Тестирование в гибкой методологии разработки

[Захаров Виктор](https://victorz.ru/author/victorzaharov) [09.02.2020](https://victorz.ru/202002091129)[Менеджмент](https://victorz.ru/category/management), [Тестирование ПО](https://victorz.ru/category/testing)

В процессе эволюции появляются новые методы и практики развития, правила и концепции жизни, и не важно, чего это касается. Это применимо ко всему. В один прекрасный день появилось понятие **гибкая методология разработки** (Agile software development) — целый ряд подходов и практик, которые основаны на ценностях манифеста гибкой разработки программного обеспечения. В данной статье мы их обсуждать не будем, а обсудим выстраивание тестирования в новых реалиях. Если хотите подробно ознакомиться с принципами и методами гибкой методологии разработки, то в интернете имеется огромное количество материалов.

Процитируем манифест гибкой методологии разработки программного обеспечения, чтобы вы понимали о чём дальше пойдёт речь.

Ценности (из манифеста):

* Люди и взаимодействие важнее процессов и инструментов.
* Работающий продукт важнее исчерпывающей документации.
* Сотрудничество с клиентом важнее согласования условий контракта.
* Готовность к изменениям важнее следования первоначальному плану.

Принципы (из манифеста):

* Наивысшим приоритетом является удовлетворение потребностей клиента, благодаря регулярной и ранней поставке ценного программного обеспечения.
* Изменение требований приветствуется, даже на поздних стадиях разработки.
* Работающий продукт следует выпускать как можно чаще, с периодичностью от пары недель до пары месяцев.
* **На протяжении всего проекта разработчики и представители бизнеса должны ежедневно работать вместе**.
* Над проектом должны работать мотивированные профессионалы. Чтобы работа была сделана, создайте условия, обеспечьте поддержку и полностью доверьтесь им.
* **Непосредственное общение является наиболее практичным и эффективным способом обмена информацией как с самой командой, так и внутри команды**.
* Работающий продукт — основной показатель прогресса.
* Инвесторы, разработчики и пользователи должны иметь возможность поддерживать постоянный ритм бесконечно.
* Постоянное внимание к техническому совершенству и качеству проектирования повышает гибкость проекта.
* Простота — искусство минимизации лишней работы — крайне необходима.
* Самые лучшие требования, архитектурные и технические решения рождаются у самоорганизующихся команд.
* Команда должна систематически анализировать возможные способы улучшения эффективности и соответственно корректировать стиль своей работы.

В принципах я выделил моменты, которые доводятся во многих российских компаниях до абсурда, возможно и в других странах этим грешат.

У любого учения есть адепты, однако стоит признать, что большинство из них впадают в крайности и своим поведением и невежеством вызывают отторжение к учению, которое они пытаются нести в массы.

Каким образом всё это относится к нам? Во многих компаниях топ-менеджерам на ухо начинают нашёптывать “дефективные” менеджеры, что есть замечательное учение о гибкой методологии разработки, которое решит все проблемы компании и настанут времена, когда некуда будет девать деньги. Для этого всего лишь надо привести в компанию “дефективных” адептов, которых они знают, ведь на одном из семинаров такие адепты красиво рассказали им о том, как это всё красиво работает и все процветают, внедрив у себя гибкую методологию разработки.

Приглашают в компанию адептов, и они начинают топ-менеджерам диктовать, как надо перекроить отделы и структуру управления, разбить их на Agile-команды включающие разработчика, тестировщика, аналитика, заказчика, руководителя команды (им может быть назначен менеджер продукта, проекта или ещё кто-то) и т.д. состоящую из 7-9 человек. Таких команд будут десятки. Адепты категорически настаивают, что команды надо обязательно собирать в одном месте по принципам манифеста гибкой разработки, но забывая сказать, что они от своего невежества интерпретируют принципы по-своему. И ссылаются на два принципа:

* На протяжении всего проекта разработчики и представители бизнеса должны ежедневно **работать вместе**.
* **Непосредственное общение** является наиболее практичным и эффективным способом обмена информацией как с самой командой, так и внутри команды.

Как видим, здесь не сказано “работать вместе в одном кабинете”, “непосредственное общение за одним столом”. Однако адептов это не останавливает и отделы разработки, тестирования, аналитиков и т.д. начинают дробить на единицы и разбрасывать по всей территории компании по Agile-командам, не принимая во внимание мнение других специалистов и не собирая обратной связи. У многих компаний со временем возникают проблемы и текучка кадров. У руководителей возникают проблемы с учётом ресурсов и равномерной загрузкой специалистов, ведь адепты почему-то учат, что нельзя ресурсы одной команды отправлять на помощь другой команде (обосновывают это чем угодно без использования здравого смысла).

Влияние на тестирование

Как это всё влияет на тестирование? Об этом сейчас и поговорим. Это также применимо к разработчикам, аналитикам и т.д.

Тестирование в компаниях должно развиваться централизованно, а не как обособленные единицы тестировщиков, независимо живущие в различных командах. Тестировщики разнесённые физически в разные команды решают локальные вопросы и задачи в рамках команд, а тестировщики сосредоточенное в едином центре решают глобальные задачи развития информационных систем компании.

Для успешного развития и поддержания на высоком уровне процессов тестирования в компании требуется руководитель имеющий практический опыт работы в тестировании, опыт управления командами тестирования, а также обладающий знаниями и навыками в области различных видов тестирования (ручное, автоматизированное, нагрузочное и т.д.). В этом случае руководитель будет задавать правильный вектор развития тестирования информационных систем компании и поддерживать процессы тестирования на высоком уровне.

Проблемы при разделении отделов тестирования на тестировщиков физически распределённых по отдельным Agile-командам:

1. Наличие в компании множества практик и стандартов тестирования. Тестировщики работающие самостоятельно в отдельных командах начинают работать так, как им удобно или так, как им хочется, и часто используют неправильные подходы и практики. В дальнейшем, если этот тестировщик попадёт в другую команду, то ему придётся перестраиваться под новые требования или пытаться переделывать существующие в другой команде практики под себя, что может приводить к конфликтам в команде.
2. «Зоопарк» различных инструментов и процессов тестирования. В каждой команде появляются свои инструменты, которые не используются тестировщиками в других командах, так как они считают, что им нужен другой инструмент, что приводит к проблемам аналогичным п. 1.
3. Тестировщик определённой команды может тестировать только определённую часть системы над которой работает его команда, другие части не берутся в работу, так как команда этого не допускает, ведь это уже не в их зоне ответственности. Если у команды проблемы при решении задач, то она не может оперативно привлечь к решению задач дополнительные ресурсы другой команды.
4. Ресурсы тестировщиков не распределяются централизованно и если у тестировщика в команде нет задач, то у тестировщика простой в работе, а в другой команде наоборот может быть загруженность, требующая переработок.
5. Нет взаимного обучения и обмена опытом с другими тестировщиками, что не позволяет тестировщикам развиваться профессионально в процессе работы (идёт технологическое отставание).
6. При работе в разрозненных командах тестировщики быстро теряют интерес к работе и меняют место работы, при этом ищут компании, где тестирование сосредоточенно в едином центре. Тестировщикам не хватает социализации на основании профессиональных интересов.

Вышеприведённые факты взяты из жизни и личных наблюдений. Пункты 5-6 хорошо проявляется на собеседованиях, когда проводишь собеседование и спрашиваешь, почему тестировщик уходит с его текущего места работы. Их гонит «дефективный» Agile.

Всё перечисленное выше снижает качество и уровень процессов тестирования, и ведёт к появлению проблем у компании.

Выгоды единой команды тестирования:

1. Единые стандарты тестирования компании. Все тестировщики работают по единым стандартам, что позволяет им без проблем подключаться к тестированию других систем.
2. Единые инструменты и процессы тестирования. Аналогично п.1.
3. Универсальность тестировщиков. Тестировщики могут тестировать несколько систем одновременно или периодически подключаться к тестированию других систем, или блоков одной системы, если у них появляется свободное время.
4. Централизованное управление ресурсами тестирования, централизованный контроль и распределение загрузки тестировщиков.
5. Взаимное обучение и профессиональное развитие тестировщиков в процессе работы. У меня в отделе ребята самостоятельно собираются в группы и после работы изучают новые инструменты и практики работы, для этого мной создаются условия и проводится ряд подготовительных работ.
6. Совместное решение задач с привлечением тестировщиков по другим системам.
7. Социализация на основании профессиональных интересов, что немаловажно для специалиста любой профессии.

Что можно сделать?

Часто нашего мнения не спрашивают, когда разбивают команду тестировщиков на отдельные единицы и раскидывают по Agile-командам. Однако, как только вы слышите, что начинается данная деятельность вы должны подготовить обоснование того почему стоит команду тестирования оставить единой, а также описать принципы работы команды с Agile-командами.

Как работать тестировщикам с Agile-командами, если они появились? Для этого надо правильно выстраивать работу отдела тестирования с самого начала его появления или с момента вашего появления в отделе как начальника отдела. Если всё правильно выстроено, то отдел может работать по любой методологии разработки.

Принципы работы отдела:

1. Каждая информационная система должна иметь свою группу тестирования.
2. Тестируйте системы, а не сквозные процессы. У нас сейчас в компании более 20-ти крупных информационных систем и столько же “мелочи”. Ни один тестировщик не сможет их изучить, чтобы протестировать процесс, проходящий через все эти системы, но он может хорошо изучить несколько систем и грамотно их тестировать, включая интеграцию с несколькими смежными системами.
3. На проект выделяйте отдельных тестировщиков из группы тестирования, которая относится к системе, по которой запустили проект, так как у них уже есть знания по системе и не надо с нуля изучать её. Выделение — не значит перемещение тестировщиков физически в отдельную команду, а выделение их времени на 100% — 80% — 60%, для решения задач проекта.
4. Если проектных задач будет мало, то тестировщикам в работу передавайте задачи по системе вне проекта.
5. После окончания проекта тестировщики должны передать все знания всей команде тестировщиков по данной системе, по которой был проект.

Заключение

Я за все методологии разработки там, где они уместны. В этом случае я быстро адаптирую отдел или отдельную команду тестированию под определённую методологию, чтобы не быть узким местом в процессе разработки.

Есть множество мировых компаний, которые работают по гибким методологиям разработки. Ряд этих компаний нанимают сотрудников со всего мира. Если бы они опускались до абсурда и интерпретировали озвученные выше принципы манифеста гибкой разработки ПО по-своему, то им пришлось бы всех своих сотрудников согнать в один офис в одной стране, но это уже не гибко и скорее всего этих компаний сейчас просто бы не существовало.

Мои рекомендации:

* Будьте всегда гибкими.
* Внедряя новые практики у себя, изучайте вопрос досконально и проработайте все варианты развития событий.
* Подстраивайте практики под свои процессы и улучшайте процессы. Не подстраивайте свои процессы под новые практики.
* Не идите на поводу у “дефективных” менеджеров, иначе они погубят ваш отдел.

В данной статье озвучено сугубо моё личное мнение, сформированное за долгие годы работы в различных командах, работающих по различным методологиям разработки.