# Тестируйте умнее, а не сложнее

## Введение: сложность ведет к бесполезности

Представьте, что мы разрабатываем веб-страницу для настройки покупки ноутбука.

Если вы никогда раньше не настраивали параметры ноутбука через Интернет, посмотрите на страницу Dell’а «customize it» для ноутбука начального уровня. На веб-странице пользователю предлагается одиннадцать вопросов, каждый из которых имеет от двух до семи вариантов ответа. Пользователь должен выбрать один из двух вариантов в первом параметре, два во втором и так далее. У пользователя есть семь возможных вариантов для последнего параметра.

Если взглянуть на все параметры вместе получается, что пользователь должен сделать (2,2,2,2,2,3,2,2,3,4,7) выборов. Это простая проблема конфигурации (настройки параметров). Количество возможных конфигураций ноутбука, которые пользователь может запросить является произведением всех возможных вариантов. На этой очень простой странице 32 256 возможностей. На момент написания этой статьи страница для настройки высокопроизводительного ноутбука Dell содержала похожий набор параметров с бОльшим количеством вариантов для каждого из них: (3,3,3,2,4,2,4,2,2,3,7,4,4). Пользователь этой страницы может запросить любую из 2 322 432 различных конфигураций ноутбука! Если бы Dell добавила еще один параметр, предлагающий пять различных вариантов, было бы более десяти миллионов возможных комбинаций!

Создание набора тестов, который пробует все два миллиона комбинаций для высокопроизводительного ноутбук, можно автоматизировать, но даже если бы выполнение каждого теста занимало одну десятую секунды, это тестирование заняло бы более 64 часов! Dell меняет свои предложения продуктов за меньшее время.

Опять же, если мы будем использовать группу серверов для распределения тестирования по десяти машинам, мы сможем выполнить его примерно за 6 часов. Игнорируя тот факт, что мы будем запускать это тестирование для каждой страницы настройки параметров Dell, 6 часов не являются необоснованными.

Проверка двух миллионов результатов — вот где нас ждет по-настоящему большая проблема. Мы не можем рассчитывать на то, что люди будут вручную проверять все результаты — это слишком трудозатратно. Мы могли бы написать другую программу, которая проверяет эти выходные данные и оценивает их с помощью системы, основанной на правилах («Если пользователь выбирает 1 ГБ ОЗУ, тогда конфигурация должна включать 1 ГБ ОЗУ» и «Цена окончательной системы должна быть скорректирована ценой 1 ГБ ОЗУ соответственно с базовой ценой системы для этой модели.»)

Есть несколько хороших инструментов проверки, основанных на правилах, но они либо являются специализированным программным обеспечением, либо настолько универсальны, что требуют вложения больших усилий, чтобы сделать их применимыми для конкретного клиента. С системой проверки, основанной на правилах, мы тратимся на поддержание правил. Правила проверки необходимо будет регулярно обновлять, поскольку Dell меняет способ размещения, настройки и цены своих ноутбуков.

Поскольку мы не Dell, мы не на таком уровне (миллиарды долларов дохода), чтобы оправдать такие инвестиции. Суть в том, что мы не можем позволить себе исчерпывающе проверить каждую комбинацию. Акционеры Dell требуют от них развития своего бизнеса, и эти страницы конфигурации являются средством, с помощью которого Dell получает миллиарды долларов дохода. Они должны это проверять. Цена ошибок (сбоев, упущенных продаж, неправильных цен товаров, недопустимых комбинации особенностей) слишком высока. При таком уровне риска цена отказа от тестирования (цена низкого качества) чрезвычайно высока.

## Мы не можем позволить себе проверять это

Несколько лет назад мне удалось посетить тренинг с Кентом Беком. Я также был удостоен возможностью насладиться с ним большим стейком и холодным пивом в тот вечер после тренинга. Когда его спросили, как он реагирует на людей, которые жалуются на цену качества, Кент сказал нам, что у него есть очень простой ответ: «Если тестирование стоит больше, чем не тестирование, не надо делать этого».

Я согласен. Есть несколько ситуаций, когда стоимость качества превышает стоимость низкого качества. Это ситуации, когда необходимая инфраструктура, время разработки тестов и затраты на обслуживание перевешивают ожидаемые затраты на наличие ошибки. («Ожидаемая стоимость» — это вероятность (в процентах) появления ошибки, умноженная на стоимость устранения ошибки.)

Методы, описанные в этой статье, предназначены для снижения затрат на качество, чтобы еще менее вероятно можно было бы сказать, что «отказ от тестирования» - лучший ответ.

## Просто протестируйте все, это же автоматизировано!

Мы должны рассмотреть два «решения»: ничего не тестировать и тестировать все. Мы бы не стали ничего тестировать, если бы не могли позволить себе протестировать программное обеспечение. Когда люди не понимают сложности тестирования или ограничений автоматизированного тестирования, они склонны хотеть «протестировать все». Проще сказать, чем сделать.

Вы когда-нибудь были в проекте, где менеджер говорил что-то вроде: «Я требую полного тестирования программного обеспечения. Наша политика - абсолютная нетерпимость. У нас не будет низкого качества в мою смену»?

С чем мы боремся здесь, так это с непониманием того, что значит иметь «полное покрытие» или любую другую гарантию определенного количества дефектов.

В достаточно сложной системе нет абсолютов - но это нормально. Есть статистика, уровни уверенности и планы управления рисками. Как инженеры и разработчики программного обеспечения, наш мозг запрограммирован на работу с ожидаемым, вероятным и возможным будущим. Мы должны помочь нашим менее техническим собратьям понять эти концепции - или, по крайней мере, взглянуть на них в перспективе.

Нас могут спросить: «Почему мы не можем просто протестировать каждую комбинацию входных данных, чтобы убедиться, что мы получаем правильные выходные данные? У нас есть автоматизированный набор тестов - просто заполните его и запустите!»

Нам нужно противостоять желанию ответить им так: «Обезьяны с пишущими машинками завершат работы Шекспира, прежде чем мы закончим один прогон нашего набора тестов!»

## Решение проблемы

Есть много приложений, у которых есть миллионы или миллиарды комбинаций входных данных. У них есть автоматизированное тестирование. У них есть решения этой проблемы. Мы только что закончили обсуждение того, насколько непрактично проводить исчерпывающее тестирование, так как же компании тестируют свое сложное программное обеспечение?

В оставшейся части статьи мы рассмотрим следующие подходы к решению проблемы.

* Случайная выборка
* Парное тестирование
* N-мудрое тестирование (N-wise testing)

Мы также рассмотрим влияние изменения порядка операций на наш подход к тестированию и методы тестирования, когда последовательность имеет значение.