GeekBrains

Основы ручного тестирования

Что такое тестирование?

На этом уроке

1. Узнаем, что такое тестирование и для чего оно нужно.
2. Поговорим о «мягких» навыках тестировщика.
3. Рассмотрим возможные пути карьерного развития тестировщика.

Оглавление

Что такое тестирование?

В широком смысле слова «тестирование» — проверка чего угодно путём проведения тестов, то есть заранее подготовленного списка вопросов, проверок или испытаний. Преподаватели тестируют знания студентов. Изготовители автомобилей проводят краш-тесты, т. е. тестируют безопасность техники. Фармацевтические компании тестируют лекарства на добровольцах.

Указанные процессы имеют ряд общих черт:

- продукт, показатели которого неизвестны — неизвестно, безопасен ли автомобиль;
- заранее подготовленный список проверок — стандартизированные краш-тесты;
- правила проведения исследования — описание процессов, протоколы;
- ожидаемый результат тестирования — если автомобиль сталкивается с препятствием, то срабатывают подушки безопасности;
- фактический результат тестирования — подушка сработала или не сработала.

Точно так же, как лекарства или автомобили, программное обеспечение нуждается в тестировании. Прежде чем отдать пользователю программу, поставщик должен убедиться, что программное обеспечение:

- безопасно;
- надёжно;
- понятно;
- доступно;
- обладает прочими показателями, совокупность которых определяет качество программы.

Тестирование программного обеспечения — это проверка соответствия между реальным и ожидаемым поведением программы, а также выявление, насколько ПО удовлетворяет потребности пользователя. Оно осуществляется на конечном наборе тестов, который составляет тестировщик.

В более широком смысле тестирование — это одна из техник контроля качества, включающая в себя:

- планирование работ (Test Management);
- проектирование тестов (Test Design);
- выполнение тестирования (Test Execution);
- анализ полученных результатов (Test Analysis).

Регламенты тестирования ПО устанавливаются внутри компании, но обычно в основе лежат стандарты ISO или IEK. Более подробно здесь.

Самый распространённый стандарт в области тестирования ПО — ISTQB* — International Software Testing Qualifications Board. Он определяет цели, виды тестирования, место тестирования в жизненном цикле разработки ПО, методы управления и инструменты тестирования.

ISTQB определяет тестирование как процесс, содержащий в себе все активности жизненного цикла как динамические, так и статические. Эти активности включают планирование, подготовку и оценку программного продукта и связанных с этим результаты работ. Задачи:

- определить, что они соответствуют описанным требованиям;
- показать, что они подходят для заявленных целей и для определения дефектов.

*Проходить сертификацию ISTQB имеет смысл, если мы собираемся работать в европейских или американских компаниях. В России наличие сертификата не считается существенным конкурентным преимуществом.

На первый взгляд, определение сложное. Разберём его по частям.

1. Тестирование — это поэтапный процесс:
    a. Планирование тестирования.
    b. Анализ требований.
    c. Проектирование и реализация тестов.
    d. Создание отчётов о ходе и результатах тестирования.
    e. Оценка качества объекта тестирования.
    f. Совокупность этапов — это жизненный цикл тестирования.

2. В тестировании выделяются динамические и статические активности. Динамические предполагают исполнение готового кода — выполнение тестов на работающей версии программного продукта. Статические не требуют запуска кода на исполнение, это анализ макетов, рецензирование документации и кода.

3. Цели тестирования:

    a. Определить, насколько ПО соответствует заявленным требованиям.

    b. Выявить дефекты, т. е. отклонения ПО от требований.

Таким образом, тестирование отвечает на вопрос: каков уровень качества ПО.

Что такое качество?

Качество программного обеспечения — это совокупность характеристик, относящихся к его способности удовлетворять установленные и предполагаемые потребности.

Качеству сложно дать неформальное определение, оно заключается в соответствии с требованиям и пригодности к использованию. Качество программного продукта характеризуется набором свойств, определяющих, насколько продукт «хорош» с точки зрения заинтересованных сторон. К ним относятся: заказчик продукта, спонсор, пользователь, разработчики и тестировщики продукта, инженеры поддержки, сотрудники отделов маркетинга, обучения и продаж. Каждый из участников может иметь своё представление о качестве продукта.

Таким образом, качество — это степень, в которой какой-то компонент, система или процесс отвечает требованиям и ожиданиям пользователя.

Последовательность действий при выборе и оценке критериев качества программного продукта:

1. Определение всех лиц, заинтересованных в исполнении и результатах проекта.

2. Определение критериев качества для каждого заинтересованного лица.

3. Приоритезация критериев с учётом важности:

    - конкретного участника для компании;

    - критерия для этого участника.

4. Определение набора критериев, которые будут отслеживаться и выполняться в рамках проекта исходя из приоритетов и возможностей проектной команды.

5. Постановка целей по каждому из критериев.

6. Определение способов и механизмов достижения каждого критерия.

7. Определение стратегии тестирования исходя из набора критериев.

Основной стандарт — ISO/IEC 25010. Он выделяет критерии качества:

1. Функциональность — степень, где продукт или система выполняют основные функции, то есть решают те задачи, для которых они были созданы.

    a. функциональная полнота — насколько функциональность охватывает задачи пользователя;

    b. функциональная правильность — насколько продукт или система обеспечивают правильные результаты с требуемой степенью точности;

    c. функциональная целесообразность — насколько функции продукта востребованы пользователем.

    Пример: основная функция интернет-магазина — продажа товаров. То есть пользователь должен иметь возможность зайти на сайт магазина, посмотреть предложения в каталоге, увидеть характеристики выбранного товара, добавить его в корзину, удалить лишнее и оплатить выбранные покупки.

2. Эффективность — производительность продукта или системы. Например, как быстро обрабатываются запросы пользователей:

    a. поведение в зависимости от времени — насколько быстро и оптимально обрабатывается запрос пользователя;

    b. использование ресурсов — насколько оптимально ПО расходует ресурсы: оперативную память, жёсткий диск и т. д.;

    c. вместимость — каковы предельные показатели системы, количество пользователей в единицу времени.

3. Надёжность — степень, в которой система, продукт или компонент выполняют указанные функции при определённых условиях в течение конкретного периода; способность быстро восстанавливать работу при отказе оборудования.

    a. завершённость — степень, в которой система, продукт или компонент удовлетворяют потребности в надёжности при нормальной работе;

    b. доступность — степень работоспособности и доступности системы, продукта или компонента при использовании;

    c. отказоустойчивость — насколько стабильно работает ПО при аппаратных или программных сбоях, например, отключилось электроснабжение, исчезло соединение с интернетом;

d. восстанавливаемость — насколько, в случае прерывания или сбоя, продукт или система могут восстановить данные и желаемое состояние системы.
   Пример:

   - стабильная работа системы во время пиковых нагрузок — распродажи «чёрной пятницы» для интернет-магазина;
   - обработка DDoS-атак;
   - стабильная работа при высоких нагрузках (сайты социальных сетей);
   - сохранение данных при сбоях и отказах.

4. Удобство использования — насколько эффективно продукт удовлетворяет потребности конкретного пользователя.

   a. узнаваемость — насколько пользователи распознают, что умеет делать продукт;

   b. обучаемость — насколько продукт даёт возможность пользователю научиться в нём работать эффективно: подсказки, «обучалки» при первом запуске, неизменность основного интерфейса;

   c. работоспособность — степень, в которой продукт или система имеют атрибуты, облегчающие управление и контроль;

   d. защита пользователя от ошибок — насколько система защищает пользователей от ошибок: предотвращает деструктивные действия, показывает сообщения при возникновении сбоев или вводе некорректной информации и т. д.;

   e. эстетика пользовательского интерфейса — степень, в которой интерфейс позволяет приятное взаимодействие для пользователя;

   f. доступность — насколько широкий круг пользователей может пользоваться продуктом. Она зависит от совместимости с программным или аппаратным обеспечением. Например, если приложение работает только на iOS и не работает на Android, то оно недоступно большому числу людей. На доступность также влияет:

      - цветовая гамма — смогут ли использовать дальтоники;
      - размер шрифта и контрастность — смогут ли использовать люди со слабым зрением и т. д.

   Пример: пользователь интуитивно понимает, как покупать в интернет-магазине. Ему понятно, как искать товары в каталоге и оформлять покупку. Приятно находиться на сайте, ничто не отвлекает, нет раздражающих картинок, цветов, надписей.

5. Поддерживаемость — степень адаптируемости приложения к изменениям.

    a. модульность — степень, в которой система состоит из отдельных компонентов, изменение одного оказывает минимальное влияние на другие;

    b. повторное использование — степень, в которой компонент системы используется более чем в одной системе или при создании других компонентов;

    c. пригодность к анализу — степень эффективности и результативности, с которой оценивается влияние на продукт или систему предполагаемого изменения одной или нескольких его частей; диагностируется продукт на наличие недостатков или причин отказов; определяются детали, подлежащие модификации;

    d. модифицируемость — степень, в которой продукт или система могут быть эффективно модифицироваться без внесения дефектов или ухудшения качества продукта;

    e. тестируемость — степень эффективности и результативности, с которой устанавливаются критерии тестирования для системы, продукта или компонента, а также выполняются тесты.
    Пример: при росте популярности приложения есть возможность увеличить его «пропускную способность». При этом социальная сеть, количество пользователей которой стремительно увеличивается, должна иметь возможности для расширения своих ресурсов. В приложение можно легко внедрить новые технологий. При возникновении сбоя с одной из подсистем (загрузка фотографий) вносятся правки, не блокируя доступа к другим частям программы — прослушивание музыки, отправка сообщений.

6. Переносимость — степень эффективности и результативности, с которой система, продукт или компонент переносятся из одного аппаратного, программного, операционного или эксплуатационного окружения в другое.

    a. адаптивность — степень, в которой продукт или система эффективно адаптируются к разным или развивающимся аппаратным средствам, программному обеспечению или другим операционным средам, а также средам использования;

    b. простота и лёгкость установки — степень эффективности и продуктивности, с которой продукт или система успешно устанавливаются и (или) удаляются в конкретной среде;

    c. заменяемость — степень, в которой продукт может заменить другой указанный программный продукт для идентичной цели и среды.
    Пример: перенос данных из одной БД в другую без ущерба для работы приложения и без потери данных. Установка приложения на разные серверы или операционные системы.

7. Безопасность — степень, в которой продукт или система защищают информацию и данные таким образом, чтобы лица или другие системы имели степень доступа к данным, соответствующую их типам и уровням авторизации.

    a. конфиденциальность — степень, в которой продукт или система гарантируют, что данные получат только те, кому разрешён к ним доступ;

    b. целостность — степень, в которой система, продукт или компонент предотвращают несанкционированный доступ или модификацию компьютерных программ (данных);

    c. невозможность отказаться — степень, в которой действия или события могут быть доказаны, без возможности их скрыть или совершить подмену;

    d. ответственность — степень, в которой действия объекта однозначно прослеживаются, а также определяются ответственные лица;

    e. подлинность — степень, в которой личность субъекта или ресурса доказывается как заявленная.
    Пример: защищённость личных данных, которая гарантируется приложением.

8. Совместимость — степень, в которой продукт, система или компонент обмениваются информацией с другими продуктами, системами или компонентами и (или) выполняют требуемые функции, совместно используя ту же аппаратную или программную среду.

    a. сосуществование — степень, в которой продукт эффективно выполняет требуемые функции при совместном использовании общей среды и ресурсов с другими продуктами без вредного воздействия на любой другой продукт;

    b. совместимость — степень, в которой системы, продукты или компоненты обмениваются информацией и используют её.
    Пример: взаимодействие интернет-магазина с внешними платёжными системами.

Цели тестирования и роль тестировщика

Зачем тестировать ПО?

Процесс тестирования гарантирует, что ПО будет работать в соответствии с ожиданиями клиентов и на имеющемся у них оборудовании.

Выявление проблем на начальном этапе разработки уменьшает объём работ по исправлению ошибок на более поздних стадиях, обеспечивает правильное использование ресурсов и предотвращает повышение стоимости.

Команда тестирования привносит взгляд клиента в процесс и находит варианты использования, о которых разработчик может не подумать.

Любой сбой, дефект или ошибка, обнаруженные клиентом в готовом продукте, разрушают доверие к компании.

Цели тестирования ПО

1. Оценить на соответствие стандартам компании следующие рабочие продукты: требования, пользовательские истории, проектирование и код.
2. Проверить, все ли указанные требования выполнены.
3. Проверить, завершился ли объект тестирования и работает ли, как ожидают пользователи и заинтересованные лица.
4. Создать уверенности в уровне качества объекта тестирования.
5. Предотвратить дефекты.
6. Обнаружить отказы и дефекты.
7. Предоставить заинтересованным лицам информацию, позволяющую им принять обоснованные решения, особенно в отношении качества объекта тестирования.
8. Снизить уровень риска ненадлежащего качества программного обеспечения, например, пропущенные сбои в работе.
9. Обеспечить соблюдение договорных, правовых или нормативных требований, а также стандартов.

Такая постановка целей даёт возможность:

1. Повысить вероятность, что приложение, предназначенное для тестирования, будет работать правильно.
2. Повысить вероятность, что приложение, предназначенное для тестирования, будет соответствовать всем описанным требованиям.
3. Предоставить актуальную информацию о состоянии продукта в конкретный момент.

Обязанности инженера QA

Инженер QA — это специалист, который ставит цели тестирования, исполняет тесты и отчитывается о проделанной работе. В его обязанности также входит:

1. Мониторинг всего процесса разработки.

2. Отслеживание результатов каждого этапа SDLC и их корректировка в соответствии с ожиданиями.
3. Изучение документации.
4. Анализ и разработка требований к тестированию.
5. Разработка и выполнение тестов, приоритизация тестирования.
6. Фиксация проблем и инцидентов в соответствии с задачами проекта и планами управления инцидентами.
7. Работа с командой приложения и клиентом для решения любых проблем, возникающих в процессе тестирования.
8. Регрессионное и повторное тестирование.

SDET — Software Development Engineer in Test — инженер по разработке ПО в тестировании. Этот IT-специалист одинаково эффективно работает в сфере разработки и тестирования и участвует в полном процессе создания ПО. В отличие от инженеров QA, которым желательны базовые знания в программировании, но нет необходимости писать код, SDET это делает на постоянной основе, совмещая в себе тестировщика и разработчика.

Таблица сравнения SDET и тестировщика.

	SDET	Тестировщик
Насколько знает систему?	От начала до конца	Ограничены тестируемой функциональностью.
В каких этапах участвует?	Во всех этапах, от проектирования до внедрения.	Только в этапе тестирования.
Какие навыки нужны?	Умеет писать код и знает теорию тестирования. Участвует в автоматизации тестирования.	Знает только теорию и приёмы тестирования. Навыки автоматизации необязательны.
Какие обязанности?	Функциональное и нефункциональное тестирование, автоматизация, улучшение процессов.	В основном ручное тестирование. Возможно, автоматизация.

Валидация и верификация

Из определения тестирование отвечает на два вопроса:

1. Программа работает так, как описано в спецификации?

2. Программа удовлетворяет потребности пользователя?

Каждому из этих вопросов соответствует свой процесс.

Верификация — это процесс оценки системы или её компонентов с целью определения, удовлетворяют ли результаты текущего этапа разработки условиям, сформированным в начале. То есть выполняются ли наши цели, сроки, задачи по разработке проекта, определённые в начале текущей фазы.

Валидация — это определение соответствия разрабатываемого ПО ожиданиям и потребностям пользователя, требованиям к системе.

Иными словами, в процессе верификации тестировщик отвечает на вопрос 1, а в процессе валидации - на вопрос 2.

Принципы тестирования

Стандарт ISTQB определяет основные правила, которых придерживается тестировщик при планировании и исполнении тестов. Они оптимизируют количество и качество работ в области QA.

Принцип 1. Тестирование демонстрирует наличие дефектов

Тестирование показывает, что дефекты есть, но не может доказать, что их нет. Оно снижает вероятность наличия дефектов, находящихся в программном обеспечении, но, даже если дефекты не обнаружились, это не доказывает его корректности.

Принцип 2. Исчерпывающее тестирование недостижимо

Полное тестирование с использованием всех комбинаций вводов и предусловий физически невыполнимо, кроме тривиальных случаев. Вместо исчерпывающего тестирования, используется анализ рисков и расстановка приоритетов, чтобы более точно сфокусировать усилия по тестированию.

Принцип 3. Раннее тестирование

Чтобы найти дефекты как можно раньше, активности в жизненном цикле разработки программного обеспечения или системы по тестированию нужно начинать как можно раньше и фокусировать их на конкретных целях.

Принцип 4. Скопление дефектов

Усилия тестирования сосредотачиваются пропорционально ожидаемой, а позже реальной плотности дефектов по модулям. Как правило, большая часть дефектов, обнаруженных при тестировании или повлёкших за собой основное количество сбоев системы, содержится лишь в некоторых модулях.

Принцип 5. Парадокс пестицида

Если одни и те же тесты выполняются много раз, в конечном счёте этот набор тестовых сценариев больше не будет находить новых дефектов. Чтобы преодолеть этот «парадокс пестицида», тестовым сценариям требуется регулярно рецензироваться и корректироваться. При этом новые тесты должны быть разносторонними, чтобы охватить все компоненты программного обеспечения или системы, и найти как можно больше дефектов.

Принцип 6. Тестирование зависит от контекста

Тестирование выполняется по-разному в зависимости от контекста. Например, программное обеспечение, где критически важна безопасность, тестируется иначе, чем сайт электронной коммерции.

Принцип 7. Заблуждение об отсутствии ошибок

Обнаружение и исправление дефектов не помогут, если созданная система не подходит пользователю и не удовлетворяет его ожиданиям и потребностям.

Тестовая документация: тест-план, стратегия тестирования

Тест-план

Тест-план — это документ, в котором описывается объём работ, используемые подходы к тестированию, ресурсы, расписание, а также определяются функции и модули для тестирования, тестовые задачи, ответственные лица и требуемые метрики. В общем виде тест-план отвечает на вопрос: что и в какие сроки будем тестировать?

Кто создаёт: QA, руководитель команды тестирования.

Когда создаётся: на стадии планирования, дизайна, кодирования, либо на стадии тестирования.

Зачем создаётся:

1. Распределить зоны ответственности и задачи между командами, если в проекте задействовано несколько команд, например, команда ручного и автоматизированного тестирования.
2. Определить ресурсы: временные, человеческие, технологические.
3. Определить сроки тестирования.
4. Определить тестовое покрытие и метрики тестирования.

Роль тестировщика: создание и поддержка тест-плана, анализ полноты и точности, согласование с менеджерами и руководителями.

Разделы тест-плана:

1. Вступление — кратко описывается, что из себя представляет документ и для чего он предназначается. Краткое резюме тест-плана.
2. Функции, которые будут тестироваться. Описывается:
    - какие функции выполняет приложение;
    - какие из них должны протестироваться.
3. Функции, которые не будут тестироваться. Описываются функции, не требующие тестирования. Например, за тестирование отвечает другая команда, либо тестирование будет проводиться на следующем этапе или итерации.
4. Тестовые единицы — более детальное описание тестируемых функций (декомпозиция функций приложения), определяется объём тестирования.
5. Тестовые подходы и тестовые техники — описывается, с помощью каких подходов и инструментов будет проводиться тестирование: использование тестовых сценариев, исследовательское тестирование.
6. Критерии тестирования — описываются критерии начала, приостановки и окончания тестирования, а также приёмочные критерии и критерии оценки качества.
7. Ресурсы — описываются требуемые:
    - технические ресурсы — аппаратное обеспечение, ПО, количество лицензий;
    - человеческие ресурсы — количество специалистов и их уровень профессионализма;
    - временные ресурсы — сроки выполнения тестирования разных модулей или областей;
    - финансовые ресурсы.
8. Расписание — календарь с описанием основных этапов тестирования и обозначение контрольных точек.
9. Роли и ответственность — уровень компетенции тестировщиков и область ответственности: специалист по нагрузочному тестированию, senior-тестировщик, автотестировщик.
10. Оценка рисков — описываются основные риски и проблемы, которые могут возникнуть в процессе тестирования, с оценкой (в процентах, баллах) вероятности их возникновения и

степени ущерба: увольнение сотрудников, изменение требований, технические неисправности. Описываются также варианты работы с рисками при их возникновении.

11. Документация — описывается документация, которая будет использоваться в процессе выполнения тестирования, указываются лица, ответственные за её подготовку и согласование.

12. Метрики — числовые характеристики показателей качества. Они позволяют оценить прогресс тестирования, необходимость приостановки тестирования, качество тестируемой системы, достаточность тестирования. Метрики в тестировании мы подробнее рассмотрим в следующих лекциях.

13. Согласования — указываются лица, ответственные за принятие результатов тестирования, а также принимающие решение о переходе на следующий этап разработки или начале новой итерации.

Тест-план обычно ещё содержит:

- раздел со списком ссылок на документы, требуемые для проведения тестирования — требования, функциональные спецификации, инструкции;
- словарь специфических терминов, относящихся к области разработки или упоминаемых в тест-плане;
- идентификатор (номер) тест-плана в системе.

Наличие тест-плана, его форма и содержание зависят от особенностей проекта. Часто тест-план содержит не все перечисленные разделы, а только ту информацию, которая используется в работе и требуется команде или заказчикам.

Тестовая стратегия

Тестовая стратегия — это документ, который описывает подходы, правила и практики, применяемые на проекте, а также виды тестирования и тестовые данные. Последние используются в процессе тестирования. В общем виде тестовая стратегия отвечает на вопрос: как будем тестировать?

Тестовая стратегия — это необязательно объёмная официальная документация. Часто она представляется в виде статей на Wiki внутри компании или отдельных файлов.

Одни компании включают тестовую стратегию в тест-план (обычно небольшие проекты). Другие — имеют одну тестовую стратегию и несколько тест-планов для каждой фазы или уровня тестирования.

Кто создаёт: руководитель команды тестирования, тест-аналитик, опытный тестировщик.

Когда создаётся: на стадии планирования тестирования.

Зачем создаётся: определить основные направления тестирования, тестовые данные и тестовое окружение.

Роль тестировщика: создание и поддержка тестовой стратегии, согласование с менеджерами и руководителями проекта.

Создание тестовой стратегии обычно начинается с анализа продуктовых рисков.

Продуктовый риск — риск того, что разрабатываемая система или ПО не удовлетворит ожидания пользователей или заказчиков. Это также называется рисками качества.

Типы продуктовых рисков:

- многоуровневая архитектура системы, усложняющая тестирование и требующая проверки;
- новые плохо освоенные технологии в проекте;
- новая команда разработки с небольшим опытом проектной работы.

На основе рисков и особенностей проекта формируется список видов тестирования, они используются во время тестирования.

В тестовой стратегии также есть:

1. Тестовые данные — даты рождения, номера телефонов, виртуальные деньги. Эти данные используются в тестировании исходя из специфики проекта:
    a. Кто создаёт тестовые данные и как их вносят — заполняются вручную или генерируются автоматически?
    b. Можно ли использовать анонимизированные данные, которые скопированы из реальной базы данных клиентов?
    c. Какой объём тестовых данных требуется для тестирования — количество уникальных клиентских записей (100, 1000 и т. д.)?
2. Документация:
    a. Какая документация требуется и какова степень её детализации?
    b. Будут использоваться тест-кейсы или достаточно чек-листов?
    c. Какие техники тест-дизайна будут использоваться при создании тест-кейсов?
3. Фазы и виды тестирования:
    a. Какова взаимозависимость и последовательность выполнения видов тестирования: что тестировать в первую очередь — интерфейс или производительность?

    b. Какие дополнительные знания требуются для проведения заявленных видов тестирования? Например, отдельный специалист по тестированию безопасности.

4. Тестовое окружение: аппаратное и программное обеспечение проекта — серверы, компьютеры, операционные системы, браузеры, виртуальные машины, объём памяти.

    a. Какое тестовое окружение требуется создать? Нужно ли отдельно разворачивать тестовое окружение для каждого вида тестирования?

    b. Кто отвечает за поддержку тестового окружения? Какой конкретно специалист или несколько обновляют настройки и приводят тестовое окружение в актуальное состояние?

5. Инструменты для тестирования — программное обеспечение, требуемое для проведения тестирования в соответствии со спецификой проекта.

6. Наличие автоматизации тестирования, инструменты и языки программирования, используемые для написания автотестов.

Тестовая среда и тестовые данные

Есть несколько сред:

Среда разработки — в ней разработчики пишут код, проводят отладку, исправляют ошибки, выполняют Unit-тестирование. За эту среду отвечают также разработчики.

Среда тестирования — в этой среде работают тестировщики: тестируются новые билды, проверяется инструментарий, проводятся регрессионные проверки, воспроизводятся ошибки.

Интеграционная среда — иногда реализуется в рамках среды тестирования, а иногда в рамках превью среды. В этой среде есть необходимая для тестирования end-to-end схема взаимодействующих друг с другом модулей, систем, продуктов. Она требуется для интеграционного тестирования.

Превью среда — в идеале, это среда идентичная или максимально приближённая к продуктивной: те же данные, то же аппаратно-программное окружение, та же производительность. Она используется, чтобы сделать финальную проверку ПО в условиях максимально приближённым к «боевым». Здесь тестировщики проводят заключительное end-to-end-тестирование, приёмку, демонстрируют работу продукта представителям бизнеса.

Продакшн-среда — среда, где работают пользователи. Тестирование на ней проводится в крайне редких случаях.

Оптимальное количество сред подбирается под конкретную компанию или проект. Если проект небольшой, то среда тестирования объединяет в себе интеграционную и превью-среду.

В общем случае среда тестирования — это настройка программного и аппаратного обеспечения для тестирования.

Настройка правильной среды тестирования гарантирует успех тестирования ПО. Любые недостатки в этом процессе обычно приводят к дополнительным затратам и времени для клиента.

Тестовые данные — это набор входных значений, требуемых для выполнения тестов. Тестировщики определяют данные в соответствии с требованиями. Они делают это вручную или использовать инструменты генерации.

Инструменты генерации — это программы, которые быстро создают тестовые данные. Их использование целесообразно в случаях:

- нужно много различных однотипных данных, например, паспортные данные;
- ручное создание занимает много времени — в случае тестирования сложных систем;
- данные нужны для автотестов.

Инструментами генерации:

- таблицы Excel;
- онлайн-утилиты, например, Pairwise Pict Online;
- скрипты для выполнения в командной строке;
- сложные программы, созданные самими тестировщиками.

Testing, QC, QA

Обеспечение качества (Quality Assurance — QA) — это совокупность мероприятий, охватывающих все технологические этапы разработки, выпуска и эксплуатации ПО. Это также совокупность информационных систем, предпринимаемых на разных стадиях жизненного цикла ПО, чтобы обеспечить требуемый уровень качества выпускаемого продукта.

Деятельность QA происходит до, во время и после написания программного кода. В неё входит:

- детализация требований;
- анализ и расчёт времени на тестирование, ретесты, исправление багов;
- разработка сценариев тестирования;
- анализ результатов тестирований;
- оптимизация процессов разработки для избежания повторного появления обнаруженных ошибок;

- разработка стандартов;
- включение тестирования в работу на самых ранних этапах;
- разработка стратегий автоматизированного тестирования.

Контроль качества (Quality Control — QC) — это подмножество QA, совокупность действий, проводимых над продуктом в процессе разработки для получения информации о его актуальном состоянии и соответствии ожидаемым результатам.

Деятельность QC происходит после написания кода. В неё входит:

- ручное исполнение сценариев тестирования;
- локализация и описание дефектов;
- ретест дефектов;
- приёмочное и регрессионное тестирование;
- исследовательское тестирование;
- написание автотестов.

Тестирование — это процесс, направленный прямо на составление и прохождение тест-кейсов, локализацию дефектов. Это всегда часть контроля качества.

Таблица сопоставления QA, QC и тестирования.

	Обеспечение качества (QA)	**Контроль качества (QC)**	**Тестирование**
На каком этапе?	Все технологические аспекты на всех этапах разработки ПО от планирования до сопровождения.	На этапах разработки и тестирования.	Только на этапе тестирования.
На чём фокус?	Процессы и средства разработки.	Исполнение тестов.	Исполнение тестов.
Какой подход?	Процессно-ориентированный.	Продуктно-ориентированный.	Продуктно-ориентированный.
Какая цель?	Превентивные меры: не допустить ошибку.	Корректирующий процесс: найти ошибку.	Корректирующий процесс: найти ошибку.

Какие «мягкие» навыки требуются тестировщику

Умение задавать вопросы

Тестировщик обязан задавать вопросы, чтобы получить исчерпывающую информацию о том, как работает продукт:

- уточнить непонятные термины в документах;
- прояснить неточности в описании логики работы системы;
- уточнить, фактическое поведение — это баг или фича;
- выяснить, с кем конкретно можно решать вопросы;
- выяснить, кто ответственен за решение проблемы, и какая информация нужна для решения.

Требуется задавать точные вопросы. «Я ничего не понял, объясните», — это плохой вопрос, на него никто не будет тратить время. «В документах указывается, что пользователь может зарегистрироваться по телефону, а у нас на странице только email. Это баг?», — это правильный вопрос тестировщика.

Умение чётко и понятно излагать свои мысли устно и письменно

1. Тестировщики создают тестовую документацию: тест-кейсы, чек-листы, отчёты о дефектах. Чем проще и понятнее они составлены, тем меньше вероятность, что дефекты останутся незамеченными. Важно помнить:
   Никто не любит длинные и запутанные предложения. Всё, что написано непонятно, проигнорируется.
2. Любая неточность — почва для бага. Дефекты рождаются из непонимания требований или других документов.
3. С вопросами «как это работает» в первую очередь идут к тестировщику. Важно уметь «на пальцах» объяснить, как функционирует продукт.

Любопытство

Это базовое свойство тестировщика. Тестировщик, не обладающий любопытством, не протестирует ни одну систему. В лучшем случае он хорошо выполнит тест-кейсы и заведёт дефекты. Любопытный тестировщик не только формально пройдёт тест-кейсы, но и проведёт исследовательское тестирование, дополнит имеющиеся проверки новыми идеями.

Дисциплинированность

Тест-кейсы обязательно пишутся в срок. Дефекты оформляются сразу после обнаружения, они перепроверяются сразу, как только соответствующий патч попал на стенд тестирования и т. д.

Если тестировщик не дисциплинирован, не может сосредоточиться на задаче и не обладает структурным мышлением, то из его рук не выйдет качественный продукт.

Стремление к знаниям

Мир IT развивается стремительно: появляются новые инструменты, технологии, молодые специалисты наступают на пятки. Если тестировщик не осваивает новые навыки, не следит за трендами развития в отрасли, то очень скоро он может оказаться за бортом.

Карьерные пути развития тестировщика

Развитие «вверх»

Развиваясь «вверх», тестировщик проходит путь от junior-специалиста, который занимается только ручным тестированием, до Senior QA Engineer или Lead Software Testing Specialist. Расширяется круг обязанностей: добавляется автоматизация тестирования, управление процессами, подбор сотрудников в команду, анализ рисков.

Развитие «в сторону»

Развиваясь горизонтально, тестировщик осваивает новые области тестирования. Например, к функциональному тестированию добавляется нагрузочное и тестирование безопасности. При этом специалист не занимает позицию лида, не руководит людьми и процессами. В основном это развитие технических навыков.

Развитие в смежных областях

Нередко тестировщики становятся разработчиками, аналитиками, владельцами продуктов, т. к. участвуют в различных процессах разработки ПО. Многие ошибочно рассматривают тестирование как лёгкий старт в IT с перспективой уйти в разработку. Разработка и тестирование ПО — это разные процессы, поэтому если душа лежит больше к разработке, то лучше сразу учиться на разработчика.

Контрольные вопросы

1. Зачем тестировать ПО?
2. Каковы критерии качества ПО?
3. Каковы обязанности инженера QA?

4. Каковы принципы тестирования?
5. Чем отличаются QA, QC и тестировщики?
6. В чём различие между тест-планом и тестовой стратегией?
7. Какие среды используются в процессе разработки и тестирования ПО?

Домашнее задание

1. Создайте копию документа и ответьте на поставленные вопросы. В качестве результата отправьте ссылку на Google-документ.
2. Тест для самопроверки https://coreapp.ai/app/player/lesson/6143431440a70c676714c749 (сдавать не нужно)

Глоссарий

Тестирование программного обеспечения — это проверка соответствия между реальным и ожидаемым поведением программы, а также выявление, насколько ПО удовлетворяет потребности пользователя. Оно осуществляется на конечном наборе тестов, который составляет тестировщик.

Качество программного обеспечения — это совокупность характеристик, относящихся к его способности удовлетворять установленные и предполагаемые потребности.

Качество — это степень, в которой какой-то компонент, система или процесс отвечает конкретным требованиям и (или) требованиям и ожиданиям пользователя.

Верификация — это процесс оценки системы или её компонентов с целью определения, удовлетворяют ли результаты текущего этапа разработки условиям, сформулированным в начале этого этапа. То есть выполняются ли наши цели, сроки, задачи по разработке проекта, определённые в начале текущей фазы.

Валидация — это определение соответствия разрабатываемого ПО ожиданиям и потребностям пользователя, требованиям к системе.

Обеспечение качества (Quality Assurance — QA) — это совокупность мероприятий, охватывающих все технологические этапы разработки, выпуска и эксплуатации ПО. Это также совокупность информационных систем, предпринимаемых на разных стадиях жизненного цикла ПО, чтобы обеспечить требуемый уровнь качества выпускаемого продукта.

Контроль качества (Quality Control — QC) — это подмножество QA, совокупность действий, проводимых над продуктом в процессе разработки, чтобы получить информацию о его актуальном состоянии и соответствии ожидаемым результатам.

Тестирование — это процесс, направленный прямо на составление и прохождение тест-кейсов, локализацию дефектов. Это всегда часть контроля качества.

Тест-план — это документ, в котором описывается объём работ, используемые подходы к тестированию, ресурсы, расписание, а также определяются функции и модули для тестирования, тестовые задачи, ответственные лица и требуемые метрики. Тест-план отвечает на вопрос: что и в какие сроки будем тестировать?

Тестовая стратегия — это документ, который описывает подходы, правила и практики, применяемые на проекте. Описываются также виды тестирования и тестовые данные, используемые в процессе тестирования. Тестовая стратегия отвечает на вопрос: как будем тестировать?

Тестовые данные — это набор входных значений, требуемых для выполнения тестов. Тестировщики определяют данные в соответствии с требованиями. Они делают это вручную или используют инструменты генерации.

Среда тестирования — в этой среде работают тестировщики: тестируются новые билды, проверяется инструментарий, проводятся регрессионные проверки, воспроизводятся ошибки.

Дополнительные материалы

1. Статья «Образ современного тестировщика. Что нужно знать и уметь».
2. Статья «Как составить стратегию тестирования: версия настоящих инженеров».
3. Статья «Семь Принципов тестирования».
4. Статья «Тест-план на одну страницу».
5. Статья «Путь развития тестировщика: как найти компанию по душе».
6. Статья «Комикс про тигра-тестировщика: жизнь, смех и слёзы».
7. Статья «Тестирование — это не поиск ошибок».
8. Статья «Soft-skills успешного тестировщика».