КАК ХОРОШЕМУ РАЗРАБОТЧИКУ НЕ СТАТЬ ПЛОХИМ МЕНЕДЖЕРОМ

КОНСТАНТИН БОРИСОВ

18+

Константин Евгеньевич Борисов

Как хорошему разработчику не стать плохим менеджером

http://www.litres.ru/pages/biblio_book/?art=51337677
SelfPub; 2020

Аннотация

В этой книге автор, сам прошедший путь от разработчика до менеджера в сфере IT, рассказывает неочевидные моменты, которые являются критически важными для правильного управления. Почему разработчики увольняются после повышения зарплаты? Как делать FixedPrice проекты? Почему Scrum не упрощает менеджмент? Книга содержит ответ на эти и многие другие вопросы. В книге есть много баек, которые показывают тяжёлую, но интересную жизнь менеджера в разработке. Иллюстратор обложки: Ксения Ерощенко. Иллюстрации в тексте книги авторские.

Содержание

Про эту книгу

За свою жизнь я встречал многих классных руководителей, от которых перенял бесценные знания. Но видел и многих менеджеров, которые допускали грубейшие ошибки, и работать с которыми было мукой для команды и для заказчика. Многие люди (как специалисты, так и менеджеры) не понимают, что вообще делает менеджер в IT, какие навыки он должен иметь, и какие ошибки он ни в коем случае не должен допускать.

Бизнес-литературы по менеджменту много, но этот "менеджмент" в области разработки программного обеспечения часто оказывается бесполезен или даже вреден.

Почему разработчики увольняются, хотя их зарплаты постоянно растут? Как делать Fixed Price проекты? Почему после внедрения Scrum'а управлять проектами не стало проще? Ответы на эти вопросы каждому приходится искать методом проб и дорогостоящих ошибок.

В этой книге я постарался дать самое главное, что нужно менеджеру – понимание происходящего. Я пишу про то, как нужно мотивировать команду, как по-разному могут работать деньги в разных ситуациях, как нужно смотреть на проект, чтобы видеть его проблемы. Я пишу именно про те вещи, которые понять мне самому оказалось труднее всего, и знание которых оказывалось критичным для меня самого.

Я часто слышал, что знания без опыта не работают, и в этой книге я передаю и свой опыт тоже. Около половины книги занимают истории из жизни[1] и кейсы с разборами. Эти истории содержат реальный опыт, и позволяют вам не делать ошибки самому, а наблюдать за ошибками других.

[1] Во всех историях я постарался изменить обстоятельства и имена героев, чтобы не нарушать право людей на личную жизнь. Если вам кажется, что вы узнали себя или кого-то другого в какой-то из историй, то уверяю вас, вам только кажется. Если вы хотели бы поделиться какой-то своей историей (или задать вопрос), то вы можете связаться со мной через почту borisovke@gmail.com.

Про автора

Автор этой книги, Константин Борисов, имеет двадцатилетний опыт работы в индустрии разработки программного обеспечения. Участвовав в десятках проектах в российских, зарубежных и международных компаниях в роли разработчика и менеджера, он накопил опыт, которым и делится в этой книге.

Связаться с автором можно с помощью электронной почты borisovke@gmail.com.

Личный блог автора доступен по адресу https://bukovka.livejournal.com/

Адреса в социальных сетях: ВКонтакте, Facebook

Создание обложки: Иллюстратор Ксения Ерощенко artbylulutyan

Раздел 1. Общие вопросы менеджмента

Менеджер взаимодействует с разными людьми: заказчиками, топ-менеджментом своей компании, своей командой, представителями других компаний. И часто все эти люди очень по-разному смотрят на процесс разработки. Чтобы выполнять свои задачи, менеджер должен обладать широким взглядом и видеть ситуацию не просто под разными углами, а в целом. Поэтому для начала давайте поговорим о менеджменте в IT в общем.

Особенности менеджмента в IT

Индустрия разработки программного обеспечения испытывает жуткую нехватку менеджерских кадров. Эта нехватка даже более выражена, чем нехватка разработчиков, тестировщиков, аналитиков и других исполнителей. Отчего так происходит? Дело в особенностях индустрии, которые требуют особенных (редких) менеджерских навыков и знаний. Давайте рассмотрим эти особенности.

1. В IT работают высококлассные исполнители, которые должны принимать решения сами.

Специалисты высокого уровня встречаются много где, но, пожалуй, только IT не имеет никаких протоколов и инструкций, и полагается на экспертные решения. Технологии, методологии и бизнес-задачи слишком часто меняются, чтобы выработать какие-то реально работающие стандартные подходы. Да, многие компании имеют некоторые документы, но они описывают второстепенные вещи, а не основную работу. Описывается, как быстро надо ответить заказчику, но не описывается, что именно надо отвечать. Указывается процент покрытия юнит-тестами, но не описывается, как именно нужно реализовывать определённый функционал.

Наверное, сейчас такое описание и невозможно. В результате менеджер должен уметь работать в ситуации, когда решения разного уровня принимаются без его участия, но ему

нужно не терять контроля над проектом. Делегирование менеджерских обязанностей используется необычайно широко. Настолько развитое делегирование характерно разве что для топ-менеджеров компаний, но они обычно гораздо дальше от конечной реализации продукта, чем менеджеры в IT.

2. Исполнители крайне своевольны, не терпят хоть сколько-нибудь жёсткого обращения и требуют очень аккуратного управления.

Следствием предыдущего пункта и ситуации на рынке труда является то, что работники в IT требуют очень нежного менеджмента притом, что сами они ведут себя часто весьма агрессивно.

Например, только в IT менеджер сталкивается с тем, что его прямой и простой приказ не только не исполняется, но и открыто оспаривается. Для менеджера из другой области это шок, а для IT-менеджера – обычная реальность. Бесполезно говорить опытному разработчику, как именно он должен реализовать конкретный код. Его можно убедить, но им нельзя командовать.

А если менеджер сорвётся и наорёт на разработчика, тот молча напишет заявление на увольнение. Опытный специалист найдёт работу в течение нескольких дней.

Взаимодействие менеджера и подчинённого в IT – это разговор на равных, когда договорённости и взаимное уважение используются очень широко, а прямые приказы не используются вовсе (ну, почти).

3. Высокий уровень рисков.

В IT менеджеру недостаточно разбираться с возникающими проблемами, ему необходимо проблемы предугадывать. Обязательно нужно вести реестр рисков и для каждого из рисков продумывать стратегию. Все проекты высокорисковые, так что всё равно срабатывает множество предвиденных и непредвиденных рисков, но если не вести реестр рисков, то проект непреодолимо скатывается в хаос.

4. Иностранные заказчики.

Знание английского языка является самым незначительным требованием при работе с иностранными заказчиками. Гораздо важнее, чтобы менеджер понимал культурные особенности и разницу в принципах ведения бизнеса.

Типичный российский менеджер не ожидает, что его слова в обычном ежедневном письме имеют вес, сравнимый с заключённым контрактом, а буквальное выполнение контракта не достаточно, если заказчик остался недовольным.

5. Техническая сложность проектов.

Это, пожалуй, наименее значимый фактор, но он отпугивает наибольшее число менеджеров не из IT. Чтобы эффективно работать с программными проектами, нужно постоянно учить хотя бы поверхностно множество технологий, принципов и конкретных систем. К этому большинство менеджеров не готовы. Поэтому чаще менеджеры в IT получаются из опытных специалистов: разработчиков, тестировщиков, аналитиков.

Если подводить итог, то можно просто сказать, что даже рядовой менеджер в IT должен иметь много навыков, которые обычно имеют только менеджеры очень высокого уровня в других областях. А, кроме того, он должен иметь большой багаж знаний, характерных конкретно для IT.

Из этого есть два следствия. Первое: в IT вы можете работать с удивительно классными профессионалами. Я с очень большой теплотой вспоминаю менеджеров, с которыми мне довелось работать самому. Многих из них я считаю своими учителями. Я вспоминаю и обдумываю то, как они работали, и это помогает мне и в работе, и в жизни. Когда я сам становился менеджером, я грел себя мыслью, что я смогу вырасти в такого же профессионала, как те, под началом которых я сам работал.

Второе следствие: очень много менеджеров не дотягивают до нужного уровня. Трудно быть суперменом и быть прокачанным во всех областях, а в реальных проектах все проблемы взаимосвязаны. Например, если менеджер недостаточно знаком с техническими особенностями используемых технологий, это может привести к срабатыванию неизвестных рисков, что вызовет недовольство заказчика, с которым трудно справиться, если не знаешь особенностей культуры заказчика. Всё это приводит к напряжённости внутри команды, которая будет только нарастать как снежный ком, если у менеджера недостаточно прокачан эмоциональный интеллект. В результате менеджеру кажется, что весь проект просто рас-

сыпается на части. А команда при этом вообще не видит, что от менеджера есть какая-то польза.

Я написал эту книгу как раз, чтобы помочь менеджерам (действующим и будущим) оказаться не во второй категории, с которыми никто не хочет работать, а в первой категории людей, знакомством с которым гордятся.

Почему проекты заканчиваются неудачей

В IT абсолютное большинство проектов завершается превышением бюджета, срывом сроков или нереализацией планируемого функционала. Почему так? Причин несколько.

1. Разработка ПО сейчас очень дёшева и заказчики хотят её такой оставить. Хотя разработчики имеют высокие зарплаты, а компании, разрабатывающие ПО, получают хорошую прибыль, но с каждым годом разработка ПО дешевеет. Один единственный самолёт Boeing 777 стоит больше $300 миллионов. Бюджет даже крупных проектов по разработке ПО составляет малую часть этой суммы. А самолётов серии Boeing 777 на данный момент выпущено более 1600. Стоимость ПО ничтожна по сравнению с общей ценой изделий.

В цифровых продуктах аналогично. Instagram в 2012м году был приобретён Facebook, и примерная стоимость сделки составила $1 млрд. Стоимость собственно разработки несравнима с этой суммой. Стоимость компьютерной графики фильма “Миссия невыполнима – 2” не идет ни в какое сравнение с гонорарами Тому Крузу. Хотя его можно было бы просто нарисовать.

Если в прошлом программные комплексы разрабатывались целыми институтами, то сейчас скрам-команда в девять человек уже считается большой. Чтобы грамотно проанали-

зировать, задокументировать и спроектировать систему, стоимость проекта нужно удвоить. А зачем? Тогда можно просто начать реализацию системы, и, если что-то пойдёт не так, то переписать её. Это даже надёжней, так как от ошибок анализа и проектирования тоже никто не застрахован.

Справедливо считается, что исправление ошибок на более поздних этапах разработки очень дорого. Это правда, но вся индустрия разработки с каждым годом всё сильнее сглаживает эту проблему. Сейчас разработка модульная и даже, если основной модуль был спроектирован неверно, то не нужно переписывать всю систему. Большую часть кода можно сохранить.

Но деньги сами по себе не так важны, как следующий пункт.

2. У заказчика нет времени на уточнение требований. Изменение требований в процессе разработки ПО сейчас настолько распространено, что The Standish Group изменила критерии провала проекта для своего Chaos Report. Если клиент доволен, то считается, что проект успешен, несмотря на то, что сделали не то, что планировали.

Почему так? Потому что заказчикам нужно менять систему ещё до того, как она будет сделана. Поэтому тратить дополнительное время на анализ и проектирование бессмысленно, требования всё равно поменяются.

Итак, заказчик не даёт денег на предварительный анализ системы, да ещё и меняет требования в процессе. Шансов,

что более-менее сложный ~~праехт~~[2] проект пройдёт по плану абсолютно никаких.

Хорошая новость заключается в том, что заказчики это понимают (обычно). Большинство заказчиков имеет возможность увеличить бюджет или убрать часть функционала, чтобы проект таки принёс им какой-то осмысленный результат.

Плохая новость заключается в том, что даже с понимающим заказчиком менеджер не может просто расслабиться и позволить проекту катиться в произвольном направлении. В случае провала проекта заказчика очень интересует, что пошло не так. Это не поиски виноватого, а здравый смысл. Если в процессе разработки выяснились новые требования или выбранные технологии не подошли к задаче, то понятно, куда двигаться дальше. Вторая итерация проекта будет нацелена на исправление того, с чем не справилась первая итерация. Таким образом, провал проекта принесёт важную информацию и будет ступенькой для реализации системы.

Совсем другая ситуация, когда проект провален из-за низкой квалификации разработчиков или, ещё хуже, когда проект провален, и никто не понимает, почему. В этой ситуации провал проекта – это просто потраченные деньги. Ни один заказчик такого не потерпит.

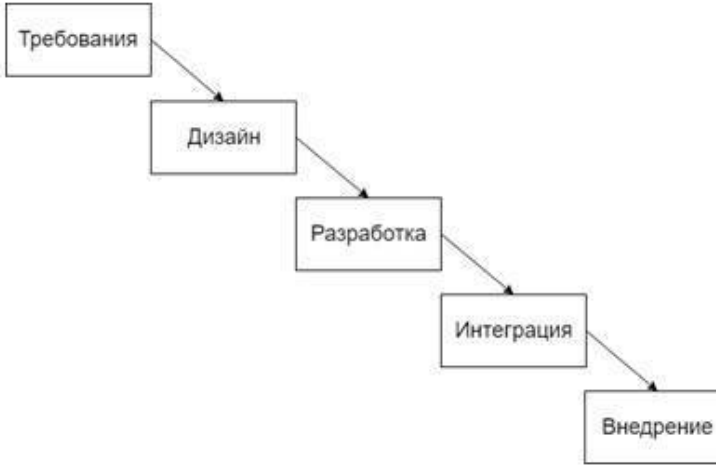
---

[2] ~~Если вы нашли в этой книге какую-то ошибку, пожалуйста, сообщите мне на почту borisovke@gmail.com, чтобы я мог её исправить в будущих изданиях.~~

Водопадная модель разработки

Периодически слышу от разных людей сожаление, что они используют водопадную модель разработки: “Мы бы и рады использовать Agile, но заказчик против”, “У нас водопад, мы к нему привыкли”, “Мы готовимся перейти на гибкие методологии, но пока у нас водопад”. Такие разговоры меня удивляют, так как встретить сейчас чистую водопадную модель разработки практически нереально.

Чтобы выяснить, почему так, давайте вспомним, что такое водопад как методология разработки, и как связаны разные этапы разработки:



Все этапы идут один за другим. Следующий этап начинается после полного окончания предыдущего этапа. Это очень-очень старая модель. Её название пошло из статьи Уинстона Уокера Ройса, опубликованной в 1970м году. Юмор заключается в том, что в той статье Ройс говорил об устарелости и ограниченности этой модели и о необходимости перехода на итеративные модели. То есть “водопад” – это то, как разрабатывали программы в 60-е годы.

Нам сейчас даже трудно представить, как это было, но давайте попробуем. Вот у какой-то компании есть нужда в какой-то программе. Она оплачивает анализ требований какому-нибудь проектному институту. В результате получает

вагон требований (буквально железнодорожный вагон документации), который принимается и подписывается. Эта документация потом направляется в другой проектный институт, который уже делает дизайн, описывает, какое оборудование и какие программы нужны для реализации задачи. Опять, весь результат оформляется, принимается и подписывается. Для реализации документация направляется по нескольким другим компаниям, которые разрабатывают аппаратно-программные комплексы. На общую задачу им плевать, они работают по документации и производят не только код, но и кучу другой документации. На этапе интеграции ещё одна компания объединяет все эти разработанные куски в единое целое и только тогда начинается внедрение (отдельной компанией или департаментом).

Что делать, если заказчик на этапе интеграции захотел изменить требования, добавить отчёт? Ничего не сделаешь. В принципе отсутствовала такая опция в те далёкие времена. Можно было дождаться полной имплементации и начать новый проект по реализации этого отчёта. Либо прекратить все работы и начать всё снова. Нельзя попросить институт, который писал требования, их изменить. Потому что это физически вагон бумаги, который уже ушёл от них. И по той документации уже что-то сделано и компании не будут ничего переделывать, так как у них в контракте описана работа по изначальной версии задания и всё. Даже просто разорвать эти контракты и остановить работы часто было невоз-

можно, так как в контрактах такая возможность могла отсутствовать. Компаниям приходилось оплачивать продолжение работ по контракту, даже когда нужда в программе отпадала. Так, например, происходило после распада СССР, когда экономическая ситуация изменилась кардинально и многие системы стали не нужны, но проекты остановить было невозможно.

Уже в 70-е годы прошлого века было понятно, что эта модель очень ограничена. Вся индустрия развивалась так, чтобы можно было в программы вносить правки, чтобы код следовал быстрым изменениям на рынке.

Вы можете представить ситуацию, что заказчик приносит в середине проекта деньги и просит изменить какую-то страницу приложения, а вы отказываетесь, заявляя, что будете делать всё, как раньше договаривались? Вряд ли. Возможно, вы попросите больше денег, чем договаривались изначально, сдвинете сроки, перепишете большой кусок системы, но вы точно не начнёте всё сначала, как этого требует водопадная модель разработки.

Близкая к водопадной модель иногда встречается на совсем небольших проектах и проектах по интеграции существующего, а не разработке нового ПО. Но там просто изменений от заказчика не поступает и решения типовые, поэтому нужда в итерациях отсутствует.

В обычном же проекте в настоящее время классического водопада нет. Да, возможно, не используется Scrum. Да, воз-

можно, вообще никто не может сказать, что именно это за методология, и она нигде не описана. Но это не водопад.

Почему это вообще важно? Потому что методология разработки оказывает очень большое влияние на всё. Если вы считаете, что работаете по одной методологии, а на самом деле работаете по другой, то вы будете принимать неправильные решения.

На практике это выглядит просто и привычно. Вы думаете, что вы применяете водопадную модель. Значит, вы можете легко делать Fixed Price[3] проекты. Нужно только собрать требования и можно сказать, в какой бюджет вы впишетесь.

Fixed price проекты могут быть очень выгодными, поэтому вы начинаете их делать. И оказывается, что вы превышаете бюджет проекта в разы. Сперва вы подозреваете, что вы неправильно оцениваете проекты, и вы начинаете совершенствовать свой процесс оценки (параллельно теряя деньги на проваленных проектах). Потом вы видите, что требования описываются недостаточно детально, и вы начинаете совершенствовать аналитику и мучать заказчика дополнительными раундами уточнения требований (снова теряя деньги на проваленных проектах). Потом вы видите, что эти прекрасно описанные требования всё равно меняются заказчиком на

[3] Fixed Price – модель работы, когда заказчик заранее договаривается о конкретной сумме за проект. Любые превышения бюджета идут за счёт компании-исполнителя. Этот термин используется в противоположность проектам Time&Material, когда заказчик оплачивает время работы разработчика на своём проекте, сколько бы этого времени не понадобилось.

более поздних этапах. Вы начинаете это ему запрещать, а заказчик возмущается, так как его бизнес уже изменился за то время, пока вы писали свою аналитику. В результате вы понимаете, что для применения чистого “водопада”, вам нужно жёстко отфильтровывать заказчиков и их проекты. После разработки и применения этих фильтров вы понимаете, что на рынке нет достаточно заказчиков, которые готовы с вами работать по “водопаду”. Но вы не переживаете, так как ваша компания вряд ли дойдёт до этого этапа. Скорее всего, денежные потери будут неприлично высоки уже на этапе попыток совершенствования оценок. Вы либо прогорите, либо измените свои подходы к проектам.

Поэтому, если вам кажется, что вы работаете по водопадной модели разработки, пожалуйста, удостоверьтесь, что вам это не кажется.

Виды проектного менеджмента

Роль менеджера гораздо слабее определена, чем роль разработчика. В разных компаниях разные менеджеры занимаются очень разными видами деятельности. Но так же, как backend-разработчик отличается от frontend-разработчика, разные руководители проектов отличаются друг от друга по набору знаний и умений.

Отсутствие чёткого разделения разных типов менеджеров очень мешает на практике. Если вы устраиваетесь на вакансию “руководитель проектов”, то вы не можете заранее угадать, что от вас потребуется.

Давайте пройдёмся по разным типам менеджмента. Но имейте в виду, что менеджеры, как и разработчики, бывают “fullstack” и на практике от вас наверняка потребуется знание нескольких перечисленных областей:

1. **“Бумажная работа”**. Самый печальный вид менеджмента, который менеджментом не является, так как не подразумевает принятия важных решений. Я бы не выделял этот пункт, если бы такой менеджмент не был настолько распространён.

Дело в том, что многие IT компании возникли как группа разработчиков, возглавляемых каким-нибудь молодым предпринимателем. Предпринимательский склад ума отличается от менеджерского, и разработческий менталитет то-

же далёк от менеджерского. В результате компании живут фактически без проектного менеджмента до тех пор, пока не вырастут до размера 30-50 человек. На этом этапе уже появляется желание работать над проектами хотя бы среднего, а не мелкого размера. Владелец компании начинает понимать, что он не может выполнять функции менеджера проектов, так как у него не хватает времени. А ключевые разработчики начинают жаловаться, что 99% их времени занимают задачи, не имеющие отношения к разработке: написание писем, проведение совещаний, отчётность и т.д. Тогда решают "завести" менеджеров проектов.

Но раз компания выросла без менеджеров, то и обязанности менеджеров определяются из соображения "то, чем разработчику не хочется заниматься", забывая, что менеджеру для эффективной работы нужно иметь возможность влиять на принятие решений. В результате менеджеру приходится добиваться результата очень косвенными методами: переговорами с владельцем компании, начальниками отделов и лидами, манипуляциями с отчётностью и отточенной риторикой.

Любому менеджеру нужно иметь продвинутые навыки работы с отчётностью и умение вести корреспонденцию. Но если вы видите, что эти навыки являются целью вашей работы, а не средством, то, возможно, чтобы заниматься настоящим менеджментом, работу вам стоит сменить.

2. **Предпродажа/продажа**. Интересный подвид проект-

ного менеджмента, когда бойкий руководитель проекта с хорошо сработавшейся командой создаёт новый бизнес. Он общается с новым для компании заказчиком, а его команда быстро облекает все хотелки заказчика в работающий код. Начало бизнеса выглядит как череда мелких проектов и прототипов, которые быстро вырастают в большие проекты.

Такой менеджер имеет продвинутые навыки оценки проектов и ведения переговоров. А, например, нанимать junior-ов и выращивать из них senior-ов он запросто может не уметь или не любить. В его работе это не нужно. Когда бизнес из стадии зарождения переходит в более стабильную фазу, такому менеджеру становится скучно, а сам он становится неэффективным. Он передаёт подготовленную площадку другому менеджеру, а сам идёт стартовать новый бизнес с новыми заказчиками.

3. **Менеджер процессов**. Умение определять процессы разработки важно для любого руководителя проектов, но есть проекты, где нужны прямо гении такого управления. В больших проектах, где работает сотня и больше людей, и где есть большой заказчик со своими заморочками, процессы могут стать очень сложными. Я не говорю про бюрократию, которая мешает работать. Я говорю про сложные бизнес-требования, когда конкретный функционал должен определяться разными группами людей.

Если написанный большой кусок кода внезапно оказался не соответствующим европейским требованиям GDPR по

защите персональных данных, то это катастрофа. Если требования, которые бизнес создавал полгода, оказалось невозможно реализовать ни в каком виде, то это тоже катастрофа. Чем больше проект, тем выше вероятность таких катастроф, и тем важнее роль процессов.

Но и в небольшом проекте, где работают 5 человек, очень здорово, когда разработчики не мешают тестировать, а заказчик может видеть нужную ему отчётность напрямую, без менеджера.

4. **People management**[4]. В компаниях с проектной организацией менеджеры проектов отвечают за найм, развитие и увольнение сотрудников. Даже, если менеджеры за это не отвечают, то они могут быть ключевыми людьми в деле мотивации и карьерного развития членов своих команд.

Это очень интересная тема, по которой откровенно мало информации. Компании понимают, что важно уметь из разрозненных людей создавать мотивированную команду, которая останется с вами много лет. Но компании абсолютно не знают, как этого добиваться, и даже просто как оценивать успех менеджера в этой области. Поэтому в этой книге я по-

[4] Вот один из примеров, когда явление настолько редкое, что даже термина нормального устоявшегося нет. Часто менеджеры занимаются "делом", а когда нужно мотивировать, выращивать специалистов, исправлять выгорание, разрешать конфликты и осуществлять прочую нетривиальную работу с людьми, зовут HR-специалистов, которые не могут ничем помочь, потому что недостаточно погружены в конкретную проблему. Члены команд опытных People Manager'ов характеризуют их как честных, справедливых, опытных или просто говорят "нормальный руководитель, приятно с ним работать".

свящу довольно много глав этой, любимой мной теме.

5. **Антикризисный менеджмент**. Тоже любимый мной и очень интересный вид менеджмента, о котором существует крайне мало информации. Периодически случается, что крупный проект вдруг начинает распадаться на части и непонятно, отчего это происходит.

В IT у антикризисного управления есть особенности, которые сильно отличают его от антикризисного управления в других областях:

a. Невозможно сменить команду. Традиционный антикризисный подход – это начать работу "с чистого листа". Вся команда меняется на новых людей, все процессы перезапускаются заново. В IT это сделать невозможно, так как люди слишком ценный ресурс и найти 20-30 незанятых человек просто невозможно. К тому же в "кризисных" проектах знания распределены по ключевым людям и без них нельзя продолжить работу. Да и заказчик не может терпеть задержку из-за "перезапуска" проектов.

Конечно, всё равно необходимо делать перестановки в команде, но делать их нужно очень осторожно и точечно. Это гораздо сложнее для руководителя, чем всех выгнать и нанять новых людей.

b. Сложно диагностировать проблемы. "Нормальный" проект тоже имеет кучу сработавших рисков и рисков, которые могут сработать в любой момент. Тяжело отличить проект в кризисе от обычного, поэтому часто проекты заканчи-

ваются провалом, а никто не успевает заметить, что они проваливаются, и ввести антикризисное управление.

Антикризисному менеджеру нужно очень хорошо понимать механизмы функционирования IT проекта, чтобы понять, на чём сконцентрировать изменения, а что можно оставить, как есть.

c. Демотивация команды серьёзно влияет на работу. Когда приходит антикризисный менеджер, люди уже сильно измучены неудачами, нелогичным менеджментом и пониманием, что проект далёк от нормального. А менеджеру требуется, чтобы они сделали трудовой подвиг и вывели проект из пике.

d. Обычным для кризисного управления, но от этого не менее сложным, является то, что менеджер окружён потоком проблем, каждая из которых выглядит приоритетной. Причём попытка решить любую из этих проблем приводит скорее к возникновению новых трудностей, чем к улучшению ситуации. Так работать очень тяжело психологически.

Если брать какого-то усреднённого менеджера, то он сочетает немного от каждого из перечисленных типов. Он ведёт переписку, он организует процесс разработки, он занимается командообразованием и решением конфликтов внутри команды, он немного занимается продажами, расширяя свой проект и предлагая заказчику новые возможности, он отрабатывает кризисные ситуации, не позволяя проекту скатиться в хаос.

История про ответственность

На одном моем новом проекте подошли ко мне лиды двух команд, Вася и Наталья. Судя по их виду, разговор предстоял серьёзный.

– Константин, нам нужно принять серьёзное решение, – сразу перешла к делу Наталья.

– Отлично, я люблю принимать решения. В чём дело?

– Нам нужно переписать наш главный модуль на микросервисную архитектуру. Он монолитный, давно написан, и теперь мы сразу несколько важных задач не можем реализовать без переписывания.

Вася и Наталья выжидающе глядели на меня. Я в ответ тоже глядел выжидающе:

– И какое решение нужно принять-то? – я пока не мог задать более осмысленного вопроса.

– Решение о переписывании модуля на новую архитектуру! – Вася и Наталья были удивительно единодушны.

– Мы можем его не переписывать?

– Нет, не можем. В том и дело. Мы не можем реализовать нужные заказчику задачи без этого.

– Так а какая альтернатива есть?

– Никакой альтернативы! В том и дело! Мы просто вынуждены его переписать!

– Тогда решения никакого принимать не нужно. Вы сами

уже приняли решение переписать модуль и просто хотите, чтобы я донёс это решение до заказчика, так?

– Нет. Ты должен принять решение!

– Так даже альтернативы никакой нет! Значит, решение вы уже приняли. В чём проблема-то?

– Э, не. Кто принимает решение, тот и ответственность несёт. А мы на себя ответственность такую брать не будем! Мы лидами уже давно работаем. С такими решениями всегда проблемы. Потом разборки начинаются, крайних ищут. И мы такими крайними быть не хотим.

Наконец-то я понял, чего от меня хотят.

– Вася, Наташа, тут можете не беспокоиться. Вся ответственность всё равно на мне. Крайнего искать не надо. Давайте лучше поглядим, что в наших планах поменяется, и что мы заказчику сегодня на митинге скажем.

И вот тогда началось обсуждение, где действительно нам надо было принять несколько решений. За которые ответственен опять был я.

Воздействия и результат

Основная трудность руководителей заключается в том, что результаты высокоуровневых менеджерских воздействий часто видны только через продолжительный период времени. Непосредственно наблюдаемый результат часто приводит к ошибочным выводам.

Чтобы пояснить, что я имею в виду, я хотел бы привести пример из замечательной книги Даниэля Канемана "Думай медленно… Решай быстро".

"Обсуждая подготовку летчиков, опытные инструкторы отмечали, что похвала за особо мягкую посадку обычно приводит к тому, что следующая посадка получается хуже, а резкая критика за грубую посадку обычно приводит к улучшению в следующей попытке. Инструкторы пришли к выводу, что словесное поощрение губительно для обучения, а словесное наказание полезно, в отличие от общепринятой психологической доктрины".

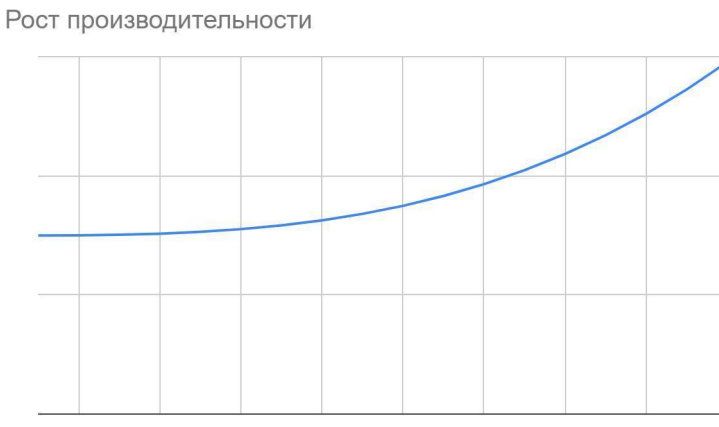
То есть инструкторам говорили, что похвала приносит пользу обучению, но они не верили. Их опыт говорил прямо обратное. Они хвалили курсанта за хорошую посадку, и следующая посадка была хуже. От инструкторов ускользало то, что по теории вероятностей после исключительно хорошей посадки, скорее всего, будет посадка хуже, хоть хвали курсантов, хоть ругай, хоть храни молчание. Они же видели

результат своими глазами!

Можете кидать обычную игральную кость и хвалить её, когда выпадает шестёрка, и ругать, когда выпадает единичка. Вы увидите, что игральная кость от похвалы "расслабляется" и выкидывает результат хуже, а от критики "собирается" и выкидывает что-то получше. Но с игральной костью все механизмы работы просты и понятны, и всегда можно поэкспериментировать. А на практике наблюдаемый результат может сбить с толку.

В менеджменте такое происходит постоянно. В моей практике был очень забавный пример, как одни и те же результаты интерпретировались по-разному. Я тогда работал в очень большой IT-компании, где группе из примерно 30 менеджеров поставили задачу увеличить производительность их команд. Разработчики выполняли некоторые задачи, каждая из которых фиксировалась тикетом в Jira. Количество зафиксированных (созданных и закрытых) тикетов и определяло производительность.

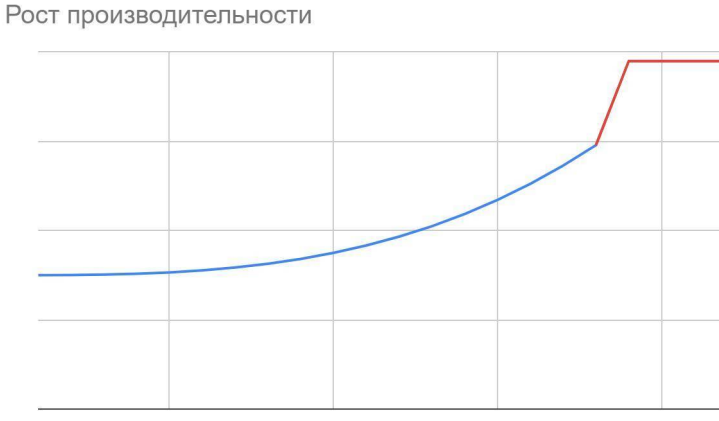
В компании к формальным метрикам производительности относились очень серьёзно. Те, кто не мог добиться улучшения производительности, рисковали потерять работу. Так что и менеджеры, и разработчики серьёзно напряглись. И через неделю суммарный график числа закрытых тикетов стал выглядеть как-то так:



На общем собрании один из менеджеров, очень активный парень по имени Пётр, так прокомментировал наблюдаемое: “Уважаемый менеджмент, поглядите, как здорово мы выполнили ваше задание. Вы попросили увеличить производительность, и мы всего за пару недель производительность фактически удвоили”.

Ему ответил вице-президент компании, Густаво: “Глядя на этот график, становится очевидно, что раньше разработчики просто не заводили тикеты на многие свои задачи. Теперь они серьёзно относятся к отчётности и в Jira зафиксировано всё, что они делают. Ваша задача как менеджеров теперь проанализировать происходящее и увеличить производительность, а не просто улучшить числа в отчёте”.

Через неделю график производительность обзавёлся очередным скачком:



Менеджер Пётр снова прокомментировал происходящее: “Вы требовали от нас увеличения производительности – мы оптимизировали процессы. На графике вы с очевидностью можете видеть результаты наших усилий. Производительность выросла не так сильно, как в прошлый раз, но всё равно значительно”.

Вице-президент, глядя на тот же график, сказал: “На этом графике вы можете наблюдать нередкое явление. Люди, не видя очевидных решений проблемы и не имея достаточно желания прикладывать усилия, решили обмануть систему.

Очевидно, что просто в Jira было добавлено много тикетов, которые не имеют отношения к делу. Пожалуйста, проверьте отчётность своих команд. На следующей неделе я сам проконтролирую, что в Jira нет мусорных записей, искажающих статистику. Тем временем я по-прежнему жду от вас плана действий по повышению производительности".

Здесь интересны не попытки избежать неожиданно свалившихся проблем. Здесь интересно, что два человека, смотря на одни и те же графики, делают абсолютно разные выводы. Причём, выводы Петра кажутся более логичными, но на практике истина была ближе к тому, что говорил Густаво.

Со мной периодически кто-нибудь спорит:

– Константин, твои методы слишком сложные. Любую проблемную ситуацию в команде можно решить, просто пригрозив увольнением. Вот жалуется человек на что-то. А я ему говорю: "Увольняйся!" – и сразу все претензии пропадают.

– Но ведь так никакой команды не построить! – в шоке возмущаюсь я.

– Зато люди делают то, что им сказано. А больше ничего не надо. Это эффективно.

Неопытный менеджер просто не знает, как работать с людьми правильно. Возникают ситуации, которые он не знает, как решить. И вдруг оказывается, что если на людей наорать, пригрозить увольнением и объявить выговор, то проблемы как бы уходят. Это кажется простым решением, а

негативные последствия носят очень сложный и отложенный характер. Да, почему-то люди увольняются, но не сразу ведь. Да, новых людей становится искать всё сложней, а некоторые резко отказываются рассматривать любые вакансии этой компании, но это же рынок труда скудный. Да, разработчики ведут себя пассивно, но для этого и нужен менеджер, чтобы их "активизировать". Так и продолжают существовать неэффективные, вредные и просто опасные приёмы менеджмента.

Отложены не только негативные, но и позитивные эффекты. Когда я работаю с сильно демотивированной командой, то явный прогресс становится виден через полгода-год. Часто я уже перехожу на другой проект и не вижу плодов своих трудов. Даже высокий менеджмент с трудом может отследить конкретный прогресс, так как уже забывает, что происходило так давно назад.

Как же работать в такой ситуации? Нужно опираться на принципы, а не на сиюминутные ощущения. Сейчас нельзя применять телесные наказания к работникам, а раньше это было распространённой практикой. Поэтому менеджеры не пытаются ударить разработчика, даже если им это кажется хорошей идеей. Аналогично и с более тонкими принципами. Когда менеджер верит, что нужно доверять команде, быть с ней открытым и уважать мнение других, то он ищет другие способы решать проблемы, нежели брызгать слюной, кричать на подчинённых и грозить увольнениями.

И очень вдумчиво нужно относиться к различным статистическим и прочим “объективным” данным. Их объективность часто полностью нивелируется неверной субъективной моделью, которую использует менеджер при интерпретации этих данных.

Если планомерно, не сдаваясь, применять свои принципы, то через некоторое время вдруг окажется, что вы управляете хорошо слаженной командой высококлассных специалистов, с которыми вы не воюете, а сотрудничаете и достигаете удивительных результатов.

История про неожиданные выводы

Однажды я работал в одной международной компании, где разработчикам (и любым другим сотрудникам) был доступен очень прозрачный процесс движения по карьерной лестнице. На сайте компании был список всех открытых вакансий с указанием зарплат. Если разработчик видел, что есть подходящая ему вакансия с более высокой зарплатой, то он буквально в один клик мог выразить желание себя попробовать, проходил цепочку тестов и переходил на более интересную для себя позицию. Разработчикам такая возможность очень нравилась, и они ей пользовались.

Но такое положение дел очень не нравилось менеджерам, так как разработчик после повышения всегда менял проект и уходил куда-то на другую должность. А менеджеру приходилось срочно затыкать образовавшуюся дыру в команде, находить другого разработчика, обучать его и т.д. Так как разработчиков на рынке не хватает, то такой уход разработчика часто создавал риски для проекта. Менеджеры роптали, и в результате компания установила порядок, при котором разработчику, чтобы начать тестирование на другую должность, надо было заручиться согласием менеджера.

Менеджеры обрадовались, но тут уже загрустили разработчики. Потому что менеджеры дружно запрещали разработчикам тестирование на другие должности. Просто и кате-

горически. Конечно, разработчик мог просто уволиться, так как ему фактически запрещали повышение. Но менеджерам в той компании не было никаких минусов от увольнения разработчика. Компания была сторонницей жёсткого менеджмента, и увольнение сотрудника в минус менеджеру не шло. Конечно, проект терял разработчика, но он и так бы потерял его, если разработчик ушёл бы на повышение. А так разработчик оставался в проекте, так как увольняться было опасно. Можно было не пройти тестирование и остаться совсем без работы. Так что негласно такой запрет повышений компанией одобрялся.

Мне подобная политика не нравилась. Я считал её губительной уже в средней перспективе и на своих проектах разработчикам разрешал переходы. У меня даже получалось договариваться со всеми заинтересованными сторонами и оставлять разработчиков после повышения на моих проектах. Повышение зарплаты составляло обычно 60%-100%, так что разработчики были заинтересованы договариваться со мной о завершении текущих дел перед переходом на новую должность. Так что я даже извлекал пользу от этого своего отхода от принятой практики.

Но однажды мои принципы подверглись испытанию. На одном из моих проектов я не мог найти архитектора в течение полугода. Я проявлял чудеса изворотливости, но проблемы копились. И я был несказанно рад, когда, наконец, нанял архитектора, Игоря. Игорь бодро приступил к своим обязан-

ностям, но проект был очень сложным, и входить в него нужно было около полугода, чтобы начать работать в полную силу. Но с архитектором уже было повеселее, проблемы не казались такими уж сложными, а многое Игорь потихоньку начал разгребать, так что я с оптимизмом смотрел в будущее.

Но, проработав всего 3 месяца, не войдя даже до конца в проект, Игорь сказал, что появилась интересная вакансия, и он хочет на неё себя попробовать. Вакансия подходила Игорю, но совершенно не подходила текущему проекту. То есть, если бы Игорь на неё прошёл, то я бы потерял архитектора, и был бы вынужден снова искать полгода кого-то другого.

Причём, я мог бы просто поступить, как остальные менеджеры, запретить переход и работать дальше без проблем. Но это противоречило бы моим принципам, и я решил разрешить Игорю это тестирование. Дело было не таким простым, так как если бы я так просто отпустил бы недавно нанятого ключевого специалиста, то меня самого могли бы обвинить в непрофессионализме и уволить. Так что я обратился к своему начальнику, вице-президенту компании, объяснил свою позицию и заручился его согласием делать, как мне кажется лучше. Я дал своё одобрение Игорю, он пошёл тестироваться, а я пошёл думать, что я буду делать, если Игорь уйдёт с моего проекта.

Игорь не прошёл один из этапов тестирования, сообщил это мне и потом у нас состоялся интересный диалог. Игорь сказал:

– Константин, а ты ведь не верил, что я пройду тест!

– Почему? У тебя же есть опыт. Ты вполне мог пройти, – удивился я.

– А, понятно. Значит, ты меня просто не ценишь и тебе пофигу, работаю я у тебя и уволюсь!

Тут у меня слов не нашлось, и я просто завис, а Игорь продолжил:

– Ну, вот другие менеджеры ценят своих сотрудников, поэтому никуда их не отпускают. Все менеджеры так делают. А раз ты меня отпустил, то не ценишь.

Я, конечно, попытался объяснить свою позицию, но до сих пор сомневаюсь, что Игорь понял меня правильно. Уж очень у нас разный подход к делу оказался. А я тогда долго удивлялся, что мне в реальном проекте пришлось работать с традиционным домостроевским подходом “Бьёт – значит любит”.

Раздел 2. Мотивация и командообразование

Область разработки программного обеспечения очень близка к науке или искусству, так как на любом проекте постоянно возникают творческие задачи, которые требуют от разработчика нестандартного подхода к решению. Роль отдельной личности очень высока, а роль команды и командной работы трудно переоценить. Именно поэтому умение менеджера находить общий язык с людьми, делать из отдельных индивидуумов команду и решать конфликты является ключевым для успеха любого проекта.

Руководитель, который не ладит с людьми, спотыкается на каждом шагу. У него возникают проблемы там, где их никто и не ждал, его указания игнорируются или понимаются неправильно, ему выдают неверную оценку, а задачи выполняются мучительно долго. И наоборот, если менеджер умеет работать с людьми, то он со стороны выглядит волшебником. Самые сложные задачи делаются как бы сами по себе, а он концентрируется только на критичных для проекта точках.

Каждому менеджеру нужно работать над своим эмоциональным интеллектом и умением работать с разными людьми. Это единственный путь, чтобы делать действительно сложные и интересные проекты.

Главное правило мотивации

В IT тема мотивации чрезвычайно популярна. Менеджерам рассказывают о важности мотивации разработчиков, и во многих компаниях много усилий вкладывается в мотивацию[5]. Каждый год строятся планы развития, где выясняются желания человека. Потом к этим желания подгоняются планы, соответствующие целям компании. Человек видит, как компания работает над исполнением его желаний.

Кроме такой стратегической мотивации, распространена мотивация локальная. Менеджеры и HRы интересуются настроением каждого, поздравляют с достижениями, уделяют внимание и т.д.

К сожалению, во многих компаниях команды демотивированы и это заметно, как снаружи компании, так и внутри коллектива. Со стороны получается очень интересно: разные компании делают вроде бы одно и то же, но у каких-то компаний получается мотивировать, а у каких-то нет. Причём тех компаний, которые реально могут добиваться высокой мотивации, крайне мало. Как так?

Секрет очень прост: начать работу с мотивацией надо с того, чтобы исключить демотивацию. Дело в том, что мотивация – это очень хрупкий объект. Чтобы человек думал о

[5] Обычно под этим подразумеваются корпоративы, бесплатные тренинги и бонусы вроде медстраховки.

всеобщем благе, стремился к самосовершенствованию и делал больше, чем необходимо, надо, чтобы у него было всё хорошо.

Демотивация же, обычно, наоборот штука грубая, больно "бьющая" по человеку и оставляющая след надолго. Если менеджер один раз вспылит, обзовёт разработчика бесполезным и пригрозит увольнением, то потом может расслабиться и больше не тратить время на мотивацию этого разработчика. Такой менеджер уже всё сломал и чтобы вывести мотивацию хотя бы в ноль, надо обладать специальными навыками и продуманно приложить значительные усилия.

Очень большая ошибка считать мотивацию и демотивацию некими величинами, которые просто складываются. Такая арифметика не работает в реальной жизни.

Например, знаю одного менеджера (очень опытного, кстати), который так говорит про свою работу: "Это я делаю все свои проекты, а не команда. Да, я использую разработчиков и тестировщиков в своей работе, но это просто инструменты для реализации проекта". Он может это мнение своей команде никогда и не озвучивает, но оно прекрасно считывается. Этот менеджер говорит, что его проекты скучные и с них народ очень часто увольняется. Но сам он остаётся работать, поэтому проекты успешно функционируют.

Нельзя с такой позицией пытаться поднять людей на подвиги. Они не будут доверять менеджеру и компании. Просто потому, что менеджер не вкладывается в отношения с ко-

мандой, не ценит свою команду. Само отношение менеджера ставит крест на любых попытках вырастить действительно высокую мотивацию. Можно делать рутину, так как даже не мотивированные разработчики будут делать свою работу, но для сложных проектов этого недостаточно.

Что же делать, если вы (или кто-то другой) всё-таки обидел человека? В первую очередь, нужно извиниться. Причём не просто формально, а так, чтобы человек поверил, что вы свою ошибку осознали, и приложите усилия к тому, чтобы её не повторить. Тут нужно подключить тот самый эмоциональный интеллект, чтобы не сделать ситуацию хуже.

Например, если вы назвали некоего Васю бракоделом, а он смертельно обиделся, то не надо говорить: "Вася, ну прости. Чего ты дуешься? Это ж я пошутил! Ты иногда и не брак делаешь! Ха-ха-ха!" Лучше подойти к делу вдумчиво, с опорой на ваши знания о человеке: "Вася, прости меня, пожалуйста. Это была дурацкая шутка. Меня самого так мой менеджер часто называет, мне слово кажется забавным. Я не думал, что ты серьёзно к моим дурачествам отнесёшься. Все же знают, что твою работу можно не перепроверять. Ты перфекционист. Прости, был не прав". Причём, кому-то эти извинения надо принести при всей команде, а кому-то только лично.

Сразу надо признать, что вопросы мотивации очень сложны, как и любые другие, связанные с эмоциональным интеллектом. В IT работают умные, тонкие люди, которые часто

имеют свои принципы и правила жизни. Их не удастся провести показной доброжелательностью и притворным вниманием. Нужно вести себя открыто с командой и выстраивать доверительные отношения. Только так можно бороться с демотивацией и получить реально мотивированную команду.

История про мотивирующие речи

В одной из компаний, где я работал, я слышал запоминающееся выступление генерального директора. Оно было недолгим и основную часть я попытаюсь здесь пересказать близко к оригиналу:

“В нашей компании идут перемены, и многим кажется, что компания развивается в неправильном направлении. Я вижу и слышу от других, что сотрудники нашей компании демотивированы. А демотивация создаёт проблемы. Вы работаете медленней, приносите меньше пользы проектам и распространяете свою демотивацию на других. Поэтому, если вы чувствуете, что вы демотивированы, мотивируйте себя сами! Будьте мотивированы”.

Этот подход, несмотря на свою полную беспомощность, является очень распространённым. Компании прекрасно видят, что люди демотивированы, понимают проблемы, которые это создаёт, но не представляют, как исправить положение дел.

Доверять команде

Тема доверия является ключевой в работе с людьми. И если посмотреть “в среднем” (а средняя картина всегда довольно печальная), то менеджеры не доверяют своим командам, а команды не доверяют менеджерам. Это абсолютно безумная ситуация, так как менеджер работает с людьми, они его инструмент. Не доверять своей команде – это то же самое, как если бы программист не доверял своему компилятору. Дело бесполезное и опасное.

Давайте сперва определим термин “доверие”, так как здесь есть разночтения. Доверие – это вера в то, человек подойдёт к своим обязанностям честно, постарается выполнить свою часть работы, вместе с вами будет идти к результату, будет вести себя порядочно. Это вера в человека, ограниченная рабочими рамками.

Как доверие менеджера выглядит на практике? Давайте рассмотрим пример. Вот есть у вас разработчик Алексей. И он частенько опаздывает даже на ключевые встречи. При этом сильно вас подставляет, так как вы один на один с заказчиком и его техническими специалистами, без своего специалиста. И вот он снова опаздывает и оправдывается: “Я честно-честно заранее вышел, но у машины колесо спустило. Я её даже на дороге бросил и на такси сразу пересел. Но всё равно немного опоздал, так как такси ждал.

Многие менеджеры принимают подобные отговорки один раз, другой, максимум третий, а потом начинают подозревать такого Алексея в обмане. Недоверчиво косятся на него, задают каверзные вопросы (Какое именно колесо спустило?), пытаются найти свидетелей прокола колеса, просят показать историю звонков, чтобы увидеть вызов такси. Им кажется важным, обманывает их Алексей или нет. Всё это ведёт к напряжённым отношениям, а проблема не решается.

Гораздо проще сразу поверить Алексею. Сомневаться в его словах во-первых, неприлично, во-вторых, контрпродуктивно. Считайте, что действительно у него случилась какая-то критическая ситуация. Лучше даже не выслушивать оправдания, так как у человека могут быть очень личные или даже не совсем приличные проблемы, о которых он вам сообщать и не должен. Не спрашивайте об оправданиях. Зачем вам они? Считайте, что человек старался по максимуму, но у него не получилось.

Удивительно, но такая позиция абсолютного доверия многим кажется “слишком мягкой”. Однако же она гораздо жёстче позиции тех менеджеров, которые не доверяют людям и ищут подтверждений их слов. Как так получается? Очень просто. Менеджеры, которых интересуют отговорки, будут терпеть Алексея вечно. У него будут оправдания, менеджер проверит эти оправдания и успокоится.

Но вас не интересуют отговорки, вам Алексей нужен вовремя на звонках. Если его там нет пару раз, вы ему просто

говорите: “Алексей, я понимаю, что у каждого случаются накладки. Но ты мне нужен на ключевых конференциях с заказчиком. Твоя роль в проекте подразумевает, что ты там будешь. Если ты считаешь по-другому, то скажи об этом сейчас. Я могу с тобой поделиться своим опытом борьбы с обстоятельствами. Опоздания – это не та беда, с которой невозможно справиться. Сейчас уже заказчик шутит, когда тебя нет на митинге. Скоро он будет не шутить, а смеяться надо мной. Я сильно не хочу до этого доводить. Ты можешь сам справиться с этой проблемой или тебе нужна какая-то помощь?”

Обратите внимание, что недоверие к Алексею вы не выражаете. Вы просто говорите, что он не соответствует занимаемой должности. Жёстко, не правда ли? И прикрыться оправданиями Алексей не может. Потому что в таком разрезе понятно, что вам нужна работа, а не отговорки. Но вы не высказываете и тени сомнения в том, что Алексей правда пытается прийти вовремя. Конечно, вы в это верите, просто вам нужен результат.

Те менеджеры, которые довольствуются отговорками, обычно, просто требуют чего-то, что на самом деле, не обязательно. Например, они требуют присутствия разработчика тогда, когда он не особо и нужен. Разработчик это чувствует и “отлынивает”, как может. А менеджер чувствует себя уязвлённым и хочет оправданий. Причём настоящих проблем у разработчика не бывает, так как серьёзных он проблем не

создаёт. Менеджер должен ориентироваться на реальные задачи и требовать то, что нужно для дела. Тогда будет очевидно, что оправдания не нужны хоть правдивые, хоть нет.

Мне на такое возражают, что иногда разработчики явно говорят неправду. Например: “Мой разработчик вот говорит, что задача невыполнима, а её сделать можно. Явно врёт, зараза! Начинаешь его пытать и действительно оказывается, что решение есть. Значит, врал!”

В подобных ситуациях я начинаю с предположения, что разработчик честно попытался найти решение задачи, но не нашёл. И переформулирую её: “Слушай, если ты говоришь, что технически разумного решения задачи нет, то так и есть. Ты спец. Но если мы заказчику не предоставим хоть какой-нибудь вариант, то он придумает свой и нам придётся с ним спорить или, ещё хуже, его реализовывать. Давай лучше придумаем какой-нибудь свой, хоть и не очень разумный вариант. Пусть заказчик над ним подумает. Может, можно от каких-то требований отказаться? Или половину системы переписать? Или удесятерить количество серверов? Или купить готовое решение?”

Опять же, обратите внимание, в такой постановке вопроса нет никакой “мягкости”. Я тут прямо угрожаю разработчику: “Заказчик придумает своё решение и нам придётся его реализовывать”. Это страшная угроза и она реальна. Но говорить разработчику, что я ему не верю, очень неправильно. Я ему верю, решения задачи нет. Просто нам теперь нужно

решить другую задачу: предложить что-то другое заказчику.

На самом деле, нет ни одного вменяемого примера, когда имеет смысл сказать человеку, что вы ему не верите. Человек врёт вам, вы ему об этом скажите, и он резко перестанет так делать? Это сумасшедшее предположение, так общение с людьми не работает. А если вы не можете сказать никому, что вы ему не доверяете, то зачем не доверять?

Подумайте над последним вопросом. Он таит за собой больше возможностей, чем кажется. Мне некоторые менеджеры говорили: "Я никак не могу поверить некоторым своим людям. Они явно работают недобросовестно".

Во-первых, я не знаю, зачем работать с людьми, которые работают недобросовестно. Это очень странное оправдание со стороны менеджера. Во-вторых, менеджер, несомненно, может доверять своей команде. Не только может, но и должен. Люди веками верят, что свист как-то связан с количеством доступных денег. Или что перебежавший дорогу чёрный кот может влиять на вероятностные события. Способность людей верить просто колоссальна. Нужно работать над собой до тех пор, пока не выработается способность доверять своей команде. Иначе работать с людьми нельзя.

И ещё иногда менеджеры смешивают доверие и технический уровень: "Задача сложная, Вася никогда не работал над такими, он может не справиться". Это не имеет никакого отношения к доверию. Доверие – это про то, что Вася будет стараться изо всех сил (может и невеликих), а не про гарантию,

что Вася справится с задачей. Работа с рисками, по-прежнему, на менеджере. Даже, если вы доверяете Васе, вы можете и должны помогать ему, контролировать его, организовывать ревью его работы более опытными разработчиками и т.д. Потому что никто (и вы в том числе) не застрахован от ошибок. И наличие ошибок не должно влиять на доверие.

Конец ознакомительного фрагмента.