Интервью с Асхатом Уразбаевым (ScrumTrek.ru), основателем сообщества AgileRussia.ru

Автор: Александр Орлов (Happy-PM.com)

Текст: Алексей Лупан (TestItQuickly.com)

Часть II

Асхат, есть ли успешные примеры применения Agile в распределенных командах?

Я пример приведу, ладно?

Появилась на рынке такая идея — Agile Swarm — рой. Когда все работают из дома. Можешь себе это представить? У тебя люди в пяти городах, каждый работает из своего дома, но при этом очень эффективно общаются через современные средства, которые уже есть, например, Skype, в котором можно «висеть» часами — и это практически бесплатно. Есть общий skype-чат всей команды, и время от времени люди там пишут какие-то важные новости.

Если хочешь парно программировать (что очень важно для налаживания коммуникаций), то проблем нет никаких — просто шаришь десктоп через тот же Skype, и вы вдвоем работаете. Как передаете контроль? Один является презентором, потом другой. Ребята рассказывали, что так работают.

Вы коммуницируете через вот такие вот средства общения. Но это не может стопроцентно заменить персонального общения. Раз в итерацию, раз в месяц, например, вы встречаетесь лично. Демонстрация, ретроспектива и планирование происходят лично. Если вы по СНГ разбросаны — не так сложно приехать друг к другу раз в три недели на один день. Съехались, провели вместе день-два, и разъехались.

Пример успешного? У меня друг так работает. У него своя компания, они занимаются разработкой ПО, у них свой продукт и они работают на заказ. Несколько проектов у них работают именно так. Каждый человек сидит в своем городе. И нормально все работают.

Если инженеры сидят в разных временных поясах, там, наверное, с коммуникациями очень плохо... То есть, Agile не очень эффективен в подобных ситуациях.

Ну, как тебе сказать... Люди же хитрые, все придумано уже. Вам нужен совместный SCRUM — это обязательно, 15-тиминутный митинг, где синхронизируется работа. Вы находите время, когда вы все присутствуете.

Например, три разных места, у всех разные тайм-зоны, но все собираются и обсуждают те вещи, которые нужно обсудить.

Всегда можно встретиться. У нас же не бывает никогда разницы большей, чем 12 часов (так часы устроены).

В моей практике такое было...

Подожди, как это может быть, больше 12 часов?

Очень просто. Был Питер, была Калифорния — разница в 11 часов. И был еще Новосибирск. Это три часа на восток от Питера. То есть, между Новосибирском и Калифорнией разница была в 14 часов.

Ну нет, подожди, 14 часов — это 10 часов, только в обратную сторону.

Это 10 часов, только в обратную сторону... Да, это 10 часов, только в обратную сторону.

Ну вот, десять часов разница.

Но мы почему-то встречались в Калифорнийское утро и в поздний Новосибирский вечер.

Ну, нашли какое-то хорошее время. Да, и дело не в этом, просто найдите время, которое вам удобно. Но я понял. В общем, стараются найти время, в которое все офисы могу пообщаться. Иначе, конечно, беда будет. Потому что информацию нужно будет передавать...

Вы встречаетесь в Skype, и проводите SCRUM. Обычно SCRUM-мастер пишет SCRUM-notes — записывает итоги скрама, которые потом рассылает всем или на wiki выкладывает. Вы уже ложитесь спать, а им еще работать целый день. И, соответственно, они, скажем, американы, в конце дня высылают вам свои SCRUM-notes, в которых указано, кто чем сегодня занимался, чтобы завтра утром вы пришли, прочитали, и смогли продолжить работу.

Time-difference, на самом деле, не убийственная вещь. Конечно, затрудняет немного. Кстати, да, это означает необходимость в большей документации. Нужно больше документировать, больше wiki использовать, другие knowledge sharing tools. Приходится чем-то жертвовать.

Еще один интересный вопрос, который мне задавали многие знакомые собственники компаний. Как продать Agile заказчику? У них обычно fixed price проекты. Они от этого страдают.

Конечно, все от этого страдают от fixed price проектов.

И непонятно, как заказчикам продать Agile. Где у них волшебная кнопка, на которую можно надавить, чтобы жить по Agile. Посоветуй им что-нибудь.

Неблагодарное дело, давать такие советы. Нужно кое-какие вещи просто понять.

Во-первых, нужно просто научиться продавать. Это относится к любой сфере деятельности. Речь ижет о том, чтобы просто научиться продавать, а не о том, возможно это или невозможно.

Что нужно понимать про fixed-price проекты. Это, в общем, удовольствие достаточно дорогое. Ты согласовываешь с заказчиком определенный объем работ. Чтобы его согласовать, тебе нужно собрать требования. Чтобы их собрать, тебе нужен классный аналитик, который посидит с тобой, поработает, и напишет определенный документ. Причем уровень детализации будет достаточно суровый. Ты рискуешь деньгами. Придет заказчик, и скажет «Оба-на!», и ты такой «Оп!» и сел на попу, и тебе придется делать за свои деньги, если ты это не опишешь заранее. А если описал, то говоришь: «Любой ваш каприз, но за дополнительные деньги». И ты в шоколаде.

То есть, тут больше работы уходит на детальное документирование. Если ты этого не делаешь, то у тебя другие финансовые риски. Все эти вещи вносятся в цену.

Что еще... Любые изменения, которые могу произойти, заказчику придется оплачивать. И самое печальное, что он рискует не столько деньгами, сколько сроками. А практически для всех заказчиков это более серьезно, нежели деньги. Когда он приносит что-то и говорит «Ребята, надо это сделать. Я понимаю, что мы об этом не договаривались, но надо». Я говорю «Хорошо, плюс пять дней» - и это его расстраивает. Вот этого в Agile тоже нет, у нас срок фиксируется, мы просто что-то менее приоритетное выбрасываем.

Еще это стоит дороже потому, что заказчик будет «просовывать». Это когда заказчику это на самом деле не нужно, но в scope вроде бы как описано. У нас были такие случаи, ребята не сделали одну задачу, в строчке в техническом задании что-то непонятное было написано и не согласовано, буквально одна строчка, а ТЗ большое, страниц на 300, и мы не сделали одну строчку, и пошли сдавать заказчику. А из этой строчки как посыпалось столько всяких вещей, которые надо сделать, и это было реально ужасно. Пришлось на 20 или 30 процентов больше работы сделать. И это из-за одной строчки, которую не сделали, просто потому что не поняли, что там было написано.

Так что, заказчик будет «просовывать». То есть, там есть строчка, где он будет просовывать. И будет просовывать даже то, что ему не нужно, потому что деньги уже уплачены. Отношение такое - «я уже деньги заплатил, я это

съем, даже если оно будет невкусное». Это такое нехорошее, какое-то послевоенное отношение, но с этим уже просто ничего не поделаешь.

У меня даже были знакомые, очень грамотные заказчики (которые работали по Agile). Представь себе, со своими субконтракторами они работали достаточно жестко, и «просовывали им много чего» просто для того, чтобы «было», вдруг пригодится. А ты под это закладываешь «буфера». Риски оплачивает заказчик.

С fixed-price нужно уметь работать. Уметь описывать, уметь scope прописывать, иметь communication policy, уметь согласовывать — change management policy — с заказчиком, чтобы все это было подписано. Это же время затягивает. Как только ты это начинаешь понимать, то тебе ясно, что в fixed-price проекте вы — враги заказчика. Вы по разные стороны баррикад, и ваша цель — у каждого своя, и если ты победил, то он — проиграл, и наоборот. То есть, это не win-win, не ситуация, когда все выигрывают.

Как только ты это начинаешь понимать, то ты начинаешь понимать, как все это продавать заказчику. Давить на жадность, прежде всего. Например, говорить: «Вы хотите увидеть продукт через две недели? Мы будем показывать его вам каждые две недели». Редкий заказчик откажется, всем это кажется «супер!». И ты говоришь: «Хорошо. Но вот смотрите — будем показывать каждые две недели, но вы не сможете сказать, что вам нравится и что не нравится. Потому что у нас в контракте будет прописано...» Он скажет: «Нет, мы хотим!» Тогда ты говоришь: «Давайте контракт немного по-другому составим, и будем это делать». Опять же — это может не получиться, заказчик может отказаться, тогда ты должен понимать, какие цели у твоего проекта, чего ты хочешь добиться. Возможно, твоя цель — просто завязать нормальные отношения с заказчиком, чтобы получить от него на поддержку кусок бизнеса. Вся поддержка идет по T&M (time and materials) потому что scope заранее не известен. Поэтому ты делаешь «пилот», делаешь его как можно более коротким, выигрываешь, а дальше переходишь на нормальные контракты T&M хоть по Agile, хоть не по Agile, но по крайней мере, без таких серьезных финансовых рисков. Но надо вложиться на первой стадии, на стадии «пилота», чтобы заказчик тебя полюбил. Может быть, даже иногда за свой счет.

И кстати, вообще при работе с заказчиком надо не бояться говорить нет заказчику. Надо понимать, что заказчик тоже человек, и ты можешь сказать ему «нет», но это должно быть конструктивное «нет». «Нет, но мы можем поступить по-другому...» И не считать, что он такой весь из себя... заказчик, неотразимый, и что ему нельзя сказать нет. Это скорее относится к менеджерам проектов, владельцы бизнеса не страдают такими проблемами, как правило.

Супер. Посоветовал, так посоветовал... Говоря об Agile, часто упоминается SCRUM или XP. Давай раз и навсегда объясним народу, кто круче — SCRUM или XP?

SCRUM против XP это как слесарь супротив столяра. Реально, это настолько не противоречащие вещи, не враги, чтобы их стравливать. Они очень хорошо друг друга дополняют. SCRUM ориентирован скорее на управление проектами, там есть такое понятие как бэклог, например. Там четко прописано, что такое самоорганизующаяся команда, как ее добиваться, и там есть такая роль как product owner — представитель заказчика. Единственный человек со стороны заказчика, который знает, что нужно.

XP в этом отношении более размыт, не так все четко структурировано и не так все конкретно. Зато XP дает тебе прекрасные практики, связанные с техническими сторонами Agile — что такое Test Driven Development, что такое рефакторинг, как все эти вещи делать... И многие из тех команд, с которыми я работал, начинают со SCRUM, потому что он дает организационные практики, позволяет построить команду, и уже хорошая команда идет внедряет XP практики.

Я с Ростова приехал, ребята уже год этим занимаются — у них уже все XP-ные практики внедрены, парное программирование, и Test Driven Development. Хотя начинали со SCRUM.

Бывает и наоборот. У меня были знакомые ребята, они начали с парного программирования, потом TDD. Они выросли как команда и потом пошли в сторону организационных практик, итераций, бэклога, планирования релиза.

И то и другое возможно, одно другому не противоречит, а дополняет друг друга.

По поводу технических различий… В SRUM, например, итерация — один месяц. Почему так много? Предполагается, что ты за этот месяц и разрабатываешь требования, и пишешь код, и тестируешь его, и это минимальный срок, за который ты можешь сделать что-нибудь осмысленное. Один месяц.

В XP итерации недельные, и предполагается, что к началу итерации у тебя уже есть полностью описанный scope в виде юзер-сториз и приемочных тестов.

Какой подход более эффективный? Да Бог его знает. Большинство команд, с которыми я работал, в этом вопросе были больше похожи на XP. Ты берешь те практики, которые считаешь нужными и оттуда, и оттуда, и те, которые подходят к твоей конкретной ситуации.

Поэтому ни SCRUM не круче, ни XP не круче. Хотя я, наверное, должен был сказать, что SCRUM круче, да?

Конечно, надо было сказать, что SCRUM многократно круче чем XP!

Этим и закончим. При всем при том, что я сказал, SCRUM, конечно, в сто раз круче. Знаешь почему? Его внедрить проще. Тебя не заставят делать парное программирование, а это очень круто. Ты внедрил Agile, а парное не надо — это же круто?

Кстати, да, многие инженеры с опаской относятся к парному программированию. Привыкли всю жизнь работать одни, а тут к тебе кого-то подсадят...

...особенно из твоей конторы. А знаешь почему? У вас там кубики, наверняка?

Не, у нас там как раз кабинеты.

Кабинеты — это еще хуже. У каждого программиста свой кабинет?

Нет, в одном кабинете — два программиста. Может, это как раз очень хорошо для XP?

Нет, это плохо. Почему? Люди привыкли в privacy. Они привыкли к тому, что в их монитор никто не заглядывает. И когда ты придешь такой веселый и говоришь «С завтра у нас XP!» все так: «Ё-моё, а как же мои любимые порно-сайты и ли чего там ещё... Ну, я же привык в аське трещать с любимой девушкой, что же делать...» Это privacy, которое народ так любит – оно немного препятствует. Хотя, в принципе, можно приучиться.

В общем, я захожу в контору и вижу, приживется ли там парное программирование. Просто — можно ли посмотреть в монитор к любому человеку с середины комнаты, или нельзя. Все это зависит людей, от культуры в команде.

В принципе, парное программирование не более медленный способ разрабатывать. Потому, что...

Народ не ходит на порносайты.

Да, больше дисциплины. Знаешь, это не единственный плюс. То есть, кажется, что парное программирование — оно в два раза медленнее, если просто логически рассуждать. Но это не совсем так, ведь когда ты пишешь код — клавиатура не клацает постоянно. Ты пописал, остановился, подумал, еще пописал. Когда вдвоем пишут, то одна клавиатура, и когда один пишет, второй думает, и когда первый остановился подумать, то второй забрал клавиатуру и сам пишет. Сокращается время на обдумывание, что тоже бывает полезно.

Когда Agile применять не надо? Есть ли такие случаи?

Конечно есть. Agile не надо применять, если тебе не нужен результат.

То есть когда тебе нужно освоить бюджет?

Типа того. Попробую рассказать на примере. Ездил в одну компанию, куда меня пригласили про Agile рассказать. Собралась куча народу, я такой стою, рассказываю, все такие задают мне правильные вопросы. А потом началось: а вот в такой-то ситуации это не подойдет, и началась перепалка на тему. Я начал задавать вопросы, и постепенно добрались до принципов Agile, и вдруг выяснилось, что у них в компании все не так, цель их конторы — освоить бюджет, и эффективность процесса означает, что там будет достаточно большое количество аналитиков, чтобы этот бюджет съесть, немерянное количество программистов, тестировщиков, и зачем тебе эффективный процесс, если это противоречит твоим бизнес-целям?

Agile — очень бизнес-ориентированный. Он очень высоко дисциплинированный процесс в отличие от того типичного бардака, который есть у всех. Он более дисциплинированный чем уотерфолл за счет того, что есть более близкие дэдлайны, и команда на них фокусируется. За счет этого он очень эффективен. Если тебе высокая эффективность не нужна, то да, Agile не нужен.

Есть еще несколько шоу-стопперов, например те, которые Алистер Коберн в своих книжках привел — не применяется Agile в больших проектах и не применяется в критически важных проектах. Но тут надо просто понимать, что такое Agile. Если Agile в смысле Extreme Programming, то, наверное, он прав. А если Agile в смысле «лучших практик индустрии», то, наверное, все-таки не прав. Если у тебя большой проект, то ты разбиваешь свою команду на несколько Agile-команд — есть примеры команд в три тысячи человек, которые работают по такой схеме.

Что касается критичных проектов — критичность определяется не тем, нужен ли тебе Agile или нет, а тем, сколько людей рядом с тобой стоит и заглядывает тебе через плечо, заглядывая, правильно ли ты делаешь, или неправильно. Есть куча организаций и людей в Америке и у нас, которые просто смотрят за тем, что ты делаешь. И это заставляет тебя писать намного больше документации, нужна она или нет — никто не определяет, просто тебя очень хотят проверять. Твой процесс определяется, в том числе, и тем, что тебя проверяют. Это тоже накладывает ограничения, хотя никто не говорит, что тебе не нужно работать, например, короткими итерациями, чтобы в конце каждой итерации иметь работающий софт. Или не писать юнит-тестов. Ты можешь работать короткими итерациями, даже при том, что у тебя есть заранее написанные требования. Это конечно шоу-стоппер, но Agile-практики (тот же TDD) будут делать твою работу только эффективнее.

В общем ответ: где не надо использовать Agile — там, где тебе не нужен результат.

Fixed-price проекты — отдельный вопрос. Fixed-price проекты вообще тяжело делать. Лучше работать и показывать результат заказчику, дразнить его при этом, то есть, говорить, нет, мы ничего не будем делать — тоже

неправильно. Ну, ты можешь делать собственные внутренние итерации, для собственного удовольствия. Такая же разница, как между сексом и стриптизом — ты можешь посмотреть, а потрогать нельзя. Не всех заказчиков это одинаково радует.

Что людям посмотреть/почитать/послушать про Agile?

Ну, я должен, наверное, сказать «приходите на тренинги». Заходите на сайт компании ScrumTrek (www.scrumtrek.ru), там есть какой-то набор тренингов. Ну и вообще, если вам самому это нужно, интересно – я вас всех приглашаю. Потому что, мне кажется, вопросы всегда одни и те же. Я на них отвечаю раз тысячу, наверное. Но когда приходит время внедрять, почему-то опять возникают серьезные трудности у многих. Не у всех.

Что касается книжек — по Scrum'у основная это Кен Швайбер Agile Software Development with Scrum. На русский она пока не переведена. Я рекомендую почитать на английском. Там очень интересные вещи написаны.

Из моих любимых книг я бы порекомендовал **User Stories Applied** by Mike Cohn и его же **Agile Estimating**. Вообще, Кон как автор – самый щедрый, он готов выложить все, что знает, книжки очень практичные в плане использования.

Что касается XP, наверное, самая практичная книжка – **Extreme Programming Applied** by Ken Auer. Она, кстати, переведена на русский. Она показалась мне самой разумной из того, что я читал.

Ну, я могу еще много книжек назвать, но зачем – все книжки все равно никто не читает…

Я думаю, хватит на первое время…

Да, начните книжек Майка Кона, они все хорошие.

Хорошо, вопросы закончились. Может быть ты хочешь сказать что-нибудь напоследок слушателям проекта Happy PM?

Чтоб такого сказать?.. Agile это просто практики, там надо голову подключать во всём, что вы делаете. Никогда не надо воевать с заказчиками, это всегда плохо заканчивается. Ты знаешь как это происходит – заказчик написал тебе что-то, ты думаешь: «Вот ведь гад, сейчас я напишу, и всем станет понятно, что он гад.» И ты написал такое письмо и выиграл раунд, и не заметил, как начал войну. И в этой войне уже есть убитые и раненые…

В этой войне ты обязательно проиграешь, ведь ты подчиненный. Если заказчик написал что-то резкое, и ты ему такой написал серьезное и злобное письмо — не отправляй письмо, отложи, потом перечитаешь.

Постарайся понять, почему он такое злобное письмо написал. И когда ты поймешь, что у него были какие-то риски, он чего-то боится, или не понимает, то ты попытаешься в своем письме отвечать не на то, что он написал, а на то, что он имел ввиду в тот момент, когда писал. Человек будет тебе очень благодарен, позже.

Надо любить людей вообще…

А не только заказчиков.

Да, не только заказчиков. И команду тоже. На твоем сайте много об этом говорится, как надо общаться с программистами, что они тоже люди. А я хотел сказать, что и заказчики тоже люди.

И когда мне кто-то говорит, что у нас неадекватные заказчики, разве можно делать Agile с неадекватным заказчиком, то я всегда вижу, что есть ситуация, в которой передается неадекватное количество информации. Недостаточно коммуникаций, кто-то чего-то не понимает. Что значит — неадекватный заказчик? Он принимает неадекватные решения? Почему он это делает? Либо у него недостаток информации, либо мы не понимаем, почему он принимает такие решения. Но раз он стал начальником и дает вам указания, значит он хотя бы немного адекватный, раз он хотя бы не в сумасшедшем доме сидит. Значит, с ним можно договориться, если понять, что он имеет ввиду.

И в этом помогает Agile.

Agile помогает налаживать коммуникацию. Agile, в некотором смысле — костыль. Если ты такой яркий, классный человек, у тебя получается налаживать коммуникацию с полпинка, и у тебя все работает безо всякого Agile — то зачем тебе Agile? Не бери его.

Agile это для нас, простых чуваков. Есть люди, которых я называю хардгейнерами — которым не так просто все достается. Я не Брэнсон и не Чичваркин, и мне нужны практики, которые мне будут помогать. У них и так все хорошо, а у меня нет. И когда я попробую применить эти практики, оно начнет получаться. Что ж теперь делать. Все эти книжки не помогут тебе стать номером один. Ты прочитаешь какой Чичваркин крутой, но это не сделает тебя таким же.

Практики помогают наладить коммуникацию.

Асхат, спасибо большое, по-моему, получился суперский разговор.

Тебе спасибо.