$

2. Предполагая, что время обнаружения следующей ошибки – случайная величина, распределенная по нормальному закону с математическим ожиданием $x[n+1]$, проводим розыгрыш этой случайной величины с помощью программных средств, которые, как правило, имеются в любой системе имитационного моделирования. Полученное значение $x[n+1]$ используем при вычислениях на этапах 3 и 4.

3. Уточняем значение N, используя значение $x[n+1]$, полученное на этапе 2, и K, полученное на этапе 4 предыдущего итерационного цикла:

$$N = \frac{1}{\sum_{i=1}^{n+1} x[i]} \left(\frac{n+1}{K} + \sum_{i=1}^{n+1} (x[i] \cdot i) \right). \qquad (4.4)$$

4. Уточняем значение K по вычисленному на этапе 3 значению N и полученному на этапе 2 значению $x[n+1]$:

$$K = \frac{1}{\sum_{i=1}^{n+1} x[i]} \sum_{i=1}^{n+1} \frac{1}{N-i} \ . \qquad (4.5)$$

5. Если определено время появления всех ошибок, расчет которых планировался, то завершаем моделирование, иначе переходим к шагу 1 – расчету времени обнаружения очередной ошибки.

Особенностью использования данной модели при расчете времени появления очередной ошибки для программ, протокол тестирования отсутствует (а именно это имеет место в нашем случае),

требуется на этапе инициализации модели подготовить исходные данные по временам обнаружения ошибок при тестировании. Эти исходные данные должны:

1) отражать основные тенденции, наблюдаемые при тестировании программ, а именно изменение времени обнаружения очередной ошибки с увеличением длительности тестирования,

2) соответствовать количеству обнаруженных ошибок за период тестирования.

Существуют различные предположения, связанные с вероятностью обнаружения очередной ошибки в процессе тестирования, а следовательно, и со временем обнаружения каждой следующей ошибки в процессе тестирования. Но основной является модель, в которой это время постоянно возрастает. Будем придерживаться этой точки зрения, и сделаем ещё одно допущение, касающееся закона изменения времени обнаружения следующей ошибки. Будем полагать, что это время возрастает в геометрической прогрессии с некоторым не известным наперед знаменателем p при начальном времени, равном 1.

Тогда, зная время тестирования программы T и определив на основе данных о ее длине количество ошибок n, выявленных при тестировании, можно рассчитать величину p, решив следующее уравнение:

$$T = t_1 \cdot \frac{p^n - 1}{p - 1}, \tag{4.6}$$

где t_1 – время обнаружения 1-й ошибки при тестировании.

Приняв $t_1 = 1$, получим

$$\frac{p^n - 1}{p - 1} = T \ . \tag{4.7}$$

Так как величина n достаточно большая, то для определения величины p данное уравнение в каждом конкретном случае целесообразно решать с помощью специализированных математических пакетов (например, Mathcad).

4.3. Расчет показателей надежности программного обеспечения, использующегося в составе вычислительных средств систем передачи данных

Подготовка исходных данных

Использование имитационной модели, описанной в предыдущем разделе, дает возможность оценить показатели надежности программного обеспечения вычислительных средств системы передачи данных типового направления связи Кризисного центра концерна «Росэнергоатом» и АС.

Подготовим исходные данные для такого расчета.

Будем считать, что для системных программ новая версия содержит около 30% кода нового кода по сравнению с предыдущей версией. Для прикладных программ это значение будет равным 50%. Объём прикладных программ возьмём равным 100000 строк.

Для определения величины программного кода Oracle Content Server for Windows 8 проведём следующую оценку. Согласно данным разработчика [10] объём программного кода этой системы составляет 339 105 291 bytes. Средняя длина машинной команды составляет 8 байт. Приняв, что каждая строка исходного программного кода транслируется в 10 машинных команд, получим, что исходный код составлял приблизительно $4{,}2*10^6$ строк.

Количество ошибок в тексте системных программ возьмем на уровне минимального из оценочных значений, а именно, 5 ошибок на 1000 строк исходного кода и 0,04 ошибки на 1000 строк кода после выполнения тестирования. Значения этих показателей для драйверов, входящих в состав ОС, установим в 3 раза выше, чем для кода ядра.

Срок подготовки к выходу в свет новой версии программного продукта, а следовательно, и длительность этапа тестирования T, положим равным 3 года.

Расчет знаменателя *p,* используемого для определения времён отказов на этапе тестирования, вычислим на основании выражения (4.7.) с использованием пакета Mathcad.

Подготовленные таким образом исходные данные представлены в табл. 4.1.

Таблица 4.1. Исходные данные для расчета надежности программного обеспечения

	Исходный код				**Этап тестирования**		
	Объём	**Объём нового кода**	**Количество ошибок на 1000 строк кода**	**Общее количест-во ошибок**	**Знаменатель при заполне-нии исходных данных**	**Количество ошибок на 1000 строк кода**	**Количество ошибок к окончанию тестирования**
ОС без учета вклада драйверов	$5*10^6$	$1,5*10^6$	5	7500	1,00026	0,04	60
Ядро ОС	$1,5*10^6$	$0,45*10^6$	5	2250		0,04	18
Драйверы	$3,5*10^6$	$1,05*10^6$	15	15750		0,12	126
ОС с учетом вклада драйверов	$5*10^6$	$1,5*10^6$		18000	1,00004		144
Oracle Content Server	$4,2*10^6$	$1,3*10^6$	5	6500	1,00037	0,04	52
Прочие программы	$1*10^5$	$0,5*10^5$	10	500	1,0116	0,08	4

Расчет показателей надежности операционной системы без учета вклада драйверов

Проведём моделирование с целью определения времён обнаружения ошибок после окончания тестирования. Результаты моделирования для первых 40 ошибок приведены на рис. 4.2.

Согласно формуле Стерджеса оптимальное значение величины интервала, которое позволяет выявить характерные закономерности рассматриваемого явления при минимальном количестве интервалов, составит:

$$H=R/(1+3,32*\lg n),$$

где n – количество наблюдаемых значений;

$R=X_{max} - X_{min}$.

Левая граница первого интервала определяется как:

$$a_1= X_{min}-H/2.$$

Правая граница i-го интервала совпадает с левой границей (i+1)-го интервала

$$b_i=a_{i+1}.$$

Правая граница последнего интервала должна удовлетворять условию:

$$b_i \geq X_{max}.$$

Интервалы, для которых частота попадания менее 5, следует объединить с соседними интервалами.

Расчеты показателей надежности, проведённые по этой формуле на основании полученных при моделировании данных, представлены в табл. 4.2.

$$X_{max} = 559$$
$$X_{min} = 52$$
$$H = 80.$$

GPSS World - [Надежность КЦ_5.149:2 - MATRIX WINDOW]

File Edit Search View Command Window Help

MM2

Dim 1	Dim 2: 1
7441	162.600
7442	160.736
7443	168.572
7444	128.473
7445	144.087
7446	108.076
7447	142.629
7448	185.105
7449	225.903
7450	191.920
7451	232.178
7452	106.128
7453	167.018
7454	220.657
7455	211.747
7456	52.263
7457	67.319
7458	309.118
7459	201.979
7460	151.286
7461	329.817
7462	243.253
7463	278.270
7464	321.985
7465	307.631
7466	254.157
7467	322.347
7468	168.684
7469	260.832
7470	442.729

Dim 1	Dim 2: 1
7470	442.729
7471	229.883
7472	242.583
7473	328.468
7474	272.727
7475	440.290
7476	234.057
7477	307.352
7478	559.150
7479	198.981
7480	447.497

Рис. 4.2. Времена обнаружения первых 40 ошибок после окончания тестирования для операционной системы без учёта вклада драйверов

Особенностью надежности программного обеспечения по сравнению с надежностью аппаратуры является то, что одно и то же ПО функционирует одновременно на большом количестве компьютеров. При этом количество обнаруженных ошибок пропорционально количеству компьютеров. В то же время ошибка, обнаруженная на одном компьютере, одновременно исправляется на всех других компьютерах. Исходя из этого, полученные оценки являются интегральными для потока отказов ПО по всем компьютерам, на которых оно установлено.

Полученные значения среднего времени наработки на отказ хорошо согласуются с информацией о том, что за время выпуска в продажу Windows XP фирмой Microsoft было официально выпущено 16 патчей для неё. Если учесть, что Windows XP эксплуатируется около 5 лет, а каждый патч в среднем содержит исправления для 3-х ошибок, то среднее время наработки на отказ составит 547 часов, а интенсивность отказов 0,0018 $час^{-1}$.

Трасса выполнения программы для каждого конкретного применения является уникальной. Ввиду практически неограниченного количества таких трасс для всех возможных использований, интенсивность обнаружения ошибок программного обеспечения на конкретном компьютере обратно пропорциональна количеству компьютеров, на которых используется данная программа.

Оценим интенсивность отказов программного обеспечения, исходя из полученных результатов моделирования и количества используемых копий операционной системы.

По данным маркетинговой компании Forrester Research [10] в настоящее время количество персональных компьютеров в мире составляет около 900 млн штук. В то же время приблизительно на 80% компьютеров установлена операционная система Windows XP. Таким образом, эта операционная система функционирует приблизительно на 720 млн компьютеров.

Отсюда можно сделать вывод, что интенсивность отказов операционной системы Windows XP из-за ошибок в программе составит для отдельного компьютера

$$\lambda_{OC}=0{,}24*10^{-2}/(720*10^{6})=0{,}33*10^{-11}\ ч^{-1}\ .$$

Таблица 4.2. Результаты расчета показателей надёжности ОС

a_i	b_i	Частота	Объединенные интервалы						
			a_i	b_i	Частота	Относит. частота	λ_i, 1/час	$\lambda_{ср}$, 1/час	$T_{отк}$, час
12	92	2	12	172	13	0,325	0,00203	0,00240	417,21
92	172	11	172	253	12	0,300	0,00370		
172	253	12	253	333	11	0,275	0,00344		
253	333	11	333	574	4	0,100	0,00041		
333	413	0							
413	493	3							
493	574	1							

Расчет показателей надежности операционной системы с учётом вклада драйверов

Для операционной системы учтём, как указывалось выше, наличие в составе ОС драйверов, имеющих, как минимум, в 3 раза большее количество ошибок, чем собственно системные программы. К тому же объём драйверов составляет до 70% всего объёма ОС.

Проведём моделирование с целью определения времён обнаружения ошибок в операционной системе после окончания тестирования с учётом указанных соображений.

Исходные данные для моделирования представлены в табл. 4.2. Отметим, что они хорошо согласуются с тем фактом, что 85% всех аварийных отказов Windows XP обуславливается наличием ошибок в коде драйверов [7] (согласно табл.4.2 эта величина составляет 126/144=0,875).

Результаты моделирования для первых 40 ошибок приведены на рис. 4.3.

Результаты расчетов по полученным данным представлены в табл. 4.3.

$$X_{max} = 166$$
$$X_{min} = 25$$
$$H = 22$$

GPSS World - [Надежность ОС и драйверов.154:2 - MATRIX WINDOW]

File Edit Search View Command Window Help

MM2

Dim 1	Dim 2: 1
17857	96.880
17858	94.758
17859	98.312
17860	74.093
17861	82.227
17862	60.997
17863	79.654
17864	102.192
17865	123.145
17866	103.182
17867	123.161
17868	55.481
17869	86.254
17870	112.391
17871	106.213
17872	25.812
17873	32.839
17874	148.846
17875	95.454
17876	70.304
17877	150.815
17878	108.965
17879	122.275
17880	138.600
17881	129.514
17882	104.614
17883	129.784
17884	66.282
17885	100.292
17886	166.110

Dim 1	Dim 2: 1
17857	96.880
17858	94.758
17859	98.312
17860	74.093
17861	82.227
17862	60.997
17863	79.654
17864	102.192
17865	123.145
17866	103.182
17867	123.161

Рис. 4.3. Времена обнаружения первых 40 ошибок после окончания тестирования для операционной системы с учётом вклада драйверов

Таблица 4.3. Результаты расчета показателей надёжности ОС с учетом драйверов

a_i	b_i	Частота	Объединенные интервалы			Относит. частота	λ_i 1/час	$\lambda_{ср}$, 1/час	$T_{отк}$, час
			a_i	b_i	Частота				
13	35	2	13	79	10	0,25	0,00379	0,00833	120,00
35	57	2	79	101	11	0,275	0,01250		
57	79	6	101	123	11	0,275	0,01250		
79	101	11	123	167	8	0,2	0,00455		
101	123	11							
123	145	4							
145	167	4							

Результаты показывают, что в этом случае интенсивность обнаружения ошибок существенно увеличивается по сравнению с ситуацией, когда драйверы не выделялись в отдельную группу.

Исходя из рассуждений, приведённых в предыдущем разделе, получим, что интенсивность отказов операционной системы Windows XP из-за ошибок в программе составит для отдельного компьютера:

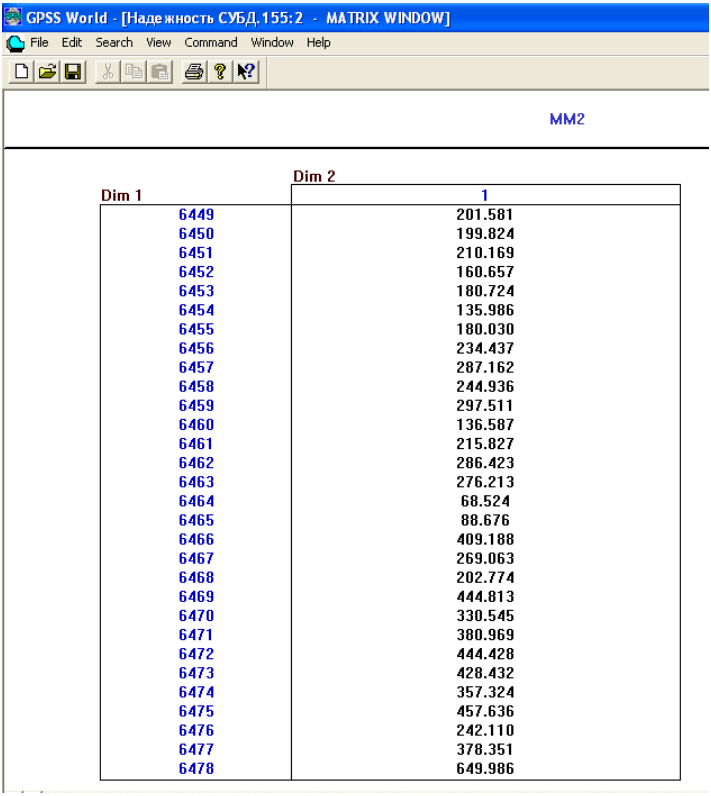
$$\lambda_{OC}=0{,}833*10^{-2}/(720*10^{6})=1{,}16*10^{-11}\ ч^{-1}.$$

Расчет показателей надежности СУБД Oracle

Результаты моделирования для первых 40 ошибок СУБД ORACLE, обнаруживаемых после завершения этапа тестирования, по исходным данным, представленным в табл. 4.2, приведены на рис. 4.4.

Результаты расчетов по полученным данным представлены в табл. 4.4.

$$X_{max} = 952$$
$$X_{min} = 68$$
$$H = 140$$



Dim 1	Dim 2: 1
6449	201.581
6450	199.824
6451	210.169
6452	160.657
6453	180.724
6454	135.986
6455	180.030
6456	234.437
6457	287.162
6458	244.936
6459	297.511
6460	136.587
6461	215.827
6462	286.423
6463	276.213
6464	68.524
6465	88.676
6466	409.188
6467	269.063
6468	202.774
6469	444.813
6470	330.545
6471	380.969
6472	444.428
6473	428.432
6474	357.324
6475	457.636
6476	242.110
6477	378.351
6478	649.986

Dim 1	Dim 2: 1
6471	380.969
6472	444.428
6473	428.432
6474	357.324
6475	457.636
6476	242.110
6477	378.351
6478	649.986
6479	342.429
6480	366.411
6481	503.728

Рис. 4.4. Времена обнаружения первых 40 ошибок после окончания тестирования для СУБД ORACLE

Таблица 4.4. Результаты расчета показателей надёжности СУБД ORACLE

a_i	b_i	Частота	Объединенные интервалы			Относит. частота	λ_i 1/час	$\lambda_{ср}$, 1/час	$T_{отк}$, час
			a_i	b_i	Частота				
0	140	4	0	280	19	0,475	0,00170	0,00124	804,70
140	280	15	280	420	10	0,250	0,00179		
280	420	10	420	560	7	0,175	0,00125		
420	560	7	560	979	4	0,100	0,00024		
560	700	2							
700	840	1							

Принимая, что программа установлена примерно на 1 млн компьютеров, получим что интенсивность отказов для СУБД ORACLE на каждом компьютере составит:

$$\lambda_{СУБД}=0{,}124*10^{-2}/(1*10^{6})=0{,}124*10^{-8}\ ч^{-1}.$$

Кодирование баз данных

При ручной подготовке машинных носителей при однократном фиксировании исходные данные имеют вероятность искажения около 10^{-3}– 10^{-4} на символ [12]. Дублированной подготовкой и логическим контролем вероятность технологической ошибки может быть снижена до уровня 10^{-5} – 10^{-7} на символ. Многократный перекрестный контроль соответствия данных в ЭВМ исходным данным позволяет доводить в отдельных случаях вероятность технологической ошибки в программе до уровня 10^{-7} – 10^{-8} [3].

Непосредственное участие человека в подготовке данных для ввода в ЭВМ и при анализе результатов функционирования программ по данным на дисплеях определяет в значительной степени их уровень достоверности и не позволяет полностью пренебрегать этим типом ошибок в программах.

Принимая в качестве расчетной наиболее достоверную оценку в 10^{-7} ошибок на один символ базы данных и оценивая объем используемых баз данных в 100 МБайт, получим, что база данных содержит около 10 ошибок кодирования.

Эти ошибки также должны быть учтены при анализе надежности программного обеспечения. Однако для этого необходимы значения интенсивности работы с базой данных. Если полное обновление базы данных проходит за 1 месяц, то интенсивность отказов, вызванных технологическими ошибками при кодировании, составит:

$$\lambda_{БД}=10 / (100*10^{6}*30*24) =0{,}139*10^{-9}\ ч^{-1}.$$

Расчет показателей надежности прикладных программ

Для прикладных программ моделирование даёт времена обнаружения 3-х ближайших ошибок, равными: t_1=1846, t_2=2731, t_3=5050.

При этом средняя интенсивность отказов составит:

$$\lambda ср=0{,}000369\ ч^{-1},$$

а среднее время безотказной работы:

$$Tср= 2712{,}728\ ч.$$

Считая, что программы используются на 100 компьютерах, получим, что интенсивность отказов прикладных программ на любом из компьютеров составит $0{,}369*10^{-5}$ $ч^{-1}$, а наработка на отказ 271272,8 ч, то есть около 31 года.

4.3.6. Расчет показателей надёжности программного обеспечения для серверов и рабочих станций вычислительных средств системы передачи данных между Кризисным центром и атомными станциями

Вычислительные средства системы передачи данных можно разделить 3 группы по составу используемого программного обеспечения:

1) ОС, прикладные программы;
2) ОС, СУБД, база данных;
3) ОС, СУБД, база данных, прикладные программы.

Считая, что отказы каждого из программных средств независимы, получим интенсивности отказов каждой из выделенных групп:

$$\lambda_1 = \lambda_{ОС} + \lambda_{ПП} = 0{,}116*10^{-9} + 0{,}369*10^{-5} \approx 0{,}369*10^{-5}\ ч^{-1}$$

$$\lambda_2 = \lambda_{ОС} + \lambda_{СУБД} + \lambda_{БД} = 0{,}116*10^{-10} + 0{,}124*10^{-8} + 0{,}139*10^{-9} \approx$$
$$\approx 0{,}149*10^{-8}\ ч^{-1}$$

$$\lambda_3 = \lambda_{ОС} + \lambda_{СУБД} + \lambda_{БД} + \lambda_{ПП} = 0{,}116*10^{-10} + 0{,}124*10^{-8} +$$
$$+0{,}139*10^{-9} + 0{,}369*10^{-5} \approx 0{,}369*10^{-5}\ ч^{-1}$$

Таким образом, для тех вычислительных средств, на которых используются прикладные программы, надёжность их программного обеспечения определяется, в основном, надёжностью этих программ. В остальных случаях надёжность ПО определяется сочетанием показателей надёжности системных средств.

Список литературы к главе 4

1. Роберт Т. Фатрелл, Дональд Ф. Шафер, Линда И. Шафер. Управление программными проектами: достижение оптимального качества при минимуме затрат: Пер. с англ. М.: Изд. дом "Вильямс", 2004. 1136 с.
2. A Lightweight Method for Building Reliable Operating Systems Despite Unreliable Device Drivers. // Technical Report IR-CS-018, January 2006.
3. Липаев В.В. Анализ и сокращение рисков проектов сложных программных средств. М.: СИНТЕГ, 2005. 224 с.
4. Symantec: тенденции и прогнозы эволюции сетевых угроз. - Электронный журнал "Спамтест". Декабрь, 2007. http://www.spamtest.ru/news.html?id=207509046
5. Нупур Д, Хамфри У., Редвайн С., Цибульски Г., Макгро Г. Процессы разработки безопасного программного обеспечения. // Открытые системы, №8, 2004.
6. Смирнов В.Е., Смирнов Е.В., Монахов К.В., Шабанов А.К. Математическая модель оценки надежности аппаратно-программного комплекса // 60-я Научная сессия, посвященная Дню радио, 17-19 мая 2005г. С.-Пб., ЗАО АВТЭКС. С. 91-93.
7. Надежность технических систем: справочник. /Под ред. И.А. Ушакова М.: Радио и связь, 1985. 608 с.
8. Майерс Г. Надежность программного обеспечения. М.: Мир, 1980. 360с.
9. Липаев В.В. Проектирование программных средств: Учебн. пособие для вузов. М.: Высш. шк., 1990. 303 с.
10. http://www.oracle.com/technology/software/products/content-management/index.html?rssid=rss_otn_soft
11. К 2008 году число персональных компьютеров в мире достигнет миллиарда: Лента.ru. 12 июня 2007. – http://www.lenta.ru/
12. Хердер Йоррит, Бос Херберт, Таненбаум Эндрью, Построение надежных операционных систем, допускающих наличие ненадежных драйверов устройств. http://www.citforum.ru/operating_systems/reliable_os

ГЛАВА 5. МОДЕЛИ НАДЕЖНОСТИ АППАРАТНЫХ СРЕДСТВ СИСТЕМ ПЕРЕДАЧИ ДАННЫХ АЭС

5.1. Обеспечение надежности современных информационно-вычислительных систем

В настоящее время в связи с повышением масштабов и сложности проектируемых информационно-вычислительных систем (ИВС), в том числе систем передачи данных (СПД), все более остро встает проблема обеспечения их надежности. Особенно это характерно для ИВС ответственного целевого назначения, например для систем мониторинга критических технологических объектов (объектов повышенной степени опасности). К таким объектам, безусловно, относятся АЭС. На рис. 5.1 представлена достаточно общая классификация методов и средств обеспечения надежности современных ИВС в широком понимании термина **надежность**, то есть с точки зрения как безотказности систем, так и с точки зрения их контролепригодности и параметров их восстановления. Для восстанавливаемых ИВС обычно задают два существенных параметра: t_0 – максимальный перерыв в их работе, при котором система не считается в состоянии простоя (неготовности к выполнению целевой задачи), и T_0 – предельно допустимое время простоя (восстановления системы после функционального отказа).

Технологические средства повышения надежности включают в себя широкий спектр процессов повышения качества системных компонентов на стадии их изготовления. Средства контроля и диагностики на этапе эксплуатации системы обеспечивают корректную информацию об ее состоянии, снижая таким образом возможные потери информации и уменьшая время пребывания системы в нерабочем состоянии.

Среди структурных методов обеспечения надежности систем выделяются две группы методов: методы, основанные на резервировании их компонентов, и методы, предусматривающие реконфигурацию системы в случае отказовых ситуаций.

Методы резервирования систем предполагают неизменность базовой рабочей конфигурации системы в процессе ее работы. В случае отказов компонентов системы происходит простая замена отказавших блоков на исправные резервные блоки. Такая замена может происходить либо в рамках оперативного переключения оборудования (в этом случае время простоя системы считается несущественным), либо в рамках процесса восстановления отказавших блоков (в этом случае при анализе надежности системы учитывается конечное время восстановления).

Методы реконфигурации систем предполагают изменение рабочей конфигурации системы в процессе ее работы. При этом изменяются отдельные показатели ее производительности, например, объем памяти, быстродействие или параметры отказоустойчивости.

Основное внимание уделяется средствам восстановления системы (из первой группы методов) и методам активного резервирования (из второй группы методов).

Рис. 5.1. Классификация методов и средств обеспечения надежности современных информационно-вычислительных систем

5.2 Модели надежности сервера типа NT-AKPB-LEN

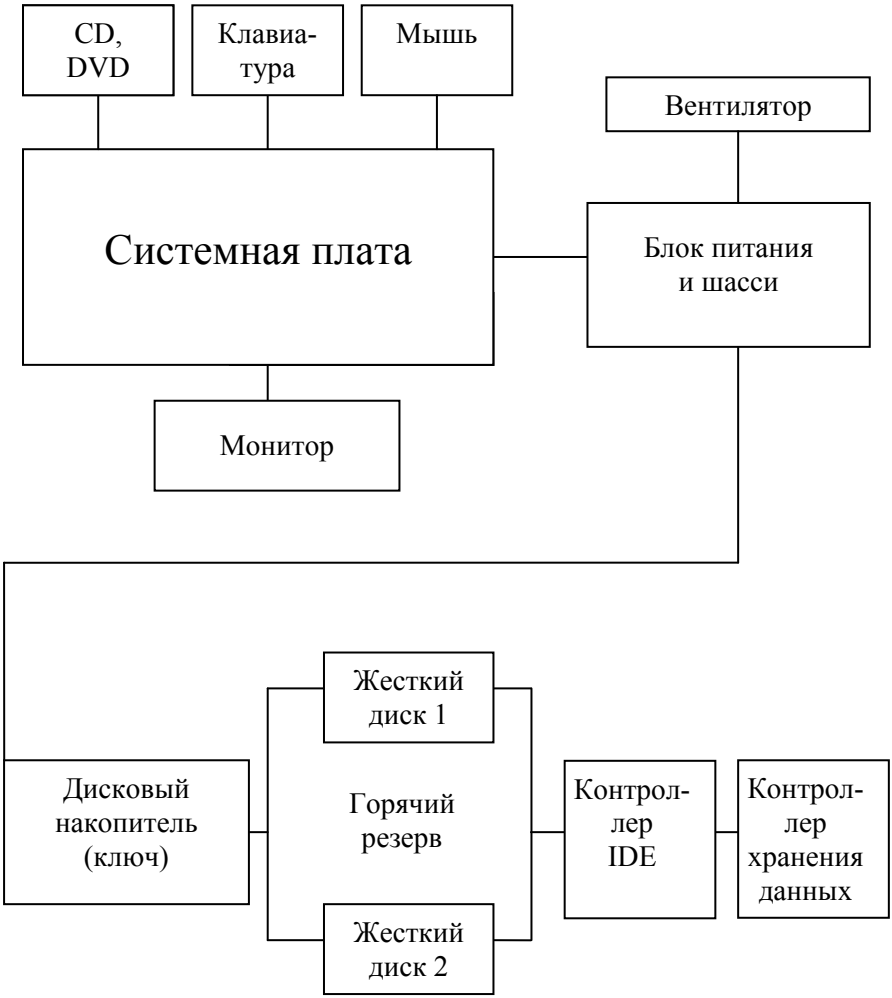


Рис.5.2. Схема компонентов сервера типа NT-AKPB-LEN для расчета его надежности

Сервер типа NT-AKPB-LEN является типовым базовым компонентом системы передачи данных (СПД), схему для расчета его надежности можно представить в виде структуры, представленной на рис. 5.2.

Для расчета надежностных характеристик сервера можно использовать несколько известных методов и методик анализа надежности, приведенных, например, в [1-4].

Расчет средней наработки сервера на отказ

Метод на основе параметров компонентов. В основе данного метода лежит известная формула суммирования параметров компонентов:

$$\Lambda = \sum_{i=1}^{N} \lambda_i \quad , \qquad (5.1)$$

где N – число различных компонентов сервера, λ_i – интенсивность отказов i-го компонента сервера, Λ – интенсивность отказов сервера.

Для простоты анализа предположим, что рассматриваются только те компоненты сервера и только те отказы этих компонентов, которые приводят к функциональному отказу сервера. Например, если отказ мыши приводит к нарушению функций сервера в одном случае из ста, то интенсивность отказов мыши для формулы (5.1) можно взять как $\lambda = 0{,}01*\lambda_0$, где λ_0 – интенсивность отказов мыши без учета их влияния на работу сервера.

В связи с тем, что сервер включает в себя резервированную (дублированную) подсистему жестких дисков, то его среднюю наработку на отказ T нельзя вычислять по простой формуле.

Для расчета средней наработки на отказ сервера можно использовать формулу:

$$T = \int_0^{\infty} P(t)dt, \qquad \text{где} \quad P(t) = e^{-\Lambda t} \cdot (2p_{жд} - p_{жд}^{\ 2}),$$

где $p_{жд} = \exp(-\lambda_{жд} t)$ — вероятность безотказной работы за время t одного жесткого диска сервера, Λ – сумма интенсивностей отказа всех компонентов сервера кроме жесткого диска ($\lambda_{жд}$).

В результате вычисления данного интеграла приходим к значению:

$$T = \frac{\Lambda + 3\lambda_{жд}}{(\Lambda + \lambda_{жд})(\Lambda + 2\lambda_{жд})} \qquad (5.2)$$

<u>Метод на основе обработки статистики</u>. Использование данного метода предполагает наличие некоторой статистической информации об отказах сервера. Метод целесообразно использовать в тех случаях, когда исходные надежностные параметры компонентов сервера труднодоступны (например, или не собраны, или составляют коммерческую тайну изготовителя). Если при этом есть статистические данные об отказах сервера, то для вычисления средней наработки на отказ достаточно применить известную формулу:

$$T = \frac{\sum_{i=1}^{M} T_i}{M},$$

где M – число зарегистрированных отказов сервера, T_i – время безотказной работы перед i-м зарегистрированным отказом.

Значения T вычисляются по правилу $T_i = T_{B\ i\text{-}1} - To_i$, где $T_{B\ i\text{-}1}$ – момент (календарный) завершения восстановления сервера после (i-1)-го отказа, To_i – момент i-го отказа.

Как правило, значения $T_{B\ i\text{-}1}$ и To_i берутся из журнала регистрации неполадок и нарушений в работе системы (станции). Естественно, при обработке статистических данных из журнала выбираются только те отказы, которые связаны с неполадками серверов СПД.

При расчете надежности сервера можно использовать ориентировочные показатели представленные в табл. 5.1 и приведенные, например, в [5 - 11].

Несмотря на относительно высокую интенсивность отказов ряда компонентов, их влияние на надежность сервера можно считать незначительным, если время их восстановления (например, из локально размещенного ЗИП) достаточно невелико.

Будем считать, что отказы устройств, время восстановления которых составляют менее допустимого интервала (например, 20 мин), слабо влияют на надежность сервера с функциональной точки зрения.

Таблица 5.1. Ориентировочные статистические данные по надежности основных компонентов сервера и рабочей станции

Номер компонента	Компонент	Диапазон наработки на отказ по экспертным оценкам (тыс. часов)
1	Системная плата	100-150
2	Жесткий диск	1000-1400
3	CD-ROM	60-125
4	Блок питания	50-70
5	Вентилятор	30-40
6	Контроллер хранения данных, IDE	500-1000
7	Монитор	50-100

К таким устройствам можно отнести периферийные устройства: мышь и клавиатуру. Отказы монитора влияют на надежность функционирования сервера только в определенных ситуациях, связанных с необходимостью срочного визуального анализа состояний или параметров. Если частоту таких ситуаций определить как f, то среднюю наработку сервера на функциональный отказ, связанный с отказом монитора, можно определить следующим образом:

Если f – частота попадания сервера в ситуации, требующие визуального отображения данных, параметров или состояний, а t_A – средняя продолжительность таких состояний, то вероятность нахождения сервера в подобных состояниях в произвольный момент времени запишется как $f \cdot t_A$. Если интенсивность отказов монитора обозначить как λ_M, то вероятность отказа монитора за некоторое время t (при малых значениях λ_M) запишется как $\lambda_M t$.

Перемножив обе вероятности, получим вероятность того, что при отказе монитора сервер попадет в состояние функционального отказа, то есть вероятность того, что при отказе монитора для продолжения работы будет необходима срочная визуализация информации. Таким образом, вероятность критического (то есть приво-

дящего к функциональному нарушению) отказа монитора за время t составит $f\,t_A\lambda_M t$.

Если теперь предположить, что времена таких критических отказов распределены по экспоненциальному закону, то их интенсивность $\lambda_{Мкрит}$ можно записать в следующем виде:

$$\lambda_{Мкрит} = \lambda_M r, \;\; где: r = f t_A \; .$$

Аналогичные соотношения можно записать и для других периферийных устройств, входящих в состав сервера. Теперь в расчетах надежности сервера вместо интенсивности отказов i-го компонента сервера можно брать интенсивность его критических отказов, то есть отказов, приводящих к функциональному отказу всего сервера. Для компонентов, отказы которых безусловно приводят к отказу сервера, соответствующие значения r будут равны единице. В этом смысле значения r можно интерпретировать как относительная доля всех отказов устройства, которые приводят к функциональному отказу сервера.

Возьмем в качестве примера следующие параметры компонентов сервера, приведенные в табл. 5.2.

Таблица 5.2. Экспертные данные по функциональным отказам компонентов сервера и рабочей станции

Компонент	**Интенсивность отказов (x 10^{-3} ч$^{-1}$)**	**Значение r (для рабочей станции)**
Системная плата	0,010	1
Жесткий диск	0,001	1
Блок питания	0,020	1
Контроллер данных, IDE	0,050	1
Монитор	0,010	0,05
CD-DVD	0,017	0,01
Клавиатура	0,010	0,01
Мышь	0,010	0,01

Общая интенсивность отказов сервера в уточненном виде запишется как:

$$\Lambda = \sum_{i=1}^{i=N} \lambda_i r_i \quad , \tag{5.3}$$

где N – число компонентов сервера, r_i – вероятность того, что отказ i-го компонента приводит к функциональному отказу сервера, λ_i – интенсивность отказов i-го компонента.

Подставляя значения из таблицы в формулу (5.3), получим окончательное значение интенсивности отказов сервера $\Lambda=0{,}082\cdot10^{-3}$, что соответствует средней наработке на отказ, примерно, 12 тыс. часов.

Следует отметить, что поскольку в составе сервера есть и неизбыточные, и резервированные устройства (дублированный жесткий диск), то интенсивность отказов, рассчитанная по формуле (5.1), не совсем точно характеризует функциональный отказ сервера. Если откажет один из двух дублированных жестких дисков, то функционального отказа сервера не произойдет, хотя сервер, по существу, перейдет в режим восстановления резервного блока. Исходя из этого, величина Λ по формуле (5.1) фактически определяет интенсивность отказа сервера с точки зрения генерации его запросов на техническое обслуживание (реконфигурацию, переключение, замену компонента). Формула (5.3) определяет запрос на устранение функционального отказа сервера (восстановление), если значение r для жесткого диска взять b/a, где a – общее число отказов дисков ($a=2T/t_0$), а b – число функциональных отказов дублированной пары дисков ($b=2T/3t_0$); t_0 – наработка на отказ жесткого диска, T – период времени.

Для расчета интенсивности функциональных отказов сервера следует либо использовать формулу (5.3), либо предположить, что формула (5.1) обеспечивает гарантированную нижнюю оценку интенсивности функционального отказа сервера.

Расчет среднего времени восстановления сервера

Параметры восстановления сервера можно определять несколькими традиционными способами:

- статистическим методом,
- аналитическим методом с учетом временной инфраструктуры восстановления,
- аналитическим методом с учетом пространственной инфраструктуры восстановления.

На основе статистики по периодам восстановления серверов после их отказов используется известная формула определения среднего времени восстановления сервера T_B после отказа [1]:

$$T_B = \frac{1}{M}\sum_{i=1}^{i=M} t_{Bi}.$$

где M – число статистических данных (опытов), t_{Bi} – время восстановления сервера после i-го отказа по статистике.

Среднее время восстановления сервера можно определить и аналитически. Если μ_i – среднее время восстановления *i*-го компонента системы, то (с учетом вероятности отказа того или иного компонента) среднее время восстановления всего сервера запишется как:

$$T_B = \frac{1}{\Lambda}\sum_{i=1}^{i=N} \frac{\lambda_i}{\mu_i},$$

где N – число различных компонентов сервера, λ_i – интенсивность отказов i-го компонента сервера, μ_i – интенсивность восстановления *i*-го компонента сервера, Λ – суммарная интенсивность отказов сервера, вычисляемая либо по формуле (5.1), либо по формуле (5.3). В первом случае получим среднее время восстановления сервера после любого отказа компонента. Во втором случае получим среднее время восстановления сервера после его функционального отказа.

Значение T_B можно рассчитать и по другой методике, исходя из соображений учета временной инфраструктуры восстановления. Например, если в течение суток 8 часов сервер работает в режиме быстрого локального восстановления (среднее время ремонта – t_1) , а 16 часов – в режиме медленного восстановления (или ожидания восстановления в течение среднего времени t_2), причем $t_1 << t_2$, то по методу усреднения, значение T_B можно рассчитать.

Для учета пространственной инфраструктуры восстановления рассмотрим организацию двух ее режимов: локального и удаленного. Локальный режим восстановления соответствует наличию на станции локальных ресурсов восстановления в форме, например,

одиночного ЗИП. Удаленный или дистанционный режим восстановления соответствует организации группового ЗИП, то есть наличию централизованного источника запасных ресурсов. Для такой формы организации введем следующие показатели:

Λ_1 – суммарная интенсивность отказов, устраняемых в локальном режиме (на основе одиночного ЗИП);

Λ_2 – суммарная интенсивность отказов, устраняемых в дистанционном режиме (на основе группового ЗИП);

T_1 – время восстановления в локальном режиме (с помощью одиночного ЗИП);

T_2 – время восстановления в дистанционном режиме (с помощью группового ЗИП).

Тогда можно записать среднее время восстановления сервера в следующем виде:

$$T_B = \frac{\Lambda_1 T_1 + \Lambda_2 T_2}{\Lambda}.$$

Если дистанционное восстановление зависит от пространственной инфраструктуры распределения резервных ресурсов, то значения Λ_1 и Λ_2 , естественно, зависят от состава локального и группового комплектов ЗИП. Чем больше запасного оборудования будет в составе локального (одиночного) ЗИП, тем быстрее будет восстанавливаться сервер, и тем меньше будет значение T_B. Чем больше запасного оборудования будет в составе удаленного (группового) ЗИП, тем более гибко будет организован процесс восстановления отказавших компонентов, и более эффективно будет использоваться запасное оборудование.

На практике выбор такого состава определяется главным образом стоимостными ограничениями на обеспечение компонентами с учетом временных параметров доставки запасных компонентов на место неполадок и нарушений. Попробуем формализовать задачу оптимизации соотношения локального и дистанционного ЗИП.

Введем две логических переменных x_i и y_i, которые будут иметь следующие значения:

x_i =1 означает, что i-й компонент сервера имеет резерв в локальном резерве;

y_i =1 означает, что i-й компонент сервера имеет резерв в удаленном резерве.

Для простоты предположим, что число компонентов i-го типа в рабочем и запасном комплектах сервера равно единице. Обозначим c_{i1} и c_{i2} – затраты на поддержку запасного компонента, соответственно, в локальном и дистанционном комплектах ЗИП. Тогда общая стоимость (обобщенные затраты) на организацию инфраструктуры восстановления компонента запишется следующим образом:

$$C = \sum_{i=1}^{i=N} (c_{i1} x_i + c_{i2} y_i). \tag{5.4}$$

Среднее время восстановления компонента при этом выразится формулой:

$$T_B = \sum_{i=1}^{i=N} (T_{i1} x_i + T_{i2} y_i), \tag{5.5}$$

где T_{i1} и T_{i2} – среднее время восстановления i-го компонента, соответственно, в локальном и дистанционном режиме.

Таким образом, задача может быть сформулирована следующим образом. Для сервера с двумя типами восстановления необходимо найти минимум функционала (5.5.) при ограничении C_0 на обобщенную стоимость инфраструктуры (5.4.). Другими словами, надо минимизировать среднее время восстановления сервера T_B при ограничении на стоимость $C<C_0$.

Более общий случай расчета, когда учитывается возможность наличия нескольких компонентов i-го типа, как в рабочей конфигурации, так и в обоих резервных комплектах, рассмотрен в разделах 5.4 и 5.5.

5.3. Модели надежности систем передачи данных

Для построения модели надежности СПД рассмотрим ее типовое коммуникационное звено, называемое типовым направлением связи (ТНС). В состав ТНС входят следующие главные компоненты: два сервера (один со стороны центра и один со стороны станции), два взаимодублирующих канала связи и две дублированные пары маршрутизаторов. Общая структура ТНС (без маршрутизаторов) представлена на рис. 5.3.

Предположим, что восстановление такой системы неограниченно, то есть любые отказавшие компоненты данной системы восстанавливаются параллельно и независимо друг от друга. Предположим также, что все серверы и маршрутизаторы в системе одинаковы по показателям надежности. Кроме того, обозначим:

$Кг_С$ – коэффициент готовности сервера, $Кг_М$ – коэффициент готовности маршрутизатора, $Кг_{к1}$ и $Кг_{к2}$ – коэффициенты готовности, соответственно, первого и второго каналов связи.

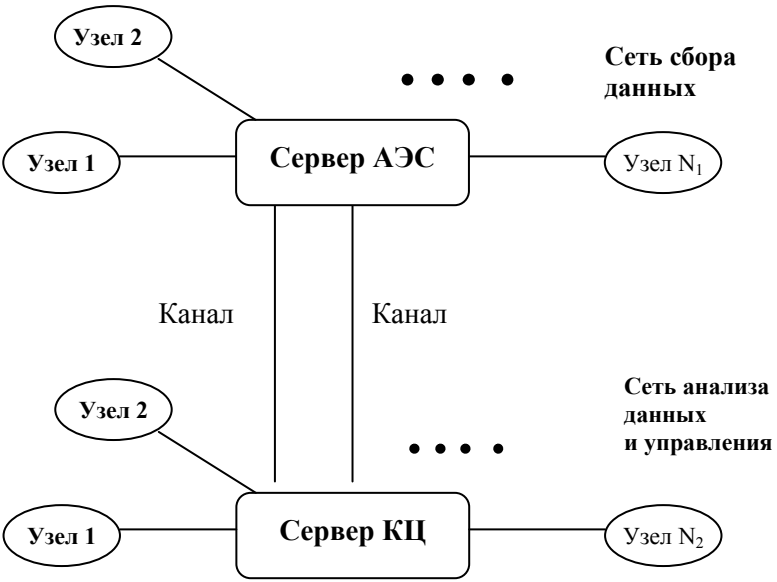


Рис. 5.3. Топологическая схема одного коммуникационного звена СПД (кризисный центр (КЦ) – АЭС)

Тогда при общем резервировании маршрутизаторов, представленном на рис. 5.4а, надежность (коэффициент готовности) ТНС запишется следующим образом (N=2):

$$Кг_{ТНС} = Кг_C{}^2(1-(1-Кг_M{}^2\ Кг_{K1})(1-Кг_M{}^2\ Кг_{K2})).$$

Для случая раздельного резервирования маршрутизаторов и каналов, представленного на рис. 5.4б, коэффициент готовности системы запишется следующим образом:

$$Кг_{ТНС} = Кг_C{}^2(1-(1-Кг_{K1})(1-Кг_{K2})(1-(1-Кг_M))^2.$$

Поскольку ТНС состоит из нерезервированной части (серверов) и дублированной части (коммуникационных компонентов), то основное влияние на общую надежность системы оказывают параметры надежности каждого сервера. В предыдущих разделах исследовались вопросы моделирования надежности сервера с точки зрения способов расчета характеристик наработки на отказ (T=1/Λ) и среднего времени восстановления T_B одного сервера системы. С помощью данных значений можно определить коэффициент готовности одного сервера:

$$Кг_C = \frac{T}{T+T_B}.$$

Рассмотрим теперь методы для расчета значений T_B, которые более точно учитывали бы конкретную пространственную конфигурацию обеспечения резервным и запасным оборудованием через одиночный и групповой ЗИП.

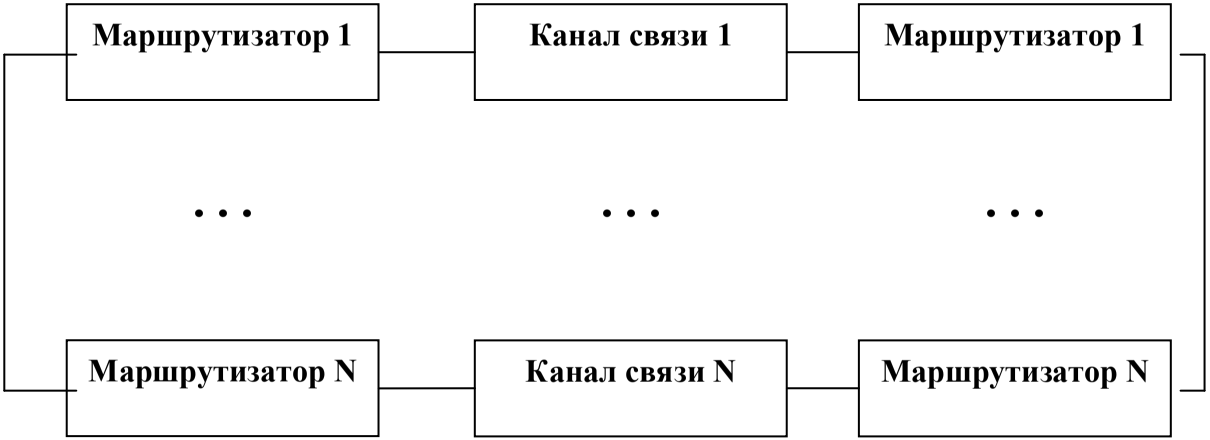


Рис. 5.4а. Общая схема резервирования коммуникационных компонентов N-канальной СПД

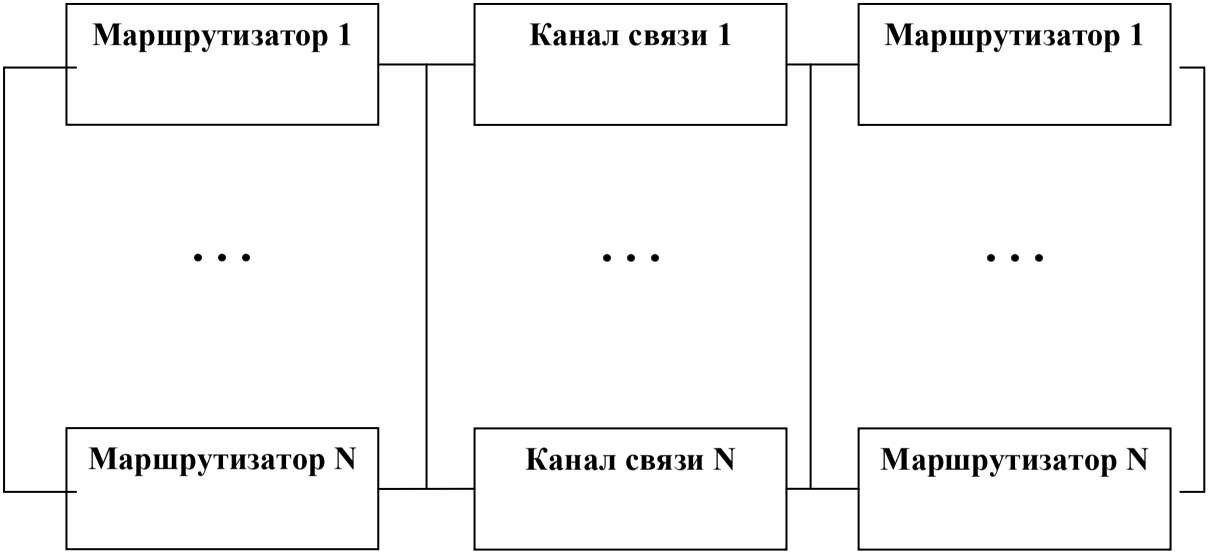


Рис. 5.4б. Раздельная схема резервирования коммуникационных компонентов N-канальной СПД.

5.4. Расчет параметров одиночного и группового ЗИП

При локальной организации ЗИП запасные компоненты каждого типа размещены в месте эксплуатации серверов. Предположим, что для каждого типа компонентов сервера существует m компонентов, работающих в составе сервера, *k* компонентов, находящихся в резерве (время их подключения вместо отказавших компонентов не превышает допустимого порога простоя t_0) и n запасных компонентов, находящихся в составе локального (одиночного) ЗИП в режиме хранения. Обозначим также λ_1, λ_2, и λ_3 - интенсивности отказов компонентов в соответствующих режимах работы, резерва и хранения.

Тогда интенсивность запросов на замену компонентов данного типа запишется как

$$\Lambda = (m\lambda_1 + k\lambda_2)(1+\delta) + n\lambda_3, \qquad (5.6)$$

где δ – процент ошибочных изъятий компонентов относительно общего потока отказов.

Среднее число заявок на замену компонентов данного типа за время пополнения ЗИП $T_п$ составит:

$$a = \Lambda T_П \ . \qquad (5.7)$$

Если предположить, что локальный ЗИП пополняется из удаленного центра поддержки, то значение *a* определяется, в основном, временем доставки комплекта или компонентов комплекта. С учетом методики, представленной в [13] и при непрерывной стратегии пополнения запасов ЗИП можно записать коэффициент готовности ЗИП по конкретному типу компонентов, как:

$$Кг_{ЗИП} = 1 - \frac{a^{n+1}}{(n+1)!\sum_{j=0}^{n+1}\frac{a^j}{j!}} \ .$$

Среднее время задержки в удовлетворении запросов на компоненты данного типа составит в этом случае:

$$T_{B_{ЗИП}} = \frac{-\ln Кг_{ЗИП}}{\Lambda} \ .$$

В режиме организации группового ЗИП поддержка восстановления отказавших компонентов сервера осуществляется из общего центра. Время пополнения ЗИП в этом случае определяется главным образом временем поиска и закупки недостающих компонентов. Для режима группового ЗИП значение коэффициента готовности ЗИП по конкретному типу компонентов в соответствии с [13] запишется как:

$$Кг_{ЗИП} = 1 - \frac{e^{-a}}{a} \sum_{j=n+2}^{\infty} (j - n - 1) \frac{a^j}{j!} .$$

Соответствующее значение времени задержки восстановления из группового ЗИП составит:

$$T_{B_{ЗИП}} = \frac{a(1 - Кг_{ЗИП})}{\Lambda}.$$

В силу относительной сложности формул расчета одиночного и группового ЗИП и с целью поддержки исследований надежностных показателей системы можно использовать программу, позволяющую автоматизировать расчет и исследование надежности ЗИП.

5.5. Программа расчета параметров ЗИП

Программа написана на языке Object Pascal в среде визуального объектно-ориентированного программирования DELPHI 7. Программа обеспечивает дружественный пользовательский интерфейс для автоматизированного расчета параметров одиночного и группового ЗИП (см. раздел 5.4).

Рассмотрим пример использования программы при исследовании надежности одиночного ЗИП при изменении трех параметров:

- интенсивности отказов рабочих компонентов сервера,
- диапазона исследования данного параметра,
- времени пополнения ЗИП.

После ввода исходных параметров для расчета одиночного ЗИП и нажатии кнопки ВВОД происходит вывод значений коэффициента готовности одиночного ЗИП и среднего времени удовлетворения запросов на компоненты заданного типа. Соответствующая экранная форма представлена на рис. 5.5.

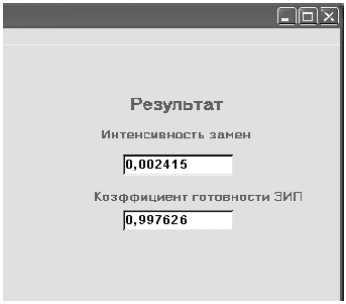


Рис. 5.5. Экранная форма вывода расчетов одиночного ЗИП

На рис. 5.6 приведена экранная форма для проведения необходимых исследований рассчитанных характеристик от выбранных параметров при выборе режима работы.

В программу введены средства скроллинга (прокрутки) исследуемых параметров в нужном диапазоне, которые позволяют в режиме реального времени визуализировать изменения данных графиков при малейшем изменении параметров. Как видно из рисунков, такими параметрами являются: интенсивности отказов компонентов в каждом из трех режимов, диапазон их исследования и время пополнения ЗИП.

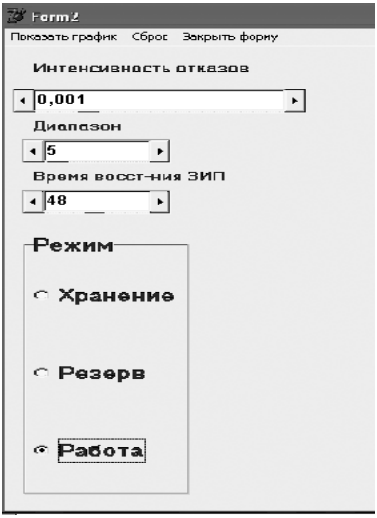


Рис. 5.6. Экранная форма для проведения исследований надежности.

Список литературы к главе 5

1. Козлов Б.А., Ушаков И.А. Справочник по расчету надежности аппаратуры радиоэлектроники и автоматики. М.: Сов. радио, 1975.
2. Надежность автоматизированных систем управления/под ред. Я.А. Хетагурова. М.: Высшая школа, 1979.
3. Справочник по цифровой вычислительной технике (электронные вычислительные машины и системы). /Б.Н. Малиновский, В.Я. Александров, В.П. Боюн и др. /Под ред. Б.Н.Малиновского. К.: Техника, 1980.
4. Александрович А.Е., Бородакий Ю.В., Чуканов В.О. Проектирование высоконадежных информационно-вычислительных систем. М.: Радио и связь, 2004.
5. Серверы серии ПАМИР. – http://www.ase.ru/pamir/vk.htm.
6. Отказоустойчивый сервер начального уровня NEC Express 5800/320lb и Straus ftServer 3300. – http://www.server.md/news.php.
7. Отказоустойчивые серверы. – http://www.rodnik.ru/htmls/s_1_5.htm.
8. Мощные блоки питания для видеокарт – ThermalTake – http://www.3dnews.ru/news/
9. Samsung объявила о начале продаж HDD емкостью 1 Тб. http://www.3dnews.ru/news/
10. Тестирование блока питания Seasonic S12II-500. - http://www.3dnews.ru/cooling/
11. Экстремальное воздушное охлаждение видеокарт: двухголовый ThermalTake Duorb против термоэлектрического Titan Elena TEC. - http://www.3dnews.ru/cooling/
12. А. Харьковский. Показатели надежности жестких дисков серьезно завышены. http://www.3dnews.ru/news/
13. Методика оценки и расчета запасов в комплектах ЗИП. РДВ50-503-84.

Всеволод Озирисович Чуканов

Надежность программного обеспечения и аппаратных средств систем передачи данных атомных электростанций

Учебное пособие

Редактор С.В. Бирюкова

Отпечатано в полном соответствии с представленным автором оригиналом-макетом

Подписано в печать 10.11.08. Формат 60x84 1/16.
Печ. л. 10,5. Уч.-изд. л. 11,0. Тираж 150 экз.
Изд. № 4/90 Заказ № 2-2427

Московский инженерно-физический институт (государственный университет).
115409, Москва, Каширское ш., 31

Типография издательства "Тровант".
г. Троицк Московской обл.