Тестировщик — Lurkmore

НЕНАВИСТЬ!

Данный текст содержит зашкаливающее количество НЕНАВИСТИ.
Мы настоятельно рекомендуем убрать от мониторов людей, животных со слабой психикой, кормящих женщин и детей.

«Внезапно, Ахтунг! Капитан как бы говорит нам: расовое небыдло в Этой стране с ФГМ и ЧСВ over 9000, алсо ПГМ и хуитой — на самом деле чуть менее, чем наполовину люто, бешено доставляющие быдло и школота, но всем похуй чуть менее, чем полностью. Инфа 100%. Я гарантирую это!»

Уровень луркоёбства в этой статье превышает все допустимые пределы. Следует почистить статью от ~~излишнего~~ lm-speak'а и излишних ссылок на наиболее популярные статьи, а автору — прописать лечебную клизму кипятком.

Ваша статья — говно, вы ничего не понимаете в «тестировании».

Если вы видите это предупреждение, значит данная статья уныла чуть менее, чем полностью, и в неё нужно добавить интересных фактов. Кроме того, возможно, что эта статья вообще никому на йух не сдалась тут. В таком случае единственной рекомендацией будет перенос статьи в и освежение её бодрящей порцией лулзов.

«После примерно пятой неудачной попытки объяснить родственникам и знакомым, кем я работаю, я говорю, что работаю программистом. »

— Откровения тестировщика

«Тестировщик заходит в бар и заказывает:

кружку пива, 2 кружки пива, 0 кружек пива, 999999999 кружек пива, ящерицу в стакане, –1 кружку пива, qwertyuip кружек пива.

Первый реальный клиент заходит в бар и спрашивает, где туалет. Бар вспыхивает пламенем и сгорает.

»

— Самый известный анекдот про сабж

Тестер (кьюэй, тестер, кликер, monkey qa, quality assurance, quality control) — человек, копающийся в кучах кода, отложенных программерами. Делятся на кликеров, технических тестеров, QA инженеров, QA лидов и QA манагеров. Первые кликают по готовой программе, без наличия какой-нибудь документации или спецификации и наугад ищут ошибки, при этом грубо троллят быдлокодеров. Технический тестер занимается тестированием исходников кода. QA инженеры пишут тест-планы, сценарии тестирования и прочую хуиту, призванную сделать процесс тестирования чуть более чем однозначным для всех вовлеченных участников. QA lead наблюдает за процессом тестирования в тиме. QA манагер делает практически то же самое, что и QA lead, но в пределах компании ещё интервьюирует кандидатов на должность быдлотестера.

Тестировщик в пределах СНГ

На просторах USSR профессии тестировщика как таковой нет. Тем

не менее, система управления качеством в СССР существовала и была весьма эффективной. Каждое предприятие имело ОТК, подчиняющийся только высшему руководству организации (а позже вообще не подчиняющийся), за счёт чего исключалось давление на проверяющих. Таким образом ОТК можно считать аналогом отдела тестирования в IT-отрасли с некоторыми оговорками. А после ОТК, в начале perestroika, появилась еще и Госприёмка, как средство контроля за работой ОТК.


Тоже тестер

В последнее время, когда IT-аутсорс набрал оборотов, народ прочел на новостных сайтах о профессии тестировщика — сиди себе и тыкай в сайт с умным видом и получай за это овер 9000 зарплаты. Естественно, все эти люди начали обивать пороги IT-компаний, и кастинг на вакансию QA интерна напоминает кастинг в модельное агентство или талант-шоу. ЧСХ, кандидаты с 5-м размером, которые удосужились прочесть книгу о тестировании ПО, в итоге находят работу, остальные — отправляются создавать жалобные топики на форумах, в которых генерят кучу лулзов, баттхерта и вкусной еды и выпивки.

Основные пункты автобиографии поца:

1. Возраст — около 25-30 лет, с отклонениями в ту или другую сторону.

Красивый и уверенный в себе тестировщик тычет программиста мордой в монитор

2. Безработный, менеджер самого среднего звена в далекой от IT отрасли.

3. В школе любил Delphi и Visual Basic, ЧСХ только сейчас озаботился построением карьеры манки-кликера.

4. Знает, как включить компутер, готов за счет работодателя изучить тонкости профессии. Так же готов (естессно, за счет работодателя) изучить программирование на любом языке.

5. Обладает вырвиглазным резюме на русском или ломаном английском языке.

6. Зарплатные ожидания — овер 9000.

7. Задается вопросом, почему рекрутеры игнорируют его письма. Некоторые особо упоротые, спустя месяцы и годы, так и не находят работу. И создают еще один топик, чем несказанно радуют местных советчиков.

Из-за наплыва этих господ расовый фашысткий форум разработчиков don даже вводил премодерацию топиков.

Откуда они?

Каждая уважающая себя ИТ-контора должна иметь независимый отдел тестирования. Исторически сложилось, что быдлокодер не замечает за собой абсолютно никаких ошибок и считает свой код абсолютно совершенным, но когда его продукт попадает к кастомерам, или хуже — к конечным юзерам, тогда он начинает огребать неиллюзорных пиздюлей. Тогда придумали ход конём: нанять независимого юзернейма, который бы смотрел на шедевр кодера как на говно и искал уязвимые места в его творении. Система дала результат — качество продукта улучшилось, кастомер признал эффективность, а анонимус сидел себе и кайфовал.

Но быдлокодеры невзлюбили этих выскочек, которые не имели достаточно знаний и авторитета, чтобы критиковать «совершенный продукт» первых и после этого появился новый мем ака холивар: **«Это не баг — это фича!!!»** До недавнего времени профессия считалась непрестижной и малооплачиваемой. Но сейчас хороший, опытный тестер получает ничуть не меньше, а иногда даже больше, чем быдлокодер. А в этой стране, тестер будет получать столько, сколько назначит директор, который зачастую сам является быдлокодером и из-за вышесказанной нелюбви имеет сотрудника, как только хочет.

Виды тестировщиков

Мануальный (clicker, ручной, быдлотестер или в просторечии софто-дрочер) — распиздяй с нестандартным типом мышления, который на всё смотрит с точки зрения «нихуя не работает». Среди кликеров чуть более, чем половина — самки, потому что быдлокодеры всегда ищут 5-й размер пары сисек себе в тим. Когда на сайте висит открытая вакансия на быдлотестера, это кагбэ намекает на обязательное наличие сисек у кандидата.

Тестеру лучше ничего не давать — сломает, сука! Багов находит много. Не обделённый, креативный тестер перегружает сервер, вплоть до поломки и ремонта на весь день (если это был общий сервер,

которому положено нести всю ношу вычислений на себе, или это был роутер, то встанет вся работа. Если это были дорогие, связанные серверы солидного отдела, который, поставленные по компоновке «связь всего со всем», которую даже специально и с сетью ботов может не получиться обвалить, то все остановится на том, что будет использована вся их мощность, часть станет падать, но прежде чем тестировщик валит один сервер, программист поднимает другой), после чего уходит пить водку и гундосить. На работу всегда опаздывает и идёт домой первым. Чаще всего уходит, свалив систему в машинный Ад (или в форсажные режим, если не упала), обспечив тем самым незабываемый секс быдлокодерам на вечер. Всегда ищет критические ошибки, чтобы система упала, и тогда сидит в любимом ЖЖ, быдлоконтакте, быдлокласниках или YouTube. Поиск критических ошибок облегчают быдлокодеры, сидящие на ютубе, lurkmore и т. п. сайтах.

Automation tester — неудавшийся быдлокодер или продвинутый кликер со знанием скриптовых языков программирования или web application testing system типа Selenium. В последнее время стоит сказать что Selenium является объектом фапания и подрожания автоматизированных тестровщиков, в связи с чем рождаются холивары на тему какой инструмент круче, Selenium или (подставьте нужное). Очень часто в автоматизированные тестировщики идут выпускники программистских специальностей за 5 лет освоившие программы типа HelloWorld но оказавшиеся слишком тупы чтобы быть в состоянии написать что-либо более сложное. Характеризуется тем что часто обитает в специализированныйх форумах посвящённых использованию какого-либо тула и создаёт темы с тупейшими вопросами по азам программирования. Может поддаваться троллингу вопросом почему он выбрал автоматизированное тестирование, а не полноценное программирование. Как правило троллинг толст, так как этим обычно занимается сам быдлокодер. Так же автотестер может быть высмеян и Ъ-тестерами, которые любят и умеют тестировать не только скриптами. В большинстве задрот или ОП, покрывающий существующий функциональный тест кейсами, чтоб кликер не парился и не делал свою рутинную работу по стопицот раз. Багов находит мало, потому что тесты рутинные и регрессионные. Систему тоже валит очень редко. Особенно эффектно этот метод катит при разработке цифровых схем — таким образом сейчас разрабатываются и тестируются процессоры и тому подобные схемы, процесс разработки которых лишь немного отличается от процесса разработки программ благодаря VHDL, Verilog и другим кошерным вещам. Алсо, на крупных проектах автоматизатор несколько итераций активно деплоит тест свиты (кто сказал сьют?), а после пинает хуец между релизами. Из-за большого разнообразия тестовых тулов успевает за год переюзать пару десятков платформ. Ввиду такого разнообразия уровень понимания данных платформ у него оказывается на уровне блондинки с ресепшена. Так что по скиллам нередко уделывает распальцованных девелоперов (по скиллам распальцовки по большей части), ибо знает хуеву тучу технологий (на уровне жонглирования малопонятными ему самому терминами), а не задрачивает одну и ту же изо дня в день.

Юнит тестировщик — такой же программист, как и все остальные, но целью которого является максимальное покрытие кода тестами, которые проверяют все возможные и невозможные ситуации. Обеспечивает безопасность кода, чтобы никакой эксплоит… То есть он знает код даже лучше, чем программисты, которые его пишут, и пишет код, который тестирует код. Очень редкий вид, потому, что чаще всего заказчики не готовы платить за такую работу, да и практически никогда не требуется от программ такого качества. Существуют в компаниях типа Sun, Intel, Nokia и т. п. Хотя есть альтернативная версия. Дело в том, что программисты тоже бывают разные, одним нужно платить больше, другим можно меньше. Написание тестов для существующего кода считается низко интеллектуальной работой, потому что нужно просто разобраться как оно работает, и написать тесты которые проверяют как оно работает. Ситуация когда технический тестировщик знает код лучше автора возникает тогда, когда на одного автора кода — белого человека, приходится 100 технических тестировщиков—негров, и каждый негр задрачивает свой определённый маленький кусочек от общей системы. Белый человек писавший этот код 10 лет назад уже и забыл про него, а негр-технический тестировщик до сих пор дрочит свои тесты на именно этот объём кода, и соотв. создаётся впечатление что он умнее автора.

QA leader — небыдло, которое переросло уже кликера или технического тестера и имеет достаточно экспириенса, чтобы рулить командой. Отвечает за конечное качество выпускаемого продукта. Всё время занят решением проблем с кастомерами, планированием графика и ресурсов тестирования, созданием тест-планов и тестовых сценариев. Если конечный пользователь нашёл баг, пропущенный отделом тестирования, тогда QA lead получает эпичных пиздюлей, ибо «во всём виноват отдел тестирования», то есть, ситуация один-в-один, как, скажем, в футболе — если игра закончилась со счётом 100500:0, то виноват один вратарь. Непосредственным тестированием занимается редко и, как правило, без особого успеха, поскольку за утрясанием планов, перепиской с заказчиками и составлением гор бумаг (чаще всего не нужных ни заказчику, ни исполнителю) уже нафиг ничего не знает в деталях. Контролирует результаты работы кликеров и технических QA, а также выступает буфером в холиварах между быдлокодерами и тестерами (то есть получает пиздюли с обеих сторон). Прокачанная версия обладает всем бумажкам подтирашку — сертификат ISTQB. Что в переводе на общеофисный означает «ЧСВ имеет законные основания».

QA manager — высший элемент пищевой цепочки тестеров. Занимается сертифицированием процессов тестирования согласно стандарту IEEE 829 по всей ИТ-конторе. Проверяет кандидатов на наличие сисек и отсутствие ФГМ. Страдает завышенным ЧСВ. Как ни странно, но среди подобной братии нередко встречаются личности, которые обладают скиллом утрясать сложные проблемы и конфликты. Рядовые кликеры успевают отделаться легким испугом, когда программоваятели, аки орки пришедшие проводить геноцид, довольные уходят из зоны поражения с долгом в два релиза и пачкой печенья.

Главные враги

- **Проект-менеджеры** и **программисты** заказчика, которые не хотят фиксить баги, которые вы считаете важными. В лучшем случае они кладут баги в бэклог (по-русски — в долгий ящик), в худшем переведут баг в статус Waived (отклонено) со словами «это норма».
- **QA тимлиды**, которые не дают добро на постинг багов, которые вы считаете важными (здесь шрифт Arial, а по макету должен быть Arial Narrow!), ИСЧХ, по-своему они правы — по плану надо пройти ещё 100500 кейсов, и если из-за постинга багов по всякой мелочи не успеем их пройти и найти/запостить что-то реально серьёзное, а потом эта проблема вылезет у конечного пользователя, то весь гнев заказчика обрушится на них. А от них уже и на вас. Также тимлиды, через которых и происходит общение команды тестирования с заказчиком, уже прекрасно знают характер и предпочтения заказчика и, соответственно, могут знать, что например некритичные баги по старому функционалу заказчик не будет фиксить ВООБЩЕ, а поэтому вы как тестировщик можете сколько угодно радоваться и тешить своё ЧСВ, что нашли какой-то малозаметный баг, которому уже 100500 лет — всем пофиг.
- Ну и разумеется, главный враг тестировщика — **время**. Ибо невозможно найти вообще все баги, иначе всё отведённое для тестов время пришлось бы копаться в одной фиче и даже не знать, что другие фичи не работают вообще. Поэтому одним из наиболее ценных качеств тестировщика является умение расставлять приоритеты — «важно»/«неважно». Грубо говоря, за день до релиза постить некрасиво расположенные буквы всё же не стоит, важнее быть уверенным, что приложение не крешнет и не зависнет в самый неожиданный для юзера момент, положив ему при этом всю систему. Кому интересно больше — можете почитать у Тёмы про метод прогрессивного джипега.

Требования

В разных компаниях аутсорс тестирования требования на вакансию «тестировщик» разные. В целом, тестировщиком может стать кто угодно, ибо с сайтами, приложениями и играми на компьютерах и смартфонах сейчас так или иначе знакомы все, а писать кейсы и нажимать на все подряд кнопки в надежде сломать приложение научить можно в принципе тоже кого угодно. Поэтому часть компаний выставляет единственное требование будущим тестировщикам «уверенное владение компьютером» (мол, остальному сами научим, всё равно на тестировщика нигде специально не учат). Другая часть компаний, наоборот, на собеседовании может задавать теоретические вопросы (например, что такое тестирование чёрного ящика). Проверяется этим обычно не степень озадроченности тестировщика, а то, гуглил ли он вообще, чем будет заниматься, или просто пришёл «на дурака». Поэтому совет тем, кто устраивается на работу тестировщиком: если вы не компьютерный задрот и не знаете, что это и с чем его едят, прочитайте перед собеседованием хотя бы книжку Романа Савина «Тестирование Дот Ком, или Пособие по жестокому обращению с багами в интернет-стартапах», чтобы хотя понимать, куда и на что идёте. Или хотя бы прочитать приведённый ниже словарь тестировщика.

Словарь тестировщика

Работа с багами

Стилизация под шансон

- **Баг** (англ. *bug*) — основной объект охоты. См. отдельную статью.
- **Постинг** (англ. *posting*) — занесение бага в трекер (систему отслеживания багов и их статусов). В баге указывается заголовок (коротко о чём баг), описание проблемы, шаги для воспроизведения по порядку, фактический результат (как есть) и ожидаемый результат (как должно быть). Аттачи в виде скриншотов и видео приветствуются. У тестировщиков есть неписаное правило: кто нашёл баг, тот его и постит.
- **Фикс** (англ. *fix*) — исправление бага или сам исправленный баг. Глагол — *фиксить*, причастие прошедшего времени — *пофикшенный*.
- **Регресс** или **верификация**, (англ. *verification*) — перепроверка пофикшенных багов, по результатам которой их либо закрывают (исправлено), либо переоткрывают (не исправлено, надо фиксить снова).
- **Кейсы** (англ. *case* — случай, ситуация)) — тестовые ситуации, которые должен проверить тестировщик. Состоят из **шагов** и **ожидаемого результата**. Например, шаги: нажать на иконку приложения, ожидаемый результат: приложение запускается. Если приложение не запускается — это баг.
- **Чеклист** (англ. *checklist*) — сборник кейсов, которые тестировщик должен проверить. Может быть в виде простой Excel таблицы или же засунут в специальную программу, например Testrail. У каждого пройденного кейса тестировщик должен проставить результат прохождения: **Pass**, если ожидаемый результат кейса выполняется, или **Fail**, если он не выполняется. Отсюда выражения «пассить кейс» и «фейлить кейс». У каждого зафейленного кейса в комментарии должна стоять ссылка на баг.

- **Реджект** (англ. *reject*) или **вейв** (англ. *waive*) – отклонение бага заказчиком с формулировкой «не

баг» или «это норма» (англ. *not an issue*). В некоторых трекерах также встречается статус Won't Fix, т.е. заказчик признаёт, что баг имеет место, но не хочет тратить время и усилия на фикс такой мелочи (или боится, что фикс породит ещё больше проблем).

Технические термины

- **Билд** (англ. *build*) — собранное приложение. В зависимости от целевой аудитории билд может быть тестовый, разработчицкий (dev), релиз кандидат (вот-вот выпустят) или **прод** (опубликованный в магазине для скачивания).
- **Клиент** (англ. *client*) – собственно само приложение. Актуально для онлайн-приложений/игр, поскольку часть функционала в них хранится на клиенте (в самом приложении), часть – на сервере. Небыдло-тестировщики иногда говорят «перезапустить/переустановить/почистить клиент» вместо «приложение».
- **Версия** (англ. *version*) — версия приложения, обычно меняется при обновлении приложения в проде. Например, была версии 2.10, стала 2.11.
- **Ревизия** (англ. *revision*) — номер каждой сборки билда. В отличие от версии, меняется каждую сборку. Т.е. тестеры могут тестить одну и ту же будущую версию 2.11, но при каждой сборке будут получать билды разных ревизий: была ревизия 1000, стала 1001.
- **Репозиторий** (англ. *repository*) — это тестировщику знать не обязательно, но всё же скажем: репозиторий — это место в интернете, где хранятся файлы приложения. Для тех, кто не в курсе: приложение — это по сути исполняемый архив, состоящий из тех самых файлов. При фиксе бага программист правит один из таких файлов и заливает его обратно в репозиторий. Потом хранимые в репозитории файлы собираются в билд.
- **Итерация** (англ. *iteration*) — цикл тестирования. При тестировании обновления приложения обычно одна итерация — одно обновление. При тестировании приложения в стадии разработки итераций может быть несколько: сначала проверяем самую сырую версию приложения, потом доработанную и т.п. (для подробностей курим Cascade vs. Agile тестирование).
- **Тест-дизайн** (англ. *test design*) — стратегия покрытия приложения кейсами.

Рабочие термины

- **Креш** (англ. *crash*)[1] — падение приложения, т.е. его самопроизвольное закрытие. Считается самым серьёзным багом, о котором нужно сразу сообщать разработчикам через руководителя команды. При постинге крешей нужно прикладывать логи (см. ниже).
- **Фриз** (англ. *freeze*) — зависание приложения, когда экран перестаёт реагировать на нажатия и невозможно совершить какие-либо действия в приложении. Тоже одна из самых серьёзных проблем, и тоже при постинге надо прикладывать логи.
- **Софтлок** (англ. *softlock*) — менее распространённая ситуация, когда интерфейс приложения работает, но пользователь не может совершить каких-либо продуктивных действий. Например, при нажатии элемента интерфейса юзер попадает на страницу, в которой отсутствует кнопка выхода и вообще какие-либо элементы управления, т.е. ничего на странице не сделать и к другим страницам сайта не вернуться. В играх софтлоком может быть ситуация, когда от игрока требуется совершить какое-то действие, а совершить его невозможно (например, недоступна соответствующая кнопка).
- **Лог** (англ. *log*) — текстовый файл, в который записываются действия, производимые на устройстве или в приложении. В особо сложных случаях их просят у тестеров разработчики, чтобы понять причину воспроизведения бага. Логи практически всегда нужны при креше (падении) или фризе (зависания) приложения. В iOS есть встроенная запись крешлогов, которые можно скинуть с девайса на комп через iTunes, в Android такого нет и логи надо снимать вручную через adb или logcat. Легче всего снимать логи в браузере компьютера — так, в Chrome для этого достаточно нажать F12 и правым кликом мыши сохранить текст в окошке справа.

Виды тестов

- **Смоук-тест** или **смок-тест** (англ. *smoke testing* — дымовое тестирование) — поверхностное тестирование с целью убедиться, что программа запускается и все основные фичи **в целом** работают. Важно: смок-тесты предполагают **только проверки с валидными значениями**, т.е. со значениями, с которыми и предназначена работать программа. Т.е. смок-тест калькулятора - это проверка, что он запускается и проводит действия с числовыми значениями, ввод невалидных значений (например, букв) в рамках смок-тестов не проверяется. Название получило потому, что изначально так тестили работу радиоэлектронных деталей: идёт из них дым или нет. IRL пример: вы покупаете фотоаппарат, консультант достаёт его, включает, щёлкает затвором — работает. Это и есть дымовое тестирование.
- **Фулл-тест** (англ. *full test*) — подробное тестирование, в котором проверяется каждая мелочь. Обычно происходит перед релизом приложения или апдейта.
- **Тестирование прерываний** (англ. *interruption testing*) — тестирование с целью убедиться, что приложение не ломается, если каким-то образом прервать его работу: свернуть, закрыть, перезапустить, заблокировать экран, для сайтов и онлайн приложений — выключить интернет, для мобильных приложений — совершить входящий звонок или смс и т.п.
- **Негативное тестирование** (англ. *negative testing*) — проверка, что ничего не ломается, если юзер попытается сделать то, что не предусмотрено приложением. Например, ввести буквы в поле ввода

номера телефона.
- **A/B тесты** (англ. *A/B tests*) — иногда разрабы решают протестировать какую-то фичу на самих юзерах, чтобы решить, стоит ли её вводить. В этом случае при скачивании приложения юзер получает один из нескольких возможных конфигов. Например, у одних юзеров текст "Приветствую" будет белым, у других красным, или в игре у одних юзеров герой будет иметь 500 золотых монет на старте, а у других 1000. В этом случае тестировщик должен убедиться, что все конфиги выпадают юзерам с заданной вероятностью и нет такого, что какие-то конфиги вообще не выпадают.
- **Тестирование нагрузки** (англ. *load testing*) — актуально для сайтов и онлайн-сервисов. Тестеры наваливаются всем народом на сайт, создавая максимально возможное число запросов к нему, и смотрят, не упадёт ли сайт от такого большого количества одновременных запросов.

Виды тестов (для небыдла)

Подробнее можно прочитать здесь.

По объекту тестирования

- **Функциональное тестирование** (англ. *functional testing*) — то, что обычно пишут работодатели в вакансиях тестировщика в графе «Обязанности». По сути, это собственно тестирование, т.е. проверка пригодности приложения к выполнению функций, для которых оно предназначено (все фичи работают, как задумано).

- **Регрессионное тестирование** (англ. *regression testing*) — проверка того, что добавленный в приложение новый функционал или пофикшенные проблемы не поломали ничего другого в приложении. Не путать с **регрессом** (см. выше) — проверкой пофикшенных багов. Регрессионный тест выполняется после смок-теста.

- **Визуальное тестирование** (англ. *visual testing*) — тестирование вёрстки и UI (пользовательского интерфейса) на соответствие макетам и здравому смыслу (например, текст ни на каком языке не вылезает за пределы поп-апа).

- **Исследовательское тестирование** (англ. *exploratory testing*) — тестирование параллельно с разработкой функционала, когда ещё нет ни чеклистов, ни каких-то других сценариев тестирования конкретного продукта.

- **Интеграционное тестирование** (англ. *integration testing*) — тестирование взаимодействия программных модулей. Как правило, программный продукт состоит из нескольких программных модулей, написанных разными программистами. Целью является выявление багов при взаимодействии между этими программными модулями и в первую очередь направлен на проверку обмена данными между этими самими модулями (особенно у клиент-серверных приложений).

По доступности исходного кода

- **Тестирование чёрного ящика** (англ. *black-box testing*) — наиболее распространённый вид тестирования, при который тестировщик видит приложение только с точки зрения юзера, не имея доступа к исходному коду и понятия, как оно всё работает изнутри.

- Соответственно, **тестирование белого ящика** (англ. *white-box testing*) — это тестирование не только самого приложения, но и его кода. Например, каждая ли строка исходного кода была выполнена и протестирована, каждая ли точка решения (вычисления истинно ли или ложно выражение) была выполнена и протестирована и т.п. Для такого тестирования недостаточно просто уметь нажимать на кнопки, нужно ещё знать программирование.

- Промежуточный вариант – **тестирование серого ящика** (англ. *grey-box testing*): тестируем с точки зрения юзера, а не программиста (как в чёрном ящике), но доступ к коду у нас всё же есть, чтобы в случае чего посмотреть, что там написано и почему так работает.

Инструменты

Если вы собираетесь тестировать не только игрушки или интерфейс сайтов, вам скорее всего придётся освоить следующие понятия и инструменты:

- **Взаимодействие клиент-сервер**. Если для работы приложения нужен постоянный выход в интернет или если мы просто имеет дело с любой страницей в браузере, перед нами взаимодействие клиент-сервер. В этом случае для почти всех действий клиент обращается к серверу за данными (отправляет запросы), сервер в свою очередь обрабатывает эти запросы и шлёт ответы на языке, понятном ~~только~~ им обоим. В случае с веб-страницами это язык гипертекстовой разметки HTML, но может быть и JSON или что-нибудь другое.

- **API** (англ. *Application Program Interface*, правильно эй-пи-а́й, *жарг.* [апи́]). По сути, электронный секретарь — составная часть сервера, которая отвечает за обработку входящих запросов и отправку

ответов. В 90% вакансий тестировщика на HeadHunter (кроме игрушек) в требованиях вы увидите именно «тестирование API» и страшные аббревиатуры REST и SOAP (нет, это не отдых и не мыло). Причём формулировка «тестирование API» сама по себе кривая: вы можете подумать, что тестировать нужно сам API, а на самом деле это просто ещё один способ тестирования функционала самого приложения/сайта/системы, просто не через видимый пользователю графический интерфейс (GUI), а с помощью всяких хитроумных программ, позволяющих отправлять запросы и получать ответы вообще без пользовательского интерфейса. Правильнее было бы сказать «Тестирование **через** API» или «Тестирование **с помощью** API». Наиболее известные программы — Postman для REST API и SOAP UI для протокола SOAP.

- **Сниффферы** (англ. *sniffers* — те, кто вынюхивают) — специальные программы для перехвата запросов и ответов. Наиболее известные — Fiddler и Charles. С их помощью можно также подменять ответы на запросы — и, например, смоделировать ситуацию, когда вместо текста «Приветствую!» в онлайн-инструкции по кофемолке юзеру появляется текст: «Ну что, уже сломал?». Сниффферы также полезны для моделирования низкой скорости соединения с сетью, чтобы выявить проблемы у юзеров с плохим интернетом.

- **Прокси-сервер** (англ. *proxy*, *жарг.* **проксѝ**[2]) — промежуточный сервер между клиентом и сервером. Если мы настраиваем на девайсе или в браузере прокси-сервер, клиент отправляет запросы сначала на прокси-сервер, а оттуда на основной сервер. В условиях тестов прокси-сервером обычно делают сниффер, чтобы отлавливать запросы с клиента на сервер и ответы сервера. Для этого в настройках браузера или девайса на текущей сети нужно указать ip-адрес и порт, совпадающий с ip и портом прокси-сервера. Если мы укажем невалидный прокси-сервер или не подтвердим использование сниффера как прокси, нас просто не будет пускать в интернет, т.к. клиент пытается отправить запросы, а прокси-сервер их не принимает. Тестировщики очень любят, когда взятый после кого-то девайс не пускает в интернет, потому что предыдущий тестер забыл убрать прокси с сети.

- **Базы данных (БД)** (англ. *database*) — хранилища информации, из которых сервер подтягивает информацию. Например, профиль юзера при его авторизации на сайте.

- **Системы управления базами данных (СУБД)** — К.О. подсказывает, что это программы для управления базами данных. Классический пример — Microsoft Access, входящий в базовый пакет Microsoft Office. Хотя многие в пику мелкомягким используют другие СУБД, например MySQL, Oracle, Postgre.

- **SQL** (англ. *structured query language* — «язык структурированных запросов») — язык программирования, с помощью которого можно делать запросы к БД, получать или изменять данные в ней. По сути это почти обычный английский, пример: SELECT maker FROM PC

WHERE price < 500. Во многих вакансиях тестировщиков вы увидите пожелания к владению им.

Преимущества умения тестировать API:

- **Раннее тестирование** — разработчики сначала делают API, а потом уже графический интерфейс. У вас есть шанс проверить логику раньше, чем ей дорисуют кнопочки в GUI.
- **Тестирование API** — графического интерфейса может в принципе не быть. Будет только API-метод. Такое часто бывает в enterprise-системах, предназначенных для хранения данных конкретной компании и не предполагающих взаимодействие с внешними юзерами.
- **Скорость** — вызвать один запрос занимает доли секунды. А вот через интерфейс повторить процедуру бывает сложно. Например, создать пользователя на 50 заполненных полей…
- **Точная локализация** — где конкретно произошла проблема? На сервере или клиенте? Проверьте работу сервера через API и узнаете точно.
- **Автоматизация** — даже если у вас нет автотестов на уровне API приложения, вы можете создать свои простенькие через Postman. Это поможет не гонять одно и то же вручную + быстро создавать большие объемы данных.

1. ↑ Менее знакомые с англо-русской практической транскрипцией говорят «краш», «крашить», хотя краш — это crush, совсем другое слово.
2. ↑ Слово склоняется: без проксѝ, убрать проксѝ, с проксёй.

Небыдло-термины

Рекомендуется прочитать, т.к. на собеседованиях любят спрашивать именно про них.

- **Waterfall** (каскад) — цикл разработки проекта, где разные виды деятельности строго разграничены по времени. Сначала планируем ресурсы и бюджет проекта, потом геймдизы планируют сам проект, потом кодеры делают ВЕСЬ проект, и только после этого подключаются тестировщики. Слабое место — поломка или внесение изменений на любом этапе стопит весь проект до исправления.

- **Agile**[1] (гибкий) — более предпочтительный цикл разработки проекта, где проект делается малыми временными циклами (спринтами). Например, в один спринт сделали полным циклом визуальное

оформление приложения (сдизайнили, закодили, протестировали), в следующий спринт уже подключаем к интерфейсу часть функционала, ещё в следующий — другую часть и т.п. Преимущество — функционал, добавляемый малыми частями, оперативно тестируется и быстрее вылавливаются критические проблемы. Кроме того, при таком подходе легче вносить изменения по ходу проекта и меньше риск, что они что-нибудь критично поломают и это будет обнаружено только перед релизом.

- **Канбан** (лун. «рекламный щит, вывеска») – система отслеживания задач по разработке продукта. Задачи наглядно представлены в виде карточек, которые вешаются на доску со столбцами: «сделать» (to do), «в процессе» (in progress), «сделано» (done) (столбцов может быть больше). По мере выполнения карточки с задачей перемещаются между столбцами, пока не достигнут столбца done. Канбан-доски могут быть и в электронном виде, например на сервисе Trello.

- **Скрам** (англ. *Scrum*) – система работы на проекте, когда проектом занимается постоянная команда (тестировщики, разработчики, дизайнеры, проект-менеджер), которая всё решает вместе и в которой все равны (т.е. нет лида). Тестирование делится на спринты (обычно 2-4 недели), в начале спринта команда собирается, оценивает задачи в бэклоге и набирает столько, сколько влазит в спринт. Дальше все берут таски в работу, потом передают на тест и далее. Также участники команды ежедневно встречаются/созваниваются, чтобы каждый отчитался, что делал вчера и что собирается делать сегодня. В этом случае ответственность за качество продукта несёт вся команда, а не только лид или проект-менеджер.

1. ↑ Правильно читать «эджайл»

Профессиональные качества и профессиональная этика

Тестер должен обладать теми же профессиональными качествами и этикой, которыми в средневековье обладал хороший профессиональный палач:

- Мучить точно по спецификации и без всяких личных чувств к пытаемому.
- Причинять максимальную боль при минимальном физическом вреде.
- Убивать и калечить лишь в том случае, если таков приговор.
- Не брать взяток, но снимать лучшие вещи с трупа.

Мемы

Хорошие, годные мемы про тестировщиков можно найти, например, тут

Цитатник

«— Почему люди курят?

— Чтобы у нас работа была!

»

— Безымянный тестер

> Некоторые из учёных, которые предсказывают постепенное вымирание человека, считают, что после нас на Земле воцарятся поумневшие бабуины. Вы будете удивлены, узнав, что в Стэмфордской зоологической школе бабуинов обучили профессии тестировщика программного обеспечения. Оказывается, обезьяны способны работать с персональным компьютером и запоминать компьютерное меню.
>
> Бабуины и шимпанзе вполне могут работать с компьютерами, заниматься тестированием программного обеспечения и даже программировать. Правда, у них возникают некоторые трудности со сложно структурированными меню. Если в меню больше двух уровней, то для бабуина оно уже представляет трудность, — говорит доктор Джеймс Маколифф. Но проблему двухуровневого меню удалось решить. Бабуины сумели научиться спускаться аж до 7 уровня. После этого они смогли освоить Windows — естественно, не очень хорошо, но всё же. Также животные работают с программой Visual Basic 3.0. При этом те из них, кто сумел освоить ПК, тут же становятся более уважаемыми в стае. А чтобы сохранить этот статус, бабуины не позволяют соплеменникам подсматривать, как они управляются с хитрой электронной машиной. Кстати, некоторые из контор, занимающихся производством программного обеспечения, уже заявили, что и дальше готовы финансировать подобные исследования. Это и

> не удивительно, ведь содержать умненького бабуина гораздо дешевле, чем платить зарплату программисту. Особенно, если дело происходит в США или Европе
>
> — *для непонявших юмора*

> xxx: Вам в компанию не нужен тестер?
>
> yyy: Посмотрим... А какой список достижений имеется? xxx: Послал тебе скрины на мыло. Там дампер упавшего Task Manager'a в аттаче, Access Violation в калькуляторе виндовом, две бубновых масти в косынке сверху и зависший ping. Достоен? yyy: В резюме шаги воспроизведения напиши - возьмем без собеседования.
>
> — *397083*

> big_nik90: думай как баг, действуй как баг, и ты найдешь баг)
>
> — *411549*

Платный бетатест

Хитрые и жадные разработчики компьютерных игр придумали способ находить баги в играх, не платя тестерам ни гроша, и даже более того — заставляя их покупать товар самим. Рецепт прост:

- Показываем кучу красивых роликов из игры.
- Выпускаем игру, какой бы забагованной она не была.
- Делаем сайт, вики или форум игры.
- Тысячи игроков приходят на форум и жалуются на баги.

За примерами далеко ходить не надо: S.T.A.L.K.E.R., Готика 3, Аллоды Онлайн, Disciples III, World of Tanks, Варфейс, Dota 2 (особо эпична тем, что бета-тест в какой-то момент стал платным, а релиз — бесплатный), Корсары 3 (замечательная идея — выпустить игру версии 0.99), The Long Dark Unity (свыше тысячи записей было добавлено в баг-треккер за первый месяц после выхода релиза), Assassin's Creed Unity, PC-версия Batman Arkham Knight. Также известна былинная история с выпуском от 1С игры Бригада Е5 (по мотивам Jagged Alliance) с такой хуевой тучей багов, что никому не удалось, купив игру, пройти её до конца. Форум Е5 чуть более, чем полностью забит темами о всевозможных вылетах игры. После официального релиза 1С сначала выпустила два патча к Е5, что сняло около 75% багов, а потом просто тупо выпустило как бы новую игру 7,62 — по сути та же Е5 избавленная от фантастического количества ошибок. Таким образом игруны, купившие Е5 за свои деньги провели тестирование для 1С. Частично игру оправдывает то, что делали ее 3,5 фаната на коленке и на тестинг у них тупо не хватило денег и времени.

Масштабы бедствия таковы, что в Стиме под подобное творчество сделан целый раздел, и некоторые игры существуют полноценной жизнью, не вылезая из него годами.

Кстати, некоторые особо параноидальные личности усматривают в этом ещё и борьбу с пиратством. Потому что очень тяжело пиратить игру, когда выходит по нескольку патчей в неделю, а то и в день. (что сомнительно)

Ссылки

- Расово верная книга «Дневник тестировщика»
- Винрарный рассказ про кликера
- Единственное кино про тестировщиков «Grandma’s boy» или в этой стране «Мальчик на троих»
- Кем тестируются игры (и для кого)

Языки программирования
++i + ++i 1C AJAX BrainFuck C Sharp C++ Dummy mode Erlang Forth FUBAR
God is real, unless explicitly declared as integer GOTO Haskell Ifconfig Java JavaScript LISP
My other car Oracle Pascal Perl PHP Prolog Pure C Python RegExp Reverse Engineering
Ruby SAP SICP Tcl TeX Xyzzy Анти-паттерн Ассемблер Быдлокодер
Выстрелить себе в ногу Грязный хак Дискета ЕГГОГ Индусский код Инжалид дежице
Капча КОИ-8 Костыль Лог Метод научного тыка Очередь Помолясь Проблема 2000
Программист Процент эс Рекурсия Свистелки и перделки Спортивное программирование
СУБД Тестировщик Умение разбираться в чужом коде Фаза Луны Фортран Хакер
Языки программирования

w:Тестирование программного обеспечения