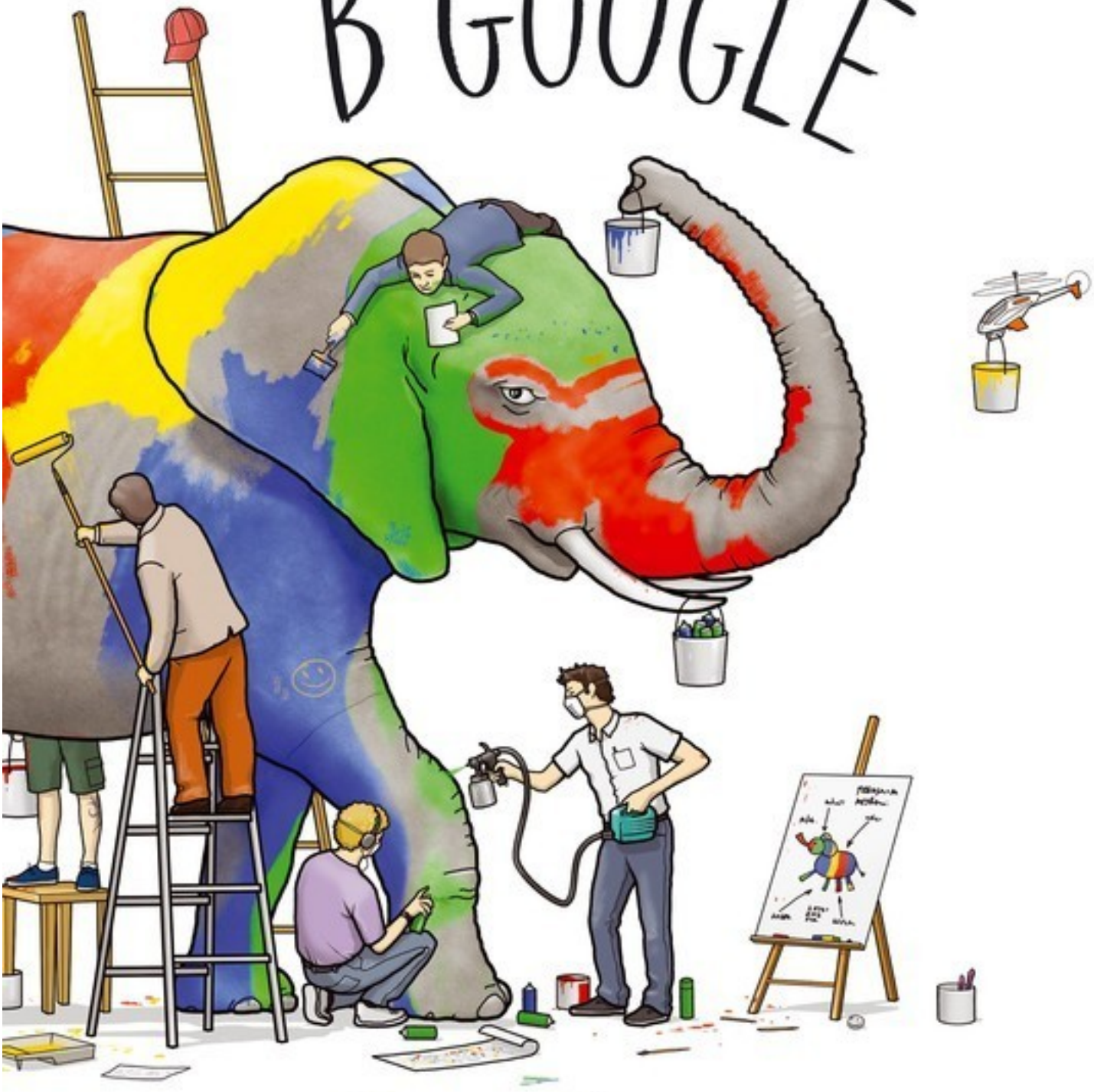
Джеймс Уиттакер
Джейсон Арбон
Джефф Кароло

КАК ТЕСТИРУЮТ В GOOGLE



Джефф Каролло

Как тестируют в Google

«Питер»
2012

Каролло Д.

Как тестируют в Google / Д. Каролло — «Питер», 2012

ISBN 978-5-496-00893-8

В книге описано тестирование программных продуктов в Google: как устроены процессы, как организованы команды, какие техники используются, кто ответственен за качество. Принципы, на которых построено тестирование в Google, применимы в проектах и компаниях любого размера. Авторы книги сами работали над продуктами Google, создавая инструменты тестирования, настраивая процессы и занимаясь непосредственно тестированием. Книга рассчитана на профессионалов из индустрии разработки программного обеспечения: специалистов по тестированию, программистов, менеджеров.

ISBN 978-5-496-00893-8

Содержание

Дж. Уиттакер, Дж. Арбон, Дж. Каролло
Как тестируют в Google

Всем тестировщикам из Google, Microsoft и всех остальных компаний, которые заставили меня мыслить нестандартно.
Джеймс А. Уиттакер

Моей жене Хизер и моим детям Луке, Матео, Данте и Одессе, которые думали, что я все это время работал в Starbucks.
Джейсон Арбон

Маме, папе, Лорен и Алексу.
Джефф Каролло

Предисловие к русскому изданию

Я пришла в тестирование в 2006 году маленьким тестировщиком на большой аутсорсный проект. Сначала я научилась тестировать, заводить баги и общаться с разработчиками и менеджерами. Со временем я стала писать тесты, научилась планировать и управлять тестированием. У меня появилась своя команда.

И чем дальше, тем больше мне становилось понятно, что тестировщики только находят проблемы, но не могут их исправить. Они не могут сделать так, чтобы проблема больше не повторилась. И я чувствовала, что тестирование может приносить больше пользы.

Я начала ездить на конференции, читала книги и статьи по тестированию, общалась с коллегами по индустрии. Везде учили, как лучше тестировать, как находить больше ошибок, как быстрее находить ошибки. Тестировщики не хотели выходить за рамки своей профессии. Им как будто нравилось чувствовать собственную важность от того, что они нашли много багов.

Ответы на свои вопросы я нашла в статьях и докладах Джеймса Уиттакера. Его основная идея в том, что тестирование должно перестать просто предоставлять информацию и начать влиять на качество. Главная задача тестирования, говорил Уиттакер, – это уменьшение количества ошибок в процессах разработки. Тогда улучшится качество выпускаемого продукта.

Создать процесс, в котором сложно допустить ошибку, – вот настоящая цель тестирования. Мы не можем полностью избавиться от ошибок, но можем построить работу так, что сделать сразу правильно будет легче, чем ошибиться.

В Google пошли именно в эту сторону, отказавшись от тестирования, которое просто сообщало об ошибках. «Служба тестирования» трансформировалась в «Направление продуктивности разработки», которое помогает разработчикам и менеджерам делать меньше ошибок и получать обратную связь как можно раньше. Тестировщики в Google влияют на качество, потому что встраивают его на всех этапах разработки программных продуктов.

Книга «Как тестируют в Google» рассказывает, как тестировщики в Google пришли к пониманию, что нужно меняться, как строили новые процессы, каких результатов достигли и как видят дальнейшее развитие тестирования. Очень здорово, что компания Google настолько открыта и делится своим опытом. Но все же, я думаю, что именно приход Уиттакера в Google послужил катализатором процесса написания этой книги. Слишком уж много у него энергии и желания делиться знаниями с внешним миром.

Я начала переписываться с Уиттакером в 2009 году, а в марте 2010-го познакомилась с ним лично на конференции Swiss Testing Day. Он оказался очень простым в общении. С ним можно было обсудить тренды мирового тестирования и посмеяться над тестировщиками-консерваторами. Джеймс очень остроумный и харизматичный оратор. В 2011-м я слушала его мастер-класс «Как тестируют в Google» на конференции EuroSTAR. Он легко парировал реплики из зала вроде «в авиационном софте не получится сделать так, как в Google» простой фразой «так вы даже не пробовали».

Эта книга повторяет легкий и остроумный стиль общения Джеймса. Вы почувствуете характер Уиттакера, читая ее. Мы очень старались сохранить в переводе эту легкость и живость. Я хотела, чтобы текст получился таким, как будто его написал кто-то очень знакомый: наш коллега, который говорит на том же языке, что и мы. В итоге книга получилась не похожей на большинство технических книг на русском языке, переведенных сухо и формально.

Если вы начинающий тестировщик – прочитайте книгу сейчас, а потом еще раз через год. Если вы опытный тестировщик – дайте ее почитать вашим разработчикам и менеджерам. Если они не загорятся идеей сделать тестирование «как в Google», задумайтесь, стоит ли с ними работать. Если вы менеджер или разработчик – прочитайте и сравните с тем, как работает тестирование в вашей компании.

Пожалуйста, будьте осторожны. Не нужно следовать советам ребят из Google вслепую: то, как у них реализованы процессы, лишь частный случай. Вычлените из книги главное: тестирование как выделенная проверка не работает, тестировать должен каждый участник проекта, качество нужно встраивать на все уровни разработки. Если применить эти принципы в вашей компании, скорее всего, вы получите совершенно другую реализацию.

Думайте, «как в Google», а не делайте «как в Google». Приятного чтения!

Спасибо всем ребятам, кто помогал в начале проекта: Илье Фомину, Ане Якшиной, Наташе Курашкиной, Леше Лянгузову, Севе Лотошникову, Игорю Любину, Ване Федорову, Жене Ткаченко, Илье Шубкину. Спасибо «Иннове» и Геворку, что этот проект состоялся. Спасибо Ксюше Развенской и Сергею Сурганову, что помогали вычитывать финальный вариант.

Юля Нечаева

P.S. У меня остался только один вопрос к Джеймсу Уиттакеру: о каком инциденте с костюмом Бедняжки Мэри говорит Фред в четвертой главе?

Вступление от Альберто Савоя

Писать вступление к книге, автором которой вы сами хотели быть, – сомнительное удовольствие. Все равно что стать шафером на свадьбе вашего друга и девушки, на которой *вы сами* давно мечтали жениться. Но Джеймс Уиттакер – хитрый малый. Прежде чем спросить, не соглашусь ли я написать это предисловие, он воспользовался моей слабостью к мексиканской кухне и угостил меня превосходным обедом. Только после того, как мы опустошили пару бутылок пива, он задал мне этот вопрос «на десерт». К этому моменту я был такой же мягкий и податливый, как порция гуакамоле, которую только что прикончил. «*Si, senor*» – вот и все, что я смог ответить. План сработал, и вот Джеймс стоит со своей книгой, как с невестой, а я собираюсь произнести свадебную речь.

Как я уже сказал, он хитрый малый.

Ну, поехали… предисловие к книге, которую я хотел бы написать сам. Звучит трогательная свадебная музыка.

Нужна ли миру еще одна книга по тестированию ПО? Особенно если это еще одна книга по тестированию ПО, написанная плодовитым Джеймсом Уиттакером, которого я зову «октомамой»[1] книг по тестированию? Разве мало книг, раздающих сомнительные и устаревшие советы по методологии тестирования? Да, таких книг достаточно, но это произведение, боюсь, к ним не относится. Вот почему я хотел бы написать его сам. Миру действительно нужна именно такая книга по тестированию.

Интернет значительно изменил подход к проектированию, разработке и распространению программных продуктов. Многие передовые методы тестирования, описанные в книгах прошлых лет, сейчас уже неэффективны, иногда вообще бесполезны, а в некоторых случаях даже вредны. В нашей отрасли условия меняются очень быстро. Книги, написанные пару лет назад, уже похожи на древние медицинские манускрипты по лечению пиявками и сверлению дырок в черепе для изгнания злых духов. Такие книги лучше переработать в подгузники для взрослых, чтобы они не попадали в руки доверчивых неофитов.

Развитие отрасли разработки ПО идет семимильными шагами, и я не удивлюсь, если через десять лет эту книгу тоже можно будет пустить на подгузники. Но пока не произошла очередная смена парадигмы, книга «Как тестируют в Google» дает очень своевременный и практичный взгляд на то, как одна из самых успешных и быстро развивающихся интернет-компаний справляется с уникальными задачами тестирования в XXI веке. Джеймс Уиттакер и его соавторы ухватили самую суть того, как Google успешно тестирует одни из самых сложных и популярных программных продуктов нашего времени. Я наблюдал становление отрасли и знаю, о чем говорю.

В 2001 году я присоединился к Google в качестве директора по разработке. Тогда у нас было около двухсот разработчиков и… целых *три* тестировщика! Мои разработчики уже тестировали свой код сами, но метод TDD (Test-Driven Development, разработка через тестирование[2]) и средства автоматизации тестирования (такие как JUnit) только делали свои первые шаги. Поэтому наше тестирование было по большей части случайным и зависело от добросовестности разработчика, написавшего код. Но это нормальная ситуация для стартапа. Нам нужно было двигаться быстро и не бояться рисковать, иначе мы бы не смогли конкурировать с матерыми игроками нашей отрасли.

---

[1] Не поняли, о чем речь? Погуглите!

[2] Разработка через тестирование – процесс разработки программного обеспечения, который основан на том, что тесты пишутся до того, как будет написан код. Таким образом, функциональные обязанности кода определяются заранее. – Примеч. перев.

Однако компания росла, и наши продукты становились все более значимыми для пользователей и клиентов. Возьмем хотя бы AdWords: один из сервисов, за которые я отвечал, он быстро превращался в основной источник монетизации веб-сайтов. Пришло время пересмотреть свой подход к тестированию и увеличить инвестиции в эту область. Имея в арсенале только трех тестировщиков, мы не нашли других вариантов, кроме как попытаться вовлечь в тестирование разработчиков. Я и еще несколько сотрудников Google продвигали, развивали и обучали юнит-тестированию. Мы агитировали разработчиков начать писать тесты и использовать хотя бы JUnit для их автоматизации. Переход шел со скрипом. Идея тестирования собственного кода не вызывала у разработчиков ажиотажа. Для стимулирования сотрудников каждую пятницу на наших пивных вечеринках с говорящим названием «Thanks God, It's Friday!» я награждал разработчиков, которые писали тесты. Я напоминал дрессировщика, который дает собачкам лакомство за выполненный трюк, но, по крайней мере, это привлекало внимание к тестированию. Неужели мне повезло, и я смог заманить разработчиков в тестирование конфетками?

К сожалению, нет. Идея с поощрением не сработала. Разработчики поняли, что для создания нормальных тестов им нужно писать две-три строки кода юнит-тестов на каждую строку тестируемого кода. Вдобавок тесты тоже нужно сопровождать, и в них самих могут заводиться баги. Стало очевидно, что одними разработчиками в юнит-тестировании сыт не будешь. Нам нужны были интеграционные тесты, системные тесты, тесты пользовательского интерфейса и прочие полезные штуки. Нам предстояло еще много узнать о тестировании и многому научиться, причем сделать это надо было быстро. Очень быстро!

К чему такая спешка? Я не верю, что даже самое изощренное тестирование может привести к успеху изначально неудачную идею или непродуманный продукт. Но я верю, что плохое тестирование может загубить хороший продукт, компанию или, по меньшей мере, сможет замедлить ее рост и предоставит фору конкурентам. Это как раз портрет Google того времени. Мы были на пути к успеху, и только проблемы в тестировании удерживали компанию от решающего шага в светлое будущее. Нужно было выработать верные стратегии тестирования, адекватные нашим сверхзвуковым скоростям роста числа пользователей, продуктов и сотрудников. Мы вовсю применяли инновационные решения, нестандартные подходы и уникальные инструменты – нам нельзя было замедляться. Конечно, не все работало. Но в процессе мы получили ценные уроки и научились практикам, которые теперь может применить любая компания, желающая развиваться со скоростью Google. Мы поняли, как сохранить баланс между качеством продукта и скоростью разработки, не теряя ни в чем. Эта книга рассказывает о процессе, к которому мы пришли, о том, какие идеи лежат в его основе, и почему он работает. Если вы хотите понять, как Google справляется с проблемами тестирования в XXI веке, в среде современных интернет-, мобильных и клиентских приложений – вы обратились по адресу. Конечно, я хотел бы оказаться тем человеком, который рассказал бы вам все это, но Джеймс и его соавторы уже сделали это за меня. Главное, что им удалось передать суть тестирования в Google.

И последняя ремарка: Джеймс Уиттакер – человек, благодаря которому эта книга написана. Он пришел в Google, проникся нашей культурой, взялся за большие и важные проекты, выпустил такие громкие продукты, как браузер Chrome, операционная система Chrome OS и десятки других поменьше. В какой-то момент он стал лицом тестирования в Google. В отличие от его других книг, в этой – большая часть материала не его авторства, он скорее выступил в роли корреспондента с места события и представил свой репортаж об эволюции тестирования ПО в Google. Помните об этом, читая книгу, потому что Джеймс наверняка постарается прибрать всю славу себе!

Пока население Google росло с 200 до 20 000 сотрудников, было немало людей, которые оставили свой след в разработке и благодаря которым наши идеи тестирования живут. Джеймс благодарит многих из них. Они появляются на страницах книги в отступлениях и

интервью. Тем не менее ни я, ни Джеймс не сделали так много, как Патрик Коупленд – архитектор нашей текущей организационной структуры и руководитель направления продуктивности разработки Google. Все тестировщики Google подотчетны Патрику. Это его видение реализовалось в наших подходах и вот теперь описано в этой книге Джеймсом. Если есть человек, благодаря которому тестирование ПО в Google организовано именно так, то это Патрик… Я говорю это не просто потому, что он мой начальник. Я говорю это потому, что он мой начальник, *а еще* он приказал мне это сказать!

Альберто Савоя – директор по разработке и популяризатор инноваций в Google. Он пришел в компанию в 2001 году и возглавил выпуск Google AdWords, а кроме того, сыграл ключевую роль в развитии культуры разработки и юнит-тестирования в компании. Он написал книгу «The Way of Testivus» и раздел «Beautiful Tests» в книге издательства O'Reilly «Beautiful Code».

Примечание от Джеймса Уиттакера: Согласен на все сто! Я ничего не придумывал, просто пересказал схему организации, которую создал Патрик. И я говорю это не потому, что он разрешил мне написать эту книгу. Как мой начальник, он *заставил* меня написать ее!

Вступление от Патрика Коупленда

Мое приключение в Google началось в марте 2005 года. Если вы прочитали вступление Альберто, то вы уже примерно представляете, что тогда происходило в компании. Google был еще маленький, но в нем уже начали проявляться признаки болезни роста. Это было время стремительных технологических изменений: на замену клиент-серверной модели приходили динамический контент и облачные технологии.

В первую неделю я, как и остальные новички, сидел в фирменной трехцветной кепке с пропеллером, слушая, как основатели компании обсуждают корпоративную стратегию на еженедельном собрании «Thanks God, It's Friday». Тогда я еще не понимал, во что влип. Я был наивен настолько, чтобы испытывать энтузиазм, и достаточно опытен, чтобы начать подозревать неладное. Мой предыдущий опыт с пятилетними циклами разработки продукта выглядел не очень убедительно по сравнению со скоростью роста и масштабами Google. Что еще хуже, я был единственным тестировщиком в кепке новичка. Я надеялся, что где-то есть и другие!

Когда я пришел в Google, там было не больше тысячи инженеров. В команду тестирования входили 50 штатных сотрудников и несколько временных, которых я никогда не мог запомнить. Все это называлось «Службой тестирования». Она занималась тестированием пользовательского интерфейса и прыгала с проекта на проект по мере надобности. Как можно догадаться, это была не самая популярная команда Google.

На тот момент этого было достаточно. Основными направлениями Google были поиск и реклама. Сфера деятельности Google тогда была значительно у́же, чем сейчас, и для поддержки качества достаточно было исследовательского тестирования. Но мир менялся. Пользователи приходили в веб в невиданных количествах, и веб стал ориентироваться на приложения, а не на документы. Все чувствовали необходимость роста и изменений. Если ты не хотел оказаться за бортом индустрии, нужно было уметь оперативно масштабировать продукт и быстро выпускать его на рынок.

Внутри Google команда службы тестирования сражалась с драконами масштаба и сложности. Решения, которые хорошо работали в мелких однородных проектах, теперь только обжигали наших славных тестировщиков, прыгающих с одного горящего проекта на другой. Прибавьте к этому стремление Google форсировать выпуск продуктов. Нужно было что-то делать. Я должен был выбрать между масштабированием ручного труда и полным изменением правил игры в тестирование. Тестирование должно было радикально измениться, чтобы соответствовать радикальным изменениям в индустрии и в Google.

Мне бы очень хотелось сказать, что я воспользовался своим огромным опытом и с ходу выдал идеальный механизм организации тестирования. Но, честно говоря, мой опыт научил меня только тому, *как делать не надо*. Каждая структура тестирования, в которой я участвовал (или которую возглавлял), была по-своему ущербной. В ней всегда что-то работало плохо: иногда код, иногда тесты, иногда сама команда. Я отлично знал, как можно оказаться погребенным под грудой технических и качественных проблем, когда каждая новая идея с ходу отвергается, если она хоть немного угрожает хрупкому проекту. Если я чему-то и научился к этому времени, так это тому, как *не нужно* проводить тестирование.

Я уяснил одно: Google уважает компьютерные технологии и искусство программирования. Если тестировщики хотят влиться в эту компанию, они должны быть технически грамотными и уметь писать код. Только так остальные инженеры будут воспринимать тестировщиков равными себе.

Ну и уж если я собрался что-то менять в структуре тестирования Google, то начинать надо было с самих тестировщиков. Я представлял идеальную команду и то, как она будет обеспечивать качество, и все время возвращался к одному очень важному условию: команда может

сделать качественный продукт только в том случае, если за качество отвечают все ее участники. Менеджеры, разработчики, тестировщики… все. Мне казалось, что лучший способ это обеспечить – превратить *тестирование* в фичу *кода*. Фича «тестирование» должна была обладать теми же правами, что и любая другая фича, которую может увидеть пользователь. А для создания фич нам нужен разработчик.

Нанять тестировщиков, которые умеют программировать, нелегко, а найти разработчиков, которые умеют тестировать, еще сложнее. Но текущее состояние дел было хуже некуда, и я решил действовать. Я хотел, чтобы тестировщики могли сделать больше для своих продуктов, и в то же время я хотел изменить природу тестирования и разделить ответственность за него. Для этого мне нужна была поддержка команд разработки. Я еще ни разу не встречал подобной организационной структуры за все время работы в отрасли, но был уверен, что она подойдет Google, и считал, что наша компания к ней готова.

К сожалению, мало кто в компании разделял мой энтузиазм по поводу столь принципиальных и фундаментальных изменений. Рассказывая о своем подходе «все будет точно так же, только по-другому», я обнаружил, что мне стало труднее найти компанию для обеда! Разработчиков, казалось, пугала мысль о том, чтобы играть более важную роль в процессе: «А тестировщики нам зачем?» Тестировщики тоже не горели желанием менять привычный стиль работы. Попытки что-либо изменить сталкивались с сильным сопротивлением.

Я продолжал гнуть свою линию, так как боялся, что инженерная работа Google погрязнет в технических долгах и проблемах качества, а мне придется вернуться к пятилетним циклам разработки, которые я с радостью оставил в доисторическом клиент-серверном мире. Google – компания гениев, ратующих за инновации, и этот дух изобретательства просто несовместим с долгими циклами выпуска. Игра стоила свеч. Я убедил себя, что однажды эти гении прочувствуют необходимость объединения практик разработки и тестирования в один цикличный и высокотехнологичный (как на фабрике) процесс и перейдут на мою сторону. Они поймут, что мы уже не стартап. Наша стремительно растущая пользовательская база, набирающий скорость снежный ком багов и плохо структурированный код могут разрушить уютный мирок разработчиков.

Я начал обходить команды разработки продуктов, разъясняя свою позицию, и старался подбирать подходящие аргументы для каждого. Разработчикам я рисовал картину непрерывной сборки, быстрого развертывания и процесса разработки настолько стремительного, что останется время на эксперименты. Общаясь с тестировщиками, я уповал на их желание стать равноправными разработчикам инженерами с равным уровнем мастерства, равным вкладом в продукт и равным уровнем оплаты.

Разработчики считали, что если уж мы собрались нанимать людей, которые могут писать функциональность, то пусть они это и делают. Некоторые так сильно противились моей идее, что почтовый ящик моего начальника был всегда полон советов, как лучше приструнить мое безумие. К счастью, мой начальник не внял этим рекомендациям.

Тестировщики, к моему удивлению, реагировали аналогично. Они привыкли к текущему состоянию дел, охотно оплакивали свой статус, но не торопились его менять.

Когда я пожаловался своему начальнику, он сказал: «Это Google. Если ты хочешь, чтобы что-то было сделано, – сделай это».

Сказано – сделано. У меня было достаточно единомышленников, чтобы стартовать конвейер собеседований, и мы начали беседовать с кандидатами. Задача была не из легких. Мы искали людей, обладающих умениями разработчика и менталитетом тестировщика. Нам были нужны специалисты, которые умели программировать и хотели применить свои навыки в разработке инфраструктуры, инструментов и автоматизации тестирования. Мы должны были придумать новые критерии набора персонала и проведения собеседований, а затем объяснить этот процесс нашим рекрутерам, которых вполне устраивал текущий ход дел.

Первые несколько месяцев были самыми тяжелыми. В процессе проверки часто отбраковывались неплохие кандидаты. Возможно, они недостаточно быстро решали какую-нибудь хитроумную задачу по программированию или плохо проявляли себя в области, важной для интервьюера, но на самом деле не имеющей никакого отношения к навыкам тестирования. Я знал, что с наймом у нас будут проблемы, и каждую неделю строчил докладную за докладной. Они уходили Ларри Пейджу, который был (и остается) последней инстанцией в процессе найма. Он утвердил достаточное количество кандидатов, и моя команда начала расти. Я иногда думаю, что мое имя у Ларри ассоциируется только с фразой «Нам нужны тестировщики!».

Конечно, я поднял столько шума, пытаясь найти поддержку, что пути назад у нас уже не было. За нами наблюдала вся компания, и неудача могла стать фатальной. От маленькой тестовой команды с вечно меняющимся штатом контрактников ожидали великих свершений. И все же – на фоне моих битв с наймом и сокращения числа внештатных сотрудников – я начал замечать изменения. Чем сильнее становился дефицит тестировщиков, тем больше тестовой работы приходилось выполнять разработчикам. Многие команды приняли вызов. Думаю, если бы технологии не развивались, уже этого было бы достаточно, чтобы со временем выйти на нужный уровень.

Но технологии не стояли на месте, поэтому правила разработки и тестирования быстро менялись. Эпоха статического веб-контента уходила в прошлое. Автоматизация не успевала за и без того отстающими браузерами. Взваливать тестирование на разработчиков в то время, когда они столкнулись с серьезными технологическими сдвигами, было неразумно. Мы не могли организовать нормальное тестирование этих приложений даже в ручном режиме, не говоря уже об автоматизации.

Команды разработчиков испытывали сильное давление. Google начал покупать компании со своими веб-приложениями (YouTube, Google Docs и прочие), которые только усиливали нагрузку на нашу внутреннюю инфраструктуру. Проблемы, с которыми я сталкивался в тестировании, не уступали проблемам, с которыми сталкивались разработчики при написании кода. Я пытался решить проблему тестирования, которую просто невозможно было решить отдельно от разработки. Рассматривать тестирование и разработку как обособленные дисциплины или даже как разнородные задачи было в корне неверно, и мы не могли решить ни одну из проблем. Улучшение команды тестирования стало бы всего лишь временным успехом.

Но все-таки положительные сдвиги начались. Полезно нанимать умных людей – они могут сами исправить ситуацию. К 2007 году позиции дисциплины тестирования укрепились. Мы неплохо справлялись с завершающей фазой цикла выпуска. Команды разработки знали, что могут рассчитывать на нас как на партнеров в производстве. Правда, наше существование как команды поддержки позднего цикла ограничивало нас рамками традиционной модели контроля качества. Несмотря на неплохие результаты, мы все еще не достигли состояния, в котором я хотел бы оказаться. Я разобрался с проблемой найма, тестирование двигалось в нужном направлении, но мы все еще слишком поздно включались в процесс.

Наша концепция «Тест-сертификации» делала успехи (вы прочитаете про нее дальше в книге). Мы общались с командами разработчиков и помогали улучшить чистоту кода и внедрить юнит-тестирование на ранней стадии. Мы разрабатывали инструменты и обучали команды основам непрерывной интеграции, чтобы наши продукты всегда были готовы к тестированию. Бесчисленные мелкие усовершенствования и приемы, многие из которых описаны в книге, развеяли скептицизм. Тем не менее чего-то не хватало. Разработка все еще была разработкой, а тестирование так и оставалось тестированием. Все ингредиенты для изменения культуры присутствовали, но нам был нужен катализатор, который смог бы запустить объединение дисциплин.

Глядя на организацию, которая выросла из моей идеи о привлечении разработчиков на позиции тестирования, я понял, что тестирование было лишь частью нашей работы. У нас были

команды, которые создавали все инструменты – от репозиториев исходного кода до инфраструктуры для багтрекинговой системы. Среди нас были инженеры по тестированию, инженеры по выпуску продуктов, разработчики инструментов и консультанты. Меня поражало, как сильно тот аспект нашей работы, который не являлся непосредственно тестированием, влиял на производительность. Возможно, наша команда называлась «службой тестирования», но наши обязанности были намного шире.

Поэтому я решил оформить все официально и сменил название команды на «направление продуктивности разработки». Смена названия привела к культурному сдвигу. Люди начали говорить не о контроле качества и тестировании, а о производительности. Повышение производительности – наша работа, а тестирование и контроль качества – работа всех, кто пишет код. Это означает, что и тестирование, и контроль качества входят в обязанности разработчика. Команда продуктивности разработки должна заботиться о том, чтобы разработчик мог заниматься этими двумя вещами.

На первых порах идея была абстрактной, а наш девиз «Ускоряем Google» поначалу звучал наигранно. Со временем мы выполнили обещание. Наши инструменты ускорили темпы разработки, мы переходили от одного бутылочного горлышка к другому, расчищали пути и решали проблемы, с которыми сталкивались разработчики. Наши инструменты помогли разработчикам писать тесты и наблюдать их результаты при каждой сборке. Тестовые сценарии уже не выполнялись локально на компьютере тестировщика. Их результаты публиковались на информационных панелях и собирались от версии к версии, так что в итоге они становились частью общедоступных данных о готовности приложения к выпуску. Мы не просто требовали участия разработчиков, мы упрощали его. Разница между тестированием и обеспечением продуктивности наконец-то стала ощутимой: Google мог создавать новое с меньшими затратами и обходиться без накопления технического долга.

Результаты? Пожалуй, я не буду предвосхищать содержание книги, потому что ее написали для того, чтобы рассказать о них миру. Авторы приложили огромные усилия, чтобы на основании собственной работы и опыта своих коллег из Google свести наш секретный рецепт успеха к конкретному набору практических методов. Мы добились успеха во многих отношениях – от сокращения времени сборки до автоматизированного тестирования в формате «запустил и забыл» и публикации в опенсорс новаторских инструментов тестирования. Когда я писал это вступление, в направлении продуктивности разработки работало около 1200 специалистов, что немногим больше численности всего технического персонала в 2005 году, когда я пришел в Google. Бренд «продуктивности» укрепился, а наше стремление ускорить Google стало неотъемлемой частью нашей технической культуры. С того первого дня, когда я сидел озадаченный и неуверенный на пятничной встрече, наша команда проделала путь длиной во много световых лет. Единственное, что не изменилось с тех пор, – моя трехцветная кепка с пропеллером. Она лежит у меня на столе и напоминает о том, как далеко мы ушли.

Патрик Коупленд – старший директор направления продуктивности разработки, вершина пирамиды тестирования в Google. Все тестировщики в компании подчиняются Патрику, а его руководителем, кстати, является Ларри Пейдж, соучредитель и исполнительный директор Google. Карьере Патрика в Google предшествовала почти десятилетняя работа в Microsoft на должности директора по тестированию. Он часто выступает публично и известен как архитектор используемой в Google технологии быстрой разработки, тестирования и развертывания ПО.

Предисловие

Разрабатывать программные продукты сложно. Тестировать эти продукты тоже сложно. Когда кто-то говорит о разработке и тестировании в масштабах веб, он подразумевает Google. Если вы хотите узнать, как в одной из самых известных интернет-компаний решаются проблемы тестирования в грандиозных рамках всей Сети, то вы держите в руках правильную книгу.

В Google ежедневно тестируются и выпускаются сотни миллионов строк кода, распределенного по миллионам исходных файлов. Миллиарды операций по сборке запускают выполнение миллионов автоматизированных тестов в сотнях тысяч экземплярах браузеров ежедневно. Операционные системы строятся, тестируются и выпускаются в пределах одного календарного года. Браузеры собираются ежедневно. Веб-приложения выпускаются почти непрерывно. В 2011 году сто новых фич Google+ были выпущены за сто дней.

Таковы масштабы и скорость Google – они же масштабы развития веб. О том, как Google организовал тестирование в таких условиях, расскажет эта книга. Мы покажем, как проектировалась, внедрялась и сопровождалась эта инфраструктура. Мы познакомим вас с людьми, которые повлияли как на разработку основных концепций этой структуры, так и на ее реализацию.

Конечно, так было не всегда. Путь, который прошел Google к тому, где он сейчас, не менее интересен, чем технологии тестирования, которые мы используем. Давайте переведем стрелки часов на шесть лет назад – тогда ситуация в Google слабо отличалась от ситуации в других компаниях, с которыми мы работали: дисциплина тестирования была второстепенной.

Специалисты этой области получали меньше, хотя и работали больше. Тестирование в основном было ручным, а люди, которые могли автоматизировать процесс, быстро переходили в разработчики, где они могли «приносить больше пользы». Основателям команды, которая сейчас называется направлением продуктивности разработки, пришлось продираться сквозь заблуждения о тестировании и корпоративную культуру, ставившую героические авралы на работе выше повседневной инженерной рутины. Сегодня зарплата тестировщиков Google соразмерна зарплате разработчиков, их премии и темпы карьерного роста сравнялись. То, что культура тестирования пережила непростой рост компании (по всем параметрам – количеству, разнообразию продуктов и объемам продаж) и структурные реорганизации, должно придать уверенности компаниям, которые решат следовать примеру Google. Тестирование можно организовать правильно. Добиться вменяемого отношения к нему со стороны разработчиков и руководства – реально.

Сейчас все больше компаний делает ставку на веб, поэтому технологии тестирования и организационная структура, описанные в этой книге, могут получить широкое распространение. Если вы с нами – читайте, эта книга объяснит вам, как добраться до цели.

Материал книги по тестированию разделен на части с описанием отдельных ролей, причастных к тестированию. В первой главе мы рассмотрим все роли в общем и разберем концепции, процессы и тонкости контроля качества Google. Обязательно прочитайте этот раздел.

Остальные главы можно читать в любом порядке. Сначала мы расскажем о роли разработчика в тестировании (Software Engineer in Test, SET), потому что с нее началось современное тестирование в Google. Разработчик в тестировании – тестировщик с высокой технической квалификацией, поэтому в этой главе много технических подробностей. Но мы постарались подать их в более общем ключе, чтобы основные концепции понял даже неподготовленный читатель. В следующей главе мы познакомимся с инженером по тестированию (Test Engineer, TE). Глава получилась большой, потому что обязанностей у инженера по тестированию много. В Google эти специалисты решают уйму разных задач на протяжении всего цикла разработки

продукта. Эта роль хорошо знакома многим традиционным тестировщикам. Скорее всего, этот раздел книги будет максимально востребован, потому что он адресован самой широкой аудитории.

В книге мы соблюдали баланс между теоретическими основами и практическими советами ключевых участников, сыгравших важные роли в становлении тестирования или в истории ключевых продуктов Google. Эти интервью будут интересны всем, кто пытается опробовать в своей команде или компании процессы тестирования в стиле Google.

Остается последняя глава, которую не должен упускать ни один читатель, интересующийся темой. В ней Джеймс Уиттакер делится своими представлениями о том, как развивается тестирование в Google, и делает прогнозы о будущем тестирования в Google и в отрасли в целом. Думаю, многие читатели найдут этот материал поучительным, а может быть, даже в чем-то провокационным.

Об иллюстрациях

Из-за сложности рассматриваемых тем и графической природы интернет-контента некоторые иллюстрации из третьей главы дают лишь общее представление о концепциях. Они не предназначены для подробного рассмотрения. Если вы захотите просмотреть эти рисунки на своем компьютере, загрузите их по адресу *www.informit.com/title/9780321803023*.

Пара слов о книге

Когда Патрик Коупленд предложил мне написать эту книгу, я колебался. В конечном итоге мои сомнения подтвердились. Я боялся, что многие будут спрашивать, нет ли в Google более подходящего человека для этой работы, – и они спрашивали. Или толпы людей захотят участвовать в работе над книгой – и это тоже сбылось. Но главная причина была в том, что все мои предыдущие книги я писал для новичков. И серия «How to Break…», и моя книга «Exploratory Testing»[3] содержат законченные, цельные истории, у которых есть начало и конец. С этой книгой все иначе. Читатели, конечно, могут «проглотить» ее за один присест, но она задумана как справочник по решению реальных задач в Google, как больших, так и малых, из которых складывается наша практика тестирования.

Я думаю, что людям с опытом тестирования ПО в IT-компаниях книга принесет больше пользы, чем новичкам, потому что они могут сравнить процессы в Google с процессами, к которым они привыкли. Я представляю себе, как матерые тестировщики, менеджеры и директора берут книгу, находят интересующий их раздел и читают его, чтобы узнать, как нужная задача решена в Google. Но я-то привык к другому способу подачи материала!

В работе над книгой ко мне присоединились два соавтора, которые раньше не публиковались. Оба – первоклассные специалисты, проработавшие в Google дольше меня. Джейсон Арбон работает инженером по тестированию, хотя по сути он прирожденный организатор. Он здорово повлиял на реализацию многих идей и технических средств, описанных в главах этой книги. Пересечение наших карьерных путей изменило нас обоих.

Джефф Каролло – тестировщик, перешедший в разработчики. Безусловно, это лучший тестировщик-разработчик, которого я когда-либо встречал. Джефф один из немногих людей, которым удалось успешно организовать «полнейшую автоматизацию» (я имею в виду такую, которую можно запустить и забыть), – код тестов написан так хорошо и настолько автономен, что не требует участия автора. Они оба великолепные соавторы, и мы постарались, чтобы наши голоса в книге звучали в унисон.

Многие мои коллеги по Google помогли с материалом для книги. У каждой статьи-отступления есть свой автор. Мы подготовили для вас интервью с ключевыми сотрудниками Google, повлиявшими на организацию тестирования. Мы решили, что проще включить мнения людей, определивших процесс тестирования Google, прямо в книгу, чем перечислять тридцать соавторов. Конечно, не каждое интервью будет интересно всем, но они четко выделены в тексте, так что вы можете выбирать, читать или пролистывать заметку. Мы от всей души благодарим этих участников и заявляем: если нам где-то не удалось воздать должное их работе, то вся вина за это лежит на нас. Английский язык меркнет перед описанием их блестящей гениальности.

Приятного чтения, отличного тестирования – и желаем вам всегда находить (и исправлять) те баги, которые вы ищете!

Джеймс Уиттакер
Джейсон Арбон
Джефф Каролло
Киркленд, штат Вашингтон

[3] Предыдущие книги Джеймса Уиттакера по тестированию «How to Break Software: A Practical Guide to Testing» («Как сломать программу: практическое пособие по тестированию») и «Exploratory Software Testing: Tips, Tricks, Tours, and Techniques to Guide Test Design» («Исследовательское тестирование: советы, хитрости, туры и техники тест-дизайна») на русский язык не переводились. – Примеч. перев.

Благодарности

Мы благодарим всех технических специалистов Google, неустанно улучшающих качество продуктов. Мы восхищены открытой и распределенной культурой управления и организации технической работы в Google, которая позволила нам свободно исследовать и экспериментировать по ходу формирования практики и методологии тестирования.

Мы особо выделим следующих ребят, которые тратили свои силы и даже рисковали, чтобы привести тестирование к желаемому виду: Алексис О. Торрес, Джо Мухарски, Даниэль Дрю, Ричард Бустаманте, По Ху, Джим Ридон, Теджас Шах, Джули Ральф, Эриэл Томас, Джо Михаил и Ибрагим Эль Фар.

Спасибо нашим редакторам, Крису Гузиковски и Крису Зану, стоически переносившим наш технический жаргон.

Спасибо собеседникам, поделившимся своими взглядами и опытом в интервью: Анкиту Мехта, Джоэлу Хиноски, Линдсей Уэбстер, Эппл Чоу, Марку Стрибеку, Нилу Норвицу, Трейси Бялик, Руссу Руферу, Теду Мао, Шелтону Мару, Ашишу Кумару, Суджай Сани, Брэду Грину, Саймону Стюарту и Хунг Дан.

Отдельное спасибо Альберто Савоя, убедившему нас в преимуществах метода быстрого создания прототипа и итераций. Спасибо Google, персоналу кафетерия и шеф-поварам за отличную еду и кофе. Спасибо за благожелательные отзывы Филу Валигоре, Алану Пейджу и Майклу Бахману. И наконец, спасибо Пату Коупленду, который нас всех собрал и обеспечил финансирование такой энергичной и талантливой группы специалистов в области качества.

Об авторах

Джеймс Уиттакер – технический директор Google[4], ответственный за тестирование Chrome, Maps и веб-приложений Google. Перешел в компанию из Microsoft, имеет профессорскую степень. Джеймс – один из самых известных в мире специалистов в области тестирования.

Джейсон Арбон – инженер по тестированию в Google, ответственный за Google Desktop, Chrome и Chrome OS. Он выступал в роли руководителя разработки многих тестовых инструментов с открытым кодом и внутренних экспериментов по персонализации. До прихода в Google работал в Microsoft.

Джефф Каролло — разработчик в тестировании, ответственный за тестирование Google Voice, Toolbar, Chrome и Chrome OS. Он консультировал десятки групп разработки Google, помогая им повышать качество кода. В 2010 году перешел в разработчики, сейчас руководит разработкой Google+ API. До прихода в Google тоже работал в Microsoft.

[4] В начале 2012 года Джеймс Уиттакер перешел из Google в Microsoft. – Примеч. перев.

Глава 1. Первое знакомство с организацией тестирования в Google

Есть один вопрос, который я слышу чаще других. В какой бы стране я ни был, на какой бы конференции ни выступал, этот вопрос обязательно всплывает. Даже наши новобранцы задают его сразу же после прохождения вводного курса: «*Так как же Google тестирует ПО?*».

Не знаю, сколько раз я уже отвечал на этот вопрос и как много разных вариантов ответа дал. Мои ответы постоянно меняются – чем дольше я работаю в Google, тем больше узнаю всевозможных тонкостей нашего подхода к тестированию. Меня давно посещали мысли, что стоит зафиксировать свои знания на бумаге. Когда Альберто (который часто грозит переработать все книжки по тестированию в подгузники для взрослых, чтобы они принесли хоть какую-то пользу) предложил мне написать книгу, я понял, что никуда от этого не денусь.

И все-таки я хотел подождать. Во-первых, я не считал себя самым подходящим автором. Было много людей, которые работали в Google намного дольше меня, и я хотел дать им возможность первыми написать о своем опыте. Во-вторых, я был директором по тестированию Chrome и Chrome OS (кстати, сейчас эту должность занимает один из моих бывших подчиненных), поэтому видел организацию тестирования в Google только с одной стороны. Мне нужно было узнать еще много всего о тестировании в Google.

В Google тестирование ПО – часть централизованной системы, которую мы называем направлением продуктивности разработки. Она охватывает инструменты для разработки, тестирования (от юнит-уровня до исследовательского) и организации процессов выпуска программных продуктов. Мы создаем и поддерживаем множество общих инструментов и тестовую инфраструктуру для веб-проектов поиска, рекламной платформы, мобильных приложений, видеосервисов. Короче, для всего, что Google делает в интернете. В Google решены проблемы производительности и масштабирования, что позволяет нам, несмотря на огромные размеры компании, выпускать программы со скоростью стартапа. Как верно заметил Патрик Коупленд в предисловии к книге, этой магией мы во многом обязаны именно команде тестирования.

На заметку

В Google тестирование ПО – часть централизованной системы, которую мы называем направлением продуктивности разработки.

В декабре 2010-го мы выпустили Chrome OS, и я передал руководство проектом одному из своих подчиненных, а сам погрузился в работу с другими продуктами. Тогда и началась эта книга. Мой первый пост в блоге «Как в Google тестируется ПО»[5] стал первой ласточкой, и с тех пор все закрутилось. Через полгода книга была готова, и я пожалел, что не начал писать ее раньше. За эти шесть месяцев я узнал о тестировании в Google больше, чем за два предыдущих года, а для новичков Google моя книга стала частью вводного курса.

Эта не единственная книга о том, как большие компании тестируют ПО. Я работал в Microsoft, когда Алан Пейдж, Би-Джей Роллисон и Кен Джонстон написали книгу «How We Test Software at Microsoft», и сам применял все методы, описанные в книге. Компания Microsoft тогда была лидером в тестировании. Она подняла тестирование на один уровень с разработкой. Тестировщики Microsoft были самыми востребованными спикерами конференций. Первый директор по тестированию, Роджер Шерман, привлекал в Редмонд талантливых ребят со способностями к тестированию со всего мира. Это был золотой век тестирования.

Компания выпустила большую книгу, чтобы описать свой опыт.

[5] http://googletesting.blogspot.com/2011/01/how-google-tests-software.html

Я не успел поработать над этой книгой, так как пришел в Microsoft поздно, но мне выпал второй шанс. Я попал в Google в тот самый момент, когда тестирование здесь было на подъеме. Команда направления продуктивности разработки стремительно выросла с пары сотен до 1200 человек. Проблемы роста, о которых Патрик говорит в своем предисловии, мы уже преодолели, и наше подразделение было на пике своего развития. Блог тестирования Google ежемесячно собирал сотни тысяч просмотров, а конференция GTAC[6] стала ведущей в индустрии тестирования ПО. Вскоре после моего прихода Патрика повысили, и теперь ему подчинялось около дюжины директоров и технических менеджеров. Если у тестирования ПО и была вторая волна развития, то центром этой стихии был Google.

Это значит, что история тестирования в Google тоже заслуживала своей большой книги. Но вот незадача – я не пишу больших книг. Да и Google славится своим простым и лаконичным подходом к разработке. Надеюсь, что моя книга получилась такой же.

Книга рассказывает, что значит быть тестировщиком в Google, и описывает, как мы решаем проблемы, связанные с масштабом, сложностью и массовым использованием наших проектов. Здесь вы найдете информацию, которой больше нигде нет, но если и этого не хватит, чтобы удовлетворить ваше жгучее любопытство к процессу тестирования, то в сети можно узнать еще больше – просто погуглите.

Есть еще кое-что, и я должен это сказать. Скорее всего, организацию тестирования «как в Google» будут перенимать многие компании по мере того, как все больше программных продуктов переходит с десктопов в веб. Если вы читали книгу от Microsoft, то не думайте, что найдете здесь много общего с ней. У обеих книг по три автора, в каждой книге описана организация тестирования в крупной компании – разработчике ПО. На этом общее заканчивается. Трудно найти два подхода к тестированию, которые различались бы сильнее.

> **На заметку**
>
> Скорее всего, организацию тестирования «как в Google» будут перенимать многие компании по мере того, как все больше программных продуктов переходит с десктопов в веб.

Патрик Коупленд в своем предисловии описал, как зарождалась методология Google. Она и сейчас гармонично развивается вместе с компанией. Для специалистов, которые приходят из других компаний, Google становится местом, где неэффективные методы переплавляются в полезные. Чем больше появлялось тестировщиков, тем больше новых идей и приемов мы пробовали. Балласт мы отбросили, а кое-что смогли перековать в прочный металл, ставший частью остова Google. Наши тестировщики готовы пробовать что угодно, но способны легко отказываться от неэффективных стратегий.

В основе Google лежат скорость работы и инновации. Мы выпускаем код именно в тот момент, когда он действительно востребован, а недовольных пользователей мало. Мы плотно взаимодействуем с первыми испытателями продукта, чтобы собирать как можно больше обратной связи. Тестирование в таких условиях тоже должно быть стремительным, а методы, требующие дотошного предварительного планирования или постоянного сопровождения, не выживут. В какой-то момент тестирование настолько тесно переплетается с разработкой, что эти две дисциплины невозможно отделить одну от другой. А иногда тестирование происходит настолько независимо, что разработчики даже ничего не подозревают.

> **На заметку**
>
> В какой-то момент тестирование настолько тесно переплетается с разработкой, что эти две дисциплины невозможно отделить одну от другой.

[6] GTAC – Конференция Google по автоматизации тестирования (Google Test Automation Conference, www.GTAc.biz).

А иногда тестирование происходит настолько независимо, что разработчики даже ничего не подозревают.

С ростом Google стремительный темп работы замедлился совсем чуть-чуть. Нам потребовалось не больше года, чтобы создать операционную систему, клиентские приложения (типа Chrome) появляются ежемесячно, а уж веб-приложения меняются ежедневно. Хотя мы уже давно перестали быть стартапом, мы сохранили свойственные тому времени темпы. В таких условиях проще описать, каким тестирование не должно быть: догматичным, трудоемким, сложно выполнимым и занимающим много времени, – чем пытаться рассказать, каким оно должно быть. Хотя мы пытаемся нашей книгой сделать именно это. Одно можно сказать точно: тестирование не должно ставить палки в колеса инновациям и разработке. Второго шанса у него не будет.

Успех Google в тестировании нельзя списать на маленькое количество тестируемых продуктов. Размер и сложность тестирования Google не уступают задачам других компаний. Google выпускает почти все типы программ – от клиентских операционных систем до веб-, мобильных, корпоративных, коммерческих и социальных приложений. Наши приложения масштабны и сложны, ими пользуются сотни миллионов людей, мы постоянно «на мышке» хакеров. Значительная часть нашего исходного кода открыта. При этом у нас много устаревшего кода. Нас часто проверяют на соответствие требованиям законодательства. Наши продукты работают в сотнях стран и на многих языках. Пользователи ждут, что продукты Google окажутся простыми в использовании и будут «просто работать». То, чем тестировщики Google занимаются каждый день, нельзя назвать легкими задачами. Тестировщики Google сталкиваются со всеми сложными проблемами, какие только можно представить.

Мы готовы обсуждать, насколько идеально (на самом деле нет) организован процесс Google. Но в одном я точно уверен: наш подход к тестированию отличается от остальных. Вслед за стремительным переходом программ с десктопов в облака вполне реально, что подход «как в Google» будет набирать популярность в отрасли. Мои соавторы и я надеемся, что книга раскроет магическую формулу Google и положит начало обсуждению того, как отрасль должна решать задачу создания надежного ПО, на которое сможет положиться весь мир. Возможно, у методологии Google есть свои недостатки, но мы все равно хотим опубликовать ее и сделать доступной международному сообществу тестировщиков, чтобы она могла развиваться и совершенствоваться.

Подход Google на первый взгляд парадоксален: у нас во всей компании меньше выделенных тестировщиков, чем у многих наших конкурентов в одной команде разработки. Мы не делаем ставку на миллионную армию рядовых тестировщиков. Мы – небольшой отряд элитного спецназа, успех которого зависит от превосходной тактики и современного вооружения. Как и в случае со спецназом, наш секретный ингредиент – ограниченность ресурсов. Мы следуем завету Ларри Пейджа, который сказал, что «дефицит приносит ясность», и это заставляет нас правильно расставлять приоритеты. Для всего, от описания функциональности до техник тестирования, у нас есть высокоэффективные приемы достижения поставленной цели – качества продукта. Дефицит заставляет ценить ресурсы тестирования и относиться к ним с уважением. Это помогает людям оставаться вовлеченными в развитие нашей отрасли. Когда меня спрашивают, в чем секрет нашего успеха, я всегда даю совет: «Не нанимайте слишком много тестировщиков».

На заметку

Когда меня спрашивают, в чем секрет нашего успеха, я всегда даю совет: «Не нанимайте слишком много тестировщиков».

Как Google справляется с таким маленьким штатом тестировщиков? Если бы мне нужно было ответить просто, я бы сказал, что в Google вся ответственность за качество лежит на

плечах тех, кто пишет код. Качество никогда не бывает проблемой «какого-то тестировщика». Каждый, кто пишет код в Google, – уже немного тестировщик, а качество – это проблема всего коллектива (рис. 1.1). Говорить о соотношении численности «разработчики/тестировщики» в Google – то же самое, что рассуждать о чистоте воздуха на поверхности Солнца. Эту тему бессмысленно поднимать. Каждый инженер является *и* тестировщиком. Если в его должности есть слово «тестирование», то он помогает другим специалистам проводить качественное тестирование.

Рис. 1.1. *Программисты Google предпочитают качество широте функциональности*

Мы выпускаем первоклассные программные продукты. Это свидетельствует о том, что наша формула достойна внимания. Возможно, какие-то ее части сработают и в других компаниях. Конечно, что-то может быть усовершенствовано. Ниже я привожу краткое описание нашей формулы. В последующих главах мы подробно проанализируем ее и разберемся, как же организовать процесс тестирования в царстве разработчиков.

Качество ≠ Тестирование

Фраза «тестирование не определяет качество» уже настолько избита, что просто обязана быть правдой. В любой области разработки, от автомобилей до софта, не получится отточить продукт до совершенства, если он изначально неправильно сконструирован. Спросите любого автопроизводителя, который хоть раз отзывал партию своих машин, чего стоят запоздалые попытки прикрутить качество на готовое изделие. Делайте все правильно с самого начала, не создавайте себе лишних трудностей.

Тем не менее это только звучит просто.

С одной стороны, качество и не создается тестированием, с другой – без тестирования сделать качественный продукт невозможно. Как вы узнаете, насколько качественный ваш продукт, не проводя тестирование?

Эта головоломка решается легко: перестаньте рассматривать разработку и тестирование по отдельности. Тестирование и разработка идут рука об руку. Напишите немного кода и протестируйте его. Затем напишите еще чуть-чуть и снова протестируйте. Тестирование не отдельная практика, это часть самого процесса разработки. Одним только тестированием качества не добиться. Рецепт получения высокого качества: смешивайте разработку и тестирование в блендере, пока они не станут единой субстанцией.

> **На заметку**
>
> Одним только тестированием качества не добиться. Рецепт получения высокого качества: смешивайте разработку и тестирование в блендере, пока они не станут единой субстанцией.

Вот чего мы добивались: мы хотели объединить дисциплины разработки и тестирования, чтобы одной нельзя было заниматься без другой. Немножко программируем, потом тестируем. Еще немного пишем код и снова тестируем. Здесь важно, *кто* тестирует. Тестировщиков в классическом понимании у нас мало. Значит, кто должен тестировать? Разработчик. Кто лучше всего протестирует свой код? Тот, кто его написал. Кто больше всех заинтересован в коде без багов? Правильно. Вот почему Google обходится таким небольшим числом тестировщиков – за качество отвечают разработчики. Если продукт сломается после выпуска, то шишки полетят в разработчика, создавшего проблему, а не в тестировщика, который ее не нашел.

В итоге качество достигается предотвращением, а не выявлением багов. Качество – часть разработки, а не тестирования. Интегрируя тестирование в разработку, мы создали поэтапный процесс, в котором можно легко откатить изменение, в котором слишком много багов. Так мы предотвращаем возникновение проблем у пользователей и сокращаем количество выделенных тестировщиков, которые выявляют баги, ставящие под угрозу выпуск продукта. В Google тестирование определяет, насколько хорошо мы предотвращаем ошибки.

Наш подход к созданию продуктов подразумевает слияние разработки и тестирования. Это подтверждают комментарии к коду типа «где твои тесты, чувак?», развешенные в туалетах плакаты с правилами тестирования[7] и многое другое. Тестирование должно стать неотделимым от разработки. Качество родится только тогда, когда разработка и тестирование начнут жить вместе.

> **На заметку**
>
> Тестирование должно стать неотделимым от разработки. Качество родится только тогда, когда разработка и тестирование начнут жить вместе.

[7] http://googletesting.blogspot.com/2007/01/introducing-testing-on-toilet.html

Роли

Чтобы девиз «Сам построил, сам и ломай» работал (и работал долго), вам нужна не только традиционная роль разработчика кода, но и другие роли. Конкретнее, должны появиться инженеры, которые помогут разработчикам тестировать эффективно и правильно. Они часто скромничают, называя себя *тестировщиками*, но на самом деле обеспечивать продуктивность – вот их основная задача. Тестировщики нужны для того, чтобы разработчики работали более продуктивно. Причем рост продуктивности основан на предотвращении появления ошибок из-за небрежной разработки. Так качество становится частью этой продуктивности. Все эти роли мы подробно рассмотрим в следующих главах, а пока обойдемся кратким описанием.

Роль **разработчика** (Software Engineer, SWE) всем знакома и привычна. Он пишет код функциональности приложений, который поставляется пользователям. Он создает проектную документацию, определяет структуры данных и общую архитектуру, а большую часть времени пишет код и проводит код-ревью[8]. Разработчик пишет много тестового кода, например во время написания тестов для TDD и юнит-тестирования, и, как будет показано дальше в этой главе, участвует в создании малых, средних и больших тестов. Разработчик отвечает за качество всего кода, к которому он прикасается: пишет, исправляет или вносит изменения. Да, все верно: если разработчик должен изменить функцию и его изменение нарушает существующий тест или требует написания нового, он должен написать этот тест. Практически 100 % рабочего времени разработчик пишет программный код.

Роль **разработчика в тестировании** (Software Engineer in Test, SET) тоже связана с разработкой, но фокусируется на тестируемости кода и создании инфраструктуры тестирования. Разработчик в тестировании анализирует архитектуру, уделяет особое внимание качеству кода и рискам проекта. Он выполняет рефакторинг, чтобы сделать код тестируемым, пишет фреймворки юнит-тестирования и средства автоматизации. Разработчик в тестировании работает с тем же кодом, что и разработчик, но больше заинтересован в улучшении качества и тестового покрытия, чем в добавлении новых фич или повышении производительности. Разработчик в тестировании тоже проводит почти 100 % своего времени за написанием кода, но делает это ради повышения качества, а не для реализации фич для пользователей.

> **На заметку**
>
> Разработчик в тестировании работает с тем же кодом, что и разработчик, но занимается скорее повышением качества и тестовым покрытием, чем добавлением новых фич или повышением производительности. Разработчик в тестировании пишет код, который помогает разработчику тестировать написанные им фичи.

Роль **инженера по тестированию** (Test Engineer, TE) связана с ролью разработчика в тестировании, но здесь на первом месте находятся пользователи и только на втором – разработчики. Некоторые инженеры по тестированию в Google пишут много кода для автотестов и управления сценариями использования и даже для имитации действий пользователя. Кроме того, именно они организуют работу по тестированию, которую выполняют другие инженеры, управляют выполнением тестов и интерпретируют их результаты тестов. Особенно важна их работа на поздних стадиях проекта, когда напряжение с приближением выпуска растет. Инже-

[8] Код-ревью (или рецензирование кода) – систематическая проверка исходного кода с целью обнаружения и исправления ошибок, допущенные при его написании. Эта практика позволяет находить ошибки раньше и способствует улучшению общего качества кода. – Примеч. перев.

неры по тестированию – это эксперты продукта, консультанты по качеству и специалисты по анализу рисков. Одни пишут много кода, другие – мало[9].

На заметку

Инженеры по тестированию фокусируются на тестировании с точки зрения пользователя. Они занимаются общей организацией контроля качества, управляют выполнением тестов, интерпретируют их результаты и строят сквозную автоматизацию тестирования.

Итак, с точки зрения качества разработчики отвечают за фичи приложения и их качество отдельно от всего остального. Они ответственны за то, чтобы архитектура была устойчивой к ошибкам, приложение восстанавливалось после сбоев, отвечают за TDD, юнит-тесты и вместе с разработчиками в тестировании пишут код для тестирования фич.

Разработчики в тестировании отвечают за фичи тестирования. Они настраивают среду для изолированного выполнения кода с помощью имитации реального рабочего окружения. Для этого они создают такие компоненты, как *заглушки* (stubs), *подставные объекты* (mocks) и *имитации* (fakes), – мы рассмотрим их позже. Настроить очередь для управления отправкой кода в репозиторий – тоже их задача. Другими словами, разработчики в тестировании пишут код, который помогает разработчикам тестировать написанные ими фичи. Большую часть тестирования выполнит сам разработчик. Разработчики в тестировании в основном следят за тем, чтобы функциональность можно было легко протестировать, а разработчики не ленились писать тест-кейсы.

Получается, что разработчики в тестировании работают для разработчиков. Их конечная цель – качество кода и отдельных фич. Для этого они создают разработчикам удобную среду для тестирования кода. О пользователях думают тестировщики.

Если разработчики провели модульное и функциональное тестирование хорошо, то следующая задача – понять, насколько хорошо их исходный код вместе с данными будет работать на благо пользователей. Тестировщики проверяют добросовестность разработчиков дважды. Любые очевидные баги свидетельствуют о том, что тестирование раннего цикла было недостаточным или небрежным. Если таких багов мало, тестировщик переходит к своей основной задаче – убедиться в том, что продукт выполняет пользовательские сценарии, соответствует ожиданиям по производительности, что он надежен, правильно локализован и т. д. Тестировщики много тестируют сами и вдобавок обеспечивают координацию с другими инженерами, внешними тестировщиками-подрядчиками, тестировщиками из сообщества, внутренними пользователями, бета– и ранними внешними пользователями. Они сводят воедино все риски, которые могут сыграть из-за недочетов в базовой архитектуре, сложности функциональности и отказов в системе предотвращения сбоев. Как только тестировщики взялись за дело, их работе не видно конца.

[9] Везде, где мы дальше пишем «тестировщики», мы имеем в виду именно роль инженера по тестированию. – Примеч. перев.

Организационная структура

В большинстве компаний, где я работал, разработчики и тестировщики относились к одной команде разработки продукта. И разработчики, и тестировщики подчинялись одному и тому же руководителю проекта. Один продукт, одна команда – все участники говорят на одном языке.

К сожалению, я еще не видел, чтобы такой подход работал на практике. Обычно командой руководят менеджеры, вышедшие из среды программирования или управления, но не из среды тестировщиков. В предрелизном аврале они предпочтут выпустить функциональность полностью и закрыть как можно больше задач по «отделке» приложения, чем позаботиться об общем качестве. Внутри команды тестированию часто отводится второстепенная роль по отношению к разработке. Это отлично заметно по обилию глючных продуктов и преждевременных выпусков в нашей отрасли. Кто тут заказывал сервис-пак?

> **На заметку**
>
> Совместными командами обычно руководят менеджеры, вышедшие из среды программирования или управления, но не из среды тестировщиков. В предрелизном аврале они предпочитают выпустить функциональность полностью и закрыть как можно больше задач по «отделке», чем позаботиться об общем качестве. Внутри команды тестированию часто отводится второстепенная роль по отношению к разработке.

Давайте разберемся, кто кому подчиняется в Google. У нас есть своя иерархия, которая делится на направления, называемые Focus Area (FA). У нас есть направления Client (проекты Chrome, Google Toolbar и т. д.), Geo (проекты Maps, Google Earth и т. д.), Ads, Apps, Mobile и т. д. Все разработчики подчиняются директору или вице-президенту направления.

Правда, тестирование ломает эту схему. Тестирование находится в отдельном горизонтальном направлении, которое отвечает за продуктивность разработки. Направление продуктивности разработки существует параллельно с продуктовыми направлениями. Тестировщиков, по сути, предоставляют командам разработки продуктов во временное пользование. Они свободно поднимают вопросы безопасности и освещают области, где нет тестов или много багов. Так как мы не подчиняемся команде разработки продукта, нам нельзя ответить, что для нас есть дела поважнее. Мы сами устанавливаем свои приоритеты, и они всегда в рамках обеспечения надежности, безопасности и всего такого, пока мы не решим, что продукту нужно что-то другое. Если команда разработки захочет провести тестирование по упрощенной схеме, это необходимо согласовать заранее, а мы всегда можем сказать «нет».

С такой структурой работы мы можем сохранять небольшой штат тестировщиков. Команда разработки продукта не может снизить техническую планку тестировщиков или нанять их больше, чтобы свалить на них рутинную работу. Рутинная работа вокруг фичи входит в обязанности разработчика, ее нельзя переложить на какого-нибудь тестировщика-несчастливчика.

Тестировщиков на проекты назначают руководители направления продуктивности разработки. Они принимают стратегические решения, опираясь на приоритеты компании, сложность продукта и потребности конкретной команды проекта в сравнении с другими. Конечно, они могут ошибаться, и такое случается. Но в целом такая схема обеспечивает баланс ресурсов между фактическими, а не предполагаемыми потребностями.

> **На заметку**
>
> Тестировщиков на проект назначают руководители направления продуктивности разработки. Они принимают стратегические решения,

опираясь на приоритеты компании, сложность продукта и потребности конкретной команды проекта в сравнении с другими. Такая схема сохраняет баланс ресурсов между фактическими, а не предполагаемыми потребностями. Централизованность подходов, идей и решений помогает всей компании использовать самые удачные из них.

Статус временно выделяемых на проект тестировщиков упрощает их перемещение по проектам. Они не только сохраняют свежий взгляд и остаются в курсе дел, но и быстро распространяют удачные идеи по компании. Прием или инструмент тестирования, который хорошо сработал в продукте направления Geo, скорее всего, пойдет в работу снова, когда этот тестировщик перейдет на Chrome. Нет более быстрого способа распространения инноваций, чем перемещение самих новаторов.

Конец ознакомительного фрагмента.