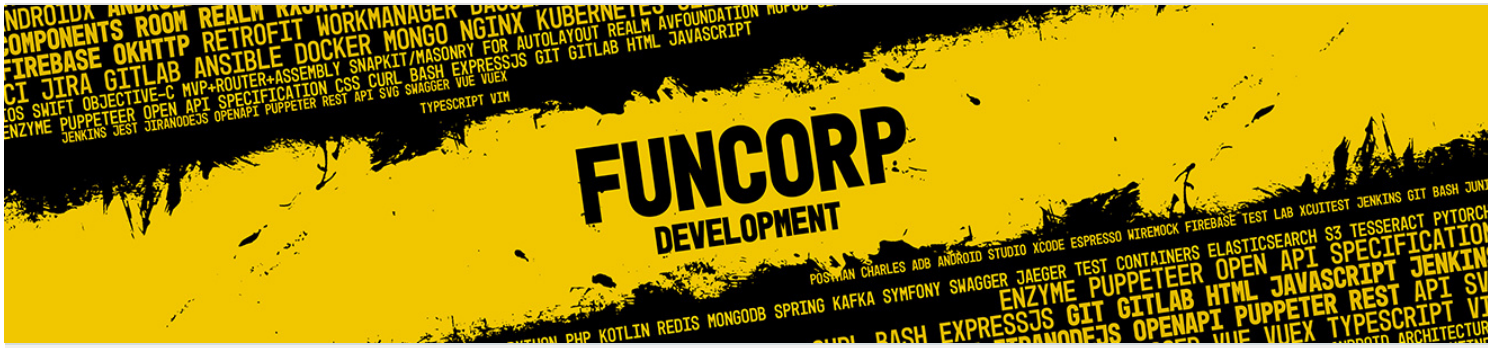


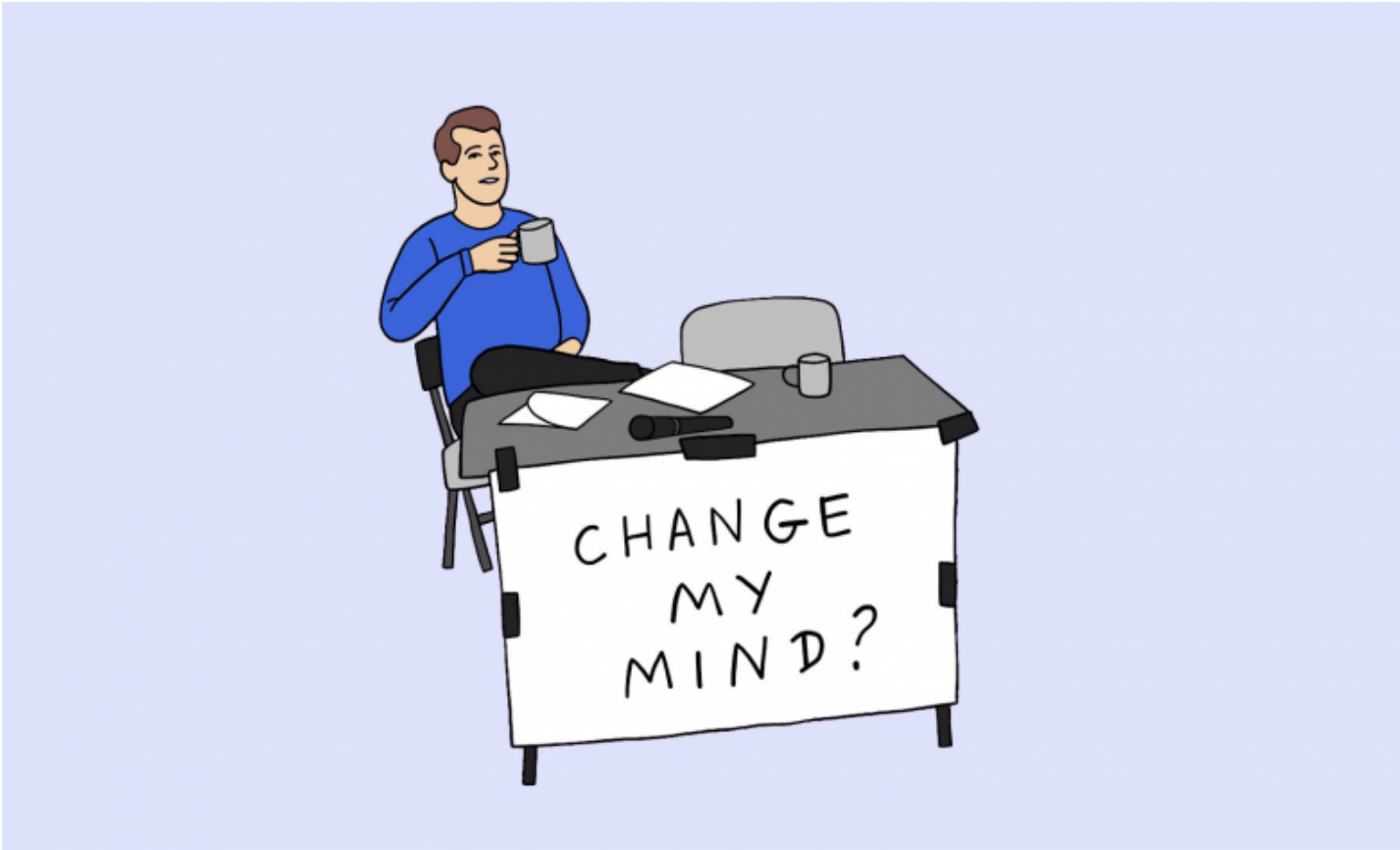
SergeAx вчера в 15:12

20 вещей, которые я узнал за 20 лет работы инженером-программистом

Блог компании FUNCORP , Программирование *, Карьера в IT-индустрии , Читальный зал

Перевод

Автор оригинала: Justin Etheredge



инженера-программиста в 20 тезисов. Я работаю в коммерческой разработке ПО больше 25 лет, и

этот текст отозвался во мне практически каждой буквой — большинство советов я тоже регулярно практикую, не облекая их в формат ёмких афоризмов. В общем, решил сделать перевод.

Особенно отзываются пункты «стройте компактные системы» и «лучший код — это отсутствие кода». Последний совет я превращаю в цитату из какого-то второсортного фильма про самураев: «Лучшая победа — та, которую ты одержал, не доставая меч из ножен» (думаю, сослуживцы за моей спиной уже закатывают глаза). И, конечно, бесконечные разговоры про легендарных 10x-программистов постоянно хочется прервать советом не связываться с 0,1x-программистами (которые реально существуют, в отличие от 10x).

Дисклеймер от автора оригинальной статьи

Учиться на чужом опыте и ошибках очень важно, но мы часто забываем, что большинство советов имеют контекст, который далеко не всегда учитывается.

«Вам просто нужно повысить цены» — говорит компания, которая 20 лет работала в бизнесе, выставляя поначалу низкие цены для привлечения клиентов.

«Вам нужно внедрять микросервисы» — говорит компания, которая построила монолит, набрала тысячи клиентов и перешла на микросервисы, когда столкнулась с проблемами масштабирования.

Без понимания контекста советы бессмысленны или, что ещё хуже, вредны. Если бы эти компании последовали собственным рекомендациям в начале пути, они, скорее всего, навредили бы сами себе.

Для понимания контекста расскажу, откуда берутся советы в этой статье. Первую половину карьеры я работал инженером-программистом в небольших компаниях и стартапах, потом перешёл в консалтинг и работал в нескольких действительно крупных компаниях. Затем основал Simple Thread, которая выросла из команды 2 человек до 25. 10 лет назад мы работали в основном с малым и средним бизнесом, сейчас — со средним и большим.

Советы в этой статье от человека, который:

- почти всегда работал в небольших командах, где приходится делать много, имея очень мало;
- ценит работающие решения выше конкретных инструментов;
- постоянно начинает новые проекты, но поддерживает ряд систем;
- ценит производительность инженера выше большинства других критериев.

Мой опыт за последние 20 лет сформировал отношение к разработке и убедил меня в некоторых утверждениях, которые я оформил в виде списка. Надеюсь, он вам будет полезен.

1. Я все ещё многого не знаю

«Как ты можешь не знать, что такое BGP?» или «Ты никогда не слышал о Rust?» — некоторые из нас не раз слышали подобное.

Причина, по которой многие любят разработку, заключается в том, что мы учимся всю жизнь. И в создании софта есть огромные области новых знаний, которые с каждым днём только растут. Можно десяток лет работать программистом и все равно иметь огромный пробел в знаниях по сравнению с кем-то, кто также провел десятилетия в, казалось бы, аналогичной роли.

Чем раньше вы это поймете, тем быстрее сможете избавиться от синдрома самозванца и с удовольствием учиться у других.

2. Самое сложное в разработке — сделать продукт, который действительно нужен

Знаю, что это уже стало клише, но многие инженеры скептически относятся к данному пункту, поскольку считают, что он обесценивает их труд. Чушь, на мой взгляд. Напротив, этот момент помогает подчеркнуть, в насколько сложной и иррациональной среде нам приходится работать, ведь именно эти аспекты так усложняют нашу деятельность.

Можно разработать самую технически впечатляющую вещь в мире, а потом никто не захочет её использовать. Такое случается постоянно. Проектирование софта часто связано со слушанием — нам приходится быть сразу инженером-программистом, экстрасенсом и антропологом.

Инвестиции в процесс проектирования (с помощью UX-специалистов или путём самообразования) принесут огромные дивиденды. Ведь как реально подсчитать стоимость разработки неправильного ПО? Это гораздо больше, чем просто потерянное время инженера.

3. Лучшие инженеры-программисты думают как дизайнеры

Великие инженеры-программисты глубоко задумываются о пользовательском опыте. Они могут не думать об этом в терминах вроде внешнего или программного API, UI, протокола или любого другого интерфейса. Великие инженеры думают о том, кто будет его использовать, почему он будет использоваться, как он будет использоваться и что важно для пользователей. Учет потребностей конечного потребителя — это залог хорошего пользовательского опыта.

4. Лучший код — это отсутствие кода или код, который не нужно поддерживать

Всё, что нужно сказать: кодеры будут кодить. Если спросить человека любой профессии, как решить ту или иную проблему, он выберет то, что у него хорошо получается. Это просто человеческая природа. Большинство инженеров-программистов всегда будут склоняться к написанию кода, особенно когда нетехническое решение не очевидно.

То же самое относится к коду, который не нужно поддерживать. Инженерные команды часто хотят изобрести колесо, когда оно уже существует. Здесь нужно соблюдать баланс, есть много причин для

развития собственных разработок, но остерегайтесь токсичного синдрома «изобретено не здесь».

5. Программное обеспечение — это средство достижения цели

Основная работа любого инженера-программиста — предоставление ценности. Немногие разработчики понимают это, еще меньше тех, кто это осознает. А ведь осознание приводит к другому способу решения проблем и другому взгляду на инструменты.

Если вы действительно верите, что выбор ПО зависит от цели, то будьте готовы найти «правильный инструмент для работы», который может и не являться софтом.

6. Иногда нужно перестать точить пилу и просто начать пилить

Некоторые сразу бросаются в проблему и начинают писать код. Другие — начинают исследовать и попадают в аналитический паралич. В таких случаях установите дедлайн и начните изучать решения. Вы гораздо быстрее узнаете что-то, когда непосредственно станете решать задачу.

7. Если у вас нет представления о границах возможного, вы не сможете спроектировать хорошую систему

С этим я часто сталкиваюсь, поскольку мои обязанности уводят меня все дальше и дальше от повседневной разработки ПО. Следить за командой разработчиков — огромный объём работы. И если вы не понимаете, на что способна ваша команда, то сможете разрабатывать решения только для самых простых проблем. И опасайтесь тех, кто давно не писал никакого кода.

8. Каждая система в конечном счёте отстой, смиритесь с этим

У Бьерна Страуструпа есть цитата: «Есть только два вида языков: те, на которые все жалуются, и те, которыми никто не пользуется». Это можно распространить и на большие системы. Не существует «правильной» архитектуры, вы никогда не закроете весь техдолг, не разработаете идеальный интерфейс, ваши тесты всегда будут слишком медленными.

Это не оправдание тому, чтобы не делать что-то лучше, а наоборот, способ дать вам перспективу. Меньше беспокойтесь об элегантности и совершенстве, стремитесь к постоянному улучшению и созданию пригодной для жизни системы, в которой вашей команде нравится работать и которая стабильно приносит пользу.

9. Никто не спрашивает «почему» в достаточной степени

Используйте любую возможность поставить под сомнение предположения и подходы, которые «всегда делались так, как надо». В команде появился новый сотрудник? Обратите внимание, где он запутался и какие вопросы задает. Поступила заявка на новую функцию, которая не имеет смысла? Убедитесь, что вы понимаете цель и то, что требует эту функциональность. Если не получаете чёткого ответа, продолжайте спрашивать пока не поймёте.

10. Сосредоточьтесь на том, чтобы избежать 0,1x-программистов, а не найти 10x-программистов

10x-программист — это глупый миф. Идея о том, что кто-то может сделать за 1 день то, что не менее компетентный, трудолюбивый, и такой же опытный программист может сделать за 2 недели, глупа.

Я видел программистов, которые пишут в 10 раз больше кода, а потом вам приходится исправлять его в 10 раз дольше. Кто-то может быть 10x-программистом только в том случае, если вы сравниваете его с 0,1x-программистом — тот, кто тратит время, не просит обратной связи, не тестирует свой код, не рассматривает крайние случаи и так далее. Нужно заботиться, чтобы не допустить 0,1x-программистов в команду, а не искать мифического 10x-программиста.

11. Одно из главных отличий между сеньором и джуном — свое мнение о том, как все должно быть

Ничто не беспокоит меня больше, чем сеньор, не имеющий никакого мнения о рабочих инструментах или о том, как подходить к созданию программного обеспечения. Лучше высказать мнение, с которым я буду категорически не согласен, чем не иметь мнения вообще.

Если вы используете инструменты, но ненавидите их, вам нужно изучить больше. Попробуйте другие языки, библиотеки и парадигмы. Существует мало способов повысить уровень своих навыков быстрее, чем активный поиск того, как другие решают задачи с помощью инструментов и методов, отличных от ваших.

12. Людям не нужны инновации

Люди много говорят об инновациях, но обычно они ищут дешёвые преимущества и новизну.

Если вы действительно привносите новшества и меняете то, что людям привычно — ждите отрицательных отзывов. Но если верите в то, что делаете, и знаете, что это действительно улучшит ситуацию, приготовьтесь к долгой битве.

13. Ваши данные — самая важная часть системы

Я видел много систем, в которых целостность данных держалась в основном на надежде. Любое отклонение от идеального сценария в такой системе, и мы получаем частичные или грязные данные, а работа с такими данными в перспективе превратится в кошмар.

Важно помнить, что данные скорее всего сильно переживут вашу кодовую базу. Приложите усилия, чтобы всё хранилось в порядке, и это окупится.

14. Ищите технологических акул

Старые технологии, которые остались на плаву, — это акулы, а не динозавры. Они настолько хороши, что пережили быстрые изменения, которые постоянно происходят в мире технологий. Заменяйте их только в том случае, если у вас есть очень веская причина. Эти инструменты не будут модными и

современными, но позволят выполнить работу без множества бессонных ночей.

15. Не путайте скромность с невежеством

Есть много программистов, которые не высказывают мнения, если их прямо не спросить. Если кто-то не высказывает свое мнение вам в лицо, это не значит, что ему нечего добавить. Иногда самых громких хочется слушать меньше всего.

Поговорите с окружающими, запросите фидбек и советы.

16. Программисты должны регулярно писать

Инженеры-программисты должны регулярно вести блог или дневник, писать документацию и вообще делать всё, что требует от них высокого уровня навыков письменного общения. Письмо помогает думать о проблемах и более эффективно общаться с командой. Это один из важнейших навыков, которым должен овладеть любой программист.

17. Рабочие процессы должны быть минимально энергозатратными

Сейчас все хотят быть agile, но «agile» — это создание вещей небольшими порциями, обучение, а затем итерация. Если кто-то пытается вложить в это гораздо больше, значит, он что-то продаёт. Для этого необязательно переставать помогать сотрудникам или отказываться от отчетности, но вы когда-нибудь слышали, чтобы люди из ваших любимых IT-компаний и масштабных опенсорс-проектов хвалились тем, какой у них клёвый Scrum? Процессы должны оставаться лёгкими и гибкими до тех пор, пока не появится потребность в большем. Доверьтесь своей команде, они всё сделают.

18. Инженеры-программисты, как и все люди, должны чувствовать ответственность

Если вы отвлечёте кого-то от результатов его работы, то он будет меньше заботиться о ней. Это почти тавтология и основная причина, почему кросс-функциональные команды работают хорошо, и почему DevOps стал таким популярным. Дело не только в передаче обслуживания и неэффективности, а в том, чтобы управлять всем процессом от начала до конца и нести прямую ответственность за предоставление ценности.

Дайте группе увлечённых людей полную ответственность за проектирование, создание и выпуск софта (или чего-то еще) — произойдут удивительные вещи.

19. Собеседования не помогут определить, насколько хорошим членом команды будет человек

Собеседования лучше проводить, чтобы понять, кто такой человек и насколько он заинтересован в нужной области знаний. Пытаться выяснить, насколько хорошим членом команды он будет, — бесполезное занятие.

То, насколько человек умён или осведомлён, также не является хорошим показателем, каким членом команды он станет. Никто не скажет вам на собеседовании, что он будет ненадёжным, жёстким

команды он станет. Никто не скажет вам на собеседовании, что он будет ненадежным, жестким, напыщенным или никогда не будет приходить на собрания вовремя. Некоторые ищут «сигналы» для таких вещей, вроде «Если они спрашивают об отгулах на первом собеседовании, значит, они точно не будут ходить на миты!», но это все ерунда. Если вы опираетесь на подобное, то просто гадаете и отбрасываете хороших кандидатов.

20. Стремитесь к созданию более компактной системы

Есть много факторов, которые будут подталкивать вас к созданию большой системы. Распределение бюджета, невозможность решить, какие функции следует сократить, желание создать «лучшую версию». Все эти вещи очень сильно подталкивают к тому, чтобы сделать слишком много. Нужно бороться с этим.

В процессе разработки вы узнаете так много нового, что в результате итераций получите гораздо более крутой результат. Для многих это крайне трудно.

Ваша история

Вот и всё: 20 лет разработки программного обеспечения превратились в 20 мудрых пунктов. Если согласны или не согласны с мыслями в статье, или если есть, что добавить и чем поделиться — пишите в комментариях.

Теги: советы бывалых, software engineer, инженер-программист, программирование, разработка, советы начинающим, software

Хабы: Блог компании FUNCORP, Программирование, Карьера в IT-индустрии, Читальный зал

Создатель и руководитель инженерных команд

Facebook Github

Комментарии 34

ПОХОЖИЕ ПУБЛИКАЦИИ

8 апреля в 13:07

Доказательная разработка или как data-driven подход добавил смысла работе

+46 6.6K 34 2 +2

10 августа 2020 в 08:03

Как будет выглядеть программирование в 2025 году?

+24 24K 54 148 +148

28 октября 2019 в 08:12

Программист-защитник сильнее энтропии

+31 7.9K 60 6 +6

ЛУЧШИЕ ПУБЛИКАЦИИ ЗА СУТКИ

вчера в 15:12

20 вещей, которые я узнал за 20 лет работы инженером-программистом

+73 14K 126 34

вчера в 09:00

Способы питания электролюминесцентных индикаторов

+58 3K 41 17 +17

вчера в 08:29

Архитектура и особенности процессора Эльбрус 2000

+45 6.8K 32 68 +68

вчера в 13:22

Создание собственного корпуса для сервера. Думаете, просто?

+30 5.7K 14 40 +40

вчера в 10:53

Сириус — белый бриллиант зимнего неба

+26 2.9K 10 7 +7

ВАКАНСИИ КОМПАНИИ «FUNCORP»

Senior Backend Software Engineer
от 250 000 до 350 000 ₽ · FUNCORP · Москва

Ведущий специалист по тестированию web-приложений
от 140 000 до 200 000 ₽ · FUNCORP · Москва

Технический писатель в команду разработчиков высоконагруженных систем
от 150 000 до 215 000 ₽ · FUNCORP · Москва

Software Development Engineer in Test (SDET)
от 140 000 до 270 000 ₽ · FUNCORP · Москва

Senior QA Mobile Engineer
от 150 000 до 200 000 ₽ · FUNCORP · Москва

Больше вакансий на Хабр Карьере

Вернуться на старую версию