КЕЙС

Как мы за три месяца выстроили прозрачную систему оценки эффективности разработчиков после 10 лет работы без процессов

Формат	Кейс
Площадка	vc.ru, раздел «Разработка», блог Михаила Кечинова
Дата	18.07.2024
Подпись	Михаил Кечинов, CEO REES46
Просмотры	3,9 тыс. просмотров (по данным vc.ru на 20.10.2026)
Примечание	Гострайтинг — текст написан целиком, от лица CEO REES46.
Оригинал	https://vc.ru/dev/1318429-kak-my-za-tri-mesyaca-vystroili-prozrachnuyu-sistemu-ocenki-effektivnosti-razrabotchikov-posle-10-let-raboty-bez-processov

Автор текста: Руфина Семыкина · Оригинал на русском языке

Привет! Меня зовут Михаил Кечинов, я CEO IT-компании REES46, которая занимается разработкой решений для автоматизированного маркетинга. Мы работаем с любым бизнесом, который есть в интернете и у которого больше ста клиентов.

Недавно на VC наткнулся на интересное обсуждение — нормально ли если сотрудник работает на двух и более работах. Рассказываю, почему ненормально и как мы сами в компании теперь оцениваем качество работы наших разрабов — без всяких ОКРов, ПИПов и даже без эйчара, а просто выстроив наконец процессы.

Почему «работать не 8 часов, а головой» нам не подходит

Несколько месяцев назад на VC был интересный тред. В статье обсуждали, что, например, аккаунт-менеджер из США работает на двух работах, хотя занят всего 40 часов в неделю. То есть работает на каждой работе по полдня, а зарабатывает $200 тысяч в год. Это, конечно, не вопрос договоренности с работодателем, а тайм-менеджмент, при котором никто не подозревает, что работы на самом деле две. Нормально ли это?

Ну, это нормально только если и сотрудник, и работодатель просто тянут лямку по 8 часов в день. Но в реальности таких работ уже почти не осталось — особенно в IT. Сотрудники могут совмещать работу с другими проектами, но в нерабочее время. Ночью или после работы — пожалуйста. Называйте меня старовером, но днем — наше время, мы платим зарплату за эти восемь часов. Человек обещает отдать результаты своей работы за восемь часов в обмен на деньги. И мы рассчитываем, что все эти восемь часов наши. Закончил задачу раньше? Прекрасно, возьми другую — недостатка задач у нас нет.

Я уверен, что большинство моих сотрудников где-то еще подрабатывают. Пусть занимаются — разнообразие же. Главное, чтобы не в ущерб нашим проектам. А так, чтобы фуллтайм больше, чем на одной работе... Это значит, что работодатели либо совсем дураки и не понимают, что ты работаешь по три часа вместо восьми, либо им просто пофиг. В

общем, карьеру с таким руководством все равно не построишь. И нам, работодателям, неприятно.

Нормально — это сколько?

Если никакого KPI нет и никто не контролирует что ты делаешь, есть соблазн не делать ничего... для этого работодателя. А делать параллельно свои пет-проджекты. Или что угодно. Но тут возникает вопрос. Как оценить нормальную скорость работы?

В крупных компаниях обычно выстроены целые системы оценки эффективности сотрудников. У Google есть OKR (Objectives and Key Results) — вдохновляющие цели (Objectives) + измеримые результаты (Key Results), которые помогают оценить, как эти цели достигаются. У Amazon — PIP (Performance Improvement Plan) для «отстающих», то есть для тех, чья производительность вызывает вопросы. Здесь и измеримые цели, и дополнительные ресурсы (например, обучение или наставничество), и регулярные проверки.

Но внедрить это все в маленькой IT-компании на 40 человек непросто. Во-первых, это требует значительных ресурсов и времени. Это целая отдельная система, в которой задействовано много людей, чья профессия — мотивировать и поддерживать здоровую атмосферу в коллективе. Это еще, как минимум 10 человек (25% нашего штата) надо нанять. Мы пока не доросли — да у нас даже HR’а в штате нет! Во-вторых, мелкие компании часто нуждаются в более гибких и адаптивных подходах к оценке эффективности сотрудников.

Пока людей мало, в общем, видишь, кто что делает и сколько времени это может занять. Зато, когда вас уже скорее 50, чем 10 надо какие-то решения для оценки «производительности труда» все-таки находить. А еще — надо учитывать два момента:

- Чем больше команда, тем дольше срок выполнения одной конкретной задачи. Задачу, которую один программист сделает за 2 часа, трое — CTO, PM и программист — сделают за два дня. Так я собрал сайт очередного Hackday (я провожу иногда хакатоны в Питере) за 6 часов, а у команды из трех программистов, дизайнера, копирайтера, тестировщика и проджект-менеджера, ровно эта же работа заняла бы недели две. Но зато, когда речь идет не об одной задаче, а о ста задачах, процессы и делегирование, наоборот, очень ускоряют работу.
- У двух разработчиков может быть одинаково большой опыт в решении очень разных задач. То, что для одного задача на пять минут, для другого — незнакомая область, в которой надо разобраться. Другое дело, что если на эту пятиминутную задачу потрачено две недели времени, и сотрудник ни к кому за это время не обратился с вопросом, то дело не в задаче.

Что сработало для нас?

Мы прошли все этапы принятия того простого факта, что KPI все-таки нужны. Мы верили, ждали, погружали в продукт, проводили много встреч, где рассказывали, как работает система, перекладывали ответственность и получали в итоге хороший результат, но только очень медленно.

[Иллюстрация] Оказавшись здесь, мы решили попробовать выстроить какую-то систему

В этом году мы решили все поменять и значительно так ускориться. И вот какие способы повысить скорость работы и оценить результаты мы нашли.

Agile как он есть

Задачи теперь формируются по контексту и объединяются в майлстоуны, после чего под майлстоун ставится ответственный инженер.

Для работы с задачами мы выбрали стратегию Kanban, так как задач всегда на порядок больше чем разработчиков.

В конце каждой недели мы формируем спринт под каждого инженера с учетом обратной связи от продуктовой команды. То есть мы всегда расставляем приоритеты — писать фичи или фиксить баги — прежде чем пойти что-то кодить.

Один из методов Agile, о котором почему-то мало кто знает/помнит — Extreme Programming. Мы его внедрили и довольно активно используем. То есть вместо нового пачборда делаем рефакторинг конкретного скоупа, это по итогу выходит дешевле, пускай и не всегда быстрее пачки новых заплаток. И в перспективе всегда выигрышней.*

> * Этот кусок наши маркетологи попросили перевести на русский — просто на всякий случай. Получилось вот что: «Вместо внесения новых исправлений мы проводим рефакторинг (улучшение внутренней структуры кода) конкретного направления. В итоге это выходит дешевле, хотя не всегда быстрее, чем точечное исправление багов. И в перспективе такой подход всегда оказывается более выгодным».

Регулярная обратная связь

Для сбора обратной связи мы используем автоматизированные DSM (без созвонов) чтобы разработчики могли сообщать о задачах, которые они решали вчера и которые будут решать сегодня.

Также, по окончании рабочего дня ребята публикуют свои ветки, открывают пул-реквесты в режиме драфта и до завершения задачи коммитятся в конце каждого рабочего дня.

А еще мы учим команду самостоятельности и рассказываем как (и в каком формате) лучше всего донести текущий статус задачи. Часто задачи бывают «особенно горящими» и чтобы не нервировать другие подразделения (от которых клиенты ждут фидбека по ним), мы придумали репортить в задачу в конце каждого дня текущий статус и что еще осталось сделать, получается удобно и наглядно.

Каждый из инженеров, продактов, аккаунтов да и вообще любых сотрудников может спокойно сходить на One-To-One с любым из топов, будь то владелец компании или же технический директор, таким образом мы обеспечиваем максимальную доступность информации внутри коллектива и предотвращаем возможные коллапсы.

Прозрачные процессы

Так как разработка находится в Github, а бизнес живет в JetBrains Space, нам нужно было придумать удобный и, главное, прозрачный механизм для взаимодействия между

подразделениями для этого мы разработали несколько микросервисов и парочку ботов, которые позволяли синхронизировать данные из Space в Github и обратно.

У всех задач есть статусы: ToDo, In Progress, Reviews, Docs. После прохождения ревью и тестирования, таска переводится в статус Docs — это значит, что ее подхватывает технический писатель и обновляет нашу документацию, чтобы она всегда оставалась актуальной.

А вот как проходит сдача-приемка задач:

1. Прежде чем отправить задачу на ревью разработчик должен опубликовать в своем пул-реквесте пруфы того, что его задача выполнена так, как она ставилась бизнесом, это может быть скриншот или скринкаст.
2. Ревью проходит по принципу Round Robin, то есть после того, как ревью проводит старший из команды оно переходит к лиду, потом уже к СТО — таким образом мы максимально сокращаем возможное количество инцидентов при релизе.
3. После того как ревью пройдено, задача отправляется в статус Testing, то есть отправляется Stage контур, который изолирован от продакшена.

Нередко бывает так, что задачи зависают в статусе Review или возвращаются после тестирования в статус Returned. Такие задачи в приоритете, потому что их затягивание неизбежно приводит к потере контекста да и просто элементарному рассинхрону между бизнесом и разработкой, когда уже никто не помнит что там было и как.

Уже сейчас понятно, что эти простые правила сильно продвинули нас вперед по скорости. Раньше на онбординг уходили месяцы, а некоторые задачи выполнялись (не выполнялись) неделями. Теперь новички берут в работу задачи с первого же дня — и сразу приносят пользу компании. И, да, действительно работают по 8 часов — прямо как в трудовом договоре написано.