Лабораторная работа №8

Проектирование надежного программного обеспечения с учетом качества артефактов проекта

Цель работы: приобретение практических навыков, необходимых при разработке программного обеспечения, учитывающего требования надёжности.

КРАТКАЯ ТЕОРИЯ И МЕТОДИЧЕСКИЕ РЕКОМЕНДАЦИИ

Основные принципы проектирования надежного ПО

Разработка программы включает задачи двух типов: проектирование и тестирование. Мы будем использовать термин “проектирование” в смысле: “придание формы в соответствии с планом”. Это определение охватывает различные виды деятельности по созданию программного обеспечения, начиная с определения требований и целей и заканчивая написанием текста программы, но подразумевает, конечно, наличие различных стадий проектирования. Фразы “разработка программного обеспечения”, “конструирование программного обеспечения” и “производство программного обеспечения” обозначают весь цикл его создания. В этой главе рассматриваются некоторые принципы, общие для всех стадий проектирования.

Все принципы и методы обеспечения надежности в соответствии с их целью можно разбить на четыре группы: предупреждение ошибок, обнаружение ошибок, исправление ошибок и обеспечение устойчивости к ошибкам. К первой группе относятся принципы и методы, позволяющие минимизировать или вообще исключить ошибки. Методы второй группы сосредоточивают внимание на функциях самого программного обеспечения, помогающих выявлять ошибки. К третьей группе относятся функции программного обеспечения, предназначенные для исправления ошибок или их последствий.

Устойчивость к ошибкам — это мера способности системы программного обеспечения продолжать функционирование при наличии ошибок.

Предупреждение ошибок К этой группе относятся принципы и методы, цель которых — не допустить появления ошибок в готовой программе. Большинство методов концентрируется на отдельных процессах перевода и направлено на предупреждение ошибок в этих процессах. Их можно разбить на следующие категории:

1. Методы, позволяющие справиться со сложностью, свести ее к минимуму, так как это главная причина ошибок перевода.
2. Методы достижения большей точности при переводе.
3. Методы улучшения обмена информацией.

4. Методы немедленного обнаружения и устранения ошибок. Эти методы направлены на обнаружение ошибок на каждом шаге перевода, а не во время тестирования программы после ее написания.

Очевидно, что предупреждение ошибок — оптимальный путь к достижению надежности программного обеспечения. Лучший способ обеспечить надежность — не допустить возникновения ошибок. Гарантировать отсутствие ошибок, однако, невозможно. Другие три группы методов опираются на предположение, что ошибки все-таки будут.

Обнаружение ошибок

Если предположить, что в программном обеспечении какие-то ошибки все же будут, то лучшая (после предупреждения ошибок) стратегия — включить средства обнаружения ошибок в само программное обеспечение. Большинство методов направлено по возможности на незамедлительное обнаружение сбоев. Немедленное обнаружение имеет два преимущества: можно минимизировать как влияние ошибки, так и последующие затруднения для человека, которому придется извлекать информацию об этой ошибке, находить ее место и исправлять.

Исправление ошибок

Следующий шаг — методы исправления ошибок; после того как ошибка обнаружена, либо она сама, либо ее последствия должны быть исправлены программным обеспечением. Исправление ошибок самой системой — плодотворный метод проектирования надежных систем аппаратного обеспечения.

Некоторые устройства способны обнаружить неисправные компоненты и перейти к использованию идентичных запасных. Аналогичные методы неприменимы к программному обеспечению вследствие глубоких внутренних различий между сбоями аппаратуры и ошибками в программах. Если некоторый программный модуль содержит ошибку, идентичные “запасные” модули также будут содержать ту же ошибку.

Другой подход к исправлению связан с попытками восстановить разрушения, вызванные ошибками, например искажения записей в базе данных или управляющих таблицах системы. Польза от методов борьбы с искажениями ограниченна, поскольку предполагается, что разработчик заранее предугадает несколько возможных типов искажений и предусмотрит программно
реализуемые функции для их устранения. Это похоже на парадокс, поскольку, если знать заранее, какие ошибки возникнут, можно было бы принять дополнительные меры по их предупреждению. Если методы ликвидации последствий сбоев не могут быть обобщены для работы со многими типами искажений, лучше всего направлять силы и средства на предупреждение ошибок. Вместо того чтобы, разрабатывая операционную систему, оснащать ее средствами обнаружения и восстановления цепочки искаженных таблиц

или управляющих блоков, следовало бы лучше спроектировать систему так, чтобы только один модуль имел доступ к этой цепочке, а затем настойчиво пытаться убедиться в правильности этого модуля.

Устойчивость к ошибкам

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

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

Второй подход к динамической избыточности — выполнять эти запасные копии только тогда, когда результаты, полученные с помощью основной копии, признаны неправильными. Если это происходит, система автоматически вызывает запасную копию. Если и ее результаты неправильны, вызывается другая запасная копия и т. д. Этот подход страдает большинством перечисленных ранее недостатков. Кроме того, вполне вероятно, что если ресурсы при работе над проектом фиксированы, то при реализации “запасных” версий проектированию и тестированию будет уделено меньше внимания, чем можно было бы уделить, если бы реализовывалась лишь одна копия и динамическая избыточность не использовалась.

2. Вторая подгруппа методов обеспечения устойчивости к ошибкам называется методами отступления или сокращенного обслуживания. Эти методы приемлемы обычно лишь тогда, когда для системы программного

обеспечения существенно важно благопристойно закончить работу. Например, если ошибка оказывается в системе, управляющей технологическими процессами, и в результате эта система выходит из строя, то может быть загружен и выполнен особый фрагмент программы, призванный подстраховать систему и обеспечить безаварийное завершение всех управляемых системой процессов. Аналогичные средства часто необходимы в операционных системах. Если операционная система обнаруживает, что она вот-вот выйдет из строя, она может загрузить аварийный фрагмент, ответственный за оповещение пользователей у терминалов о предстоящем сбое и за сохранение всех критических для системы данных.

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

Из этих трех подгрупп методов обеспечения устойчивости к ошибкам только третья, изоляция ошибок, применима для большинства систем программного обеспечения.

Важное обстоятельство, касающееся всех четырех подходов, состоит в том, что обнаружение, исправление ошибок и устойчивость к ошибкам в некотором отношении противоположны методам предупреждения ошибок. В частности, обнаружение, исправление и устойчивость требуют дополнительных функций от самого программного обеспечения. Тем самым не только увеличивается сложность готовой системы, но и появляется возможность внести новые ошибки при реализации этих функций. Как правило, все рассматриваемые методы предупреждения и многие методы обнаружения ошибок применимы к любому программному проекту. Методы исправления ошибок и обеспечения устойчивости применяются не очень широко. Это, однако, зависит от области приложения. Если рассматривается, скажем, система реального времени, то ясно, что она должна сохранить работоспособность и при наличии ошибок, а тогда могут оказаться желательными и методы исправления и обеспечения устойчивости. К

системам такого типа относятся телефонные переключательные системы, системы управления технологическими процессами, аэрокосмические и авиационные диспетчерские системы и операционные системы широкого назначения.
Принципы — это независимые от области приложения стратегии обеспечения надежности программных систем. Рассматриваются принципы проектирования системы, программы, модуля, направленные на предупреждение
ошибок. Для остальных трех категорий принципы неизвестны. Методы — это более мелкие, обычно зависящие от области приложения тактические средства.

ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

Задание 1

При выполнении лабораторной работы, необходимо спроектировать программное обеспечение, обладающее следующими свойствами: обнаружения ошибок, исправления ошибок и обеспечения устойчивости к ошибкам. Для обнаружения ошибок в каждой подпрограмме необходимо предусмотреть средства обнаружения ошибок. Для этих целей все входные данные необходимо проверять на наличие ошибок. Далее необходимо проверять на информацию на наличие сбоев. После того как ошибка обнаружена, либо она сама, либо ее последствия должны быть исправлены программным обеспечением.

Программа должна соответствовать стилю программирования Microsoft в плане задания имен переменных, функций, классов и др.

Программа должна решать задачу на определение наиболее точного равенства.

[Определить, какое равенство точнее.

50/19=2,630 $\sqrt{27}$ =5,190

]

Числа для вычисления должны вводиться с клавиатуры.

Задание 2

Написать программу с помощью урока, размещенного по следующей ссылке:

https://www.bestprog.net/ru/2016/04/29/011-c-an-example-of-creating-of-two-dimensional-matrix-on-the-form-the-analogue-of-tstringgrid-component-in-delphi_ru/

Задание 3.

Необходимо познакомиться с разработанной Microsoft программой для самостоятельной оценки рисков, связанных с безопасностью – Microsoft Security Assessment Tool (MSAT). Она бесплатно доступна на сайте Microsoft по ссылке:

http://www.microsoft.com/ru-ru/download/details.aspx?id=12273

Необходимо подробно описать реально существующее или вымышленное малое предприятие: сферу деятельности, состав и структуру информационной системы, особенности организации процесса защиты информации, применяемые методы и средства, кто из сотрудников или подрядчиков имеет доступ к информационной системе предприятия и т.д.

С помощью программы MSAT провести оценку рисков для предприятия.

Порядок выполнения работы

1. Произвести инсталляцию программы (при необходимости).

2. Создать новый профиль (Меню «Файл» -> «Настройка профилей»).

3. Задать параметры компании.

4. Создать новую оценку.

5. Заполнить параметры оценки (вопросы по инфраструктуре, приложениям, операциям, персоналу). При ответе на вопросы обращайте внимание на дополнительную информацию (рис. 3, а)) и информацию по передовым отраслевым методикам в данном вопросе (рис. 3, б))

а) б)

Рис. 3. Обозначения в программе MSAT:
а) дополнительная информация;
б) информация по передовым отраслевым методикам в данном вопросе

6. Сформировать в программе MSAT отчет по итогам анализа компании.

7. Сформулировать направления (путей) улучшения качества информационной безопасности на предприятии.

Требования к содержанию отчета

Отчет должен содержать:

1. Подробное описание компании/предприятия, которое рассматривается в данной системе (часть данных пойдет в анализ в программе MSAT);

2. Перечисление факторов, которые по мнению разработчиков программы, являются ключевыми в оценке информационной безопасности предприятия (информационная структура предприятия, система защиты и т.д., подробно).

3. Сформированный отчет по итогам анализа исходной компании;

4. Сформулированные направления (путей) улучшения качества ИБ на предприятии.