Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение
высшего профессионального образования

«Владимирский государственный университет
имени Александра Григорьевича и Николая Григорьевича Столетовых»

КОМПЛЕКСНАЯ ЗАЩИТА
ОБЪЕКТОВ ИНФОРМАТИЗАЦИИ

КНИГА 22

Ю. М. МОНАХОВ

ФУНКЦИОНАЛЬНАЯ УСТОЙЧИВОСТЬ ИНФОРМАЦИОННЫХ СИСТЕМ

В трех частях

Часть 1. Надежность программного обеспечения

Учебное пособие

Владимир 2011

УДК 930.1
ББК 32.81
М77

Редактор серии – доктор технических наук
профессор М. Ю. Монахов

Рецензенты:

Кандидат технических наук, профессор
зав. кафедрой информатики и вычислительной техники
Владимирского государственного гуманитарного университета
Ю. А. Медведев

Доктор технических наук, профессор
зав. кафедрой безопасности жизнедеятельности
Владимирского государственного университета
О. В. Веселов

Печатается по решению редакционного совета
Владимирского государственного университета

Монахов, Ю. М.

М77 Функциональная устойчивость информационных систем. В 3 ч. Ч. 1. Надежность программного обеспечения : учеб. пособие / Ю. М. Монахов ; Владим. гос. ун-т. – Владимир : Изд-во Владим. гос. ун-та, 2011. – 60 с. – ISBN 978-5-9984-0189-3.

В первой части учебного пособия представлен систематизированный материал по основным моделям оценки надежности программного обеспечения информационных систем.

Предназначено для магистрантов направления «Информационные системы» всех форм обучения.

Рекомендовано для формирования профессиональных компетенций в соответствии с ФГОС 3-го поколения.

Табл. 1. Ил. 5. Библиогр.: 49 назв.

УДК 930.1
ББК 32.81

ISBN 978-5-9984-0189-3

Предисловие

Функционирование больших информационных систем (ИС), предназначенных для автоматизации управления, происходит в постоянном взаимодействии с внешней средой. При этом обширный класс такого взаимодействия представляет разнообразные конфликты, существенно влияющие на достижение общесистемной цели. Такие (информационные) конфликты приводят к разрушению информационных ресурсов, нарушению штатных информационных процессов, и как следствие срыву выполнения системных и прикладных функций. Все это предопределяет наличие в ИС механизмов, которые должны обеспечивать новое качество – способность сохранения и/или восстановления данных (устойчивость) функций в условиях различного рода неблагоприятных воздействий.

Данное качество назовем функциональной устойчивостью (ФУ) ИС. ФУ есть интегральное свойство, включающее надежность, живучесть и безопасность. Соответственно оценка показателей ФУ, необходимая для сравнения различных вариантов проектирования, является сложной научно-практической задачей. Еще сложнее поиск наилучшего варианта, при котором достигаются оптимальные показатели ФУ при некоторых ограничениях. В этом направлении получен ряд научно-практических результатов, но отсутствие учебно-методической литературы по данной важной проблеме приводит к определенным недостаткам в подготовке специалистов, занимающихся созданием больших ИС.

Цель учебного пособия – анализ путей оценки и повышения ФУ ИС.

Автор полагает, что в рамках такой обширной области науки и практики, как функциональная устойчивость информационных систем, наименее изученными в настоящий момент являются задачи:

- практической оценки надежности программного обеспечения ИС;

- раннего обнаружения и противодействия вредоносным информационным воздействиям внешней среды;

- разработки механизмов повышения живучести ИС.

Выделенные задачи составляют структуру учебного пособия. В первой части рассматриваются методы практической оценки надежности программного обеспечения информационных систем, во второй – анализируются и предлагаются новые алгоритмы обнаружения и противодействия вредоносным информационным воздействиям на ИС; в третьей – предлагаются новые механизмы оценки и повышения живучести распределенных информационных систем.

Введение

Опыт создания и применения информационных систем в последние десятилетия выявил множество ситуаций, при которых сбои и отказы их функционирования были обусловлены дефектами комплексов программ и сопровождались большим ущербом. Вследствие ошибок в программах автоматического управления погибло несколько отечественных, американских и французских спутников, происходили отказы и катастрофы в сложных административных, банковских и технологических ИС.

В результате около двадцати лет назад появились первые обобщающие работы, в которых были сформулированы концепция и основные положения теории надежности программного обеспечения (ПО) для ИС. В это время были заложены основы методологии и технологии создания надежных комплексов программ, созданы множество методов и моделей исследования надежности ПО, однако единого подхода к решению указанной проблемы не предполагалось. Причина этого заключается в уникальности каждой программной системы. Тем не менее при разработке ответственных проектов их создатели и заказчики стараются получить оценку надежности ПО, как правило, на основе результатов конечных испытаний.

Современные программные системы используют при работе колоссальный объем данных, проходящих через стандартные модули и функции. Поэтому выявить все связи и пути обработки информации, даже для достаточно несложной программы, практически невозможно. Исходя из этого детализация элементов расчета надежности

обычно ограничивается законченными программными образованиями (условно называемыми в пособии программными модулями), взаимодействующими между собой и составляющими более сложное объединение, надежность которого нас интересует.

Экспериментальное определение реальной надежности функционирования ПО весьма трудоемкая, трудно автоматизируемая и не всегда безопасная часть жизненного цикла ПО. Накоплен значительный опыт определения надежности ПО, применяемых в авиационной, ракетно-космической и других областях современной высокоинтеллектуальной техники. В этих областях недопустимо для тестирования и определения надежности ПО использовать функционирование реальных объектов. В результате особое значение приобрели методы и средства моделирования внешней среды для автоматизированной генерации тестов при испытаниях надежности таких ПО. В этих случаях на базе программных моделей и компонент реальных систем создаются моделирующие испытательные стенды, обеспечивающие возможность определения надежности функционирования конкретных ПО в условиях штатных и критических внешних воздействий, соответствующих подлинным характеристикам внешней среды.

Определим надежность программного обеспечения ИС как способности комплекса программ выполнять заданные функции, сохраняя во времени значения установленных показателей в заданных пределах. При эксплуатации изменение надежности ПО со временем существенно отличается от изменения надежности технических средств. ПО не подвержено износу, практически отсутствуют ошибки производства, так как они встречаются редко и легко могут быть исправлены. Ненадежность ПО целиком определяется ошибками разработки.

Достижения в области повышения надежности технических средств в настоящее время более ощутимы, чем надежность ПО. Это объясняется рядом причин:

- ПО по своей природе сложнее технических средств;
- ПО «чувствительно» к области применения;
- элементы ПО недостаточно унифицированы.

Из множества факторов, влияющих на надежность ПО, наиболее важный – количество оставшихся в программе ошибок.

Как правило, оценка надежности ПО базируется на теории надежности технических систем, где разработано значительное количе-

ство математических моделей и методов. Исходя из данных соображений, обычно рассматривается ряд моделей, качественно группируемых по следующим признакам:

- модели, связанные с теорией надежности технических средств и содержащие предположения о вероятностном распределении ошибок в ПО;

- модели, не базирующиеся на теории надежности аппаратуры, но позволяющие получить приемлемые результаты оценки;

- модели, позволяющие оценить ПО с учетом сложности.

Все эти модели являются количественными и могут быть использованы при измерении, управлении и предсказании уровня «завершенности» ПО на стадиях разработки и тестирования.

В качестве показателей надежности ПО используются:

- вероятность работоспособного состояния ПО на заданном интервале времени;

- среднее время проявления ошибки;

- интенсивность ошибки.

Одним из способов оценки среднего времени проявления ошибки ПО является наблюдение за поведением программы в определенный промежуток времени и на участке между двумя последующими ошибками. Время между обнаружением двух последовательных ошибок имеет тенденцию к возрастанию по мере обнаружения и исправления ошибок. Экстраполируя этот ряд, можно с определенной вероятностью предсказать общее количество ошибок в ПО. Отметим, что гораздо лучшим приемом, требующим меньшего количества данных для той же точности прогноза, является постулирование модели для исправления ошибок и использование тестовых данных для оценки.

Модель ошибки может основываться на предварительной работе, связывающей вероятность работоспособного состояния ПО на заданном интервале времени и среднее время проявления ошибки с данными исправления (отладки). Число ошибок, остающихся в программе, моделируется статистически в терминах числа исправленных ошибок, размера программы и начального числа ошибок. Часто делается дополнительное допущение о том, что темп сбоев ПО пропорционален числу остающихся ошибок. Модель содержит две неизвестные константы: число первоначальных ошибок и константу пропор-

циональности, для оценки которых используется информация функционального теста.

Первая глава представляет собой обзорную часть. В ней рассматриваются и анализируются классические модели оценки надежности программных средств. Вторая глава посвящена алгоритмам практической оценки надежности программ и методик обработки результатов, полученных при тестировании. В третьей главе представлены результаты и выводы тестирования стандартных программных сервисов доступа информации в ИС.

Глава 1. Анализ современного состояния проблемы оценки надежности программного обеспечения

1.1. Обзор работ по проблеме

При написании учебного пособия было проанализировано большое количество статей, монографий, учебных пособий и других информационных источников по проблеме надежности в целом и надежности программного обеспечения в частности. В [7, 8, 10, 11, 19, 21, 23, 34] основное внимание уделяется основам теории надежности: понятиям, определениям и постулатам, подробной классификации отказов, характеристик надежности при внезапных и постепенных отказах. В указанных источниках выявлены базовые характеристики надежности как показатели безотказности, ремонтопригодности, долговечности, сохраняемости. Рассмотрены общие методы расчета надежности технических систем различного назначения как нерезервированных, так и резервированных.

Выделим работу В. Р. Матвеевского [21], посвященную математическим моделям в теории надежности. В ней приведены аналитические зависимости интенсивности отказов от времени для распределений Вейбулла, Релея и др. Так, нормальное распределение и распределение Вейбулла используются для расчета надежности «стареющих» элементов, а экспоненциальное распределение, являясь частным случаем распределения Вейбулла, – для расчета надежности «нестареющих» элементов. Кроме этого рассматриваются мероприятия по

формированию показателей надёжности на различных стадиях проектирования, методы повышения надежности.

Практикум по теории надежности А. М. Половко и С. В. Гурова [31] содержит набор задач по расчету показателей надежности нерезервированных невосстанавливаемых систем, резервированных невосстанавливаемых систем, нерезервированных восстанавливаемых систем, резервированных восстанавливаемых систем. Большую часть практикума составляют лабораторные работы, посвященные исследованию надежности и свойств систем различной структуры, анализу влияния профилактики на надежность систем, а также исследованию влияния временного резервирования на надежность системы.

В учебном пособии С. М. Вихарева [6] большое внимание уделено структурно-логическому анализу технических систем. Такой анализ проводится с целью оценки влияния каждого элемента на работоспособность системы в целом. Структурно-логический анализ предполагает представление технической системы в виде схемы из последовательно и параллельно соединенных элементов. При данном подходе прослеживается определенная аналогия с цепью, составленной из проводящих элементов (исправный элемент пропускает ток, отказавший – не пропускает): работоспособному состоянию технической системы соответствует возможность протекания тока от входа до выхода цепи.

К сожалению, методология, представленная в выделенных информационных источниках, не позволяет применить ее в полном объеме при оценке надежности ПО. Напимер, данных основ не достаточно для расчета надежности сервисов доступа к информации, так они не позволяют оценить надежность ПО, в работе которых присутствуют циклические операции.

В статье С. Г. Романюка [33] рассматривается вероятностный подход к проблеме надежности, который при изучении надежности состоит в анализе исследуемого объекта (самолета, системы охраны, компьютерной программы и т.д.), построении, исходя из "физических" соображений о его природе, пространств элементарных событий, во введении на них вероятностной меры и рассмотрении случайных величин. В работе выдвинуты интересные соображения по поводу взаимосвязи надежности программы и наличия в ней ошибок:

- число ошибок в программе – величина "ненаблюдаемая", наблюдаются не сами ошибки, а результат их проявления;

- неверное срабатывание программы может быть следствием не одной, а сразу нескольких ошибок;

- ошибки могут компенсировать друг друга, так что после исправления какой-то одной ошибки программа может начать "работать хуже";

- надежность характеризует частоту проявления ошибок, но не их количество. В то же время хорошо известно, что ошибки проявляются с разной частотой: некоторые остаются невыявленными после многих месяцев и даже лет эксплуатации, но, нетрудно привести примеры, когда одна ошибка приводит к неверному срабатыванию программы при любых исходных данных, т.е. к нулевой надежности.

Статья Билла Грэма, Пола Н. Леру, Тодда Лендри [3] посвящена статическому и динамическому анализу программного кода на наличие разнообразных дефектов и слабых мест. В ней рассматривается метод статического анализа. Авторы показывают, как с его помощью можно предотвратить отдельные проблемы программного кода и гарантировать, что новый код соответствует стандарту. Используя разные техники, например, проверку абстрактного синтаксического дерева (abstract syntax tree, AST) и анализ кодовых путей, инструменты статического анализа могут выявить скрытые уязвимости, логические ошибки, дефекты реализации и другие проблемы. Это возможно как на этапе разработки, так и во время компоновки системы. Здесь же предлагается метод динамического анализа, который можно использовать на этапе разработки программных модулей и их системной интеграции и который позволяет выявить проблемы, пропущенные при статическом анализе. При динамическом анализе не только обнаруживаются ошибки, связанные с указателями и другими некорректностями в программном коде, но также появляется возможность оптимизировать использование циклов центрального процессора, оперативной памяти, флеш-памяти и других ресурсов. Кроме того, в статье обсуждаются варианты комбинирования статического и динамического анализа, что поможет предотвращать возврат к более ранним этапам разработки по мере "созревания" продукта. Такой подход с использованием сразу двух методик помогает избежать проявления большинства проблем еще на ранних этапах разработки, когда их легче и дешевле всего исправить.

Концептуальная книга Г. Майерса [20] целиком посвящена надежности ПО. Книга состоит из четырех частей. Первые две части по-

священы общим понятиям и проектированию надежного ПО, третья – тестированию программ, где уделяется большое внимание принципам и методам тестирования. Рассматриваются такие методы, как восходящее тестирование, нисходящее тестирование, модифицированный нисходящий метод, метод большого скачка, метод сэндвича, модифицированный метод сэндвича. Представлены аксиомы тестирования. Рассматриваются подходы к тестированию модулей, внешних функций и комплексное тестирование. Отдельные главы посвящены проектированию и выполнению теста, выбору инструментария для тестирования, большое внимание уделяется поиску и исправлению ошибок, а также инструментам для поиска ошибок. Четвертая часть посвящена дополнительным вопросам надежности. В ней рассматривается влияние на надежность таких факторов, как методы руководства, языки программирования, архитектура ЭВМ. Целая глава посвящена доказательству правильности программ, также здесь приводятся модели надежности. Модель роста надежности тесно связана с теорией надежности аппаратуры и существенно опирается на определенные предположения о распределении вероятностей отказов программного обеспечения. Статическая модель Миллса и простые интуитивные модели дают сходные результаты, но не связаны с теорией надежности аппаратуры. Последние несколько моделей – модели сложности – предназначены для предсказания сложности программных систем.

В литературе [14 – 17, 32, 45] представлены различные модели надежности ПО, их классификация, приведены примеры по использованию моделей.

В учебнике В. В. Шуракова [48] изложены модели, базирующиеся на классических моделях теории надежности технических систем, одной из которых является модель переходных вероятностей Маркова. Эта модель позволяет предсказать характеристики системы во время проверки моделированием до того, как программа будет реализована. Здесь же рассматриваются модели с внесением (рассеиванием) ошибок. Процесс рассеивания ошибок – наиболее слабое звено данной модели вследствие предположения, что исходные и рассеянные ошибки имеют одинаковую, но неизвестную вероятность быть обнаруженными. При этом предполагается, что рассеянные ошибки должны быть подобраны с тем, чтобы походить на типичные ошибки программиста, проще их искать без особых ухищрений типа «рассеи-

вания». Если же рассматривать этот недостаток модели, рассеивающей ошибки, в свете недостатков других моделей, то можно утверждать, что эта проблема относительно небольшая и разрешима с практической точки зрения. Модель, рассеивающая ошибки, полезна на практике, так как она оказывает положительное психологическое воздействие на лиц, выполняющих тестирование, уже только тем, что они знают: в программу внесены ошибки.

В статье Н. В. Василенко и В. А. Макарова [4] дается обширное описание динамических и статических моделей надежности. Так, в динамических моделях в отличие от статических учитывается время появления ошибок. К динамическим моделям относятся модель Шумана, модель Ла Падула, модель Джелинского – Моранды, модель Шика – Волвертона, модель Муса, модель переходных вероятностей, к статическим – модель Миллса, модель Липова, простая интуитивная модель, модель Коркорэна, модель последовательности испытаний Бернулли, модель Нельсона. Также в статье представлено несколько эмпирических моделей, основанных на анализе накопленной информации о функционировании ранее разработанных программ. Примерами эмпирических моделей являются модель фирмы IBM и модель Холстеда.

В работе В. В. Азовцева [1] помимо описания некоторых статических и динамических моделей приводятся примеры по их использованию. Кроме того, рассматриваются особенности объектно-ориентированного программирования в данных задачах.

Что касается оценки надежности сложных программных комплексов, то здесь можно выделить работы В. А. Смагина [39 – 43], А. С. Можаева [22 – 24], О. А. Панина [28 – 30], И. А. Рябинина [35 – 37].

В [43] изучены прогностические возможности модели Седякина – Джелинского – Моранды на основе функции максимального правдоподобия. Приведено основное аналитическое уравнение для прогноза будущего интервала времени, лежащего за последним наблюденным событием потока. Дано обобщение прогностической модели на случай использования произвольных распределений. Построена модель прогнозирования момента наступления очередного случайного события в потоке событий. Данная модель позволяет по наблюдениям интервалов между событиями – ошибками функционирования программного модуля при условии справедливости существования экс-

поненциального распределения интервалов времени – определить ожидаемое число ошибок в модуле и интенсивность проявления любой одной ошибки. Прикладное значение полученных результатов состоит в том, что они могут непосредственно использоваться при анализе и обеспечении надежности сложных программных комплексов. Кроме того, полученные теоретические результаты могут найти применение и при изучении других потоков событий, природа которых не связана со временем. В работе предложена стратегия повышения вероятности безошибочного функционирования сложного программного комплекса, использование которой позволяет достичь заданной или максимальной вероятности безошибочного функционирования в зависимости от исходных условий решаемой задачи. Стратегия основана на вероятностном прогнозе времени ожидаемой в будущем ошибки критичного к ней программного модуля и повторного применения выражения для расчета вероятности безошибочного функционирования программного комплекса, представленного в виде графовой модели.

В анализируемой статье [43] рассматриваются две задачи, необходимые при исследовании надежности программных комплексов. Первая задача состоит в том, чтобы по заданной структуре программного комплекса, состоящего из некоторой совокупности программных модулей, имеющих показатели надежности, найти показатель надежности программного комплекса. Эту задачу традиционно называют прямой. Наряду с ней может решаться вторая задача – достижение максимального (минимального) значения показателя надежности при ограничениях на ресурсы, в качестве которых выступают время, стоимость и другие параметры (обратная задача).

Предложенный метод позволяет производить оценку надежности сложных программных комплексов при известных показателях надежности составляющих модулей и их вероятностной взаимосвязи в стохастическом графе.

В книге И. А. Рябинина [35] изложены основы логико-вероятностного исчисления, необходимые для исследования надежности и безопасности структурно-сложных систем. Рассмотрены проблемы исходных данных о безотказности элементов при малых объемах статистической информации, доверительные и допустимые интервалы оценки надежности. Приведены аналитические и графические формы

представления условий работоспособности и опасного состояния системы, изложены логико-вероятностные методы исследования надежности и безопасности на большом числе примеров.

В статье О. А. Панина [28] приведен математический аппарат, составляющий основу логико-вероятностных методов. Дана процедура анализа, которая заключается в составлении схемы системы.

Выделим работу Е. Н. Зайцевой и Ю. В. Поттосина [13], в которой для анализа надежности систем с несколькими уровнями работоспособности предложено использовать так называемые «направленные логические производные», которые позволяют определить наборы переменных при заданных изменениях переменной и для заданного изменения функции.

В пособии В. А. Яворского [49] изложены основы планирования эксперимента, анализ экспериментальных данных, правила ведения лабораторного журнала и оформления результатов эксперимента.

Что касается последних разработок, связанных с оценкой надежности, то исходя из обзора [44] программных комплексов для оценки надежности, можно сделать вывод, что они в основном привязаны к логико-вероятностному подходу. Здесь прежде всего отметим работы [5, 26, 47].

В диссертации А. А. Нозика [27] проведен анализ, выполнена разработка комплекса методов и методик, необходимых для оценки надежности структурно-сложных технических систем большой размерности. Все предложенные механизмы имеют алгоритмический уровень разработки, что позволило осуществить их непосредственную реализацию в программном комплексе автоматизированного структурно-логического моделирования и расчета надежности и безопасности структурно-сложных технических систем большой размерности, основанного на общем логико-вероятностном методе.

В диссертации С. В. Гурова [9] предложена универсальная математическая модель функционирования невосстанавливаемых и восстанавливаемых систем с произвольными распределениями отказов и восстановлений элементов и параметров. На ее основе дано базисное математическое описание стационарного и нестационарного режимов функционирования, пригодное для построения математических моделей различных классов систем. Эти модели могут применяться для оценки надежности систем.

В работе В. Н. Задорожного [12] рассматривается аналитический метод расчета надежности структурно-сложных систем, основанный на использовании редукции. Практической реализацией результатов данной работы является программа в среде Java. Разработанная программа может функционировать на любой платформе, для которой имеется среда Java, т.е. она является платформно-независимой. Основные элементы разработанной программы – две большие части: первая – графический редактор, вторая – на основе полученной информации производит редукцию, рассчитывает надежность. Основное назначение графического редактора состоит в создании удобной интерактивной среды, позволяющей пользователю эффективно выполнять все виды работ по вводу структурной схемы анализируемой системы (добавление, удаление перемещение), заданию значений надежности элементов и общей организации компьютерного моделирования и расчета показателей надежности исследуемых систем.

1.2. Непрерывные динамические модели оценки надежности

В таких моделях ошибки во время тестирования не исправляются.

Модель Джелински – Моранды

Модель основана на допущениях, что время до следующего отказа распределено экспоненциально, а интенсивность отказов программы пропорциональна количеству оставшихся в программе ошибок.

Согласно этим допущениям вероятность безотказной работы ПО как функция времени t_i равна

$$P(t_i) = e^{-\lambda_i t_i}, \quad (1.1)$$

где λ_i – интенсивность отказов,

$$\lambda_i = C_D(N - (i-1)), \quad (1.2)$$

здесь C_D – коэффициент пропорциональности; N – первоначальное количество ошибок.

В (1.1) отсчет времени начинается от момента последнего (i – 1)-го отказа программы.

По методу максимума правдоподобия на основании (1.1), обозначая через k номер прогнозируемого отказа, получим, что функция правдоподобия имеет вид

$$F = \prod_{i=1}^{k-1} C_D(N - i + 1)e^{-C_D(N-i+1)t_i}. \quad (1.3)$$

Логарифмическая функция правдоподобия имеет вид

$$L = \ln F = \sum_{i=1}^{k-1}[\ln\big(C_D(N - i + 1)\big) - C_D(N - i + 1)t_i]. \quad (1.4)$$

Отсюда условия для нахождения экстремума

$$\frac{\partial L}{\partial C_D} = \sum_{i=1}^{k-1}\left[\frac{1}{C_D} - (N - i + 1)t_i\right] = 0, \quad (1.5)$$

$$\frac{\partial L}{\partial N} = \sum_{i=1}^{k-1}\left[\frac{1}{N-i+1} - C_D t_i\right] = 0. \quad (1.6)$$

Из (1.6) получим

$$C_D = \frac{\sum_{i=1}^{k-1}\frac{1}{N-i+1}}{\sum_{i=1}^{k-1} t_i}. \quad (1.7)$$

Подставим (1.7) в (1.5), получим

$$(k-1)\frac{\sum_{i=1}^{k-1} t_i}{\sum_{i=1}^{k-1}\frac{1}{N-i+1}} = \sum_{i=1}^{k-1}(N - i + 1)t_i. \quad (1.8)$$

При известных значениях k; t_1, t_2, ..., t_k из (1.7) и (1.8) можно найти значения параметров модели C_D и N, а затем интенсивность отказов, время между отказами t_{k+1}, вероятность безотказной работы через время t_{k+1} после последнего отказа.

Преимущества и недостатки модели. Основное преимущество модели – простота расчетов, недостаток состоит в том, что при неточном определении величины N интенсивность отказов программы может стать отрицательной, что приводит к бессмысленному результату. Кроме того, предполагается, что при исправлении обнаруженных ошибок не вносятся новые ошибки, что тоже не всегда выполняется.

Модель переходных вероятностей Маркова

Модель позволяет получить оценки и предсказания вероятного числа ошибок, которые будут исправлены в заданное время, на основе предварительного моделирования интенсивности случающихся ошибок l, а также принятой системы исправления ошибок, работающей с интенсивностью m. Модель позволяет получить предсказания для готовности $A(t)$ и надежности $R(t)$ системы ПО.

Принимаются следующие основные ограничения разрабатываемой модели:

- любая ошибка рассматривается как случайная и без градации последствий, которые она порождает;

- интенсивность проявления ошибок постоянна и равна l;

- интенсивность исправления ошибок постоянна и равна m;

время перехода системы из одного состояния в другое бесконечно мало.

Рассмотрим систему, начинающую работать в момент времени $t = 0$. Система работает до появления ошибки в соответствии с предопределенным критерием. Результаты эксперимента собираются в отрезки времени, за которые могут произойти отказы в работе. Тогда переменная времени случайного сбоя может быть определена как

$$t'(\xi) = \xi;\ \xi \geq 0, \tag{1.9}$$

где ξ – местоположение точек на дискретной временной оси эксперимента. Предположим, что случайная переменная имеет функцию распределения

$$F(t) = P\{\xi: t(\xi) \leq t\}, \tag{1.10}$$

и, если она существует, то плотность функции распределения будет

$$f(t) = \frac{dF(t)}{dt}. \tag{1.11}$$

Надежность системы *R(t)* определяется вероятностью отсутствия сбоя в интервале [0, t]:

$$R(t) = P\{t' \geq t\}. \tag{1.12}$$

Под готовностью системы к моменту времени t понимается вероятность того, что система находится в рабочем состоянии во время t:

$$A(t) = P. \tag{1.13}$$

Предположим, что в начальный период ($t = 0$) система содержит неизвестное число (n) ошибок. В качестве начала отсчета времени работы системы выбирается начало фазы тестирования. Принимаем также, что процессы обнаружения и исправления ошибок реализуются попеременно и последовательно.

Ряд состояний системы $\{n, n - 1, n - 2, \ldots\}$ соответствует процессам обнаружения ошибок. По аналогии для случая устранения ошибок введем состояния системы $\{m, m - 1, m - 2, \ldots\}$. Система находится в состоянии $(n - k)$, если ошибка $(k - 1)$ уже исправлена, а ошибка k еще не обнаружена. В то же время система будет находить-

ся в состоянии $(m - k)$ после того, как ошибка k обнаружена, но еще не исправлена. Общая схема модели с указанием вероятностей перехода между состояниями показана на рис. 1.

Пусть $S'(t)$ есть случайная переменная, через которую обозначено состояние системы в момент времени t. Эксперимент будет построен так, что в некоторый момент времени предполагаем систему остановленной и наблюдаем ее состояние. Пространство возможных состояний S системы может быть представлено так:

$$S = \{n, m, n-1, m-1, n-2, m-2, \dots\}.$$

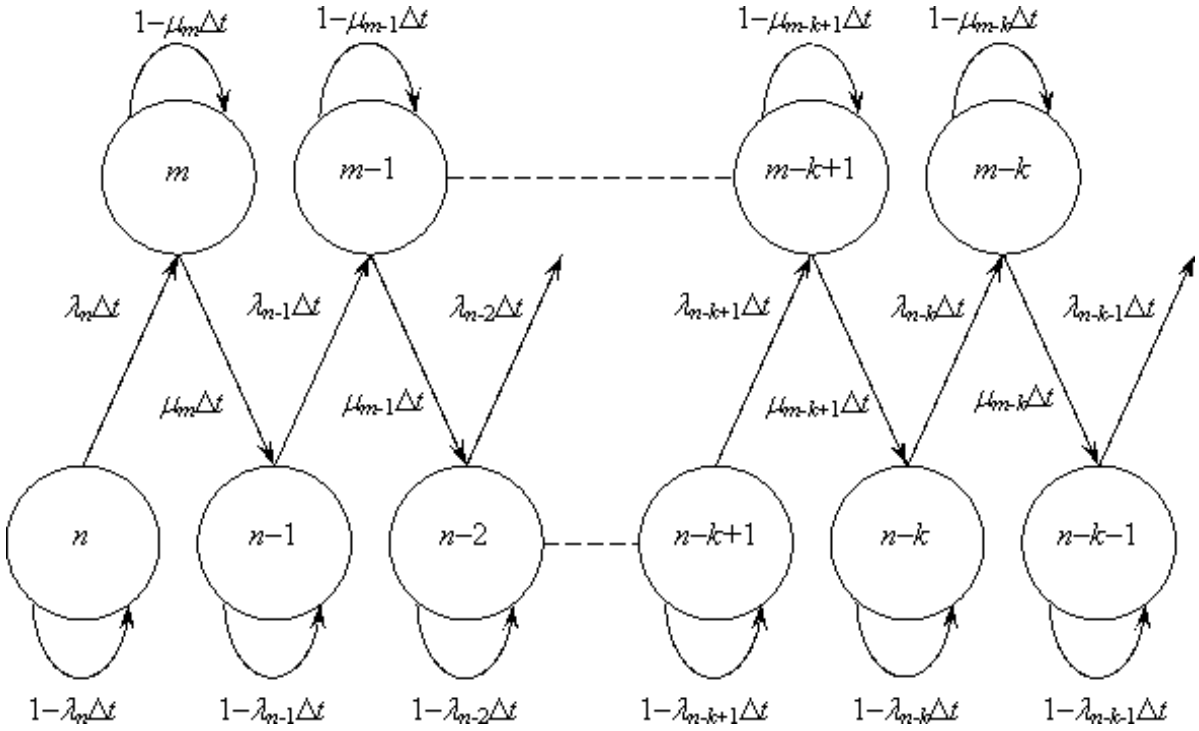

Рис. 1. Модель многих состояний для оценки характеристик ПО

Теперь предположим, что в моменты $t_1 < t_2 < \cdots t_l < \cdots < t$ (любая последовательность наблюдений) последовательность случайных переменных $S'(t_1), S'(t_2), \dots, S'(t)$ удовлетворяет для любого положительного целого числа ℓ следующему равенству:

$$P\{S'(t) = r \mid S'(t_l) = r-1, S'(t_{l-1}) = r-2, \dots, S'(t_1) = r-l\} =$$
$$= P\{S'(t) = r \mid S'(t_l) = r-1\}, \qquad (1.14)$$

где $t, r-1, r-2, \dots r-t$ соответствуют последовательности состояний $(n-k), (m-k+1), (n-k+1), \dots, (n-2), (m-1), (n-1), m, n$.

Таким образом, любое состояние модели определяется рядом переходных вероятностей $\{P_{ij}\}$, где P_{ij} обозначает вероятность перехода из состояния i в состояние j и не зависит от предшествующих и последующих состояний системы, кроме состояний i и j. Вероятность перехода из состояния $(n - k)$ к состоянию $(m - k)$ равна $\lambda_{n-k}\Delta t$ при $k = 0, 1, 2, \ldots$. Аналогично этому вероятность перехода из состояния $(m - k)$ к состоянию $(n - k - 1)$ равна $\mu_{m-k}\Delta t$ при $k = 0, 1, 2, \ldots$.

Интенсивности перехода l_j и m_j зависят от текущего состояния системы. Для системы ПО l_j означает интенсивность возникновения (проявления), а m_j – интенсивность устранения ошибок. Следовательно, полная матрица переходных вероятностей системы может быть представлена следующим образом:

$$\begin{pmatrix} 1-\lambda_n\Delta t & \lambda_n\Delta t & 0 & 0 & \ldots & 0 & 0 & \ldots \\ 0 & 1-\mu_m\Delta t & \mu_m\Delta t & 0 & \ldots & 0 & 0 & \ldots \\ 0 & 0 & 1-\lambda_{n-1}\Delta t & \lambda_{n-1}\Delta t & \ldots & 0 & 0 & \ldots \\ 0 & 0 & 0 & 1-\mu_{m-1}\Delta t & \ldots & 0 & 0 & \ldots \\ 0 & 0 & 0 & 0 & \ldots & 1-\lambda_{n-k}\Delta t & \lambda_{n-k}\Delta t & \ldots \\ 0 & 0 & 0 & 0 & \ldots & 0 & 1-\mu_{n-k}\Delta t & \ldots \\ \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \end{pmatrix}. \quad (1.15)$$

Далее получим выражения для готовности $A(t)$ и надежности $R(t)$ системы в терминах вероятности состояния занятости системы:

$$P_{n-k}(t) = P\{S'(t) = n - k\};\ k = 0, 1, 2, \ldots ; \quad (1.16)$$

$$P_{m-k}(t) = P\{S'(t) = m - k\};\ k = 0, 1, 2, \ldots . \quad (1.17)$$

Выражение для готовности системы во время t ($t > 0$) получим на основе ее определения

$$A(t) = \sum_{k=0}^{\infty} P_{n-k}(t). \quad (1.18)$$

Готовность системы во время t определяется как результат простого сложения всех вероятностей состояний занятости.

Надежность системы зависит от степени ее отладки, т.е. чем выше степень отладки системы, тем больше ожидаемая надежность. Предположим, что к моменту t система только что вошла в состояние $(n - k)$, т. е. ошибка k только что устранена. Назовем это время как t. Тогда в интервале времени $(0, T_k + 1)$, где $t = T_k + 1$ может проявиться ошибка $(k + 1)$ при принятой постоянной интенсивности проявления ошибок l_k.

На основании формулы функции надежности, порождающей вероятность отсутствия сбоев в интервале времени от 0 до t, $R(t) = e^{-\lambda t}$, получим выражение для надежности:

$$R_k(\tau) = e^{-\lambda_k t}; 0 \le \tau \le T_{k+1};\ k = 0, 1, 2, \dots . \qquad (1.19)$$

Недостатки: модель предсказывает поведение системы во время функционирования в среднем. На практике интенсивность исправления ошибок запаздывает по отношению к их обнаружению, что в известной мере затрудняет процесс.

Достоинства: предполагается, что модель в начальный период будет использоваться со значениями l и m, которые были получены на базе накопления прошлого опыта. В связи с тем что последующая работа модели позволит, в свою очередь, накопить данные об ошибках, возможно дальнейшее повышение точности анализа при использовании данных предыдущего моделирования.

Таким образом, модель дает возможность предсказать характеристики системы путем проведения моделирования до того, как программа будет осуществлена, или после ее осуществления. Модель может быть полезна в определении момента достаточной степени отлаженности системы ПО, т.е. удовлетворения характеристикам надежности.

1.3. Дискретные динамические модели оценки надежности

В таких моделях в случае появления отказов ищутся и исправляются все ошибки, из-за которых произошли отказы.

Модель Шумана

В модели предполагается, что тестирование проводится в несколько этапов, каждый из которых представляет собой выполнение программы по набору тестовых данных. Выявленные в течение этапа тестирования ошибки регистрируются, но не исправляются. По завершении этапа исправляются все обнаруженные ошибки, корректируются тестовые наборы и проводится новый этап тестирования.

Предполагается, что при корректировке новые ошибки не вносятся, интенсивность обнаружения ошибок пропорциональна числу оставшихся.

Пусть всего проводятся k этапов тестирования. Обозначим продолжительность каждого этапа через $t_1, \ldots, t_k$, а число ошибок, обнаруженных на каждом этапе, через $m_1, \ldots, m_k$.

Пусть $T = t_1 + \ldots + t_k$ – общее время тестирования; $n = m_1 + \ldots + m_k$ – общее число обнаруженных и исправленных при тестировании ошибок; $n_i = m_1 + \ldots + m_i$ – число ошибок, исправленных к началу $(i + 1)$-го этапа тестирования ($n_0 = 0$).

Модель Шумана ПО на i-м этапе тестирования описывается функцией надежности

$$R_i(t) = e^{-\lambda_i t}, \tag{1.20}$$

где $\lambda_i = (N - n_{i-1})C$; N – первоначальное количество ошибок в ПО; $N - n_{i-1}$ – количество ошибок, оставшихся к началу i-го этапа; C – коэффициент пропорциональности, равный

$$C = \frac{\sum_{i=1}^{k} \frac{m_i}{N - n_{i-1}}}{\sum_{i=1}^{k} t_i}. \tag{1.21}$$

Для нахождения первоначального количества ошибок N используется уравнение

$$\sum_{i=1}^{k} m_i \frac{\sum_{i=1}^{k} t_i}{\sum_{i=1}^{k} \frac{m_i}{N - n_{i-1}}} = \sum_{i=1}^{k} (N - n_{i-1}) t_i. \tag{1.22}$$

При известных значениях k; $t_1, t_2, \ldots, t_k$; $m_0, m_1, \ldots, m_k$ из (1.21) и (1.22) можно найти значения параметров модели C и N. После чего определим следующие показатели:

1) число оставшихся ошибок в программном обеспечении

$$N_r = N - n; \tag{1.23}$$

2) функцию надежности программного обеспечения по завершении тестирования

$$R(t) = e^{-\lambda t}, \tag{1.24}$$

где $\lambda = C(N - n)$.

Преимущества и недостатки модели. К преимуществам относится то, что по модели можно определить все неизвестные параметры, т.е. нет необходимости обращаться к другим моделям – это сокращает время расчета надежности.

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

Модель Муса

В этой модели надежность ПО на этапе эксплуатации оценивается по результатам тестирования.

Пусть T – суммарное время тестирования, n – число отказов, произошедших за время тестирования, тогда по модели Муса средняя наработка до отказа после тестирования на этапе эксплуатации определяется по формуле

$$\tau = \tau_0 \left(\frac{CT}{n\tau_0}\right), \tag{1.25}$$

где τ_0 – средняя наработка до отказа до начала тестирования; C – коэффициент, учитывающий уплотнение тестового времени по сравнению с временем реальной эксплуатации.

Например, если один час тестирования соответствует 12 ч работы в реальных условиях, то $C = 12$.

Неизвестный параметр τ_0 можно оценить из следующего соотношения:

$$\tau_0 = \frac{1}{NKf}, \tag{1.26}$$

где N – первоначальное число ошибок в ПО. Его можно оценить с помощью другой модели, позволяющей определить N на основе статистических данных, полученных при тестировании; K – коэффициент проявления ошибок. Значение K определяется эмпирическим путем по однотипным программам. Обычно это значение изменяется от $1{,}5\times10^{-7}$ до 4×10^{-7}; f – средняя скорость выполнения одного оператора программы, равная отношению средней скорости исполнения программного обеспечения (A) к числу команд (операторов) (B).

Надежность ПО для периода эксплуатации t определяется по формуле

$$R(t) = e^{-\frac{t}{x}}. \tag{1.27}$$

Преимущества и недостатки модели. К преимуществам модели можно отнести то, что нет необходимости фиксировать моменты от-

казов. В случае появления отказов ошибки регистрируются, а исправляются лишь по завершении этапа тестирования.

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

1.4. Статические модели оценки надежности

Статические модели отличаются от динамических прежде всего тем, что в них не учитывается время появления ошибок.

Модель Миллса

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

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

Пусть после тестирования обнаружено n собственных ошибок программы и v искусственно внесенных. Тогда первоначальное число ошибок в программе N можно оценить по формуле Миллса [6]:

$$N = n\frac{S}{v}, \tag{1.28}$$

где S – количество искусственно внесенных ошибок.

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

$$p = \begin{cases} 0, & \text{если } n > K, \\ \frac{S}{S+K+1}, & \text{если } n \le K. \end{cases} \tag{1.29}$$

Формулу (1.29) можно использовать только в случае, если обнаружены все S искусственно внесенных ошибок. Если же обнаружено только v искусственно внесенных ошибок, то применяют формулу

$$p = \begin{cases} 0, & \text{если } n > K, \\ \dfrac{C_S^{v-1}}{C_{S+K+1}^{K+v}}, & \text{если } n \le K, \end{cases} \qquad (1.30)$$

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

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

Однако есть недостатки:

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

б) достаточно вольное допущение величины K, которое основывается исключительно на интуиции и опыте человека, производящего оценку, то есть допускается большое влияние субъективного фактора.

Модель Нельсона

Модель [13] была разработана с учетом основных свойств машинных программ и практически не использует методы теории вероятности. Все приближения, принятые в модели, четко определены, и границы их применимости известны. Поскольку в основу модели Нельсона положены свойства программного обеспечения, она допускает развитие за счет более детального описания других аспектов надежности и может использоваться для расчета надежности программного обеспечения на всех этапах его жизненного цикла.

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

Пусть к моменту оценки надежности было выполнено n_i прогонов программного обеспечения на наборах данных из области Z_i, и n_i^- из этих прогонов закончились отказом.

Тогда надежность ПО оценивается по формуле

$$R = 1 - \sum_{i=1}^{k} \frac{n_i^-}{n_i} p_i. \quad (1.31)$$

Преимущества и недостатки модели. Основным преимуществом модели является то, что она была специально создана для определения надежности программного обеспечения, условия создания не исходили из теории надежности аппаратуры, как у других моделей (кроме модели Миллса), поэтому модель может использоваться для расчета надежности программного обеспечения на всех этапах его жизненного цикла.

Недостаток заключается в том, что на ранних стадиях использовать эту модель не очень удобно, так как для объективной оценки надежности требуется большое число прогонов ПО. Поэтому ниже рассмотрим модель Нельсона при расчете надежности на стадии эксплуатации.

Модель Коркорэна

Предполагает наличие в ПО многих источников программных отказов, связанных с различными типами ошибок, и разную вероятность их появления. Аргументом модели является число прогонов программы *n*. При этом оценка надежности ПО имеет вид

$$R(n) = \frac{n^+}{n} + \sum_{i=1}^{k} \delta_i \frac{n_i^- - 1}{n}, \quad (1.32)$$

где n^+ – число успешных прогонов программного обеспечения; n_i^- – число обнаруженных ошибок *i*-го типа, устраняемых с вероятностью p_i; δ_i – коэффициент, определяемый следующим образом:

$$\delta_i = \begin{cases} p_i, & \text{если } n_i^- > 0, \\ 0, & \text{если } n_i^- = 0. \end{cases} \quad (1.33)$$

Преимущества и недостатки модели. К преимуществам модели можно отнести то, что она учитывает существование в программном обеспечении нескольких источников ошибок, а также то, что расчет надежности с математической точки зрения проще, чем в других моделях. К недостаткам следует отнести необходимость определения статистическим методом вероятность того, что для очередного прогона программы будет выбран набор данных из предполагаемой облас-

ти, а это затрудняет расчеты. Поэтому обычно для расчетов надежности ПО используют **обобщенную модель Нельсона – Коркорэна**. После тестирования, на этапе эксплуатации ПО при росте числа прогонов n и выполнении условий $n^- \ll n$ и $n^+ + n^- = n$ формула определения надежности имеет вид

$$R(n) = 1 - \frac{n^-}{n}. \qquad (1.34)$$

1.5. Надежность сложных программных комплексов

Для удобства анализа показателей надежности сложных программных комплексов целесообразно представить их в виде совокупности менее сложных составляющих – программных модулей. Такими модулями могут быть программные комплексы, отдельные программы, блоки или операторы. Количество модулей в программном комплексе может быть слишком большим для обработки, поэтому чаще программные модули группируют по типам. Каждый тип содержит программные модули, близкие по свойствам, в том числе по надежности. По заданной структуре программного комплекса, состоящего из некоторой совокупности программных модулей, имеющих известные показатели надежности, существует возможность найти показатель надежности программного комплекса. Для этого используются так называемые графовые модели программы.

Модель Нельсона для графовой модели программы

Для завершающей стадии жизненного цикла программного комплекса следует использовать модель определения надежности с системно-независимым аргументом (количество прогонов ПО), например модель Нельсона. Однако практическое использование данной модели вызывает трудности, особенно для относительно больших программных комплексов массового применения, так как связывает оценку надежности программного обеспечения с количеством возможных программных маршрутов реализации вычислений и не рассматривает характеристики этих маршрутов.

Обобщенная модель

В общем случае сложный программный комплекс (СПК) представим состоящим из M отдельных программных модулей (ПМ), соединенных между собой вероятностными связями. По структуре СПК построим стохастический граф, который будет содержать M +2 вершин. Вершина 0 является истоком, а вершина M +1 – стоком графа. Каждый ПМ вызывается для решения им задачи с заданной вероятностью, определяемой исходя из целей функционирования или значений исходных данных. Задача заключается в нахождении вероятности безошибочного функционирования СПК по вероятностям безошибочного функционирования всех ПМ и вероятностям переходов между ними.

Достоинство метода состоит в том, что он позволяет производить оценку надежности сложных программных комплексов при известных показателях надежности составляющих модулей и их вероятностной зависимости в стохастическом графе.

Логико-вероятностный метод

Он заключается в том, что одновременно используется как аппарат теории вероятностей, так и аппарат алгебры логики высказываний. Это объясняется следующим: в настоящее время одним из перспективных направлений является разработка логико-вероятностных методов, математическая сущность которых заключается в использовании функций алгебры логики для аналитической записи условий работоспособности системы и в разработке строгих способов перехода от логических функций к вероятностным, объективно выражающим безотказность системы.

В основе сценарного логико-вероятностного подхода могут находиться безусловная вероятность, условная вероятность и характеристическая функция. Предлагаемый подход позволяет количественно оценить степень надежности системы.

Глава 2. Практичекие методы оценки надежности программного обеспечения

2.1. Метод оценки надежности, основанный на модели Джелински – Моранды

Данный метод представим в виде следующего алгоритма.

Алгоритм 2.1

Шаг 1. Тестирование программы. В таблицу «Результаты тестирования» записываем интервалы времени (часы) безотказной работы программы. При этом предполагается, что после каждого отказа ошибка должна исправляться, и при исправлении не вносятся новые ошибки (k – номер прогнозируемого отказа, t_i – длительность i-го интервала).

Результаты тестирования

Интервалы	t, ч
1	t_1
…	…
$k-1$	t_k

Шаг 2. Нахождение первоначального количества ошибок в программе (N). Данный параметр определяется методом подбора по формуле $(k-1)\frac{\sum_{i=1}^{k-1} t_i}{\sum_{i=1}^{k-1}\frac{1}{N-i+1}} = \sum_{i=1}^{k-1}(N-i+1)\,t_i$. Так как N может принимать только целые значения, то необходимо добиться минимальной разницы между правой и левой частями формулы.

Шаг 3. Определить интенсивность отказов: $\lambda_k = C_D(N-(k-1))$, где $C_D = \frac{\sum_{i=1}^{k-1}\frac{1}{N-i+1}}{\sum_{i=1}^{k-1} t_i}$ – коэффициент пропорциональности.

Шаг 4. Найти среднее время до следующего отказа: $t_k = \frac{1}{\lambda_k}$.

Шаг 5. Найти вероятность отсутствия k-го отказа: $P(t_k) = e^{-\lambda_k t_k}$.

Конец алгоритма.

Достоинство: относительная простота расчетов.

Недостатки: при неточном определении величины N интенсивность отказов программы может стать отрицательной, что приведет к

бессмысленному результату. Также предполагается, что при исправлении обнаруженных ошибок не вносятся новые, что тоже не всегда выполняется.

Пример 2.1. Расчет надежности программного обеспечения по модели Джелински – Моранды.

Пусть в ходе отладки зафиксированы интервалы времени, представленные в таблице (шаг 1 модели):

Интервалы	t_i, ч
1	10
2	20
3	25

Необходимо определить вероятность отсутствия следующего (четвертого) отказа.

Решение. Находим первоначальное количество ошибок N (шаг 2). Если $N = 3$, то в левой части формулы имеем $3\frac{10+20+25}{1/3+1/2+1} = 90$, а в правой части: 3·10 + 2·20 + 1·25 = 95. При $N = 4$ левая и правая части соответственно равны 152 и 150, если $N = 5$, то – 210 и 205. Следовательно, наименьшую ошибку при решении уравнения обеспечит $N = 4$.

Находим интенсивность отказов (шаг 3). Коэффициент пропорциональности $C_D = \frac{1/4+1/3+1/2}{10+20+25} = \frac{1{,}08}{55} = 0{,}02$. Интенсивность отказов $\lambda_4 = 0{,}02\left(4-(4-1)\right) = 0{,}02$.

Среднее время до следующего отказа (шаг 4) $t_4 = \frac{1}{0{,}02} = 50$ ч. Вероятность отсутствия четвертого отказа $P(t_4) = e^{-0{,}02\cdot 50} = e^{-1}$.

2.2. Метод оценки надежности, основанный на модели Шумана

Данный метод представим в виде следующего алгоритма.

Алгоритм 2.2

Шаг 1. Тестирование программы. Тестирование проводится в несколько этапов. Каждый этап представляет собой выполнение программы по набору тестовых данных. Выявленные в течение этапа тестирования ошибки регистрируются, но не исправляются. По завершении этапа исправляются все обна-

руженные на этапе ошибки, корректируются тестовые наборы и проводится новый этап тестирования. Результаты тестирования заносятся в таблицу (k – количество этапов тестирования, t_i – длительность i-го этапа тестирования, m_i – количество ошибок на i-м этапе тестирования).

Этапы	t, ч	m
1	t_1	m_1
…	…	…
k	t_k	m_k

Шаг 2. Найти методом подбора первоначальное количество ошибок N в программе из зависимости $\sum_{i=1}^{k} m_i \frac{\sum_{i=1}^{k} t_i}{\sum_{i=1}^{k} \frac{m_i}{N-n_{i-1}}} = = \sum_{i=1}^{k} (N - n_{i-1})\, t_i$ где n_{i-1} – число ошибок, исправленных к началу i-го этапа, т.е. $n_{i-1} = m_1 + \ldots + m_{i-1}$.

Шаг 3. Определить интенсивность отказов: $\lambda = (N - n)C$, где n – общее число обнаруженных и исправленных при тестировании ошибок, $C = \frac{\sum_{i=1}^{k} \frac{m_i}{N-n_{i-1}}}{\sum_{i=1}^{k} t_i}$ – коэффициент пропорциональности.

Шаг 4. Функцию надежности программы по завершении тестирования найти по формуле $R(t) = e^{-\lambda t}$.

Конец алгоритма.

Преимущества: можно определить все неизвестные параметры (надежность, интенсивность отказов, число оставшихся ошибок), т.е. нет необходимости обращаться к другим моделям, что сокращает время расчета надежности.

Недостаток: предполагается, что при корректировке не вносятся новые ошибки, а это не всегда так.

Пример 2.2. Расчет надежности программного обеспечения по модели Шумана.

Решение. Тестирование программы. По его результатам получены данные, представленные в таблице:

Этапы	t, ч	m
1	20	3
2	25	2
3	35	2

Методом подбора найдем, что первоначальное количество ошибок $N = 8$. Коэффициент пропорциональности $C = \frac{\frac{3}{8}+\frac{2}{8-3}+\frac{2}{8-(3+2)}}{20+25+35} = 1{,}802 \cdot 10^{-2}$. Интенсивность отказов $\lambda = 1{,}802 \cdot 10^{-2}\big(8-(3+2+2)\big) = = 1{,}802 \cdot 10^{-2}$. Находим функцию надежности программы по завершении тестирования $R(t) = e^{-1{,}802 \cdot 10^{-2} t}$.

2.3. Метод оценки надежности, основанный на модели Нельсона – Коркорэна

Данный метод представим в виде следующего алгоритма.

Алгоритм 2.3

Шаг 1. Тестирование программы. Программа «прогоняется» n раз. Подсчитывается число прогонов n^-, которые закончились отказом.

Шаг 2. Определяется надежность по формуле $R(n) = 1 - \frac{n^-}{n}$.

Конец алгоритма.

Достоинство: простота.

Недостаток: модель применима только при $n^- \ll n$.

Пример 2.3. Расчет надежности программного обеспечения по модели Нельсона – Коркорэна.

Решение. Тестирование программы. Общее число прогонов программы $n = 100$, число прогонов, закончившихся отказом, $n^- = 15$. Рассчитываем надежность $R(100) = 1 - \frac{15}{100} = 0{,}85$.

2.4. Метод расчета вероятности безошибочного функционирования сложного программного комплекса

Данный метод представим в виде следующего алгоритма.

Алгоритм 2.4

Шаг 1. Разбиваем сложный программный комплекс на программные модули, соединенные между собой вероятностными связями. Каждый ПМ вызывается для решения им задачи с заданной вероятностью, определяемой исходя из целей функционирования или значений исходных данных.

Шаг 2. По структуре СПК строим стахостический граф, который будет содержать $M + 2$ вершин. Вершина 0 является истоком, а вершина $M + 1$ – стоком графа. Пример графа представлен на рис. 2. ($P_i(t_i)$ – вероятность безошибочного функционирования i-го программного модуля в течение времени t_i, P_{ij} – веротность перехода от i-го модуля к j-му). В данном графе 0 – истоковая вершина, вершина 4 – стоковая, $t_0 = t_5 = 0, \times$ $\times P_0(t_0) = P_5(t_5) = 1$.

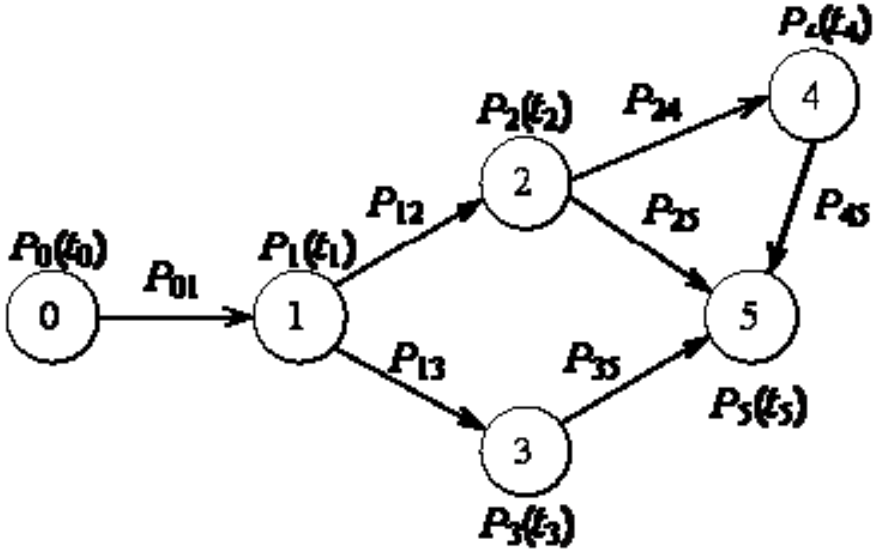

Рис. 2.1. Пример стохастического графа сложного программного комплекса

Шаг 3. По графу строим матрицу $G = G(t),\ t = (t_0, t_1, \ldots, t_{M+1})$, элементами которой являются произведения $p_{ij}P_i(t_i),\ i, j = = (0, \ldots, M + 1)$, где p_{ij} – вероятность перехода от i-го к j-му ПМ, а $P_i(t_i)$ – вероятность безошибочного функционирования i-го ПМ в течение времени t_i решения им задачи.

$$G = \begin{pmatrix} 0 & P_{01} & P_{02} & \ldots & P_{0M} & P_{0M+1} \\ 0 & 0 & P_{12}P_1(t_1) & \ldots & P_{1M}P_1(t_1) & P_{1M+1}P_1(t_1) \\ 0 & P_{21}P_2(t_2) & 0 & \ldots & P_{2M}P_2(t_2) & P_{2M+1}P_2(t_2) \\ \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\ 0 & P_{M1}P_M(t_M) & P_{M2}P_M(t_M) & \ldots & 0 & P_{M1}P_M(t_M) \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}.$$

Шаг 4. Вычисляем вероятность безошибочной работы СПК: $P(t) = Q(t)/R(t)$, где $Q(t)$ – алгебраическое дополнение элемента с номером (M + 1,0) матрицы ($I - G(t)$); $R(t)$ – главный определитель матрицы ($I - G(t)$); I – единичная матрица.

Конец алгоритма.

Выполнив указанные преобразования, получим искомые выражения для вероятности безошибочного функционирования программного комплекса с учетом всех возможных маршрутов вычислений.

Достоинства: данный метод позволяет производить оценку надежности сложных программных комплексов при известных показателях надежности составляющих модулей и их вероятностной зависимости в стохастическом графе.

Пример 2.4. Расчет вероятности безошибочного функционирования сложного программного комплекса.

Решение. При оценке надежности безотказной работы i-го модуля воспользуемся формулой из модели Шумана: $P_i(t_i) = e^{-\lambda_i t_i}$, где λ_i – интенсивность отказов i-го модуля; t_i – время работы i-го модуля.

Исходные данные. Длительности работы модулей: $t_1 = 1$ [illegible]; $t_2 = 7$ с; $t_3 = 10$ с. Интенсивности отказов модулей: $\lambda_1 = 0{,}008$; $\lambda_2 = 0{,}0087$; $\lambda_3 = 0{,}0049$. Вероятности переходов между модулями:

$P_{01} = 1$; $P_{12} = 0{,}7$; $P_{13} = 0{,}3$; $P_{23} = 0{,}6$; $P_{24} = 0{,}4$; $P_{31} = 0{,}8$; $P_{34} = 0{,}2$.

Находим вероятности безотказной работы каждого модуля исходя из модели Шумана: $P_0(t_0) = P_4(t_4) = 1$; $P_1(t_1) = e^{-0{,}008 \cdot 1} = 0{,}992$; $P_2(t_2) = 0{,}941$; $P_3(t_3) = 0{,}952$.

Граф СПК представлен на рис. 3.

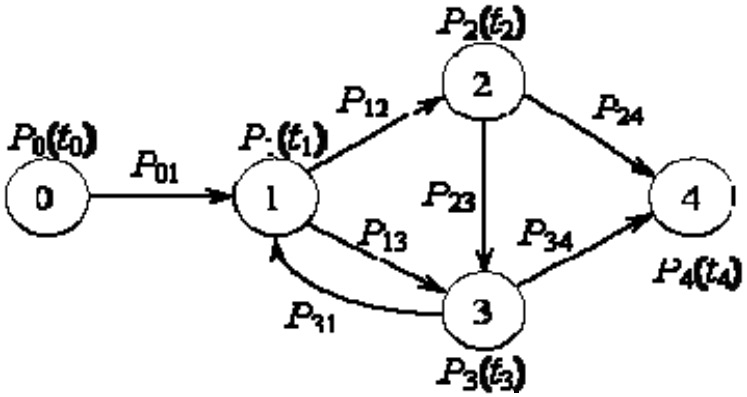

Рис. 2.2. Стохастический граф СПК

Строим матрицу:

$$G = \begin{pmatrix} 0 & P_{01}P_0(t_0) & 0 & 0 & 0 \\ 0 & 0 & P_{12}P_1(t_1) & P_{13}P_1(t_1) & 0 \\ 0 & 0 & 0 & P_{23}P_2(t_2) & P_{24}P_2(t_2) \\ 0 & P_{31}P_3(t_3) & 0 & 0 & P_{34}P_3(t_3) \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix},$$

$$G := \begin{pmatrix} 0 & 1 \cdot 1 & 0 & 0 & 0 \\ 0 & 0 & 0.7 \cdot 0.992 & 0.3 \cdot 0.992 & 0 \\ 0 & 0 & 0 & 0.6 \cdot 0.941 & 0.4 \cdot 0.941 \\ 0 & 0.8 \cdot 0.952 & 0 & 0 & 0.2 \cdot 0.952 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}.$$

Вычисляем вероятность безошибочной работы СПК:

$$I := \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}, \qquad I - G = \begin{pmatrix} 1 & -1 & 0 & 0 & 0 \\ 0 & 1 & -0.694 & -0.298 & 0 \\ 0 & 0 & 1 & -0.565 & -0.376 \\ 0 & -0.762 & 0 & 1 & -0.19 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix},$$

Находим алгебраическое дополнение элемента с номером (4,0) матрицы $(I - G(t))$:

$$|I - G| \cdot (I - G)^{-1} = \begin{pmatrix} 0.475 & 1 & 0.694 & 0.69 & 0.393 \\ 0 & 1 & 0.694 & 0.69 & 0.393 \\ 0 & 0.43 & 0.773 & 0.565 & 0.399 \\ 0 & 0.762 & 0.529 & 1 & 0.389 \\ 0 & 0 & 0 & 0 & 0.475 \end{pmatrix},$$

$Q = 0{,}393$.

Главный определитель матрицы $|I - G| = 0{,}475$.

$$R = 0{,}475;$$

$$P(t) = \frac{0{,}393}{0{,}475} = 0{,}827.$$

2.5. Оценка погрешности показателя надежности программного обеспечения

Естественно оценивать надежность ПО совокупностью методов и взять, например, среднее значение показателя надежности, полученное по множеству испытаний и трем методам (Джелински – Моранды, Шумана, Нельсона – Коркорэна).

Следующий алгоритм позволяет выделить этапы нахождения абсолютной (статистической) погрешности оцениваемого показателя надежности ПО.

Алгоритм 2.5

Шаг 1. Составляется N наборов данных для тестирования программы.

Шаг 2. Проводится расчет надежности программы для каждого из набора данных по методам Джелински – Моранды, Шумана, Нельсона-Коркорэна (в общем плане возможны и другие модели оценки надежности). Полученные результаты заносятся в таблицу.

Наборы данных	Джелински – Моранды	Шумана	…	Нельсона – Коркорэна
1	P_{11}	P_{21}		P_{K1}
…	…			…
N	P_{1N}	P_{1N}		P_{KN}

Шаг 3. Находится среднее значение $< P > = \frac{1}{nk}\sum_{\substack{i=1\\j=1}}^{n,k} P_{kn}$, дисперсия $\sigma^2 = \frac{1}{nk-1}\sum_{\substack{i=1\\j=1}}^{n,k}(P_i - < P >)^2$, среднее квадратичное отклонение (как квадратный корень из дисперсии), среднее квадратичное отклонение конечного результата $\sigma_{<p>} = \frac{\sigma}{\sqrt{n}}$.

Шаг 4. По таблице находим коэффициент Стьюдента (n – количество измерений; α – доверительная вероятность, $\alpha \approx \sqrt{1 - \exp\left(-\frac{2\varepsilon^2}{\pi}\right)}$, $\varepsilon = \frac{\Delta x}{\sigma}$).

Коэффициенты Стьюдента

n	α			
1	0,68	0,95	0,99	0,999
2	2,0	12,7	63,7	636,6
3	1,4	4,3	9,9	31,6
4	1,3	3,2	5,8	12,9
5	1,2	2,8	4,6	8,6
6	1,2	2,6	4,0	6,9
7	1,1	2,4	3,7	6,0
8	1,1	2,4	3,5	5,4
9	1,1	2,3	3,4	5,0
10	1,1	2,3	3,3	4,8
15	1,1	2,1	3,0	4,1
20	1,1	2,1	2,9	3,9
30	1,1	2,0	2,8	3,7
50	1,1	2,0	2,7	3,5
100	1,0	2,0	2,6	3,4

Шаг 5. Находятся случайная погрешность конечного результата $(\Delta x)_{\text{случ}} = t(\alpha, n)\,\sigma_{<x>}$ и абсолютная погрешность $\Delta x = (\Delta x)_{\text{случ}}$.

Конец алгоритма.

Пример 2.5. Расчет погрешности оценки показателя надежности программы.

Пусть в результате экспериментов были получены данные, представленные в таблице:

Набор данных	Джелински – Моранды	Шумана	Нельсона – Коркорэна
1	0,9	0,93	0,89
2	0,97	0,98	0,95
3	0,92	0,91	0,88
4	0,94	0,94	0,96
5	0,98	0,97	0,95

Среднее значение $< P >= 0{,}938$, дисперсия $\sigma^2 = 0{,}001046$, среднее квадратичное отклонение $\sigma = 0{,}032338$, среднее квадратич-

ное отклонение конечного результата $\sigma_{<p>} = 0{,}00835$. Доверительную вероятность принимаем $\alpha = 0{,}95$, $n = 15$. Коэффициент Стьюдента $t(\alpha, n) = 2{,}1$. Случайная погрешность $(\Delta x)_{\text{случ}} = 0{,}017534$, абсолютная погрешность $\Delta x = (\Delta x)_{\text{случ}} = 0{,}017534$.

Таким образом, надежность тестируемой программы $P = 0{,}938 \pm \pm 0{,}0175$.

Глава 3. Примеры практической оценки надежности стандартного программного обеспечения

3.1. Процедура тестирования

Тестирование ПО для оценки его надежности проведем в соответствии с алгоритмом, представленными ниже.

Алгоритм 3.1

Шаг 1. Описание условий проведения тестирования.

Шаг 2. Определение наборов входных данных для тестирования, необходимого числа измерений (либо времени тестирования) для каждого набора данных.

Шаг 3. Тестирование. Проводится для каждого из набора данных по методикам Джелински – Моранды, Шумана и Нельсона – Коркорэна. В процессе проведения тестирования указываются условия и особенности проведения тестирования.

Шаг 4. Обработка полученных данных, включая расчет погрешностей.

Конец алгоритма.

Для обработки результатов, полученных в результате тестирования, было создано специальное программное обеспечение. В его возможности входит:

- расчет надежности отдельных программ по экспериментальным данным;

- расчет надежности сложных программных комплексов.

Входными данными для данной программы являются данные, полученные в результате тестирования ПО, выходными – результаты расчетов: интенсивность отказов, среднее время до следующего отказа, вероятность безотказной работы и др.

3.2. Оценка надежности веб-серверов Apache и IIS

Для определения характеристик надежности были выбраны два наиболее популярных веб-сервера - Apache под linux и Internet Information Services (IIS) под Windows.

Тестирование Apache

1. Конфигурация сервера.

Аппаратная часть: процессор Intel(R) Core(TM) 2DuoCPU E4600@2.4 GHz; оперативная память 2048 Мб; материнская плата Gigabyte GA-P35-DS3L.

Программное обеспечение: ОС Debian 5.1 CNU GPL; Apache httpd 2.2.9-10+lenny6; mysql 5.0.51a – 24; postgresql – 8.4; oracle-xe 10.2.0.1-1.1; php5 5.2.6.dfsg.1-1; ОС Windows XP Service Pack 3; Internet Information Services 5.1.

Настройки Apache: StartServers – 2; MaxClients – 150; MinSpareThreads – 25; MaxSpareThreads – 75; ThreadsPerChild – 25; MaxRequestPerChild – 0.

2. В качестве нагрузки используется «пустая» html страница.

3. Тестирование (с помощью программы Jakarta JMeter) проводилось в 20 этапов по одной минуте каждый, с постепенным увеличением числа запросов от 1000 до 20000 в минуту.

4. Усредненные результаты тестирования: $< P > = 0{,}98$; $\sigma^2 = 0{,}0000410$; $\sigma_{<p>} = 0{,}0014311$. Приняв $\alpha = 0{,}95$ при $n = 20$ $t(\alpha, n) = 2.1$. $(\Delta x)_{\text{случ}} = 0{,}0030052$. Вероятность безотказной работы системы при данных режимах нагрузки равна $P = 0{,}98 \pm 0{,}0030052$.

Тестирование IIS

1. Конфигурация сервера.

Аппаратная часть: процессор Intel(R) Core(TM) 2DuoCPU E4600@2.4 GHz; оперативная память 2048 Мб; материнская плата Gigabyte GA-P35-DS3L.

Программное обеспечение: ОС Debian 5.1 CNU GPL; Apache httpd 2.2.9-10+lenny6; mysql 5.0.51a – 24; postgresql – 8.4; oracle-xe 10.2.0.1-1.1; php5 5.2.6.dfsg.1-1; ОС Windows XP Service Pack 3; Internet Information Services 5.1.

Настройки Apache: StartServers – 2; MaxClients – 150; MinSpareThreads – 25; MaxSpareThreads – 75; ThreadsPerChild – 25; MaxRequestPerChild – 0.

2. В качестве нагрузки используется «пустая» html страница.

3. Тестирование (с помощью программы Jakarta JMeter) проводилось в 20 этапов по одной минуте каждый, с постепенным увеличением числа запросов от 1000 до 20000 в минуту.

4. Усредненные результаты тестирования: $< P > = 0{,}48$; $\sigma^2 = 0{,}15$; $\sigma_{<p>} = 0{,}085$; $(\Delta x)_{\text{случ}} = 0{,}18$. Вероятность безотказной работы системы при данных режимах нагрузки равна $P = 0{,}48 \pm 0{,}18$.

В процессе тестирования Apache показал высокую надежность и стабильные результаты, IIS – достаточно низкую надежность. Так, при 1000 запросах в минуту надежность составляет 0,98, а при 20000 – 0,071. Резкое падение надежности наблюдалось при 8000 запросах в минуту (с 0,95 до 0,56). Это можно объяснить нехваткой процессорных ресурсов, так как уже при 7000 запросах в минуту загрузка процессора составляла 90 %.

3.3. Оценка надежности баз данных mysql, postgresql и oracle

Тестирование MySQL

1. Объект – база данных

```
CREATE TABLE ‘sbtest‘ (
‘id‘ int(10) unsigned NOT NULL auto_increment,
‘k‘ int(10) unsigned NOT NULL default `0`,
‘c‘ char(120) NOT NULL default ``,
‘pad‘ char(60) NOT NULL default ``,
PRIMARY KEY (‘id‘),
KEY ‘k‘ (‘k‘);
```

Число записей – 500000. База данных была установлена на сервере с конфигурацией:

- процессор Intel(R) Core(TM) 2DuoCPU E8400@3/006 Hz; оперативная память 2048 Мб; материнская плата Intel Corporation 82Q33;

- программное обеспечение: ОС Debian 5.1 CNU GPL; Apache httpd 2.2.9; mysql 5.0.51a – 24; php5 5.2.10.dfsg.1-2 ubur;

- настройки Apache: StartServers – 2; MaxClients – 150; MinSpareThreads – 25; MaxSpareThreads – 75; ThreadsPerChild – 25; MaxRequestPerChild – 0.

2. Тестирование проводилось в режимах Simple, Advanced transactional (complex) и Non-transactional, которые характеризуются следующими запросами:

Simple

In this mode each thread runs simple queries of the following form:

```
SELECT c FROM sbtest WHERE id=N
```

where N takes a random value in range 1..<table size>

Advanced transactional

- Point queries:

```
SELECT c FROM sbtest WHERE id=N
```

- Range queries:

```
SELECT c FROM sbtest WHERE id BETWEEN N AND M
```

- Range SUM() queries:

```
SELECT SUM(K) FROM sbtest WHERE id BETWEEN N and M
```

- Range ORDER BY queries:

```
SELECT c FROM sbtest WHERE id between N and M ORDER BY c
```

- Range DISTINCT queries:

```
SELECT DISTINCT c FROM sbtest WHERE id BETWEEN N and M ORDER BY c
```

- UPDATEs on index column:

```
UPDATE sbtest SET k=k+1 WHERE id=N
```

- UPDATEs on non-index column:

```
UPDATE sbtest SET c=N WHERE id=M
```

- DELETE queries:

```
DELETE FROM sbtest WHERE id=N
```

- INSERT queries:

```
INSERT INTO sbtest VALUES (...)
```

Non-transactional

- Point queries:

```
SELECT pad FROM sbtest WHERE id=N
```

- UPDATEs on index column:

```
UPDATE sbtest SET k=k+1 WHERE id=N
```

- UPDATEs on non-index column:

```
UPDATE sbtest SET c=N WHERE id=M
```

- DELETE queries:

```
DELETE FROM sbtest WHERE id=N
```

Отказом считались все запросы, время которых превысило ($t_{ср} + t_{ср}$·0,5), где $t_{ср}$ – среднее время запроса для данного режима.

3. Время тестирования ограничивалось суммарным количеством запросов (по всем потокам). Суммарное количество запросов – 10000. Тестирование проводилось с помощью программы sysbench.

4. Усредненные результаты тестирования: $< P > = 0{,}65$; $\sigma^2 = 0{,}08$; $(\Delta x)_{случ} = 0{,}077$; вероятность безотказной работы системы при данных режимах нагрузки $P = 0{,}65 \pm 0{,}077$.

Тестирование Postgresql

1. Объект – база данных

```
CREATE TABLE ‘sbtest‘ (
‘id‘ int(10) unsigned NOT NULL auto_increment,
‘k‘ int(10) unsigned NOT NULL default `0`,
‘c‘ char(120) NOT NULL default ``,
‘pad‘ char(60) NOT NULL default ``,
PRIMARY KEY  (‘id‘),
KEY ‘k‘ (‘k‘);
```

Число записей – 500000. База данных установлена на сервере с конфигурацией:

- аппаратная часть: процессор Intel(R) Core(TM) 2DuoCPU E4600@2.4 GHz; оперативная память 2048 Мб; материнская плата Gigabyte GA-P35-DS3L;

- программное обеспечение: ОС Debian 5.1 CNU GPL; Apache httpd 2.2.9-10+lenny6; mysql 5.0.51a – 24; postgresql – 8.4; oracle-xe 10.2.0.1-1.1; php5 5.2.6.dfsg.1-1; ОС Windows XP Service Pack 3; Internet Information Services 5.1;

- настройки Apache: StartServers – 2; MaxClients – 150; MinSpareThreads – 25; MaxSpareThreads – 75; ThreadsPerChild – 25; MaxRequestPerChild – 0.

2. Тестирование проводилось в режимах Simple, Advanced transactional (complex) и Non-transactional, которые характеризовались следующими запросами:

Simple

In this mode each thread runs simple queries of the following form:

```
SELECT c FROM sbtest WHERE id=N
```

where N takes a random value in range 1..<table size>

Advanced transactional

- Point queries:

```
SELECT c FROM sbtest WHERE id=N
```

- Range queries:

```
SELECT c FROM sbtest WHERE id BETWEEN N AND M
```

- Range SUM() queries:

```
SELECT SUM(K) FROM sbtest WHERE id BETWEEN N and M
```

- Range ORDER BY queries:

```
SELECT c FROM sbtest WHERE id between N and M ORDER BY c
```

- Range DISTINCT queries:

```
SELECT DISTINCT c FROM sbtest WHERE id BETWEEN N and M ORDER BY c
```

- UPDATEs on index column:

```
UPDATE sbtest SET k=k+1 WHERE id=N
```

- UPDATEs on non-index column:

```
UPDATE sbtest SET c=N WHERE id=M
```

- DELETE queries:

```
DELETE FROM sbtest WHERE id=N
```

- INSERT queries:

```
INSERT INTO sbtest VALUES (...)
```

Non-transactional

- Point queries:

```
SELECT pad FROM sbtest WHERE id=N
```

- UPDATEs on index column:

```
UPDATE sbtest SET k=k+1 WHERE id=N
```

- UPDATEs on non-index column:

```
UPDATE sbtest SET c=N WHERE id=M
```

- DELETE queries:

```
DELETE FROM sbtest WHERE id=N
```

Отказом считались все запросы, время которых превысило ($t_{ср} + t_{ср} \cdot 0{,}5$), где $t_{ср}$ – среднее время запроса для данного режима.

3. Время тестирования ограничивалось суммарным количеством запросов (по всем потокам). Суммарное количество запросов – 10000. Тестирование проводилось с помощью программы sysbench.

4. Усредненные результаты тестирования: $< P > = 0{,}32$; $\sigma^2 = 0{,}12$; $(\Delta x)_{случ} = 0{,}092$; вероятность безотказной работы системы при данных режимах нагрузки $P = 0{,}2 \pm 0{,}1$.

Тестирование Oracle

1. Объект – база данных

```
CREATE TABLE ‘sbtest‘ (
‘id‘ int(10) unsigned NOT NULL auto_increment,
‘k‘ int(10) unsigned NOT NULL default `0`,
‘c‘ char(120) NOT NULL default ``,
‘pad‘ char(60) NOT NULL default ``,
PRIMARY KEY  (‘id‘),
KEY ‘k‘ (‘k‘);
```

Число записей 500000. База данных установлена на сервере с конфигурацией как для Postgresql.

2. Тестирование проводилось в режимах Simple, Advanced transactional (complex) и Non-transactional, которые характеризовались следующими запросами:

Simple

In this mode each thread runs simple queries of the following form:

```
SELECT c FROM sbtest WHERE id=N
```

where N takes a random value in range 1..<table size>

Advanced transactional

- Point queries:

```
SELECT c FROM sbtest WHERE id=N
```

- Range queries:

```
SELECT c FROM sbtest WHERE id BETWEEN N AND M
```

- Range SUM() queries:

```
SELECT SUM(K) FROM sbtest WHERE id BE-
TWEEN N and M
```

- Range ORDER BY queries:

```
SELECT c FROM sbtest WHERE id between N
and M ORDER BY c
```

- Range DISTINCT queries:

```
SELECT DISTINCT c FROM sbtest WHERE id
BETWEEN N and M ORDER BY c
```

- UPDATEs on index column:

```
UPDATE sbtest SET k=k+1 WHERE id=N
```

- UPDATEs on non-index column:

```
UPDATE sbtest SET c=N WHERE id=M
```

- DELETE queries:

```
DELETE FROM sbtest WHERE id=N
```

- INSERT queries:

```
INSERT INTO sbtest VALUES (...)
```

Non-transactional

- Point queries:

```
SELECT pad FROM sbtest WHERE id=N
```

- UPDATEs on index column:

```
UPDATE sbtest SET k=k+1 WHERE id=N
```

- UPDATEs on non-index column:

```
UPDATE sbtest SET c=N WHERE id=M
```

- DELETE queries:

```
DELETE FROM sbtest WHERE id=N
```

Отказом считались все запросы, время которых превысило ($t_{ср} + t_{ср} \cdot 0{,}5$), где $t_{ср}$ – среднее время запроса для данного режима.

3. Время тестирования ограничивалось суммарным количеством запросов (по всем потокам). Суммарное количество запросов – 10000. Тестирование проводилось с помощью программы sysbench. Заметим, что в режиме Complex тестирование невозможно:

debian:/home/stud# sysbench --test=oltp --db-driver=oracle --oracle-user=system --oracle-db=XE --oracle-password=oracle --oltp-table-size=500000 --oltp-test-mode=complex --num-threads=5 run

sysbench 0.4.10: multi-threaded system evaluation benchmark

Running the test with following options:

Number of threads: 5

Doing OLTP test.

Running mixed OLTP test

Using Special distribution (12 iterations, 1 pct of values are returned in 75 pct cases)

Using "BEGIN" for starting transactions

Using auto_inc on the id column

Maximum number of requests for OLTP test is limited to 10000

Threads started!

Ошибка сегментирования

4. Усредненные результаты тестирования: $< P > = 0{,}344$; $\sigma^2 = 0{,}10$; $(\Delta x)_{\text{случ}} = 0{,}086$; вероятность безотказной работы системы при данных режимах нагрузки $P = 0{,}344 \pm 0{,}086$.

Выводы: наилучшие результаты показал mysql, для него характерны высокая надежность по сранению с postgresql и oracle, а также постепенное снижение надежности при увеличенни нагрузки. Postgresql и oracle показали значительно худшие результаты. Postgesql обладает достаточно высокой надежностью до значения нагрузи в 35 – 40 потоков, после чего наблюдается ее резкое падение. У oracle, как и у mysql, надежность падает постепенно, по мере увеличения нагрузки, однако более интенсивно. При тестировании всех баз данных загрузка процессора не составляла более 70 %. Поэтому падение надежности можно обьяснить увеличением выполнения всех запросов в целом. Для postgresql и oracle характерны критические запросы с очень большим временем выполнения. Так, максимальное время запроса при тестировании postgresql составило 31285 мс, при этом время выполнения 95 % запросов не превышало 125 мс.

3.4. Оценка надежности PHP

1. В качестве наборов данных использовалось различное число пользователей (потоков), от которых поступали запросы. Продолжительность тестирования для каждого из «наборов данных» 1 минута.

2. Так как нарпямую протестировать php не представлялось возможным, то тестирование проводилось по аналогии с тестированием веб-серверов. Отличие заключается лишь в наполнении сервера. В данном случае тестировался скрипт phpinfo(). Для исключения влиянии веб-сервера нагрузка определялась следующим образом:

Нагрузка PHP = Нагрузка (Apache+PHP) – Нагрузка (Apache).

3. Тестирование проводилось с помощью программы Jakarta JMeter.

4. Усредненные результаты тестирования: вероятность безотказной работы системы при данных режимах нагрузки $P = 0{,}56 \pm 0{,}122628$.

Низкую надежность можно объяснить нехваткой процессорных ресурсов, так как уже при 10000 запросах в минуту загрузка составляет 70 %.

3.5. Оценка надежности сложного программного комплекса Apache + PHP + mysql

Тестирование проводилось аналогично тестированию веб-серверов с помощью программы JMeter. В качестве объекта тестирования был выбран сайт www.izi.vlsu.ru.

Вероятность безотказной работы системы при выбранных режимах нагрузки $P = 0{,}169 \pm 0{,}069$.

Эксперимент проводился на реальной системе, поэтому на точность результатов могли сказаться действия обычных пользователей.

По полученным данным проведем расчет надежности (Apache + +PHP + MySQL) как сложного программного комплекса в соответствии с методом, описанным в гл. 2.

Расчет надежности СПК (Apache + PHP + MySQL).

Схема информационного обмена СПК представлена на рис. 4.

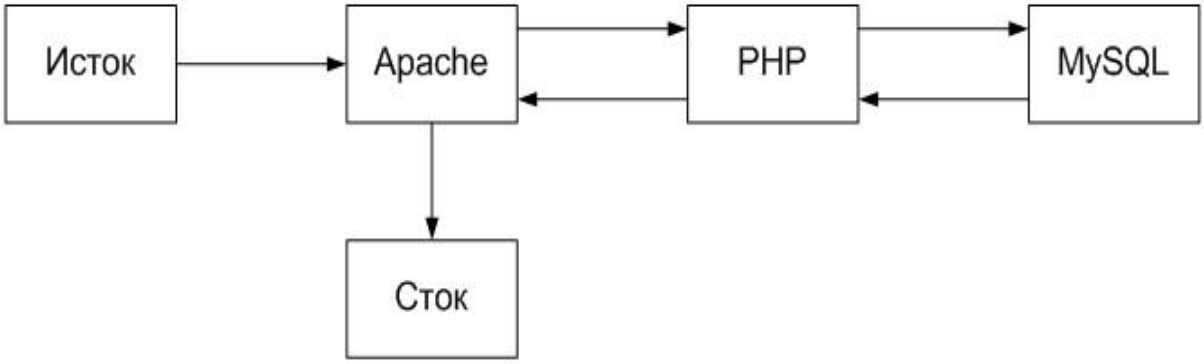

Рис. 4. Схема информационного обмена СПК

Стохастический граф СПК представлен на рис. 5.

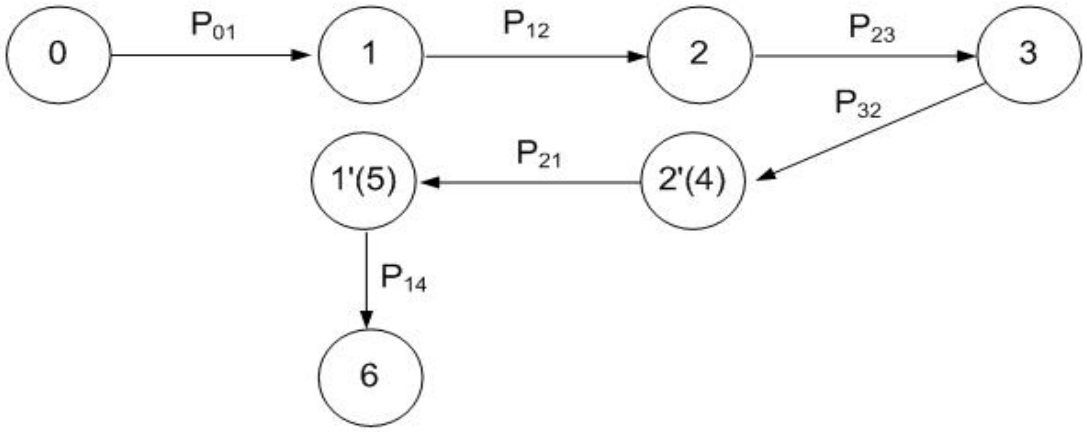

Рис. 5. Стохастический граф СПК

Исходные данные:

Так как содержимое сайта хранилось в базе данных, то при запросе страницы были задействованы все модули. Поэтому вероятности переходов между модулями принимаем равными единице. Вероятности безотказной работы каждого модуля: $P_0(t_0) = P_4(t_4) = 1$; $P_1(t_1) = 0{,}98$; $P_2(t_2) = 0{,}56$; $P_3(t_3) = 0{,}65$.

Строим матрицу

$$G := \begin{bmatrix} 0 & P_{01}P_0(t_0) & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & P_{12}P_1(t_1) & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & P_{23}P_2(t_2) & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & P_{34}P_3(t_3) & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & P_{45}P_4(t_4) & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & P_{56}P_5(t_5) \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix},$$

$$G := \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.982 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.5624 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.65 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.5624 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0.982 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Вычисляем вероятность безошибочной работы СПК.

Находим алгебраическое дополнение элемента с номером (6,0) матрицы $(I - G(t))$; $Q = 0{,}198$. Главный определитель матрицы $(I - G(t))\ R = 1$. $P(t) = \frac{0{,}198}{1} = 0{,}198$.

Выводы.

Вероятность безотказной работы СПК (Apache + PHP + MySQL), полученная в результате тестирования ($P_{\text{тест}} = 0{,}169$) практически совпадает с результатом, полученным при моделировании ($P_{\text{мод}} = 0{,}198$). Это дает основание полагать, что модель достаточно точно описывает реальную систему.

Низкую вероятность безотказной работы можно объяснить нехваткой процессорных ресурсов для одновременного выполнения процессов, так как уже при 10000 запросах в минуту во время тестирования PHP.

Задания для самостоятельной работы

1. Соотнесите описанные в книге модели надежности с известными вам методиками тестирования. Укажите ситуации, в которых использование приведенных методов оценки надежности программного обеспечения невозможно или затруднено.

2. Приведите случаи, в которых обобщенная модель Нельсона – Коркорэна будет давать неудовлетворительный результат.

3. Рассмотрите зависимость готовности программного средства от информативности сообщений об ошибках, выдаваемых его интерфейсом.

4. Проведите сравнительный анализ методик оценки надежности проприетарных программных продуктов и продуктов с открытым исходным кодом.

5. Проанализируйте целесообразность оперативной оценки надежности программного обеспечения на разных этапах его жизненного цикла.

6. Рассмотрите возможность применения моделей оценки надежности сложного программного комплекса для любых программных средств с ярко выраженной модульностью архитектуры.

7. Выявите характер изменения показателя надежности по модели Джелински – Моранды в ходе устаревания исследуемого программного обеспечения.

8. Проведите экспериментальное исследование функции распределения отказов произвольного программного средства. Укажите, всегда ли данная функция представляет собой нормальную.

9. Проанализируйте правомерность экстраполяции результатов исследования надежности программного обеспечения на длительные промежутки времени.

10. Определите физический смысл показателей надежности из ГОСТ 27.002-89 применительно к современному программному обеспечению.

11. Предложите формальное описание оптимизационной задачи по повышению надежности программного обеспечения

12. Оцените сложность построения бинарного дерева прецедентов для логико-вероятностного подхода оценки надежности.

13. Приведите разумные на ваш взгляд требования к надежности произвольного общеупотребительного программного средства, выра-

женные в допустимых значениях показателей, приведенных в данной книге.

14. Проведите экспериментальное исследование степени влияния протоколов обмена данными на надежность сервисов. Обращайте особое внимание на время обработки тестовых данных.

Заключение

Быстрый рост сфер использования, сложности и ответственности функций, выполняемых ПО ИС, существенно повысил в последнее время требования к надежности их функционирования. Для удовлетворения данных требований в жизненном цикле (ЖЦ) ПО необходимы выделение задач и работ по обеспечению надежности программ и концентрация усилий разработчиков на теоретическом и практическом их решении. Для каждого проекта ПО ИС должны разрабатываться и применяться специальные план и программа, методология и инструментальные средства, предупреждающие и выявляющие дефекты, а также удостоверяющие надежность программ. Для систематической, координированной борьбы с угрозами надежности должны проводиться исследования конкретных факторов, влияющих на качество функционирования программ со стороны реально существующих и потенциально возможных дефектов в создаваемых комплексах программ. В каждом проекте должен целенаправленно разрабатываться скоординированный комплекс методов и средств обеспечения заданной надежности функционирования ПО при реально достижимом снижении уровня дефектов и ошибок разработки. Учет факторов, влияющих на затраты ресурсов при создании конкретного ПО, должен позволять рационализировать их использование и добиваться заданной надежности функционирования ПО при минимальных или допустимых затратах.

Требуемый уровень надежности ПО достигается в процессе его создания и контроля. Основные принципы разработки (в процессе создания) могут быть объеденены в четыре группы: избежание, обнаружение, исправление и допуск ошибок.

Избежание ошибок включает описание принципов, выполнение которых обеспечивает минимизацию ошибок, возникающих в процессе создания ПО. *Обнаружение ошибок* базируется на механизмах,

обеспечивающих исправление ошибок. *Допуск ошибки* предусматривает средства и приемы, обеспечивающие выполнение заданной работы при наличии ошибок.

Обеспечение надежности в процессе контроля схематично представляется следующей схемой: ПО разрабатывается как совокупность формально самостоятельных частей – программных модулей. Модули разрабатываются и тестируются с учетом удовлетворения частных задач и требований на их реализацию. Если в процессе тестирования достигается заданная степень надежности программных модулей, то можно переходить к их комплексированию и испытаниям ПО в целом. Если в процессе тестирования обнаружились ошибки в составных частях или в ПО в целом, то производится доработка составных частей или ПО в целом. Тестирование продолжается до тех пор, пока не будет достигнут заданный показатель надежности.

Формальная процедура выбора необходимого сочетания методов повышения надежности ПО производится на основе зависимости

$$P(\text{ПО}) = \min\{P(C(\text{ПМ}_{1j})), P(C(\text{ПМ}_{2j})),...,P(C(\text{ПМ}_{kj}))\},$$

где $C(\text{ПМ}_{ij})$ – ресурсы, затрачиваемые на *i*-й механизм повышения надежности *j*-го программного модуля (ПМ);

$$P(\text{ПМ}_j) \to \max;\ \sum_i^k C(M_{ij}) \le C_{\text{o}j};\ X_i(M_{1j}, M_{2j},..., M_{kj}) \le X_{i\text{o}},$$

где $C_{\text{o}j}$ – общее количество ресурсов на повышение надежности *j*-го программного модуля; X_i – *i*-я характеристика ПМ, которая не должна быть хуже заданного значения.

Решая задачу оптимизации при различных значениях $C_{\text{o}j}$, получаем зависимость надежности от совокупной стоимости ПО.

Для обеспечения надежности ПО в конкретных проектах должны быть организованы и стимулированы разработка, освоение и применение современных автоматизированных технологий и инструментальных средств, обеспечивающих предупреждение или исключение большинства видов дефектов и ошибок при создании и модификации ПО и его компонент. Ограниченные ресурсы на разработку приводят к необходимости упорядоченного применения методов и рационального использования средств автоматизации проектирования. Поэтому процесс разработки должен планироваться и последовательно прохо-

дить этапы, охватывающие все компоненты ПО. Контроль надежности создаваемых и модифицируемых программ необходим на всем ЖЦ ПО посредством специальной технологической системы обеспечения их качества. Предупреждение ошибок должно поддерживаться также высококачественной документацией в процессе создания ПО в целом и их компонент.

Один из эффективных путей повышения надежности ПО – стандартизация технологических процессов и объектов проектирования, разработки и сопровождения программ. В стандартах ЖЦ ПО обобщаются опыт и результаты исследований множества специалистов и рекомендуются наиболее эффективные современные методы и процессы. В результате таких обобщений отрабатываются технологические процессы и приемы разработки, а также методическая база для их автоматизации. Стандарты ЖЦ ПО могут использоваться как непосредственные директивные, руководящие или как рекомендательные документы, а также как организационная база при создании средств автоматизации соответствующих технологических этапов или процессов. Подобная стандартизация процессов отражается не только на их технико-экономических показателях, но и, что особенно важно, на качестве создаваемого ПО. Надежность программ тесно связана с методами и технологией их разработки, поэтому важной группой стандартов в этой области являются стандарты по обеспечению качества ПО.

Поддержка этапов и работ ЖЦ ПО международными стандартами весьма неравномерная. Наиболее полно стандартизированы этапы ЖЦ ПО, прошедшие длительное историческое развитие и требующие наименее квалифицированных специалистов. При создании сложных проектов ПО и обеспечении их ЖЦ целесообразно применять выборку из всей совокупности существующих стандартов, а имеющиеся весьма обширные пробелы в стандартизации заполнять утвержденными технологическими документами, регламентирующими применение выбранных средств автоматизации разработки ПО. В результате на начальном этапе проектирования следует формировать весь комплект документов – профиль, обеспечивающий регламентирование всех этапов и работ при создании надежных ПО. Для реализации положений этих документов должны быть выбраны инструментальные средства, совместно образующие взаимосвязанный комплекс

технологической поддержки и автоматизации ЖЦ и не противоречащие предварительно скомпонованному набору нормативных документов профиля. Применение профилей при проектировании ПО позволяет ориентироваться на построение систем из крупных функциональных узлов, отвечающих требованиям стандартов профиля, применять достаточно отработанные и проверенные проектные методы и решения.

Для обнаружения и устранения ошибок проектирования все этапы разработки и сопровождения ПО должны поддерживаться методами и средствами систематического, автоматизированного тестирования и испытаний. При разработке ПО целесообразно применять различные методы, эталоны и виды тестирования, каждый из которых ориентирован на обнаружение, локализацию или диагностику определенных типов дефектов. Достигнутому качеству и надежности функционирования сложных, критических программных комплексов должна сопутствовать обязательная сертификация, проводимая аттестованными проблемно-ориентированными сертификационными лабораториями.

В отечественных ИС все больше применяются программные компоненты зарубежных фирм, которые также не могут быть абсолютно гарантированы от проявления дефектов проектирования, программирования и документации. Для обеспечения надежности функционирования комплексов программ с использованием «импортных» компонент следует закупать только лицензионно-чистые продукты, поддерживаемые гарантированным сопровождением конкретных фирм-поставщиков. Необходимо чтобы эти компоненты сопровождались полной эксплуатационной и технической документацией, сертификатом соответствия и комплектами тестов. В контрактах на закупку должны специально фиксироваться обязательства поставщиков по длительному сопровождению и замене версий ПО при выявлении дефектов или совершенствовании функций. Все версии зарубежных ПО следует проверять на надежность функционирования в конкретном окружении проекта ИС путем повторных испытаний или отдельными проверками, подтверждающими зарубежный сертификат.

Рекомендательный библиографический список

1. Азовцев, В. В. Исследование методов оценки и повышения надежности программного обеспечения [Электронный ресурс] / В. В. Азовцев. – Загл. с экрана. – URL: http://www.azovikdip.ru/ (дата обращения: 16.06.2011).

2. Беляев, Ю. К. Надежность технических систем : справочник / Ю. К. Беляев, В. А. Богатырев, В. В. Болотин. – М. : Радио и связь, 1985. – 608 с.

3. Грэм, П. Использование статического и динамического анализа для повышения качества продукции и эффективности разработки [Электронный ресурс] / П. Грэм, Н. Леру, Т. Ландри // Операционная система реального времени QNX. – Загл. с экрана. – URL: http://www.swd.ru/ index.php3/pid=828 (дата обращения: 16.06.2011).

4. Василенко, Н. В. Модели оценки надежности программного обеспечения / Н. В. Василенко, В. А. Макаров // Вестник Новгор. гос. ун-та. – 2004. – № 28. – С. 126 – 132.

5. Викторова, В. С. Агрегирование моделей анализа надежности и безопасности технических систем сложной структуры : автореф. дис. ... д-ра техн. наук / Викторова В. С. ; ИПУ. – М., 2009. – 45 с.

6. Вихарев, С. М. Надежность автоматизированных систем : учеб. пособие / С. М. Вихарев, А. А. Баринов. – Кострома : Изд-во КГТУ, 2007. – 80 с. – ISBN 978-5-8285-0305-6.

7. Громов, Ю. Ю. Надёжность информационных систем : учеб. пособие / Ю. Ю. Громов [и др.]. – Тамбов : Изд-во ТГТУ, 2010. – 160 с. – ISBN 978-5-8265-0911-1.

8. Гнеденко, Б. В. Математические методы в теории надежности / Б. В. Гнеденко, Ю. К. Беляев, А. Д. Соловьев. – М. : Наука, 1965. – 524 с.

9. Гуров, С. В. Методы и модели анализа надежности сложных технических систем с переменной структурой и произвольными законами распределения случайных параметров, отказов и восстановлений : автореф. дис. ... канд. техн. наук / Гуров С. В. – СПб., 2001. – 18 с.

10. Дружинин, Г. В. Надежность автоматизированных систем / Г. В. Дружинин. – М. : Энергия, 1977. – 536 с.

11. Ермаков, А. А. Основы надежности информационных систем : учеб. пособие / А. А. Ермаков. – Иркутск : ИрГУПС, 2006. – 151 с.

12. Задорожный, В. Н. Применение редукции для расчета надежности структурно-сложных систем / В. Н. Задорожный, А. В. Ли // Надежность и контроль качества. – 1997. – № 10. – С. 35 – 38.

13. Зайцева, Е. Н. Оценка вероятности отказа невосстанавливаемой системы с использованием методов алгебры логики / Е. Н. Зайцева, Ю. В. Поттосин // Информатика. – 2007. – № 2. – С. 77 – 85.

14. Иыуду, К. А. Надежность, контроль и диагностика вычислительных машин и систем : учеб. пособие для вузов по специальности "Вычислительные машины, комплексы, системы и сети" / К. А. Иыуду. – М. : Высш. шк., 1989 . – 216 с. – ISBN 5-06-000130-X .

15. Кабак, И. С. О надежности объектно-ориентированного программмного обеспечения [Электронный ресурс] / И. С. Кабак, Б. М. Позднеев. – М. : МГТУ СТАНКИН. – Загл. с экрана. – URL: http://www.stankin.ru/ rus_ver/sc_prj/magazine/art/10/index.html (дата обращения 16.06.2011).

16. Канер С. Тестирование программного обеспечения / С. Канер, Дж. Фолк, Е. К. Нгуен. – 2-е изд., стер. – Киев : ДиаСофт, 2000. – 543 с. – ISBN 966-7393-42-9.

17. Карповский, Е. Я. Надежность программной продукции / Е. Я. Карповский, С. А. Чижов. – Киев : Тэхника, 1990. – 159 с. – ISBN 5-335-00586-6.

18. Кузнецов, В. В. Прямая и обратная задача надежности сложных программных комплексов / В. В. Кузнецов, В. А. Смагин // Надежность и контроль качества. – 1997. – № 10. – С. 56 – 62.

19. Липаев, В. В. Надежность программных средств / Липаев В. В. – М. : СИНТЕГ, 1998. – 232 с. – (Информатизация России на пороге XXI века). – ISBN 5-89638-008-9.

20. Майерс, Г. Дж. Надежность программного обеспечения : монография : пер. с англ. / Г. Дж. Майерс ; ред. лит. по математическим наукам. – М. : Мир, 1980. – 360 с.

21. Матвеевский, В. Р. Надежность технических систем : учеб. пособие / В. Р. Матвеевский ; Моск. гос. ин-т электроники и математики. – М., 2002. – 113 с.

22. Можаев, А. С. Общий логико-вероятностный метод анализа надежности структурно-сложных систем : учеб. пособие / А. С. Можаев. – Л. : ВМА, 1988. – 68 с.

23. Можаев, А. С. Современное состояние и некоторые направления развития логико-вероятностных методов анализа систем / А. С. Можаев // Теория и информационная технология моделирования безопасности сложных систем / под ред. И. А. Рябинина. – СПб. : ИПМАШ РАН, 1994. – С. 23 – 53.

24. Можаев, А. С. Учет временной последовательности отказов элементов в логико-вероятностных моделях надежности / А. С. Можаев // Надежность систем энергетики. – Новочеркасск, 1990. – С. 94 – 103.

25. Матвеев, Е. В. Моделирование характеристик информационной безопасности объекта с помощью логико-вероятностного подхода / Е. В. Матвеев, М. А. Смирнова // Координационный совет по информатизации администрации Владимирской области [сайт]. – URL: http://ksi.avo.ru/seminar/22.pdf (дата обращения : 16.06.2011).

26. Мосягин, А. А. Мониторинг потенциально опасных объектов на основе логико-вероятностного моделирования : автореф. дис. … канд. техн. наук / Мосягин А. А. – М., 2009. – 33 с.

27. Нозик, А.А. Оценка надежности и безопасности структурно-сложных технических систем. : автореф. дис. … канд. техн. наук / Нозик А. А. – СПб., – 2005. – 24 с.

28. Панин, О. А. Анализ безопасности интегрированных систем защиты: логико-вероятностный подход / О. А. Панин // Специальная техника [сайт]. – URL: http://st.ess.ru/publications/5-2004/panin.pdf (дата обращения : 16.06.2011).

29. Панин, О. А. Проблемы оценки эффективности функционирования систем физической защиты объектов / О. А. Панин // БДИ – безопасность, достоверность, информация. – 2007. – № 3 (72). – С. 23 – 27.

30. Панин, О. А. Как измерить эффективность? Логико-вероятностное моделирование в задачах оценки систем физической защиты // БДИ – безопасность, достоверность, информация. – 2008. – № 2 (77). – С. 20 – 24.

31. Половко, А. М. Основы теории надежности : практикум / А. М. Половко, С. В. Гуров. – СПб. : БХВ-Петербург, 2006. – 560 с. – ISBN 5-94157-542-4.

32. Полонников, Р. И. Методы оценки показателей надежности программного обеспечения / Р. И. Полонников, А. В. Никандров. – СПб. : Политехника, 1992. – 77 с. – ISBN 5-7325-0185-1.

33. Романюк, С. Г. Оценка надежности программного обеспечения / С. Г. Романюк // Открытые системы [сайт]. – URL: http://www.osp.ru/os/1994/04/178540/ (дата обращения : 16.06.2011).

34. Рыжкин, А. А. Основы теории надежности : учеб. пособие / А. А. Рыжкин, Б. Н. Слюсарь, К. Г. Шучев. – Ростов н/Д : Издат. центр ДГТУ, 2002. – 182 с. – ISBN 5-7890-0209-9.

35. Рябинин, И. А. Надежность и безопасность структурно-сложных систем / И. А. Рябинин. – СПб. : Изд-во СПб. гос. ун-та, 2007. – 276 с. – ISBN 978-5-288-04296-6.

36. Рябинин, И. А. Математическое понятие логико-вероятностного исчисления и особенности его применения в случае немонотонных, повторных и правильных функций алгебры логики / И. А. Рябинин // Труды междунар. науч. шк. «Моделирование и анализ безопасности и риска в сложных системах» МАБР-2009 [сайт]. – URL: http:/logic-cor.narod.ru/Ryabinin.pdf (дата обращения : 16.06.2011).

37. Рябинин, И. А. Логико-вероятностное исчисление как аппарат исследования надежности и безопасности структурно-сложных систем / И. А. Рябинин // Автоматика и телемеханика. – 2003. – № 7. – С. 178 – 186.

38. Рябинин, И. А. Логико-вероятностный анализ проблем надежности, живучести и безопасности : очерки разных лет / И. А. Рябинин ; Военно-мор. акад. им. Н. Г. Кузнецова [и др.]. – Новочеркасск : Изд-во ЮРГТУ, 2009. – 599 с.

39. Смагин, В. А. Расчет вероятностно-временных характеристик пребывания задач в сетевой модели массового обслуживания / В. А. Смагин, В. П. Бубнов, Г. В. Филимонихин // Известия вузов. Приборостроение. – 1989. – № 2. – Т. XXXII.

40. Смагин, В. А. Основы теории надёжности программного обеспечения : учеб. для вузов / В. А. Смагин, А. Н. Дорохов ; БГТУ "ВОЕНМЕХ". – СПб. : [б. и.], 2009. – 303 с. – ISBN 978-5-85546-479-9.

41. Смагин, В. А. Моделирование и обеспечение надёжности программных средств АСУ / В. А. Смагин, В. С. Солдатенко, В. В. Кузнецов. – СПб. : ВИКУ им. А. Ф. Можайского, 1999. – 49 с.

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

43. Смагин, В. А. Стратегия последовательных приближений для повышения надежности функционирования сложных программных комплексов / В. А. Смагин // Надежность и контроль качества. – 2001. – № 7. – С. 35 – 43.

44. Строгонов, А. В. Обзор программных комплексов по расчету надежности сложных технических систем / А. В. Строгонов, В. В. Жаднов, С. Н. Полесский // Компоненты и технологии. – 2007. – № 5. – С. 183 – 190.

45. Фатуев, В. А. Надежность автоматизированных информационных систем : учеб. пособие / В. А. Фатуев, В. И. Высоцкий, В. И. Бушинский ; Тул. гос. ун-т. – Тула : Изд-во ТулГУ, 1998. – 104 с. – ISBN 5-767-90170-8.

46. Фатуев, В. А. Построение оптимальных моделей динамики по экспериментальным данным : учеб. пособие / В. А. Фатуев. – Тула : [б. и.], 1993. – 104 с. – ISBN 5-230-25958-2.

47. Чертков, Р. А. Анализ структурной надежности при проектировании сложных систем передачи информации с применением логико-вероятностных методов : автореф. дис. … канд. техн. наук / Чертков Р. А. – Воронеж, 2009. – 33 с.

48. Шураков, В. В. Надежность программного обеспечения систем обработки данных : учеб. для вузов по специальности "Организация механизир. обработки экон. информации" / В. В. Шураков. – М. : Статистика, 1981. – 216 с.

49. Яворский, В. А. Планирование научного эксперимента и обработка экспериментальных данных : учеб.-метод. пособие / В. А. Яворский. – М. : МФТИ, 2006. – 24 с.

Оглавление

Учебное издание

Комплексная защита объектов информатизации. Книга 22

МОНАХОВ Юрий Михайлович

ФУНКЦИОНАЛЬНАЯ УСТОЙЧИВОСТЬ
ИНФОРМАЦИОННЫХ СИСТЕМ

Часть 1. Надежность программного обеспечения

Учебное пособие

Подписано в печать 25.11.11.
Формат 60x84/16. Усл. печ. л. 3,49. Тираж 70 экз.
Заказ
Издательство
Владимирского государственного университета
имени Александра Григорьевича и Николая Григорьевича Столетовых.
600000, Владимир, ул. Горького, 87.