Министерство науки и высшего образования РФ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
АМУРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
(ФГБОУ ВО «АмГУ»)

Надежность автоматизированных систем

сборник научно-методических материалов

для направления подготовки 10.03.01 – Информационная безопасность

Благовещенск, 2019

Печатается по решению
редакционно-издательского совета
факультета математики и информатики
Амурского государственного
университета

Составитель: Еремина В.В.
Надежность автоматизированных систем: сборник учебно-методических материалов для направления подготовки 10.03.01. – Благовещенск: Амурский гос. ун-т, 2019.

Краткое изложение лекционного материала

Тема 1. Основные понятия теории надежности. Критерии надежности систем

§1. Основные понятия теории надежности

Опр. ***Надежность*** – это свойство объекта выполнять заданные функции, сохраняя во времени значения установленных эксплуатационных показателей в заданных пределах, соответствующих заданным режимам и условиям использования, технического обслуживания, ремонта, хранения и транспортирования.

Т.о., надежность является внутренним свойством системы, заложенным при ее изготовлении и проявляющимся при эксплуатации. Это свойство проявляется только во времени и без более или менее длительного наблюдения нельзя сделать заключения о надежности системы. Характеристики надежности зависят от условий эксплуатации, поэтому при определении их значений необходимо учитывать ее особенности.

Свойства надежности изделия изучаются теорией надежности.

Опр. ***Теория надежности*** – это система определенных идей, математических моделей и методов, направленных на решение проблем предсказания, оценки и оптимизации различных показателей надежности. Для количественной оценки надежности применяется ряд показателей, выбор которых зависит от конкретного типа и области применения системы. В основе этих характеристик лежат понятия о двух возможных состояниях объекта или системы: работоспособном и неработоспособном.

Опр. ***Работоспособным*** называется такое состояние объекта, при котором он способен выполнять заданные функции с параметрами, установленными требованиями технической документации. В процессе функционирования возможен переход объекта из работоспособного состояния в неработоспособное состояние и обратно.

С этими переходами связаны события отказа и восстановления.

Опр. ***Отказ*** – это событие, заключающееся в нарушении работоспособности.

Опр. ***Восстановление*** – это событие, заключающееся в переходе объекта из неработоспособного состояния в работоспособное в результате устранения отказа.

Т.о., отказ связан с нарушением требований технической документации, соответствующих работоспособному состоянию объекта. Однако не всякое нарушение требований приводит к отказу; возможны также события, называемые неисправностью. При ***неисправностях*** работоспособность объекта в значительной степени сохраняется, и только частично нарушаются второстепенные требования.

По полноте мероприятий, которые необходимо проводить для восстановления, различаются отказы:

1. ***устойчивые***, для устранения которых требуется проведение специальных мероприятий по восстановлению работоспособности;
2. ***сбой*** – это самоустраняющийся отказ, не требующий внешнего вмешательства ля замены отказавших компонент. Самоустраняющийся отказ характеризуется достаточно быстрым восстановлением работоспособности без внешнего вмешательства;
3. ***перемежающийся отказ*** представляет собой многократно повторяющиеся сбои, для полного устранения которых требуется внешнее вмешательство.

Содержание понятия надежности раскрывается совокупностью таких понятий как:

- ***безотказность*** характеризует свойство объекта сохранять работоспособность без вынужденных перерывов. Этому понятию соответствует измеряемый показатель длительности наработки (функционирования) до первого отказа, если после этого прекращается эксплуатация объекта, или среднее время между соседними отказами, если объект используется многократно с восстановлением после каждого отказа. Кроме того, это свойство объекта можно измерять вероятностью безотказной работы в течение заданного интервала времени и интенсивностью отказов в единицу времени;

- ***восстанавливаемость*** учитывает способность объекта к предупреждению, обнаружению и устранению отказов. Это свойство зависит от характера самого объекта, а также от условий его эксплуатации. Основным измеряемым показателем является время восстановления, которое включает в себя время обнаружения отказа, время его локализации, время устранения отказа и время предпусковой проверки работоспособности. Все составляющие времени восстановления зависят от методов, применяемых при эксплуатации, от степени автоматизации восстановительных работ, количества и уровня квалификации специалистов, восстанавливающих работоспособность объекта;
- ***сохраняемость***;
- ***долговечность системы***.

По возможности восстановления работоспособности в процессе эксплуатации объекты делятся на:
- ***невосстанавливаемые объекты***, которые не допускают ремонта или замены отказавших компонент и не обладают самовосстанавливаемостью во время выполнения своих функций. Такие объекты или системы могут эксплуатироваться либо до первого отказа, либо до полного выполнения своих функций, либо до момента достижения некоторого предельного состояния, после чего они снимаются с эксплуатации. Длительность возможного последующего ремонта при этом не учитывается.
- ***восстанавливаемые объекты*** допускают ремонт и замену отказавших компонент или обладают возможностью самовосстановления. При этом существенную роль играет длительность пребывания в неработоспособном состоянии и проведения восстановительных работ. Рентабельность восстановления и допустимая длительность восстановительных работ зависят от конкретного назначения и условий эксплуатации системы. В некоторых случаях при наличии принципиальной возможности восстановления оно может оказаться чересчур дорогим и сложным, в результате чего объект целесообразно рассматривать как невосстанавливаемый.

§2. Критерии надежности систем

Опр. ***Критерии*** – это показатели, позволяющие оценить предпочтительность тех или иных решений при создании и эксплуатации системы по степени достижения основных целей с учетом затрат, при которых эти цели достигаются.

При исследовании надежности основная цель состоит в разработке эффективных методов и обеспечении длительной работоспособности систем с заданными функциональными характеристиками. Для этого необходимо установить количественные показатели, характеризующие не только факт работоспособности системы, но и степень соответствия работоспособности основным требованиям, отраженным в технической документации.

Критерии, используемые в теории надежности, являются вероятностными, статистическими и в том или ином виде учитывают временные показатели.

I. Для невосстанавливаемых систем основным критерием является ***длительность наработки до первого отказа*** T. Вероятностные характеристики этой СВ в нескольких формах могут рассматриваться как критерии надежности. ***Вероятностью безотказной работы*** называется вероятность того, что при заданных условиях эксплуатации в течение интервала времени t не возникнет отказа, т. е. система будет работоспособна:

$$P(t) = P(T \geq t). \qquad (1)$$

Соответственно, ***вероятностью отказа*** является вероятность того, что в течение заданного времени произойдет хотя бы один отказ:

$$Q(t) = P(T < t) = 1 - P(t) \quad . \qquad (2)$$

Частота отказов $\alpha(t)$ есть плотность распределения времени работы до первого отказа:

$$\alpha(t) = Q'(t) = -P'(t);\ P(t) = \int_t^{\infty} \alpha(x)dx\,. \qquad (3)$$

Интегральным критерием является ***интенсивность отказов*** $\lambda(t)$, которая характеризует плотность распределения наработки до первого отказа, рассчитанную при условии, что до рассмотренного момента времени система проработала безотказно:

$$\lambda(t)=\frac{\alpha(t)}{P(t)}=-\frac{d\ln P(t)}{dt}. \qquad (4)$$

Перечисленные показатели связаны между собой, и их применение определяется спецификой конкретной системы. При этом предполагается. Что начало эксплуатации или проверки соответствует нулевому моменту времени. Если исследуется система, которая проработала в течение некоторого времени t_0, то используются условные показатели надежности. В этом случае вероятность безотказной работы учитывает, что отказа не было в течение момента времени t_0:

$$P(t_0,t)=P(T\ge t_0+t/T\ge t_0)=\frac{P(t_0+t)}{P(t_0)}. \qquad (5)$$

Аналогично можно представить другие характеристики, описывающие надежность для невосстанавливаемых систем при условии что они уже функционировали в течение времени t_0.

II. Для оценки надежности восстанавливаемых систем, кроме вероятностных характеристик наработки до первого отказа, необходимо знать характеристики функционирования после отказа в процессе восстановления. В результате критерии надежности восстанавливаемых систем учитывают возможность многократных отказов и восстановлений. Основные показатели надежности в этом случае характеризуют:

- распределение наработки до *i*-го отказа: $F_i(t)=P(T_i<t)$;

- распределение времени до *i*-го восстановления: $V_i(t)=P(T_{вi}<t)$;

- вероятность возникновения *n* отказов до достижения длительности наработки *t*: $P_n(t)$.

Кроме того, отказы можно характеризовать средним числом $H(t)$ за время *t*, интенсивностью (параметром) потока отказов

$$\omega(t)=\frac{dH(t)}{dt}$$

и наработкой на отказ

$$T(t)=\frac{t}{H(t)}.$$

Процесс восстановления достаточно полно описывается показателями, подобными показателям, характеризующим отказы: вероятностью восстановления за время *t*; плотностью распределения времени восстановления и средним временем восстановления. Кроме того, используются показатели среднего числа восстановлений за некоторое время и параметр потока восстановлений.

В перечисленных показателях процессы отказов и восстановлений могут рассматриваться как независимые. Объединение характеристик отказов и восстановлений производится в критерии ***коэффициент готовности*** $K_Г(t)$. Этот показатель характеризует вероятность застать в заданный момент времени восстанавливаемую систему в работоспособном состоянии. Значение коэффициента готовности соответствует доле времени полезной работы системы на достаточно большом интервале, содержащем отказы и восстановления.

Для оценки надежности стареющей и разрушающейся аппаратуры в теории надежности вводится ряд показателей, которые не применимы к сложным комплексам программ. К ним относятся показатели надежности, связанные с хранением, транспортированием, ремонтом и старением объектов.

Тема 2. Определение надежного программного обеспечения

§1. Определение надежного программного обеспечения

При анализе надежности программного обеспечения возникает проблема определения: что такое ошибка в программном обеспечении и что такое надежность программного обеспечения? Необходимо договориться о стандартном определении, чтобы избежать таких ситуаций, когда пользователь утверждает, что обнаружил в системе ошибку, а разработчик отвечает: "Нет, система так и была задумана".

Пр. Система раннего обнаружения баллистических снарядов Ballistic Missile Early Warning System должна была наблюдать за объектами, движущимися по направлению к Соединенным Штатам, и, если объект не опознан, начать последовательность защитных мероприятий - от попыток установить с объектом связь до перехвата и, возможно, ответного удара. Одна из ранних версий системы ошибочно приняла Луну за снаряд, летящий над северным полушарием. Ошибка ли это? С точки зрения пользователя (Министерства обороны США) - да. С точки зрения разработчика системы - возможно, и нет. Разработчик может настаивать на том, что в соответствии с требованиями или спецификациями защитные действия должны быть начаты по отношению к любому движущемуся объекту, появившемуся над горизонтом и не опознанному как мирный летательный аппарат.

Дело в том, что разные люди по-разному понимают, что такое ошибка в программном обеспечении. Прежде чем приступать к обсуждению методов устранения таких ошибок, следует дать определение ошибки. Для начала проанализируем некоторые определения.

Определение 1. Согласно одному из определений, *программное обеспечение содержит ошибку, если его поведение не соответствует спецификациям*. Это определение страдает существенным недостатком: неявно предполагается, что спецификации корректны. Такое предположение редко бывает справедливым, так как подготовка спецификаций - один из основных источников ошибок. Если поведение программного продукта не соответствует его спецификациям, ошибка, вероятно, имеется. Однако, если система ведет себя в соответствии со спецификациями, мы не можем утверждать, что она не содержит ошибок.

Определение 2. Второе определение гласит, что *программное обеспечение содержит ошибку, если его поведение не соответствует спецификациям при использовании в установленных при разработке пределах*. Это определение еще хуже предыдущего. Если система случайно используется в непредусмотренной ситуации, ее поведение должно оставаться разумным. Если это не так, она содержит ошибку. Рассмотрим авиационную систему диспетчеризации, которая прослеживает и координирует движение самолетов над некоторым географическим районом. Предположим, что, согласно спецификациям, система должна управлять движением до 200 самолетов одновременно. Но однажды по непредвиденным обстоятельствам в районе появился 201 самолет. Если поведение системы неразумно - скажем, она забывает об одном из самолетов или выходит из строя - система содержит ошибку, хотя она используется вне пределов, установленных при проектировании.

Определение 3. Согласно третьему определению, *ошибка имеется тогда, когда программное обеспечение ведет себя не в соответствии с официальной документацией и поставленными пользователю инструкциями*. Это определение также страдает несколькими изъянами. Возможны ситуации, когда программное обеспечение ведет себя в соответствии с официальными инструкциями, но ошибки все-таки имеются, так как они могут содержаться и в программе, и в инструкциях. Другая проблема возникает из-за тенденции описывать в инструкциях для пользователей только ожидаемую и планируемую работу с системой. Предположим, что в инструкции для пользователей системы говорится: "После ввода новой команды, нажмите один раз клавишу "Ввод". Предположим, что пользователь случайно нажимает клавишу "Ввод" дважды и система программного обеспечения выходит из строя, пото-

му что ее разработчики не предусмотрели такой ситуации. Система, очевидно, содержит ошибку, но мы не можем утверждать, что она ведет себя не в соответствии с инструкциями.

Определение 4. Согласно последнему определению, которое иногда используется, *ошибка определяется, как неспособность системы действовать в соответствии с исходным контрактом или перечнем требований пользователя*. Хотя это определение лучше трех предыдущих, оно также содержит недостатки. Если по требованиям пользователя система должна обеспечивать среднее время между отказами из-за ошибки в программном обеспечении на уровне 100 часов, а для действующей системы этот показатель равен 150 часам, система все же имеет ошибки (поскольку ее среднее время между отказами конечно), несмотря на то, что она превышает требования пользователя. Кроме того, письменно согласованные требования пользователя редко детализированы настолько, чтобы описывать поведение программного обеспечения при всех возможных обстоятельствах.

Определение 5. Приведем определение ошибки в программном обеспечении, решающее перечисленные выше проблемы. В **программном обеспечении имеется ошибка,** *если оно не выполняет того, что пользователю разумно от него ожидать*. **Отказ программного обеспечения** - *это проявление в нем ошибки*. Реакцией пользователя на это определение будет: "Точно!" Разработчик программного обеспечения может возразить: "Определение непрактично. Откуда мне знать, что пользователю разумно ожидать?" Если разработчик программного обеспечения хочет спроектировать хорошую систему, то он всегда должен понимать, что именно ее пользователям "разумно ожидать". Слово "разумно" употреблено в определении для того, чтобы исключить ситуации, когда, например, к терминалу информационно-поисковой системы публичной библиотеки подходит человек и просит определить объем своего вклада в местном банке. Под "пользователем" понимается любой человек, вводящий информацию в систему, исследующий выходные данные или взаимодействующий с системой каким-то иным образом. Большая система программного обеспечения будет иметь много различных пользователей, в том числе, общающихся с системой через удаленные терминалы. Кроме них, пользователями будут программисты и администраторы системы.

Из приведенного определения ошибки можно сделать следующие важные выводы: *ошибки в программном обеспечении являются не только внутренним его свойством, а наличие ошибок есть функция, как самого программного обеспечения, так и ожиданий его пользователей.* Это значит, что, как бы долго мы ни тестировали программу или "доказывали" правильность ее спецификаций, мы никогда не сможем найти в ней все ошибки. Можно обнаружить некоторые ошибки (например, бесконечный цикл), но, по самой природе ошибок в программном обеспечении, никогда нельзя рассчитывать найти все ошибки.

Следующий термин, который следует четко определить, - это надежность программного обеспечения. Проанализируем определение, согласно которому надежность есть вероятность того, что при функционировании системы в течение некоторого периода времени не будет обнаружено ни одной ошибки. Основной недостаток такого определения — это то, что в нем не учтено различие между ошибками разных типов. Рассмотрим авиационную систему диспетчеризации с двумя ошибками в программном обеспечении: из-за одной теряется след самолета на дисплее, а другая состоит в том, что в сообщении оператору неправильно печатается одно слово (например, ТРАНСАММЕРИКАНСКИЙ вместо ТРАНСАМЕРИКАНСКИЙ). По своим последствиям эти ошибки далеко не одинаковы, поэтому надежность должна быть определена как функция не только частоты ошибок, но и их *серьезности.* В соответствии с этим дадим следующее определение.

Надежность программного обеспечения *есть вероятность его работы без отказов в течение определенного периода времени, рассчитанная с учетом стоимости для пользователя каждого отказа*. Таким образом, надежность программного обеспечения является функцией воздействия ошибок на пользователя системы, она не обязательно напрямую связана с внутренней оценкой качества программного обеспечения. Даже крупный просчет в проектировании может оказаться не слишком заметным для пользователя. С другой стороны, тривиальная ошибка может иметь катастрофические последствия. Например, первый запуск

космического корабля на Венеру потерпел неудачу из-за того, что в операторе DO программы на Фортране была пропущена запятая. Надежность не является внутренним свойством программы, она во многом связана с тем, как программа используется.

Тема 3. Проблемы надежности программного обеспечения

§1. Проблемы надежности программного обеспечения

Программы для вычислительных машин можно разделить на три основных типа:

1. К первому относятся программы, разрабатываемые для решения инженерных и научно-исследовательских задач. Они характеризуются неполным использованием ресурсов вычислительных систем и относительно небольшим временем жизненного цикла. Длительность разработки этих программ обычно невелика. Их эксплуатация носит эпизодический и кратковременный характер, отсутствуют жесткие ограничения 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. Второй тип представлен сложными комплексами программ для информационно-справочных систем и систем автоматизации обработки информации, которые функционируют вне реального времени (системы организационного типа). Период их эксплуатации обычно значительно превышает длительность разработки, однако в ходе эксплуатации они могут развиваться и обновляться. Соответственно изменяются характеристики комплексов программ и сопровождающая их документация. Программы этого типа можно классифицировать как системы, и имеется возможность применять к ним понятия теории надежности. Для таких комплексов программ техническими документами могут быть определены функции и характеристики, а также промежуток времени, на котором должны сохраняться заданные показатели в соответствии с определениями теории надежности и требованиями технической документации. Изменение комплексов программ в процессе развития и модернизации системы приводит к тому, что содержание и значения показателей надежности оказываются нестационарными.
3. К третьему типу относятся комплексы программ автоматического или автоматизированного управления, непосредственно входящие в контур управления и функционирующие в реальном масштабе времени. Такие комплексы программ обычно практически полностью используют ресурсы вычислительной машины по памяти и производительности, снабжают подробной документацией и эксплуатируются многие годы и даже десятилетия. Эти комплексы определяют степень автоматизации производства в промышленности и качество управления объектами в народном хозяйстве и военной технике. Комплексы программ этого типа обладают всеми характерными чертами промышленных изделий, к ним в наибольшей степени применимы основные подходы и понятия теории надежности.

Стабильность длительной эксплуатации, наличие требований технической документации и возможность дефектов в комплексах программ, вызывающих сбои и отказы при функционировании, позволяют анализировать показатели надежности программ второго и третьего типов. Реальная надежность программного обеспечения нередко оказывается ниже, чем надежность аппаратурных средств, и определяет надежность функционирования системы в целом.

Надежность аппаратуры в технических системах определяется в основном двумя факторами: надежностью компонент и ошибками в конструкции, допущенными при проектировании или изготовлении. Относительно невысокая надежность компонент, их глубокая взаимозависимость и способность к разрушению, старению или снижению надежности в процессе эксплуатации привели к тому, что этот фактор оказался превалирующим для надежности большинства комплексов аппаратуры. Этому способствовала также относительно невысокая

сложность многих технических систем, вследствие чего, ошибки проектирования проявлялись редко и вуалировались отказами компонент.

Однако есть системы, где эти два фактора соизмеримы. Так, например, надежность первых образцов вновь созданной ЭВМ обычно значительно ниже, чем серийных экземпляров той же ЭВМ после нескольких лет выпуска. Это является не только следствием улучшения технологии производства, но и результатом обнаружения и устранения множества логических и технических ошибок проектирования. Если ошибки проектирования устраняются не только в изделиях, выпускаемых заводом в данное время, но и в изделиях, ранее выпущенных с ошибками и находящихся в эксплуатации, то в результате повышается надежность всех изделий данного типа. После доработки ЭВМ определяющей становится техническая надежность компонент, и постепенно доработки из-за ошибок проектирования прекращаются.

Надежность сложных программных комплексов определяется теми же двумя факторами. Однако степень их влияния иная. Хранение программ на магнитных носителях при отсутствии внешнего вмешательства характеризуется высокой надежностью. Даже при многолетней эксплуатации маловероятны ситуации искажения программ в результате старения носителей. Превалирующим для надежности комплексов программ является второй фактор — ошибки проектирования.

Отмеченные факторы, определяющие надежность систем, различаются по своей природе, проявлению и методам устранения. Отказы компонент аппаратуры, обусловленные старением или разрушением элементов, в большинстве случаев требуют их замены. Восстановительные работы и особенно замена компонент аппаратуры трудно автоматизируются и сохраняют преимущественно ручной характер. Это определяет длительность восстановления, которая измеряется минутами и часами.

Проявление ошибок проектирования обусловлено конкретными ситуациями и сочетанием данных, подлежащих обработке. Возникающий в результате 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 (стирание) массивов данных о внешней обстановке, могут быть устранены программными методами. Поэтому процесс оперативного восстановления при отказах в комплексах программ реализуется преимущественно автоматизированными методами. Проблема восстановления при отказах переходит в проблему преобразования отказов в автоматически устраняемые сбои. При этом не устраняется причина отказа и в такой же ситуации отказ должен повториться, однако повторение данных, приводящих к отказу, маловероятно, и наработка на отказ может быть удовлетворительной, тем более что его проявление сведено к кратковременному сбою. Накопление сведений о сбоях и отказах в программах позволяет их устранить в период профилактических работ.

Понятие ошибок проектирования и, в частности, ошибок в программах так же, как понятия надежности, трудно сформулировать без учета исполнения программ и результатов их функционирования. В технических устройствах во многих случаях имеется эталонное изделие, с которым сравниваются аналогичные, вновь изготовленные. Такое сравнение позволяет в статике выявить ошибки тиражирования и изготовления или неисправные и разрушившиеся компоненты; при этом ошибки проектирования сохраняются соответствующими эталонному изделию. Тиражирование программ может производиться очень точно, а их физическое разрушение маловероятно и легко устранимо. Однако для выявления ошибок проектирования программ невозможно создать абсолютный эталон, сравнивая с которым каждую программу в статике, можно было бы обнаружить отличие и классифицировать его как ошибку. В процессе отладки и испытаний комплекса программ устраняются многие ошибки, и программы асимптотически приближаются к идеальным - безошибочным. Однако степень приближения остается неизвестной, и копии программ содержат ошибки эталонной программы. При анализе показателей надежности любых систем в первую очередь исследуют эти показатели с позиции пользователя - внешнего абонента системы. Для пользователя важ-

ны характеристики отказов и восстановлений, механизм их возникновения и устранения имеет второстепенное значение. Анализ причин отказов важен для создателя системы или эксплуатационника для профилактики аналогичных отказов и скорейшего их устранения.

Программа любой сложности и назначения при фиксированных исходных данных и абсолютно надежной аппаратуре исполняется по заданному маршруту и дает на выходе определенный результат. Многократное исполнение программы при заданных условиях даст неизменный результат. Однако комбинаторный характер исходных данных и накопленной при обработке информации, а также множество условных переходов создают огромное число различных маршрутов исполнения для каждого комплекса программ, которые обычно на несколько порядков больше числа команд в программе. Такое число вариантов исполнения программы не может быть проверено полностью из-за ограничений на длительность отладки и объем приемочных испытаний. Некоторые маршруты обработки информации характеризуются малой вероятностью исполнения. Опыт отладки и эксплуатации сложных комплексов программ показывает, что при отладке невозможно проверить все варианты обработки информации, и даже после нескольких лет эксплуатации встречаются непроверенные сочетания исходных данных, при которых работающий комплекс программ дает неверные результаты. Подобные искажения результатов с точки зрения пользователя рассматриваются как сбои или отказы. Возможна тенденциозная проверка комплекса программ в предельных и критических условиях, которые могут искусственно создаваться заказчиком при испытаниях и приемке системы. Такие условия могут давать существенно искаженные и ухудшенные показатели надежности функционирования программного обеспечения по сравнению со средними типовыми условиями эксплуатации. Подобная проверка по существу является расширением требований технического задания в период испытаний вследствие повышения запросов заказчика. В то же время такие требования являются изменением и расширением заказа и связаны с увеличением объема работ. Поэтому важно согласовывать с заказчиком методику и условия испытаний комплексов программ на надежность с учетом испытаний их в критических режимах. Следует отметить глубокую взаимосвязь надежности программ и надежности вычислительных систем (ВС), на которых они исполняются. Надежность аппаратуры ВС не может быть исследована и измерена без длительного исполнения некоторого комплекса программ. Замена комплекса программ, с помощью которого проверяется надежность аппаратуры ВС и условий испытаний, приводит к изменению характеристик надежности вследствие различий в широте и глубине проверки аппаратуры. Кроме того, проверяющие программы содержат не выявленные ошибки и различаются помехоустойчивостью к сбоям и к частичным отказам аппаратуры. Разделить причины отказов и искажении выходных результатов на обусловленные только аппаратурой или программами в ряде случаев оказывается очень сложно. Поэтому для оценки показателей надежности ВС между разработчиками аппаратуры и программ и заказчиком, кроме правил проверки на надежность, согласовываются состав и характеристики тестовых программ, по которым производится проверка. Особенно сложна эта задача в начале эксплуатации комплекса программ на вновь разработанной ВС.

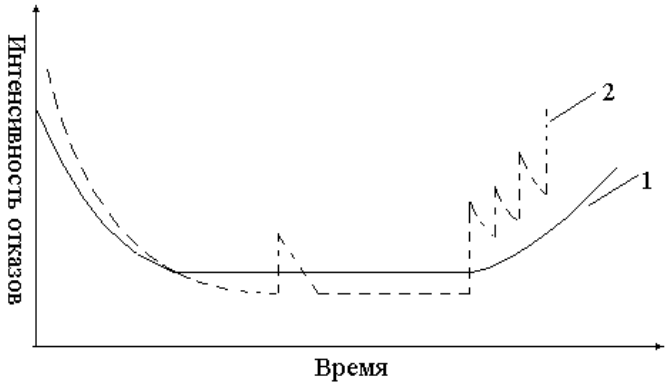


Рис 1.1. Зависимость интенсивности отказов от времени при разработке и эксплуатации для аппаратных (1) и программных комплексов (2)

Частота проявления ошибок в комплексах программ в зависимости от времени подобна частоте отказов аппаратуры (рис. 1.1). Отказы из-за ошибок в программах вначале уменьшаются вследствие их обнаружения и устранения в процессе отладки. Этот период по своим характеристикам (но не по физической сущности) похож на период "приработки" в аппаратуре. Далее следует период эксплуатации, который характеризуется постоянной интенсивностью отказов из-за программных ошибок, если они не корректируются. Модификация приводит к появлению вторичных ошибок, которые также необходимо устранять, и после нескольких доработок комплекс программ морально устаревает и подлежит замене, что отражено на рис. 1.1 в виде возрастания интенсивности искажений в конце интервала жизни.

Высокая стоимость проектирования и, в частности, отладки сложных программных комплексов не позволяют создавать программы с гарантированным отсутствием ошибок и абсолютно устойчивых к любым возмущениям. Для каждого типа систем должны устанавливаться свои рациональные и необходимые значения показателей надежности программного обеспечения, соизмеримые с показателями надежности аппаратуры. Затраты на обеспечение надежности должны сопоставляться с возможным ущербом вследствие отказа системы с данным комплексом программ.

Тема 4. Основные понятия теории надежности комплексов программ. Типы отказов программного обеспечения

§1. Основные понятия теории надежности комплексов программ

Применение основных понятий теории надежности к проектированию и оценке качества сложных комплексов программ позволяет адаптировать и развивать эту теорию в направлении анализа надежности программного обеспечения. В результате к настоящему времени сформировался предмет исследований и соответствующее научно-техническое направление - надежность сложных комплексов программ, функционирующих в реальном масштабе времени, или надежность программного обеспечения. Требования разработки крупных программных комплексов выдвинули проблему надежности их функционирования в разряд самых актуальных. К задачам анализа надежности программного обеспечения можно отнести следующие:

- формулирование основных понятий, используемых при исследовании параметров и показателей надежности программ;
- выявление и исследование основных факторов, определяющих характеристики надежности сложных программных комплексов;

• выбор и обоснование критериев надежности комплексов программ различного типа и назначения;
• исследование характеристик искажений исходных данных от различных типов источников и их влияния на надежность функционирования комплексов программ;
• исследование ошибок в программах, динамики их изменения при отладке и модернизации и влияния на надежность;
• разработка и исследование методов структурного синтеза сложных комплексов программ, повышающих их надежность;
• исследование методов и средств контроля и защиты от искажений вычислительного процесса и данных в памяти путем ввода различных видов избыточности и помехозащищенности, обеспечивающих автоматизацию и сокращение времени восстановления;
• разработка методов прогнозирования характеристик надежности комплексов программ с учетом их сложности, структурного построения и технологии проектирования;
• разработка методов создания комплексов программ с заданной надежностью функционирования и средств защиты от сбоев и отказов с учетом важности решаемых системой задач.

Результаты решения этих задач являются основой создания современной высокоэффективной технологии проектирования программного обеспечения с заданными показателями надежности. Для их решения используются результаты экспериментального исследования факторов и характеристик, разработанных и разрабатываемых комплексов программ. Имеются попытки создания аналитических моделей и методов определения надежности программного обеспечения. Использование и объединение результатов экспериментальных и теоретических исследований надежности сложных комплексов программ позволило заложить основы теории и методов проектирования надежного программного обеспечения. Уточним фундаментальные понятия теории надежности (сбой, отказ, восстановление, надежность и т.д.) при их использовании для анализа характеристик функционирования комплексов программ.

Понятие *отказа в аппаратуре* связано с нарушением работоспособности изделия и его соответствия требованиям технической документации. Потеря работоспособности подразумевает наличие физического разрушения элементов или компонент изделия, которое требует их ремонта или замены. Для устранения отказа, как правило, необходимо вмешательство специалистов ремонтной бригады, которые ремонтируют отказавшие компоненты либо заменяют их исправными. Возможно восстановление, когда используются резервные компоненты, которые автоматически подключаются вместо отказавших. Однако и в этом случае замененные компоненты содержат физические нарушения, не позволяющие использовать их без ремонта.

Рассмотрим специфику нарушения работоспособности программ в предположении абсолютной безотказности аппаратуры ВС. Отказ при исполнении комплекса программ может проявиться вследствие:
• нарушения кодов записи команд в памяти программ;
• стирания или искажения данных в оперативной или долговременной памяти ВС;
• нарушения нормального хода вычислительного процесса.

Перечисленные искажения могут действовать совместно. Отказ может проявляться в виде программной остановки или зацикливания, систематического пропуска исполнения некоторой группы команд, однократного систематического искажения данных и т. д. Программные отказы приводят к прекращению выдачи абонентам информации и управляющих воздействий или к значительному искажению ее содержания и темпа выдачи, соответствующих нарушению работоспособности комплекса программ.

Основной причиной нарушения работоспособности программ при безотказности аппаратуры является конфликт между исходными данными, подлежащими обработке, и харак-

теристиками программы, осуществляющей эту обработку. При этом под исходными данными понимаются как вновь поступившие сообщения, так и вся информация, накопленная за время предыдущего функционирования программ. Исходные данные могут находиться в области, определяемой техническим заданием, однако вне области, проверенной при тестировании и испытаниях программы на надежность (области II и IV на рис. 1.2).

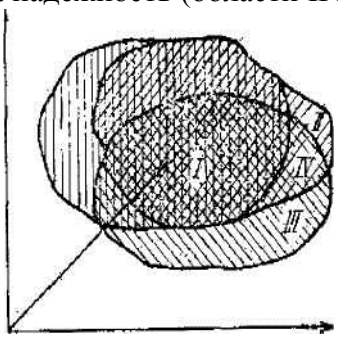


Рис. 1.2. Пространство исходных данных:
область предполагаемых данных - штриховка "///",
область тестовых данных - штриховка "|||",
область реализуемых данных — штриховка "\\\"

Работоспособность комплекса программ можно гарантировать при исходных данных, которые использовались при тестировании и испытаниях. При попадании данных в области II или IV возможно нарушение работоспособности из-за не выявленных в программе ошибок, которые проявляются только при таких исходных данных. Кроме того, реальные исходные данные могут иметь значения, отличающиеся от определяемых техническим заданием и от проверенных при испытаниях программ (область III). При таких исходных данных функционирование программ трудно предсказать заранее, и вероятны аномалии, завершающиеся отказами.

Понятие "сбой" в теории надежности трактуется как самоустраняющийся отказ, не требующий внешнего вмешательства для замены отказавших компонент. Таким образом, понятия сбой и отказ применительно к аппаратуре отличаются степенью физического разрушения компонент и необходимостью их замены. Отсюда следует связь этого события с временем восстановления: сбой устраняется "мгновенно", а отказ устраняется "долго". Однако отсутствует четкое определение временного порога, позволяющего разделять сбои и отказы и связывать эти понятия с функциональными назначениями и параметрами анализируемой системы. Превалирующей остается классификация, базирующаяся на учете степени физического разрушения и различных методах устранения нарушений работоспособности аппаратуры.

При конфликтах исходных данных и характеристик программы отсутствует физическое разрушение аппаратуры ВС и не требуется замены или ремонта материальных компонент. Восстановление после программного отказа в принципе всегда может быть осуществлено программными средствами без вмешательства человека. Это приводит к изменению метода разделения событий отказа и сбоя. Основным принципом классификации сбоев и отказов становится показатель длительности восстановления после искажения программ, данных или вычислительного процесса. При этом необходимо рассматривать всю временную шкалу и устанавливать пороговое значение времени разделения $t_Д$. При длительности восстановления, меньшей порога $t_Д$, аномалии при функционировании программ относят к сбоям, а при восстановлении, превышающем по длительности пороговое значение $t_Д$, искажения соответствуют отказу. При этом рассматриваются искажения, которые нарушают работоспособность комплекса программ. Классификация программных сбоев и отказов по длительности восстановления приводит к необходимости анализа динамических характеристик абонентов-потребителей данных и временных характеристик функционирова-

ния программ. При взаимодействии комплекса программ с абонентом существенны следующие динамические параметры системы:

- инерционность объекта, являющегося источником или потребителем информации;
- среднее время и частота (темп) решения задачи по обработке информации для данного абонента;
- необходимая длительность отклика или время реакции ВС от момента поступления исходных данных до момента выдачи обработанных результатов;
- средний интервал времени между однотипными сообщениями, поступающими на обработку комплексу программ.

Перечисленные параметры коррелированны и могут служить основой для установления порогового значения $t_Д$ при классификации сбоев и отказов. Инерционность объекта управления или потребителя информации, обрабатываемой данным комплексом программ, является первичным показателем, который используется для определения темпа выполнения программ и времени реакции комплекса программ, т.е. динамические характеристики определяют темп выработки управляющих воздействий и поступления обработанных данных. Снижение темпа решения задач ведет к ухудшению характеристик функционирования объекта и при некотором темпе его работоспособность нарушается, что эквивалентно отказу.

При нормальном темпе решения задачи управления t^1 (рис. 1.3) отклонения объекта от траектории, рассчитываемой комплексом программ, находятся в допустимых пределах. Для любого объекта существует допустимое время отсутствия управляющих воздействий $t_{Д1}$, при котором достигается предельное искажение δ_1 выходных данных. Для менее инерционного объекта, допустимое искажение δ_1 — δ_2 достигается за меньшее время $t_{Д2} < t_{Д1;}$ т.е. временная зона перерыва выдачи информации и нарушения работоспособности (зона сбоя) тем шире, чем инерционнее объект. Таким образом, установив пороговое значение δ ошибок обработанных данных, можно определить интервал времени $t_Д$ состояния комплекса программ, который разделяет события сбоя и отказа. В некоторых типах систем такое пороговое значение времени определяется функциональными характеристиками решаемых задач, однако определяющими являются динамические характеристики источников и потребителей информации. Для иллюстрации взаимосвязей между перечисленными параметрами в табл. 1.1 приведены временные характеристики основных типов систем управления и обработки информации по времени решения задач, необходимому времени отклика и среднему интервалу времени между однотипными сообщениями. Там же дана оценка порогового времени восстановления работоспособного состояния, при превышении которого следует фиксировать отказ системы. Это время соответствует времени решения задач и пропорционально необходимому времени отклика при обработке сообщений от абонента.

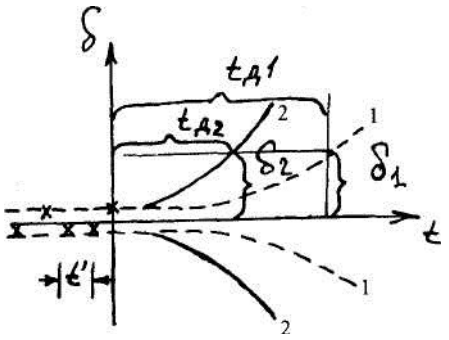


Рис. 1.3. Траектории возможного движения объектов с большой (1) и малой (2) инерционностью

В ряде работ отмечается отсутствие тождественности понятий правильная и надежная программы. Понятие правильной программы рассматривается статически, вне временного функционирования. **Правильная программа** должна обеспечивать выходные данные, соответствующие эталонным, в области изменения исходных данных, заданных требованиями технического задания или спецификации. Степень правильности программы можно характеризовать вероятностью попадания в область исходных данных, которая предусматривалась требованиями спецификации, однако не была проверена при тестировании и испытаниях (II и IV на рис. 1.2). В этой области исходных данных не гарантируется получение эталонных результатов из-за возможных ошибок в программе, не обнаруженных при отладке. Таким образом, правильность программы определяете, совмещением событий: попаданием исходных данных в область, заданную требованиями спецификаций, но не проверенную при тестировании и испытаниях (вероятности P_{II} и P_{IV}), и наличием ошибки в программе при обработке таких данных (вероятность P_o). Правильность программы не определена вне области данных, заданной спецификациями, и не зависит от динамики функционирования в реальном времени.

Надежная программа должна обеспечивать низкую вероятность отказа в процессе функционирования. Быстрая реакция на искажения программ, данных или вычислительного процесса и восстановление работоспособности за время, меньшее порогового, позволяют обеспечить, высокую надежность программ. При этом неправильная программа может функционировать абсолютно надежно. Действительно, если при появлении реальных исходных данных, попадающих в области II и IV (см. рис. 1.2) и стимулирующих неправильные результаты, они не приводят к отказам, то такая программа функционирует надежно, хотя и не всегда правильно.

Правильная программа определена в области исходных данных, заданных требованиями технического задания. В реальных условиях исходные данные могут попадать в область III (см. рис. 1.2), не проверенную при отладке и не соответствующую требованиям спецификации. В результате, формально правильная программа окажется ненадежной из-за ошибки при задании области изменения исходных данных. Если восстановление происходит за время, меньше порогового между отказом и сбоем, то такие события не влияют на основной показатель надежности – наработку на отказ. Следовательно, отказ при функционировании программы является понятием динамическим и произойдет при совмещении следующих событий:

- появление на входе программы данных, попадающих в непроверенные при тестировании и испытаниях области III или IV (вероятности P_{III} и P_{IV});
- обработка этих данных компонентами программы, содержащими ошибку, достаточную для появления ситуации отказа (вероятность P_0);
- длительность восстановления после возникновения ситуации отказа, превышает пороговое значение (вероятность $P_в$).

Правильная и надежная программы различаются областями изменения исходных данных, которые определяют степень неправильности (область II) или степень не надежности (область III). В некоторой области изменения исходных данных (область IV) правильность и надежность программы коррелированны. Это соответствует данным, определенным техническим заданием, но не проверенным при тестировании. Однако и при таких исходных данных, если восстановление после сбоя производится оперативно, программа может иметь высокие показатели надежности.

Отсутствие физического разрушения компонент функционирующего комплекса программ позволяет подойти к процессу восстановления иначе, чем для аппаратуры. Так как нет необходимости ремонта и замены компонент с участием человека, можно добиться высокой автоматизации восстановления программы. Главной задачей становится восстановление за время, не превышающее порогового значения между сбоем и отказом. При автоматизации процесса и сокращении времени восстановления отказы можно преобразовать в сбои и тем самым улучшить показатели надежности функционирования системы.

Для решения этой задачи в комплексе программ должны быть средства, позволяющие:

- проводить систематический контроль и обнаруживать аномалии состояния программ, данных и процесса функционирования;
- диагностировать обнаруженные искажения;
- вырабатывать решения и выбирать методы и средства оперативного восстановления;
- реализовывать оперативное восстановление нормальной работоспособности;
- регистрировать каждый происшедший сбой или отказ, обобщать данные искажений для выявления случаев, требующих доработки программ или аппаратуры.

Реализация таких средств осуществляется за счет введения избыточности в программы, данные и процесс функционирования комплекса программ. Избыточность можно разделить на:

- программную, включающую все программные компоненты, предназначенные для контроля, обнаружения диагностики и восстановления работоспособности комплекса программ;
- информационную, заключающуюся в дублированном хранении данных на различных средствах защиты информации;
- временную, состоящую в выделении необходимых резервов времени ВС на исполнение программ, обеспечивающих оперативный контроль и восстановление.

Перечисленные виды избыточности используются совместно и требуют дополнительных ресурсов ВС по объему оперативной памяти, памяти команд и производительности. При возрастании ресурсов на 5-10% надежность функционирования (наработка на отказ) комплекса программ возрастает на один-два порядка.

§2. Типы отказов программного обеспечения

Отказы и сбои по степени их влияния на функционирование комплекса программ и на всю систему управления в целом делятся на три крупные группы:

- искажения вычислительного процесса и данных, вызывающие полное прекращение выполнения функций системой управления на длительное или неопределенное время, - отказ, в значительной степени обесценивающий результаты предыдущего функционирования;
- искажения, кратковременно прерывающие функционирование системы и мало искажающие накопленные данные, и выдаваемые результаты, - частичный отказ или длительный сбой, в некоторой степени обесценивающий предыдущие результаты;
- искажения, кратковременно и мало отражающиеся на вычислительном процессе и обрабатываемых данных, - сбои, практически не обесценивающие результаты функционирования комплекса программ.

В зависимости от глубины контроля и длительности запаздывания в обнаружении отказа, а также в зависимости от качества средств, осуществляющих восстановление, одни и те же ситуации искажений вычислительного процесса или данных могут быть отнесены к разным типам отказов или сбоев. Наиболее типичными полными отказами при функционировании сложных комплексов программ являются:

- зацикливание, т. е. последовательно повторяющееся исполнение группы команд, которое не прекращается без внешнего вмешательства, блокируя функционирование всех остальных программ данного комплекса;
- остановка ЭВМ и полное прекращение решения функциональных задач, при этом может сохраниться возможность приема и выдачи информации и выполнения некоторых функций, стимулируемых прерываниями;
- значительное искажение или полная потеря накопленных данных о состоянии внешних абонентов и процессе их функционирования;

• прекращение или недопустимое снижение темпа решения некоторых задач, пропуск или потеря необработанных групп сообщений вследствие перегрузки ЭВМ или ВС по пропускной способности.

Эти отказы существенно влияют на выполнение основных функций комплексом программ. В несколько меньшей степени на вычислительный процесс и обрабатываемые данные влияют искажения, приводящие к следующим типам частичных отказов или длительных сбоев:

• искажение заданной последовательности вызова подпрограмм, приводящее к пропуску исполнения отдельных подпрограмм или их частей, что в свою очередь, может привести к неправильному или неполному решению некоторых задач и к искажению выходных результатов;

• использование и обработка искаженных исходных данных, отражающиеся на логике решения задач и приводящие к искажению отдельных накопленных и выдаваемых данных.

Эти типы искажений заключаются в кратковременном нарушении нормального вычислительного процесса либо в искажении исходных, промежуточных или результирующих данных, В зависимости от повторяемости и глубины распространения искажения квалифицируются как частичные отказы либо как сбои с длительными последствиями.

В наименьшей степени на надежность функционирования влияет третий тип сбоев, которые незначительно искажают общие результаты. Не нарушая практически логики функционирования комплекса программ, такие сбои искажают преимущественно отдельные обрабатываемые и результирующие данные. При поступлении аналогичных данных в последующие моменты времени можно ликвидировать последствия сбоев, не проводя специальных восстановительных работ.

Тема 5. Основные факторы, влияющие на надежность программного обеспечения

§1. Основные факторы, влияющие на надежность программного обеспечения

Одни и те же типы сбоев и отказов при исполнении комплексов программ могут быть вызваны различными факторами, которые можно разделить на три группы (табл. 1.2).

В **первую группу** входят факторы, непосредственно вызывающие сбой или отказ при исполнении программы, причинами которых могут быть:

• искажения исходной информации, поступающей от внешних абонентов;
• самоустраняющиеся отказы или сбои в аппаратуре вычислительной системы;
• не выявленные ошибки в комплексе программ.

Ко **второй группе** факторов относятся архитектура комплекса программ и структурное построение компонент. Структура программ определяет возможность расширения последствий искажений информации или вычислительного процесса, влияет на вероятность превращения искажения в отказ и на время восстановления после отказа.

Третья группа факторов влияет на длительность восстановления и глубину последствий от возникающих отказов. В эту группу входят факторы, определяющие качество контроля вычислительного процесса и обрабатываемых данных, запаздывание в обнаружении искажений, качество классификации искажений и длительность проявления их последствий. Они определяют, прежде всего, длительность восстановления, время наработки на отказ и способствуют быстрой локализации искажений. Рассмотрим основные особенности факторов, приводящих к сбоям или отказам (табл. 1.2).

Искажения исходной информации в большинстве случаев непосредственно не влияют на надёжность исполнения программ. Искаженные данные могут обрабатываться и являться причиной ошибок в результатах, выдаваемых внешним абонентам или накапливаемых в памяти ВС. Однако некоторые искажения выводят за область допустимых значений переменных (см. рис. 1.2). При этом возрастает вероятность того, что искаженная величина будет обрабатываться некоторым сочетанием команд, приводящим к отказу либо к сбою функционирования.

Причинами искажений данных, поступающих от внешних абонентов, могут быть:

- искажения данных на первичных носителях информации;
- шумы и сбои в каналах связи при передаче сообщений по линиям связи;
- сбои и частичные отказы в аппаратуре передачи или приема информации;
- потери или искажения сообщений в ограниченных буферных накопителях вычислительной системы;
- ошибки в документах, используемых для подготовки данных, вводимых в вычислительную систему.

Таблица 1.2

Факторы, влияющие на надежность функционирования комплексов программ

Факторы, определяющие надежность	Методы проектирования надежных комплексов программ	Методы повышения надежности программ
Особенности объектов управления и абонентов ВС: - требования к показателям надежности; - инерционность объектов; - необходимое время реакции ВС; - средний темп обмена сообщениями. Характеристики искажений исходных данных: - искажения, обусловленные взаимодействием человека с ВС; - искажения информации в системах передачи данных; - искажения данных при накоплении и хранении в ВС. Характеристики ошибок в программах и их проявления Статистические характеристики ошибок и искажений: - программ; - массивов данных; - вычислительного процесса. Характеристики классов ошибок: - программных; - алгоритмических; - системных.	Структурное проектирование программ и данных: - структурное проектирование программных модулей; - структурное проектирование комплексов программ и взаимодействия модулей; - структурирование массивов данных; - контроль выполнения правил структурирования. Тестирование программ: - детерминированное; - статическое; - контроль пропускной способности в реальном времени. Испытания и определение надежности программ: -экспериментальное определение надежности в нормальных условиях функционирования; - форсированные испытания на надежность программ; - расчетно - экспериментальные методы определения надежности программ.	Контроль состояния и функционирования программ. Использование избыточности: - программной; - информационной; -временной. Контроль программ, данных и решение задач при профилактических работах: - предстартовый контроль; оперативный контроль. Программное восстановление: - восстановление текстов программ; - восстановление искаженных данных; - корректировка вычислительного процесса.

Для представления исходных данных различного вида используются масштабы величин и разрядность, определяемые природой и физической сущностью каждой переменной. Рассмотрим особенности основных типов данных, поступающих в вычислительную систему, и специфику их описания. Значительная часть данных представляет собой квантованные результаты измерения непрерывных физических характеристик. Такие величины связаны условиями гладкости, т. е. условиями малых изменений производных этих переменных по времени или по другим параметрам. Разрядность таких переменных определяется точностью их измерения и в большинстве случаев находится в диапазоне 2 – 4 байтов. Область определения каждой переменной является замкнутой и односвязной. Например, при измерении координат объектов с погрешностью 0,01% координата кодируется 14 – 15 двоичными разрядами. Для исследователя представляет интерес искажение полной величины измеренной координаты. Поэтому целесообразно оценивать вероятность искажения слова соответствующего размера.

В сообщениях от управляемых объектов содержится также и кодовая информация, характеризующая состояние отдельных компонент объекта. Каждая кодовая часть сообщения соответствует определенному устройству или некоторому набору состояний. Такая логически связанная часть может содержать сжатые данные в пределах 1 бита (да - нет, включено - выключено) или более сложные сведения, кодируемые несколькими битами или даже байтами. Статистический анализ показывает, что подавляющее большинство логических и кодовых переменных кодируется в пределах 4 бит – 1 байта.

Для некоторых типов систем значительная часть информации является символьной, описывающей логически связанные тексты, либо содержащей десятичные цифры. Стандартное кодирование таких данных производится в пределах 1 байта на символ. Таким образом, исходные данные преимущественно имеют объем, кратный целому числу байтов. Во многих вычислительных системах интерфейс с внешними абонентами, как и процесс обработки информации, построен на байтовой, структурной основе. Поэтому для оценки показателей достоверности исходной информации для комплексов программ в качестве наиболее целесообразной единицы следует выбрать вероятность искажения 1 байт.

В подготовке и вводе исходных данных в ВС участвует человек. Это приводит к тому, что соответствующая часть данных характеризуется невысокой достоверностью с вероятностью ошибки около 10^{-4} на 1 байт. В автоматических устройствах подготовки и передачи информации вероятность ошибки может быть значительно ниже и достигать значения 10^{-6} – 10^{-7}. Однако и при такой достоверности данные не всегда пригодны для обработки без проведения контроля и предварительной селекции и могут оставаться существенной причиной отказов или сбоев при их обработке. Повышение достоверности исходной информации может производиться за счет использования избыточности при подготовке первичных данных и при вводе их в ВС. Эта избыточность используется для обнаружения искажений и исключения ложных данных, а в отдельных случаях и для исправления ошибок.

Самоустраняющиеся отказы и сбои в аппаратуре вычислительных систем являются фактором, существенно влияющим на надежность функционирования комплексов программ. За последние годы значительно возросла надежность вычислительных систем. Существенно снизилась вероятность полного отказа аппаратуры ВС. Существуют системы, характеризующиеся средним временем наработки на отказ, исчисляемым десятками тысяч часов, однако для однопроцессорных ЭВМ наработка на устойчивый отказ, как правило, измеряется сотнями часов.

Значительно чаще происходят сбои или трудно обнаруживаемые кратковременные отказы. Большинство из них выявляется и устраняется средствами аппаратурного контроля и не влияет на исполнение программ. Однако некоторая часть аппаратурных сбоев может приводить к искажениям исполнения программ или к искажениям переменных. Причинами таких сбоев и отказов являются преимущественно внешние воздействия на аппаратуру ВС, влияющие на нарушение контактов и пропадание сигналов, или индустриальные электриче-

ские помехи. Это приводит к тому, что обнаруживаемые тестами сбои и самоустраняющиеся отказы происходят на один - два порядка чаще, чем устойчивые отказы. Ещё чаще происходят сбои, которые не удается обнаружить и зафиксировать при функционировании комплекса программ в процессе нормальной обработки информации и управления. Такие сбои проявляются в случайные моменты времени, и практически невозможно добиться их повторяемости. Трудность их регистрации и изучения, а также не заинтересованность фирм, производящих ЭВМ, в выявлении характеристик сбоев приводят к тому, что достоверные данные о них практически отсутствуют. Тем не менее, искажения переменных и процесса исполнения программ из-за сбоев аппаратуры иногда приводят к зацикливанию, останову или искажению массивов данных.

Если среднее время наработки на устойчивый отказ в однопроцессорной ЭВМ составляет 100 ч, то интервалы времени между обнаруживаемыми сбоями и самоустраняющимися отказами составляют около 1 ч. Еще чаще происходят сбои, которые невозможно зарегистрировать, например искажения младших разрядов переменных, являющихся результатом измерения гладких физических величин. При среднем быстродействии ЭВМ 100 тыс. операций в секунду это соответствует выполнению произвольной операции с вероятностью искажения около $10^{-7} – 10^{-8}$.

Имеющийся опыт создания надежных комплексов программ приводит к тому, что защита от программных ошибок одновременно обеспечивает защиту от сбоев и самоустраняющихся отказов аппаратуры ЭВМ. Не выявленные ошибки в комплексах программ являются основной причиной ненадежности функционирования. В процессе отладки основная часть ошибок в программах обнаруживается, и устранятся, однако всегда есть риск пропуска нескольких ошибок. Любая отладка «может показать наличие ошибок, но не может доказать их отсутствие». В процессе тестирования и отладки сложных комплексов программ практически невозможно выполнение абсолютно полных проверок, гарантирующих отсутствие непроверенных компонент программы и полное выявление всех возможных ошибок. В результате в сложных программах всегда существует некоторое количество не выявленных ошибок. Разнообразие реальных сочетаний значений переменных в исходных данных, обрабатываемых программой, создает ситуации, при которых последствия отдельных ошибок в программах проявляются аналогично последствиям сбоев в аппаратуре ЭВМ. Если удается зафиксировать условия, при которых обнаружены отказы или искажения результатов, то во многих случаях можно выявить и ошибку в программе.

При объеме комплекса программ 100 тыс. команд даже 10 не выявленных ошибок могут приводить к заметным искажениям результатов и отказам. Действительно, при вероятности ошибок в программе 10^{-4}, при равновероятном выполнении всех команд в комплексе программ и при быстродействии ЭВМ 100 тыс. операций в секунду неправильные команды исполнялись бы в среднем 10 раз в секунду, т. е. 10 раз в секунду обнаруживался бы отказ, сбои или искажение результатов. Однако распределение вероятностей исполнения маршрутов в программах отличаются от равномерного. Основные наиболее вероятные маршруты обработки информации в программе проверяются и отлаживаются наиболее тщательно. Поэтому команды с ошибками исполняются преимущественно при прохождении по маловероятным маршрутам, по крайней мере, на один - два порядка реже, чем в приведенных выше оценках. Кроме того, не каждая ошибка в программе вызывает сбои или отказ при исполнении программы. Количество не выявленных ошибок в начале регулярной эксплуатации комплекса программ может быть существенно больше 10. В результате вероятность искажения из-за ошибок в сложных комплексах программ, по-видимому, находится на уровне 10^{-5} - 10^{-7} на одну исполненную команду.

Высокое быстродействие современных ВС, сложность и большой объем функционирующих комплексов программ, а также огромное разнообразие исходной информации приводят к тому, что даже малое количество не установленных в программе ошибок может значительно снижать надежность программного обеспечения ВС. Поэтому разработка профилактических мероприятий, направленных на предотвращение ошибок в программах, методов

обнаружения и исправления ошибок, а также методов и средств автоматической защиты от последствий проявления ошибок и методов быстрого восстановления является основной проблемой обеспечения высокой надежности сложных комплексов программ.

На надежность функционирования программного обеспечения влияет структура и технология разработки комплексов программ. В зависимости от структурного построения комплекса программ последствия ошибки могут быть локализованы в некотором небольшом участке программ и данных либо распространиться на значительно большее расстояние от места расположения ошибки. Строгое иерархическое построение крупных комплексов программ на базе единообразно оформленных законченных программных модулей обеспечивает снижение вероятности ошибки в каждой команде программы и снижает возможность распространения последствий ошибок за пределы программного модуля и используемых в нем массивов данных.

В программах существуют конструкции потенциально ненадежные и способные приводить к отказам при небольших искажениях вычислительного процесса или массивов данных. Например, циклы, выход из которых организован с помощью строгого сравнения переменной цикла с некоторым параметром, могут стать "бесконечными" при изменении этого параметра или шага внутри тела цикла. Значительно более устойчивым является цикл, имеющий организацию выхода при достижении числа прохождений циклом заданного значения. Не отличаются устойчивостью к искажениям вычислений программные конструкции, содержащие безусловный переход по содержимому ячейки (оператор GO TO). Любое искажение переменной в этой ячейке может явиться причиной отказа вследствие переадресации в произвольные места программы или в массивы данных. На потенциальную надежность функционирования сложных комплексов программ влияет также структурное распределение оперативной памяти для локальных и глобальных переменных. Для обеспечения устойчивости при произвольном вызове подпрограмм и снижения вероятности разрушения данных, целесообразно выделять зоны локальной памяти для каждой подпрограммы либо для подпрограммы одного иерархического уровня.

Таким образом, при формализации правил структурного построения подпрограмм и всего комплекса, выборе языка программирования и основных допустимых конструкций в программах следует учитывать кроме эффективности программирования потенциальную склонность рекомендуемых конструкций к локализации и снижению вредных последствий любой ошибки или аппаратного сбоя. Строгое модульно-иерархическое построение комплексов программ и максимально возможная автономность зон оперативной памяти значительно повышают надежность функционирования таких комплексов.

Контроль функционирования и обеспечение устойчивости к ошибкам комплексов программ. Надежность восстанавливаемых объектов определяется длительностью наработки на отказ и длительностью восстановления. В программах отсутствуют компоненты, которые разрушаются физически и требуют для восстановления замены и ремонта. Поэтому время восстановления не зависит от объема технологических работ, ремонтных бригад, от их занятости, от наличия инструмента, комплектующих изделий и других факторов. При искажении вычислительного процесса или данных задача состоит в максимально быстром обнаружении искажения, в точной классификации возможных последствий искажения, а также в проведении мероприятий, обеспечивающих быстрое восстановление нормального функционирования. Средства контроля и обеспечение устойчивости к ошибкам программ и данных должны обеспечивать оперативное обнаружение и локализацию искажения, предотвращая расширение их последствий до уровня длительных отказов в функционировании комплексов программ.

В системах реального времени допустимы некоторые перерывы в обработке данных и выдаче управляющих воздействий. Кратковременные перерывы в допустимых пределах классифицируются как случайные сбои и практически не влияют на эффективность всей системы. Таким образом, так же как при анализе надежности аппаратуры, разделение сбоя и отказа базируется на длительности восстановления после их проявления. Однако при

программном восстановлении это разделение можно провести более четко, основываясь на времени допустимого прерывания функционирования программ и отсутствия выдачи информации и управляющих воздействий внешним абонентом.

Введение средств контроля и обеспечения устойчивости к ошибкам в программах позволяет скомпенсировать их неполную отладку, а также снизить влияние возмущений других типов. Однако только средствами контроля и обеспечения программной устойчивости к ошибкам невозможно достигнуть высокой надежности функционирования комплексов программ. Возникает оптимизационная задача распределения ресурсов на отладку и обеспечение устойчивости к ошибкам, гарантирующих заданную надежность функционирования комплексов программ при минимальных суммарных затратах. Таким образом, так же как в аппаратурных комплексах, заданная надежность может быть достигнута повышением надежности компонент (отладкой подпрограмм), введением избыточности для контроля и резервирования (контроль и обеспечение устойчивости к ошибкам комплекса программ) либо совместным сбалансированным применением этих методов повышения надежности.

Следует подчеркнуть существенное различие методов и средств контроля состояния и обеспечение устойчивости к ошибкам процесса исполнения программ и процесса записи, хранения и использования массивов данных. Исполнение программ - процесс динамический, и его искажение необходимо выявить оперативно с минимальным запаздыванием. Столь же оперативно желательно восстанавливать вычислительный процесс. Поэтому средства контроля должны органически входить в состав исполняемых программ и гибко реагировать на любые аномалии при обработке данных. Хранение и использование обработанных данных характеризуется меньшей динамичностью изменения, а ряд задач можно рассматривать как статические. Статическими являются также задачи защиты от несанкционированного доступа и от преднамеренных искажений массивов данных при их долговременном хранении. Однако задачи и методы защиты данных в процессе их оперативной обработки и записи для долговременного хранения являются динамическими. Эти особенности процессов определяют необходимость применения для контроля и защиты различных видов избыточности. Для контроля и защиты исполнения программ применяется преимущественно временная избыточность и оперативные методы восстановления. Для защиты данных в наибольшей степени применяются методы их помехозащитного кодирования и информационная избыточность.

Тема 6. Критерии надежности сложных программных комплексов

Для оценки надежности программ, как и при исследовании характеристик аппаратуры, как правило, приходится ограничиваться интегральными показателями наработки на отказ и средним временем восстановления. Определение остальных показателей сопряжено с большими трудностями, которые обусловлены тем, что для определения показателей надежности комплексов программ необходимы длительные эксперименты или сложные расчеты при определенных исходных данных.

Возможность быстрого автоматического восстановления и неполной потери функциональных характеристик при отказах приводит к предпочтительности анализа надежности комплексов программ при частичных ("мягких") отказах и сбоях с сохранением основных функций системы. Эти особенности комплексов программ приводят к необходимости анализа (наряду с отказами и сбоями) характеристик искажений результатов, определяющих достоверность результатов функционирования программ при наличии сбоев и отказов.

Оценка достоверности результатов и надежности функционирования комплекса программ представляет собой сложную задачу из-за "проклятия размерности". Естественным становится статистический подход к анализу надежности функционирования и статистическая оценка достоверности результатов. Качество отладки определяется частотой отказов и значениями ошибок в выходных результатах, полученными за счет не выявленных ошибок в программах и искажений исходных данных.

Рассмотрим ошибки, приводящие к искажениям выходных результатов. В этом случае критерием качества отладки программ можно считать величину дополнительной ошибки σ_j в j - х результатах решения задачи за счет пропущенных при отладке ошибок и оценивать степень отладки программ статистически следующим образом. Пусть при длительности отладки τ величина $P_k(\tau)$ есть вероятность наличия в программе ошибки k - го типа, которая при исполнении программы вносит в результирующую j - ю переменную дополнительную ошибку Δ_{kj}. Тогда значение ошибки в j - й переменной можно представить выражением

$$\sigma_j(\tau) = \sum_{k=1}^{m} P_k(\tau)\Delta_{kj} ,$$

где m - полное число типов, не выявленных в программе ошибок.

Приведенным выражением для определения статистической ошибки в результирующих величинах j - го типа можно пользоваться при близких значениях влияния различных ошибок на эффективность системы. Если одинаковые по значению ошибки в различных выходных данных различаются по своему воздействию на эффективность системы, то это влияние в каждой j-ой величине может быть учтено некоторым условным весом X_j. Формальная оценка значений X_j и Δ_{kj} затруднительна. Существующие методы оценки основаны на экспертном опросе при условии предварительной классификации m типов ошибок в программах (индекс k) и q выходных величин (индекс j). Результатом является общая средневзвешенная ошибка функционирования системы вследствие неполной отладки программ:

$$\sigma(\tau) = \sum_{j=1}^{q} \chi_j \sigma_j(\tau) = \sum_{j=1}^{q} \chi_j \sum_{k=1}^{m} P_k(\tau)\Delta_{kj} .$$

Потери эффективности управляющих программ за счет неполной отладки можно считать прямо пропорциональными (с коэффициентом δ) средним квадратическим ошибкам в выходных результатах:

$$\xi(\tau) = \delta_\sigma(\tau) = \delta \sum_{j=1}^{q} \chi_j \sum_{k=1}^{m} P_k(\tau)\Delta_{kj} .$$

Таким образом, оценка уровня отладки функционирования программ может производиться по значениям потерь из-за не устраненных ошибок в программе. Однако при такой оценке трудно учитывать ошибки, приводящие к отказам. Точное определение полного количества ошибок в программе прямыми методами измерения невозможно. Весьма проблематично определение вероятностей ошибок каждого типа. Имеются только косвенные пути статистической оценки их полного количества или вероятности ошибки в каждой команде программы. Такие оценки базируются на построении математических моделей в предположении жесткой корреляции между общим количеством и проявлениями ошибок в комплексе программ после его отладки в течение времени τ, т.е. между следующими параметрами (рис.1):

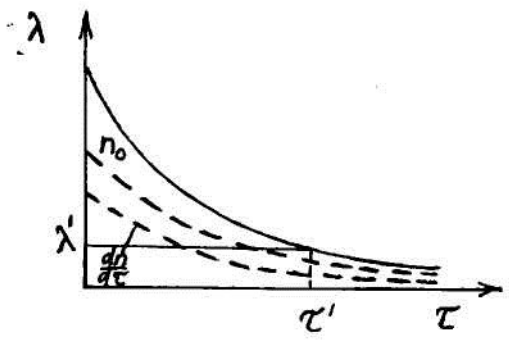


Рисунок 1. Зависимость суммарного количества ошибок в комплексе программ n_0, количества выявляемых в единицу времени ошибок $dn/d\tau$ и интенсивности отказов комплекса программ λ от времени отладки τ

• суммарным количеством ошибок n_0 в комплексе программ или вероятностью ошибки в каждой команде программы;
• количеством ошибок, выявляемых в единицу времени $dn/d\tau$ в процессе тестирования и отладки при постоянных усилиях на её проведение;
• интенсивностью отказов λ числом искажений результатов на выходе комплекса программ вследствие не выявленных ошибок при нормальном функционировании системы в единицу времени. Частота проявления ошибок при функционировании программы объёмом R команд и частота их обнаружения при отладке зависят от общего количества ошибок в программе n_0 или от вероятности ошибки в команде $P_0 = n_0 /R$. Наиболее доступно для измерения количество ошибок в программе, выявляемых в единицу времени в процессе отладки. Возможна непосредственная регистрация отказов и наиболее крупных искажений результатов, выявляемых средствами оперативного контроля в процессе нормального функционирования системы. Можно предположить, что все три показателя связаны некоторыми коэффициентами пропорциональности, значения которых зависят от интервала времени, на котором производится сопоставление, от быстродействия ВС, от эффективности средств автоматизации отладки и от некоторых других параметров. При фиксированных условиях разработки и функционирования некоторого комплекса программ эти коэффициенты имеют вполне определённые значения. Для другой подобной системы коэффициенты могут несколько измениться, однако оценки, проведенные для нескольких подобных конкретных систем, позволят прогнозировать эти характеристики и, следовательно, надежность функционирования программного обеспечения в зависимости от длительности отладки и ряда других факторов. Убывание ошибок в комплексе программ и интенсивности их обнаружения не беспредельны. После отладки в течение некоторого времени τ' интенсивность обнаружения ошибок при самых жестких условиях тестирования снижается настолько, что коллектив, ведущий разработку, попадает в зону нечувствительности к ошибкам и отказам. Создается представление о полном отсутствии ошибок, о невозможности и бесцельности их поиска, вследствие чего усилия на отладку сокращаются, и интенсивность обнаружения ошибок ещё больше снижается. Этой предельной интенсивности обнаружения отказов λ' соответствует наработка на отказ T' (рис. 2), при которой прекращается улучшение характеристик надежности комплекса программ на этапах отладки и испытаний у заказчика.

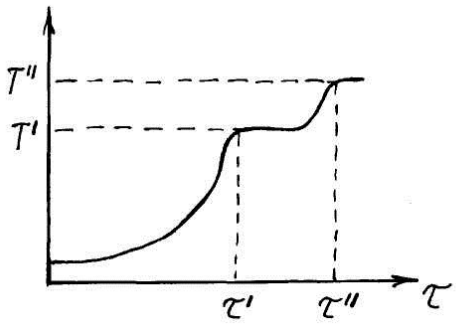


Рисунок 2. Зависимость наработки на отказ T от длительности отладки τ

При серийном выпуске систем с данным комплексом программ благодаря значительному расширению условий эксплуатации, возможно, некоторое возрастание суммарной (по всем экземплярам системы) интенсивности обнаружения ошибок. Это позволяет поднять наработку на отказ до значения T'' при проведении эксплуатации и проверок в течение времени τ''. Таким образом, можно определять и прогнозировать надежность комплекса программ в зависимости от длительности отладки и тиражирования программ. Для получения

конкретных оценок необходимы исходные данные, которые могут быть получены путем анализа процесса разработки и эксплуатации аналогичных систем и уточнены при создании конкретного анализируемого комплекса программ.

Как уже отмечалось, заданную наработку на отказ можно достигнуть не только путем устранения ошибок в комплексе программ, но и путем защиты от их проявления. Сокращая время восстановления до некоторого уровня, специфического для каждой системы, отказы можно преобразовать в сбои и тем самым повысить надежность системы и, прежде всего, наработку на отказ.

Для автоматического обнаружения отказов, их классификации и восстановления данных или вычислительного процесса необходимо создание соответствующих программ. Эти программы должны оперативно реагировать на аномалии функционирования комплекса программ и быстро осуществлять восстановление. Средства оперативного контроля и восстановления требуют затрат на их разработку и ресурсов для регулярного функционирования в вычислительной системе. Для достижения заданных характеристик надежности затраты на отладку комплекса программ могут быть несколько сокращены, если ввести в систему средства оперативного контроля и восстановления программными методами. Затраты на эти средства тем больше, чем больше осталось ошибок в программах и чем ниже затраты на отладку.

Таким образом, если задана некоторая наработка на отказ, то увеличение длительности отладки приводит к увеличению затрат на нее C_o и одновременно позволяет сократить затраты на оперативный контроль и восстановление в процессе эксплуатации $C_э$ (рис. 3). Существует некоторое оптимальное значение длительности отладки τ_0 , при котором заданная надежность обеспечивается при минимальных суммарных затратах. Возникает экстремальная задача определения τ_0 для заданной надежности функционирования комплекса программ. При изменении требований к величине наработки на отказ должна изменяться глубина контроля и характеристики восстановления, что отражено на рис. 3 пунктиром. Соответственно изменится оптимальное время отладки, при котором суммарные затраты минимальны. Приведенные в данном параграфе качественные зависимости между показателями надежности и пути их оценки иллюстрируют специфику критериев надежности комплексов программ. Трудность реального определения численных значений характеристик надежности ограничивает набор применяемых критериев надежности для анализа комплексов программ.

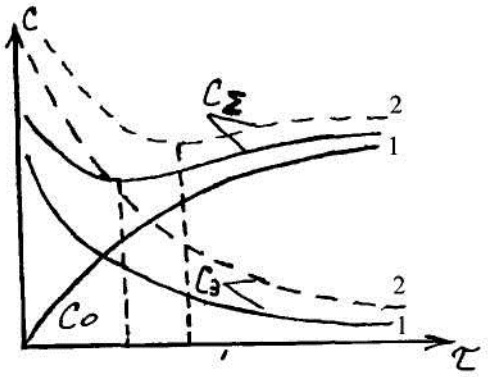


Рисунок 3. Зависимость затрат на отладку C_o, на оперативный контроль и восстановление $C_э$ и суммарных затрат C_Σ от времени отладки, обеспечивающих заданную наработку на отказ T (1) или T' (2)

Этот набор в настоящее время включает в себя среднее время наработки на отказ T, время восстановления $t_в$ и коэффициент готовности $K_г$. Для применения функции распределения показателей надежности и более тонких критериев надежности требуются

большой экспериментальный материал и дальнейшие исследования реальных характеристик надежности программ.

Тема 7. Модели надежности программного обеспечения

Для оценки значений показателей надежности программного обеспечения (ПО) по результатам тестирования и эксплуатации комплексов программ используют математические модели надежности. Под математической моделью надежности понимают выражение, связывающее значение одного из показателей надежности с непосредственно измеряемыми при тестировании системы параметрами. Например, в качестве показателей надежности могут выступать вероятность безотказного функционирования, средняя наработка на отказ, функция риска и т. д. Одним из главных требований к показателям надежности ПО является удобство их использования. Значения показателей будут зависеть от наблюдаемых при тестировании ПО отказов. Отказом ПО будем называть его любую неадекватную реакцию на ввод данных или вывод неверного результата вычислений в результате исполнения программы.

Математические модели, используемые для оценки показателей надежности оборудования в классической теории надежности, не подходят для расчета показателей надежности ПО. Это обусловлено тем, что причины возникновения отказов этих объектов имеют разную природу. Надежность оборудования зависит от двух факторов: от производственных дефектов оборудования и фактора старения и износа его деталей. Как правило, производственные дефекты оборудования являются легко обнаруживаемыми, и такое дефектное оборудование бракуется, поэтому основным фактором, влияющим на надежность оборудования, является фактор старения и износа его деталей.

На надежность ПО влияют те же факторы. Однако физическое хранение программ на магнитном носителе характеризуется очень высокой надежностью, поэтому принято считать, что показатели надежности ПО от времени не зависят. Следовательно, на надежность ПО влияют только наличие и характер производственных дефектов программы, то есть ошибок ПО. Из сказанного вытекает, что место и время возникновения ошибки невозможно прогнозировать. Для расчета показателей надежности программ за последние десятилетия было разработано немало моделей надежности.

Существующие математические модели позволяют оценивать характеристики ошибок в программах и прогнозировать их надежность при проектировании и эксплуатации. Модели имеют вероятностный характер, и достоверность прогнозов зависит от точности исходных данных и глубины прогнозирования по времени. Эти математические модели предназначены для оценки:

- показателей надежности комплексов программ в процессе отладки;
- количества ошибок, оставшихся не выявленными;
- времени, необходимого для обнаружения следующей ошибки в функционирующей программе;
- времени, необходимого для выявления всех ошибок с заданной вероятностью.

Использование моделей позволяет эффективно и целенаправленно проводить отладку и испытания комплексов программ, помогает принять рациональное решение о времени прекращения отладочных работ. Малый объем выборок реального количества обнаруженных ошибок и большой разброс времени обнаружения последовательных ошибок при завершении отладки не позволяют построить высокоточные математические модели. Поэтому целесообразно применять простейшие модели, точность которых близка к точности, обусловленной исходными данными.

Модели надежности ПО являются **эмпирическими моделями -** это модели, которые используют результаты тестирования программы в качестве исходных данных. Модели надежности ПО подразделяются на динамические и статические. **Динамическая модель** - это модель надежности ПО, которая является функцией от времени. При этом подразумевается, что текст программы параллельно с испытаниями дорабатывается. **Статическая модель** - это модель надежности ПО, которая не изменяется во времени. Динамические мо-

дели надежности ПО в свою очередь можно разделить на непрерывные модели надежности и ступенчатые модели надежности. **Непрерывная модель надежности** - это математическая модель, показатель надежности которой является строго монотонной функцией от времени. Первыми моделями надежности ПО были непрерывные модели, т. к. они были созданы по аналогии с моделями надежности оборудования. **Ступенчатая модель надежности** — это математическая модель, показатель надежности которой остается постоянным на каждом из интервалов тестирования ПО, но изменяется от интервала к интервалу. При обосновании математических моделей выдвигают некоторые гипотезы о характере проявления ошибок в программном обеспечении. Эти гипотезы апробированы при обработке данных реальных разработок. Рассмотрим более подробно несколько конкретных моделей.

1.1. Экспоненциальная модель

Экспоненциальная модель - это математическая модель надежности программного обеспечения, основанная на гипотезе, что количество ошибок, обнаруженных на некотором интервале времени, распределено по закону Пуассона. Эта непрерывная динамическая модель надежности является моделью изменения количества ошибок в программе в процессе отладки. Модель базируется на предположениях, которые заключаются в следующем:

1. Любые ошибки в программе являются независимыми и проявляются в случайные моменты времени с постоянной средней интенсивностью при отсутствии корректировок на всем интервале времени функционирования программы. Это свойство определяется наличием большого числа разнородных данных, необходимых для исполнения программ, что приводит к внешне случайному выбору маршрута, по которому исполняется программа в каждом конкретном случае.
2. Типы исполняемых в программе команд перемешаны, и время работы между ошибками определяется средним временем выполнения команды на данной ЭВМ и средним числом команд, используемым между ошибками. Это означает, что интенсивность проявления ошибок при реальном функционировании программ зависит от среднего быстродействия ЭВМ и практически не зависит от распределения типов команд на маршрутах обработки данных между ошибками.
3. Потенциальное множество тестов при отладке должно покрывать все множество реальных исходных данных при нормальном функционировании комплекса программ. Выбор отладочных тестов должен быть представительным и случайным, с тем, чтобы исключить концентрацию необнаруженных ошибок для некоторых реальных условий функционирования программы. В результате отсутствуют априорные данные для искусственного повышения интенсивности ошибок, и распределение частоты появления ошибок во времени следует считать равномерным и не зависящим от внешних факторов.
4. Ошибка, являющаяся причиной искажения результатов, фиксируется и исправляется после завершения тестирования либо вообще не обнаруживается. В период тестирования и отладки благодаря повышенному вниманию к искажениям результатов менее вероятно пропустить ошибку, чем в период нормальной эксплуатации программы. Однако при тиражировании программ и резком увеличении числа пользователей в течение некоторого времени, возможно, значительное возрастание интенсивности обнаруживаемых ошибок.
5. Количество ошибок, появляющихся в некотором интервале времени, распределено по закону Пуассона. В результате длительность непрерывной работы между искажениями распределена экспоненциально.
6. Количество обнаруживаемых и исправляемых ошибок в единицу времени пропорционально их общему количеству в программе на данном интервале времени.

Предположим, что в начале отладки комплекса программ при $\tau = 0$ в нем содержалось N_o ошибок. После отладки в течение времени τ осталось n_0 ошибок и устранено n ошибок $(n_o+n = N_O)$. При этом время τ соответствует длительности исполнения программ на ЭВМ для обнаружения ошибок и не учитывает простои машины, необходимые для анализа ре-

зультатов и проведения корректировок. Календарное время отладочных и испытательных работ с реальным комплексом программ значительно больше, так как после тестирования программ, на которое затрачивается машинное время τ, необходимо время для анализа результатов, на обнаружение и локализацию ошибок, а также на их устранение. Однако для определения характеристик надежности существенна только длительность непосредственного функционирования комплекса программ.

При постоянных усилиях на отладку интенсивность обнаружения искажений вычислительного процесса, программ или данных пропорциональна числу оставшихся ошибок в комплексе программ. Такое предположение естественно и проверено анализом реальных характеристик процесса обнаружения ошибок. Этот анализ позволил установить сильную корреляцию между значениями n_o и $dn/d\tau$. Тогда,

$dn/d\tau = K'\lambda = Kn_0 = K(N_0 - n)$,

где коэффициенты K и K' учитывают масштаб изменения времени, используемого для описания процесса обнаружения значений на входе проверяемого комплекса и другие параметры. Значение коэффициента K' можно определить как изменение темпа проявления искажений при переходе от функционирования программ на специальных тестах к функционированию на нормальных исходных данных. В начале отладки это различие может быть значительным, однако при завершении тестовые данные совпадают с исходными данными при нормальной эксплуатации. При этом K' равно единице ($K' = 1$).

Таким образом, интенсивность обнаружения ошибок в программе и абсолютное количество устранённых ошибок связываются уравнением

$dn/d\tau + Kn = KN_0$.

Если предположить, что в начале отладки при $\tau = 0$ отсутствуют обнаруженные ошибки, то решение уравнения имеет вид:

$n = N_0[1 - exp(-K\tau)]$.

Количество оставшихся ошибок в комплексе программ

$n_0 = N_0\, exp(-K\tau)$

пропорционально интенсивности обнаружения $dn/d\tau$ с точностью до коэффициента K.

Наработка на отказ, который рассматривается как обнаруживаемое искажение программ, данных или вычислительного процесса, нарушающее работоспособность, равна величине, обратной интенсивности обнаружения отказов (ошибок):

$$T = \frac{d\tau}{dn} = \frac{1}{KN_0}\exp(K\tau).$$

Если учесть, что до начала тестирования в комплексе программ содержалось N_0 ошибок, и этому соответствовала наработка на отказ T_o, то функцию наработки на отказ от длительности проверок можно представить в следующем виде:

$$T = T_0 \exp\left(\frac{\tau}{N_0 T_0}\right).$$

Если известны моменты обнаружения ошибок t_i, и каждый раз в эти моменты обнаруживается и достоверно устраняется одна ошибка, то, используя метод максимального правдоподобия, можно получить уравнение для определения значения начального числа ошибок N_o:

$$\sum_{i=1}^{n}\frac{1}{N_0 - (i-1)} = \frac{n\sum_{i=1}^{n} t_i}{N_0\sum_{i=1}^{n} t_i - \sum_{i=1}^{n}(i-1)t_i}$$

а также выражение для расчета коэффициента пропорциональности

$$K = \frac{n}{N_0 \sum_{i=1}^{n} t_i - \sum_{i=1}^{n} (i-1) t_i}$$

В результате можно рассчитать число оставшихся в программе ошибок и среднюю наработку на отказ $T = 1/\lambda$, т. е. получить оценку времени до обнаружения следующей ошибки.

В процессе отладки и испытания программ для повышения наработки на отказ от T_1 до T_2 необходимо обнаружить и устранить Δn ошибок. Величину Δn можно определить, выразив число обнаруживаемых ошибок через длительность наработки на отказ. Тогда

$$\Delta n = N_0 T_0 \left(\frac{1}{T_1} - \frac{1}{T_2} \right).$$

Аналогичными несложными преобразованиями получим выражение для определения затрат времени $\Delta\tau$ на проведение отладки, которые позволяют устранить Δn ошибок и соответственно повысить наработку на отказ от значения T_1 до T_2:

$$\Delta\tau = \frac{N_0 T_0}{K} \ln \frac{T_2}{T_1}.$$

Дальнейшая детализация модели надежности комплексов программ связана с уточнением содержания и значения коэффициента пропорциональности K. В процессе тестирования выбираются исходные данные для детальной проверки комплекса программ, что повышает вероятность обнаружения ошибок по сравнению с условиями нормальной эксплуатации. С одной стороны, при обнаружении каждого отказа часто выявляется не одна ошибка, с другой стороны некоторые корректировки программ могут вносить дополнительные ошибки. Например, для обнаружения одной ошибки требуется 0,61 прогона теста, а для проверки после корректировки программы требуется 1,35 прогонов тестов.

Следует подчеркнуть статистический характер приведенных соотношений. Неравномерность выбора маршрутов исполнения программы при нормальной эксплуатации, разное влияние конкретных ошибок в программах на надежность их функционирования, а также сравнительно небольшие значения n и Δn, особенно на заключительных этапах отладки, приводят к тому, что флюктуации соответствующих изменений наработки на отказ могут быть весьма значительными.

Развитием экспоненциальной модели описания количества ошибок является модель, учитывающая число команд R в испытываемой программе. В этом случае измеряется не абсолютное количество ошибок в программе, а количество ошибок на одну команду программы $P_0 = n_0 / R$. Это позволяет удобнее сравнивать характеристики программ более или менее близких по объему. При различии объема программ более чем на порядок программы могут столь сильно различаться по сложности, что относительное количество обнаруживаемых ошибок, будет различаться в несколько раз. Таким образом, учет количества ошибок на одну команду P_0 сглаживает, но не компенсирует полностью различие статистических данных для программ разного объема и сложности. Приведенные выше основные выражения практически сохраняются, только в некоторых местах вместо абсолютного количества ошибок следует использовать значение количества ошибок, нормированное на одну команду в программе.

1.2 модель частоты появления ошибок

Модель частоты появления ошибок - это математическая модель надежности программного обеспечения, основанная на гипотезе, что частота появления ошибок изменяется пропорционально количеству ошибок в программе и времени тестирования. Это ступенчатая динамическая модель надежности ПО. Модель базируется на следующих предположениях.

1. Частота появления ошибок линейно зависит от количества оставшихся в программе ошибок и от периода времени между моментами обнаружения двух последовательных ошибок.
2. Интенсивность обнаружения ошибок постоянна между моментами появления ошибок и изменяется после обнаружения и исправления очередной ошибки. Такая модель также называется моделью роста надежности ПО, так как предполагается, что по мере обнаружения и исправления ошибок время между последовательно обнаруженными ошибками возрастает.
3. Появление всех ошибок программы равновероятно и независимо.
4. Все ошибки имеют одинаковую степень серьезности.
5. Время непрерывной работы между отказами распределено экспоненциально.
6. При испытаниях ПО находится в среде близкой к реальным условиям.
7. Ошибки программы после их обнаружения незамедлительно корректируются, причем это не приводит к внесению новых ошибок.

Тогда, частота проявления ошибок зависит от времени испытания t_i между моментами обнаружения последовательных i - й и (i — l) - й ошибок:

$$\lambda(t_i) = K[N_0 - (i-1)] \cdot t_i$$

где N_0 - начальное количество ошибок; K – коэффициент пропорциональности, обеспечивающий равенство единице площади под кривой вероятности обнаружения ошибок.

Следует обратить внимание на отличие данной модели от экспоненциальной (1.1). Оно заключается в появлении множителя t_i, в выражении для частоты отказа. Для оценки надежности (наработки на отказ) получается выражение, соответствующее распределению Релея:

$$T(t_i) = \exp\left[- K[N_0 - (i-1)]\frac{t_i^2}{2} \right].$$

Отсюда плотность распределения времени наработки на отказ:

$$f(t_i) = -T'(t_i) = K[N_0 - (i-1)] \cdot t_i \cdot \exp\left[- K\left[N_0 - (i-1)\frac{t_i^2}{2} \right] \right].$$

Используя функцию максимального правдоподобия, получим оценку для общего количества ошибок N_0 и коэффициента K:

$$N_0 = \left[\frac{2n}{K} + \sum_{i=1}^{n} (i-1) t_i^2 \right] \frac{1}{\sum_{i=1}^{n} t_i^2};$$

$$N_0 = \left[\sum_{i=1}^{n} \frac{2}{N_0 - (i-1)} \right] \frac{1}{\sum_{i=1}^{n} t_i^2}.$$

В результате обработки экспериментальных данных по моментам обнаружения ошибок t_j может быть построена функция, позволяющая прогнозировать надежность комплекса программ и ожидаемое время обнаружения следующей ошибки. Эта функция отличается от приведенной ранее экспоненциальной модели, так как различаются исходные гипотезы о характере изменения частоты появления ошибок в процессе отладки. Для выбора наиболее адекватной модели необходимо исследовать реальные процессы обнаружения **и** устранения ошибок при разработке комплексов программ различных типов с большой статистикой выявления ошибок.

1.3 Модель Вейбулла

Модель Вейбулла - это математическая модель надежности программного обеспечения, в которой функция плотности распределения наработки на отказ описывается распреде-

лением Вейбулла. Это ступенчатая динамическая модель надежности ПО. Ее основная особенность в том, что она описывает как уменьшение, так и увеличение интенсивности отказов при устранении очередной ошибки в процессе тестирования.

В качестве основной функции в модели рассматривается распределение времени наработки на отказ $T(t)$. Если ошибка не устраняется, то интенсивность отказов является постоянной, что приводит к экспоненциальной модели для распределения:

$$T(t) = exp(-\lambda t).$$

Отсюда плотность распределения наработки на отказ определяется выражением

$$f(t) = \lambda e^{-\lambda t},$$

где $t > 0$, $\lambda > 0$ и $1/\lambda$ - среднее время наработки на отказ.

Для аппроксимации изменения частоты отказов от времени при обнаружении и устранении ошибок используется функция следующего вида:

$$\lambda(t) = \lambda\beta \cdot t^{\beta-1}.$$

Причем, если $0 < \beta < 1$, то интенсивность отказов снижается по мере отладки или в процессе эксплуатации;

если $\beta > 1$, то интенсивность отказов увеличивается по мере отладки или в процессе эксплуатации;

если $\beta = 1$, то интенсивность отказов постоянна во времени.

При таком виде функции $\lambda(t)$ плотность функции распределения наработки на отказ описывается двухпараметрическим распределением Вейбулла:

$$f(t) = \lambda\beta \cdot t^{\beta-1} \exp(-\lambda \cdot t^{\beta}).$$

Данная математическая модель использовалась в процессе проектирования и эксплуатации некоторых систем, для которых определены коэффициент λ и β. Распределение Вейбулла достаточно хорошо отражает реальные зависимости при расчёте времени наработки на отказ.

1.4 Статистическая модель Миллса

Статистическая модель - это математическая модель надежности программного обеспечения, которая строится на статистическом анализе количества ошибок в программе. Рассмотрим статистическую модель, разработанную Миллсом. В ней не используется никаких предположений о поведении функции риска. Эта модель является статической, так как надежность ПО в ней не изменяется во времени. 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 при последующем тестировании одинакова и зависит только от их количества. В результате обнаруживаемые при отладке ошибки состоят из ошибок двух типов: ранее неизвестных и искусственно введённых, известных только специалистам, проводящим эксперимент. Трудности искусственного ввода статистических ошибок стимулировали развитие метода, при котором ошибки не вводятся, а только помечаются (каждая реально обнаруженная ошибка). При последующей регистрации отказа производится диагностика: относится ошибка к группе ранее обнаруженных ошибок или является совершенно новой. Тестируя программу в течение некоторого времени и сортируя собственные и внесенные ошибки, можно оценить N - первоначальное число ошибок в программе.

Предположим, что в программу было внесено s ошибок, после чего разрешено начать тестирование. Пусть при тестировании обнаружено $n + v$ ошибок, причем, n - число найденных собственных ошибок, а v — число найденных внесенных ошибок. Тогда оценка для N по методу максимального правдоподобия будет такой:

$$N = s\, n / v.$$

Например, если в программу внесено 20 ошибок и к некоторому моменту тестирования обнаружено 15 собственных и 5 внесенных ошибок, значение N можно оценить в 60. В действительности N можно оценивать после обнаружения 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 об *N*. Примем, что в программе имеется не более *к* собственных ошибок, и внесем в нее еще *s* ошибок. Теперь программа тестируется, пока не будут обнаружены все внесенные ошибки, причем в этот момент подсчитывается число обнаруженных собственных ошибок (обозначим его *n).* Уровень значимости *C* вычисляется по следующей формуле:

C = 1 , при $n > к$,

$C = s / (s + к + 1)$, при $n \le к$.

Величина *C* является мерой доверия к модели. Это вероятность того, что модель будет правильно отклонять ложное предположение. Например, если мы утверждаем, что в программе нет ошибок ($к = 0$), и, внеся в программу 4 ошибки, все их обнаруживаем, не встретив ни одной исходной ошибки, то C = 0,8. Чтобы достичь уровня 95%, нам надо было бы внести в программу 19 ошибок. Если мы утверждаем, что в программе не более трех исходных ошибок, и, внеся шесть ошибок, обнаруживаем их все и не более трех исходных, уровень значимости равен 60%. Формула для *C* имеет под собой прочные статистические основания, выведена она Миллсом.

Эти две формулы и образуют полезную модель ошибок. Первая предсказывает число ошибок, а вторая может использоваться для установления доверительного уровня прогноза. Слабость этой формулы в том, что *C* нельзя предсказать до тех пор, пока не будут обнаружены все внесенные ошибки, а это может не произойти до самого конца этапа тестирования. Чтобы справиться с этой трудностью, можно модифицировать формулу для *C* так, чтобы *C* можно было оценить после того, как найдено *j* внесенных ошибок $(j \le s)$:

$C = 1$, при $n > к$, (3.22)

$C = (s / (j - 1)) / ((s + к + 1) / (к + j))$, при $n \le к$.

В предыдущем примере, где $к = 3$, а $s = 6$, если найдены 5 из 6 внесенных ошибок, величина *C* опускается с 60 до 33 %. Еще один график, который полезно строить во время тестирования, - текущее значение верхней границы *к* для некоторого фиксированного доверительного уровня, например 90%.

1.5. Проверка математических моделей

Обоснованием математических моделей занимались ряд зарубежных авторов: Hamilton P.A., Musa I.D., Sukert A.N. Ими контролировались и обрабатывались экспериментальные данные интенсивности обнаружения ошибок *dn/dt* на фиксированном интервале времени, количества обнаруженных ошибок *n* или наработки на отказ *T* в зависимости от времени функционирования программ на вычислительной системе. Характеристики, полученные по расчетам с использованием математических моделей, сопоставлялись с полученными экспериментальными значениями и применялись для прогнозирования показателей с последующим анализом отклонений от экспериментальных данных.

В одной из работ приведен пример анализа экспоненциальной модели. Определялся и прогнозировался интервал времени между последовательными отказами при непрерывном функционировании комплекса программ в зависимости от количества *n* обнаруженных и устраненных ошибок. Время на локализацию и устранение отказа не учитывалось, и средства для оперативного восстановления не применялись. Суммарное время функционирования программ, необходимое для достижения заданной наработки *T*, соответствует площади под кривой, которая ограничена вертикалью, исходящей из значения на кривой для заданного *T'*. Задаваясь необходимым T', можно оценить суммарную длительность функционирования программ и число ошибок, которое необходимо устранить для достижения *T'*. Экспериментальные данные достаточно хорошо совпадают с теоретическими (кривая на рис. 1).

В другой работе были исследованы ошибки в четырех комплексах программ, каждый из которых имел объем 100-120 тыс. команд. Комплексы программ предназначались для обработки радиолокационной информации в реальном масштабе времени и для управления на ко-

мандных пунктах. В трёх комплексах программ количество выявленных ошибок N_o составляло около 1,5-2 тыс. в каждом, что соответствовало 1,5-2 % числа команд. В одном из комплексов программ доля ошибок была на порядок меньше. Экспериментальные данные сравнивались с тремя теоретическими моделями. Модели соответствовали приведенным ранее экспоненциальной модели и модели частоты появления ошибок, а также несколько модифицированной модели частоты появления ошибок. Параметры моделей определялись методами наименьших квадратов и максимального правдоподобия по числу ошибок, выявленных за день и за неделю. Недельный интервал накопления ошибок позволяет прогнозировать точнее, а все три модели близки по достоверности. Прогнозы количества ошибок наиболее интересны при их малом количестве и относительно редком обнаружении, однако в этой области возрастает дисперсия прогнозируемой величины и трудно отдать предпочтение одной из исследованных моделей.

В третьей работе для оценки достоверности моделей анализировалось количество ошибок и, выявленных при функционировании комплексов
программ в течение времени τ. Значения N_o и K определялись методом максимального правдоподобия для каждого из 16 -исследованных вариантов создания больших программ. Пример изменения количества выявленных ошибок в зависимости от времени функционирования одного из комплексов программ представлен на рис. 2. Из графика следует, что экспоненциальная модель хорошо аппроксимирует количество ошибок во всём исследованном интервале времени. При значениях $n > 288$ отклонение реального количества ошибок от расчетного составляет около 21%.

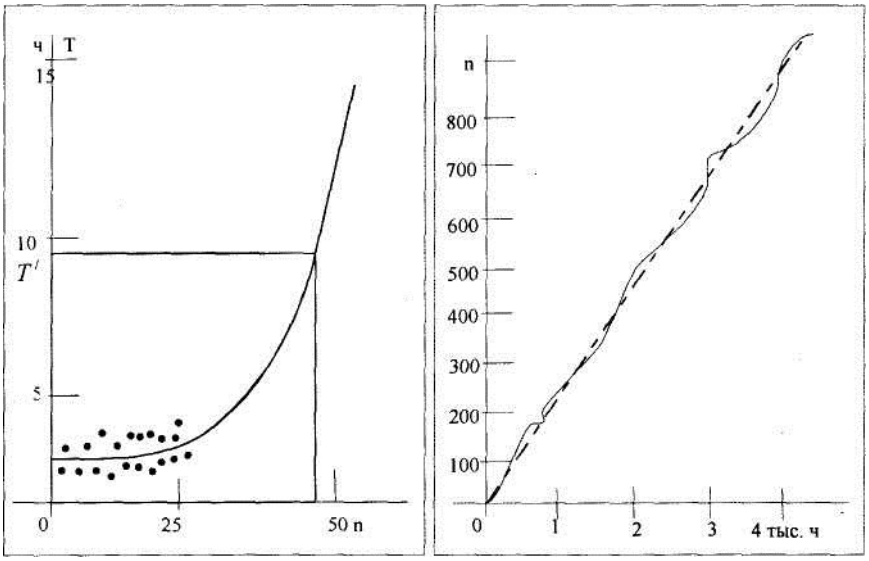


Рис. 1. Наработка на отказ T в зависимости от количества обнаруженных ошибок n(точки – экспериментальные данные, кривая соответствует первой модели)

Рис. 2. Количество выявленных ошибок n в зависимости от длительности

Тема 8. Методы проектирования надежного программного обеспечения

Термин "проектирование" обозначает различные виды деятельности по созданию программного обеспечения, начиная с определения требований и заканчивая написанием текста программы и ее тестированием. Он подразумевает наличие нескольких стадий проектирования, охватывающих весь жизненный цикл создания программного обеспечения. Рассмотрим некоторые принципы, общие для всех стадий проектирования.

Основными причинами ошибок программного обеспечения являются:

- большая сложность программного обеспечения, например, по сравнению с аппаратурой ЭВМ;
- неправильный перевод информации из одного представления в другое на макроуровне и микроуровне.

На макроуровне, уровне проекта, осуществляется передача и преобразование различных видов информации между организациями, подразделениями и конкретными исполнителями на всех этапах жизненного цикла ПО. На микроуровне, уровне исполнителя, производится преобразование информации по схеме: получить информацию - запомнить - выбрать из памяти (вспомнить) - воспроизвести информацию (передать).

Источниками ошибок (угрозами надежности) программного обеспечения являются:
- внутренние: ошибки проектирования, ошибки алгоритмизации, ошибки программирования, недостаточное качество средств защиты, ошибки в документации;
- внешние: ошибки пользователей, сбои и отказы аппаратуры ЭВМ, искажения информации в каналах связи, изменения конфигурации системы.

Методы проектирования надежного программного обеспечения в соответствии с их целями можно разбить на три группы: предупреждение ошибок, обнаружение ошибок и обеспечение устойчивости к ошибкам.

Предупреждение ошибок

Предупреждение ошибок - это методы проектирования, позволяющие минимизировать или исключить появление ошибок в программном обеспечении. Методы предупреждения ошибок концентрируются на отдельных этапах процесса проектирования программного обеспечения. Их можно разбить на следующие категории:

1. Методы, позволяющие справиться со сложностью системы.
2. Методы достижения большей точности при переводе информации.
3. Методы улучшения обмена информацией.
4. Методы немедленного обнаружения и устранения ошибок на каждом шаге (этапе) проектирования, не откладывая их на этап тестирования программы.

Сложность системы является одной из главных причин низкой надежности программного обеспечения. В общем случае, сложность объекта является функцией взаимодействия (количества связей) между его компонентами. В борьбе со сложностью программного обеспечения используются две концепции:

1. *Иерархическая структура.* Иерархия позволяет разбить систему по уровням понимания (абстракции, управления). Концепция уровней позволяет анализировать систему, скрывая несущественные для данного уровня детали реализации других уровней. Иерархия позволяет понимать, проектировать и описывать сложные системы.
2. *Независимость.* В соответствии с этой концепцией, для минимизации сложности, необходимо максимально усилить независимость элементов системы. Это означает такую декомпозицию системы, чтобы её высокочастотная динамика была заключена в отдельных компонентах, а межкомпонентные взаимодействия (связи) описывали только низкочастотную динамику системы.

Предупреждение ошибок — лучший путь повышения надёжности программного обеспечения. Для его реализации целесообразно использовать методику проектирования, соответствующую спиральной модели жизненного цикла программного обеспечения. Она предусматривает последовательное понижение сложности на всех этапах проектирования. Методика включает следующие основные этапы: анализ, проектирование, внедрение и сопровождение.

На этапе *анализа* необходимо определить, что должна делать проектируемая система. Этот этап включает в себя следующие шаги. Обследование объекта и определение требований пользователей. Формулирование целей и принципов построения системы, обеспечивающих выполнение требований пользователей. Разработка архитектуры - декомпозиция функциональной структуры объекта с использованием методологии структурного анализа и проектирования. Разработка модели функционирования объекта по методологии диаграмм

потоков данных, которые описывают процессы преобразования информации от ее ввода в систему до передачи конечным пользователям.

На этапе *проектирования* определяется, как система будет работать. Этот этап состоит из следующих шагов. Разработка системы классификации - определение многомерного пространства решений системы. Разработка информационной модели системы - проектирование логической и физической структуры базы данных системы с помощью диаграмм "сущность - связь". Синтез структуры программного обеспечения: в программные модули системы объединяются функции, выполняемые, как правило, одним пользователем системы на ограниченном подмножестве информационной модели системы. Проектирование модулей -разработка спецификаций, описывающих связи между модулями, и проектирование логики модулей. Программирование модулей с использованием методов структурного и объектно-ориентированного программирования. Интеграция - сборка компонентов системы и её тестирование. Разработка методического обеспечения: руководств пользователей, инструкций по эксплуатации, технологических инструкций.

Этап *внедрения* автоматизированной системы управления на конкретном объекте включает следующие виды работ. Структурно-функциональный анализ объекта и реинжиниринг его бизнес-процессов. Конфигурирование системы под программно-техническую и организационно-экономическую структуру объекта, разграничение прав доступа пользователей к ресурсам. Классификация объектов учета и управления и идентификация их параметров. Адаптация входных интерфейсов и настройка бланков выходных документов. Разработка алгоритмов бизнес-процессов. Обучение пользователей, ввод системы в опытную и промышленную эксплуатацию.

Сопровождение объединяет промышленную эксплуатацию и развитие системы. На этом этапе осуществляется поиск и исправление ошибок, устранение замечаний, функциональное расширение системы на основании дополнительных требований пользователей. В зависимости от вида выполняемых работ осуществляется переход на один из предыдущих этапов проектирования системы.

Очевидно, что предупреждение ошибок - оптимальный путь к достижению надежности программного обеспечения. Лучший способ обеспечить надежность - прежде всего не допустить возникновения ошибок. Однако, гарантировать отсутствие ошибок невозможно никогда. Другие методы опираются на предположение, что ошибки все-таки будут.

Тема 9. Обнаружение ошибок. Обеспечение устойчивости к ошибкам

Обнаружение ошибок — это методы проектирования, направленные на разработку дополнительных функций программного обеспечения, помогающих выявить ошибки. Если предполагать, что в программном обеспечении ошибки все же будут, то лучшая (после предупреждения ошибок) стратегия - включить средства обнаружения ошибок в само программное обеспечение.

Большинство методов этой группы направлено, по возможности, на немедленное обнаружение сбоев. Немедленное обнаружение имеет два преимущества: можно минимизировать как влияние ошибки, так и последующие затруднения для человека, которому придется искать информацию об этой ошибке, находить ее место в программе и исправлять. Методы обнаружения ошибок базируются на введении в программное обеспечение системы различных видов избыточности: временной, информационной и программной.

Обеспечение устойчивости к ошибкам

Обеспечение устойчивости к ошибкам - методы проектирования, направленные на исправление ошибок и их последствий и обеспечивающие функционирование системы при наличии ошибок. Следующий шаг после того, как ошибка обнаружена, либо она сама, либо ее последствия должны быть исправлены программным обеспечением. Исправление ошибок самой системой -плодотворный метод проектирования надежных систем аппаратного обеспечения. Некоторые устройства способны обнаружить неисправные компоненты и перейти к использованию идентичных запасных. Аналогичные методы неприменимы к программному обеспечению вследствие глубоких внутренних различий между сбоями аппаратуры и ошибками в программах. Если некоторый программный модуль содержит ошибку, идентичные "запасные" модули также будут содержать ту же ошибку.

Методы этой группы ставят своей целью обеспечить функционирование программной системы при наличии в ней ошибок. Они разбиваются на три подгруппы: динамическая избыточность, методы отступления и методы изоляции ошибок.

1. Истоки концепции *динамической избыточности* лежат в проектировании аппаратного обеспечения. Один из подходов к динамической избыточности - метод *голосования.* Данные обрабатываются независимо несколькими идентичными устройствами, и результаты сравниваются. Если большинство устройств выработало одинаковый результат, этот результат и считается правильным. И опять, вследствие особой природы ошибок в программном обеспечении ошибка, имеющаяся в копии программного модуля, будет также присутствовать во всех других его копиях, поэтому идея голосования здесь, видимо, неприемлема. Предлагаемый иногда подход к решению этой проблемы состоит в том, чтобы иметь несколько неидентичных копий модуля. Это значит, что все копии выполняют одну и ту же функцию, но либо реализуют различные алгоритмы, либо созданы разными разработчиками. Этот подход не всегда эффективен по следующим причинам. Часто трудно получить существенно разные версии модуля, выполняющие одинаковые функции. Кроме того, возникает необходимость в дополнительном программном обеспечении для организации выполнения этих версий параллельно или последовательно и сравнения результатов. Это дополнительное программное обеспечение повышает уровень сложности системы, что, конечно, противоречит основной идее предупреждения ошибок -стремиться в первую очередь минимизировать сложность.

Второй подход к динамической избыточности - выполнять запасные копии модулей только тогда, когда полученные с помощью основной копии результаты признаны неправильными. Если это происходит, то система автоматически вызывает запасную копию. Если и ее результаты неправильны, вызывается другая запасная копия и т. д. Этот подход также страдает большинством перечисленных ранее недостатков. Кроме того, вполне вероятно, что если ресурсы на реализацию проекта фиксированы, то при реализации "запасных" версий на проектирование и тестирование будет выделено меньше ресурсов, чем можно было выделить, если бы реализовывалась лишь одна копия и динамическая избыточность не использовалась.

2. Вторая подгруппа методов обеспечения устойчивости к ошибкам называется методами *отступления* или сокращенного обслуживания. Эти методы приемлемы обычно лишь тогда, когда для системы программного обеспечения важно благополучно закончить работу. Например, если ошибка оказывается в системе, управляющей технологическими процессами, и в результате эта система выходит из строя, то может быть загружен и выполнен особый фрагмент программы, призванный подстраховать систему и обеспечить безаварийное завершение всех управляемых системой процессов. Аналогичные средства необходимы в операционных системах. Если операционная система обнаруживает, что она вот-вот выйдет из строя, она может загрузить аварийный фрагмент, отвечающий за оповещение пользователей у терминалов о предстоящем сбое и за сохранение всех критических для системы данных.

3. Последняя подгруппа - методы *изоляции ошибок.* Основная их идея - не дать последствиям ошибки выйти за пределы как можно

меньшей части системы программного обеспечения. Если ошибка возникнет, то не вся система оказывается неработоспособной, а отключаются лишь отдельные функции в системе либо некоторые ее пользователи. Например, во многих операционных системах изолируются ошибки отдельных пользователей так, что сбой влияет лишь на некоторое подмножество пользователей, а система в целом продолжает функционировать. В телефонных переключательных системах для восстановления после ошибки, чтобы не рисковать выходом из строя всей системы, просто разрывают телефонную связь. Другие методы изоляции ошибок связаны с защитой каждой из программ в системе от ошибок других программ. Ошибка в прикладной программе, выполняемой под управлением операционной системы, должна оказывать влияние только на эту программу. Она не должна сказываться на самой операционной системе или других программах, функционирующих в этой системе.

На базе методов обеспечения устойчивости к ошибкам, направленных на минимизацию ущерба при появлении ошибок, разрабатываются средства повышения надежности программного обеспечения, реализующие выполнение следующих функций.

Метод *динамической избыточности* реализуют средства:
- динамического изменения конфигурации;
- повторного выполнения операций. Метод *отступления* реализуют средства:
- обработки ошибок;
- сокращенного обслуживания в случае отказа отдельных функций системы.

Метод *изоляции ошибок* реализуют средства:
- контроля действий пользователей;
- обработки сбоев аппаратуры;
- анализа корректности данных;
- копирования и восстановления данных.

Важное обстоятельство, касающееся всех методов, состоит в том, что обнаружение ошибок и обеспечение устойчивости к ошибкам в некотором отношении противоположны методам предупреждения ошибок. В частности, эти методы требуют дополнительных функций от самого программного обеспечения. Тем самым не только увеличивается сложность готовой системы, но и появляется возможность внести дополнительные ошибки при реализации этих функций. Как правило, методы предупреждения и обнаружения ошибок применимы к любому программному проекту. Методы обеспечения устойчивости к ошибкам применяются не так широко, это, однако, зависит от области приложения. Если рассматривается, скажем, система реального времени, то ясно, что она должна сохранить работоспособность и при наличии ошибок, а тогда могут оказаться желательными и методы исправления и обеспечения устойчивости к ошибкам. К системам такого типа относятся телефонные переключательные системы, системы управления технологическими процессами, аэрокосмические и авиационные диспетчерские системы и операционные системы широкого назначения.

Тема 9. Виды избыточности программного обеспечения

В процессе проектирования недостаточно создать правильные программы, выдающие необходимые данные при идеальных исходных данных и абсолютном отсутствии возмущений. Требуется разрабатывать надежные программы, устойчивые к различным возмущениям и способные сохранять требуемое качество обработки в реальных условиях функционирования. В любых ситуациях должны быть исключены катастрофические последствия и длительные отказы или, в крайнем случае, их влияние на внешних абонентов должно быть минимальным.

Комплексы программ в процессе функционирования находятся под воздействием шумов - возмущений различных типов и происхождения. Для снижения их влияния применяются фильтры, позволяющие обнаруживать искажения, выбирать их, устранять или умень-

шать вредные последствия. Разнообразие видов искажений приводит к необходимости построения системы фильтров, каждый из которых выбирает некоторые виды искажений.
Реализация фильтров основана на использовании избыточности. Избыточность - это процесс, который используется для обнаружения ошибок и выработки мер по снижению их последствий при функционировании комплексов программ. Избыточность используется для контроля искажений как самого процесса функционирования комплекса программ, так и обрабатываемых им данных. Основная задача ввода избыточности состоит в ограничении аварийных последствий от возмущений, соответствующих отказу системы.

Любые аномалии при исполнении программ необходимо блокировать и ограничивать по последствиям на уровне сбоя с максимально быстрым восстановлением. Не обязательно всегда устанавливать причину искажения, главная задача сводится к быстрому восстановлению нормального функционирования. Для этого в комплексы программ вводят средства, решающие следующие задачи (рис. 1.):

Рис. 1. Этапы обеспечения надежности комплексов программ

- оперативное обнаружение искажений программ, вычислительного процесса, промежуточных или результирующих данных;
- селекцию и распознавание искажений, оценку возможных последствий для функционирования комплекса программ;
- принятие решения и выбор операций для восстановления нормального функционирования;
- регистрацию и накопление данных о выявленных искажениях и оперативно принятых мерах для их ликвидации или ограничения области влияния.

Для обеспечения защиты вычислительного процесса и информации программно-алгоритмическими методами используются программная, информационная и временная избыточность.

Временная избыточность

Под **временной избыточностью** понимается процесс, использующий часть производительности ЭВМ для контроля исполнения и восстановления работоспособности системы после сбоя. При проектировании комплекса программ должен предусматриваться запас производительности, который будет использоваться для контроля и повышения надежности функционирования. Значение временной избыточности зависит от требований к надежности функционирования системы и составляет от 5—10% производительности однопроцессорной ЭВМ до трех- и четырехкратного дублирования производительности ЭВМ в мажоритарных вычислительных комплексах. При функционировании комплексов программ в реаль-

ном масштабе времени резерв времени для контроля и восстановления вычислительного процесса и информации не устанавливается заранее. Для диагностики искажений и операций восстановления требуется в общем случае небольшой интервал времени, который выделяется за счет резерва или сокращения времени решения функциональных задач. После восстановления резерв времени считается полностью пополненным.

Информационная избыточность

Информационная избыточность - это процесс дублирования части данных вычислительной системы для обеспечения надежности и контроля данных. Избыточность используется для обеспечения достоверности входных и промежуточных данных, которые в наибольшей степени влияют на нормальное функционирование комплекса программ, а также данных, которые требуют значительного времени восстановления. Для восстановления таких данных требуется их длительное накопление и обработка. Информационная избыточность способствует не только обнаружению искажения данных, но и устранению ошибок. Данные защищают двух- и трехкратным дублированием с соответствующей дисциплиной контроля сохранности и периодическим обновлением. Для менее важных данных информационная избыточность используется в виде помехозащитных кодов, позволяющих только обнаруживать искажения. Искаженные данные исключаются из обработки, и происходит их естественное обновление в процессе последующего функционирования. Многие данные вычислительной системы не защищаются вследствие их частого обновления или слабого влияния возможных искажений на решение основных функциональных задач.

Программная избыточность

Программная избыточность - это процесс, используемый для контроля и обеспечения достоверности важных функций управления и обработки информации. Программная избыточность базируется на выполнении системой следующих функций:

- взаимное недоверие элементов системы;
- немедленное обнаружение и регистрация ошибок;
- выполнение одинаковых функций разными модулями системы и сопоставление результатов обработки;
- контроль и восстановление данных с использованием других видов избыточности.

При использовании метода взаимного недоверия компоненты системы проектируются, исходя из предположения, что другие компоненты и исходные данные содержат ошибки. Перед началом обработки программные компоненты должны проверить корректность входных параметров и исходных данных и попытаться обнаружить в них ошибки. В случае обнаружения ошибки информация о ней передается другим модулям системы или пользователю для принятия решения.

Для повышения надежности выполнения важных функций системы в комплексах программ применяются несколько вариантов модулей, различающихся методами решения задачи или алгоритмом реализации одного и того же метода. Тем самым, появляется возможность сопоставлять результаты обработки одинаковых исходных данных разными модулями и исключать искажение результатов, обусловленное программными ошибками или сбоями в системе.

Программная избыточность также необходима для реализации программ контроля и восстановления данных с использованием информационной избыточности и для функционирования средств защиты, использующих временную избыточность.

Следует отметить психологические трудности введения избыточности для обеспечения надежности программ. Традиционный глубокий оптимизм создателей комплексов программ относительно возможности собственных ошибок и внешних возмущений приводит к массовому пренебрежению методами программной защиты или их незначительному использованию. Этому способствует также ограниченность ресурсов вычислительных систем. Избыточность, необходимая для обеспечения надежного функционирования комплексов программ, должна определяться на начальных этапах проектирования и целенаправленно использоваться для решения задач контроля и восстановления.

Тема 10. Организация и проведение испытаний на надежность ПО

Принципы и задачи статистического тестирования программ

Детерминированное тестирование позволяет проверить программы при ограниченном количестве фиксированных исходных данных. Реальное функционирование комплексов программ проходит в стохастической среде. Поэтому сложные комплексы программ необходимо подвергать статистическому тестированию для расширения условий функционирования и для получения более достоверных данных о надежности программ. Статистическая проверка программ преследует две цели:

- обнаружение ошибок в комплексе программ, отладку программ и обеспечение характеристик комплекса программ в соответствии с требованиями технического задания **(комплексная отладка программ);**
- испытание комплекса программ в реальных условиях и демонстрация заказчику полноты выполнения требований технического задания.

Статистические проверки проводятся комплексно для получения различных характеристик программ, в том числе надежности. Далее рассматриваются общие методы и задачи статистического тестирования с некоторым акцентом на особенности, необходимые для получения надежных комплексов программ и определения показателей надежности. Статистической проверке наиболее широко подвергаются сложные группы программ, так как их невозможно с необходимой достоверностью проверить методами формализованного контроля и по детерминированным тестам. Статистическую проверку и отладку делят на две части: отладку по статистическим характеристикам выходных результатов реализации тестов и отладку по статистическим характеристикам динамики функционирования комплекса программ и использования им вычислительных ресурсов ВС.

Первая часть статистической отладки предусматривает получение и анализ выходных результатов функционирования групп и всего комплекса программ при изменении исходных переменных в соответствии со статистическими законами работы источников информации. Эти результаты проверяются по их законам распределения и параметрам этих законов на соответствие эталонам, полученным различными методами или заданным требованиями технического задания. Статистическое изменение исходных данных расширяет диапазон возможных сочетаний переменных и соответствующих им логических маршрутов обработки информации. В результате увеличивается глубина проверки программ по сравнению с той, которую удается достигнуть при детерминированном тестировании. Отклонение параметров или законов распределения выходных данных от эталонных является признаком наличия в программах ошибок, которые подлежат выявлению и локализации всеми доступными методами и, прежде всего, с помощью детерминированного тестирования.

Статистическая отладка по характеристикам динамики функционирования программ и использования ими ресурсов ВС предусматривает проверку характеристик программ по времени их реализации, последовательности включения, реальному использованию памяти и внешних устройств ВС и т. д. Эта отладка должна обеспечивать эффективное динамическое использование ресурсов ВС и выявлять системные ошибки, обусловленные превышением потребности в ресурсах для реализации комплекса программ по сравнению с имеющимися, при различных режимах функционирования внешних абонентов. Признаком наличия ошибок может быть повышенная частота вызова некоторых подпрограмм, частота обращения к зонам памяти и другие динамические характеристики процесса исполнения программ. Ошибки такого типа могут обнаруживаться путем аналитического расчета статистических характеристик отлаживаемого комплекса программ (по использованию производительности и других параметров) ВС или путем контроля использования производительности ВС при проверке функционирования программ в реальном масштабе времени в условиях, близких к определяемым техническим заданием.

Статистическими методами обнаруживаются принципиальные алгоритмические и системные ошибки, для устранения которых требуется значительная доработка или замена подпрограмм. Для этого метода отладки удается автоматизировать в основном формирование тестов исходных данных и статистическую обработку результатов функционирования программ. Все остальные задачи по обнаружению, выявлению и исправлению ошибок пока решаются вручную.

Основным принципом статистической отладки комплексов программ является последовательное иерархическое усложнение задач, увеличение объема совместно функционирующих программ, анализируемой и преобразуемой информации. По уровню сложности, степени связи с реальным масштабом времени и функционированием аппаратуры процесс отладки комплексов программ можно разделить на следующие этапы:

- автономная отладка отдельных функциональных и стандартных подпрограмм в статике;
- комплексная отладка групп функциональных программ в статике;
- комплексная отладка комплекса программ в динамике без подключения к внешним абонентам;
- комплексная отладка в реальных условиях функционирования с внешними абонентами.

Следует подчеркнуть, что обнаружение и локализация ошибок при автономной отладке осуществляется значительно проще и быстрее, чем при комплексной отладке. К комплексной отладке следует переходить только тогда, когда использованы практически все доступные методы автономной отладки и обеспечена уверенность в работоспособности подпрограмм и соответствие их заданным требованиям.

Статистическая комплексная отладка программ

Основная задача статистической отладки комплекса программ состоит в завершении разработки всего изделия и в доведении его характеристик до значений, заданных требованиями технического задания. Для решения этой задачи должно быть:

- проверено сопряжение и взаимодействие по передачам управления и информации всех подпрограмм, входящих в комплекс;
- обеспечено получение в процессе функционирования всех характеристик (в том числе характеристик надежности), заданных требованиями технического задания, а также вспомогательных, обоснованных при подготовке технического проекта;
- проверена полнота и состав технической документации и точное соответствие ее изделию - комплексу программ.

Важнейшим принципом комплексной отладки является последовательное завершение отработки функционально законченных групп программ и их сопряжение в соответствии с иерархической схемой комплекса в целом. Каждая функционально законченная группа программ должна проходить отладку "от простого к сложному" до уровня, позволяющего проверить удовлетворение требований технического задания. Среди частных критериев внимание следует уделять тем, к которым наиболее чувствительны параметры, зафиксированные в техническом задании.

Статическая комплексная отладка должна обеспечивать проверку и корректировку сопряжения автономно отлаженных подпрограмм по информации и по управлению в некоторые фиксированные моменты времени. При этом не полностью учитывается динамика последовательного включения подсистем и решения различных функциональных задач во времени. По сравнению с автономной отладкой объем одновременно функционирующих программ увеличивается, однако, по существу, в каждый момент времени статически проверяется определенная функциональная группа программ. При объеме комплекса программ 100 тыс. команд в систему включаются сопряженные группы программ объемом в 5—15 тыс. команд.

При статической комплексной отладке взаимодействие подпрограмм подлежит проверке и корректировке без включения их в контур управления и без обработки информации реальных объектов. Реакция абонентов на поступающие данные с учетом временных харак-

теристик не рассматривается, упрощенно имитируются процессы и программы обмена с внешними абонентами.

Динамическая комплексная отладка без **реальных абонентов.** На этом этапе осуществляется динамическая проверка взаимодействия при решении различных функциональных задач во времени. Внешние абоненты имитируются специальными программами либо техническими устройствами, моделирующими поступающую информацию и объекты управления. Случайные моменты времени поступления сообщений и заявок на включение определенных подпрограмм, изменение интенсивностей потоков заявок, случайный состав накопленной информации и время решения каждой задачи приводят к случайному характеру подключения и условий взаимодействия подпрограмм, что подлежит контролю и отладке на данном этапе. Его следует рассматривать как этап, завершающий комплексную отладку и обеспечивающий получение основных статистических характеристик системы на уровне требований технического задания.

Основными задачами динамической комплексной отладки программ без реальных абонентов являются:

- проверка и отладка начального режима включения комплекса программ при отсутствии информации от внешних абонентов;
- проверка и отладка взаимодействия функциональных программ с подпрограммами обмена информацией и с другими прерывающими подпрограммами;
- проверка и отладка взаимодействия групп функциональных подпрограмм при различной последовательности их включения в систему в статистическом режиме нормальной загрузки;
- проверка и отладка взаимодействия программ в критических режимах ВС, при перегрузке, запаздываниях, потере сообщений и заявок на включение подпрограмм и при сочетаниях исходных данных, наиболее опасных для функционирования системы;
- проверка и обеспечение надежности функционирования комплекса программ при искажении информации внешних абонентов, при наличии ошибок в программах и при сбоях и отказах компонент ВС;
- проверка и обеспечение эффективности системы предстартового и оперативного контроля, а также средств восстановления функционирования программ и взаимодействия компонент ВС.

Последовательное подключение и наращивание объема программ для динамической комплексной отладки осуществляется с использованием двух стратегий. При первой к системе вначале подключаются независимые и слабозависимые функциональные программы. Их наращивание идет в порядке увеличения взаимозависимости. Такая последовательность позволяет сократить объем имитируемой информации и потребность в разработке имитаторов и моделей. В ряде случаев целесообразна стратегия, использующая признаки взаимодействия программ. С помощью имитаторов стремятся обеспечить возможность статистического ввода внешней информации с последовательным увеличением объема и усложнением состава вводимых данных. Функциональные программы подключаются в зависимости от поступающих сообщений или формируемых заявок.

Перед отладкой составляется временная диаграмма включения групп функциональных программ и схема изменения состава информации, используемой взаимодействующими программами. Процесс контроля интерфейсной информации является самым сложным при динамической комплексной отладке в связи с разнообразием последовательностей включения программ в реальном масштабе времени.

Тема 11. Динамическая комплексная отладка с реальными абонентами

Этот этап динамической отладки предназначен для завершения отладки с учетом статистических характеристик аппаратуры, источников информации и внешних абонентов. В

динамике функционирования системы уточняются характеристики процессов управления и обработки информации, выявляются неучтенные статистические особенности функционирования внешних абонентов, проверяется выполнение требований технического задания. Отладка носит итерационный характер с возвратом на предыдущие этапы вплоть до детерминированной отладки. Объем работ определяется качеством и полнотой отладки на предыдущих этапах. Он зависит от достоверности имитации характеристик реальных абонентов и логики их функционирования, стоимости и возможности проведения статистических экспериментов. Степень неадекватности моделей определяет необходимый объем динамической комплексной отладки. При этом в результатах функционирования обнаруживаются некоторые особенности, которые прежде всего обусловлены отличием реакции реальных абонентов от заложенной в модели, а также проявлением различного рода неучтенных помех и возмущений при взаимодействии с реальными абонентами.

При тщательной отладке с имитаторами этот этап отладки сводится к контрольной проверке и корректировке программ в соответствии с отклонениями статистических характеристик реальных объектов от предполагавшихся. Однако в ряде случаев затраты на создание адекватных моделей и имитаторов представляются проектировщикам излишними, так как они исходят из предположения (практически всегда неверного), что в разработанных и прошедших предшествующие этапы отладки программах отсутствуют ошибки. Центр тяжести динамической комплексной отладки переносится на этап отладки с реальными абонентами, что сложнее и дороже и не может обеспечить варьирование переменных во всем диапазоне их изменения, особенно при тех значениях, при которых создаются критические и опасные ситуации.

На этом этапе проверяются и отлаживаются средства предстартового и оперативного контроля, обеспечивающие помехоустойчивость и надежность функционирования комплекса программ. Эти задачи трудно решить с имитаторами и моделями, так как наиболее ненадежные звенья комплекса программ и источники помех не всегда можно предвидеть заранее и они выявляются в процессе функционирования системы.

Для серийных систем динамическая комплексная отладка обеспечивает отработку технических условий (в том числе показателей надежности) на комплекс программ, по которым заказчик принимает ВС с комплексом программ. В процессе отладки необходимо подготовить и испытать некоторые тесты, включаемые в технические условия и обеспечивающие проверку ВС. Следует отметить, что на всех этапах отладки происходит уточнение задач системы, методов их решения, способов их алгоритмической и программной реализации, параметров технического задания.

Тема 12. Статистическая проверка длительности исполнения комплекса программ и пропускной способности системы

Для решения задач в реальном масштабе времени необходимо статистическое соблюдение временного баланса между темпом вызова программ и длительностью их исполнения. Темп вызова функциональных программ зависит от интенсивности поступления сообщений на обработку и от заданных периодов решения периодических задач. Загрузка меняется и в некоторые моменты времени производительность ВС может оказаться недостаточной для обработки поступающих сообщений. Следствием этого являются отказы в обработке и ухудшение показателей надежности системы. Задача состоит в определении вероятности, с которой нарушается соответствие между потребностью в производительности и реальными возможностями ВС. Для определения показателей надежности комплекса программ, связанных с ограниченной производительностью ВС, необходимо:

- определить средние значения интенсивности поступающих сообщений и распределения плотности потоков для различных типов сообщений;
- определить длительность решения каждой из функциональных задач, обрабатывающих сообщение или включаемых периодически;

- определить загрузку ВС в нормальном режиме поступления сообщений, а также распределение значений и длительностей перегрузки;
- определить влияние пропуска в обработке сообщений и снижения темпа решения периодических задач на показатели надежности функционирования комплекса программ.

Перечисленные задачи могут быть решены в процессе статистических испытаний и эксплуатации комплекса программ, однако при этом велик риск, что производительность ВС окажется недостаточной для решения заданной совокупности задач в реальном масштабе времени. Кроме того, условия испытаний или эксплуатации ограниченного числа систем не всегда соответствуют режимам массового применения. Поэтому при экспериментах требуется принимать специальные меры для создания реальных и достаточно тяжелых по загрузке условий функционирования комплексов программ. Определение пропускной способности при этом проводится на завершенном комплексе программ. В случае несоответствия пропускной способности ВС реальным потребностям разработчик оказывается перед необходимостью значительно перерабатывать готовые, отлаженные и испытанные программы.

Достоверность определения пропускной способности ВС с комплексом программ зависит от корректности моделирования потоков внешних сообщений и от достоверности определения длительностей исполнения программ. При подготовке технического задания на комплекс программ следует согласовать с заказчиком внешние условия, в которых должна работать система. Должны быть определены темпы решения периодических задач и допустимое увеличение периодов при повышенной загрузке. Эти условия следует детализировать до уровня, позволяющего определить значения интенсивностей решения различных задач в нормальном режиме работы системы, в режиме допустимой предельной загрузки и в режиме кратковременной аварийной перегрузки и повышенной частоте отказов.

Стохастическая модель внешней обстановки применяется для оценки пропускной способности и надежности функционирования системы при комплексной отладке и испытаниях разработанного комплекса программ. Эта модель, соответствующая нормальным и предельным условиям функционирования комплекса программ, может использоваться как основа для построения вероятностной модели исполнения программ при расчете показателей надежности. Она позволяет оценить средние и предельные значения числа реализаций циклов и вероятности условных переходов в подпрограммах, определяющих маршруты обработки информации. Для определения пропускной способности системы модель используется на двух этапах: в трансформированном вероятностном виде для определения длительностей исполнения программ и в первоначальном - для формирования потоков информации.

Длительность исполнения программ зависит от обрабатываемых исходных данных, в зависимости от набора значений исходных данных осуществляется последовательный выбор ветвей маршрута исполнения программы. Для полного описания длительностей исполнения программ необходимо иметь вероятность каждой комбинации исходных данных и соответствующую ей длительность исполнения программы. После упорядочения значений длительностей получается распределение вероятностей маршрутов в зависимости от их длительностей. Для сложных групп программ трудно определить вероятность каждой комбинации исходных данных. Поэтому на практике ограничиваются средними или наиболее вероятными значениями исходных данных.

Экспериментальное определение длительностей исполнения программ производится на реальной ВС или на ЭВМ при интерпретации исполнения программ. Преимуществом последнего метода является отсутствие специальных программных вставок для запоминания показаний счетчика реального времени. Точность определения длительности исполнения программы зависит от точности используемых значений длительностей исполнения команд.

Приближенные временные характеристики могут быть получены методами аналитического расчета. Однако при аналитическом определении длительностей трудно учесть взаимозависимость направлений условных переходов, выбираемых при реальном исполнении программы, и возможность изменения значений вероятностей условных переходов, происходящего в ходе реального функционирования программы. Вследствие этого при ана-

лизе модели могут учитываться маршруты, не связанные ни с одной из возможных в действительности реализаций. Определение длительностей исполнения программ может производиться автоматизированной системой численного анализа вероятностных моделей комплексов программ. Взвешенная графовая модель программы является основным видом исходной информации для расчета. По модели программы создается массив длительностей исполнения отдельных операторов, которые определяются по кодам операций. Массив вероятностей условных переходов задается разработчиком на основе его представления о динамике работы программ.

Перед проведением статистических расчетов длительностей путем эквивалентных преобразований необходимо устранить циклы в модели и с помощью эквивалентной разметки привести ее к ациклическому виду. При разметке используется неформальная информация, что затрудняет ее непосредственное применение в автоматической системе анализа. Приведение графа программы к ациклическому виду может носить многоэтапный характер, поскольку на практике встречается вложение одних циклов в другие, и их разметка происходит последовательно. Подобным образом могут анализироваться не только автономные подпрограммы, но и их комплексы. Для аналитического получения временных характеристик комплекса программ он должен разрабатываться в соответствии с правилами иерархического структурного проектирования.

Аналитически или экспериментально полученные значения длительностей исполнения групп программ служат исходными данными для определения загрузки вычислительной системы в различных режимах функционирования. В результате устанавливается вероятность отказа в обработке сообщений или решении задач для каждого режима, которая пересчитывается в значения показателей надежности. Окончательные проверки показателей пропускной способности производятся в реальных условиях функционирования комплекса программ. Отказы из-за перегрузки следует пересчитывать с учетом частоты их появления для корректного определения показателей надежности.

Тема 13. Статистические испытания

После завершения отладки комплекс программ предъявляется заказчику для проверки и приемки в эксплуатацию. Он подвергается испытаниям, которые проходят две стадии: испытания главного
конструктора (предварительные) и испытания заказчика при участии разработчиков (совместные). Основная цель испытаний состоит в проверке соответствия разработанных программ требованиям (в том числе по надежности функционирования) технического задания заказчика и другим руководящим документам (ГОСТ, ОСТ и т. д.). В процессе испытаний проверяются достоверность и качество технической документации на комплекс программ и ее пригодность для серийного выпуска изделий и их эксплуатации.

При испытаниях главного конструктора, которые иногда сочетаются с завершением комплексной отладки, производятся, по существу те же проверки, что и на совместных испытаниях, только в меньшем объеме. Эти проверки оформляются документально и являются основанием для предъявления программ заказчику на совместные испытания. Совместные испытания проводятся комиссией заказчика, в которой участвуют главный конструктор разработки и ведущие разработчики. Комиссия при испытаниях руководствуется следующими документами:

- утвержденным заказчиком и согласованным с разработчиком техническим заданием на комплекс программ;
- государственными и отраслевыми стандартами на проектирование и испытания сложных систем и на техническую документацию;
- программой испытаний по всем требованиям технического задания и
- методиками испытаний по каждому разделу требований технического задания.

Программа испытаний является планом проведения серии статистических и детерминированных экспериментов и должна разрабатываться с позиции минимизации объема экспериментов при согласованной с заказчиком достоверности получаемых результатов. Для этого определяются последовательность и объем каждого эксперимента в процессе проведения испытаний для проверки выполнения требований технического задания. Программа испытаний должна содержать следующие разделы:

- объект испытаний, его назначение и перечень основных документов, определивших его разработку;
- цель испытаний с указанием основных требований технического задания, подлежащих проверке, и ограничений на проведение испытаний;
- собственно программу испытаний, содержащую общую проверку состава спроектированного комплекса программ в соответствии с техническим заданием и полнотой технической документации; программу специальных проверок по разделам технического задания и дополнительным требованиям, формализованным отдельными решениями;
- методики испытаний, определяющие понятия проверяемых характеристик, условия их проверки, средства, используемые для испытаний, методики обработки и оценки результатов по каждому разделу программы испытаний.

Сложность комплекса программ приводит к необходимости тщательной формулировки условий и значений параметров, при которых должна производиться проверка по каждому разделу программы испытаний. Должно быть гарантировано качество измерительной аппаратуры и средств автоматизации испытаний. Система автоматизированных испытаний комплексов программ определенного назначения почти всегда уникальна. Универсализация таких систем затруднена разнообразием требований технических заданий, подлежащих проверке. Однако отдельные компоненты имитаторов исходной информации, программ обработки результатов, средств управления системами автоматизации испытаний могут быть унифицированы и могут использоваться в качестве стандартных подпрограмм.

Существующие средства автоматизации применяются:

- для испытаний опытного образца комплекса программ на соответствие требованиям технического задания;
- для контроля на заводе-изготовителе серийных образцов комплекса программ на соответствие утвержденным техническим условиям;
- для контроля функционирования каждого комплекса программ в процессе его эксплуатации и решения задач по управлению реальными объектами или обработке информации.

Перечисленные системы упрощаются по диапазону допустимых проверок, и если системы первых двух типов являются уникальными, то последняя имеется в каждой системе. Одновременно происходит сокращение допустимой длительности проверок и использования контролирующих средств. Каждая последующая система автоматизации проверок должна обеспечивать получение результатов быстрее, чем предыдущая. Средства автоматизации испытаний должны обеспечивать (рис. 1.):

- формирование и варьирование исходных данных в соответствии с программой испытаний;
- исполнение проверяемых программ с регистрацией необходимых для анализа исходных, промежуточных и результирующих данных;

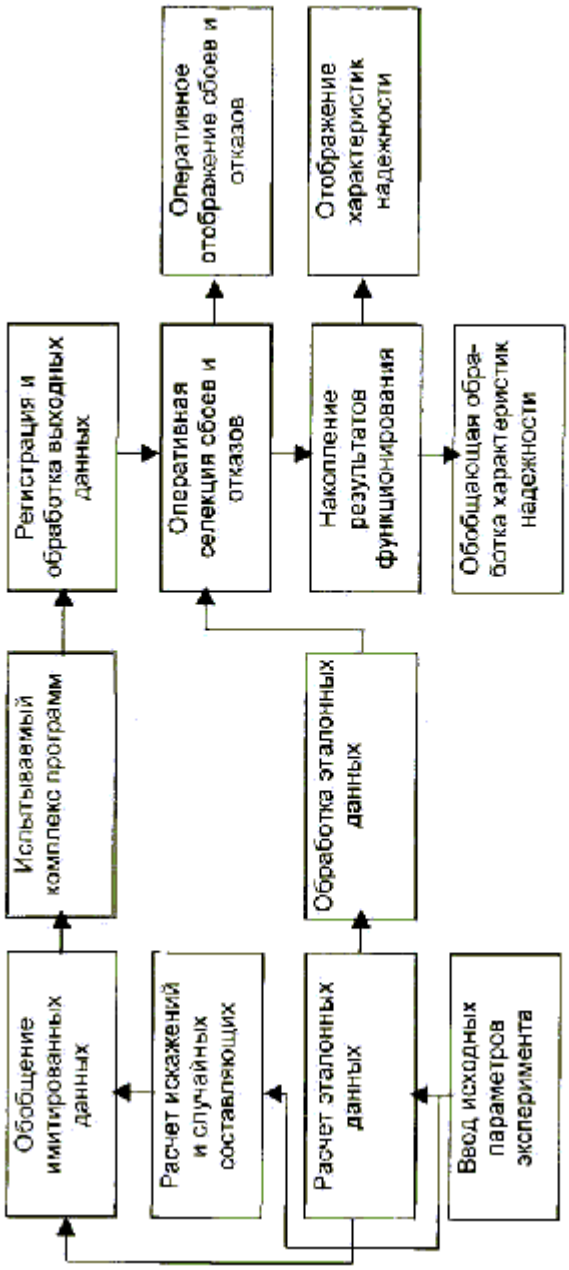


Рис. 1. Схема испытаний комплексов программ на надежность

- обработку результатов функционирования комплекса программ и обобщение частных результатов в виде интегральных характеристик.

Во многих случаях разработка средств автоматизации испытаний сводится к созданию специального комплекса программ, объем которого • может в несколько раз превышать объем комплекса проверяемых программ. Поэтому работы по созданию испытательных комплексов и их эксплуатации следует планировать задолго до начала испытаний. Корректность постановки экспериментов и возможности средств по их проведению существенно определяют достоверность и длительность испытаний. По итогам совместных испытаний составляется акт с описанием результатов проведенных экспериментов и фиксируется степень соответствия результатов испытаний требованиям технического задания. Как уже отмечалось, для сложных комплексов программ трудно на начальных этапах проектирования предусмотреть и корректно сформулировать все требования технического задания. Поэтому при завершении испытаний часто выявляется, что некоторые требования технического задания оказываются невыполненными и иногда принципиально не могут быть выполнены при самом добросовестном отношении разработчика. В этом случае необходим совместный поиск компромиссного решения заказчика и разработчика при завершении испытаний и составлении заключения.

Тема 14. Особенности испытания программных систем на надежность

Изложенные выше общие принципы и этапы испытаний полностью применимы для статистической проверки надежности функционирования комплексов программ. Методы испытаний сводятся к трем основным группам:

- прямые экспериментальные методы определения показателей надежности систем в условиях нормального функционирования;
- форсированные методы испытаний реальных систем на надежность;
- расчетно-экспериментальные методы, при использовании которых ряд исходных данных получается экспериментально, а окончательные показатели надежности рассчитываются.

Прямые экспериментальные методы в ряде случаев трудно использовать из-за больших значений времени наработки на отказ (сотни и тысячи часов). Сложность выявления и регистрации отказов, высокая стоимость экспериментов при длительном функционировании сложных комплексов программ приводят к тому, что на испытаниях объем статистических выборок зарегистрированных отказов оказывается малым. При таких экспериментах трудно гарантировать требуемую представительность выборки исходных данных. Для повышения достоверности испытаний используются данные об обнаруженных ошибках на завершающих этапах отладки программ. Обработка этих данных расширяет объем статистических выборок, сокращает испытания и позволяет прогнозировать показатели надежности комплексов программ более корректно.

Форсированные методы испытаний на надежность программ отличаются от традиционных методов испытаний аппаратуры. Основными факторами, влияющими на надежность комплексов программ, являются исходные данные и их взаимодействие с ошибками программ или сбоями в аппаратуре ВС. Поэтому форсирование испытаний выполняется повышением интенсивности искажений в сообщениях и увеличением загрузки комплекса программ. Планирование форсированных испытаний предусматривает последующий пересчет полученных показателей надежности на условия нормального функционирования. Для этого необходимо изучить зависимости надежности испытываемых программ от интенсивности искажений и от характеристик перегрузки, способы корректного пересчета получаемых показателей на нормальные условия эксплуатации.

Особым видом форсированных испытаний являются специальные проверки средств контроля и восстановления программ, данных и вычислительного процесса. При этом должны имитироваться запланированные экстремальные условия функционирования комплекса программ, при которых стимулируется срабатывание испытываемого средства программного контроля или восстановления. При таких испытаниях основная задача состоит в проверке реализации и качества логики функционирования средств повышения надежности.

Форсированные методы испытаний программ иногда применяются заказчиком без предварительного согласования с разработчиком условий таких испытаний и методики пересчета получаемых показателей на нормальный режим функционирования. В результате показатели надежности оказываются ниже заданных техническим заданием. Поэтому выбранные заказчиком форсированные условия функционирования должны быть включены в состав требований технического задания по дополнительному соглашению. В любом случае при использовании форсированных методов испытаний заказчиком недопустимо произвольное варьирование исходных данных за пределами требований согласованного задания.

Расчетно-экспериментальные методы определения надежности сложной аппаратуры базируются на экспериментальном определении показателей надежности компонент (деталей, микросхем и т. д.) и аналитическом расчете надежности систем, построенных из таких компонент. При анализе надежности программ применение расчетно-экспериментальных методов ограничено. Это обусловлено неоднородностью характеристик надежности основных компонент: программных модулей, групп программ, массивов данных

и т. д. Однако в некоторых случаях расчетным путем можно оценить некоторые характеристики надежности комплексов программ. Если экспериментально определены характеристики искажения массива данных при функционировании комплекса программ, то аналитически можно рассчитать надежность хранения данных при типовых схемах их дублированного хранения и оперативного восстановления при искажениях. Аналитически на основе экспериментальных исходных данных можно оценить изменение наработки на отказ при введении методов оперативного контроля и восстановления. Сочетание экспериментальных и аналитических методов применяется для определения пропускной способности комплекса программ на конкретной ВС и влияния перегрузки на надежность его функционирования.

Следует выделить использование расчетно-экспериментального метода испытаний на надежность при длительном хранении программ на магнитных носителях. Ряд комплексов программ может эксплуатироваться в течение 10-20 лет. Экспериментально установить надежность хранения программ на таком интервале невозможно. Консервируемые комплексы программ имеют эталонную версию магнитных носителей (диски, ленты), которая с высокой вероятностью должна сохраняться в течение этого времени. Для этого готовится несколько копий эталонной версии комплекса программ, которые периодически контролируются, сопоставляются и восстанавливаются. Экспериментально определяется вероятность разрушения программ на одной копии в течение небольшого времени (недели, месяца) между последовательными фазами контроля. По этим данным рассчитывается число необходимых копий и период их контроля и восстановления, гарантирующие заданную вероятность сохранения программ в течение установленного времени "жизни" комплекса.

При проверке аппаратуры применяются определительные и контрольные испытания. Эти же виды испытаний могут использоваться при определении характеристик надежности программ. Определительным испытаниям подвергается опытный образец комплекса программ, для которого перечисленными методами оцениваются основные показатели надежности. В большинстве случаев при этом получают усредненные показатели (среднюю наработку на отказ, среднее время восстановления, коэффициенты готовности и др.). Эти испытания проводятся на ограниченном интервале времени либо до получения заданного числа отказов с полным восстановлением после каждого отказа. В отдельных случаях возможно определение доверительных интервалов для некоторых показателей надежности. Значения доверительных интервалов более полно характеризуют надежность программ и позволяют корректнее сравнивать надежность разных программ или одного и того же комплекса программ на разных этапах разработки. Основным препятствием для получения таких показателей является малый объем экспериментальных данных.

Контрольные испытания применяются для оценки надежности комплексов программ при их тиражировании и серийном выпуске. Считаются известными основные показатели надежности, которым должен соответствовать каждый комплекс программ. Проверка проводится для получения требуемых характеристик и их сравнения с заданными. Для сокращения объема и длительности проверок могут применяться методы последовательного анализа. Устанавливаются значения риска разработчика и риска заказчика и испытания проводятся до получения характеристик надежности, позволяющих принять или забраковать комплекс программ. Изменения характеристик конкретных внешних абонентов и условий эксплуатации даже в пределах требований технического задания могут приводить к отбраковке комплексов программ на контрольных испытаниях надежности серийных образцов, когда опытный образец успешно выдержал определительные испытания. В результате контрольных испытаний могут обнаруживаться ошибки проектирования, требующие доработки всех образцов комплекса программ. Проведение контрольных испытаний связано с технологией сопровождения и модификации программ на интервале жизни каждого программного комплекса. Поэтому управление надежностью и уточнение показателей не прекращается после определительных испытаний опытного образца, а должно проводиться при эксплуатации серийных экземпляров комплексов программ.

Тема 15. Надежность программных комплексов при эксплуатации и сопровождении

После завершения испытаний и сдачи комплекса программ заказчику он может тиражироваться и эксплуатироваться 10-20 лет. В этой части жизненного цикла программ расширяются условия их использования и характеристики исходных данных, вследствие чего могут потребоваться изменения в программах. Кроме того, возможны искажения программ в процессе их хранения и регулярной эксплуатации. Эти факторы способны значительно изменять достигнутые показатели надежности. Достаточно одной, не тщательно проверенной корректировки в программе, для того чтобы снизить характеристики t наработки на отказ всего комплекса на один - два порядка.

Для сохранения и улучшения показателей надежности комплексов программ в процессе длительного сопровождения необходимо четко регламентировать передачу комплексов программ пользователям или ввод их в состав систем управления. Целесообразно накапливать необходимые изменения в программах и вводить их группами, формируя очередную версию комплекса программ с измененными характеристиками. Версии комплекса программ можно разделить на эталонные и пользовательские (или конкретного объекта управления). Эталонные версии развиваются, дорабатываются и модернизируются основными разработчиками комплекса программ или специалистами, выделенными для их сопровождения. Они снабжаются откорректированной технической документацией, полностью соответствующей программам, и точным перечнем всех изменений, введенных в данную версию по сравнению с предыдущей. После оформления очередной эталонной версии она подвергается испытаниям, особенно тщательным в той части программ, которая подверглась изменениям. Однако необходимы также общие проверки работоспособности и сохранности всех программ комплекса, так как некоторые некорректные изменения могут иметь отдаленные трудно предсказуемые последствия, нарушающие работоспособность некорректировавшихся программ.

Пользовательские версии или версии конкретной системы управления формируются из эталонных по инструкциям и правилам, содержащимся в эксплуатационной документации. Допустимые изменения ограничены и локализуются в выделенных компонентах комплекса. Для корректности выполнения изменений они снабжаются методиками проверки и правилами подготовки контролирующих тестов. Относительная легкость изменения отдельных программ и комплектации версий комплекса способствует появлению у пользователей (особенно малоквалифицированных) стремления к самостоятельному "улучшению" комплекса программ за пределами, разрешенными инструкциями для пользовательских версий. Такие изменения должны быть организационно заблокированы правилами внедрения комплексов программ и технически ограничены передачей эксплуатационной документации, минимально необходимой для правильной подготовки пользовательской версии и работы с ней. Целесообразно ограничивать доступ широких пользователей к технологической документации, содержащей подробные сведения о содержании и логике функционирования программ. Такие меры в некоторой степени предотвращают возможность резкого ухудшения показателей надежности и других характеристик комплекса программ из-за неквалифицированного вмешательства пользователей.

Для корректного изменения программ специалистами, ответственными за сопровождение, необходима технологическая документация, более полная, чем эксплуатационная. В технологической документации раскрываются все детали содержания программ и данных, позволяющие новому квалифицированному специалисту полностью понять функционирование программ и корректно их модифицировать. Для каждой очередной эталонной версии регистрируются и документально оформляются все изменения, которые позволяют сформировать комплект технологической документации, полностью адекватный данной эталонной версии. Таким образом, могут сопровождаться 2—4 эталонные версии с разной номенклату-

рой изменений. Более старые версии целесообразно снимать с эксплуатации и сопровождения.

Для сохранения показателей надежности комплекса программ в течение всего жизненного цикла необходимо исключить возможность физического разрушения программ и документации эталонных версий. Для этого применяются методы дублирования и регулярного контроля сохранности магнитных носителей информации. Регистрация и накопление всех изменений, выполненных между оформлением очередных версий, позволяет восстанавливать эталонные версии в аварийных ситуациях не только на базе копий, но и с использованием нескольких предыстории.

Продолжительный жизненный цикл комплексов программ необходимо обеспечить сохранностью средств автоматизации проектирования, с использованием которых создавались эталонные версии. В течение 10-20 лет необходимо сохранять и учитывать версии технологических средств, с использованием которых созданы все сопровождаемые версии комплекса программ, либо принимать специальные меры для обеспечения возможности проверки и корректировки ранних версий при изменении технологических средств.

Большие затраты, необходимые для достижения и проверки высоких значений показателей надежности в сложных комплексах программ, сочетаются с относительной легкостью резкого ухудшения этих показателей при корректировках программ. Отсюда возникает необходимость особенно тщательной проверки влияния на надежность каждого изменения эксплуатируемых программ. Локальные проверки и форсированные испытания на надежность, направленные на контроль конкретных изменений, обычно более экономичны, чем полные повторные испытания всего комплекса программ. Однако некоторые эталонные версии в процессе сопровождения необходимо периодически подвергать полным контрольным испытаниям для корректного подтверждения достигнутых показателей надежности.

Методические рекомендации (указания) к лабораторным занятиям

Лабораторная работа № 1

Определение характеристик простейшего потока

Цель работы: Разработать алгоритмы решений задач по определению характеристик простейшего потока с производственным сценарием.

Указания к работе:

1. Свести условие задач, сформулированных на естественном языке, к формальным символам и обозначениям.
2. Произвести решение задач в этих формальных символах. При этом:
   2.1. Построить решение задачи, если решение известно и условие задачи является достаточным для этого решения, а само решение адекватно заданию.
   2.2. Найти решение, адекватное заданию для случая, когда условие не является достаточным, однако из условия могут быть найдены недостающие данные.
3. Построить алгоритм на основании решения, полученного в п. 2.
4. Написать и отладить программу на базе алгоритма (п. 3).
5. Оформить отчет по работе.

Ход работы:

a) формулировка задачи на естественном языке (текст).
b) Формализация условия задачи (введение обозначений и пояснений к ним).
c) Описание решений в терминах принятых обозначений (в том числе и по пункту 2.2 указаний к работе).

d) Построить блок – схему алгоритма на основе правила и решений.
e) Написать программу решения на алгоритмическом языке по полученной блок – схеме.
f) Привести листинг рабочей программы и результаты решения.

Пример:

a) Интенсивность отказов щеточно-коллекторного узла генератора равна $1{,}1 \cdot 10^{-3}$ 1/час. Определить вероятность того, что через 60000 часов работы откажет ровно 7 щеточно - коллекторных узлов. Поток отказов считать простейшим.

b) $\lambda = 1{,}1 \cdot 10^{-3}$ 1/час; $\tau = 60000$ час.; $K = 7$ – количество отказов; $P(\xi = K)$ – искомая вероятность.

c) В соответствии с формулой Пуассона искомая вероятность равна:

$$P(\xi = K) = \frac{a^K}{K!} e^{-a};$$

$$a = \lambda \cdot \tau;$$

$$P(\xi = K) = \frac{(\lambda \cdot \tau)^K}{K!} e^{-\lambda \cdot \tau}.$$

d) Построить и привести блок – схему алгоритма решения в соответствии с приведенной последовательностью применения формул.

e) Написать программу решения на алгоритмическом языке по полученной блок – схеме с введением данных, приведенных в условии задачи.

$$P(\xi = K) = \frac{(\lambda \cdot \tau)^K}{K!} e^{-\lambda \cdot \tau} = \frac{(1{,}1 \cdot 10^{-3} \cdot 60000)^7}{7!} e^{1{,}1 \cdot 10^{-3} \cdot 60000} \approx 0{,}0054.$$

Размерность в решении выдержана. Действительно. В числителе формулы Пуассона, а также в показателе степени числа *e* получается безразмерная величина: (1/час·час) = 1.

f) Привести листинг рабочей программы и результаты решения.

Вопросы для самоконтроля:

1. Какое состояние ТУ называется работоспособным.
2. Сформулировать определение понятия – отказ.
3. Какой поток называется простейшим.
4. Свойства простейшего потока и их характеристики.
5. Среднее число событий, наступающих в простейшем потоке.

Задание:

Задача 1.

Определить интенсивность отказов некоторого ТУ, если за 500 часов ТУ такого же типа среднее число отказов за это же время равно 1.

Задача 2.

На автоматическую телефонную станцию поступает простейший поток вызовов с интенсивностью, равной 0,8 (вызовов в минуту). Найти вероятность того, что за две минуты: *a*) не придет ни одного вызова; *b*) придет ровно один вызов; *c*) придет хотя бы один вызов.

Задача 3.

Поток вызовов на АТС – пуассоновский нестационарный с интенсивностью $\lambda(t)$, зависящей от времени На участке времени от 0 час. до 6 час. 40 мин. интенсивность $\lambda(t)$ возрастает по линейному закону:

$$\lambda(t) = bt + c,$$

причем к 0 час. она равна 0,2 вызова в минуту, а в 6 часов 40 минут – 0,4 вызова в минуту, Найти вероятность того, что за 10 минут от 3 часов 15 минут до 3 часов 25 минут придет не менее трех вызовов.

Замечание:

Если интенсивность простейшего потока λ не стационарна (не постоянна), а зависит от времени $\lambda = \lambda(t)$, то вероятность попадания ровно m событий на участок длины τ, начинающийся в точке t_0 и кончающийся в точке $t_0 + \tau$, имеет также распределение Пуассона:

$$P(\xi = m) = \frac{a^m}{m!} e^{-a} \ (m = 0, 1, 2, \ldots), \text{ где } a = \int_{t_0}^{t_0+\tau} \lambda(t)dt.$$

Лабораторная работа № 2

Определение основных характеристик надежности невосстанавливаемых элементов информационных систем

Цель работы: Разработать алгоритмы решений задач по расчету основных характеристик надежности невосстанавливаемых элементов информационных систем.

Указания к работе:

1. Свести условие задач, сформулированных на естественном языке, к формальным символам и обозначениям.
2. Произвести решение задач в этих формальных символах. При этом:
   2.1. Построить решение задачи, если решение известно и условие задачи является достаточным для этого решения, а само решение адекватно заданию.
   2.2. Найти решение, адекватное заданию для случая, когда условие не является достаточным, однако из условия могут быть найдены недостающие данные.
3. Построить алгоритм на основании решения, полученного в п. 2.
4. Написать и отладить программу на базе алгоритма (п. 3).
5. Оформить отчет по образцу:

Ход работы:

a) формулировка задачи на естественном языке (текст).
b) Формализация условия задачи (введение обозначений и пояснений к ним).
c) Описание решений в терминах принятых обозначений (в том числе и по пункту 2.2 указаний к работе).
d) Построить блок – схему алгоритма на основе правила и решений.
e) Написать программу решения на алгоритмическом языке по полученной блок – схеме.
f) Привести листинг рабочей программы и результаты решения.

Пример:

a) На эксплуатацию одновременно было поставлено 300 однотипных электронных цифровых приборов. Известно, что через 100 часов отказало 27 приборов, а еще через следующие 100 часов отказало еще 34. определить статистические вероятности безотказной работы и отказов через 100 и 200 часов, условную вероятность безотказной работы через 200 часов, при условии, что приборы проработали уже 100 часов, а также определить статистические значения плотности вероятности отказов через 100 и 200 часов.

b) $p*(t)$ - статистическая вероятность безотказной работы через t часов; $q(t)$ – статистическая вероятность отказов через t часов; $p*(t_i / t_j)$ - условная вероятность безотказной работы через t_i часов при условии, что приборы проработали безотказно уже t_j часов; $f*(t)$ - статистическое значение плотности вероятности отказов через t часов; N – число поставленных на эксплуатацию приборов; $\Delta n(t)$ – число, отказавших ко

времени t приборов; $N(t)$ – число приборов, оставшихся работоспособными к моменту времени t.

c) Задача решается в соответствии со следующими выражениями:

$$p*(t)=\frac{N-N(t)}{N};\ q(t)=\frac{n(t)}{N};\ p*(t_i/t_j)=\frac{p(t_i)}{p(t_j)},\ (t_i\ge t_j);\ f*(t)=\frac{\Delta n(t)}{N(t)}.$$

d) Построить и привести блок – схему алгоритма решения в соответствии с приведенной последовательностью применения формул:

$$p*(t=100)=\frac{N-N(t=100)}{N}=\frac{300-27}{300}=0{,}91;$$

$$p*(t=200)=\frac{N-N(t=200)}{N}=\frac{300-27-34}{300}=0{,}8;$$

$$q(t=100)=\frac{n(t=100)}{N}=\frac{27}{300}=0{,}09;$$

$$q(t=200)=1-p(t=200)=1-0{,}8=0{,}2;$$

$$f*(t=200)=\frac{\Delta n(t=200)}{N(t=200)}=\frac{34}{300\cdot 200}\approx 5{,}7\cdot 10^{-4}(1/час).$$

Размерность в решении выдержана.

e) Привести листинг рабочей программы и результаты решения.

Вопросы для самоконтроля:

1. В чем заключается принцип резервирования.
2. Что является основной характеристикой резервирования.
3. Основные особенности общего резервирования.
4. Основные особенности раздельного резервирования.
5. Плотность вероятности отказов.
6. Какие характерные участки имеет кривая интенсивности отказов невосстанавливаемых технических устройств.
7. Среднее время безотказной работы.
8. Среднее статистическое время безотказной работы.
9. Какова зависимость между $f(t)$ и $p(t)$, $p(t)$ и $\lambda(t)$, $f(t)$ и $\lambda(t)$, T и $\lambda(t)$.
10. Как зависит $p(t)$, $f(t)$ и T от $\lambda(t)$ при $\lambda(t) = \lambda = const$.
11. Основные расчетные соотношения между показателями надежности для случая, когда $t << T$.

Задание:

Задача 1.

В течении 2000 часов наблюдали за 480 видеоадаптерами. Определить статистическую вероятность безотказной работы этих устройств, если в течении указанного срока зарегистрировано 52 отказа, причем 12 из них произошли в первые 400 часов.

Задача 2.

Найти статистическое распределение плотности вероятности отказов и статистическую кривую интенсивности отказов 500 блоков питания для ПЭВМ, испытанных на эксплуатационном режиме до полного отказа всех блоков. Отказы блоков питания подсчитаны через каждые 100 часов, результаты сведены в таблицу.

№ п/п	Δt_i в часах	Δn_i	№ п/п	Δt_i в часах	Δn_i
1	0 – 100	9	16	1500 – 1600	3

№ п/п	Δt_i в часах	Δn_i	№ п/п	Δt_i в часах	Δn_i
2	100 – 200	13	17	1600 – 1700	5
3	200 – 300	16	18	1700 – 1800	11
4	300 – 400	15	19	1800 – 1900	18
5	400 – 500	11	20	1900 – 2000	27
6	500 – 600	6	21	2000 – 2100	38
7	600 – 700	3	22	2100 – 2200	48
8	700 – 800	2	23	2200 – 2300	55
9	800 – 900	4	24	2300 – 2400	52
10	900 – 1000	3	25	2400 – 2500	46
11	1000 – 1100	2	26	2500 – 2600	42
12	1100 – 1200	5	27	2600 – 2700	31
13	1200 – 1300	3	28	2700 – 2800	15
14	1300 – 1400	3	29	2800 – 2900	7
15	1400 – 1500	4	30	2900 – 3000	3

Задача 3.

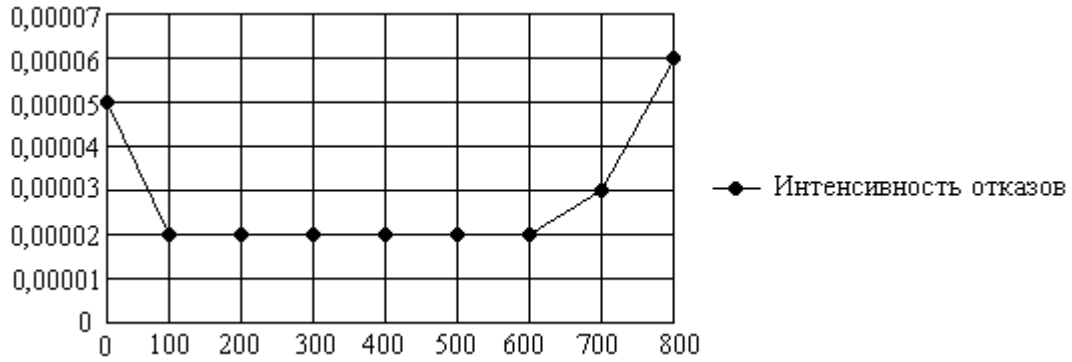
На эксплуатацию было поставлено $N = 67$ однотипных ТУ, за которыми велось наблюдение в течение 400 часов. Определить статистическую интенсивность отказов $\lambda^*(t)$ и статистическую вероятность безотказной работы $p^*(t)$ в точках наблюдения, если известно, что каждые 50 часов отказывало по 2 устройства.

Задача 4.

Вероятность безотказной работы проволочных сопротивлений определяется законом, выраженным формулой $p(t) = e^{-0,0098t}$. На эксплуатацию поставлено 500 сопротивлений. Определить среднее количество отказов через 50 часов наработки.

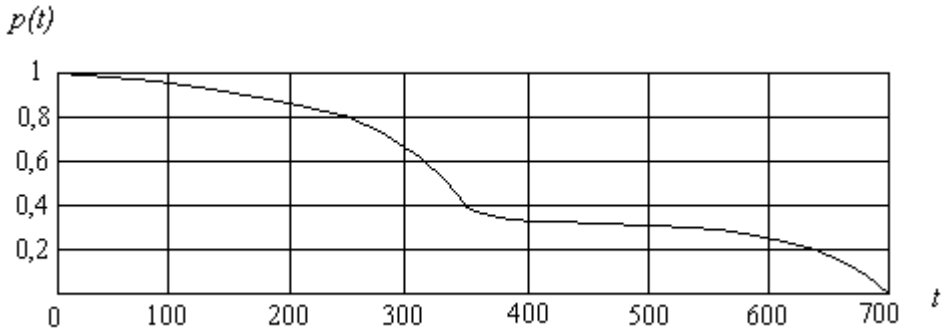
Задача 5.

На основании графика интенсивности отказов ТУ построить график вероятности его безотказной работы.



Задача 6.

График вероятности безотказной работы керамического конденсатора показан на рисунке. На эксплуатацию было поставлено 180 конденсаторов. определить среднее количество отказавших конденсаторов через 300 часов и интенсивность безотказной работы через 600 часов.



Замечание:

Известно, что $\lambda_i^* = \frac{\Delta n_i}{N_i \Delta t_i}$. Индекс «*i*» представляет собой указатель интервала. Для которого рассчитывается интенсивность отказа. Для расчета по приведенной формуле необходимо знать величины Δn_i, N_i, Δt_i. Обычно из условия задачи известны количество отказавших ТУ Δn_i и величина интервала времени Δt_i. Величина N_i по своей сути представляет собой математическое ожидание числа безотказно проработавших ТУ в течение *i* – го интервала времени. Наиболее очевидной статистической оценкой этой величины могло бы стать среднеарифметическое $N_i^* = \frac{\sum_{i=1}^{8}(N - \Delta n_i)}{i}$. Однако существует оценка, которая с большей точностью соответствует значению математического ожидания $N_i = N - \sum_{k=1}^{i-1} \Delta n_k - \frac{\Delta n_i}{2}$.

Лабораторная работа № 3

Определение показателей надежности в период процесса эксплуатации систем

Цель работы: Разработать алгоритмы определения показателей надежности в период процесса эксплуатации информационных систем.

Указания к работе:

1. Свести условие задач, сформулированных на естественном языке, к формальным символам и обозначениям.
2. Произвести решение задач в этих формальных символах. При этом:
   2.1. Построить решение задачи, если решение известно и условие задачи является достаточным для этого решения, а само решение адекватно заданию.
   2.2. Найти решение, адекватное заданию для случая, когда условие не является достаточным, однако из условия могут быть найдены недостающие данные.
3. Построить алгоритм на основании решения, полученного в п. 2.
4. Написать и отладить программу на базе алгоритма (п. 3).
5. Оформить отчет по образцу:

Ход работы:

- *a)* формулировка задачи на естественном языке (текст).
- *b)* Формализация условия задачи (введение обозначений и пояснений к ним).
- *c)* Описание решений в терминах принятых обозначений (в том числе и по пункту 2.2 указаний к работе).
- *d)* Построить блок – схему алгоритма на основе правила и решений.
- *e)* Написать программу решения на алгоритмическом языке по полученной блок – схеме.

f) Привести листинг рабочей программы и результаты решения.

Пример:

a) Система состоит из3 блоков. У всех блоков системы период нормальной эксплуатации начинается одновременно. Эксплуатация же самой системы начинается с момента начала периода нормальной эксплуатации. Интенсивности отказов блоков системы равны: $\lambda_1 = 3{,}03\cdot10^{-3}$ час$^{-1}$, $\lambda_2 = 3{,}13\cdot10^{-3}$ час$^{-1}$, $\lambda_3 = 2{,}97\cdot10^{-3}$ час$^{-1}$. Для всей системы характерно соотношение $\frac{t_p}{T} = 0{,}58$. Определить ресурс системы.

b) λ_1, λ_2, λ_3 – интенсивности отказов первого, второго, третьего блоков системы соответственно; T – среднее время безотказной работы системы; t_p – ресурс системы.

c) Задача решается в соответствии со следующими выражениями. Ресурс системы t_p будет рассчитываться по среднему времени безотказной работы системы. Средним временем безотказной работы системы будет считаться среднее время безотказной работы наименее Надежного блока. Наименее надежным блоком считается тот, интенсивность отказов у которого максимальна. По условию задачи это второй блок, т.к. $\lambda_2 > \lambda_1$, $\lambda_2 > \lambda_3$. Поэтому: $T = T_2 = \frac{1}{\lambda_2}$. Ресурс системы будет равен: $t_p = 0{,}58T$.

d) Построить и привести блок – схему алгоритма решения в соответствии с приведенной последовательностью применения формул:

$$T = T_2 = \frac{1}{\lambda_2} = \frac{1}{3{,}13\cdot10^{-3}} \approx 319{,}5(ч);$$

$t_p = 0{,}58T = 0{,}58 \cdot 319{,}5 = 185(ч)$.

Размерность в решении выдержана.

e) Привести листинг рабочей программы и результаты решения.

Вопросы для самоконтроля:

1. Какие основные виды интенсивностей отказов могут иметь технические устройства.
2. Дайте определение календарному сроку службы.
3. Что такое ресурс, чем он отличается от календарного срока службы.
4. Виды ресурса.
5. Что такое средний срок сохраняемости.
6. Какой характер имеет поведение интенсивностей отказов в нормальный период эксплуатации и в период износа и старения.
7. Каким законом может быть описано распределение времени безотказной работы в период износа и старения.
8. Как определяется общая вероятность безотказной работы технического устройства с учетом внезапных и постепенных отказов.

Задание:

Задача 1.

Испытания наблюдения велись за 1000 вентиляторами для ПЭВМ. Испытания проводятся в режиме нормальной эксплуатации до полного отказа всех ламп. Число отказов вентиляторов подсчитывалось в каждом интервале времени Δt = 500 часов. Результаты испытаний занесены в таблицу. Пользуясь данными таблицы построить график функции интенсивности отказов от времени и провести анализ этого графика.

Интервал времени Δt, час	Кол-во отказов	Интервал времени Δt, час	Кол-во отказов	Интервал времени Δt, час	Кол-во отказов
0 – 500	60	3000 – 3500	40	6000 – 6500	20
500 – 1000	200	3500 – 4000	30	6500 – 7000	20
1000 – 1500	197	4000 – 4500	20	7000 – 7500	20

1500 – 2000	150	4500 – 5000	20	7500 – 8000	30

Интервал времени Δt, час	Кол-во отказов	Интервал времени Δt, час	Кол-во отказов	Интервал времени Δt, час	Кол-во отказов
2000 – 2500	128	5000 – 5500	20	8000 – 8500	20
2500 – 3000	72	5500 – 6000	20	8500 – 9000	10

Задача 2.

Испытания на надежность в номинальном рабочем режиме подверглись 900 лазерных головок дисководов вплоть до отказа всех этих приборов. Лазерные головки дисководов относятся к элементам стареющего типа, их отказы считаются независимыми, имеют случайный характер и в ходе испытания подсчитывались через каждые 200 часов наработки. Результаты испытаний приведены в таблице. Определить среднюю наработку головки дисковода до отказа, построить график вероятности безотказной работы $p^*(t)$, график вероятности отказа $q^*(t)$, график плотности вероятности отказов $f^*(t)$, график интенсивности отказов $\lambda^*(t)$ в собственных осях координат и провести анализ этих графиков. Определить общую вероятность безотказной работы лазерных головок дисководов с учетом внезапных и постепенных отказов к моменту времени t = 5800 часов от начала испытаний.

№ п/п	Δt_i в часах	Δn_i	№ п/п	Δt_i в часах	Δn_i
1	0 – 200	30	16	3000 – 3200	11
2	200 – 400	36	17	3200 – 3400	8
3	400 – 600	40	18	3400 – 3600	12
4	600 – 800	42	19	3600 – 3800	12
5	800 – 1000	35	20	3800 – 4000	21
6	1000 – 1200	27	21	4000 – 4200	39
7	1200 – 1400	18	22	4200 – 4400	52
8	1400 – 1600	16	23	4400 – 4600	61
9	1600 – 1800	15	24	4600 – 4800	74
10	1800 – 2000	13	25	4800 – 5000	82
11	2000 – 2200	15	26	5000 – 5200	80
12	2200 – 2400	12	27	5200 – 5400	56
13	2400 – 2600	11	28	5400 – 5600	25
14	2600 – 2800	10	29	5600 – 5800	23
15	2800 – 3000	9	30	5800 – 6000	15

Лабораторная работа № 4

Разработка алгоритмов расчета структурных схем надежности

Цель работы: Разработать алгоритмы расчета структурных схем надежности.

Указания к работе:

1. Свести условие задач, сформулированных на естественном языке, к формальным символам и обозначениям.
2. Произвести решение задач в этих формальных символах. При этом:
   2.1. Построить решение задачи, если решение известно и условие задачи является достаточным для этого решения, а само решение адекватно заданию.
   2.2. Найти решение, адекватное заданию для случая, когда условие не является достаточным, однако из условия могут быть найдены недостающие данные.

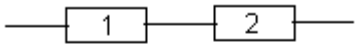
3. Построить алгоритм на основании решения, полученного в п. 2.
4. Написать и отладить программу на базе алгоритма (п. 3).
5. Оформить отчет по образцу:

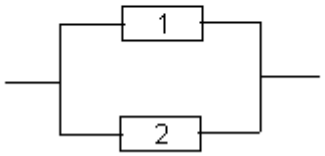
Ход работы:

a) формулировка задачи на естественном языке (текст).
b) Формализация условия задачи (введение обозначений и пояснений к ним).
c) Описание решений в терминах принятых обозначений (в том числе и по пункту 2.2 указаний к работе).
d) Построить блок – схему алгоритма на основе правила и решений.
e) Написать программу решения на алгоритмическом языке по полученной блок – схеме.
f) Привести листинг рабочей программы и результаты решения.

Пример:

a) Даны структурные схемы надежности:





Определить вероятность безотказной работы изображенных структурных схем надежности для случаев:

a) вероятности безотказной работы не равны между собой: $p_1 = 0{,}99$; $p_2 = 0{,}97$.
b) вероятности безотказной работы равны между собой: $p_1 = p_2 = 0{,}98$.

b) p_1, p_2 – вероятности безотказной работы соответственно первого и второго элементов представленных структурных схем надежности, p_c - вероятность безотказной работы структурной схемы надежности.
c) Первый рисунок изображает последовательное соединение в структурной схеме надежности, состоящей из двух элементов, второй - параллельное соединение двух элементов в структурной схеме надежности. Поэтому:

- для последовательной структурной схемы надежности 1*a*) при равенстве вероятностей безотказной работы; 1*b*) при неравенстве вероятностей безотказной работы имеют место выражения

1*a*) $p_c = p \cdot p = p^2$; 1*b*) $p_c = p_1 \cdot p_2$.

- для параллельной структурной схемы надежности 2*a*) при равенстве вероятностей безотказной работы; 2*b*) при неравенстве вероятностей безотказной работы имеют место выражения

2*a*) $p_c = 1 - (1 - p)^2$; 2*b*) $p_c = 1 - q_c = 1 - q_1 \cdot q_2$.

d) Решить задачу, построить и привести блок – схему алгоритма решения в соответствии с приведенной последовательностью применения формул:

1*a*) $p_c = p \cdot p = p^2 = 0{,}98 \cdot 0{,}98 = 0{,}9604$; 1*b*) $p_c = p_1 \cdot p_2 = 0{,}99 \cdot 0{,}97 = 0{,}9603$.
2*a*) $p_c = 1 - (1 - p)^2 = 0{,}9996$; 2*b*) $p_c = 1 - q_c = 1 - q_1 \cdot q_2 = 0{,}9997$.

e) Привести листинг рабочей программы и результаты решения.

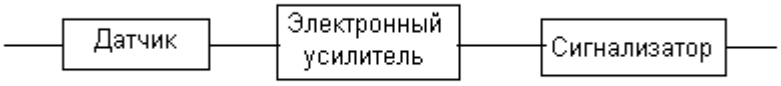
Вопросы для самоконтроля:

1. Что такое структурная схема надежности и чем она отличается от принципиальной схемы ТУ.
2. Что такое структурная схема надежности с последовательным соединением элементов.
3. Что такое структурная схема надежности с параллельным соединением элементов.
4. Надежность при структурной схеме с последовательным соединением элементов
5. Надежность при структурной схеме с параллельным соединением элементов.
6. Что такое сложная произвольная структурная схема надежности.
7. Надежность при произвольной структурной схеме.

Задание:

Задача 1.

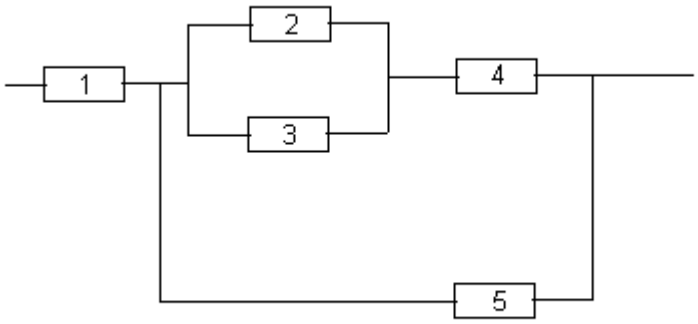
Система сигнализации состоит из трех блоков: датчика, электронного усилителя и сигнализатора. Известно, что надежность датчика выше надежности электронного усилителя, а наименее надежным агрегатом в системе является сигнализатор.



Определить вероятность безотказной работы каждого элемента системы и системы сигнализации в целом, если известно, что $q_1 = 0{,}05$; $q_2 = 0{,}001$; $q_3 = 0{,}01$.

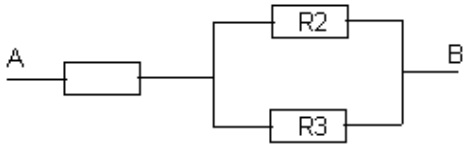
Задача 2.

Определить вероятность безотказной работы системы, структурная схема надежности которой изображена на рисунке. Вероятности безотказной работы элементов равны:
$p_1 = p$; $p_2 = 0{,}5p_1$; $p_3 = p_1$; $p_4 = 0{,}75p_3$; $p_5 = p_4^2$; $p = 0{,}998$.

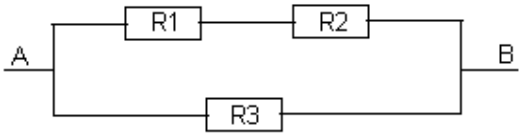


Задача 3.

Определить вероятность прохождения сигнала на участке АВ электрической сети, если вероятность исправного состояния резисторов R1, R2, R3 за время прохождения сигнала соответственно равны 0,6; 0,8; 0,9.



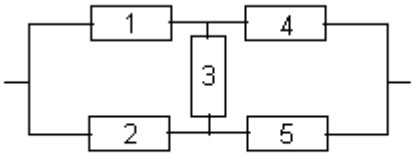
a)



b)

Задача 4.

На рисунке приведена структурная схема надежности типа «мостик». Вероятности безотказной работы элементов системы равны: $p_1 = 0{,}98$; $p_2 = 0{,}98$; $p_3 = 0{,}99$; $p_4 = 0{,}97$; $p_5 = 0{,}98$. Определить вероятность безотказной работы системы.



Лабораторная работа № 5

Разработка алгоритмов расчета структурных схем надежности

Цель работы: Разработать алгоритмы расчета надежности при резервировании.

Указания к работе:

1. Свести условие задач, сформулированных на естественном языке, к формальным символам и обозначениям.
2. Произвести решение задач в этих формальных символах. При этом:
   2.1. Построить решение задачи, если решение известно и условие задачи является достаточным для этого решения, а само решение адекватно заданию.
   2.2. Найти решение, адекватное заданию для случая, когда условие не является достаточным, однако из условия могут быть найдены недостающие данные.
3. Построить алгоритм на основании решения, полученного в п. 2.
4. Написать и отладить программу на базе алгоритма (п. 3).
5. Оформить отчет по образцу:

Ход работы:

a) формулировка задачи на естественном языке (текст).
b) Формализация условия задачи (введение обозначений и пояснений к ним).
c) Описание решений в терминах принятых обозначений (в том числе и по пункту 2.2 указаний к работе).
d) Построить блок – схему алгоритма на основе правила и решений.
e) Написать программу решения на алгоритмическом языке по полученной блок – схеме.
f) Привести листинг рабочей программы и результаты решения.

Пример:

a) Техническая система состоит из одного элемента с вероятностью отказа $q = 0{,}0007$. Она однократно резервируется элементом подобного типа с вероятностью безотказной работы $p_p = 0{,}99$. Определить вероятность безотказной работы системы после резервирования и сделать выводы.
b) $q = 0{,}0007$ – вероятность отказа основной системы; p – вероятность безотказной работы основной системы; $p_p = 0{,}99$ – вероятность безотказной работы резервирующего элемента; $q_{ср}$ – вероятность отказа системы после резервирования; $p_{ср}$ – вероятность безотказной работы системы после резервирования.
c) Так как система состоит из одного элемента, то ее однократное резервирование нельзя отнести к общему или раздельному типу резервирования. Основной элемент вместе с резервирующим составят структурную схему надежности с параллельным соединением элементов. Поэтому искомая вероятность ищется следующим образом:
$p_{ср} = 1 - q_{ср}$;

$q_{ср} = q \cdot q_p$;
$q_p = 1 - p_p$.

d) Решить задачу, построить и привести блок-схему алгоритма решения в соответствии с приведенной последовательностью применения формул. Решение задачи начинается с последнего выражения предыдущего пункта в связи с тем, что именно такая последовательность устанавливает логику решения задачи – определения неизвестных по мере их использования:
$p_{ср} = 1 - q_{ср} = 1 - 0{,}000007 = 0{,}999993$;
$q_{ср} = q \cdot q_p = 0{,}007 \cdot 0{,}001 = 0{,}000007$;
$q_p = 1 - p_p = 1 - 0{,}99 = 0{,}001$.
Из полученного результата видно, что система после резервирования даже менее надежным элементом, чем элемент основной системы стала более надежной, а эффективность резервирования равна

$$R = \frac{q}{q_{ср}} = \frac{0{,}007}{0{,}000007} = 1000.$$

Далее строится подробная блок-схема алгоритма решения задачи.

e) Приводится листинг решения задачи.

Вопросы для самоконтроля:

1. В чем заключается принцип резервирования.
2. Что является основной характеристикой резервирования.
3. Основные особенности общего резервирования.
4. Основные особенности раздельного резервирования.
5. Что такое эффективность резервирования.
6. Расчет надежности при общем резервировании
7. Расчет надежности при раздельном резервировании.
8. Как определить необходимое количество резервных элементов при общем резервировании.
9. Как определить необходимое количество резервных элементов при раздельном резервировании.
10. Что такое эффективность при общем и раздельном резервировании.
11. Дать сравнительную оценку общего и раздельного резервирования.
12. В чем заключается особенности резервирования электрических схем.
13. Что такое каноническое уравнение резервированной системы элементов.

Задание:

Задача 1.

Система состоит из трех равнонадежных элементов с вероятностью безотказной работы $p_1 = p_2 = p_3 = 0{,}9$ каждого элемента. Элементы объединены в структурную схему с последовательным соединением. Определить вероятность безотказной работы системы без резервирования, с общим однократным резервированием; с раздельным однократным резервированием. Определить эффективность общего и раздельного резервирования, дать сравнительную оценку общего и раздельного резервирования. При решении задачи учесть, что резервирование осуществляется элементами, аналогичными по надежности элементам основной системы.

Задача 2.

Решить предыдущую задачу, но для вероятностей безотказной работы элементов основной системы, равных соответственно:
$p_1 = 0{,}9$; $p_2 = 0{,}86$; $p_3 = 0{,}92$.

Задача 3.

Определить необходимое число резервных элементов, если задана вероятность отказа основного элемента, равная q = 0,1. Допустимая вероятность резервированной системы должна быть равна Q_p = 0,0001.

Лабораторная работа № 6

Освоение алгоритмов диагноза и прогноза

Цель работы: Разработать алгоритмы диагностирования состояния технических систем и их прогноза.

Указания к работе:

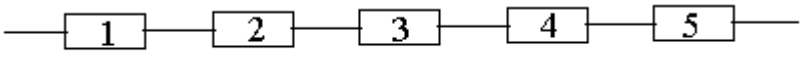
1. Свести условие задач, сформулированных на естественном языке, к формальным символам и обозначениям.
2. Произвести решение задач в этих формальных символах. При этом:
   2.1. Построить решение задачи, если решение известно и условие задачи является достаточным для этого решения, а само решение адекватно заданию.
   2.2. Найти решение, адекватное заданию для случая, когда условие не является достаточным, однако из условия могут быть найдены недостающие данные.
3. Построить алгоритм на основании решения, полученного в п. 2.
4. Написать и отладить программу на базе алгоритма (п. 3).
5. Оформить отчет по образцу:

Ход работы:

a) формулировка задачи на естественном языке (текст).
b) Формализация условия задачи (введение обозначений и пояснений к ним).
c) Описание решений в терминах принятых обозначений (в том числе и по пункту 2.2 указаний к работе).
d) Построить блок – схему алгоритма на основе правила и решений.
e) Написать программу решения на алгоритмическом языке по полученной блок – схеме.
f) Привести листинг рабочей программы и результаты решения.

Пример:

a) Построить алгоритм диагноза состояний системы последовательного типа, состоящей из пяти элементов, используя метод половинного разбиения (МПР).
b) $y_i(t)$ – измеренное значение параметра; y_i – признак параметра; j = 1, 2, …, 5.



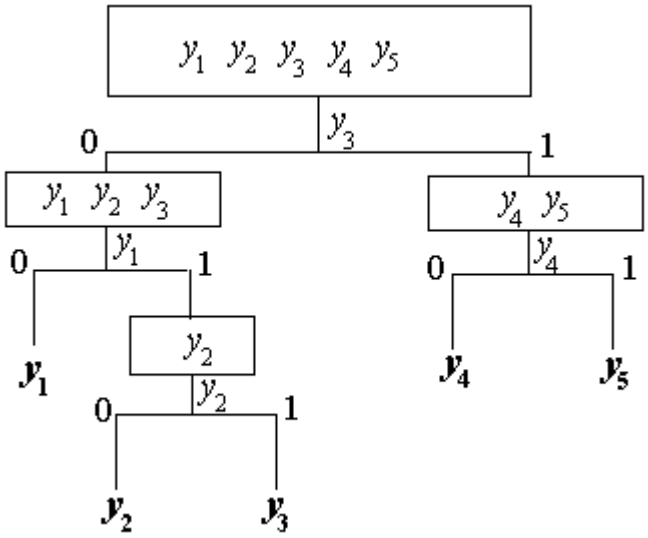
c) *Метод половинного разбиения*: система состоит из нечетного числа элементов, поэтому первую проверку в МПР можно делать после второго или после третьего элемента. Пусть это будет третий элемент. Будем считать:

$$y_i = \begin{cases} 0, y_i(t) \in y_{idop}, \\ 1, y_i(t) \notin y_{idop}. \end{cases}$$

где y_{idop} - область допустимых значений параметра y_i , а i = 1, 2, 3, 4, 5.

Алгоритм поиска места отказа методом половинного разбиения, при условии, что система вообще отказала, т.е. y_5 = 0, будет выглядеть следующим образом: если y_3 = 0, то проверяется левая от третьего элемента ветвь путем деления ее пополам (т.к. в ней снова нечетное число элементов, то для проверки можно взять любой из элементов, находящихся слева от третьего) и выбора, например, первого элемента; если y_1 = 0, то отказал

первый элемент; $y_1 = 1$, то проверяется второй элемент; если $y_2 = 0$ – то отказал второй элемент; если $y_2 = 1$, то отказал третий элемент; если $y_3 = 1$, то проверяют правую от третьего элемента ветвь, делят ее пополам и проверяют четвертый элемент; если $y_4 = 0$, то отказал четвертый элемент; если $y_4 = 1$, то отказал пятый элемент.



Т.о., как видно из графа, построенного по алгоритму поиска отказов, отказ первого элемента распознается за два измерения ($y_3 = 0$, $y_1 = 0$), отказ второго элемента – за три измерения ($y_3 = 0$, $y_1 = 1$, $y_2 = 0$), отказ третьего элемента – за три измерения ($y_3 = 0$, $y_1 = 1$, $y_2 = 1$), отказ четвертого – за два измерения ($y_3 = 1$, $y_4 = 0$) и отказ пятого – за два измерения ($y_3 = 1$, $y_4 = 1$).

Задание:

Задача 1.

Построить алгоритм диагноза состояний системы последовательного типа, состоящей из 11 элементов, используя метод половинного разбиения.

Методические указания для самостоятельной работы студентов

Тестовые задания

20 заданий
время тестирования – 40 минут

ВАРИАНТ 1

Инструкция: все задания имеют одну и ту же форму – с выбором одного ответа из четырех предложенных

1. Формула $P(t) = P(T \geq t)$ описывает…
   1) вероятность отказа
   2) вероятность безотказной работы
   3) интенсивность отказов
   4) коэффициент готовности системы

2. Формула $\lambda(t) = -\frac{d \ln P(t)}{dt}$ описывает…
   1) вероятность отказа
   2) вероятность безотказной работы
   3) интенсивность отказов
   4) коэффициент готовности системы

3. Основные понятия надежности
   1) сложным комплексам программ для информа-

практически не применимы к…

ционно-справочных систем
2) системам автоматизации обработки информации, которые функционируют вне реального времени
3) программам, разрабатываемым для решения инженерных и научно-исследовательских задач
4) комплексам программ автоматического или автоматизированного управления

4. Структуры надежностно - функциональных схем включают в себя…

1) измерительные каналы
2) инерционность объекта
3) последовательную, параллельную, мажоритарную
4) расчетно - экспериментальную

5. Надежность сложных программных комплексов определяется…

1) надежностью компонент и ошибками в конструкции, допущенными при проектировании или изготовлении
2) качеством тестирования
3) условиями тиражирования
4) уровнем развития пользователя

6.Надежность при структурной схеме с последовательным соединением элементов определяется формулой…

1) $P_i(t) = \sum_{j=0}^{m-1} A_{jn}$

2) $P_i(t) = 1 - \prod_{j=1}^{n} \left[1 - P_{ij}(t)\right]$

3) $P_i(t) = 1 - \prod_{j=1}^{n} P_{ij}(t)$

4) $P_i(t) = \prod_{j=1}^{n} P_{ij}(t)$

7. По полноте мероприятий, которые необходимо проводить для восстановления, различаются отказы…

1) восстанавливаемые и невосстанавливаемые
2) устойчивые и перемежающиеся
3) сбои и устойчивые отказы
4) устойчивые, самоустраняющиеся, перемежающиеся

8. Модели надежности программного обеспечения подразделяются на…

1) эмпирические и теоретические
2) динамические и статистические
3) непрерывные и дискретные
4) непрерывные и ступенчатые

9. Плотность вероятности момента отказа распределения Рэлея имеет вид…

1) $f(t) = \lambda\delta t^{\delta-1} \cdot e^{-\left(\lambda t^{\delta}\right)}$

2) $f(t) = \lambda \cdot e^{-(\lambda t)}$

3) $f(t) = \frac{t}{\delta^2} e^{\left(-\frac{t^2}{2\delta^2}\right)}$

4) $f(t) = \frac{1}{\sigma\sqrt{2\pi}} \cdot e^{\left(-\frac{(t-a)^2}{2\sigma^2}\right)}$

10. Предупреждение ошибок в программном обеспечении достигается за счет…

1) иерархической структуры и независимости элементов программного обеспечения
2) временной избыточности
3) изоляции ошибок
4) методов борьбы со сложностью, достижения точности при переводе информации, улучшения обмена информацией, немедленного обнаружения и устранения ошибок

11. Обнаружение ошибок в программном обеспечении достигается за счет…

1) динамической избыточности
2) временной, информационной и программной избыточности
3) изоляции ошибок
4) достижения большей точности при переводе информации

12. Временная избыточность достигается за счет…

1) использования части производительности ЭВМ
2) взаимного недоверия элементов системы, немедленного обнаружения и регистрации ошибок, дублирование выполнения одинаковых функций, контроля и восстановления данных
3) дублирования части данных
4) достижения большей точности при переводе информации

13. Обеспечение устойчивости к ошибкам достигается за счет…

1) динамической избыточности, обработки ошибок, сокращенного обслуживания, изоляции ошибок
2) иерархической структуры и независимости элементов
3) дублирования части данных
4) использования части производительности ЭВМ

14. Методы проектирования надежного программного обеспечения включают в себя…

1) прямые экспериментальные
2) предупреждение ошибок, обнаружение ошибок, обеспечение устойчивости к ошибкам
3) расчетно -экспериментальные
4) методы борьбы со сложностью

15. Экспоненциальная модель является…

1) статической моделью
2) ступенчатой моделью

3) непрерывной динамической моделью
4) частным случаем распределения Рэлея

16. Модель Вейбулла является…

1) статической моделью
2) ступенчатой динамической моделью
3) непрерывной динамической моделью
4) частным случаем распределения Рэлея

17. Процесс выполнения программы (или части программы) с намерением (или целью) найти ошибки есть…

1) верификация
2) валидация
3) сертификация
4) тестирование

18. Авторитетное подтверждение правильности программы есть…

1) верификация
2) валидация
3) аттестация
4) тестирование

19. Метод сборки, при котором система (программа) собирается и тестируется снизу вверх, только «терминальные модули» (модули самого нижнего уровня, не вызывающие других модулей) тестируются автономно есть…

1) метод сандвича
2) метод «большого скачка»
3)восходящее тестирование
4) нисходящее тестирование

20. Метод сборки, при котором нижние уровни собираются и тестируются снизу вверх, а модули верхних уровней сначала тестируются изолированно, а затем собираются нисходящим методом есть…

1) метод сандвича
2) метод «большого скачка»
3) восходящее тестирование
4) нисходящее тестирование

ВАРИАНТ 2

Инструкция: все задания имеют одну и ту же форму – с выбором одного ответа из четырех предложенных

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

1) вероятность отказа
2) вероятность безотказной работы
3) интенсивность отказов
4) коэффициент готовности системы

2. Формула $\alpha(t) = Q'(t) = -P'(t)$ описывает…

1) вероятность отказа
2) вероятность безотказной работы
3) интенсивность отказов
4) частота отказов

3. Основные понятия надежности в наибольшей степени применимы к…

1) сложным комплексам программ для информационно-справочных систем
2) системам автоматизации обработки информации, которые функционируют вне реального времени

	3) программам, разрабатываемым для решения инженерных и научно-исследовательских задач 4) комплексам программ автоматического или автоматизированного управления
4. При параллельной структуре надежностно - функциональной схемы…	1) отказ любого из элементов, реализующих функцию, приводит к отказу функции 2) к отказу функции приводит лишь совместный отказ всех реализующих его элементов 3) к отказу функции приводит отказ определенного числа из реализующих ее элементов 4) к отказу функции приводит отказ одного из реализующих его элементов
5. Надежность аппаратуры в технических системах определяется…	1) надежностью компонент и ошибками в конструкции, допущенными при проектировании или изготовлении 2) качеством тестирования 3) условиями тиражирования 4) уровнем развития пользователя
6.Надежность при структурной схеме с параллельным соединением элементов определяется формулой…	1) $P_i(t) = \sum_{j=0}^{m-1} A_{jn}$ 2) $P_i(t) = 1 - \prod_{j=1}^{n} \left[1 - P_{ij}(t)\right]$ 3) $P_i(t) = 1 - \prod_{j=1}^{n} P_{ij}(t)$ 4) $P_i(t) = \prod_{j=1}^{n} P_{ij}(t)$
7. Отказ это…	1) событие, заключающееся в переходе объекта из работоспособного состояния в неработоспособное 2) событие, заключающееся в нарушении работоспособности 3) такое состояние объекта, при котором он не способен выполнять заданные функции с параметрами, установленными требованиями технической документации 4) свойство объекта не выполнять заданные функции
8. Динамические модели надежности программного обеспечения можно разделить на…	1) эмпирические и теоретические 2) динамические и статистические 3) непрерывные и дискретные 4) непрерывные и ступенчатые
9. Плотность вероятности момента отказа распределения Вейбулла имеет вид…	1) $f(t) = \lambda\delta t^{\delta-1} \cdot e^{-\left(\lambda t^{\delta}\right)}$ 2) $f(t) = \lambda \cdot e^{-(\lambda t)}$

3) $f(t)=\frac{t}{\delta^2}e^{\left(-\frac{t^2}{2\delta^2}\right)}$

4) $f(t)=\frac{1}{\sigma\sqrt{2\pi}}\cdot e^{\left(-\frac{(t-a)^2}{2\sigma^2}\right)}$

10. Методы борьбы со сложностью включают в себя…

1) достижение иерархической структуры и независимости элементов программного обеспечения
2) временную и программную избыточности
3) изоляцию ошибок
4) методы борьбы со сложностью, достижения точности при переводе информации, улучшения обмена информацией, немедленного обнаружения и устранения ошибок

11. Информационная избыточность достигается за счет…

1) использования части производительности ЭВМ
2) взаимного недоверия элементов системы, немедленного обнаружения и регистрации ошибок, дублирование выполнения одинаковых функций, контроля и восстановления данных
3) дублирования части данных
4) достижения большей точности при переводе информации

12. Программная избыточность достигается за счет…

1) использования части производительности ЭВМ
2) взаимного недоверия элементов системы, немедленного обнаружения и регистрации ошибок, дублирование выполнения одинаковых функций, контроля и восстановления данных
3) дублирования части данных
4) достижения большей точности при переводе информации

13. Динамическая избыточность достигается за счет…

1) дублирования части данных
2) иерархической структуры и независимости элементов
3) динамического изменения конфигурации и повторного выполнения операций
4) использования части производительности ЭВМ

14. Методы испытаний комплексов программ на надежность включают в себя…

1) прямые экспериментальные, расчетно-экспериментальные, форсированные
2) предупреждение ошибок, обнаружение ошибок, обеспечение устойчивости к ошибкам
3) дублирование части данных
4) методы борьбы со сложностью

15. Модель частоты появления ошибок является…

1) статической моделью
2) ступенчатой динамической моделью
3) непрерывной динамической моделью

4) частным случаем распределения Рэлея

16. Модель Миллса является…
1) статической моделью
2) ступенчатой динамической моделью
3) непрерывной динамической моделью
4) частным случаем распределения Рэлея

17. Попытка найти ошибки, выполняя программу в тестовой, или моделируемой, среде есть…
1) верификация
2) валидация
3) сертификация
4) тестирование

18. Попытка найти ошибки, выполняя программу в заданной реальной среде есть…
1) верификация
2) валидация
3) сертификация
4) тестирование

19. Метод сборки, при котором система (программа) собирается и тестируется сверху вниз, изолированно тестируется только головной модуль, есть…
1) метод сандвича
2) метод «большого скачка»
3) восходящее тестирование
4) нисходящее тестирование

20. Метод сборки, при котором каждый модуль тестируется автономно, после чего модули интегрируются в систему сразу, есть…
1) метод сандвича
2) метод «большого скачка»
3) восходящее тестирование
4) нисходящее тестирование

1.1. Самостоятельная работа студентов

1.1.1. Построение математических моделей надежности систем: запись закона распределения и расчет его числовых характеристик– 20 ч.

Рекомендуемая литература:

1. Перегруда А.И. Методы расчета показателей надежности ЭВМ. Обнинск, 1994. – 271 с.

1.1.2. Методы тестирования моделей реальных систем – 20 ч.

Рекомендуемая литература:

1. Редькин Н.П. Надежность и диагностика схем. М.: Изд-во МГУ, 1992. – 337 с.
2. Игнатьев М.Б. Активные методы обеспечения надежности алгоритмов и программ. СПб., 1992. – 267 с.

1.1.3. Принципы построения моделей систем – 20 ч.

Рекомендуемая литература:

1. Липаев В.В. Проектирование математического обеспечения АСу. М: Советское радио, 1997. – 271 с.

1.2. Вопросы к экзамену

1.2.1. Показатели надежности АСОИиУ

1.2.2. Потоки отказов

1.2.3. Основные показатели долговечности
1.2.4. Комплексные показатели надежности (коэффициент готовности, коэффициент оперативной готовности)
1.2.5. Комплексные показатели надежности (коэффициент технического использования)
1.2.6. Распределение Вейбулла
1.2.7. Экспоненциальное распределение
1.2.8. Распределение Рэлея
1.2.9. Распределение Гаусса
1.2.10. Эксплуатация аппаратных средств
1.2.11. Техническая документация
1.2.12. Рабочее место и условия эксплуатации
1.2.13. Системный блок
1.2.14. Платы расширения
1.2.15. Надежность и эксплуатация программного обеспечения
1.2.16. Эксплуатация и защита ОС
1.2.17. Эксплуатация и защита файлов
1.2.18. Вирусы
1.2.19. Архивирование данных
1.2.20. Дефрагментация, оптимизация и коррекция дисков
1.2.21. Модернизация аппаратных и программных средств (основные принципы и технико-экономическое обоснование)
1.2.22. Модернизация системного блока
1.2.23. Модернизация дисковой памяти
1.2.24. Модернизация видеоподсистемы
1.2.25. Модернизация видеоподсистемы
1.2.26. Модернизация программного обеспечения
1.2.27. Временная избыточность ПО
1.2.28. Информационная избыточность ПО
1.2.29. Программная избыточность ПО
1.2.30. Средства обеспечения надежности АСОИиУ производственного назначения (средства, базирующиеся на временной избыточности)
1.2.31. Средства обеспечения надежности АСОИиУ производственного назначения (средства, базирующиеся на информационной избыточности)
1.2.32. Средства обеспечения надежности АСОИиУ производственного назначения (средства, базирующиеся на программной избыточности)
1.2.33. Средства обеспечения надежности АСОИиУ производственного назначения (средства, обеспечивающие устойчивость к ошибкам)
1.2.34. Основы эргономического обеспечения разработки АСОИиУ. Характеристика человека как звена системы «человек - машина»
1.2.35. Эргономика аппаратных и программных средств АСОИиУ
1.2.36. Организация компьютеризированных рабочих мест
1.2.37. Организация диалога человек-ЭВМ
1.2.38. Требование к интерфейсу пользователя
1.2.39. Характеристика математических моделей в эргономике
1.2.40. Математическое моделирование деятельности человека-оператора
1.2.41. Моделирование систем «человек-машина» в эргономике
1.2.42. Эргономическая экспертиза
1.2.43. Методы эргономической оценки промышленных изделий и проектных решений
1.2.44. Специфика оценки проекта рабочей системы и ее реализации
1.2.45. Основные характеристики качества ПО

1.2.46. Модель обеспечения качества
1.2.47. Документирование ПС
1.2.48. Тестирование (основные определения)
1.2.49. Тестирование (принципы)
1.2.50. Этапы тестирования
1.2.51. Стратегии тестирования
1.2.52. Методы интеграции системы
1.2.53. Комплексное тестирование
1.2.54. Аксиомы тестирования
1.2.55. Планирование при тестировании
1.2.56. Управление при тестировании
1.2.57. Методы руководства и качество АСОИиУ (организация и подбор кадров)
1.2.58. Методы руководства и качество АСОИиУ (программист-библиотекарь)
1.2.59. Методы руководства и качество АСОИиУ (бригады программистов)
1.2.60. Методы руководства и качество АСОИиУ (принципы хорошего руководства)

1.3. Оценочные критерии

1.3.1. Студент получает зачет по изучаемой дисциплине в случае, если он свободно владеет основными теоретическими понятиями и определениями, а также умеет правильно использовать рассмотренные практические методы.

1.3.2. При оценке знаний на экзамене учитывается: правильность и осознанность изложения содержания ответа на вопросы, полнота раскрытия понятий и закономерностей, точность употребления и трактовки общенаучных и специальных терминов; степень сформированности интеллектуальных и научных способностей экзаменуемого; самостоятельность ответа; речевая грамотность и логическая последовательность ответа. Критерии оценок:

- отлично – полно раскрыто содержание вопросов в объеме программы и рекомендованной литературы; четко и правильно даны определения и раскрыто содержание концептуальных понятий, закономерностей, корректно использованы научные термины; для доказательства использованы различные теоретические знания, выводы из наблюдений и опытов; ответ самостоятельный, исчерпывающий, без наводящих дополнительных вопросов, с опорой на знания, приобретенные в процессе специализации по выбранному направлению информатики.
- хорошо – раскрыто основное содержание вопросов; в основном правильно даны определения понятий и использованы научные термины; ответ самостоятельный; определения понятий неполные, допущены нарушения последовательности изложения, небольшие неточности при использовании научных терминов или в выводах и обобщениях, исправляемые по дополнительным вопросам экзаменаторов.
- удовлетворительно – усвоено основное содержание учебного материала, но изложено фрагментарно, не всегда последовательно; определение понятий недостаточно четкое; не использованы в качестве доказательства выводы из наблюдений и опытов или допущены ошибки при их изложении; допущены ошибки и неточности в использовании научной терминологии, определении понятий;
- неудовлетворительно – ответ неправильный, не раскрыто основное содержание программного материала; не даны ответы на вспомогательные вопросы экзаменаторов; допущены грубые ошибки в определении понятий и использовании терминологии.

Содержание