

Основные понятия в тестировании

Что представляет собой тестирование?

Как определить качество ПО.

Категории программных ошибок.

Терминология.

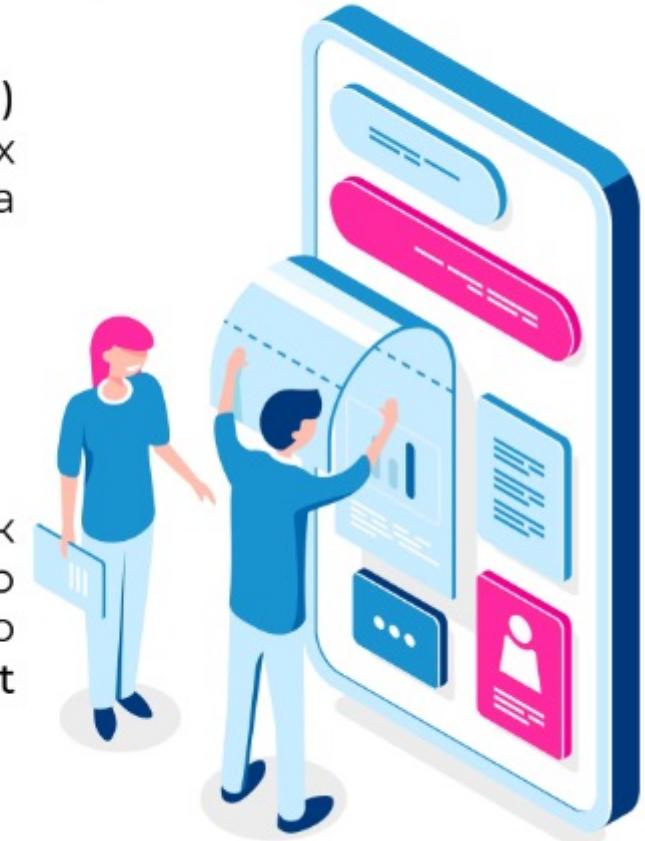


Что представляет собой тестирование

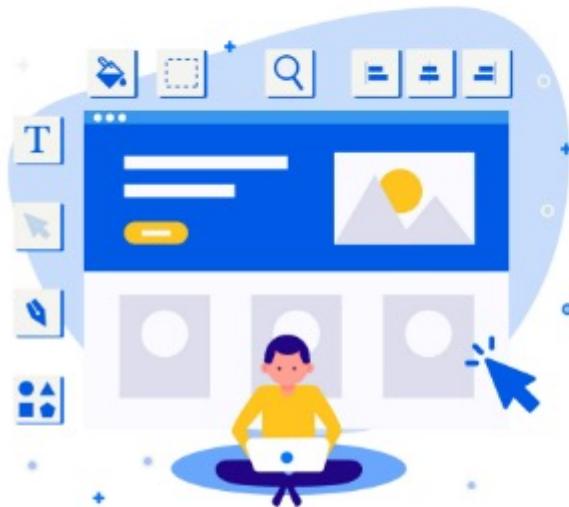
Тестирование программного обеспечения (Software Testing)

— это проверка соответствия реальных и ожидаемых результатов поведения программы, осуществляемая на конечном наборе тестов, выбранном определенным образом.
[IEEE Guide to Software Engineering Body of Knowledge,
SWEBOK, 2004]

Говоря более широко, тестирование — это одна из техник контроля качества, которая включает активности по планированию работ (Test Management), проектированию тестов (Test Design), выполнению тестирования (Test Execution) и анализу полученных результатов (Test Analysis).



Что представляет собой тестирование



- Тестировщик — специалист, который занимается тестированием. В его обязанности входит поиск возможных ошибок и сбоев в функционале тестирования объекта (например, приложения).
- Тестировщик моделирует ситуации, вероятные при использовании тестируемого объекта, чтобы потом разработчики могли устранить обнаруженные неполадки.

Инженер-тестировщик — это человек, решающий проблемы технического характера, связанные с разработкой ПО. Он и пользователь, и эксперт в одном лице: он должен быть способен как воспроизвести действия пользователя, так и проанализировать с точки зрения инженера поведение программы, входные параметры и результаты на выходе.

Тестировщики — это исследователи мира разработки.

Основные «эпохи тестирования»

В 50–60-х годах прошлого века процесс тестирования был предельно формализован, отделён от процесса непосредственной разработки ПО и «математизирован».

Фактически тестирование представляло собой скорее отладку программ (*debugging*).

Существовала концепция «исчерпывающего тестирования (*exhaustive testing*)» — проверки всех возможных путей выполнения кода со всеми возможными входными данными.



Основные «эпохи тестирования»

Очень скоро было выяснено, что исчерпывающее тестирование невозможно, т.к. количество возможных путей и входных данных очень велико, а также при таком подходе сложно найти проблемы в документации.



Основные «эпохи тестирования»

В 70-х годах фактически родились две фундаментальные идеи тестирования: тестирование сначала рассматривалось как процесс доказательства работоспособности программы в некоторых заданных условиях (*positive testing*), а затем — строго наоборот: как процесс доказательства неработоспособности программы в некоторых заданных условиях (*negative testing*).

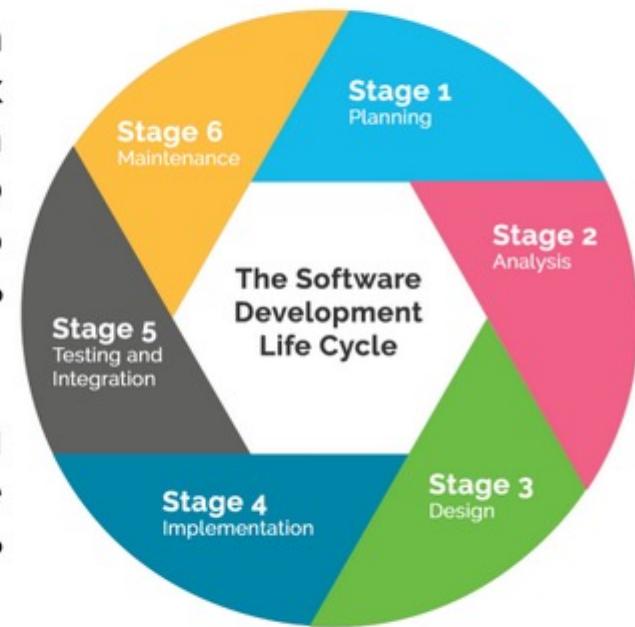
Это внутреннее противоречие не только не исчезло со временем, но и в наши дни многими авторами совершенно справедливо отмечается как две взаимодополняющие цели тестирования.

- тестирование позволяет удостовериться, что программа соответствует требованиям;
- тестирование позволяет определить условия, при которых программа ведёт себя некорректно.

Основные «эпохи тестирования»

В 80-х годах произошло ключевое изменение места тестирования в разработке ПО: вместо одной из финальных стадий создания проекта тестирование стало применяться на протяжении всего цикла разработки (software lifecycle), что позволило в очень многих случаях не только быстро обнаруживать и устранять проблемы, но даже предсказывать и предотвращать их появление.

В этот же период времени отмечено бурное развитие и формализация методологий тестирования и появление первых элементарных попыток автоматизировать тестирование.



Основные «эпохи тестирования»

В 90-х годах произошёл переход от тестирования как такового к более всеобъемлющему процессу, который называется «обеспечение качества (quality assurance), охватывает весь цикл разработки ПО и затрагивает процессы планирования, проектирования, создания и выполнения тест-кейсов, поддержку имеющихся тест-кейсов и тестовых окружений.

Тестирование вышло на качественно новый уровень, который естественным образом привёл к дальнейшему развитию методологий, появлению достаточно мощных инструментов управления процессом тестирования и инструментальных средств автоматизации тестирования, уже вполне похожих на своих нынешних потомков.

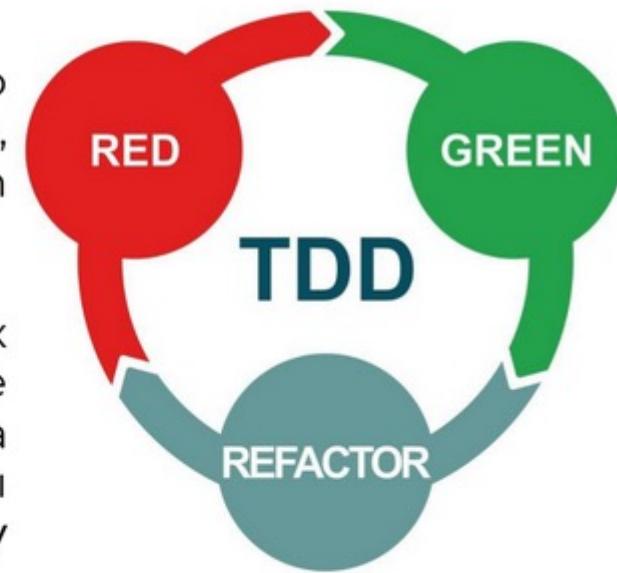


Основные «эпохи тестирования»

В нулевые годы нынешнего века развитие тестирования продолжалось в контексте поиска всё новых и новых путей, методологий, техник и подходов к обеспечению качества.

Серьёзное влияние на понимание тестирования оказало появление гибких методологий разработки и таких подходов, как «разработка под управлением тестированием (test-driven development, TDD)».

Автоматизация тестирования уже воспринималась как обычная неотъемлемая часть большинства проектов, а также стали популярны идеи о том, что во главу процесса тестирования следует ставить не соответствие программы требованиям, а её способность предоставить конечному пользователю возможность эффективно решать свои задачи.



Цель тестирования

Цепь тестирования — это проверка соответствия ПО предъявляемым требованиям, обеспечение уверенности в качестве ПО, поиск очевидных ошибок в программном обеспечении, которые должны быть выявлены до того, как их обнаружат пользователи программы.

Обнаружение и исправление ошибок составляет 40-80% общей стоимости разработки ПО. Такие большие средства тратятся на поиск и устранение ошибок, которые есть в любом продукте.

Цепь тестировщика — найти максимально возможное количество ошибок, и чем серьезнее обнаруженные проблемы — тем лучше. В итоге, максимум ошибок исправляется, и качество ПО повышается.

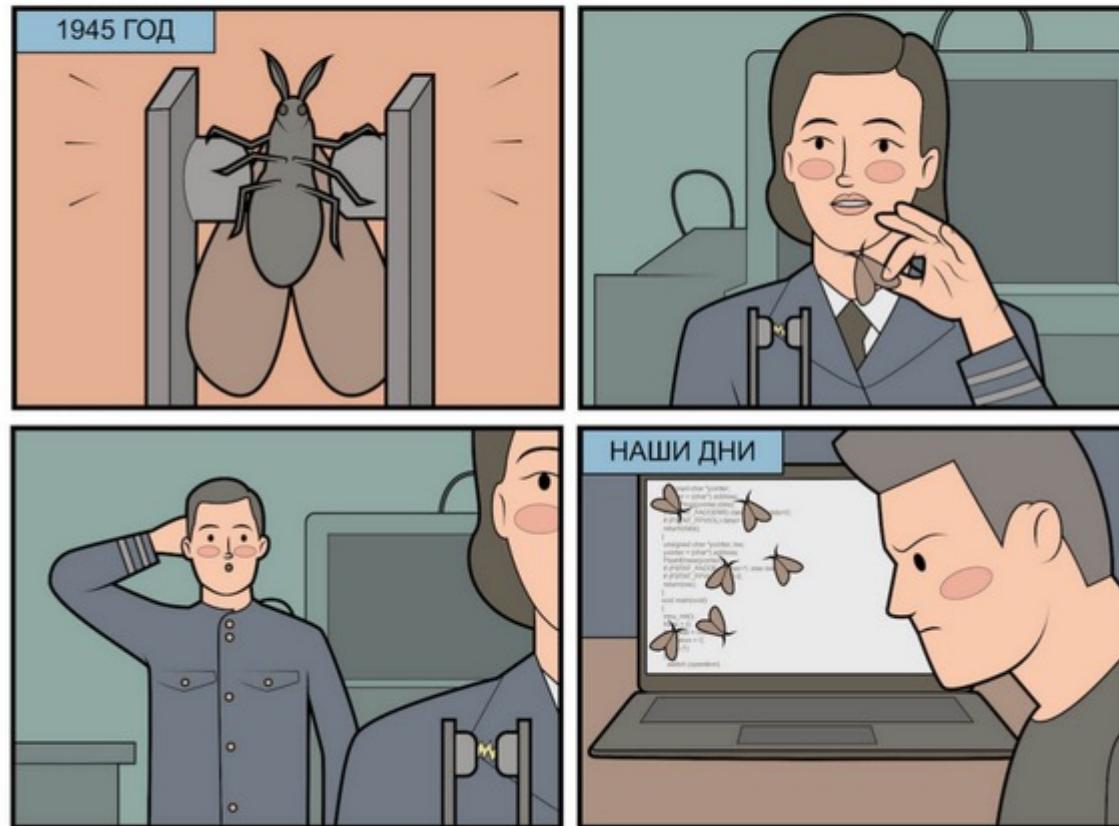


Баг

9 сентября 1945 года Грейс Мюррей Хоппер проверяла первый компьютер Mark I на неисправности. Оказалось, что внутри застрял мотылек.

В этот момент в комнату вошёл сослуживец и поинтересовался, чем занимается Грей.

Она ответила, что очищает компьютер от насекомых (англ. bug — насекомое, debugging — избавление от насекомых).



Баг

Баг — это несоответствие ожидаемого и фактического результатов проверок сервиса.

Ожидаемый результат (ОР) — то, как сервис должен работать согласно требованиям.

Фактический результат (ФР) — то, как сервис работает на самом деле.

Ожидаемый результат



Фактический результат



Задание

Ты собираешься домой и берёшь машину в каршеринге.

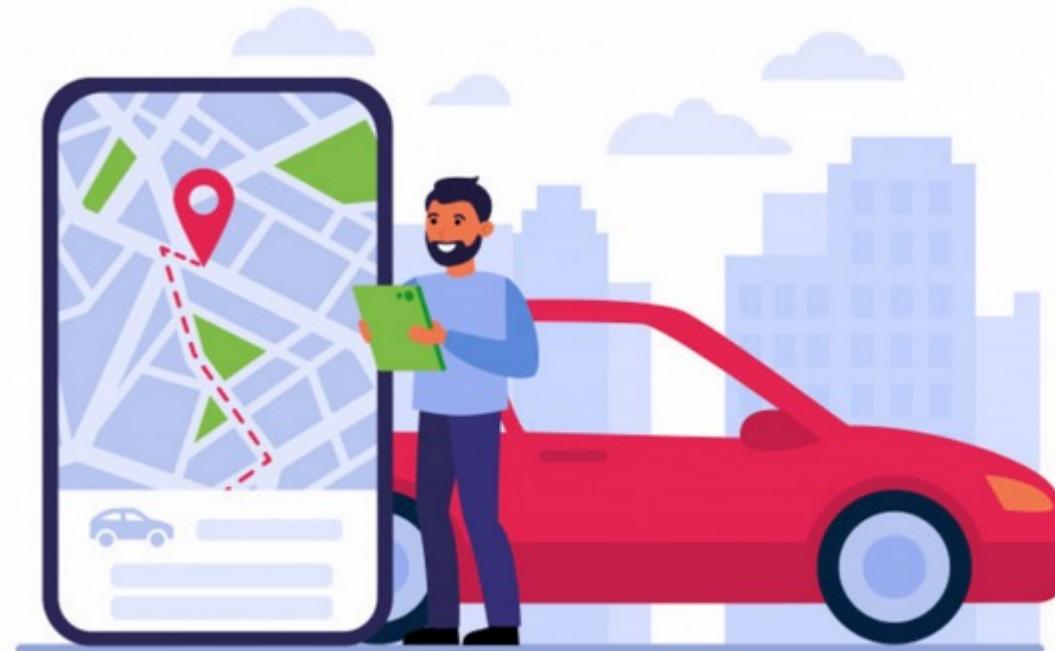
Сервис показывает примерное время поездки:

14 минут за 112 рублей.

По дороге образовалась пробка — поездка заняла 22 минуты
и стоила 176 рублей.

Что здесь — ОР, а что — ФР?

- ОР: Стоимость поездки 112 рублей;
- ФБ: Стоимость поездки 112 рублей;
- ОР: Стоимость поездки 176 рублей;
- ФБ: Стоимость поездки 176 рублей.



Решение

Ты собираешься домой и берёшь машину в каршеринге.

Сервис показывает примерное время поездки:

14 минут за 112 рублей.

По дороге образовалась пробка — поездка заняла 22 минуты
и стоила 176 рублей.

Что здесь — ОР, а что — ФР?

- ОР: Стоимость поездки 112 рублей;
- ФБ: Стоимость поездки 112 рублей;
- ОР: Стоимость поездки 176 рублей;
- ФБ: Стоимость поездки 176 рублей.



Задание

Ты тестируешь сервис Авиабилеты.

Требование: если билеты есть в наличии, результаты поиска отображаются списком.

Определи, верно ли утверждение: ОР — при поиске билета в город, куда не летают самолёты, появляется сообщение «Билеты не найдены».

- да;
- нет;
- нет однозначного ответа;
- не знаю.



Решение

Ты тестируешь сервис Авиабилеты.

Требование: если билеты есть в наличии, результаты поиска отображаются списком.

Определи, верно ли утверждение: ОР — при поиске билета в город, куда не летают самолёты, появляется сообщение «Билеты не найдены».

- нет однозначного ответа;

Без требований невозможно определить ОР, поэтому нельзя ответить точно. Тогда лучше уточнить эту деталь у коллег.



Как определить качество ПО

Абстрактное понятие «качество ПО» всегда на слуху. Если спросить тестировщика или программиста, что это такое, то каждый выдаст свое толкование.

Если программа делается по заказу конкретного клиента, то он может участвовать в ее проектировании. В этом случае качество будет означать точное соответствие спецификации клиента.

Еще один критерий качества – надежность. Чем она выше, тем реже сбоит программа, особенно такая, где сбой влечет, например, потерю данных. Но надежность – далеко не единственный критерий.

Если программа не дает юзеру сделать то, что он считает важным, – он не будет ею доволен. А если пользователь разочарован – значит, качество ПО невысокое.

Как определить качество ПО

Качество ПО определяется:

- возможностями, из-за которых она нравится пользователю;
- недостатками, которые заставляют юзера приобретать другую программу.

Главное, что тестировщик должен сделать для повышения качества программы, – это найти ее недостатки, сбои и явные ошибки.

Если руководитель проекта принимает решение в последний момент добавить какую-нибудь очень необходимую функцию, это тоже может повысить качество.

Ни функциональность, ни надежность программы не бывают стопроцентными.

Качество – это баланс между этими характеристиками.



Как определить качество ПО

Рассмотрим определение «качества ПО» в международных стандартах:

- Качество программного обеспечения – это степень, в которой ПО обладает требуемой комбинацией свойств. [1061-1998 IEEE Standard for Software Quality Metrics Methodology];
- Качество программного обеспечения – это совокупность характеристик ПО, относящихся к его способности удовлетворять установленные и предполагаемые потребности. [ISO 8402:1994 Quality management and quality assurance];
- А в соответствии со стандартом ISO 9126 [1-4] качество представляется как совокупность понятий внутреннего качества , связанного с характеристиками ПО самого по себе, и внешнего качества , характеризующего ПО с точки зрения его поведения, а также качества ПО при использовании в различных контекстах.

Характеристики качества ПО

На сегодня наиболее популярна Многоуровневая Модель Качества программного обеспечения (описывает внутреннее и внешнее качество ПО), которая представлена в серии стандартов ISO 9126.

На верхнем уровне выделяют 6 основных характеристик качества ПО, каждую из которых описывают набором атрибутов, имеющих соответствующие метрики для оценок.



Характеристики качества ПО

Согласно этой модели, функциональность (Functionality) определяется умением ПО решать задачи, которые соответствуют известным и предполагаемым потребностям юзера при заданных условиях использования.

Т.е. эта характеристика отвечает за то, что ПО работает безошибочно и точно, функционально совместимо, имеет соответствие стандартам отрасли и защиту от несанкционированного доступа.

Надежность (Reliability) – умение ПО выполнять требуемые задачи в поставленных условиях на заданном отрезке времени.

Атрибуты этой характеристики – это завершенность и целостность всей системы, способность к самостоятельному и корректному восстановлению, отказоустойчивость.



Характеристики качества ПО

Удобство использования (Usability) – интуитивная понятность использования и изучения, удобство ПО для пользователя.

Эффективность (Efficiency) – способность ПО обеспечивать необходимый уровень производительности при выделенных ресурсах, времени и других заданных условиях.

Удобство сопровождения (Maintainability) – легкость для анализа, тестирования, изменения для исправления дефектов, для реализации новых задач, для облегчения дальнейшего сопровождения и возможности легкой адаптации.

Портативность (Portability) – характеристика ПО с точки зрения беспроблемности его переноса с одного окружения (software/hardware) на другое.



Характеристики и атрибуты качества ПО

Что ПО должно делать?

Пример: позволить клиенту оформить заказ(ы) и обеспечить доставку.

Насколько надежно?

Например: работа 7 дней в неделю и 24 часа в сутки; допустимая неработоспособность – 3 часа в год.

Никакие введенные данные при сбое не должны потеряться.

Каково удобство пользования?

Например: покупатель должен легко найти необходимый товар.

Насколько эффективно?

Например: поддерживает обслуживание до 10000 запросов\сек; время отклика на запрос при максимальной загрузке не более 3 секунд.



Характеристики и атрибуты качества ПО

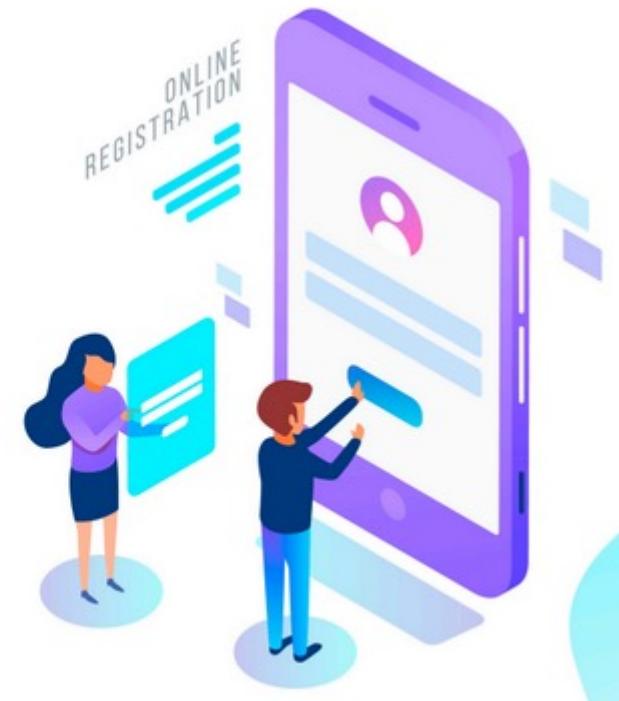
Удобно ли в сопровождении?

Например: для добавления нового вида запросов не должно требоваться больше 3 человека\дней.

Насколько переносимо и заменяемо?

Например: ПО должно функционировать на системах Linux, Windows и MacOS X;
поддерживать работу с документами Word и HTML;
уметь сохранять файлы отчетов MS Excel, HTML, RTF;
сопрягаться с имеющейся системой записи данных.

И это далеко не исчерпывающее понятие качества ПО.



Метрики качества ПО

Помимо перечисленных характеристик и атрибутов качества, стандарт ISO 9126 определяет наборы метрик для оценки каждого атрибута:

- Полнота реализации функций – % реализованных функций относительно перечисленных в требованиях.

Применяется для измерения функциональной пригодности.

- Корректность реализации функций – правильность реализации функций относительно требований.

Применяется для измерения функциональной пригодности.

- Отношение числа обнаруженных дефектов к прогнозируемому.

Применяется в определении зрелости.



Метрики качества ПО

Помимо перечисленных характеристик и атрибутов качества, стандарт ISO 9126 определяет наборы метрик для оценки каждого атрибута:

- Отношение числа проведенных тестов к общему их числу.
Применяется в определении зрелости.
- Отношение числа доступных проектных документов к указанному в их списке. Применяется для измерения удобства проведения анализа.
- Наглядность и полнота документации.
Применяется для оценки уровня понятности.



Как контролировать качество системы?

Чтобы понять, что программа делает то, что необходимо, и ничего лишнего, а также надежна, удобна при пользовании и переносима, применяются процессы верификации и валидации.

Верификация – проверка, что ПО реализовано в соответствии с определенными требованиями к нему, или что очередной этап разработки выполнен в соответствии с ограничениями, которые были сформулированы на предшествующих шагах.

Валидация – проверка, что продукт правильный; подтверждение, что он удовлетворяет требованиям и ожиданиям заказчика, юзеров и других заинтересованных лиц.



Категории программных ошибок

Нет ни абсолютно точного определения ошибок, ни такого критерия их наличия в программе. Можно лишь показать, насколько программа не выполняет свою задачу, – а это **субъективная характеристика**.

Программная ошибка – неисправность в разработке программы, вызывающая несоответствие ожидаемых и полученных результатов при ее работе.

Ошибка может появиться на стадии программирования, формулировки требований или при проектировании, либо из-за некорректных данных. Ошибкой может быть также нечто не соответствующее ожиданиям заказчика.

Категории программных ошибок

Программные ошибки еще называют **багами** (от англ. «bug»).

Термин используется в связи с легендой, по которой при **тестировании** вычислительной **машины Mark II** в Гарвардском университете был обнаружен мотылек, застрявший между контактами реле.

Извлеченнное насекомое было вклеено скотчем в технический дневник с надписью **«First actual case of bug being found»**.

Категории программных ошибок

Рассмотрим **13 категорий ошибок в ПО**, которые охватывают все возможные варианты:

1. Ошибки пользовательского интерфейса

- Функциональность (если задачу, которая предусмотрена программой, трудно выполнить, она решается «криво» или при каких-либо обстоятельствах вовсе не может быть решена):
 - а. в программе нет функции, описанной в спецификации или очевидно необходимой функции;
 - б. функция программы должна делать одно (как правило, описанное в спецификации), а выполняет что-то другое.

Категории программных ошибок

- Взаимодействие программы с пользователем (UI, UX)

Насколько трудно пользователю понять, как работать с ПО? Есть ли подсказки на экране? Понятны ли они? Есть ли в программе справка и полезна ли она? Насколько корректно программа показывает пользователю его ошибки и поясняет, как их исправлять? Есть ли в программе элементы, которые могут вызвать раздражение пользователя?)

- a. Нет названия программы;
- b. Нет индикатора оставшегося времени обработки задания;
- c. После выбора пользователем определенного режима или команды на экране остается информация предыдущего режима, большая часть которой – излишняя или не относится к делу;

Категории программных ошибок

- d. Ошибки в правописании;
- e. Одна и та же функция не может иметь несколько значений: либо «ОК», либо «Сохранить» – надо выбирать;
- f. В сообщениях об ошибках не должно быть восклицательных знаков, слов «авария», «сбой», «нарушение», «потеря данных», шрифта красного цвета;
- g. Сообщение об ошибке должно быть информативным: содержать причину и способ решения. Сообщения типа «ERROR 010» – недопустимы;
- h. Не выделены активные элементы экрана.

Категории программных ошибок

- Организация программы

Легко ли потеряться в вашей программе? Нет ли похожих или одинаковых команд? Какие ошибки чаще всего совершает юзер? На что тратится больше всего времени и по какой причине?

- а. Плохое меню: связанные команды должны быть объединены в одну группу. группы должны четко отделяться друг от друга;
- б. Диалоговые окна должны появляться в одном и том же месте дисплея, а их текст должен иметь одинаковый шрифт и выравнивание. Заголовок окна должен называть команды, которая его открыла. Поля ввода и выбора должны иметь одинаковое выравнивание;

Категории программных ошибок

- с. Гармоничное сочетание цветов;
- д. Предсказуемость поведения программы при возникновении ошибки. Программа не должна внезапно завершаться, перезапускаться, выдавать белый экран;
- е. В меню не должно быть невозможных для выполнения команд. На экране будет написано: «Для получения справки нажмите <F1>» – а когда пользователь нажмет эту клавишу, программа скажет: «К сожалению, справка по данной теме отсутствует»;
- ф. Множество путей к одному и тому же месту. Если создается ощущение, что в программе можно откуда захочешь попасть куда угодно, значит ее внутренняя структура требует реорганизации. Также эта ситуация чревата «тухлыми ссылками», если тестируемая программа – веб-сайт.

Категории программных ошибок

Пропущенные команды

Чего в программе не хватает? Нет ли у некоторых действий странных или неэффективных способов? Допускается ли хотя бы небольшая настройка?

- a. Пользователь должен иметь возможность отменить последнее действие;
- b. Пользователь должен иметь возможность прервать текущее задание и вернуться к начальному состоянию;
- c. При удалении больших объемов данных ПО обязательно должно спрашивать у пользователя подтверждение действий;
- d. Средства защиты программы не должны быть навязчивыми;
- e. При ошибке в середине длинной последовательности действий некоторое ПО заставляет юзера все повторить сначала.

Категории программных ошибок

Производительность

Важный момент в интерактивном ПО – скорость. Есть ли у пользователя впечатление, что ПО функционирует медленно, с большой задержкой реакции?

- a. Низкая скорость работы ПО;
- b. Медленное отображение вводимых данных: заторможенное перемещение курсора, голосовой ввод;
- c. Программа, которая занята другими приложениями, не распознает ввод текста. Надо, чтобы программа запоминала вводимые данные и отображала их чуть позже;
- d. Надоедливые напоминания и вопросы вроде: «Диск заполнен на 89%. Пожалуйста, освободите место»;
- e. Длинное меню и красивые картинки должны мгновенно прорисовываться.

Категории программных ошибок

Выходные данные

Большинство программ выдают выходные данные: на экране, печатают или сохраняют в файлы.

Корректно ли сформированы отчеты?

Наглядные ли графики, достаточно четкие ли они на бумаге?

Есть ли у программы достаточная гибкость настройки под конкретного пользователя?

Категории программных ошибок

2. Обработка ошибок

Процедуры обработки ошибок – важная задача ПО. В этом месте часто бывают недочеты. Однако, правильно выявив ошибку, программа не всегда выводит о ней полное и информативное сообщение.

- Если исполняемый код находится в нескольких файлах, кто-то может попробовать воспользоваться новой версией одних файлов со старой версией других;
- Ввод неправильных данных – «защита от дурака»;
- Переполнение: если результат вычислений настолько большой, что программа не в состоянии их обработать;
- Невозможные значения. Например, дата 30 февраля;

Категории программных ошибок

3. Ошибки, связанные с обработкой граничных условий

Простейшие граничные условия – числовые. Но есть и множество других граничных ситуаций. Внутри диапазонов программа отрабатывает в большинстве случаев идеально, а вот на границах возникают неожиданные отклонения.

Числовые ограничения

У треугольника ровно 3 стороны, сумма его углов равна 180 градусам. В одном байте могут храниться значения от 0 до 255;

Количественные ограничения

Длина строки – 80 символов. Что произойдет при вводе 79, 80, 81;

Пространственные ограничения

Программа выводит график в прямоугольнике определенного размера. Что произойдет, если одну из точек графика нарисовать вне прямоугольника?

Ограничения времени

Допустим, есть 30 секунд для ответа на телефонный звонок. Через 30 секунд телефон перестает звонить, и вызов перенаправляется оператору. Что произойдет, если трубку снимут на 30 секунде?

Ограничения объема памяти

Категории программных ошибок

3. Ошибки вычислений

Даже самые простые арифметические операции всегда чреваты ошибками.

Самые распространенные — ошибки округления.

- Выполнение сложения вместо вычитания;
- Выражения с большим количеством скобок;
- Неправильный порядок операторов.
- Переполнение и потеря значащих разрядов.
- Ошибки отсечения и округления
- Неверная формула.

Категории программных ошибок

3. Начальное и последующие состояния

Случается, что при выполнении какой-либо функции сбой происходит всего лишь раз — при самом первом обращении к ней,

Например, на экране — искаженное изображение. Может быть, неправильно выполняются расчеты, начнутся бесконечные циклы или операционная система выдаст сообщение о нехватке памяти.

Причиной может быть отсутствие файла с инициализационной информацией. После первого же старта программа сделает этот файл, и дальше все заработает корректно. Выходит, эту ошибку нельзя повторить (точнее, для ее повторения нужно установить новую копию программы). Но не надо думать, что ошибка, которая проявляется лишь при первом запуске программы, безвредная: это будет первый баг, с которым столкнется каждый новый юзер.

Категории программных ошибок

3. Ошибки управления потоком

Если по логике программы за первым действием должно следовать второе, а она выполняет третье, это означает, что в управлении потоком – ошибка.

Ее сложно не заметить: в худшем случае в работе программы будет сбой.

- «Зависание» компьютера. Причина: бесконечные циклы, бесконечное ожидание ответа какого-либо устройства;
- Завершение работы программы.
- Неверно определены действия для всех остальных случаев. Предположим, программист полагает, что переменная VAR может принимать только четыре значения. Он явно указывает три из них, а четвертое обрабатывает как «остальные». Но будут ли запрограммированные на этот случай действия правильными, если переменная примет пятое или шестое значение?
- Установленные условия проверок не должны пересекаться.
- Неверно заданные условия проверки. Программа выполняет Задание_16, только когда $(\text{VAR} < 6) \text{ AND } (\text{VAR} > 18)$. Это означает, что Задание_16 не выполняется никогда.

Категории программных ошибок

3. Ошибки передачи или интерпретации данных

Модули могут передавать данные между собой или другим программам множество раз, и в какой-то момент они могут быть потеряны или неправильно интерпретированы. Изменения, которые внесены одной частью программы, могут потеряться или достичь не всех частей системы, где они имеют важность.

- Неправильная интерпретация данных. Программа передает подпрограмме температуру по Цельсию, а та интерпретирует ее по Фаренгейту;
- Неадекватная информация об ошибке. Столкнувшись с ошибкой, программа не установила ее флаг. Или выдала сигнал ошибки без сопутствующей информации, в результате чего вызывающая программа не может знать, как ее обработать;
- Затирание кода другого процесса. Может возникнуть в случае, когда два процесса имеют доступ к одной и той же области памяти;
- Не сохранены введенные данные. Запрошенные у пользователя данные не сохраняются программой. Причина: недоступность файла, в котором должна сохраняться эта информация.

Категории программных ошибок

4. Ситуация гонок

Классическая ситуация гонок описывается так: в системе ожидаются два события, А и Б. Первым может произойти любое из них. Но если первым происходит событие А, программа выполнение продолжает, а если Б, то происходит сбой.

Программист думал, что первым всегда должно быть событие А, и не учел, что Б может победить в гонках. Тестировать ситуации гонок сложно. Они наиболее типичны для систем, где параллельно выполняются взаимодействующие процессы и потоки, а также для многопользовательских систем реального времени. Ошибки в таких системах трудно воспроизвести, и они требуют много времени на выявление.

Гонки при обновлении данных. Клиент банка получил \$500 и потратил \$100, а до этого на его счете было \$1000. Первая процедура считала с диска остаток в \$1000 и вычла из него \$100. До того, как она записала результат на диск, вторая процедура считала остаток и добавила к нему \$500. Затем первая процедура записала свой результат, а вторая – свой.

В результате вместо \$1400 остаток получился равным \$1500;

Категории программных ошибок

4. Перегрузки

ПО не удается справиться с повышенными нагрузками. Например, программа может не выдерживать работу с большим объемом данных. Сбои могут возникать из-за нехватки памяти или других нужных ресурсов. У каждой программы свои пределы. Вопрос в том, соответствуют ли реальные возможности и требования программы к ресурсам ее спецификации, и как программа себя поведет при перегрузках.

- Требуемый ресурс недоступен (диск или стек полон, очередь печати заполнена, в принтере нет ленты);
- Потеря информации о нажатых клавишах из-за недостаточного размера буфера ввода или очереди.

Категории программных ошибок

4. Аппаратное обеспечение

Программы могут запрашивать у устройств неверные данные, игнорировать их сообщения об ошибках, пытаться воспользоваться занятыми или отсутствующими устройствами. Программа не должна «глючить» при попытке обращения к устройствам.

- Неверный адрес устройства;
- Устройство недоступно;
- Данной программе или устройству доступ к устройству запрещен.

Документация

Документация – это часть программного продукта. Если она плохо написана, у пользователя может появиться мысль, что и сама программа не лучше.

Категории программных ошибок

4. Контроль версий

Случается, что старые ошибки появляются вновь по причине того, что про грамму скомпоновали с устаревшей версией одной из подпрограмм. Версии всех составляющих частей необходимо всегда контролировать.

Также надо убедиться, что корректны все сообщения об авторских правах, названии и номере версии ПО. Контроль программы и исходного кода обычно производит группа контроля качества.

- В программе всплывают старые ошибки, которые ранее были устраниены (программист собрал старую версию одной программы с новыми версиями остальных);
- Ошибка, которую исправили в одном месте, обнаружилась в другом (программист применял один и тот же код в нескольких модулях. В одном месте изменил, а про другие – забыл);
- Неверный номер версии программы в заголовке экрана;

Категории программных ошибок

4. Ошибки тестирования

Если программист допускает по полторы ошибки на каждую строку программного кода, то сколько их допускает тестировщик на каждый тест?

Поиск ошибок, которые допустили тестировщики, – дело обычное. Бывают случаи, что ошибки тестировщика указывают на проблемы пользовательского интерфейса: если программа вынуждает пользователя совершать ошибки, то с ней что-то не так.

Категории программных ошибок

Конечно, множество ошибок тестирования обусловлены неверными тестовыми данными.

- Пропущены ошибки в программе. После этого надо улучшить процедуру тестирования;
- Не задокументирована ошибка\проблема;
- Не выполнен запланированный тест;
- Не обратили внимание на предложения программистов – программист знает, что писалось в спешке, какие тесты выявили больше важных ошибок, и где в программе могут встретиться ошибки подобного рода;
- Документирование ошибки, которой нет на самом деле. Может оказаться, что проблема связана с неправильными действиями тестировщика, его непониманием программы: «После залогинивания в системе в браузере IE, в браузере FireFox авторизационные данные не вводятся автоматически»;
- Ошибка выявлена и забыта.