

Джанет Грегори, Лайза Криспин

Agile-тестирование. Обучающий курс для всей команды

Научный редактор Сергей Виноградов

Издано с разрешения Pearson Education, Inc. (Addison-Wesley Professional)

Благодарим за консультации Илью Таджибаева

Все права защищены.

Никакая часть данной книги не может быть воспроизведена в какой бы то ни было форме без письменного разрешения владельцев авторских прав.

Authorized translation from the English language edition, entitled MORE AGILE TESTING: LEARNING JOURNEYS FOR THE WHOLE TEAM, 1st Edition; ISBN: 0321967054; by GREGORY, JANET and by CRISPIN, LISA; published by Pearson Education, Inc., publishing as Addison-Wesley Professional.

© 2015 by Lisa Crispin and Janet Gregory. All rights reserved. No part of this book may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording or by any information storage retrieval system, without permission from Pearson Education, Inc. RUSSIAN language edition published by MANN, IVANOV, and FERBER PUBLISHERS. © 2019.

© Перевод на русский язык, издание на русском языке, оформление. ООО «Манн, Иванов и Фербер», 2019 * * *

Моим внукам Лорен, Брейдену и Джо, которые радовали меня весь год. Джанет

Моей семье. Тем, кто со мной, и тем, кого уже, к сожалению, нет с нами. Моим дорогим друзьям, которые являются частью той семьи, что я выбрала сама. Лайза

Предисловие Элизабет Хендриксон

Еще десять лет назад Agile считался чем-то радикальным. Маргинальным. Станным. Стандартный подход к разработке программного обеспечения включал анализ, дизайн, написание кода и тестирование. Интеграция и тестирование проводились на заключительном этапе. Полный цикл разработки занимал месяцы, даже годы.

Если вы никогда не работали в компании с долгими циклами и прерывными фазами, это покажется немного странным, но десять лет назад стандарты были такими. Когда Agile только зарождался, он был направлен в основном на программистов. Джанет, Лайза и еще несколько специалистов по контролю/обеспечению качества (QA) и тестированию были в их числе. В то же время многие последователи Agile не чувствовали необходимости в QA. Конечно, они ошибались. QA изменился, подстраиваясь под новые сферы деятельности, но никуда не исчез.

Такие люди, как Джанет и Лайза, показали, как можно встроить QA в Agile, вместо того чтобы игнорировать его. В своей первой книге[1] Джанет и Лайза рассказали о корпоративных изменениях, необходимых для полной интеграции тестирования и разработки, научили преодолевать трудности. Рекомендую эту книгу.

Однако остались вопросы. Как эти методы применять в различных сферах? С чего начать? Что должен знать тестировщик, чтобы быть наиболее эффективным? Эта книга начинается с того, на чем заканчивается первая, и отвечает на вышеназванные и многие другие вопросы. И даже если бы в книге содержались только эти ответы, она уже была бы замечательным продолжением. Но она намного больше. Здесь вы столкнетесь с темой, которую Джанет и Лайза так искусно вплели в текст, что ее можно даже не заметить. Я же хочу обратить ваше внимание на то, что эта книга об адаптации.

Обдумывайте и адаптируйте. Этот простой способ поможет вашей компании прийти к Agile. Экспериментируйте, пробуйте разные подходы, фильтруйте полученные знания, повторяйте. Так вы почувствуете, что ваша организация подвижна и гибка, способна двигаться в ногу с запросами рынка и постепенно удовлетворять их. Эта книга учит адаптировать, несмотря на то что она в общем-то об Agile-тестировании.

Часть вторая «Обучение для улучшения тестирования» касается не только того, как вы учитесь сами, но и того, как построить учебную среду.

Часть седьмая «Разные сферы деятельности» не просто рассказывает о разных вариантах Agile, приспособленных под различные условия, а является своего рода путеводителем по разного рода адаптациям.

Мир невероятно быстро меняется. Еще десять лет назад Agile был чем-то странным, теперь это мейнстрим. Планшеты типа iPad теперь повсюду, а несколько лет назад никто не знал об их существовании. Методы, инструменты, технологии меняются наравне с рынком так быстро, что за ними трудно успевать. Уже недостаточно уметь что-то делать одним способом, необходимо открывать новые пути. Эта книга – фантастический источник для Agile-тестирования. Она научит адаптироваться и чувствовать себя комфортно в процессе изменений.

Надеюсь, она понравится вам так же, как мне.

Элизабет Хендриксон,

основатель и президент Quality Tree Software Inc.

Предисловие Джоанны Ротман

Что делают тестировщики? Они представляют информацию о продукте на основании тестов, и это позволяет определить задачи для команды.

Именно это Джанет Грегори и Лайза Криспин сделали в своей книге. У вас проблемы с гибкостью? Существует множество способов, которые помогут понять ценность совместной работы, создания и обучения коллектива и вашу роль в процессе тестирования.

Не знаете, как выстроить тестирование конкретного продукта, коллектива или программы? Здесь вы найдете ответ.

Как вы взаимодействуете с коллегами за офисной перегородкой? А с теми, кто находится на другом конце мира? Джанет и Лайза сталкивались с этим и знают о проблеме не понаслышке. Они делают акцент на ролях, а не на должностях. И это особенно ценно.

В книге много изображений, так что вам не придется постоянно думать, что авторы имели в виду. Они не просто рассказывают – они показывают.

«Agile-тестирование. Обучающий курс для всей команды» – больше, чем книга о тестировании. Она о том, как с помощью тестирования выстроить работу команды, отдела, организации и обеспечить наиболее эффективный переход к Agile. Не в этом ли и заключается суть представления информации об организации на основе тестов, которые выявляют существующие проблемы?

Тестировщикам или руководителям команды тестировщиков нужна эта книга. Если вы внедряете тестирование в свою организацию, она также пригодится. Иначе как вы узнаете, что должен делать тестировщик?

Джоанна Ротман,

консультант по вопросам управления

Пролог

Эта книга – продолжение темы, о которой мы говорили в предыдущей книге. Мы не будем повторяться, но выстроим контекст так, чтобы все было понятно, даже если вы не читали ее. Мы ссылаемся на «Гибкое тестирование», когда нам кажется, что читателю необходимо получить детальное представление об основных определениях. **ДЛЯ КОГО ЭТА КНИГА**

Мы предполагаем, что вы, читатель, не новичок в мире Agile-тестирования, что у вас уже есть определенный опыт работы с Agile и опыт тестирования, и теперь вам нужна помощь в областях за их пределами. Если требуется введение в развитие Agile и в основы Agile-тестирования, перед прочтением этой книги рекомендуем ознакомиться с работой Расмуссона «Гибкое управление IT-проектами. Руководство для настоящих самураев»[2].

Эта книга для всех, кто интересуется процессами тестирования в Agile-командах. Судя по нашему опыту, это не только тестировщики и руководители команд, но и программисты, заказчики, бизнес-аналитики, DevOps-специалисты, руководители направлений, – одним словом, почти все. **ПРИЕМОЧНОЕ ТЕСТИРОВАНИЕ**

В дополнение к тому, чем мы делимся с вами, о чем узнали за последние несколько лет, хотим, чтобы эта книга была полезна читателям. Нам хотелось знать, какие вопросы остались

у прочитавших первую книгу. Поэтому мы попросили всех пользователей в нашей емейл-рассылке пройти своеобразное «приемочное тестирование». Мы отфильтровали ответы и приложили все усилия, чтобы раскрыть суть возникших вопросов во второй книге.

Вы заметите, что мы используем разработку через поведение (Behaviour Driven Development, BDD), о которой подробнее поговорим в главе 11.

Например, <исходные данные>,,

Когда <триггер, действие>,,

Я должен <ожидаемый результат>.

- Я Agile-тестировщик или руководитель. Когда нанимаю новых тестировщиков без опыта работы по системе Agile, я должен знать, как ввести их в работу, не бросая за борт без спасательного жилета.
- Я работаю в команде Agile. Когда прочту эту книгу, я должен узнать как подходить к исследовательскому тестированию с автоматизированными тестами и получать общую картину покрытия тестами без использования сложных инструментов.
- Я опытный руководитель команды тестировщиков Agile. Когда прочитаю эту книгу, должен понять, как использовать Agile-тестирование в разных командах, чтобы моя компания могла развиваться.
- Я опытный руководитель команды тестировщиков. Прочитав эту книгу, должен понять, как координировать автоматизированные процессы тестирования на разных этапах с различными командами, как их улучшить.
- Я опытный руководитель команды тестировщиков Agile. Когда прочту эту книгу, должен понять, как другие команды адаптировали методы Agile-тестирования для различных областей, и получить представление, как применить это в своей сфере.
- Я работаю в команде Agile и интересуюсь тестированием. Закончив читать эту книгу, я пойму, какими должны быть тесты, а какими нет, и как эффективно организовать тестирование.
- Я опытный Agile-тестировщик. Наткнувшись в книге на интересную тему, я хочу получить ссылки на онлайн-ресурсы и другие книги.
- Я опытный Agile-тренер или руководитель. Если в книге упоминается идея, полезная для моей команды, то должно быть достаточно информации, чтобы разработать стратегию и убедить коллектив попробовать ее в деле.
- Я работаю в команде Agile, и мои задачи касаются тестирования и информирования клиентов. Когда прочту эту книгу, я должен понять, как говорить с командой заказчика о процессах тестирования.
- Я опытный руководитель команды тестировщиков Agile. Прочитав эту книгу, хочу знать о наиболее распространенных методах внедрения Agile, понимать рабочую среду тестировщиков из других компаний, если они претендуют на место в моей команде.

(Важно: это приемочное тестирование – не часть книги, но мы думаем, что некоторые

примеры и истории, рассказанные здесь, помогут добиться результатов.) КАК ЧИТАТЬ ЭТУ КНИГУ

Мы распределили информацию так, как считаем удобным. Тем не менее вам не обязательно читать книгу с первой главы. Вы можете начать с любой наиболее интересной темы. Мы стараемся рассказать лишь раз о каждой, но так как многие методы и принципы пересекаются, вы обнаружите похожие идеи в разных главах.

ЧАСТЬ ПЕРВАЯ. ВВЕДЕНИЕ

Прочтите эту часть, чтобы понять, как появилось Agile-тестирование, как оно превратилось в краеугольный камень развития системы Agile и продолжение разработки продуктов. Отчасти успешное внедрение Agile зависит от способности компании определять наиболее важные задачи для успешного Agile-тестирования в долгосрочной перспективе.

ЧАСТЬ ВТОРАЯ. ОБУЧЕНИЕ ДЛЯ УЛУЧШЕНИЯ ТЕСТИРОВАНИЯ

Технологии и мастерство тестирования постоянно развиваются, и границы между различными областями постепенно размываются. Даже опытным специалистам нужно постоянно повышать свои навыки. В этой части рассказывается, что необходимо знать тестировщикам, бизнес-аналитикам или программистам, чтобы преодолевать трудности, возникающие в процессе тестирования. Мы объясним плюсы специалистов широкого профиля и перечислим неочевидные умения и отдельные техники, которые будут полезны тестировщикам и всему коллективу. В этой части раскрываются различные аспекты того, что и как изучать.

ЧАСТЬ ТРЕТЬЯ. ПЛАНИРОВАНИЕ РАДИ ЦЕЛОСТНОЙ КАРТИНЫ

Планировать ровно столько, сколько необходимо, – секрет равновесия. Пока мы работаем на небольших отрезках, необходимо следить за всем процессом, за всей системой. В этой части мы рассказываем о различных аспектах планирования: от выпуска продукта до постановки задач. Здесь вы также найдете информацию о разных моделях, таких как квадраты Agile-тестирования и предложенные стратегии адаптации.

ЧАСТЬ ЧЕТВЕРТАЯ. ТЕСТИРОВАНИЕ БИЗНЕС-ЦЕННОСТИ

Если вы, как и многие Agile-команды, разрабатываете надежный код только для того, чтобы понять, что это не то, чего хотел заказчик, вам пригодится информация из этой части. Мы рассказываем об инструментах и принципах, особенно из сферы бизнес-аналитики, которые помогут определить идеи и варианты на ранней стадии, а также убедиться, что каждый знает свои задачи. Мы также обращаемся к смежным дисциплинам и расширяем мышление.

ЧАСТЬ ПЯТАЯ. ИССЛЕДОВАТЕЛЬСКОЕ ТЕСТИРОВАНИЕ

Программисты прислали вам некий код для тестирования. С чего начать? Если вы или ваша команда испытываете трудности с исследовательским тестированием, то в этой части вы найдете много полезного. Мы выделили несколько техник исследовательского тестирования:

использование искусственных образов (персон) и туров (маршрутов) для генерации новых идей и тестов, сессионное тестирование (SBTM) и тестирование вокруг различных направлений («цепочек») работы (TBTM).

Наряду со всеми этими подходами мы рассматриваем другие пути удостовериться, что написанный код удовлетворяет потребности бизнеса и пользователей. В этой части мы рассматриваем пути минимизации рисков и сбора полезной информации в ходе нескольких видов тестирования, которые представляют трудности для Agile-команд.

ЧАСТЬ ШЕСТАЯ. АВТОМАТИЗАЦИЯ ТЕСТИРОВАНИЯ

Все больше и больше команд преуспевают в автоматизации тестирования, но при этом многие сталкиваются с единичными сбоями, разобраться в которых дорогого стоит. Каждый такой сбой может потребовать больше времени (и денег), чем все тестирование в целом. Автоматизированные тесты таят множество ловушек. В этой части мы приведем примеры, как распознать технический долг. Рассмотрим различные способы использования пирамиды тестирования, что поможет в планировании. Изучим некоторые альтернативные виды пирамиды тестирования для подхода к автоматизации под другим углом. Вы научитесь проектировать надежные и простые в использовании автоматизированные тесты. В этой части также приводятся примеры внедрения автоматизированных тестов в крупных компаниях.

ЧАСТЬ СЕДЬМАЯ. РАЗНЫЕ СФЕРЫ ДЕЯТЕЛЬНОСТИ

Ваш подход к Agile-тестированию будет определяться сферой деятельности. Вы работаете с крупными предприятиями? Возможно, перед вами стоит новая задача протестировать мобильное приложение или встроенное программное обеспечение. Может быть, ваша команда столкнулась с трудностями при поиске оптимального тестирования данных, которые помогают коммерческим предприятиям принимать решения. Интересовались ли вы, как Agile может работать при тестировании установленного ПО? Мы рассмотрим связь между тестированием и шагами в интеграции разработки ПО. Главы этой части касаются различных областей, поэтому мы включили истории людей, уже работающих в данных условиях. Некоторые из глав, возможно, не имеют отношения к сфере, в которой вы сегодня работаете. Но кто знает, что будет завтра.

ЧАСТЬ ВОСЬМАЯ. AGILE-ТЕСТИРОВАНИЕ НА ПРАКТИКЕ

В заключительной части мы рассмотрим, как команды могут визуализировать качество и тестирование, подведем итог всему, что узнали о методах Agile-тестирования. Это позволит чувствовать себя увереннее на стадии принятия решений. Создание общей версии для вашей команды невероятно важно для успешного исхода, так что мы поделимся моделью, которая поможет донести суть процесса тестирования до всех сотрудников.

В книге есть также два приложения:

- Приложение А. Page Object на практике: примеры.
- Приложение Б. С чего начать. ДРУГИЕ СОСТАВЛЯЮЩИЕ

Поскольку команды используют различные методы и подходы Agile, мы постарались сохранить общую терминологию, насколько это возможно. Чтобы говорить на одном языке, добавили глоссарий используемых терминов.

Надеемся, вы захотите подробнее узнать о некоторых методах, техниках и инструментах, которых мы коснемся. Пожалуйста, посмотрите список литературы в конце книги, сайты, статьи и блоги, на которые мы ссылаемся. Мы разбили их согласно частям книги, чтобы необходимую информацию было проще найти во время чтения. Источники, упомянутые в самой книге, для удобства размещены в алфавитном порядке. ЭКСПЕРИМЕНТ

Линда Райзинг много лет назад уговорила нас провести несколько маленьких экспериментов, изучить их результаты и продолжить решать задачи по частям, постепенно добиваясь поставленных целей. Если что-то в этой книге покажется полезным для вас и вашей команды, попробуйте внедрить это хотя бы пару раз. Оглядываясь назад, посмотрите, сработали ли нововведения, и при необходимости измените что-то. Если ничего не сработало, не огорчайтесь: вы получили опыт и сможете попробовать что-то другое.

Надеемся, на страницах этой книги вы найдете множество полезных для себя вещей.

Часть 1. Введение

В первой части мы быстренько пробежимся по истории того, как изменилось Agile-тестирование с тех пор, как мы впервые стали работать с Agile-командами. Представим некоторые новые концепции и подробно поговорим о важности изучения корпоративной культуры.

- Глава 1. Как изменилось Agile-тестирование.
- Глава 2. Важность корпоративной культуры.

Глава 1. Как изменилось Agile-тестирование

Каждый из нас начинал карьеру в области Agile как одиночка на поприще экстремального программирования (Extreme Programming, XP) во времена, когда даже не упоминалось о том, что в каждой команде могут быть свои тестировщики. Было всего два игрока: программист и заказчик. Клиенты определяли необходимые приемочные тесты, программисты автоматизировали их: раз, два и готово. На конференциях по экстремальному тестированию мы были единственными, кто определял себя как тестировщиков, хотя и понимали, что они могут внести значительный вклад в работу коллектива. Мы экспериментировали, обсуждали тестирование с первопроходцами XP, обменивались мыслями.

Но Agile развивался, а вместе с ним менялось и Agile-тестирование. Команды стали создавать библиотеки и фреймворки по автоматизации тестов и выводу их на новый уровень. Многие специалисты, внедрявшие Agile, оценили вклад опытных тестировщиков, таких как Элизабет Хендриксон и Майкл Болтон. Брайан Марик и Джошуа Кериевски впервые приняли на вооружение идею использования тестирования на основе примеров и историй, чтобы управлять разработкой.

Уорд Каннингем создал Fit (фреймворк для комплексного тестирования) – инструмент, помогающий выявить такие примеры. Дэн Норт представил BDD, которая проложила путь новым популярным инструментам (North, 2006). Agile-команды осознали ценность исследовательского тестирования, и оно стало для них не просто функциональным. Как показал Брайан Марик в своей матрице, которую мы адаптировали для квадратов Agile, тестирование охватывает теперь множество областей (Marick, 2003).

Конечно, все еще оставались трудности, препятствующие успеху Agile-тестирования. Мы, тестировщики, завидовали всем тем инструментам, которые имелись у программистов в свободной интегрированной среде разработок (Integrated Development Environments, IDEs). Нам хотелось, чтобы для нас все было так же просто. Мы начали эффективно использовать «силу трех», или «трех друзей», как говорит Джордж Динвидди. Когда заказчик, программист и тестировщик работают вместе, всегда возникают вопросы, как должен функционировать тот или иной элемент (Dinwiddie, 2010). Тем не менее было непросто учесть все запросы клиентов: дизайн, юзабилити, данные и другие составляющие качественного продукта. О некоторых из них мы поговорим в этой книге.

Практикующие специалисты в разных областях заполняют пробелы в Agile-тестировании. Мы счастливы поделиться историями реальных людей, рассказать о том, как они успешно использовали Agile-тестирование в разных сферах.

Тестирование в командах Agile – это процесс, а не конечный пункт. Эту идею, витавшую в наших мыслях, нам подсказала Элизабет Хендриксон (Hendrickson, 2006). В книге мы подчеркиваем: в процессе разработки софта тестирование должно учитываться на всех этапах.

Постоянно изучая лучшие техники Agile-тестирования, мы обнаружили, что подготовка специалистов широкого профиля с углубленными знаниями и навыками помогает справиться со сложными задачами. Опытные эксперты создали методы и шаблоны, которые позволяют членам команды из разных сфер сотрудничать и узнавать друг от друга ровно столько, сколько им необходимо.

Практикующие специалисты, такие как члены Agile-альянса инструментов функционального тестирования (Agile Alliance Functional Test Tools, AA-FTT), проложили путь к созданию более эффективных инструментов тестирования. Сегодня подход к написанию кода тестов так же важен, как и подход к написанию кода основного приложения. Мы научились определять, какой фреймворк, тестовая библиотека или драйвер подходят для конкретных нужд команды.

Выявляя потребности клиентов, бизнес-аналитики внесли свой вклад в развитие Agile. Тестирование и бизнес-аналитика обладают некоторыми взаимосвязанными качествами, которые помогают определить коммерческую ценность продукта. Таким же образом эксперты по пользовательскому интерфейсу (User Experience, UX) показали простой действенный способ вовлечения клиентов на этапах разработки новых элементов. DevOps-специалисты соединили разработку, функциональную часть и тестирование для улучшения качества в новых условиях (разработка и запуск). Их метод также позволяет сократить цикл запуска, снизить риски и повлиять на заказчика.

По сравнению с тем, что мы наблюдали несколько лет назад, сейчас Agile-команды понимают важность исследовательского тестирования. Хотя очевидно, что пользы от него может быть в несколько раз больше. Мы рады, что коллективы, использующие его в работе, делятся своим опытом.

Появились новые, творческие подходы к планированию тестирований и сотрудничеству. Сегодня у компаний куда больше возможностей визуализировать качество. Матрицы и пирамида тестирования адаптированы и расширены для применения в различных областях.

Все больше команд сталкиваются с тестированием мобильных приложений и встроенного ПО на расширяющемся поле устройств и платформ. Огромные объемы информации, содержащие новые технологии хранения, управления, анализа, поиска и визуализации, формируют новую категорию – большие данные (Big Data). И тестирование должно соответствовать новым реалиям.

Изначально Agile подразумевалась как методика для небольших коллективов, работающих в одном помещении. Сейчас мы видим, что Agile используют как крупные организации (нередко начинавшие как маленькие Agile-компании), так и распределенные команды. В организациях с общекорпоративным ПО тестирование сопряжено со строгими ограничениями. В то же время организации находят новые пути использования приложений с минимальным функционалом (Minimum Viable Products, MVPs), обеспечивающим итерационные релизы с быстрыми циклами обратной связи и доступное обучение.

В 2008 году некоторые компании были заняты тестированием в продакшн (на живой среде); мы не слышали об этой технике, если не считать случаев выпуска продукта без тестирования с одной только надеждой на успех. Сегодня выпуск обновлений для небольшого количества пользователей путем изучения логов на предмет ошибок и другие методы получения быстрой обратной связи могут стать важной и необходимой стратегией в определенных условиях.

Все эти изменения и инновации подвигли нас и других специалистов поделиться знаниями и опытом. Agile постоянно развивается, и многие из вас уже внедряют эту методологию, изменяя бизнес-процессы в обучении. Актуальные задачи, связанные с тестированием, во многом отличаются от тех, что вы решали, когда только начинали вникать в Agile. Надеемся, опыт, представленный в этой книге, породит идеи для экспериментов, чтобы можно было визуализировать качество вашего продукта, а затем изучить и адаптировать свои процессы.

РЕЗЮМЕ

С развитием программного обеспечения Agile мы постоянно изучаем и актуализируем методы тестирования. В этой главе мы рассмотрели, как именно менялось Agile-тестирование.

- Мы поняли, что тестирование – это процесс, невероятно важный для успешности продукта, и поэтому в нем должен быть заинтересован каждый член команды: от совета директоров до разработчиков и службы поддержки.
- Команды, работающие по Agile, четко осознают, как специалисты из разных областей (бизнес-аналитики, UX-дизайнеры и DevOps) расширили наши возможности, чтобы добиваться нужного для заказчика уровня качества.
- Инструменты для Agile-тестирования продолжают стремительно развиваться, позволяя Agile-командам внедрять инфраструктуру, необходимую для поддержки обучения и быстрой обратной связи.
- Agile-команды познают ценность исследовательского тестирования и других методов, которые помогают донести необходимую информацию до клиентов и разработчиков.
- Agile-тестирование должно идти в ногу с быстро меняющимися технологиями и новым контентом. Далее мы поговорим об этом.

Глава 2. О важности корпоративной культуры

Мы заметили схожие тенденции в организациях, переходящих на Agile. Команды разработчиков способны внедрить ценности, методы и практики Agile. Однако бизнес медленнее понимает преимущества Agile, и для изменений требуется больше времени. Для успешного перехода на Agile необходимо, чтобы между разработчиками и работниками коммерческих отделов существовало взаимопонимание и имелось представление о работе друг друга.

Если ваше руководство осознаёт, что через качество можно наилучшим образом донести коммерческую ценность до клиентов, скорее всего, и организация, и команда преуспеют. Регулярность и слаженность работы со временем совершенствуются. Наоборот, если компания отдаёт предпочтение скорости, качество часто страдает. Технические недоработки в хрупком, тяжелом коде и долгое время обратной связи снижают способность слаженно работать.

Если команды находятся под давлением нереальных сроков для выполнения определенных объемов, они не контролируют свою загруженность. С приближением дедлайна выбор у разработчиков сокращается, и они стремятся лишь управиться в срок, а не удовлетворить потребности клиента (Rogalsky, 2012). Скорее всего, они начнут торопиться и принимать решения, не думая о последствиях, и таким образом войдут в штопор постоянно увеличивающегося количества технических недоработок.

Командам необходимо обозначить опасность неразумных шагов, чтобы руководители компании знали о рисках и последствиях. Иногда нужно сбавить обороты, чтобы заняться техническими недоработками, или изучить новые концепции, или применить новые идеи. В своих презентациях Джанет использует рисунок с застрявшими в пробке суперкарами. Он показывает, как добавление новых элементов, не согласующихся с общей инфраструктурой, может парализовать работу. Зачем добавлять что-то, если система не будет работать, как ожидалось? Когда у вас накапливается слишком много технических недоработок, команда застревает. НЕ ЖАЛЕЙТЕ ВРЕМЕНИ

Чтобы представить клиентам полезный софт, необходимо подойти к задаче творчески. Людям нужно время, чтобы все обдумать и применить свое мастерство. Мы также должны постоянно осваивать новые технические навыки, изучать новые инструменты. Это помогает проводить небольшие эксперименты, которые могут быть полезны при решении различных задач.

В статье «Медленные идеи» (Gawande, 2013) Атул Гаванде объяснил, почему некоторые инновации, такие как хирургическая анестезия, внедряются быстро, а другим, например антисептикам, требуются годы (а то и вообще этого не происходит). Вывод: изменения происходят только путем обучения, практики и личной заинтересованности. В программах, которые анализировал Гаванде, люди обучались лучше всего, когда сами выполняли новые действия, описывая их своими словами, под наблюдением преподавателя. Может показаться, что дешевле и быстрее нанять тренера, который покажет, как нужно делать, или заставить людей посмотреть обучающее видео, но практический подход обеспечивает реальные стабильные изменения.

В сфере программного обеспечения (как и во многих других) люди порой забывают, что для овладения новыми навыками требуется время и практика. Они часто отказываются от тренера, который бы обучал и направлял сотрудников. Тратьте время на то, что нужно

Дэвид Эванс

, Agile-тренер по качеству, поделился метафорой, к которой прибегает, когда объясняет, что необходимо тратить время на то, что действительно нужно.

Иногда приходится сталкиваться с сопротивлением со стороны менеджмента, когда я предлагаю потратить время на совместное описание приемочного тестирования, прежде чем внедрять то или иное требование. Обычный аргумент: если команда потратит больше времени на приемочные тесты, это замедлит темпы разработки. В некотором смысле, конечно, замедлит, но ровно настолько, насколько замедляет движение автобуса, чтобы забрать пассажиров с остановки. Попытки оптимизировать скорость автобуса ни к чему не ведут, если не согласуются с целью усовершенствовать работу общественного транспорта. Представьте, если бы водитель для улучшения показателей предложил не останавливаться для пожилых пассажиров, поскольку они медленнее заходят в двери. Если бы автобус вовсе не останавливался и не брал пассажиров, он бы двигался быстрее. По скоростному шоссе он мог бы даже вернуться в парк!

Это та же ошибочная логика, согласно которой работа над тестами замедляет темпы разработки. Это хорошо иллюстрирует утверждение Кента Бека (Beck, 2002): «Код, который не прошел тестирование, не работает. Это, кажется, самое благоразумное утверждение». Если создание приемочных тестов перед тем, как приступить к разработке кода, что-то и замедляет, то лишь нашу работу над кодом, который не будет функционировать.

Компании, работающие с концепциями бережливого производства, научили нас, что уважение способности каждого сотрудника вкладывать идеи или непрерывно работать над улучшением качества приводит к созданию новых продуктов, которые нравятся клиентам. Команды, имеющие время и поддержку на обучение и эксперименты, справляются с работой куда лучше. Во второй части «Обучение для улучшения тестирования» мы поговорим о некоторых достойных, на наш взгляд, методах.

Гойко Аджич (Adzic, 2013) заметил, что отличный результат получается, когда:

- люди знают, зачем они делают свою работу;
- организации сосредоточены на результате и влиянии, а не на отдельных элементах;
- команды решают, что делать дальше, основываясь на актуальной обратной связи от пользователей;
- никому не наплевать.

Некоторые компании попробовали разгрузить своих сотрудников, чтобы дать им время на обучение, позволяя посвящать свободные рабочие часы личным проектам или собственному профессиональному росту. Лайза работала с командами, практикующими этот метод, и они демонстрировали хорошие результаты, поддерживая низкий уровень технических недоработок и сохраняя объемы производства. Однако важно понимать принципы, стоящие за эффективным использованием и возможностями, иначе команде придется втискивать еще больше работы в жесткие сроки. Загрузка производственных мощностей

Алексей Жеглов

, консультант по бережливому производству и Kanban для организаций, использующих умственный труд, объясняет теорию использования производственных мощностей и учит, как избежать потери 20 % времени. Мы приводим часть его статьи.

Главная причина потери 20 % времени в том, что при неполной загрузке мощности

используются на 80 %, а не на 100 %. Подумайте о компании, производящей ПО, как о системе, которая преобразует запрос на элемент в разработку этого элемента. Таким образом, мы можем смоделировать производство, используя теорию очередей.

Теория

Если запросы прибывают быстрее, чем система успевает их обрабатывать, они ставятся в очередь. Когда скорость поступления снижается, очередь уменьшается. Поскольку поступление новых запросов и их обработка происходят непоследовательно, длина очереди меняется.

J-кривая мощностей

Средний размер очереди остается на низком уровне, в то время как эффективное использование поднимается до 0,8, а потом резко возрастает и увеличивается до бесконечности. Это легко понять, представив процессор компьютера: когда мощность на низком уровне, он быстро реагирует на запросы, а когда фоновые задачи доводят ее до 100 %, компьютер начинает работать слишком медленно, чтобы отвечать на каждый клик.

Метод

Именно неполная загрузка сотрудников позволяет системе вовремя отвечать на запросы. Вот несколько практических советов, чего делать не следует.

- Калькуляция издержек (время инженера стоит X, но компания не может себе этого позволить). Экономические выгоды получаются при сокращении затрат из-за задержек.
- Создание проектной системы, чтобы люди могли в свои 20 % времени заниматься другими текущими проектами.
- Отслеживание 20 % времени путем заполнения ведомостей учета.
- Использование нововведений для того, чтобы занять 20 % времени. Не проблема, если только 20 % проектов выливаются в продукты. Проблема, если ваша компания не может внедрять новые технологии в основное время работы!
- Попытка вдохновить на творчество за счет этих 20 % времени. Утверждая, что вы раскроете творческий потенциал за 20 % времени, подумайте, почему вы до сих пор его не раскрыли в основное время.
- Выделять 20 % свободного времени каждую неделю по пятницам.

Это то, чего не нужно делать. А что же надо?

Окей, как насчет того, чтобы сделать все правильно? Ответим лучшим вопросом, который нам задали во время обсуждений со специалистами: «Если 20 % мощностей заняты задачами, не связанными с запросами в очереди, вы просто снижаете мощность до 80 %, и эти 80 % теперь составляют ваши 100 %, верно?».

Да, «80 – это новые 100». Это утверждение подчеркивает главную проблему, которая подстерегает попытки поддерживать 20 % свободного времени, не понимая сути теории. Вам хочется избежать ловушки снижения мощностей, не застрять в ней и иначе распределить время.

Мощность зависит от двух процессов: входящего (запросы) и сервисного (возможности). На самом деле вы не можете регулировать свои мощности. Они такие, какие есть, потому что процессы такие, а не иные. Однако вы можете поработать над процессами, улучшив возможности разработки софта и начав обрабатывать запросы. Когда у вас это получится, тогда и образуется неполная занятость.

Список литературы к первой части включает источники, из которых вы сможете больше узнать о производственных мощностях и результатах. Важно помнить, что перегрузка мощностей влияет на ход работы, а непосильные задачи на самом деле лишь снижают показатели. О ВАЖНОСТИ КУЛЬТУРЫ ОБУЧЕНИЯ

Если у вас есть возможность рисковать, и при этом провал не будет стоить слишком дорого, можете учиться на ошибках. Используйте быструю обратную связь, чтобы усовершенствовать ваши тесты для изменения чего-то, что работало «нормально», в то, что будет работать «прекрасно». Если ваша корпоративная культура нетерпима к ошибкам, вряд ли удастся внедрить инновации. Придерживаться одних и тех же процессов, даже если они не работают, всегда безопаснее, так что нет причин что-то менять. История Лайзы

К счастью, б

ольшую часть своей карьеры я работала в эффективных командах, где ценили непрерывное совершенствование. Когда сталкиваюсь с коллективами, где не поощряются вопросы и эксперименты, всегда чувствую себя так, словно врезаюсь в кирпичную стену. В одной команде, с которой я работала некоторое время, Scrum-мастер сказал, что я «тормозила весь коллектив», потому что постоянно задавала вопросы. Руководство давило на него из-за ускорения темпов разработки. Он не желал, чтобы его тормозили, и бесконечно указывал, что за три недели так и не создано никаких условий для тестирования. В той компании работали талантливые люди, но им не позволяли в полной мере проявить себя. Мои попытки стать катализатором перемен провалились, так что я поменяла компанию.

Рассказывая о силе ретроспектив (Rising, 2009), Линда Райзинг отметила, что некоторые руководители, сталкиваясь с такими методами, как парное программирование или ретроспективы, отвечают: «У нас нет времени, чтобы думать». Но если мы верим, что можем улучшить работу, у нас получится. Пусть не всё сразу, пусть не за одну ночь, продвигаясь вперед маленькими шажками, но мы найдем лучшие методы. ПОДДЕРЖАНИЕ КУЛЬТУРЫ ОБУЧЕНИЯ

Соберите свою команду, чтобы узнать, чему вам нужно научиться, кто из сотрудников хочет учиться. Позвольте людям строить собственный рабочий график, чтобы они могли найти время для экспериментов, чтения, тренингов, практики в чем-то новом. Члены команды должны хотеть узнавать о новых техниках и пробовать новые инструменты. Например, можно разрешить посвящать некоторое время работе над тем, что им нравится, будь то организационные навыки или поиск нового решения для автоматизации регрессионных тестов. Экспериментальные недели, во время которых сотрудники могут поработать над специальными проектами или изучить новые технологии, – один из самых популярных методов. Люди делятся приобретенными знаниями, поддерживая тем самым культуру обучения.

Гойко Аджич рассказал, как преуспел в работе с руководством, спросив, рассчитывает ли оно, что процессы со временем улучшатся, или же думает, что процессы уже налажены

наилучшим образом. Выяснив, что процессы можно улучшить, он начинает работать над бюджетом, не обязательно в плане финансов, а, например, в области временн

ых затрат. Скажем, сотрудники должны проводить 10 % рабочего времени, занимаясь улучшением рабочих процессов. Каждая команда сама решает, как использовать это время, но одно из условий – руководство при необходимости может урезать бюджет. Постоянное урезание бюджета – тревожный сигнал, который указывает: требуется еще один серьезный разговор.

Один из способов поощрения культуры обучения – создать сообщества по интересам (некоторые называют их «союзы»), в рамках которых сотрудники смогут в рабочее время делиться с коллегами мыслями. Мы постоянно слышим, как в таких сообществах обсуждают книги и делятся профессиональным опытом с единомышленниками. Союзы качества

Августо Евангелисти

, инженер по тестированию в Paddy Power, рассказывает, как тестировщики в его компании создали союз качества.

В моей компании создан союз качества, состоящий на 20 % из тестировщиков, на 20 % из методистов Kanban и на 60 % из разработчиков. Нужен ли нам кто-то, кто будет говорить, что делать, какие задачи решать, над какими нововведениями работать? Поверьте, нет. Сначала нас было пятеро, спустя пару месяцев – уже двадцать человек с удивительными идеями по повышению качества и улучшению нашей жизни. Мы встречаемся каждые две недели, и у нас всегда есть список тем для обсуждения.

Попробовал сделать то же самое на должности руководителя, и все закончилось тем, что ко мне пришли пять тестировщиков, которые сидели и ждали, пока я скажу, что они должны делать. Все зависит от того, насколько люди хотят улучшить собственную жизнь.

Мы любим встречи за чашкой кофе, где каждый может открыто высказать свои мысли и поделиться идеями. Подобные кофе-брейки, открытые для всех сотрудников организации, могут помочь преодолеть непонимание. Посмотрите соответствующие ссылки в списке литературы для первой части, чтобы понимать, с чего начать.

Просто совместная работа с коллегой, удаленно или в одном офисе, иногда становится прекрасной возможностью для обучения и развития, особенно если вы пытаетесь освоить новые навыки, которыми уже владеет другой человек.

Изменить устоявшуюся культуру непросто. Если ваши сотрудники еще не знакомы с Agile, помогите им адаптироваться, шаг за шагом. Посодействуйте им в приобретении новых привычек, убедитесь, что каждый член команды чувствует свою ценность и безопасность. Учитесь адаптироваться к переменам

Бернис Рухланд

, директор отдела качества, делится своим опытом ежедневных стендапов.

Многие коллективы причисляют себя к Agile, но не все следуют его принципам. Они Agile по духу, но необязательно на практике, общей для самоуправляемых компаний. Поймите суть методов и применяйте то, что работает. Начните с малого, и пусть ваши мысли развиваются естественным образом. Неудачи могут принести новую информацию, которая поможет понять, когда и как внедрять различные подходы и техники.

Например, если вы хотите проводить ежедневные стендапы, выделите в расписании пятнадцать минут в одно и то же время. Начните с вопросов, традиционно используемых Agile-командами: чем вы занимались вчера, чем собираетесь заниматься сегодня, есть ли у вас какие-то трудности. Оцените, насколько продуктивны эти вопросы. Возможно, для вашего проекта актуальными будут другие.

Короткие стендапы – отличный способ продолжать двигаться в заданном направлении и укреплять личное общение. Сосредоточьтесь на позитивном результате от подобных собраний, а не на том, как бы сохранить формат. Не затягивайте встречи, чтобы участники чувствовали, что вы цените их время. При необходимости измените формат.

В главе 6 мы рассмотрим больше вариантов обучения и оттачивания навыков, необходимых для успешного тестирования в Agile-командах. **ПРОЗРАЧНОСТЬ И ОБРАТНАЯ СВЯЗЬ**

Оперативная обратная связь – важный аспект не только для тестирования, но и для всего цикла разработки по методике Agile. Прозрачность в построении направления необходима для долгосрочного успеха в предоставлении высококачественного софта.

Если вы руководитель направления в IT-компании или собираетесь им стать, изучите продуктивные способы выстраивания процесса обратной связи и прозрачности вашей корпоративной культуры. Читайте книги по Agile-разработкам, такие как «Управление 3.0» Йюргена Аппело (Management 3.0, Appelo, 2011). Множество отличных идей можно найти в статьях и блогах лидеров по корпоративной культуре Agile – Джоанны Ротман и Эстер Дерби. (Смотрите список литературы к первой части.)

Прозрачность нашей работы для организации снижает необходимость контроля. Наглядность того, что делают разработчики, укрепляет доверие заинтересованных сторон и руководства. Доверие выстраивается со временем, терпеливо, методично и последовательно.

Поддержание ценностей Agile требует серьезных изменений. Как сказала Джоанна Ротман, Agile – это система для тех, кто хочет и способен предпринимать необходимые изменения в корпоративной культуре (Rothman, 2012a). Наша цель – это не внедрение Agile, а предоставление продукта, который хотят видеть клиенты. Цифровой музыкальный сервис Spotify часто приводится как пример компании, успешно внедрившей Agile-культуру и выросшей до полной прозрачности процессов. Хенрик Книберг поделился опытом Spotify в анимированном видео, которое, безусловно, стоит посмотреть (Kniberg and Spotify Labs, 2013).

Важно обращать внимание на любые успехи, большие и малые. Сообщайте людям об успехах других, чтобы им хотелось стать частью коллектива, где отмечают достижения отдельных сотрудников и всей команды. **ОБУЧЕНИЕ ОРГАНИЗАЦИИ**

Чтобы изменить культуру в компании, высшее руководство должно быть в этом заинтересовано. Топ-менеджеры должны понимать, зачем им нужны перемены.

Прежде чем рассказать руководству компании о чем-то, изучите их компетенции и обязанности. Например, вице-президент по финансам планирует финансовые показатели и пытается спрогнозировать окупаемость потенциальных инвестиций. Этот человек отлично осведомлен обо всех аспектах бизнеса и, скорее всего, поймет, насколько быстро развивается ПО, что может стать причиной задержек в производстве.

Найдите способ просчитать и объяснить преимущества необходимых, с вашей точки зрения, изменений, так же, например, как и стоимость предстоящих задач, выявления технических недоработок. Покажите руководству, как проблемы влияют на ключевые показатели бизнеса.

Когда руководители работают вместе с командами разработчиков для достижения

коммерческих целей, все в коллективе приходят к пониманию того, что действительно важно. После чего они могут предложить расставить приоритеты и сократить ненужные активы. В главе 9 подробнее остановимся на идее impact mapping, которая поможет в этом.

Для заинтересованных сторон приоритетным может быть результат. В таком случае сотрудники проводят необходимые доработки за отведенное время, используя обратную связь, продолжая улучшать продукт и показывать развитие проектов и разработок.

Стройте доверие на объяснении выгоды, даже если она небольшая. Боб Мартин (Martin, 2011) предупреждает: не говорите «Я попробую», когда вам ставят явно невыполнимые сроки. Вместо этого помогите руководителям понять, что реально может сделать ваша команда, рассчитайте затраты на любые вероятные сокращения и помогите пересмотреть объемы, чтобы увидеть важнейшие приоритеты более четко.

Судя по нашему опыту, коммерческие директора вполне нормально относятся к сомнениям, возникающим при разработке ПО. Как разработчики софта, мы можем реально спрогнозировать сроки выпуска простых элементов или тех, над которыми приходилось работать ранее, которые нам понятны. Однако заказчики, как правило, хотят новеньких возможностей, которых еще нет у конкурентов. Если требуется что-то новое, что нельзя спрогнозировать, объясните все возникающие сложности. Используйте такие методы, как «реальный опцион» (Matts and Maassen, 2007). Крис Мэттс позаимствовал этот термин из сферы финансов, где условия остаются открытыми до определенного момента – до принятия окончательного решения. Обращайте внимание на то, что расширенные возможности требуют времени на изучение альтернативных задач, которые позволят командам принять более продуктивные решения. Например, мы можем выбрать технологии, которые упрощают процесс, или согласовать плавающие сроки вместо жестких дедлайнов (Keogh, 2012b).

Фреймворк Synefin, созданный Дэйвом Сноуденом, позволяет оценить, будет ли новый элемент прост в разработке или потребует дополнительного изучения до принятия решения. Больше информации о том, как работать с ситуациями, вызывающими сомнения, вы найдете в списке литературы в конце этой части (Keogh, 2013b).

Независимо от подходов или модели для управления рисками и погрешностями, команды могут разъяснить людям, принимающим решения, что не всегда можно точно предсказывать развитие ситуации и что тестирование поможет держать их в курсе дела. История Джанет

В одной из организаций, где мне довелось работать, нашли весьма необычный способ быстро донести до руководства некоторые трудности, с которыми столкнулась команда. Множество сотрудников в организации работали над одним продуктом, так что их функционал пересекался. Все ежедневные встречи планирования начинались либо в 9:15, либо в 9:30. В 10:00 работающие по системе Scrum уже собирались в кабинете, где обсуждали все пересекающиеся обязанности. В 10:15 один Scrum-мастер оставался, и в кабинет мог войти любой заинтересованный руководитель. Здесь обсуждали новый вопрос, после чего возвращались к достойным предложениям, которые были уже определены, записаны и имели установленные сроки. Если вопрос был исчерпан, его стирали. Любые задачи, которые нужно было решить более чем за пятнадцать отведенных минут, записывались на доске с установленными сроками.

В книге «Изменения без страха» Маннс и Райзинг (Fearless Change, Manns and Rising, 2005) рассказывается о хороших способах представления новых идей. Для помощи во время совещаний с руководством посмотрите шаблон «Шепот на ухо генералу» (Whisper in the General's Ear).

Когда вы общаетесь с теми, кто не входит в команду разработчиков, никогда не упускайте возможности объяснить им, в чем состоит ваша работа. Спросите, что еще им полезно было

бы знать, будьте готовы понятно изложить, чего хотите от них. КАК РУКОВОДИТЬ ТЕСТИРОВЩИКАМИ

В «Гибком тестировании» мы немного говорили о начальниках отделов тестирования и их возможных ролях. Однако нам по-прежнему задают много вопросов на эту тему. Мы постоянно наблюдаем обсуждения на форумах. Тут нет правильного ответа, слишком многое зависит от контекста. Например, если у вас маленькая компания и всего пара команд, вам, возможно, и вовсе не нужен отдельный человек на подобную должность.

В некоторых коллективах сотрудники отчитываются ведущему разработчику или менеджеру по развитию. И это отлично работает, если у последнего есть четкое представление о том, что делают тестировщики. История Августо Евангелисти о союзах, которую мы рассказывали чуть раньше, иллюстрирует этот подход. В его компании старшие тестировщики обучают новеньких, и все вместе работают над совершенствованием процессов. Сейчас у них шесть команд, и Августо уверен, что по мере того как они расширяются, их корпоративная культура позволит сохранить этот стиль работы.

Некоторым организациям может потребоваться дополнительная структура отчетности или сотрудник, который бы конкретно отвечал за обмен опытом между командами. Адам Найт поделился своим опытом. Для него роль начальника отдела тестирования выходит за рамки ежедневной работы в Agile-команде. Б

ольшая часть его обязанностей заключается в том, чтобы видеть дальше рабочих циклов, больше фокусируясь на культурных и стратегических нуждах тестирования. По его мнению, необходимо, чтобы кто-то представлял тестирование на уровне директора по развитию для обеспечения баланса, учитывающего интересы всех отделов. В крупных компаниях должностные обязанности прописаны более конкретно, и в этом есть смысл. Адам всегда стремится убедиться, что у организации достаточно навыков для тестирования, что тестировщики в Agile-коллективах работают сообща, а их обязанности не дублируются.

Начальник отдела тестирования внутри организации может быть и кем-то вроде тренера. Его задача не говорить, что нужно делать, а организовать обучающие семинары, где все заинтересованные смогут обменяться идеями и обсудить их. Как мы уже сказали, нет единого верного пути. Поймите, что нужно именно вам, чтобы можно было начать практиковать гибкость и принести своей компании максимальную пользу.

Обратитесь к всемирным разработкам ПО и сообществам тестировщиков за помощью во внедрении изменений в корпоративную культуру, которые позволят лучше обеспечивать качество, быстро рассчитывать стоимость и работать на достойном уровне. Мы поговорим о полезных онлайн-сообществах в главе 6. РЕЗЮМЕ

Эта глава посвящена роли изменений в корпоративной культуре. Мы подчеркнули некоторые идеи, на которые стоит обратить особое внимание.

- Сосредоточьтесь на качестве, которое ведет к долгосрочной успешной работе и стабильности в разработке программного обеспечения.
- Разгрузите сотрудников, пересмотрев производственные мощности и регулируя объемы работ, чтобы ваша компания могла использовать все возможности.
- Развивайте корпоративную культуру, где команды могли бы выявлять недостатки качества и имели бы возможность немного экспериментировать в улучшении методов и процессов тестирования.

- Обучайте клиентов, рассказывайте им о достоинствах и выгоде хороших методов.
- Сделайте процессы наглядными, чтобы выстроить доверие между руководством и командами разработчиков.
- Помните, что важно отмечать любые успехи, большие и малые.

Часть 2. Обучение для улучшения тестирования

Мы все чаще сталкиваемся с коллективами, где в процесс тестирования вовлечены не только тестировщики, но и другие сотрудники. В то же время нам как тестировщикам нужно углублять и расширять свои знания и навыки, чтобы помочь компаниям, где мы работаем, обеспечить достойное качество продукта, на которое рассчитывают наши клиенты.

Во второй части поговорим о ролях и компетенциях, связанных с тестированием и качеством, а также о том, какие навыки необходимы для создания высококачественного софта – от интеллектуальных до технических. В последней главе этой части подчеркивается важность обучения для каждого сотрудника в отдельности и для всего коллектива в целом.

- Глава 3. Роли и компетенции.
- Глава 4. Интеллектуальные навыки тестировщика.
- Глава 5. Техническая подготовка.
- Глава 6. Образовательные ресурсы.

Глава 3. Роли и компетенции

Ряд задач, с которыми сталкиваются любые команды, ежедневно расширяется. Ваш отдел может работать над веб-продуктами, и вдруг руководство решает, что непременно нужно еще и мобильное приложение. Команды внедряют новые технологии, разрабатывают методы, инструменты, фреймворки, а в это же время приоритеты клиентов меняются. Что означает, что и сам процесс тестирования должен измениться.

Один из способов справиться с такими переменами и не погрузиться в хаос – контролировать, чему и как вы обучаетесь. Самоуправляемые команды

Бернис Рухланд

, директор по QA, делится своими мыслями о самоуправляемых командах.

Мы часто слышим о преимуществах и успехах самоуправляемых команд. Для кого-то такие изменения могут облегчить работу, а кому-то, наоборот, покажутся трудными. Перспектива отдать команде проект и позволить решать, как его вести, может быть пугающей. Мне приходилось встречать самоуправляемые команды, которые не могли и шагу сделать без

согласования с руководством. Некоторым необходимо, чтобы им ставили четкие задачи. Другие не любят говорить работникам, что делать. Порой пара человек оказываются завалены работой под завязку.

Это не значит, что такие команды не могут стать самоуправляемыми. Им просто требуется помощь. Мне кажется, в таких случаях полезными будут собрания, где можно обсудить детали проекта, ответить на вопросы и помочь в установлении основных правил с присутствующими менеджерами, тренерами или координаторами. Как координатор, я могу направить сотрудников, позволив им самим решать и определять стиль работы. Если вы руководитель или менеджер, попробуйте делегировать б

ольшую часть ответственности команде. Если им трудно справиться с самоуправлением, позаботьтесь, чтобы тренер периодически посещал их встречи, собрания и при необходимости направлял их. Как только станет возможно, позвольте команде вновь самостоятельно проводить собрания.

Также хорошо себя зарекомендовали мероприятия, во время которых каждый сотрудник может обсудить свои навыки и потенциальный вклад в проект. Часто у людей бывают нераскрытые способности или какие-то умения, о которых коллегам неизвестно. Например, тестировщик может разбираться в языке программирования, что пригодится для автоматизации повторяющихся действий. Со временем, помимо технического мастерства, начинайте обсуждать такие навыки, как управление, умение находить компромиссы и отношения внутри и за пределами коллектива, которые тоже могут быть полезны. При необходимости для начала составьте матрицу навыков. Постепенно переключайтесь от освоенных умений к тем, что требуются для построения здоровых отношений внутри коллектива.

Доверие между сотрудниками особенно важно для самоуправляемых команд. Отдельные коллективы разделяют общие интересы и строят личные отношения. Существует множество вариантов для развития сплоченности. Просто найдите те, которые подходят вам.

Обсуждая принятые меры, интересуйтесь, какие нововведения способствовали улучшению, а что показалось сложным. Не забывайте отмечать успехи и помогайте внедрять изменения, когда это необходимо. КОМПЕТЕНЦИИ ПРОТИВ РОЛЕЙ

Мы видим значительный прогресс в том, что касается важности компетенций внутри команд, а не ролей или должностей. Когда происходят такие изменения, чаще вместо «Это не моя работа» можно услышать «Как я могу помочь?». Сотрудники по-прежнему имеют ключевые навыки в каких-то вопросах, но уже не так сильно привязаны к определенным ролям. Например, фраза «Я тестировщик» означает: «В основном я занимаюсь тестированием, потому что люблю это занятие и разбираюсь в нем. Я могу руководить и направлять остальных, а могу помочь и в других областях». Важны ли должности?

Однажды на конференции Пит Волен из Мичигана раздавал свои визитки. Джанет с радостью приняла одну из них. Однако ее привлекла указанная на ней должность. Она спросила об этом, и вот что Пит ответил.

На моей визитке, кроме имени и контактов, есть еще три вещи.

Самое очевидное – «тестировщик ПО». Это то, чем я занимаюсь. Я тестирую софт и помогаю коллегам улучшить показатели в этой области.

«Антрополог ПО» было добавлено после долгих раздумий на Conference of the Association for Software Testing (CAST) в 2011 году. Майкл Болтон выступал с докладом о развитии

программного обеспечения, в частности тестирования как социальной науки. Это зацепило меня и заставило о многом задуматься. Главным образом мои мысли касались отношений между людьми и приложениями. При оценке принципов работы ПО эти взаимоотношения весьма важны и составляют существенную часть сути тестирования.

Это подводит нас к третьему и самому важному пункту – «постановщик вопросов». Это похоже на QA, и часто термин путают с тестированием. Сколько помню себя в сфере разработки ПО, так и было. Это раздражало меня некоторое время. В 2009 году я разговаривал с Майклом Болтоном, Фионой Чарльз, Линн МакКи, Нэнси Кельн и другими. В этой беседе постоянно использовалась формулировка: «Тестировщики задают вопросы, чтобы узнать, как реагирует или должно реагировать ПО». Постановка вопросов ведет к получению информации о софте, о том, как мы взаимодействуем с ним, и, определенно, к еще большему количеству вопросов. По крайней мере, до тех пор, пока на большинство вопросов не будут найдены ответы, удовлетворяющие заинтересованные стороны.

А это вновь возвращает нас к «тестировщику ПО».

Нам нравится термин «постановщик вопросов», который можно использовать и в планировании совещаний. Более подробно на этом остановимся в четвертой части.

В командах, с которыми работала Лайза, границы между ролями размылись. Некоторые программисты еще и опытные тестировщики. Иногда именно тестировщик находит простое решение для сложной задачи в написании кода. К тому же люди с широким кругом компетенций могут соответствовать более чем одной роли в коллективе. Например, в команде, где сейчас работает Лайза, есть программисты – системные администраторы с обязанностями DevOps-менеджеров.

В последние несколько лет термины типа DevOps стали широко использоваться, подчеркивая взаимосвязь между разработчиками ПО и сетевым оборудованием и операциями. Хотя термины и могут казаться новыми, отличительной особенностью Agile-команд всегда было то, что здесь выполняют работу, которую раньше обычно делал человек, занимающий другую должность. (Мы поговорим о взаимовыгодном сотрудничестве между специалистами DevOps и тестировщиками в главе 23.) Забудьте о разработчиках в тестировании. Нам нужны тестировщики в разработке

Триша Кху

, инженер по тестированию из Австралии, делится опытом и говорит о том, что происходит, когда вся команда думает о тестировании.

В прошлом году я устроилась в небольшую команду, где разработчики действительно ценили тестирование и рассматривали его как неотъемлемую часть всего цикла создания продукта. Я никогда не забуду одну из первых планерок, где мы обсуждали новый элемент. Один из разработчиков насупился и сказал: «Да уж, но как мы собираемся его тестировать?». В результате весь проект изменили.

Я едва не упала в обморок, потому что не слышала о подобном за всю свою карьеру. Важным было то, что не команда спрашивала меня, тестировщика, как

я собираюсь тестировать это. Вопрос задавался всей команде: «Как мы вместе собираемся это тестировать? Как нам сделать так, чтобы быть уверенными, что это будет работать так, как задумано?».

По ходу создания элементов разработчики всегда писали тесты: от модульных до

браузерных. Кто-то из разработчиков всегда вручную проводил тестирование перед тем, как передать продукт мне. Именно благодаря этому я редко обнаруживала баги, вызванные невнимательностью. Большинство были связаны с пользовательскими или системными сценариями, которые не проявлялись раньше.

Вы можете подумать, что мне как тестировщику в таком случае не приходилось слишком много работать. Мой самый ценный вклад в процесс заключался в том, что я анализировала продукт с точки зрения тестировщика и пользователя. Я поняла, что тестирование было не так важно в конце процесса, но невероятно важно в его начале.

Чем больше внимания я уделяла тестированию на начальной стадии, тем меньше усилий требовалось в ручном тестировании в конце, потому что в результате возникало гораздо меньше багов. Я бы хотела выделить последнюю мысль в умную цитату, потому что, на мой взгляд, это важно.

Но основной частью этого было то, что я знала: разработчики тестировали продукт качественно, вдумчиво писали тесты, тестировали их вручную по мере разработки. Я точно знала, что, если на встрече планирования мы говорили о сценарии, он будет качественно разработан и к концу процесса протестирован с помощью автоматизированных регрессионных тестов.

В этом году мне часто приходилось обсуждать роль тестировщика. Давайте оставим это сейчас и подумаем о роли разработчика софта. Этот человек должен быть уверен, что его продукт будет функционировать так, как задумано. Знания о том, как делать это на элементарном уровне, – важное качество для хорошего разработчика. Именно поэтому требуется больше тестирований на стадии разработки. И они должны проводиться именно теми, кто создает продукт.

Иметь в команде тестировщика ценно, но не стоит возлагать всю ответственность за тестирование на одного человека. У вас может быть отдельный специалист по базам данных, но это не значит, что он единственный, кто работает над этим. То же самое справедливо и для тестирований. Специалист может помочь в действительно сложных вопросах, зная, что все остальные члены команды в состоянии справиться с более простыми задачами.

Это заметно сокращает время обратной связи, повышает уверенность и ускоряет процессы QA. Кому такое может не понравиться?

Во многих Agile-командах появились специалисты с различными компетенциями. Например, сейчас часто можно встретить как бизнес-аналитиков, так и тестировщиков, плотно занимающихся различными вопросами бизнес-анализа. Границы между ролями и обязанностями размываются. В то же время такое пересечение обязанностей внутри команды не означает, что можно совсем обойтись без определенных специалистов. По нашему опыту, коллективы заходят в тупик из-за отсутствия у них специалистов с конкретными навыками. Порой решение простое: нанять человека, обладающего этими навыками, или обучить уже имеющихся сотрудников. История Лайзы

Команда, с которой я недавно работала, находилась в растерянности из-за того, что мы взялись за несколько проектов, которые не были достаточно оценены бизнес-экспертами головного офиса. В результате мы постоянно обращались к заказчику за разъяснениями требований или показывали готовые разработки только для того, чтобы в ответ услышать, что мы все сделали неверно.

Мы, тестировщики, предложили нанять бизнес-аналитика, который бы помог клиенту прояснить, что именно ему нужно. Наш руководитель продукта не был знаком с новой головной компанией, и хотя он встретился с заинтересованными лицами, тем не менее не знал, какие вопросы задавать, чтобы добраться до сути того, что нужно. Опытный

бизнес-аналитик знал бы, как вести переговоры с заказчиком, и задавал бы правильные вопросы.

К сожалению, менеджер не захотел нанимать бизнес-аналитика, так что мы создали свое сообщество по методам бизнес-анализа. Тестировщики, руководители продуктов, Scrum-мастера, менеджер-разработчик и заинтересованные программисты собрались вместе, чтобы прокачать свои аналитические навыки. Мы читали книги и статьи, посещали конференции, мастер-классы и участвовали в семинарах, чтобы разобраться в бизнес-аналитике. Мы постоянно встречались, делились информацией и записывали все в собственную вики-энциклопедию.

Возможно, лучше было бы нанять эксперта, но наши старания принесли плоды. Руководитель продукта обучился методам и техникам общения, которые мог использовать на встрече с представителями головной компании. В итоге он лучше стал понимать нюансы. Нам и теперь порой не хватает информации, какие именно задачи хочет решить клиент, но мы больше не тратим так много времени на то, чтобы ходить туда-сюда и спрашивать про каждую мелочь.

Т-ОБРАЗНАЯ СХЕМА НАБОРА НАВЫКОВ

В «Гибком тестировании» мы говорили о десяти принципах для тестировщиков, касающихся отношения и технических навыков. Давайте быстро повторим их.

- Предоставлять постоянную обратную связь.
- Создавать выгодные предложения для клиента.
- Создать условия для личного общения.
- Не бояться.
- Не усложнять.
- Постоянно совершенствоваться.
- Принимать перемены.
- Быть самоорганизованным.
- Сосредоточиваться на людях.
- Наслаждаться.

Однако до сих пор мы получаем вопросы типа: «Должны ли тестировщики в Agile-командах быть еще и программистами?».

Мы отвечаем, что тестировщикам необходима Т-образная схема навыков, которую впервые придумал Дэвид Гест (Guest, 1991). Чтобы эффективно работать в любой команде, вы должны обладать навыками настолько же широкими, насколько и глубокими. Обширные знания не только в своей сфере позволяют продуктивно общаться со специалистами разных областей. Глубокое понимание и разнообразные методы в одной сфере способствуют внесению значительного вклада в работу команды (Lambert, 2012).

Верхняя часть буквы «Т» у тестировщиков обычно включает технические навыки, например базовое понимание структуры системы, знание основного программного продукта и

принципов дизайна, умение создавать простые запросы базы данных, а также обращаться с такими инструментами, как платформа управления проектом и исходным кодом (Integrated Development Environments, IDEs) и непрерывно интегрируемые (Continuous Integration, CI) панели инструментов.

Другие члены команды должны, помимо навыков в верхней части «Т», владеть основными понятиями тестировщика. Поверхностное владение материалом может быть приемлемо в определенных ситуациях.

Умение приносить прибыль требует, чтобы некоторые члены команды, возможно даже бизнес-аналитики, обладали более глубокими знаниями. Соберите весь коллектив, чтобы обсудить Т-образную схему навыков и варианты заполнения пробелов. Не забывайте о десяти принципах Agile-тестировщиков. Отношение действительно определяет все. Команда квадратного типа

Адам Найт

, директор по QA и поддержке в Великобритании, рассказывает о своем опыте использования Т-образной схемы для создания команды квадратного типа.

Моя команда семь лет тестировала систему хранения большого объема данных на основе модуля SQL-запросов, которая была бы совместима с разными операционными системами, главным образом с Linux. Одна из основных проблем, с которыми я сталкивался во время тестирований такого рода продуктов, – ряд необходимых навыков, требующихся для выполнения всех задач тестирования системы. Тестировщики должны были не только проверить продукт с точки зрения различных заинтересованных сторон, но создать и поддерживать различные схемы тестирования. Перечислим некоторые навыки, которые нам требовались.

- Знание Linux/UNIX для создания и поддержания тестовой среды и ее мониторинга, чтобы оценивать влияние софта.
- Визуализация и знание принципов работы облачных систем хранения для расширения тестирования в условиях, которые бы поддерживали многофункциональную механизированную среду.
- Знание скриптов для постоянного развития и поддержания различных средств, необходимых для тестирования продукта с помощью инструментов командной строки.
- Знание программирования для развития и поддержания средств, необходимых для функционального тестирования и масштабирования клиентского программного интерфейса, если он отличается от основного языка программирования C и C++.
- Знание SQL и баз данных для понимания области применения и тестирования расширенного движка SQL на примере реальных запросов.
- Навыки исследовательского тестирования для определения и работы с рядом состояний и комбинаций операций, которые могут оказать влияние на уровне хранения данных.

Мы быстро поняли, что вряд ли найдем одного человека, обладающего всеми необходимыми навыками. Вместо этого я попытался набрать в команду тестировщиков, способных

справиться с поставленными задачами. У каждого сотрудника должны были быть определенные умения, позволяющие понимать продукт и основные подходы к тестированию.

Мы должны были быть уверены, что каждый способен включиться в решение наших приоритетных задач. Такие умения и служили дополнением к навыкам основных работников и всего коллектива.

Когда я впервые прочел о Т-образной схеме, я понял: это то, что делали мы. Идея широкого спектра основных навыков, совмещенная с глубокими знаниями узкоспециальных умений, – это описание именно того сотрудника, которого мы искали. Так, мне очень повезло работать с одним тестировщиком, который довольно хорошо знал базы данных и SQL благодаря прошлому месту работы администратором баз данных (DBA).

Мы наняли человека, отлично разбирающегося в скриптах, для поддержания основных средств тестирования, подкрепленных прекрасными возможностями пользовательской отладки. Другой сотрудник замечательно разбирался в операционных системах. Это нужно было для создания условий тестирования, виртуальных кластеров, для облачного и Hadoop-тестирования, а также продолжительного тестирования и тестирования рабочих характеристик. На рисунке ниже показаны навыки тестировщиков.

Три тестировщика с различными навыками

На мой взгляд, в концепции Т-образной схемы не хватает ключевого обоснования для подхода, который использовали мы. Представление о сотруднике, вписанном в Т-образную схему, ограничено им самим. Я же думал над истинной силой тестировщиков, которая проявлялась, когда их навыки сочетались с командными так, как у нас. Эту концепцию я назвал «командой квадратного типа» (см. ниже).

Каждый нанятый нами тестировщик привносил в команду какой-то уникальный навык. Некоторые из них были приоритетны при приеме на работу, другие могли не выделяться, но все равно присутствовали. К примеру, один тестировщик был не совсем очевидным кандидатом для работы над проектами внедрения с опытом работы в консалтинге. Однако его умения в составлении отчетов и анализ требований позволяли прекрасно понять желания заинтересованных сторон и установить соответствующие приемлемые критерии. Я также использовал отчеты этого работника по тестам как образец для остальных сотрудников. Если рассматривать таких специалистов вне контекста, их, возможно, не приняли бы на должность тестировщиков продуктов. Подходя к созданию команды комплексно, мы без проблем смогли ввести новых игроков. При этом все выигрывали от индивидуальных качеств каждого.

Команда квадратного типа

Как и Адам, мы полагаем, что важно оценивать и навыки команды в целом, и умения каждого сотрудника в отдельности. Сплоченной команде по плечу любые задачи. Не бойтесь использовать знания, которыми обладают другие. СПЕЦИАЛИСТЫ ШИРОКОГО ПРОФИЛЯ

Agile привлекает специалистов широкого профиля. Так называют людей с глубокими знаниями по крайней мере в одной области и с пониманием еще одной. Хм, похоже на Т-образную схему. Иногда так еще называют тех, кто хорошо разбирается во всем. Однако это не совсем то, что имеем в виду мы. В этом случае есть риск разбавить наши сильные стороны, и вместо того, чтобы делать хорошо несколько вещей, делать многое посредственно. Правда, для успешной совместной работы с другими членами команды на других должностях нужно иметь общее представление об их обязанностях. Как стать

специалистом широкого профиля

Мэтт Баркомб

, специалист по организационному дизайну, делится своими идеями о том, откуда берутся специалисты широкого профиля.

Одно дело знать, что такое специалист широкого профиля, совсем другое – понимать, как стать одним из них. Большинство проводит много времени, совершенствуясь в своей специальности, но не понимает, как расширить профиль, если не стать просто хорошим специалистом еще в какой-нибудь области.

Кого не назвать специалистом широкого профиля

За последние несколько лет я видел множество компаний, менеджеры которых слишком рьяно привязывались к этому термину. Заманчивая, но неверная мысль заключалась в том, что программисты, тестировщики, дизайнеры, аналитики, составители технической документации, выпускающие инженеры, все, кроме менеджеров, будут вдруг знать, как делать работу друг друга. Следовательно, они могут быть заменены любым специалистом из команды. Представьте, как просто было бы сбалансировать ресурсы проекта и что-либо спланировать!

Однако это не коллектив специалистов широкого профиля. Это команда универсальных специалистов. И это фантастика, поскольку невозможно, чтобы все занятые в производстве софта знали функции всех остальных с той же глубиной и в таких же объемах, которые необходимы для эффективных разработок.

Так кто же такой специалист широкого профиля?

Быть универсалом в многофункциональном коллективе – значит уметь ценить других и эффективно сотрудничать с коллегами на разных должностях. Это не значит, что человек может делать ту же работу, что и другой специалист такого же уровня или такой же энтузиаст.

Например, программист, хорошо разбирающийся в кодах, не заменит тестировщика, и наоборот. Однако оба могут отлично сотрудничать, быть партнерами. В идеале разница очевидна, и подобный пример можно применить к любым специалистам, тестировщикам, дизайнерам, аналитикам, системным администраторам, составителям технической документации и даже менеджерам!

Как стать специалистом широкого профиля

Существует множество подходов к тому, как стать специалистом широкого профиля. Единого метода обучения нет, но совершенно точно, что этот процесс полностью построен на обучении.

Один из подходов заключается в том, чтобы придать приобретаемым навыкам Т-образную форму. Здесь существует множество моделей освоения тех или иных умений, но с целью стать именно специалистом широкого профиля я использую три категории навыков: (1)

основные, (2) продвинутые, (3) мета, – связанные со знаниями, которым «необходимо обучиться». Количество таких знаний варьируется в зависимости от категории: для основных – небольшое, для продвинутых – наоборот, очень большое, наконец, для метанавыков оно меньше, но все же не настолько мало, как для основных.

Модель показана на рисунке ниже.

Эта модель наглядно демонстрирует, как стать специалистом. Однако в ней содержатся и намеки на то, как стать универсалом. Как видите, каждый новичок должен сначала выучить основы своей специальности (тестирования или программирования). Это в равной мере справедливо и для специалистов широкого профиля, и для тех, кто только стремится к этому.

Модель становления специалиста широкого профиля

Итак, первый шаг на пути к тому, чтобы стать универсальным сотрудником, – изучение основ вашей специальности. Для тестировщика в многофункциональной команде это может включать техническую подготовку. Например, тестировщикам потребуется подробно изучить процесс разработки или целевую среду или вникнуть в запросы к базам данных, синтаксические конструкции, структуры и инструменты, которые программисты используют для разработки продукта.

После этого на изучение продвинутых материалов по специальности могут уйти годы (иногда

большая часть карьеры), а метазнания требуют еще больше времени. Очевидно, заставить всех сотрудников многофункциональной команды провести столько времени за обучением – не самый лучший способ вырастить специалистов широкого профиля. Вопрос в том, как сотруднику продолжать расти в качестве универсального специалиста, не тратя годы на овладение продвинутыми и метазнаниями.

Ответ на этот вопрос лежит в понимании самой природы развитых метаспособностей. Метазнания – это та самая интуиция, тактика, которые развиваются у человека, когда он годами практикует продвинутые навыки по своей специальности, применяя их в самых разных ситуациях. Это практически шестое чувство в конкретной области. Специалист может использовать в описании кода, дизайна или структуры продукта такие выражения, как «Кажется, что-то не так», «Довести до ума» или «Чувствуется, что...» Суть в том, чтобы специалист замечал признаки чего-то, что нужно доработать.

Подобные признаки можно объяснить неспециалисту как эвристические или как правило «большого пальца». И это еще один способ вырастить универсального сотрудника. Такие люди могут работать в паре со специалистом и помогать, указывая на потенциальные проблемы. Эвристика не всегда безукоризненна, но работает отлично. Так же хорошо, как правило «большого пальца». Вместо того чтобы годами учиться применять метазнания в различных ситуациях, можно обучить универсального специалиста искать эвристические знаки.

Примером такого развития тестировщика в сторону программиста может быть овладение правилами чистого кода (Martin, 2009), принципами неповторения своих ошибок, продолжительность или количество уроков, обоснованные аргументы или множество «если» в его подходе. Универсалу не всегда нужно знать, как исправить те или иные ошибки. Иногда находятся веские причины для того, чтобы нарушить правила. Дополнительные ресурсы помогают обнаруживать признаки, а не исправлять ошибки.

Это ценно, потому что сотрудник оказывается вовлечен в процесс на более низком уровне,

другими словами, он сконцентрирован на анализе при выполнении задания или применении какого-то инструмента. Применение метазнаний, или эвристический подход, требует более высокого или отвлеченного взгляда на задачу. Поэтому человеку трудно одновременно рассматривать проблему и аналитически, и абстрактно.

Когда универсальные сотрудники овладевают основными навыками, а потом (что еще важнее) эвристическим подходом, они лучше работают с более узкими специалистами. Это первый и крайне важный шаг в создании эффективной многофункциональной команды. Переход от общения к сотрудничеству – залог еще более успешной многофункциональной команды.

Идея специалистов широкого профиля подходит для всех должностей, а не только для тестировщиков. Когда программисты знают основы тестирования, они более эффективно работают с тестировщиками и учатся предотвращать недостатки, что повышает качество разработки ПО. По нашему опыту, чтобы программисту лучше освоить навыки тестирования, следует плотнее работать с тестировщиками.

В команде, где работает Лайза, помогает составление чек-листов и запись узкоспециальной информации для тестировщиков в общую вики-энциклопедию. Конечно, данные в такой энциклопедии не заменят личного общения, но могут служить отличной шпаргалкой.

В главе 6 мы расскажем, как тестировщики могут овладеть основами программирования и создания баз данных, что поможет им более плодотворно общаться с коллегами. ПОДБОР НУЖНЫХ ЛЮДЕЙ

В начале карьеры на Лайзу сильное впечатление произвела статья Алистера Кокберна Characterizing People as Non-Linear, First-Order Components in Software Development (Cockburn, 1999). За двадцать лет изучив десятки проектов по разработке софта, Кокберн пришел к выводу, что общим фактором, обеспечивающим успех для всех, было то, что «в нужный момент подключались нужные люди». Именно хорошие специалисты, а не язык программирования, инструменты или методология делают проект успешным.

Наш опыт подсказывает, что крайне важно набрать в команду людей с правильным отношением. Не торопитесь, ищите тестировщиков, которые впишутся в ваш проект. Если люди, которых вы наняли, любопытны, хотят учиться и не боятся выйти из зоны комфорта, мы можете обучить их всем необходимым навыкам. Безусловно, работа в одном офисе имеет преимущества перед удаленкой, однако все же советуем расширить географию поиска. Удаленные тестировщики могут быть эффективны в командах, где правильно выстроен процесс коммуникации, где умеют извлекать пользу из современных технологий.

Книга «Чудаки на своем месте» Джоанны Ротман (Geeks That Fit, Rothman, 2012b) поможет найти и нанять на работу тестировщиков, обладающих необходимыми навыками и способных вписаться в вашу корпоративную культуру. Каждый сотрудник привносит в коллектив свои сильные стороны и свой взгляд, и важно учитывать все аспекты при оценке вклада того или иного члена команды. Разнообразие внутри коллектива необходимо для того, чтобы появлялись различные точки зрения. Иногда человек, который не вписывался в коллектив в самом начале, может быть полезным благодаря желанию разрабатывать качественный продукт для клиента. АДАПТАЦИЯ ТЕСТИРОВЩИКОВ

Даже опытным Agile-тестировщикам нужна помощь, чтобы освоиться в новом коллективе. Нам кажется, что устанавливать определенные ожидания полезно. Что человек должен узнать к концу своего первого дня? Первой недели? Первого месяца работы? Пусть эта информация будет доступна на вашей корпоративной вики-странице или в любой другой легкодоступной форме.

Помогите новичку познакомиться с разработчиками и представителями клиента.

Предоставьте ему список обязанностей каждого сотрудника и познакомьте со всеми. Назначьте время для встречи новичка с бизнес-экспертами, чтобы он понял, чем они занимаются. Удостоверьтесь, что у нового тестировщика достаточно времени для обучения, что ему не приходится сразу бросаться в работу. Заверьте новичка, что он может обращаться за помощью и задавать вопросы в любое время.

По нашему опыту, совместная деятельность помогает втянуться в процесс. Сведите нового сотрудника с другими тестировщиками, программистами, бизнес-аналитиками, специалистами DevOps, бизнес-экспертами. Хорошо бы поначалу назначить куратора или опекуна, чтобы вдвоем они могли работать как партнеры. Новые люди в организации приносят свои преимущества – у них незамутненный чистый взгляд. История Лайзы

Я как-то работала в паре с одним очень опытным тестировщиком, только пришедшим в финансовую сферу. Один из результатов нашего теста на несколько пенни отличался от того, что мы ожидали. За годы работы я сталкивалась с подобным множество раз и списала это на разницу при округлении. Однако новому тестировщику несоответствие не давало покоя. Он решил обсудить это с одним из программистов.

Оказалось, что это был самый настоящий баг, на который мы годами закрывали глаза. Тесты были написаны, проблемы решены, и мы больше не видели несоответствия. Пенни добавлялись, и было неловко, что мы будто бы немного обсчитывали некоторых своих клиентов. Ничто не сравнится с новым незамутненным взглядом!

Урок: не пренебрегайте тем, что замечает новый тестировщик!

Проявите творческий подход и в обучении нового специалиста. Сведите его с кем-то, кто объяснит структуру системы. Выделите время для программиста или системного администратора, чтобы ввести новичка в курс непрерывных интеграционных процессов. Организуйте обучение по автоматизированным тестам и сопроводительной документации. Открытые обсуждения полезны в совокупности с изучением документации, поскольку новый сотрудник в это время может задавать вопросы. Новичкам может быть непросто все сразу выучить. Попробуйте разбить обучение на небольшие отрезки. Возможно, после курса следует предоставить руководство по тестированию в печатном виде, чтобы человек мог еще раз прочесть все и изучить вопрос. (Подробнее о документах и исследовательском тестировании мы поговорим в главе 12.)

Майк Токс рассказал, что старается посвящать помощи новичкам 15–30 минут каждый день первые пару недель. После чего переходит на один раз в неделю, позже – раз в месяц. Требуется месяцы, чтобы новые сотрудники разогнались до нужной скорости, поэтому не торопитесь. От разумного подхода адаптации тестировщика в итоге выиграют все. РЕЗЮМЕ

В успешных командах, производящих высококачественное ПО, люди занимают разные должности, играют различные роли. При этом они имеют широкие компетенции. Посмотрите, какие из предложенных возможностей необходимы именно вашей команде.

Конец ознакомительного фрагмента.

Текст предоставлен ООО «ЛитРес».

Прочитайте эту книгу целиком, перейдя по ссылке https://www.litres.ru/pages/biblio_book/?lfrom=329574480&art=40457110 и купив полную легальную версию на ЛитРес.

Безопасно оплатить книгу можно банковской картой Visa, MasterCard, Maestro, со счета мобильного телефона, с платежного терминала, в салоне МТС или Связной, через PayPal, WebMoney, Яндекс.Деньги, QIWI Кошелек, бонусными картами или другим удобным Вам способом.

Сноски

1

Криспин Л. Грегори Дж. Гибкое тестирование. Практическое руководство для тестировщиков ПО и гибких команд. М.: Вильямс, 2016.

2

Расмуссон Дж. Гибкое управление IT-проектами. Руководство для настоящих самураев. СПб.: Питер, 2012.