УВАЖАЕМЫЕ СТУДЕНТЫ! Изучите приведенную лекцию, законспектируйте основные тезисы и основные правила оптимизации.

Ответы на вопросы, фотоотчет, предоставить преподавателю на e-mail: r.bigangel@gmail.com до 20.02.2023.

При возникновении вопросов по приведенному материалу обращаться по следующему номеру телефона: (072)111-37-59, (Viber, WhatsApp), vk.com: https://vk.com/daykini

ВНИМАНИЕ!!! При отправке работы, не забывайте указывать ФИО студента, наименование дисциплины, дата проведения занятия (по расписанию).

Лекция 31 (продолжение)

Тема: «Основные ошибки при разработке программного обеспечения»

План лекции:

1.1 Понятие ошибки программного обеспечения

1.2 Классификация ошибок

1.3 Методы поиска ошибок

1.1 Понятие ошибки программного обеспечения

Ошибка – это свойство программы, приводящее к ухудшению её качества по любому из трёх критериев качества: работоспособности, сопровождаемости, гибкости.

Прошу задержать внимание на определении, так как мне приходилось часто наблюдать проблемы, связанные с непониманием того, что считать ошибкой.

Например, встречал случаи, когда на официальном уровне ошибкой считалось только то, о чем сообщили пользователи. Изменения можно было вносить только по заявкам конечных пользователей, и найденный в ходе чтения кода баг, не мог быть легально исправлен, кроме как партизанскими методами, путем сочинения писем пользователям, с просьбой зарегистрировать их как обращения от своего имени.

Крайне важно понимать, что ошибка — это имманентное свойство текущей версии программы, а не некоторое событие, явление или баг-репорт и уж тем более не исключение (exception), которое во многих случаях ошибкой не является.

1.2 Классификация ошибок

В краткой классификации выделяются следующие ошибки.

- Ошибки пользовательского интерфейса.
- Ошибки вычислений.
- Ошибки управления потоком.
- Ошибки передачи или интерпретации данных.
- Перегрузки.
- Контроль версий.
- Ошибка выявлена и забыта.
- Ошибки тестирования.

1. Ошибки пользовательского интерфейса.

Многие из них субъективны, т.к. часто они являются скорее неудобствами, чем «чистыми» логическими ошибками. Однако они могут провоцировать ошибки пользователя программы или же замедлять время его работы до неприемлемой величины. В результате чего мы будем иметь ошибки информационной системы (ИС) в целом. Основным источником таких ошибок является сложный компромисс между функциональностью программы и простотой обучения и работы пользователя с этой программой. Проблему надо начинать решать при проектировании системы на уровне ее декомпозиции на отдельные модули, исходя из того, что вряд ли удастся спроектировать простой и удобный пользовательский интерфейс для модуля, перегруженного различными функциями. Кроме того, необходимо учитывать рекомендации по проектированию пользовательских интерфейсов. На этапе тестирования ПО полезно предусмотреть встроенные средства тестирования, которые бы запоминали последовательности действий пользователя, время

совершения отдельных операций, расстояния перемещения курсора мыши. Кроме этого, возможно применение гораздо более сложных средств психофизического тестирования на этапе тестирования интерфейса пользователя, которые позволят оценить скорость реакции пользователя, частоту этих реакций, утомляемость и т.п. Необходимо отметить, что такие ошибки очень критичны с точки зрения коммерческого успеха разрабатываемого ПО, т.к. они будут в первую очередь оцениваться потенциальным заказчиком.

2. Ошибки вычислений.

Выделяют следующие причины возникновения таких ошибок:

- неверная логика (может быть следствием, как ошибок проектирования, так и кодирования);

- неправильно выполняются арифметические операции (как правило - это ошибки кодирования);

- неточные вычисления (могут быть следствием, как ошибок проектирования, так и кодирования). Очень сложная тема, надо выработать свое отношение к ней с точки зрения разработки безопасного ПО.

Выделяются подпункты: устаревшие константы; ошибки вычислений; неверно расставленные скобки; неправильный порядок операторов; неверно работает базовая функция; переполнение и потеря значащих разрядов; ошибки отсечения и округления; путаница с представлением данных; неправильное преобразование данных из одного формата в другой; неверная формула; неправильное приближение.

3. Ошибки управления потоком.

В этот раздел относится все то, что связано с последовательностью и обстоятельствами выполнения операторов программы.

Выделяются подпункты:

- очевидно неверное поведение программы;

- переход по GOTO;

- логика, основанная на определении вызывающей подпрограммы;

- использование таблиц переходов;

- выполнение данных (вместо команд). Ситуация возможна из-за ошибок работы с указателями, отсутствия проверок границ массивов, ошибок перехода, вызванных, например, ошибкой в таблице адресов перехода, ошибок сегментирования памяти.

4. Ошибки обработки или интерпретации данных.

Выделяются подпункты:

- проблемы при передаче данных между подпрограммами (сюда включены несколько видов ошибок: параметры указаны не в том порядке или пропущены, несоответствие типов данных, псевдонимы и различная интерпретация содержимого одной и той же области памяти, неправильная интерпретация данных, неадекватная информация об ошибке, перед аварийным выходом из подпрограммы не восстановлено правильное состояние данных, устаревшие копии данных, связанные переменные не синхронизированы, локальная установка глобальных данных (имеется в виду путаница локальных и глобальных переменных), глобальное использование локальных переменных, неверная маска битового поля, неверное значение из таблицы);

- границы расположения данных (сюда включены несколько видов ошибок: не обозначен конец нуль-терминированной строки, неожиданный конец строки, запись/чтение за границами структуры данных или ее элемента, чтение за пределами буфера сообщения, чтение за пределами буфера сообщения, дополнение переменных до полного слова, переполнение и выход за нижнюю границу стека данных, затирание кода или данных другого процесса);

- проблемы с обменом сообщений (сюда включены несколько видов ошибок: отправка сообщения не тому процессу или не в тот порт, ошибка распознавания полученного сообщения, недостающие или несинхронизированные сообщения, сообщение передано только N процессам из N+1, порча данных, хранящихся на внешнем устройстве, потеря

изменений, не сохранены введенные данные, объем данных слишком велик для процесса-получателя, неудачная попытка отмены записи данных).

5. Повышенные нагрузки.

При повышенных нагрузках или нехватке ресурсов могут возникнуть дополнительные ошибки. Выделяются подпункты: требуемый ресурс недоступен; не освобожден ресурс; нет сигнала об освобождении устройства; старый файл не удален с накопителя; системе не возвращена неиспользуемая память; лишние затраты компьютерного времени; нет свободного блока памяти достаточного размера; недостаточный размер буфера ввода или очереди; не очищен элемент очереди, буфера или стека; потерянные сообщения; снижение производительности; повышение вероятности ситуационных гонок; при повышенной нагрузке объем необязательных данных не сокращается; не распознается сокращенный вывод другого процесса при повышенной загрузке; не приостанавливаются задания с низким приоритетом.

В этом разделе хотелось бы обратить внимание на следующее:

1) Часть ошибок из этого раздела могут проявляться и при не очень высоких нагрузках, но, возможно, они будут проявляться реже и через более длительные интервалы времени;

2) Многие ошибки из 2-х предыдущих разделов уже в своей формулировке носят вероятностный характер, поэтому следует предположить возможность использования вероятностных моделей и методов для их выявления.

6. Контроль версий и идентификаторов.

Выделяются подпункты: таинственным образом появляются старые ошибки; обновление не всех копий данных или программных файлов; отсутствие заголовка; отсутствие номера версии; неверный номер версии в заголовке экрана; отсутствующая или неверная информация об авторских правах; программа, скомпилированная из архивной копии, не соответствует проданному варианту; готовые диски содержат неверный код или данные.

7. Ошибки тестирования.

Являются ошибками сотрудников группы тестирования, а не программы. Выделяются подпункты:

- пропущенные ошибки в программе;

- не замечена проблема (отмечаются следующие причины этого: тестировщик не знает, каким должен быть правильный результат, ошибка затерялась в большом объеме выходных данных, тестировщик не ожидал такого результата теста, тестировщик устал и невнимателен, ему скучно, механизм выполнения теста настолько сложен, что тестировщик уделяет ему больше внимания, чем результатам);

- пропуск ошибок на экране;

- не документирована проблема (отмечаются следующие причины этого: тестировщик неаккуратно ведет записи, тестировщик не уверен в том, что данные действия программы являются ошибочными, ошибка показалась слишком незначительной, тестировщик считает, что ошибку не будет исправлена, тестировщика просили не документировать больше подобные ошибки).

8. Ошибка выявлена и забыта.

Описываются ошибки использования результатов тестирования. По-моему, раздел следует объединить с предыдущим. Выделяются подпункты: не составлен итоговый отчет; серьезная проблема не документирована повторно; не проверено исправление; перед выпуском продукта не проанализирован список нерешенных проблем.

Необходимо заметить, что изложенные в 2-х последних разделах ошибки тестирования требуют для устранения средств автоматизации тестирования и составления отчетов. В идеальном случае, эти средства должны быть проинтегрированы со средствами и технологиями проектирования ПО. Они должны стать важными инструментальными средствами создания высококачественного ПО. При разработке средств автоматизированного тестирования следует избегать ошибок, которые

присущи любому ПО, поэтому нужно потребовать, чтобы такие средства обладали более высокими характеристиками надежности, чем проверяемое с их помощью ПО.

Основные пути борьбы с ошибками

Учитывая рассмотренные особенности действий человека при переводе, можно указать следующие пути борьбы с ошибками:

– сужение пространства перебора (упрощение создаваемых систем),

– обеспечение требуемого уровня подготовки разработчика (это функции менеджеров коллектива разработчиков),

– обеспечение однозначности интерпретации представления информации,

– контроль правильности перевода (включая и контроль однозначности интерпретации).

1.3 Методы поиска ошибок

Главное, о чем нужно помнить, что ошибка должна быть обнаружена как можно ближе к моменту её появления как свойства программы, а не как события в работе программы. Ниже приведу шесть методов поиска, упорядоченных по примерной хронологии и по убыванию желательности обнаружения на данном этапе.

Поиск в процессе разработки

На этом этапе может быть отсеяна львиная доля ошибок. Здесь применяется ревью кода и первичное тестирование. Способ этот ценен тем, что ошибка, найденная разработчиком в процессе разработки, гораздо легче и быстрее исправляется. Ведь программист уже погружен в контекст ошибки и накладные расходы минимальны.

Инспекция кода

Независимо от того занимаются ли сами разработчики поиском ошибок и тестированием, необходимость инспекции кода вполне очевидна.

Во-первых, инспектор может быть не погружен в контекст, но он как правило видит общую картину и глаз его не «замылен».

Во-вторых, сам факт наличия организованного процесса инспекции кода приводит к повышению качества выпускаемого кода.

В-третьих, инспектор, будучи более квалифицированным специалистом, сможет увидеть проблемы неочевидные для рядовых разработчиков.

Момент проведения инспекции вопрос обсуждаемый. Обычно инспекцию проводят перед отправкой на тестирование функциональности, но после автоматической проверки. Такая схема позволяет снизить нагрузку на инспектора, и в то же время уже до этапа тестирования сократить число ошибок.

Профилирование

Под профилированием здесь обобщенно понимается комплекс мер по снятию и анализу различных метрик работы программы. Перечислю вкратце наиболее доступные методики:

- **Замер производительности.** Можно выполнять и встроенными средствами для чернового результата, но лучше использовать более точные замеры с логированием факта и времени событий.

- **Поиск узких мест в алгоритмах.** Здесь вполне подойдет встроенный в платформу 1с замер производительности. Но смотреть здесь будем не время, а количество вызовов. Таким образом легко и дешево выявляются нагруженные участки кода, на которые необходимо обратить внимание. Как правило, при вводе в эксплуатацию этого метода, число найденных ошибок резко возрастает и по мере их исправление производительность программы начинает расти.

- **Поиск проблем в конвертации запросов из языка 1с в язык СУБД.** Как многие из вас знают, запросы, написанные в 1с, интерпретируются в язык используемой СУБД. В процессе интерпретации неоптимальных запросов, могут рождаться «монстры». В MS SQL Server есть специальный инструмент, который так и называется «Profiler». С другими СУБД знаком слабо, но полагаю, аналогичные возможности есть и там.

Тестирование

Тестирование может быть, как простым, так и комплексным процессом. Элементы тестирования могут применяться уже на этапе разработки в виде синтаксического и семантического контроля кода. При этом синтаксический анализ выполняется автоматически, средствами платформы, а семантический требует либо отдельных инструментов, либо применения паттернов для поиска ошибок (см. раздел 1.2.4 текущей статьи). Но именно на семантическом уровне и расположены главные гнёзда ошибок.

Разумеется, нельзя забывать и о функциональном тестировании, когда программа пропускается через различные рабочие сценарии вручную или автоматически. Такой способ тестирования несмотря на очевидную простоту, находит, как ни странно, самые очевидные ошибки.

Анализ логов работы и разбор исключений

Данный способ применяется уже не только к тестовым версиям, но и к продуктовым релизам. Общая схема такова:

Логирование работы наиболее важных участков кода. Не встречал необходимости логировать всё. А вот противопоказания к логированию есть - чем больше делается записей, тем меньше шансов, что их будут читать. Поэтому логируем только то, что необходимо.

Анализ ошибок (уровень регистрации) в журнале регистрации. Данный подход неплох сам по себе, при условии уверенного парсинга записей журнала. Рабочая база содержит очень много записей об ошибках, но ошибками являются далеко не все. Второй путь - перехват исключений и регистрация их в нужном формате. Первый подход безальтернативно для типовых, второй - для самописных конфигураций или новых подсистем в типовых.

Жалобы и пожелания пользователей

Если ошибка пробралась незамеченной через все выставленные заслоны (или свободно прошла через неохраняемую зону), то конечные

пользователи без сомнения выявят наиболее заметные и неприятные из них. У способа есть ряд недостатков:

- Ошибка, найденная конечными пользователями, прямо означает, что IT со своей работой справляется плохо. Пусть это вызвано объективными причинами, и ребята не виноваты, но вывод простой и очевидный.

- Далеко не все ошибки регистрируются. Лишь самые надоедливые и мешающие работе дойдут до программистов. Упрощение отправки багрепорта или его автоматизация, несколько сглаживает проблему, но окончательно не устраняет.

- Накладные затраты на работу с ошибками, происходящими у конечных пользователей выше чем на предшествующих этапах.

Наблюдение за работой пользователей

Помимо автоматических средств контроля, существует довольно эффективный метод поиска ошибок. Суть его заключается в прямом физическом наблюдении за работой пользователя. Подчеркиваю, что речь идет о физическом наблюдении. Средства удаленного наблюдения за экраном пользователя, без наблюдения самого пользователя даёт неполную и часто искаженную картину.

В личной практике, подобный метод наиболее эффективно применялся при поиске ошибок в пользовательском интерфейсе. Использование данной методики для нагруженных операциями интерфейсов, позволяло улучшить фактическое быстродействие программы, за счет повышения эргономичности форм. Результат в цифрах может различаться в зависимости от стартовых условий, но обычно наблюдался эффект роста реального быстродействия программы 1,5-3 раза.

Контрольные вопросы:

1 Понятие ошибки программного обеспечения

2 Классификация ошибок

3 Методы поиска ошибок