12+

Александр Рут

Код реальности

О чём молчат успешные разработчики

Александр Рут

Код реальности. О чём молчат успешные разработчики

https://litres.ru/74143504
ISBN 9785006982963

Аннотация

Эта книга — не история успеха. Здесь нет героя, который в двадцать три года написал стартап и продал его за миллион.

«Код реальности» — честный разговор о том, каково это на самом деле: уйти с найма, начать брать заказы, выстраивать жизнь так, чтобы работа не съедала её целиком.

Для тех, кто думает о фрилансе, уже на фрилансе или просто хочет понять — как оно бывает у тех, кто не пишет об этом в сторис.

Содержание

Код реальности
О чём молчат успешные разработчики

Александр Рут

ISBN 978-5-0069-8296-3
Создано в интеллектуальной издательской системе Ridero

ПРЕДИСЛОВИЕ

Зачем ещё одна книга про программирование?

Наверное, потому что мне надоело читать правильные книги.

Те, где всё получается. Где герой в двадцать три года пишет стартап в гараже, через год продаёт его за миллион, а на последней странице улыбается с обложки Forbes. Красиво. Вдохновляюще. И абсолютно не похоже на то, как оно бывает на самом деле.

Я хотел написать другую книгу. Про то, как задеплоил обновление в пятницу вечером — и сломал прод. Про клиента, который три недели говорил «всё устраивает», а потом в последний день попросил переделать всё с нуля. Про письмо, которое я отправил с предложением помочь — и не получил в ответ ни слова. Про страх, усталость, выгорание — и про то, почему несмотря на всё это, я продолжаю.

Эта книга — не учебник. Не пошаговая инструкция «как стать успешным разработчиком за сто дней». Это просто мои истории. Живые, местами неловкие, иногда смешные — и, надеюсь, честные.

Читай её, если чувствуешь, что застрял. Если думаешь, что у всех получается, а у тебя почему-то нет. Если сомневаешься — правильно ли идёшь. Спойлер: скорее всего, правильно. Просто никто не рассказывает вслух, насколько у

всех бывает сложно.

ЧАСТЬ I
Как меня сюда занесло. История о том, как техподдержка стала точкой невозврата

ГЛАВА 1

ЗДРАВСТВУЙТЕ, ВАШ КОМПЬЮТЕР ЗАВИС

Или: как мальчик из села с самодельной сигнализацией оказался в московской техподдержке — и почему это было только начало.

Я вырос там, где интернет был роскошью, а сотовый телефон — событием.

Маленькое село. Деревянные заборы, огороды, соседи, которые знают друг про друга всё. Технологическая новость номер один — кто-то из жителей купил новый телевизор. Это обсуждали неделю.

Но я почему-то смотрел в другую сторону. Где-то там, за горизонтом, существовал другой мир — с роботами, ком-

пьютерами и вещами, которых ещё не изобрели. Я был уверен в этом с детства, хотя объяснить, откуда эта уверенность, — не смог бы.

Когда в школе проводили олимпиады и конкурсы, я придумал систему безопасности. Не нарисовал — именно придумал: как она работает, из чего собирается, что делает. Датчик на дверь, провод, кнопочный телефон. Если кто-то открывал дверь — на запрограммированный номер уходил один звонок. Один гудок. Значит, кто-то вошёл.

Надо понимать контекст. На всё наше село тогда приходилось один-два сотовых телефона. Не на семью — на всё село. Сотовая связь только начинала появляться, это было почти фантастикой, недостижимой роскошью. Но именно поэтому идея казалась мне такой интересной: взять то, что есть, соединить необычным способом — и получить что-то новое.

Денег на реализацию не было. Проект так и остался на бумаге. На конкурсе я не выиграл — судьи не могут оценить то, чего не существует в металле. Но в призёры попал. И главное — ощущение осталось. То самое, которое трудно описать словами: я могу придумывать вещи, которых ещё нет.

Как говорил Стив Джобс: «Творчество — это просто умение соединять точки». В тот момент я ещё не знал этой цитаты. Но уже делал именно это.

Это ощущение я пронесу через годы корпоративных офисов, сотни звонков в техподдержку и чужих зависших принтеров. Оно никуда не денется. Просто ляжет на дно — и бу-

дет тихо ждать своего часа.

МОСКВА. БЕЗ ОПЫТА. БЕЗ СВЯЗЕЙ.

Университет я окончил с красным дипломом.

Звучит гордо. На практике — это красивая корочка, которая открывает примерно столько же дверей, сколько обычная. То есть почти никаких, если за ней нет опыта работы.

В маленьком городке расти было некуда. Потолок был виден сразу — и он был низким. Поэтому я собрал вещи, позвал друга, и мы поехали в Москву. С амбициями, красными дипломами и абсолютным непониманием того, как устроена настоящая жизнь за пределами учебных аудиторий.

Первый урок Москва преподала быстро.

Я ходил по собеседованиям. Рассылал резюме. Объяснял, что учился хорошо, схватываю быстро, мотивирован. В ответ слышал одно и то же: «нам нужен человек с опытом». Но откуда взять опыт, если никто не берёт без опыта? Этот замкнутый круг — одна из самых жестоких ловушек для любого начинающего специалиста. Ты стоишь перед закрытой дверью и держишь в руках ключ, который к ней не подходит.

Особенно раздражало молчание. Никто не объяснял причин отказа. «Мы вам перезвоним» — и тишина. Компании больших и малых размеров хотели людей, которые знают теорию от корки до корки, хотя большую часть этой теории на практике никто никогда не применяет. Странный мир.

Несколько месяцев я пробовал разное. Стоял за прилавком с электроникой — не моё, совсем. Откликался на всё

подряд — молчание. И вот наконец — небольшая компания, техподдержка компьютеров и принтеров. Не мечта. Но старт. И уже это было хорошо.

ГОД СРЕДИ ПРИНТЕРОВ

Первый год в техподдержке был честным в самом простом смысле слова.

Ездил по офисам, настраивал компьютеры, чинил принтеры, терпеливо объяснял людям, почему «оно само не работает» — и делал это снова на следующей точке. Скучно? Бывало. Полезно? Очень. Я узнавал город. Видел корпоративные офисы изнутри — как они устроены, как в них думают, чего боятся, что ценят. Учился разговаривать с людьми, которые злятся на технику. А злятся они, как правило, не на технику — на себя, на потерянное время, на ощущение беспомощности. И если ты это понимаешь, разговор идёт совсем по-другому.

Через год стало понятно: этот уровень пройден. Рутина — отличный учитель, но плохой работодатель на всю жизнь. Пора двигаться дальше.

ТЕЛЕФОН, КОТОРЫЙ РВАЛСЯ

Следующую работу я искал девять месяцев.

Не девять дней. Девять месяцев.

Снова отклики, снова молчание, снова «мы вам перезвоним». Снова собеседования, где меня гоняли по теории, которую я знал, но которая в реальной работе нужна редко. Я начинал понимать правила игры — но они мне не нравились.

И вдруг — другое собеседование. Вместо теоретических вопросов — практические задачи. «Покажи, что умеешь». Просто, по делу, без лишнего. Меня предупредили честно: работы будет много, темп высокий. Я так же честно ответил: готов. Через месяц проверок — взяли. В компанию больше тысячи сотрудников.

Я был искренне рад. Почти наивно рад — как человек, который долго стучался в закрытую дверь и наконец услышал щелчок замка.

То, что за этой дверью ждёт настоящий хаос — я узнал в первый же день.

Огромная база клиентов. Техника, настроенная наперекосяк — у каждого компьютера своя история, своя версия программ, свои костыли. И телефон. Телефон, который звонил. Не переставая.

К вечеру первого дня у меня гудели уши. Я разговаривал больше, чем за несколько предыдущих месяцев вместе взятых. Это было что-то среднее между боевым крещением и добровольной пыткой.

Но мы не просто отвечали на звонки — мы разбирались с причинами. Настраивали всё по единому шаблону. Одна версия операционной системы, одна конфигурация, один по-

рядок. Никакого зоопарка программ. Через год — парк техники обновлён, звонков стало в разы меньше. Двадцать в день превратились в три-семь. Появилось время думать.

И именно тогда я начал думать о том, чтобы создавать — а не только поддерживать.

ПЕРВЫЕ СТРОЧКИ КОДА

Сначала — базы данных.

Не потому что кто-то сказал. Просто была задача: огромный список компьютеров, серийные номера, пользователи. Вытаскивать нужное руками — часами. Я начал писать запросы. Потом скрипты в Excel. Маленькие, примитивные — но они работали. Данные появлялись сами, документы генерировались автоматически.

Это было моё первое настоящее ощущение от кода: ты написал несколько строк — и машина делает то, что раньше делал человек. Часами. Снова и снова.

Коллеги-разработчики иногда подходили, смотрели через плечо, объясняли. Я смотрел на их код — и не понимал почти ничего. Несколько сотен строк, которые ты не писал и не знаешь — это страшно. Рука не поднимается изменить даже пробел. Кажется, что всё рухнет.

Этот страх живёт в каждом начинающем разработчике. Страх сломать, страх не починить, страх оказаться недостаточно умным. Я прожил с этим страхом восемь лет.

Восемь лет в поддержке. Ипотека, стабильная зарплата, корпоративные поездки, бонусы в конце года. Я ходил на работу с удовольствием — это не было мучением. Скорее привычным, удобным хобби, за которое ещё и платят. Но где-то глубоко, под слоем комфорта и стабильности, тихо жило то самое детское ощущение: хочу создавать. Не поддерживать чужое. Своё.

«Единственный способ делать великую работу — любить то, что делаешь. Если ещё не нашёл — продолжай искать. Не останавливайся.»

— Стив Джобс

Я продолжал. Просто очень медленно.

РЕШЕНИЕ, КОТОРОЕ ОТКЛАДЫВАЛ ВОСЕМЬ ЛЕТ

В какой-то момент стало понятно: или сейчас, или никогда.

Не потому что случилось что-то драматическое. Просто цифра «восемь лет» в какой-то день прозвучала в голове по-новому. Восемь лет — это не мало. Это треть взрослой жизни.

Я пошёл учиться программированию. Серьёзные курсы, почти два года. Python с нуля — логика, структуры данных, алгоритмы, веб-разработка. Финальный проект — аналог HeadHunter, с авторизацией, поиском, базой данных.

И случилось то, чего я не ожидал.

Впервые в жизни я учился чему-то — и понимал, что это моё. Не «надо», не «пригодится», не «так правильно». А именно моё. То, чем хочу заниматься. Это странное чувство, когда учёба перестаёт быть усилием и превращается в удовольствие — я запомнил его навсегда.

«Инвестиции в знания всегда платят наилучшие дивиденды.»

— Бенджамин Франклин

ПЕРВЫЙ ВКУС

После курсов — первый настоящий проект.

Друг предложил идею: сайт-конструктор для фармацевтики. Вносишь информацию — сохраняется в базе, всё структурировано, удобно. Я тогда не до конца понимал, насколько это сложно. Но согласился. Взялись втроём. Потом вдвоём. Потом я остался один — партнёры ушли по разным причинам, заказчик не горел желанием торопиться.

По ночам, после основной работы — два-три часа за кодом. Это было странное время. Усталость, иногда отчаяние, когда что-то не работало и непонятно почему. И одновременно — что-то живое. Ощущение, что ты создаёшь не для галочки, а для людей, которые завтра этим воспользуются.

Когда ты один ночью пишешь код, который завтра будут использовать живые люди — это совершенно другое. Не учё-

ба. Не упражнение. Настоящее.

Полтора года спустя заказчик предложил выкупить права на приложение. Я продал. Получил деньги — и вместе с облегчением почувствовал что-то похожее на потерю. Как будто отдал часть себя. Договорились, что моё участие продолжится в виде доработок — это немного смягчило.

Потом я решил: надо двигаться дальше. В голове была логичная мысль — чем больше языков знаешь, тем больше стоишь на рынке. Так думал тогда. Ошибался.

Перешёл в новую компанию уже на должность разработчика. Основной язык — C#. Не Python, с которым я сработался и который чувствовал как свой. Другой синтаксис, другая логика, другой способ думать о коде. Я рассчитывал быстро освоиться — в конце концов, программирование везде программирование, разве нет?

Оказалось — нет. Не везде.

Переключаться между двумя языками одновременно — это не просто «другой синтаксис». Это как думать на двух иностранных языках сразу: пока ищешь слово на одном, второй уже начинает путаться. Днём C#, вечером Python для своих проектов — к ночи голова отказывалась работать на обоих. Я путался в синтаксисе, делал глупые ошибки, которых раньше не делал, злился на себя.

Постоянные созвоны, задачи, которыми потом никто не пользовался, ощущение, что создаёшь ради создания. Бег по кругу с новым языком, который так и не стал своим.

Со временем понял: знать много языков поверхностно — хуже, чем знать один глубоко. Глубина важнее ширины. По крайней мере — на старте.

Однажды утром я просто понял: хватит.

Написал заявление. Попрощался. Закрыл ноутбук с логотипом компании.

И открыл свой — уже дома, уже для себя. С чистого листа. С полной свободой. И со страхом, который, впрочем, меня уже не пугал так, как раньше. Сломаешь — починишь. Это просто код. Финансовая подушка была — и это давало время. Но ждать у моря погоды я не умею.

* * *

В следующей главе: первый реальный заказ — и почему «три часа работы» никогда не бывает тремя часами.

ЧАСТЬ II
Первые проекты, первые клиенты

ГЛАВА 2

Первый проект. Первая кровь.

Про то, как «простая программка» превращается в полноценный сайт, банк отказывается сотрудничать, а клиент под одной строчкой ТЗ подразумевает целый конструктор резюме.

Первый самостоятельный проект похож на первый прыжок с парашютом.

Ты готовился. Читал инструкцию. Смотрел как делают другие. Всё понятно, всё логично. Но когда оказываешься в воздухе — земля выглядит совсем не так, как на схеме. И ты понимаешь, что теория и практика — это два разных языка.

Заказчик хотел аналитику по своему бизнесу: видеть ключевые показатели, понимать что происходит с деньгами. Я предложил Telegram-бота — элегантное решение. Никаких сложных серверов, никакого хостинга, простая интеграция с банкингом. Клиент согласился. Я всё сделал.

Клиент посмотрел. Подумал. И сказал: «Это дико неудобно. Хочу нормальный сайт».

Вот тут я и выучил первое правило фриланса, которое не напишут ни в одном учебнике: то, что кажется тебе очевидным решением — клиенту может казаться чем угодно. И наоборот. Никогда не угадаешь заранее.

TELEGRAM-БОТ, КОТОРЫЙ СТАЛ САЙТОМ

Мы уже подписали договор. Технически — я был прав. Мог упереться.

Но что-то остановило. Может, я сам где-то не так объяснил? Может, неверно понял задачу?

Я взял и переделал. Перенёс всю логику из бота на вебстраницу. Бесплатно, за счёт своего времени. Потому что думал: так правильно. Так честно.

Наладилось не всё.

Один из банков, с которым нужна была интеграция, жил в своём особом измерении. Он не отказывал — он просто не отвечал. Дни шли, я писал вежливые письма, ждал. Тишина. Потом снова письмо. Снова тишина. Это особая форма пытки — ждать ответа от человека, который тебя игнорирует, и при этом оставаться вежливым.

«Терпение — это не пассивное ожидание. Это активное принятие процесса.»

— Рэй Далио

Я принял. Куда деваться. Но вывод сделал навсегда: если

в проекте есть интеграция с банком — закладывай в сроки в два раза больше. Банки живут в своём времени. Оно медленнее нашего.

Урок 1: Сначала продумай архитектуру и дизайн до последней кнопки — потом называй цену и сроки. Никак не наоборот.

КОНВЕРТЕР ВАЛЮТ, КОТОРЫЙ ВЫРОС В ПЛАТФОРМУ

Тот же клиент предложил второй проект. Приложение для конвертации валют. Дизайна не было — я делал как видел, как казалось правильным. Клиент смотрел, кивал, говорил «окей, но надо немного доработать». Я дорабатывал. Он снова кивал. Снова «немного доработать».

Снежный ком, который начинается как маленький шарик — и катится. Простой конвертер превратился в полноценный многостраничный сайт с функциями, которых изначально не планировалось. А я всё продолжал добавлять — потому что «это же быстро».

В разработке это явление называется scope creep. В переводе на русский — «а давайте ещё вот это добавим, это же быстро». Никогда не быстро. Никогда.

«Простота — это высшая форма изощрённости.»

— Леонардо да Винчи

В итоге клиенту кто-то сказал, что сайт не современный,

что нужен дизайнер. Я технарь — могу реализовать всё что угодно по готовому макету. Но придумать красивое с нуля, выстроить визуальную иерархию, выбрать правильные цвета — это другой склад ума. Мой — не такой. И это не слабость. Это просто честность с собой. Я нашёл дизайнера, и через две недели приступил к полной переработке сайта.

Урок 2: Если ты не дизайнер — не притворяйся им. Дизайнер в бюджете — это не расход, это страховка от переделок.

ТЗ НА ОДНУ СТРОЧКУ

Один клиент написал заявку. Бюджет приличный, задача, казалось бы, понятная: сайт с личным кабинетом. Мы обсудили детали, договорились о цене, я начал делать.

Спустя неделю работы — уточнение.

В ТЗ была строчка: «создать резюме как на странице». Я прочитал её как «добавить кнопку „скачать резюме“». Просто. Понятно.

Клиент имел в виду полноценный конструктор резюме. С шаблонами. С редактором полей. С предпросмотром. С экспортом в разные форматы. Отдельный продукт внутри продукта. И это было лишь одно из нескольких таких недоразумений!

Это был один из самых ярких уроков, которые мне преподал фриланс. Никогда не додумывай за клиента. То, что

кажется тебе очевидным прочтением одной фразы — у него в голове может быть целой вселенной возможностей.

Я сказал честно: за такой объём цена будет другой. Клиент был недоволен — считал, что я меняю условия на ходу. Я считал, что изначально не было достаточно деталей. Мы оба были правы. И оба неправы.

В итоге я оставил ему схематические макеты и прописанную логику работы — и вышел из проекта. Спустя девять месяцев сайт так и не обновился. Видимо, никто другой тоже не согласился на те условия.

Урок 3: Перед тем как назвать цену — задай все вопросы. Особенно те, на которые ответы кажутся очевидными. Именно там прячутся сюрпризы.

ПРО БАГИ В ПРОДАКШН

Есть соблазн, которому я поддавался не раз.

Правишь маленькую вещь. Одна строчка, буквально. «Ничего не сломается» — думаешь ты. Деплоишь без тестирования. Потому что торопишься. Потому что уверен. Потому что в голове ещё три других проекта и нет времени проверять.

Ломается. Всегда. Без исключений.

Особенно когда проектов несколько. Ты переключаешься между ними, держишь в голове разные архитектуры, разных клиентов, разные договорённости. И в какой-то момент пе-

рестаёшь помнить, что где. Правишь в одном месте — ломается в другом. Правишь там — ломается в третьем.

Правило, которое я выработал болью: даже если правка занимает две минуты — трать ещё пять на тест. Это всегда дешевле, чем потом объяснять клиенту, почему то, что работало вчера, не работает сегодня.

* * *

В следующей главе: объявление выложено, портфолио готово. Осталось подождать заказов. Это была моя первая ошибка во фрилансе.

ГЛАВА 3

ТЫ НЕДОСТАТОЧНО НАСТОЯЩИЙ

Или: почему каждый разработчик хотя бы раз думал, что его вот-вот разоблачат.

Есть одна вещь, о которой не принято говорить на собеседованиях, в резюме и в профилях на LinkedIn.

О том, что ты не уверен — настоящий ли ты разработчик.

Не в смысле юридическом. А в том самом, внутреннем. Когда смотришь на чужой код — красивый, лаконичный, непонятный — и думаешь: «я так никогда не напишу». Когда коллега с лёгкостью называет термины, которые ты слышишь впервые. Когда клиент задаёт вопрос — и ты знаешь ответ, но почему-то мнёшься, как будто боишься оказаться неправым.

Это называется синдром самозванца. И у разработчиков он встречается, наверное, чаще, чем в любой другой профессии.

ПОЧЕМУ ИМЕННО У НАС

Программирование — странная профессия. Она одновременно очень широкая и очень глубокая. Можно написать ра-

бочее приложение и при этом не знать, как работает компилятор. Можно знать три языка программирования — и не разбираться в четвёртом. Можно отлично делать бэкенд и совершенно не понимать в инфраструктуре.

И вот эта бесконечность создаёт ловушку. Ты всегда видишь то, чего не знаешь. Всегда. Потому что область необъятна — и как бы ты ни учился, горизонт не приближается.

Плюс — Stack Overflow, GitHub, технические конференции. Везде люди демонстрируют лучшее. Никто не постит в Twitter: «сегодня три часа не мог понять почему не работает запрос — оказалось, опечатка». Все постят элегантные решения. И у тебя складывается ощущение, что все вокруг знают что-то, чего не знаешь ты.

Я помню, как первый раз показал свой код опытному разработчику. Сердце билось так, будто я сдавал экзамен, к которому не готовился. Ждал, что он скажет: «это написано неправильно, ты вообще понимаешь, что делаешь?»

Он посмотрел. Покивал. Сказал: «в целом нормально, вот здесь можно было сделать проще».

Всё. Никакого разоблачения.

Но ощущение никуда не делось — просто притихло до следующего раза.

КАК ЭТО ВЫГЛЯДИТ НА ПРАКТИКЕ

Синдром самозванца у разработчика — это не просто

скромность. Это конкретное поведение, которое мешает работать.

Ты долго не отправляешь код на ревью, потому что «ещё не готово». Хотя на самом деле — готово. Просто страшно.

Ты занижаешь цену, потому что «ну я же не настоящий специалист». Хотя задачу решаешь не хуже тех, кто берёт втрое больше.

Ты молчишь на митинге, хотя видишь проблему. Потому что вдруг ошибаешься, и это заметят.

Ты берёшь проекты ниже своего уровня — потому что в сложных боишься облажаться.

Я делал всё это. Иногда делаю до сих пор. Просто теперь замечаю — и спрашиваю себя: это реальная неуверенность или привычный страх?

ЧТО С ЭТИМ ДЕЛАТЬ

Первое и самое важное: понять, что это не только у тебя.

Я не знаю ни одного разработчика — ни одного, который работал бы достаточно долго — кто бы не сталкивался с этим. Джуниоры боятся, что недостаточно знают. Мидлы боятся, что не тянут до синьора. Синьоры боятся, что их решения неправильные. Это не проходит с опытом — просто меняет форму.

Второе: отделять факты от ощущений. Ощущение «я недостаточно хорош» — это не факт. Факт — это «вот зада-

ча, вот я её решил или не решил». Клиенты возвращаются? Задачи закрываются? Код работает? Это факты. Остальное — шум в голове.

Третье: делать несмотря на страх. Не ждать, пока страх пройдёт — он не пройдёт. Просто начать делать. Страх проходит только через действие, никак иначе.

«Смелость — это не отсутствие страха. Это решение, что что-то важнее страха.»

— Нельсон Мандела

Я всё ещё иногда открываю сложный проект и думаю: «а вдруг не справлюсь». А потом просто начинаю. И в девяти случаях из десяти — справляюсь. В остальном обсуждаем с клиентом вместе: иногда просят сделать так, что по логике работать не будет — тогда разбираем, находим решение, которое работает.

Самозванец живёт только в ожидании. В работе — его нет.

* * *

В следующей главе: первый день в должности разработчика — и почему эйфория заканчивается быстрее, чем хотелось бы.

ГЛАВА 4

Разработчик — это звучит гордо (нет)

Про митинги ради митингов, правила которые никто не соблюдает, смену курса каждый понедельник — и момент, когда понимаешь: всё это время тебе продавали красивую картинку.

Когда получаешь должность разработчика — кажется, что добрался.

Позади принтеры и бесконечные звонки. Впереди — настоящее дело. Создавать, а не чинить. Строить, а не поддерживать. Первые недели — лёгкая эйфория. Ты среди своих. Ты делаешь то, чего хотел.

Потом начинается обычная жизнь.

И оказывается, что «разработчик в компании» — это не совсем то, что ты себе представлял из-за экрана учебного курса.

МИТИНГ, КОТОРЫЙ МОГ БЫТЬ ПИСЬМОМ

Каждую неделю — общий созвон. Иногда чаще. Все включают камеры. Кто-то ест. Кто-то смотрит в другой экран. Обсуждают в основном одно: кто за что отвечает и кто что должен сделать.

То, что можно написать в одном сообщении за три минуты, занимает сорок пять. И это ещё короткий созвон.

«Время — самый ценный ресурс в мире. Нельзя хранить его, брать в аренду или купить. Можно только тратить — правильно или нет.»

— Маргарет Тэтчер

В корпоративном мире существует негласный закон: чем больше митингов — тем меньше реальной работы. Я долго не хотел в это верить. Потом принял.

Урок 1: Если тебе нужен ответ на конкретный вопрос — пиши. Если нужно согласие нескольких людей — звони. Во всех остальных случаях — письмо.

СЕГОДНЯ ДЕЛАЕМ ОДНО, ЗАВТРА — ДРУГОЕ

Митинги — это раздражение. Но сильнее выматывала смена курса.

Конец ознакомительного фрагмента.