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

Автор: Александр Орлов (Happy-PM.com)

Текст: Алексей Лупан (TestItQuickly.com)

Часть I

Асхат, привет, спасибо за то, что ты согласился дать интервью проекту Happy PM. Расскажи твою личную историю о том, как ты пришел в менеджмент, стал самым известным Agile-евангелистом?

На самом деле, достаточно стандартно – в 2000 году закончил Московский Физико-Технический Институт с отличием, и думал – куда пойти-податься. Был неплохой шанс стать нефтянником, я в одном институте вел научную работу... И когда сдавал диплом, мне так показалось, я там неплохо выступил. И после сдачи вернулся в свою комнату, забрал что там у меня было, и мы пошли гулять. Мне потом рассказали, что через минуту туда защел какой-то очень солидный чувачок, при галстуке, и сказал, что он меня искал с тем, чтобы пригласить работать в одну нефтяную компанию. А я уже ушел. Я их потом чуть не убил, потому что никто не догадался спросить телефон.

И так получилось, что стал я программистом. Я попал в одну аутсорсинговую компанию. На самом деле, по-знакомству. Меня туда взяли младшим программистом, прошел довольно стандартный путь от младшего сотрудника до менеджера проектов. Какое-то время управлял тестированием. Сам представляешь – маленькая аутсорсинговая компания, технологии – все, роли – все, все разные... Большой был у нас интерес к разным методологиям, мы пробовали и RUP, и еще что-то, но до Agile, правда, не добрались.

Потом я попал в Luxoft. На самом деле, потому что выше некуда было расти. Выше был только директор, и большого смысла оставаться уже не было. Застой какой-то случился. И я пошел работать в Luxoft. Собеседовали меня на эксперта по .Net. Я был в шоке, я думал, какой из меня эксперт… Ну, я программировал на .Net относительно неплохо на тот момент, но, как мне кажется, до эксперта не дотягивал.

Меня успешно зарубили на интервью, задавая вопросы типа «сортировка»... На какие-то ответил, на какие-то нет... Ну в общем, всем было все понятно уже на интервью, какой из меня эксперт по .Net… Тем не менее – я, наверное, какие-то слова говорил в больших количествах – и меня потом Денис Запиркин собеседовал на инженера по процессам. Он тогда был начальником отдела SEPG. И я там как-то собеседовался

успешно, и стал инженером по процессам, стал заниматься всякими процессами.

Начал с процесса измерений проектов. Обычно заказчики хотят, чтобы их проекты каким-то образом измерялись...

Что вы меряли, если не секрет?

Ну, если ты «большая и толстая» организация, то тебе хочется получить метрики, чтобы обеспечить прозрачность процесса. Метрик очень много, например, связанных с качеством – это плотность дефектов на тысячу строчек кода…

Скорость устранения дефектов…

Да, там с десяток параметров и все они полезные. Производительность кодирования – количество строк кода, которые пишутся за один день. Сколько их было – мы в итоге свели их в экселевский файл, в котором было в районе двухсот метрик. Но они, на самом деле, все не нужны. Из них менеджеру проекта нужны в лучшем случае три, и то, если он умеет с ними работать.

И то, если хорошо подумать, то и они не нужны…

Вот именно. Ты, как менеджер, и так знаешь, что происходит на проекте, и метрики тебе не сильно помогут.

Другое дело, если у тебя огромный проект, работают тысяча человек – действительно, какие-то метрики там нужны. Но таких проектов было очень немного, прямо скажу – ни одного, в котором было бы больше тысячи человек. В относительно крупных проектов это хоть как-то использовалось, а в остальных проектах это был чистый overhead.

Короче говоря, с измерениями дело обстоит так: не все из них одинаково полезны, и их полезность сильно зависит от того, умеешь ты их использовать или нет. Реально немногие менеджеры проектов умеют это использовать, но мы же не об этом разговариваем.

Значит, о процессах. Я там краем задел процессы управления проектом, и еще какие-то процессы, в которых я начал более менее профессионально разбираться на уровне чтения тренингов проектным командам и товарищам внутри Luxoft. А потом у нас появился один заказчик, очень хороший, и сказал слово Agile, в первый раз сказал «Agile». Мы, конечно, все напряглись (в хорошем смысле), потому что было безумно интересно. Съездили несколько раз к заказчику. Это было безумно интересно. И мне показалось, что именно это – правильно.

В тот момент мы сталкивались в работе с серьезными трудностями. Ты весь такой из себя умный приходишь в команду, и говоришь «Ребята, я вас сейчас научу жить!», а все тебе говорят: «Ты кто такой вообще? Зачем

пришел? Мы и до тебя жили, и после тебя будем жить...» А если получаешь административную поддержку, то всё еще хуже – в том смысле… В котором говорил Некрасов про русский народ – вынес и эту дорогу железную, вынесет все, что Бог ни пошлет...

И ты чувствуешь, что, конечно, ты такой умный весь, но вроде твоя работа не особо кому-то нужна. А тут такой Agile и улучшение процессов самой командой. И я загорелся. А когда я загораюсь, я обычно стараюсь рассказать о том, что мне было интересно... Я там докапывался до всего, всем рассказывал. Начальнику своему много об этом рассказывал, и книжки читал, стал местным теоретиком по этому делу. И когда руководство вдруг захотело узнать — что же это за штука Agile — меня попросили, чтобы я им рассказал. Я пришел и рассказал, ну и «засветился» в нужном месте в нужное время. А потом был внутренний проект по Agile, меня туда поставили представителем заказчика.

Мы там наступили на все грабли, на которые только можно было наступить. Просто на все я наступил. Когда я говорю о том, чего нельзя делать — это все оттуда. Кажется, все, что не надо было делать, я успел там сделать... Конечно, все более-менее вырулил, но получил очень полезный опыт. И занялся в Luxoft тем, что помогал методологической поддержкой командам и заказчикам рулить процессом. Это было очень полезно.

Через какое-то время обнаружил себя в компании, которая профессионально занимается методологической поддержкой всяких организаций и компаний, помогает им решать проблемы.

Обнаружил себя в собственной компании?

Я не считаю, что я в этом сильно поучаствовал, оно как бы само получилось. Хотя мне очень хотелось, и конечно, часть моей вины в этом есть.

Отличная история. Давай крупными мазками пробежимся по мега-плюсам гибких подходов. Чем они особенно хороши?

Очень сильно зависит от того, что у тебя болит. К доктору приходишь и спрашиваешь – а чем хороши ваши врачебные предписания? И доктор должен ответить. А это очень сильно зависит от того, что у тебя болит. Там много разных практик, они для разных целей предназначены...

Первое – короткие итерации. Евангелисты Agile говорят, что короткие итерации нужны для того, чтобы получать обратную связь от рынка. Что-то происходит, и чтобы узнать, что надо поменять в твоем программном обеспечении, нужно иметь эту связь. Чтобы показать софт кому-то, чтобы он попробовал, и ты развиваешь короткими итерациями продукт, и в конце каждой итерации имеешь полностью рабочий софт. Это снижает огромное количество рисков, связанных с интеграцией, с получением обратной связи, да и просто с видимостью прогресса, с планированием...

Потому что если оно у тебя полусобранное, то тебе надо всякие взаимосвязи учитывать при планировании. И ты там рисуешь диаграммы Гантта. А так у тебя полностью собранный софт. Когда ты планируешь следующую итерацию, ты просто вставляешь какой-то набор фич, которые тебе нужно сделать, и нет геморроя с планированием.

Дальше что? Какие-то вещи, связанные с повышением мотивации команды. С тем, чтобы команда у тебя была более сплоченной, чтобы она умела эффективно решать поставленные перед нею задачи. В Agile это особенно важно, потому что у тебя короткие итерации, а в конце итерации должен быть полностью работающий софт. Соответственно, твой инструмент как менеджера — у тебя есть два варианта как работать с командой: ты можешь персонально бегать между ними, а можешь использовать документы типа диаграммы и все такое прочее. На самом деле тебе нужно чтобы они коммуницировали друг с другом. Потому что за одну итерацию лично передавать информацию — очень трудно. Тебе придется очень сильно бегать, если ты сам будешь лично все передавать.

Я много бегал, поэтому в курсе…

Да, ну представь себе — у тебя три программиста. И у тебя три модуля. Ты запланировал свою итерацию так, что у тебя три человека копают три дорожки. А потом — раз, и наступил конец итерации. А наступил он, сам понимаешь, неожиданно. И у тебя все три дорожки недокопаны до конца. Ты приходишь на демонстрацию заказчику, и тебе вообще показывать нечего. Ну, то есть, у тебя есть какие-то классы, но их показывать неинтересно. Интересно показывать работающий софт.

То есть, тебе нужно работать так, чтобы все три копали одну грядку. Потом взялись за вторую, а потом за третью. Тогда получится, что даже если ты что-то не успеешь — это будет только одна грядка (причем это будет самой неприоритетной грядкой, которая тебе и не нужна); а самые приоритетные — с клубникой — с ними все в порядке. Но для того, чтобы они все вместе копали одну грядку тебе нужно чтобы они намного больше сплелись, чем если бы каждый копал свою. Понимаешь?

Вопрос жизненный: что делать, когда землекопы — разной квалификации?

Они должны пообщаться. Мамы каждые нужны, мамы разные важны. Не бывает ведь, что есть один мега-человек — лучше всех во всём. У тебя есть, как правило, специалисты разного уровня и в разных областях компетенций. Например, есть один спец по базам данных, один спец по Java, и один спец по User Interface. Ну, по крайней мере, предполагается, что ты так команду собрал. Ты же не собрал туда одних идиотов? Или тех людей, которые знают только одну область — все трое специалисты только по базам данных — так же не бывает. Не бывает?

Бывает... В большой компании вообще всякое бывает.

Ну, неважно. Идея в чем — у тебя есть разные специалисты в разных областях. И все что тебе нужно — чтобы они целиком выполняли ту работу, которая у вас есть. Тогда они разберут ее сами. Ну, если есть самоорганизация, то, чего мы добиваемся. Вот — есть задача, сели вместе и подумали, как ее решать самым эффективным образом.

Конечно, знание куска модуля или знания каких-то технологий у одного могут быть больше чем у других в этой, конкретной задаче. Ну, значит, он там будет самым главным, он всем поможет, всех обучит, все будут работать под его руководством.

Вот. Сели вместе и сделали работу, которую обязались сделать.

Рано или поздно люди станут достаточно кросс-функциональными...

Да, это то, к чему мы стремимся. Они по-английски называются generalists. Люди, которые могут делать не все (все – это безумие), а все, что нужно для достижения целей проекта. До какой-то степени.

То есть, тебе не нужны спецы, которые знают все обо всём. Было бы супер, но не так уж это возможно. Но до какой-то степени они могут решать любые задачи. Потому, что если этого не будет, то ты лишаешься следующего: наступает следующая итерация, тебе надо ее спланировать, и тебе нужно там три важных фичи сделать, они все про один модуль — ну, так бизнес распорядился, так на рынке сложилось — ты как product owner, представитель заказчика, приходишь и говоришь: «Слушайте, ребята, давайте вот это сделаем». А тебе говорят: «Ни фига! Потому что у нас есть только один спец, который знает этот модуль и может эту фичу копать. А он в отпуске. Дай-ка нам что-то ненужное, мы его сделаем». А в нашем с тобой бизнесе, если у нас рынок диктует то, что нужно, то делать ненужное — это очень серьезная потеря денег. Мы просто теряем на этом деньги. Вернее, мы не зарабатываем деньги, которые могли бы заработать, если бы делали то, что нужно. А вот это уже никого не устраивает. И это надо менять.

Поэтому – generalists. Но опять — это все до какой-то степени. Кто-то может подумать, что это фигня, это все невозможно, ведь у меня есть спец по Oracle, и чтобы этому обучиться — нужно десять лет... Ну, супер, если он у вас есть. Он вам поможет решать вопросы, просто — кто чем будет заниматься — это команда это решает, а не менеджер. В чем разница? В том, что мы, хотим чтобы вся команда отвечала за весь результат. А для этого нужно, чтобы мы смогли уничтожить отношение к делу типа «моя хата с краю, я маленький программист, я тут написал, а что не работает — виноват не я». От того, как эффективно мы взаимодействуем, зависит то, насколько качественно получается продукт. Иначе будут дефекты, а это потеря денег. Чем позже баг обнаружен, тем дороже он стоит.

Еще — мы же рассматриваем ситуацию в динамике. Мы хотим, чтобы люди учились, чтобы они могли самосовершенствоваться, потому что

через два года, когда продукт у нас будет взрослый и большой, у нас уже будут взрослые и большие программисты. И они будут лучше разрабатывать софт. То есть, они смогут обучиться. Научиться друг у друга. Друг друга обучить.

То есть — это супер, когда в команде все разные. Тебе не нужны одинаковые люди в команде — так и не бывает. Из чем более разных людей собрана команда, чем более разный у них опыт, тем более эффективно они будут работать. Это касается не только компетенции в каких-то областях — БД, например. Это касается вообще психологического состояния людей. Чем более они разные, тем более эффективно они будут работать друг с другом.

Дальше пойдет как пойдет — они будут учить другу друга тому, что нужно для проекта. Это будет очень здорово.

Про командную ответственность — есть такая проблема у многих менеджеров: они любят когда за что-то отвечает один человек. Они хотят его лично вызывать на ковер и спрашивать, когда все будет готово. А командная ответственность, когда все отвечают за проект — это значит, что никто толком не отвечает.

Ага, есть такие менеджеры.

Что ты можешь на это ответить?

Ответственность — это когда ты отвечаешь за то, что делаешь, но при этом у тебя есть соответствующие полномочия. Согласен? То есть, ты имеешь право принимать решения. Ответственность — это в первую очередь право принимать решения. Тогда ты можешь внутренне взять на себя ответственность.

Если человеку говорят — чтобы к пятнице было готово, ты же ответственный человек, ты должен сделать. Ну какая тут к черту ответственность? Его же никто не спрашивал. Он ответственность не взял — внутренне. А даже, если взялЮ то есть, если ты такой хитрый менеджер, что сумел это сделать манипулятивными практиками — он взял на себя ответственность — это ложное чувство ответственности, это приводит к демотивации.

А выглядит это так: чуваки просто говорят, что тут менеджмент странный, и вообще никому ничего не нужно, никто ни за что не отвечает, и вообще в компании бардак. Это симптом. К каким проблемам это приводит в разработке ПО? У тебя есть ответственный программист, и ты хочешь, чтобы он ответственно относился к своей работе. И ты поручаешь ему модуль, и он в рамках этого модуля несет за него ответственность. Мы уже говорили, что это не всегда полезно, нам нужно, чтобы люди взаимодействовали.

Например, я пишу код, и у меня все хорошо, но из-за того, что я что-то поменял в коде, у кого-то другого что-то свалилось в его коде. Я не чувствую себя виноватым, потому что в моем коде все в порядке. Может, я ему помогу, но сделать так, чтобы у него больше не падало — мне не очень интересно. Я отвечаю за свой кусок. И с этим связаны определенные проблемы. Потому что нам хочется, чтобы дефектов было поменьше, особенно связанных с взаимодействием между различными кусками программы. Чтобы все хотели, чтобы весь продукт работал хорошо как единое целое.

Откуда у человека берется желанием помогать Васе с соседним модулем?

Вот именно, откуда его взять? Надо, чтобы за весь продукт отвечал не менеджер, а Вася, этот конкретный Вася. Понимаешь? А не только и не столько за свой кусок. А как это сделать?

Единственный возможный способ это сделать — дать ему возможность принимать решение относительно всего продукта, а не только своего модуля. Тогда он хоть как-то начнет нести за него ответственность.

К чему это приводит? Естественно, решение надо принимать коллективно, а не персонально. Каждый не может принимать решение за весь продукт. Поэтому решения принимаются коллективно. Вся команда вместе решает, как мы это будем делать и развивать. И тут возникает та самая ответственность. Коллективная ответственность невозможна без того, чтобы люди принимали коллективные решения.

И второе, без чего она невозможна — без прозрачности. Например, когда у тебя есть один чувак, и он отвечает теперь за весь продукт, но на самом деле делать, что хочет, и никто не знает, чем он занимается. Ну, какая же это ответственность? Я могу ничего не делать, и ни к чему плохому это не будет приводить. Поэтому нужно сделать так, чтобы была полная прозрачность, чтобы все понимали, что происходит. Это Agile тоже обеспечивает — daily scrum'ы, task board, куча существует практик, которые помогают каждому понять, кто чем занимается.

И до некоторой степени ребята, которые говорят, что коллективная ответственность, это когда никто не отвечает – они правы. На каком-то уровне должна быть персональная ответственность. Мы коллективно отвечаем за итог целой итерации и персонально каждый отвечает за свою задачу. У нас есть task board, где висят наши бумажки с задачами, и эти бумажки принадлежат всей команде. Я беру бумажку с задачей, перекладываю ее в сторону “In progress”, и подписываюсь под ней — что теперь за нее отвечаю я перед командой. Без этой персональной ответственности, без прозрачности не будет никакой коллективной ответственности.

Ну, и размер команды еще важно — она должна быть небольшая. Если больше 12-ти человек, уже будет трудно все это сделать.

Хорошо. Тогда вопрос больше методологического плана. У нас была проблема в одном из проектов — мы отдавали на аутсорсинг некоторую задачу. У исполнителей тоже были итерации. И очень нашего начальника интересовало — а что будет сделано через три месяца? На что ребята говорили: «Мы ничего больше чем на три недели не планируем, поэтому когда останется три недели до конца трех месяцев — мы вам скажем».

Долгосрочное планирование — как это делается в Agile?

Идея какая – у тебя есть релиз, больший чем на одну итерацию. На три месяца, или на полгода, например. И ты на этот срок в принципе можешь планировать. Ты разбиваешь всю свою работу на небольшой набор фич (или юзер-сториз — зависит от того, как ты это делаешь). Они оцениваются командой — садятся все вместе, и все голосуют, сколько чего займет — но не времени, а условных единиц (стори-пойнтов) — в попугаях. Берут одну из самых маленьких задач, обозначают ее, например, двойкой. Все остальные ранжируют по тому, сколько они занимают времени по отношению к этой. Что в два раза сложнее — это четверка. Если где-то между двойкой и четверкой — ну, видимо, будет тройка. Так мы получаем условные единицы (стори-пойнты) для каждой фичи. Потом мы можем прогнать одну итерацию, к примеру, трехнедельную, и посмотреть, сколько стори-пойнтов мы успеваем сделать за одну итерацию. Всё, можно знать примерно, когда мы закончим все эти фичи. Конечно, конец релиза надо оставить на стабилизацию, никаких новых фич не вносить, и какой-то рисковый буфер добавить.

Единственно, что стори-пойнты вызывают у некоторых ступор — пока не попробуешь, действительно кажется, что все это очень странно... Нужно попробовать, тогда все получится.

Многих людей с богатым опытом тестирования интересует — где в Agile, в SCRUM — где там тестирование? Как оно туда вплетается?

А какие сложности? Что именно непонятно?

Юнит-тесты — это понятно. А системные тесты, нагрузочные, измерение покрытия, оценка качества тестирования — это как-то приплетается к Agile?

Конечно. Давай самое главное поймем про тестирование — оно постоянное. Мы тестируем не в конце, а постоянно. Потому что в конце итерации должен быть полностью протестированный, в идеале, софт. То есть, тебе нужно протестировать каждую фичу и сделать полное, системное тестирование приложения. Как ты это будешь делать? Ну, в Agile рекомендуется автоматизировать всё. У тебя есть автоматизированные

приемочные тесты, ты нажимаешь кнопочку, уходишь обедать, возвращаешься — и у тебя есть результаты прогона тестов. Это в идеале.

На практике, у тебя обычно есть и legacy code, и непонятно, как это делать, когда много визуальной информации и минимум бизнес логики, что не особо покроешь тестами и т.д. То есть, в реальности — конечно, объем ручного тестирования может быть достаточно высоким. Но это означает, что ты будешь тестировать вручную... Раз в какое-то время, в зависимости от того, сколько тебе нужно времени на регрессионное тестирование, забираешь билд, мучаешь его как хочешь, и у тебя есть список дефектов. Эти баги ставятся на следующую итерацию. Перед тем, как забрать билд на тестирование, ты проводишь build acceptance test (или smoke-тесты их еще называют) — просто проверяешь, что билд находится в рабочем состоянии, и что его можно тестировать, что там нет багов, которые являются show-stopper'ами для тестирования.

Это может быть небольшой набор тестов. В идеале, но реально это не всегда получается Если не получается — имеет смысл выписать для этого отдельный тест-план и начинать с него. Первое тестирование — оно неполноценное, а только по какой-то части, smoke-тестами. Чтобы ты мог его делать за час — каждый день, проверять, что у тебя основные вещи не отвалились.

Баги ставятся в план на следующую итерацию, приоритезируются как полагается, product owner'ом. Только вот — баги не надо копить, надо с ними что-то делать. Те баги, которые являются enhancement — ставятся в план. А те баги, которые действительно баги - ну, надо стараться, чтобы их не было. Они имеют достаточно высокий приоритет и исправляются в первую очередь.

А если работа по тестированию — написание фреймворка, или написание пары сотен тестов... Это на таск-боарде висит отдельными задачами?

Да, совершенно верно, это задачи, которые принадлежат команде. Команда должна их выполнять. Не тестировщики, а кто-то внутри команды. Конечно, если тестировщики умеют писать нагрузочные тесты, скорее всего, он их и напишет. Но если говорить об интеграционных тестах, то это может быть задачей программистов. Это логично — оно висит на таск-боарде, подходишь к таск-боарду, видишь, что висит задача по написанию интеграционного теста, ты ее просто перевешиваешь и подписываешься под ней.

Что мы уничтожаем? Уничтожаем отношение к тому, что это вот мое дело, а это — не мое. И работа у нас общая.

Да, да, это очень позитивно... Потому что пинг-понг между тестовыми командами и разработчиками — это боль.

Как в фильме «Бригада» - мы в этом деле вместе, и отвечать тоже будем вместе. Если помнишь, Саша Белый как раз все и сломал за счет того, что он начал тянуть одеяло на себя, брать на себя ответственность, никому ничего не говорил, и у них из-за этого пошли ссоры, потому что люди перестали друг другу доверять, из-за того что не было прозрачности.

Саша Белый про Agile ничего не знал...

Да! Вот если бы он знал про Agile, он бы знал, насколько важна прозрачность, у них были бы ежедневные SCRUM-митинги, в 11 утра, они бы встречались, и каждый рассказывал бы, что он делал вчера, что он будет делать сегодня, и какие у него проблемы. Они бы помогали друг другу. А так Саша там что-то мутит-крутит, Пчел там тоже, Кос наркотиками занимается…

А вот еще вопрос применительно к «Бригаде» - про денежные компенсации. Когда команда что-то сделала хорошо, и дают команде бонус. Мы предполагаем, что награждение тоже командное. Как этот бонус распределяется внутри команды? Как это надо делать с твоей точки зрения?

Чего мы должны избежать — это всяких конкуренций внутри команды. Первая проблема в том, что тебе это не нужно. Ну, будет известно, что мне дали меньше денег, чем Васе. Васе теперь не то что руки не подам — бог с ним, на виду буду корректно себя вести, но в следующий раз он придет, скажет «Асхат, помоги мне», я про себя подумаю «я такой добрый, хожу, всем помогаю, а потом меня из-за этого бонусов лишают». Тут даже не деньги важны, а message, сообщение. Ты даешь своей команде такое «Сообщение»: Чуваки, не помогайте друг другу! Каждый сам за себя...

Второе — не существует никакого объективного способа посчитать эффективность каждого человека объективно – так, чтобы он не обиделся. Ты никогда не придумаешь правильного «потому, что». Не существует даже комплекса таких метрик. Почему? Объем кода? Нет, конечно. Количество дефектов? Я могу не писать код, и не будет у меня никаких дефектов. Сроки исполнения? Ну, если я называю сроки исполнения, то я умножу их как-нибудь на три или на пять, чтобы как-то себя защитить. Вдобавок, меня наказывают не за то, что я соблюдаю сроки, а за то, что я оптимистичный товарищ. Был ли я при этом вреден для проекта? Совершенно не факт. Может быть, я оптимистичный от природы, а кода пишу много и хорошо — меня тут обижают...

Даже то, что ты видишь как менеджер проекта — этот чувак работает хорошо, а этот — плохо, - а вдруг он душа команды? И благодаря ему есть коммуникация в команде, все сплоченные? Это то, что нельзя никак измерить, но это очень важно для команды.

Поэтому — не паримся, и всё. Если бонус есть — он командный. Каждый получает, например, 50% от зарплаты в бонусный день. Либо

награждается команда целиком, либо нет. Деньги делятся пропорционально зарплате, но без дальнейшей дифференциации, чтобы мы не нарушили командность.