Принципы тестирования

В переводе с латинского При́нцип - это основа, начало, первоначало, и можно сказать, что принципы тестирования — это основы тестирования

**1)-Исчерпывающее тестирование невозможно.**

Насколько бы тщательным тестирование не было, нельзя учесть все возможные сценарии и предвидеть все возможные ошибки.

Исчерпывающее тестирование в теории призвано проверить приложение «со всеми возможными комбинациями всех возможных входных данных во всех возможных условиях выполнения». Но — это невозможно физически, даже для одного простого поля для ввода имени пользователя может существовать порядка 2.4 позитивных проверок и бесконечное количество негативных проверок. Потому в силу законов физики нет ни малейшего шанса протестировать программный продукт полностью, «исчерпывающе». Однако, из этого не следует, что тестирование как таковое не является эффективным. Вдумчивый анализ требований, учёт рисков, расстановка приоритетов, анализ предметной области, моделирование, работа с конечными пользователями, применение специальных техник тестирования — эти и многие другие подходы позволяют выявить те области или условия эксплуатации продукта, которые требуют особенно тщательной проверки. И поскольку здесь объём работы несоизмеримо меньше — такое тестирование уже не просто возможно, но и выполняется на каждодневной основе.

**2️)- Тестирование демонстрирует наличие дефектов, а не их отсутствие.**

Тестирование может выявить тот момент, что ошибки присутствуют, но не может доказать в полной мере, что дефектов нет. Даже если мы полностью проверили систему.

Представьте, что вы купили кому-то апельсин. Идеальный. Самый лучший в мире. Достойный стать эталоном апельсинов на все времена. Но тот, кому вы покупали этот апельсин, разочарован — он ведь просил грейпфрут. Так и программный продукт должен не только быть избавлен от дефектов настолько, насколько это возможно, но и удовлетворять требованиям заказчика и конечных пользователей — в противном случае он станет непригодным для использования. Стоит отметить, что нередко нарушение данного принципа заключается в недостаточной проработке и реализации нефункциональных требований{41} к продукту, что влечёт за собой справедливую критику со стороны конечных пользователей и общее падение популярности продукта. Если объединить этот принцип с предыдущим, получается: именно понимание контекста продукта и потребностей пользователей позволяет тестировщикам выбрать наилучшую стратегию и добиться наилучшего результата.

**3)-Заблуждение об отсутствии ошибок.**

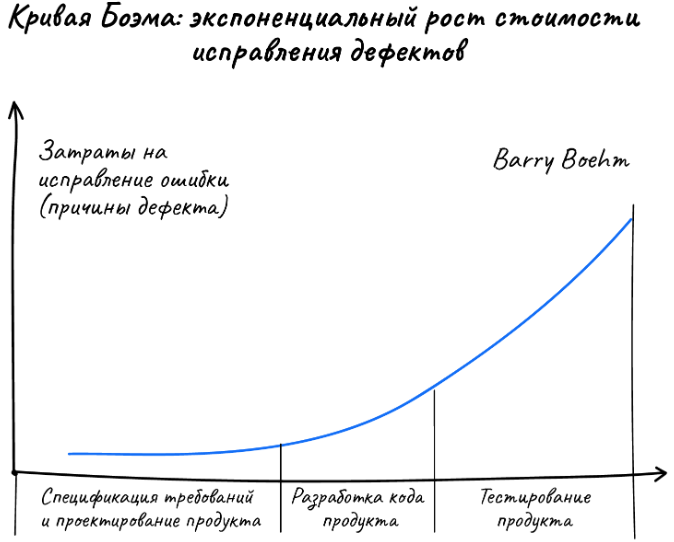
Можно сколько угодно находить ошибки, и даже, казалось бы, не обнаруживая их больше, нет гарантии того, что ошибки найдены все и продукт полностью качественный и готовый.

Дефекты однозначно будут. Но в тестировании и нет такой задачи, чтобы выявить 100% багов, т.к. мы уже знаем, что это невозможно, исходя из первых трёх принципов. Главное здесь – найти наиболее критичные ошибки.

**4)-Раннее тестирование сохраняет время и деньги, т.е. тестирование тем эффективнее, чем раньше оно выполняется**

чем раньше выявляется та или иная проблема – тем меньше средств и трудозатрат потребуется для её устранения. Соответственно, если баг попадёт в «прод» или ещё хуже, если его найдёт пользователь – исправление такого дефекта обойдётся немалой кровью для всей команды. Помимо того, что удаление его будет стоить бо́льших денег, нежели на начальной стадии разработки, может получиться так, что этот дефект затронет другой функционал. И тогда проблемы начнут накапливаться как снежный ком. Сколько кода потребуется переписать разработчикам? Сколько времени уйдет на исправление и тестирование?

У данного принципа есть прекрасная аналогия из обычной повседневной жизни. Представьте, что вы собираетесь в поездку и продумываете список вещей, которые необходимо взять с собой. На стадии обдумывания добавить, изменить, удалить любой пункт в этом списке не стоит ничего. На стадии поездки по магазинам для закупки необходимого недоработки в списке уже могут привести к необходимости повторной поездки в магазин. На стадии отправки на место назначения недоработки в списке вещей явно приведут к ощутимой потере нервов, времени и денег. А если фатальный недостаток списка вещей выяснится только по прибытии, может так оказаться, что вся поездка потеряла смысл.



**5)-Принцип скопления или кластеризация дефектов**

Существует такое определение – наибо́льшее количество дефектов обычно содержится в небольшо́м количестве модулей. Простыми словами кластеризация – это группировка (на кластеры) множества объектов, схожих между собой по каким-либо параметрам. Представим полки и витрины в магазине – товары подразделены на хлебобулочные, молочные, мясные, напитки и др. Это и есть кластеризация.

Давайте проведём параллель с багами. Ошибки скапливаются в определённых местах, например, там, где код наиболее сложный или некорректно написан. Любой продукт состоит из модулей – кластеров в нашем случае. Если в каком-то модуле нашлось несколько багов, - это сигнал к тому, чтобы ещё внимательнее протестировать или даже перелопатить его с особой тщательностью на наличие скрытых дефектов.

В таком случае обычно руководствуются частным случаем закона Паретто: в 20% модулей приложения сокрыто 80% всех дефектов.

**6)-Тестирование зависит от контекста**

Согласитесь, вы будете по-разному подходить к приготовлению «чего-нибудь перекусить для себя» и к организации семейного ужина по какому-то очень торжественному поводу. В тестировании логика та же: программные продукты могут относиться к разным предметным областям, быть построены с использованием различных технологий, использоваться для решения различных задач . — всё это и многое другое влияет на то, как должен быть организован процесс тестирования.

Набор характеристик программного продукта влияет на глубину тестирования, используемый набор техник и инструментов, принципы организации работы тестировщиков и т.д. Основная идея данного принципа состоит в том, что невозможно выработать некий «универсальный подход к тестированию» на все случаи жизни, и даже просто бездумное копирование подходов к тестированию с одних проектов на другие часто не заканчивается ничем хорошим. Если же принимать во внимание как общие, так и уникальные свойства текущего проекта и выстраивать тестирования соответствующим образом — оно оказывается наиболее эффективным и результативным.

Для разного софта будут применяться разные подходы к его тестированию. К примеру, способ тестирования мобильного приложения будет отличаться от того, которым тестируется коммерческий сайт.

По каким характеристикам можно различать контекст:

по типу продукта – web, desktop, мобильное приложение, сервис и др.;

по цели продукта – обеспечение безопасности, Game, продажа товаров и др.;

по проектной команде – специализация, количество человек, опыт и т.д.;

по доступным инструментам – что присутствует на проекте, для успешной реализации;

по срокам – как построен рабочий процесс, как часто выходят релизы, время между ними на подготовку;

по ожидаемому уровню качества – чем выше требования, тем тщательнее нужно тестировать.

**7)-Парадокс пестицида**

Если к какому-либо функционалу применять постоянно повторяющийся набор тестов – то эти проверки в скором времени будут неэффективны в нахождении новых дефектов.

Поэтому тест-кейсы должны постоянно обновляться и видоизменяться. Важно пользоваться такими рекомендациями:

-добавлять новые тесты;

-просматривать и изменять существующие;

-применять разные виды и техники тестирования;

-осуществлять тестирование новыми сотрудниками и др.

-В целом посмотреть на продукт под другим углом.