GeekBrains

Основы ручного тестирования

Регрессионное тестирование

На этом уроке

1. Узнаем, что такое регрессионное тестирование и зачем оно нужно.
2. Рассмотрим эвристики тестирования.

Оглавление

Регрессионное тестирование

Регрессионное и подтверждающее тестирование

При разработке новых функций или починке багов в систему вносятся изменения, которые привносят риск, что программа сломалась в другом месте. Для минимизации риска и повышения качества продукта тестировщики проводят регрессионное и подтверждающее (или повторное) тестирование.

Цель подтверждающего тестирования — удостовериться, что дефект исправлен. Тестировщик заново проходит сценарии, которые завершились с ошибкой из-за найденного дефекта, и повторно выполняет шаги, вызвавшие сбой.

Изменение, внесённое в одну часть кода, может случайно повлиять на поведение других частей ПО. Такие непреднамеренные побочные эффекты называются регрессиями. **Регрессионное тестирование** включает выполнение тестов для обнаружения непреднамеренных побочных эффектов.

Подтверждающее и регрессионное тестирование проводятся на всех уровнях тестирования: от компонентного до системного.

Регрессионное тестирование строится на основе тест-кейсов или чек-листов, которые использовались ранее для проверки новой функциональности.

Для **полного регрессионного тестирования** используются все тест-кейсы, созданные для проверки функциональности. Такой вид регрессионного тестирования улучшает качество проверки и обнаруживает больше дефектов, но требует много времени и усилий тестировщиков.

При **выборочном регрессионном тестировании** выполняются только самые важные и приоритетные тест-кейсы из всех наборов. Это быстрее, но эффективность ниже.

Добавление новых тест-кейсов в тестовые наборы происходит не только при появлении новой функциональности, но и в случае исправления дефекта, чтобы исключить его повторное появление в будущем. Так улучшают тестовое покрытие — в случае с выборочным регрессионным тестированием, когда тестовое покрытие неполное.

Помимо добавления новых тест-кейсов в регрессионные наборы, требуется также удалять из них дубликаты — тест-кейсы, которые проверяют одно и то же требование, но не улучшают тестовое покрытие. Стоит также избавляться от тест-кейсов, которые обнаруживают ошибки, уже выявленные другими тест-кейсами.

Важно следить за актуальным состоянием тест-кейсов в наборах для регрессионного тестирования. Периодически проверять и заменять их, чтобы избежать «эффекта пестицида» и «замыливания».

Постоянное добавление тест-кейсов приводит к тому, что регрессионное тестирование становится трудоёмким и требует больше времени. Проходить одни и те же сценарии утомительно для тестировщиков. Поэтому в работе с регрессионными тест-кейсами часто прибегают к автоматизации.

Функциональные критерии выбора тестов

При подборе тестов для регрессионного тестирования тестировщик руководствуется рядом функциональных критериев. Функциональные критерии обеспечивают контроль выполнения требований заказчика в программном продукте.

Виды функциональных критериев

Тестирование пунктов спецификации — набор тестов обеспечивает проверку каждого тестируемого пункта. Спецификация требований содержит множество требований к программному продукту и каждое из них проверяется хотя бы одним тестом.

Тестирование классов входных данных — набор тестов обеспечивает проверку представителя от каждого класса входных данных. Для оптимального тестового покрытия используются техники тест-дизайна: классы эквивалентности, граничные значения, попарное тестирование.

Тестирование правил — набор тестов обеспечивает проверку каждого правила, если входные и выходные значения описываются набором правил. Для тестирования правил обычно применяются таблицы решений.

Классификация тестов при отборе для регрессионного тестирования

Создание наборов регрессионных тестов начинается с множества исходных тестов. Все исходные тесты подразделяются на четыре группы.

Тесты, пригодные для повторного использования

Тесты, которые уже запускались, но затрагивают только компоненты программы, не претерпевшие изменений. При повторном выполнении выходные данные таких тестов совпадут с выходными данными, полученными ранее. Следовательно, такие тесты не требуют перезапуска.

Тесты, требующие повторного запуска

Тесты, которые уже запускались, но требуют перезапуска, поскольку затрагивают, по крайней мере, один изменённый компонент, подлежащий повторному тестированию. При повторном выполнении такие тесты выдают результат, отличный от результата, показанного ранее.

Устаревшие тесты

Тесты, более неприменимые к программе и непригодные для дальнейшего тестирования, поскольку они затрагивают только удалённые при изменении программы компоненты. Их можно удалить из набора регрессионных тестов.

Новые тесты, которые ещё не запускались и могут использоваться для тестирования

Сразу после создания тест вводится в структуру тестовой модели как новый. После исполнения новый тест переходит в категорию тестов, пригодных для повторного использования либо устаревших. Если выполнение теста способствовало увеличению текущей степени покрытия требований, тест помечается как пригодный для повторного использования. В противном случае он фиксируется как устаревший и отбрасывается.

Окончательный набор регрессионных тестов включает:

- тесты, пригодные для повторного использования;
- тесты, требующие повторного запуска;
- новые тесты.

Устаревшие и избыточные тесты удаляются из набора, поскольку не проверяют новые функциональные возможности и не увеличивают покрытие.

Типы регрессии

Регрессия багов — попытка доказать, что исправленная ошибка на самом деле не исправлена.

Регрессия старых багов — попытка доказать, что недавнее изменение кода или данных сломало исправление старых ошибок, т. е. старые баги стали снова воспроизводиться.

Регрессия побочного эффекта — попытка доказать, что недавнее изменение кода или данных сломало другие части разрабатываемого приложения.

Эвристики тестирования

Эвристики — это быстрые способы решения проблемы или принятия решения. Процесс тестирования на основе эвристик — это технология тестирования алгоритмов, приложений и программ, при использовании которой стратегия тестирования основывается на предыдущем опыте и данных о вероятности наступления различных событий.

Тестировщики часто сталкиваются с эвристиками в форме чек-листов, мнемоник и чит-листов. Их иногда называют оракулами или моделями. Неважно, как они называются и в какой форме их

получили: если они служат когнитивным коротким путём, помогающим решать проблемы и принимать решения, то это эвристики.

Маша и медведи

Эвристика концентрируется на подходе «слишком много, слишком мало, и в самый раз».

Применение: тестирование поля ввода.

1. Слишком много — очень длинная строка или большое число: имя из 100 букв, возраст человека — 1000 лет.
2. Слишком мало — очень короткая или пустая строки, ноль.
3. В самый раз — типичная для параметра длина строки или число: имя из 4–10 букв, возраст — 35 лет.

RCRCRC

Мнемоника RCRCRC поддерживает регрессионное тестирование. Буквы расшифровываются

1. Recent — недавнее: новый инструментарий или починка багов.
2. Core — ключевое: главные функции, дымовое тестирование.
3. Risky — рискованное: сложная логика или недостаточность требований.
4. Configuration — конфигурационное: изменения в файлах настроек.
5. Repaired — исправленное: починка багов.
6. Chronic — хроническое: места, где баги возникают постоянно.

FEW HICCUPS

FEW HICCUPS — это мнемоника, которая помогает запомнить ключевые слова для источников ожидаемого результата тестирования. Иногда их называют оракулами. Оракулы особенно полезны, если спецификация отсутствует или содержит неадекватную информацию.

Буквы в мнемонике означают:

1. Familiar — известность. ПО не воспроизводит известные проблемы других программных продуктов.
2. Explainability — объяснимость. Работа ПО понятна, пользователь может её объяснить.
3. World — мир. ПО соответствует знаниям и фактам окружающей действительности.
4. History — история. Новая версия ПО не противоречит предыдущей.

5. Image — имидж. ПО соответствует имиджу компании, которая его разрабатывает.
6. Comparable product — конкуренты. ПО не хуже, чем аналогичные продукты конкурентов.
7. Claims — заявления. ПО выполняет то, что заявляется в рекламе, пресс-релизах и т. д.
8. User Expectations — ожидания пользователя. ПО отвечает ожиданиям людей, которые его используют.
9. Product — продукт. Все элементы ПО работают как единое целое.
10. Purpose — цель. ПО решает ту задачу, ради которой оно создавалось.
11. Standards — стандарты. ПО соответствует стандартам, установленным в отрасли.

Есть множество других эвристик, их легко найти в интернете.

Личные эвристики тестирования

Практикующий тестировщик использует собственные эвристики, иногда бессознательно. Это связано с тем, что использование и создание эвристик — интуитивный процесс. Но также это вызвано и тем, что всё, чем заняты тестировщики в тестировании, рассматривается как эвристика.

Начиная с принципов проектирования тестов и заканчивая тест-идеями, пришедшими в голову во время исследовательского тестирования, всё это рождается благодаря когнитивным коротким маршрутам, помогающим быстро принимать решения и решать проблемы.

Тестировщик может следовать определённой рутине перед началом тестирования, например, открывать конкретные приложения. Опыт говорит, что они понадобятся во время тестирования, и их заблаговременное открытие позволит решить проблему утраты концентрации посреди сессии, чтобы их запустить.

Выявить личные эвристики непросто. Зачастую мы бессознательно создаём их на основании прошлого опыта. Наилучший способ заметить их — это размышлять над своими действиями во время и после каждой тест-сессии.

Регулярное размышление над своим тестированием выявит паттерны в поведении и потенциально определит личные тест-эвристики. Эти размышления также выявят когнитивные искажения.

Создать свою

Первый шаг в осознании и разработке собственной тест-эвристики — определить, где она уже используется. Это очень сложный процесс. Мы привыкли объяснять, что делаем, но объяснить, почему, значительно сложнее — там и прячутся эвристики.

Обдумывание

Обдумывание тестирования часто вскрывает паттерны. А они, возможно, результат применения эвристик. Регулярное обдумывание во время каждой тест-сессии и после неё повысит шансы на идентификацию эвристик. К примеру, можно потратить несколько минут после каждой сессии, чтобы спросить себя «почему я провёл именно эти тесты», и «почему я провёл их именно в таком порядке». Если Использовать хорошие тест-заметки, можно выявить паттерны, делая заметки и пересматривая их.

Вербализация

Другой способ идентификации и создания собственных эвристик — это вербализация (проговаривание, объяснение) тестирования другим людям. Процесс вербализации поощряет обдумывание, так как мы ищем слова, чтобы объяснить, что делаем или сделали. Обсуждая решения с другими людьми, обнаруживаем нюансы, о которых не знали, и создаём новые связи между мыслями и идеями, описывая их. Можно рассказывать это самим себе, используя эвристику резиновой уточки.

Применение теории на практике

Эвристики – мощный инструмент, и он ещё мощнее, если тестировщик отдаёт себе отчёт в существовании эвристик, понимает их и может сознательно применять их в тестировании.

Контрольные вопросы

1. На что опираться, составляя набор регрессионных тестов?
2. Какие эвристики тестирования заинтересовали вас?
3. Придумайте свою эвристику тестирования или той области, в которой работаете сейчас.

Практическое задание

Маркетологи сделали новую версию лендинга, и теперь известно, что появятся изменения:

1. В Header добавятся логотипы новых партнёров.
2. В расписании изменятся даты и наполнение уроков.
3. Изменятся ссылки на все нормативные документы.
4. Разработчики проведут полный рефакторинг кода, отвечающего за анимацию на странице.
5. Изменится ссылка на видеопрезентацию.

Откройте документ, созданный в практическом задании №2. На вкладку «Регресс» скопируйте только те пункты чек-листа, которые нужно перепроверить. Времени мало, проверить весь чек-лист не успеем.

*Есть и другие эвристики тестирования, помимо тех, что разбирались на лекции. Поищите их в интернете и напишите, какие вам понравились больше. Придумайте собственную эвристику.

(Тест для самопроверки https://coreapp.ai/app/player/lesson/614344d540a70c676714c758 - сдавать не нужно)

Глоссарий

Подтверждающее тестирование — тестирование с целью удостовериться, что дефект исправлен. Тестировщик заново проходит сценарии, которые завершились с ошибкой из-за найденного дефекта, и повторно выполняет шаги, вызвавшие сбой.

Регрессионное тестирование — тестирование с целью убедиться, что новая функциональность не привела к появлению багов в старой.

Регрессия багов — попытка доказать, что исправленная ошибка на самом деле не исправлена.

Регрессия побочного эффекта — попытка доказать, что недавнее изменение кода или данных сломало другие части разрабатываемого приложения.

Регрессия старых багов — попытка доказать, что недавнее изменение кода или данных сломало исправление старых ошибок, т. е. старые баги стали снова воспроизводиться.

Эвристики — это быстрые способы решения проблемы или принятия решения.

Дополнительные материалы

1. Статья «Эвристика FEW HICCUPS».
2. Статья «Критерии выбора тестов».
3. Статья «Критерии тестирования».
4. Статья «Метод резиновой уточки».