Лекция 5. Дестабилизирующие факторы и методы обеспечения надежности программных средств

1. Модель факторов, определяющих надежность ПС

При любом виде деятельности людям свойственно непредумышленно ошибаться, результаты чего проявляются в процессе создания или применения изделий или систем. В общем случае под ошибкой подразумевается дефект, погрешность или неумышленное искажение объекта или процесса. При этом предполагается, что известно правильное, эталонное состояние объекта, по отношению к которому может быть определено наличие отклонения — дефекта или ошибки. Для систематической, координированной борьбы с ними необходимы исследования факторов, влияющих на надежность ПС со стороны случайных, существующих и потенциально возможных дефектов в конкретных программах. Это позволит целенаправленно разрабатывать комплексы методов и средств обеспечения надежности сложных ПС различного назначения при реально достижимом снижении уровня дефектов проектирования.

При строго фиксированных исходных данных программы исполняются по определенным маршрутам и выдают совершенно определенные результаты. Многочисленные варианты исполнения программ при разнообразных исходных данных представляются для внешнего наблюдателя как случайные. В связи с этим дефекты функционирования программных средств, не имеющие злоумышленных источников, проявляются внешне как случайные, имеют разную природу и последствия. В частности, они могут приводить к последствиям, соответствующим нарушениям работоспособности, и к отказам при использовании ПС.

Последующий анализ надежности ПС базируется на модели взаимодействия основных компонентов.

Объекты уязвимости, влияющие на надежность ПС

Объектами уязвимости, влияющими на надежность ПС, являются:

- динамический вычислительный процесс обработки данных, автоматизированной подготовки решений и выработки управляющих воздействий;

- информация, накопленная в базах данных, отражающая объекты внешней среды, и процессы ее обработки;
- объектный код программ, исполняемых вычислительными средствами в процессе функционирования ПС;
- информация, выдаваемая потребителям и на исполнительные механизмы, являющаяся результатом обработки исходных данных и информации, накопленной в базе данных.

На эти объекты воздействуют различные дестабилизирующие факторы, которые можно разделить на внутренние, присущие самим объектам уязвимости, и внешние, обусловленные средой, в которой эти объекты функционируют.

Дестабилизирующие факторы, влияющие на надежность ПС

Внутренними источниками угроз надежности функционирования сложных ПС можно считать следующие дефекты программ:

- системные ошибки при постановке целей и задач создания ПС, при формулировке требований к функциям и характеристикам решения задач, определении условий и параметров внешней среды, в которой предстоит применять ПС;
- алгоритмические ошибки разработки при непосредственной спецификации функций программных средств, при определе-нии структуры и взаимодействия компонентов комплексов про-грамм, а также при использовании информации баз данных;
- ошибки программирования в текстах программ и описаниях данных, а также в исходной и результирующей документации на компоненты и ПС в целом;
- недостаточную эффективность используемых методов и средств оперативной защиты программ и данных от сбоев и отказов и обеспечения надежности функционирования ПС в условиях случайных негативных воздействий.

Внешними дестабилизирующими факторами, отражающимися на надежности функционирования перечисленных объектов уязвимости в ПС, являются:

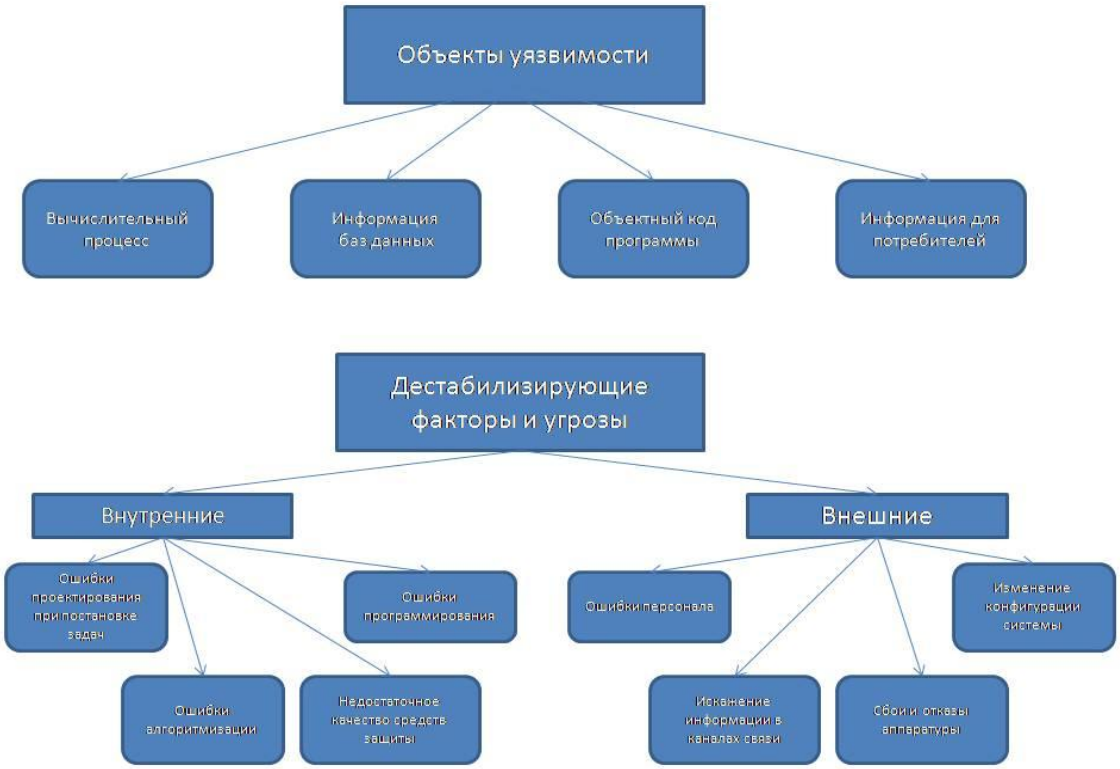
- ошибки оперативного и обслуживающего персонала в процессе эксплуатации ПС;
- искажения в каналах телекоммуникации информации, поступающей от внешних источников и передаваемой потребителям, а также недопустимые для конкретной информационной систе-мы характеристики потоков внешней информации;

- сбои и отказы в аппаратуре вычислительных средств;
- изменения состава и конфигурации комплекса взаимодействующей аппаратуры информационной системы за пределы, проверенные при испытаниях или сертификации и отраженные в эксплуатационной документации.

Полное устранение перечисленных негативных воздействий и дефектов, отражающихся на надежности функционирования сложных ПС, принципиально невозможно. Проблема состоит в выявлении факторов, от которых они зависят, создании методов и средств уменьшения их влияния на надежность ПС, а также в эф-фективном распределении ресурсов на защиту для обеспечения необходимой надежности комплекса программ, равноправного при их реальных воздействиях.

Схема модели анализа надежности ПС

Все сказанное выше илюстрируется следующей схемой модели анализа надежности ПС.



2. Методы обеспечения надежности ПС

В современных автоматизированных технологиях создания и развития сложных ПС с позиции обеспечения их необходимой и заданной надежности можно выделить методы и средства, позволяющие:

- создавать программные модули и функциональные компоненты высокого, гарантированного качества;
- предотвращать дефекты проектирования за счет эффективных технологий и средств автоматизации обеспечения всего жиз-ненного цикла комплексов программ и баз данных;
- обнаруживать и устранять различные дефекты и ошибки проектирования, разработки и сопровождения программ путем систематического тестирования на всех этапах жизненного цикла ПС;
- удостоверять достигнутое качество и надежность функционирования ПС в процессе их испытаний и сертификации перед передачей в регулярную эксплуатацию;
- оперативно выявлять последствия дефектов программ и данных и восстанавливать нормальное, надежное функционирование комплексов программ.

Комплексное, скоординированное применение этих методов и средств в процессе создания, развития и применения ПС позволяет исключать некоторые виды угроз или значительно ослаблять их влияние. Тем самым уровень достигаемой надежности ПС становится предсказуемым и управляемым, непосредственно зависящим от ресурсов, выделяемых на его достижение, а главное от качества и эффективности технологии, используемой на всех этапах жизненного цикла ПС.

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 (четвертая группа) — 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, не откладывая до тестирования программы после ее написания.

Должно быть очевидно, что предупреждение ошибок — оптимальный путь к достижению надежности программного обеспечения.

Лучший способ обеспечить надежность — прежде всего не допустить возникновения ошибок. Гарантировать отсутствие ошибок, однако, невозможно никогда. Другие три группы методов опираются на предположение, что ошибки все-таки будут.

Обнаружение ошибок

Если предполагать, 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 влияние ошибки и последующие затруднения для человека, которому придется извлекать информацию о ней, находить ее и исправлять.

Меры по обнаружению ошибок можно разбить на две подгруппы: пассивные попытки обнаружить симптомы ошибки в процессе «обычной» работы программного

обеспечения и активные попытки программной системы периодически обследовать свое состояние в поисках признаков ошибок.

Пассивное обнаружение. Меры по обнаружению ошибок могут быть приняты на нескольких структурных уровнях программной системы. Здесь мы будем рассматривать уровень подсистем, или компонентов, т.е. нас будут интересовать меры по обнаружению симптомов ошибок, предпринимаемые при переходе от одного компонента к другому, а также внутри компонента. Все это, конечно, применимо также к отдельным модулям внутри компонента.

Разрабатывая эти меры, мы будем опираться на следующее.

1. Взаимное недоверие. Каждый из компонентов должен пред-полагать, что все другие содержат ошибки. Когда он получает какие-нибудь данные от другого компонента или из источника вне системы, он должен предполагать, что данные могут быть неправильными, и пытаться найти в них ошибки.

2. Немедленное обнаружение. Ошибки необходимо обнаружить как можно раньше. Это не только ограничивает наносимый ими ущерб, но и значительно упрощает задачу отладки.

3. Избыточность. Все средства обнаружения ошибок основаны на некоторой форме избыточности (явной или неявной).

Когда разрабатываются меры по обнаружению ошибок, важно принять согласованную стратегию для всей системы. Действия, предпринимаемые после обнаружения ошибки в программном обеспечении, должны быть единообразными для всех компонентов системы. Это ставит вопрос о том, какие именно действия следует предпринять, когда ошибка обнаружена. Наилучшее решение — немедленно завершить выполнение программы или (в случае операционной системы) перевести центральный процессор в состояние ожидания. С точки зрения предоставления человеку, отлаживающему программу, например системному программисту, самых благоприятных условий для диагностики ошибок немедленное завершение представляется наилучшей стратегией. Конечно, во многих системах подобная стратегия бывает нецелесообразной (например, может оказаться, что приостанавливать работу системы нельзя). В таком случае используется метод регистрации ошибок. Описание симптомов ошибки и «моментальный снимок» состояния системы сохраняются во внешнем файле, после чего

система может продолжать работу. Этот файл позднее будет изучен обслуживающим персоналом.

Всегда, когда это возможно, лучше приостановить выполнение программы, чем регистрировать ошибки (либо обеспечить как дополнительную возможность работу системы в любом из этих режимов). Различие между этими методами проиллюстриру-ем на способах выявления причин возникающего иногда скреже-та вашего автомобиля. Если автомеханик находится на заднем сиденье, то он может обследовать состояние машины в тот мо-мент, когда скрежет возникает. Если вы выбираете метод регистрации ошибок, задача диагностики станет сложнее.

Активное обнаружение ошибок. Не все ошибки можно выявить пассивными методами, поскольку эти методы обнаруживают ошибку лишь тогда, когда ее симптомы подвергаются соответствующей проверке. Можно делать и дополнительные проверки, если спроектировать специальные программные средства для активного поиска признаков ошибок в системе. Такие средства называются средствами активного обнаружения ошибок.

Активные средства обнаружения ошибок обычно объединя-ются в диагностический монитор: параллельный процесс, кото-рый периодически анализирует состояние системы с целью обна-ружить ошибку. Большие программные системы, управляющие ресурсами, часто содержат ошибки, приводящие к потере ресурсов на длительное время. Например, управление памятью операционной системы сдает блоки памяти «в аренду» программам пользователей и другим частям операционной системы. Ошибка в этих самых «других частях» системы может иногда вести к неправильной работе блока управления памятью, занимающегося возвратом сданной ранее в аренду памяти, что вызывает медленное вырождение системы.

Диагностический монитор можно реализовать как периодически выполняемую задачу (например, она планируется на каждый час) либо как задачу с низким приоритетом, которая планируется для выполнения в то время, когда система переходит в состояние ожидания. Как и прежде, выполняемые монитором конкретные проверки зависят от специфики системы, но некоторые идеи будут понятны из примеров. Монитор может обследовать основную память, чтобы обнаружить блоки памяти, не выделенные ни одной из выполняемых задач и не включенные в системный список свободной памяти. Он может проверять также необычные ситуации: например, процесс не планировался для

выполнения в течение некоторого разумного интервала времени. Монитор может осуществлять поиск «затерявшихся» внутри системы сообщений или операций ввода-вывода, которые необычно долгое время остаются незавершенными, участков памяти на диске, которые не помечены как выделенные и не включены в список свободной памяти, а также различного рода странностей в файлах данных.

Иногда желательно, чтобы в чрезвычайных обстоятельствах монитор выполнял диагностические тесты системы. Он может вызывать определенные системные функции, сравнивая их результат с заранее определенным и проверяя, насколько разумно время вы-полнения. Монитор может также периодически предъявлять системе «пустые» или «легкие» задания, чтобы убедиться, что система функционирует хотя бы самым примитивным образом.