

ТЕСТИРОВАНИЕ МОБИЛЬНЫХ ПРИЛОЖЕНИЙ*, РАЗРАБОТКА ПОД IOS*, XCODE*, SWIFT*,
БЛОГ КОМПАНИИ TOUCH INSTINCT

Как стать тимлидом и не взорваться

niklnd 1 июля 2017 в 14:38 👁 49,7k



Два года назад я начал негласно исполнять роль iOS-lead в компании Touch Instinct и формированием стабильной работы iOS-отдела. Спустя полгода это трансформировалось в официальную должность. Из-за отсутствия опыта у меня возникало огромное количество проблем, которые вызывали жжение в области верхней части кресла. Это происходило из-за ряда факторов:

- Нехватка опыта менеджмента.
- Отсутствие рядом компетентного человека, уже прошедшего путь становления от новой для себя роли к человеку, который

понимает устройство процессов, обязанностей и пути решения проблем.

- Нестабильность общих процессов в компании из-за её молодого возраста на тот момент.

Если вы стали лидером и первоначальная эйфория сменилась небольшим горением и унынием, то пара советов не будет лишней.

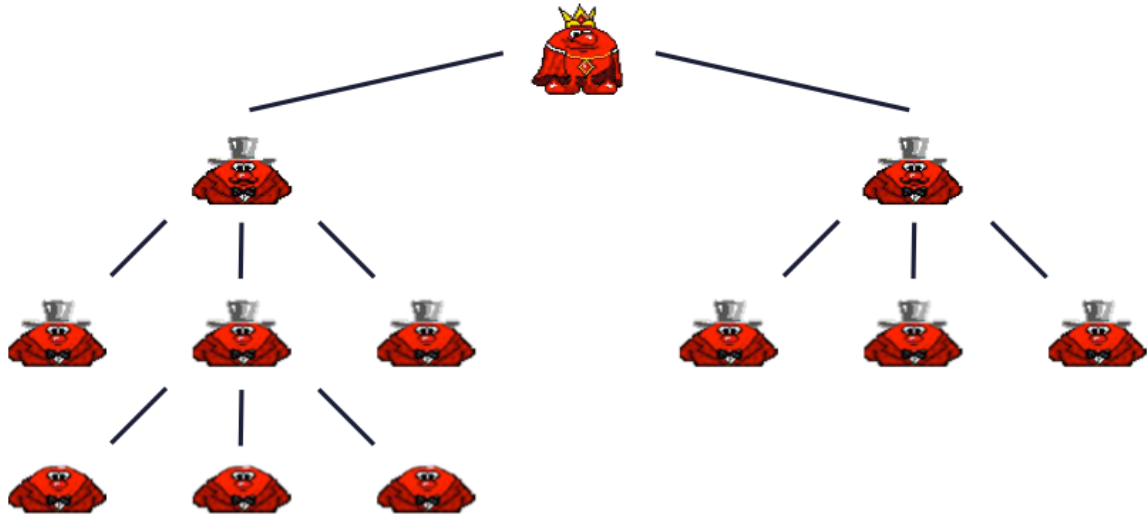
Откуда берутся лиды

Исторически сложилось, что существует две классические структурные модели управления компанией.

Первая — **горизонтальная система управления**. Её практикуют реже, к примеру basesamp или **37 signals**. Смысл заключается в том, что у вас есть ряд сильных специалистов, способных самостоятельно регулировать свою деятельность.



Вторая — иерархическая система управления.



Есть разработчик, за ним стоит platform lead, за ним СТО, далее СЕО. Каждый участник курирует определенный вектор развития. Чем ниже располагается в иерархии человек, тем за более узкоспециализированный участок он отвечает. Разработчик отвечает лишь за код, который он производит. Lead отвечает целиком за платформу и за её развитие. Технический директор отвечает за техническую составляющую в компании. А генеральный директор — за развитие компании.

Чем больше становится компания, чем больше появляется процессов и участников, тем сложнее становится иерархия. Появляются дополнительные роли, такие как Mobile Lead. У него в подчинении находятся лиды мобильных платформ, которых может быть больше одного на платформе. Это зависит от количества подчиненных на конкретном уровне в компании.

Оптимальное количество людей, которых может контролировать один человек, сильно варьируется от сферы деятельности и от модели управления. В IT в классической литературе это число колеблется от 5 до 9 человек.



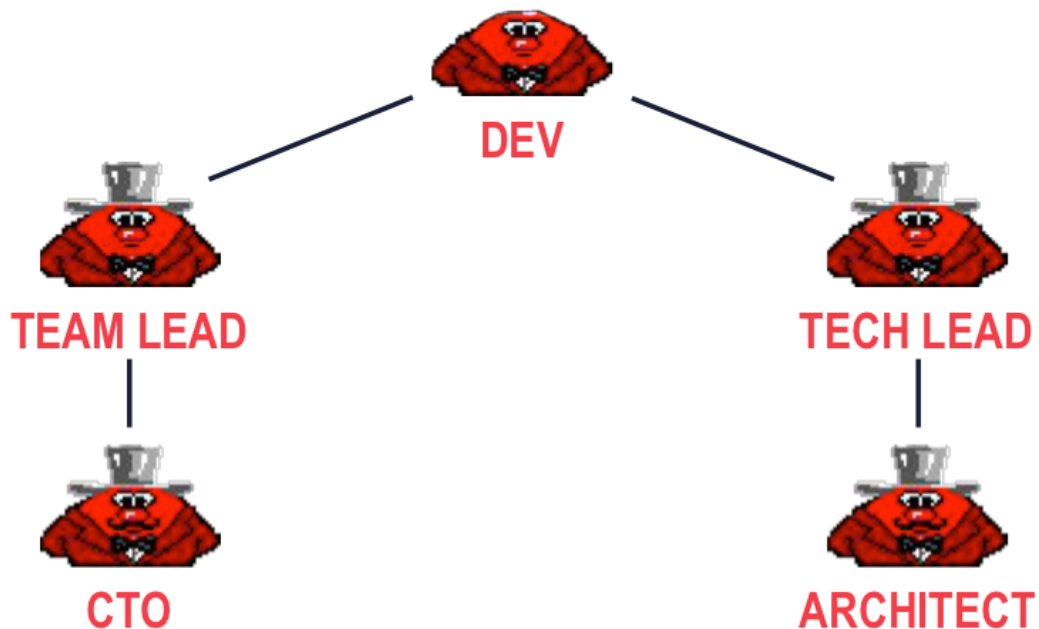
Меньшее количество людей ведет к тому, что дополнительная иерархическая роль избыточна и просто усложняет процессы.



Большее ведет к тому, что контроль, обучение и другие функции становится сложнее выполнять. Необходимо добавлять новые роли.



Рассмотрим классическую ситуацию карьерного роста в IT-компании. Когда человек достигает определенного уровня квалификации, он может либо перейти на следующий уровень иерархии при наличии определенных личностных качеств, либо сменить род деятельности/область деятельности/компанию и расти дальше в технической стезе. На картинке ниже представлена классическая краткая форма развития. Следующая ступень развития разработчика — team lead либо tech lead. Первая предполагает уход в сторону менеджмента, вторая — глубокий технический рост специалиста. Team lead дальше уходит в platform lead. Из tech lead получают архитекторы разного калибра.



Роль лида в компании

Для начала определимся, что ждет руководство от лида. В зависимости от размера компании его роль может сводиться как к управлению конкретной командой, так и развитию направления целиком. При этом необходимо быть готовым к тому, что разработчики будут требовать от вас высоких технических навыков, а управленцы более высокого звена — выполнения менеджерских задач. Чаще всего team lead это в некоторой степени программист, и в некоторой степени менеджер.

Техническая роль

Говорят, что со временем team lead начинает терять навыки программиста с течением времени. В реальности же большинство основ программирования были сформулированы более 40 лет назад. Порой эти знания более ценные, чем знание конкретного

sdk. Но чаще всего наставничество помимо помощи в самых простых программных вопросах сводится к следующему:

Построение экосистемы. Сейчас в IT бизнесе выигрывает та компания, которая предоставляет не просто продукт, а старается построить экосистему. Которая может решать множество задач бизнеса и позволяет пользоваться различными функциями, которые предоставляет компания. Это касается и Apple, и Google, и Facebook. Со временем любой бизнес начинает превращаться в платформу. Похожая ситуация наблюдается и в разработке. Вам необходимо построить для разработчика экосистему целиком, чтобы вопросов возникало как много меньше. Это касается и процессных вещей, и архитектуры. Нужно сформировать документированную систему гайдов, которые позволят новым разработчикам быстро влиться в процесс, а участникам команды не забывать принятые нормы.

Построение системы обучения. Адекватный лид должен грамотно выстраивать процесс обучения всех членов команды, правильно соотнося их собственные интересы с бизнес-задачами компании. Если человек хочет изучить сложный кастомный UI, то задача лида придумать, как это можно использовать во благо. Это касается и других направлений, будь то технические знания или менеджерский бэкграунд. Важно следить за выполнением этих задач, обычно разработчики в своей голове ставят более низкий приоритет данным задачам. Старайтесь помочь в развитии даже в тех областях, в которых у вас меньше знаний. Сводите с нужным специалистом или дайте время разобраться самому, а потом пусть расскажет другим. Так человек укрепит свои знания. Ситуация,

когда у разработчика или у вас отсутствуют знания в какой-то области, нормальна. Осознание и принятие этого способствует обучению.

Уважаемые студенты



ДА НАЧНЕТСЯ ОБУЧЕНИЕ

Привитие принципов отношения к коду. Очень важно донести до разработчика мысль, что сам код не стоит ни гроша. Лишь функциональность, которая выполняется этим кодом, имеет смысл. Отсюда и растёт вся экосистема, которую необходимо построить. Важно понимать, что бизнес диктует правила для написания кода, а не наоборот. Если у вас банковская система с крайне высокой надёжностью, то TDD вам необходим. Если же вы

пишете MVP-приложения, то смысла строить сложную систему и архитектуру на первом этапе попросту нет.

Психологическая роль

Бытует мнение, что большинство разработчиков являются интровертами. В действительности, это очень близко к истине. Благодаря этому знанию появляется несколько методик, которые можно применять в работе.

Организуйте ежемесячные встречи с разработчиками.

Сценарий таких встреч каждый месяц повторяется. Поначалу разработчик говорит, что всё ок, всё хорошо, проблем нет. Но основная сила психолога в вопросах. В беседе узнается, что и систему оценки на проекте в прошлом месяце можно улучшить, и взаимоотношения между отделами подтянуть, а ещё было придумано оригинальное решение, которое можно вынести как базовое и написать по нему гайд. Ведь на другом проекте возникли такие же проблемы и решали их дольше, чем нужно. Беседу обязательно надо конспектировать, потому что в ней могут быть отличные мысли. Их можно и нужно претворять в жизнь. После таких бесед часть проблем нивелируется. Так как процесс итеративный, он помогает устранять проблемы и не занимает много времени. Такие встречи не нужно превращать в совещания отделов. Это должны быть именно тет-а-тет беседы в спокойной обстановке. Так вы сможете решить даже сложные личностные проблемы.



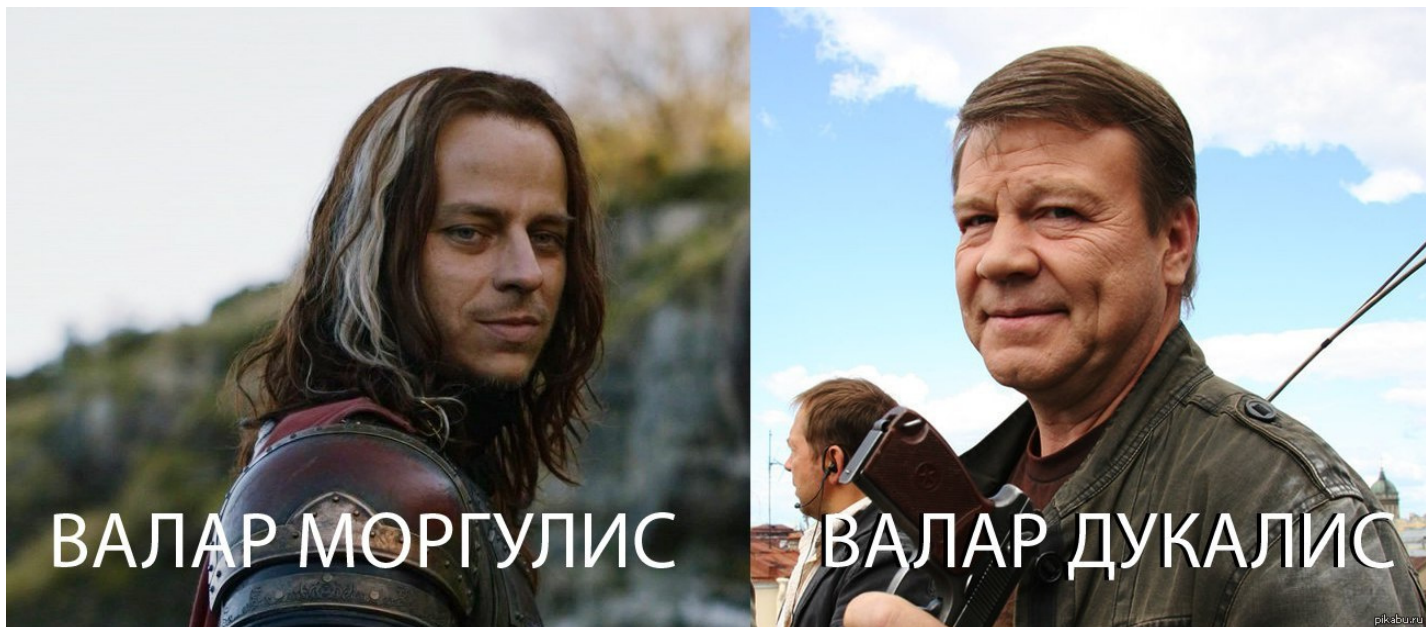
Станьте «большим братом». Lead несет ответственность за работу своего подразделения. Он должен быть в курсе задач, сроков исполнения и качества функциональности на выходе для каждого разработчика. Однако, в процессе работы не стоит впадать в крайности. Пословица «Всё хорошо в меру» работает здесь отлично. Старайтесь не вмешиваться, если всё идёт хорошо. При этом всегда держите руку на пульсе и предотвращайте кризисные ситуации. Сотрудник комфортнее всего работает, когда на него не давят сверху, но при этом он чувствует влияние невидимого «большого брата», который является не контроллером, а советчиком.



BIG BROTHER IS WATCHING YOU

COPYLEFT BY FREDERIC GUMONT, 2004. WWW.THEALCORN.COM

Внедряйте «безликий код». Самый долгий и трудный процесс — привести код вашего отдела к «безликому коду», когда нельзя по стилю определить, кто его написал. Это направление правильное и трудное. Если возможна вариативность решения, вы встретите несколько лагерей. Чтобы в случае кризисной ситуации в ваш адрес не посыпалось «вот решили тогда так, а это оказалось плохо, и сейчас все минусы всплыли», необходимо дипломатично решать каждый вопрос. Помните, что с базовыми решениями, которые вы разрабатываете и согласовываете, будут работать другие люди. Делайте удобно для них.



Как сделать, чтобы стул не сгорел раньше

Классика — когда самый продуктивный разработчик становится лидером. Руководство часто думает, что раз вы делаете фичи быстрее, то можете делать столько же, сколько и средний разработчик, плюс взять обязанности лида. Можете попросить прописать прямо в контракте, какой процент рабочего времени вы будете программировать. Зона ответственности становится совсем другой, ваши обязанности меняются. Есть два варианта развития событий.

- Начинаете работать намного больше, чтобы успеть и как разработчик, и как team lead. Обычно это ведет к перегоранию.
- Работаете столько же, сколько и раньше. В итоге не успеете сделать и фичи, и задачи лида.

На этапе согласования новой роли заранее продумайте список вопросов, ожиданий от новой роли. Спросите, чего ожидают от вас. Поймите, что вам конкретно нужно сделать и сформулируйте это.

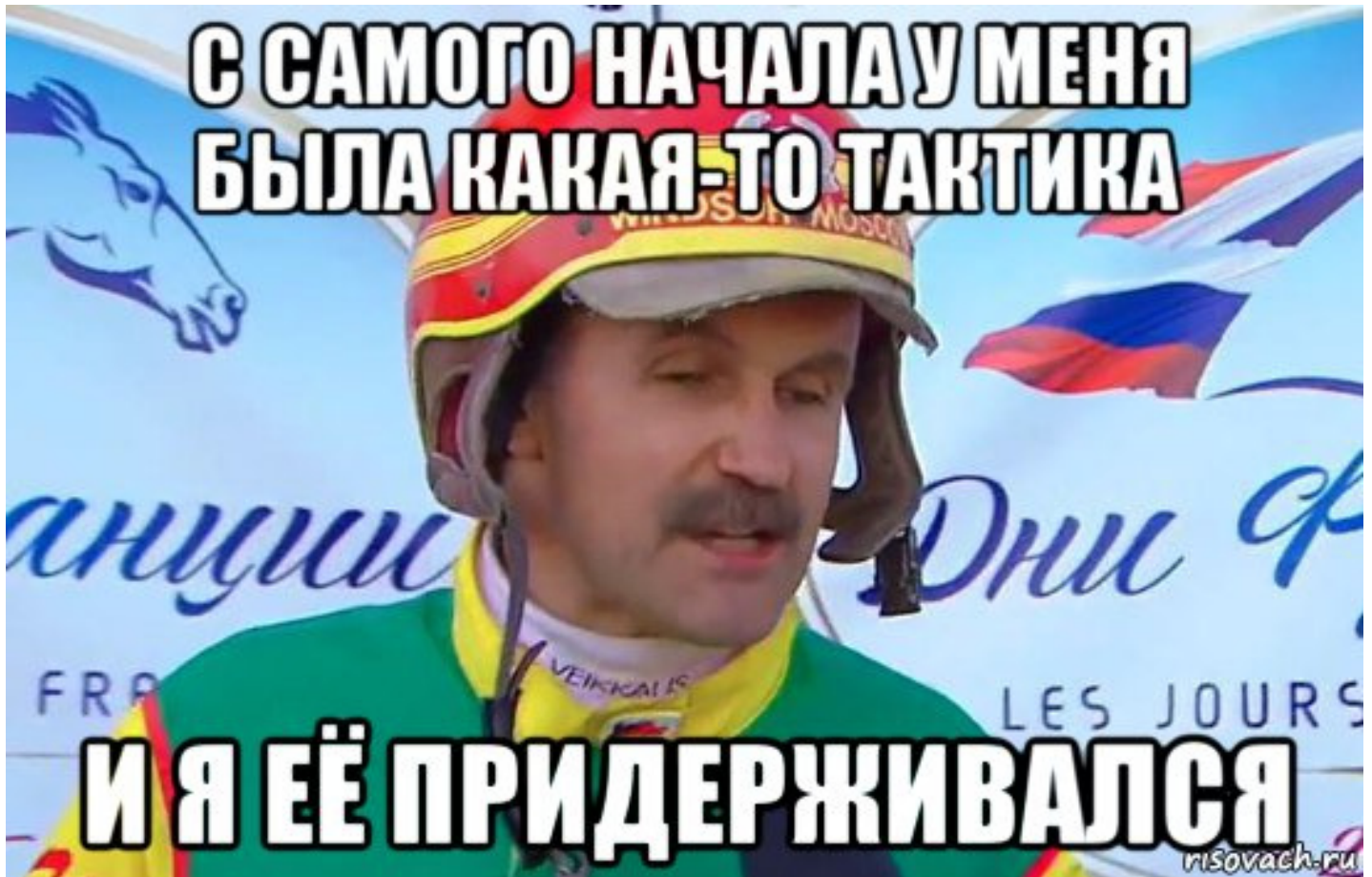
Только тогда приступайте к работе. Иначе так и будете не укладываться в сроки разработки, не наладите процессы, систему обучения и все остальные задачи. Отношения с менеджерами, разработчиками и остальными людьми в компании так или иначе ухудшатся. Вы будете думать, что вы герой, который горбатится на выходных ради блага компании. А на самом деле стали человеком, на которого нельзя положиться. Ведь неизвестно, будет ли сделана задача и насколько качественно.

Именно поэтому нужно четко регламентировать, зачем вам программировать, сколько времени и что это даст отделу и в целом компании. Если речь идет о 30% времени, в течение которых вы будете проектировать архитектуру общих решений, библиотек или стандартов — одно дело. Это поможет не заниматься рутинными задачами, не забыть код и посмотреть на него более глобально. Но если вам говорят о 70% или 90% времени, то люди просто не понимают, зачем им нужен team lead. Или заранее планируют, что вы будете работать больше 40 часов. Можете либо аргументированно объяснить, как сделать лучше, либо просто ответить отказом. Лучше всего поговорить об ожиданиях.

Составьте план развития. Необходимо сделать четкий план развития себя и отдела хотя бы на ближайший год. Это может быть наработка общих процессов, создание базовых общих решений, ведение курса лекций или создание школы. Всё исходя из потребностей, которые есть у компании сейчас. Если цель — научиться писать код, то уход в сторону архитектуры и мета-принципов. Если заявить о себе на рынке — участие в

конференциях и публикации. Если такого плана нет, то начинается ряд проблем:

- люди не понимают, чем вы занимаетесь;
- вы и сами не понимаете, чем занимаетесь и что нужно сделать. Находитесь в совершенно неконтролируемом хаосе.



Найдите наставника. Очень здорово, когда есть человек, который уже проходил этот путь. Если он может откровенно ответить на ваши вопросы — это огромный плюс и не пользоваться этим большая ошибка. Идеально, если это лид вашей платформы, которого вы лично знаете и уважаете как специалиста.



**Слушаться должен
наставника своего**

**И только тогда достигнешь
ты новых высот**

Не надумывайте. Возьмите себе за привычку вести прямой диалог в случае возникновения непонимания. Решится огромная куча проблем, при этом они не будут перерастать во что-то большее, включая личностные конфликты.

Отслеживайте выполнение. Необходимо сразу начинать направлять и контролировать процесс выполнения. Что это нам дает? То, что мы перестаем делать кажущиеся важными рутинные

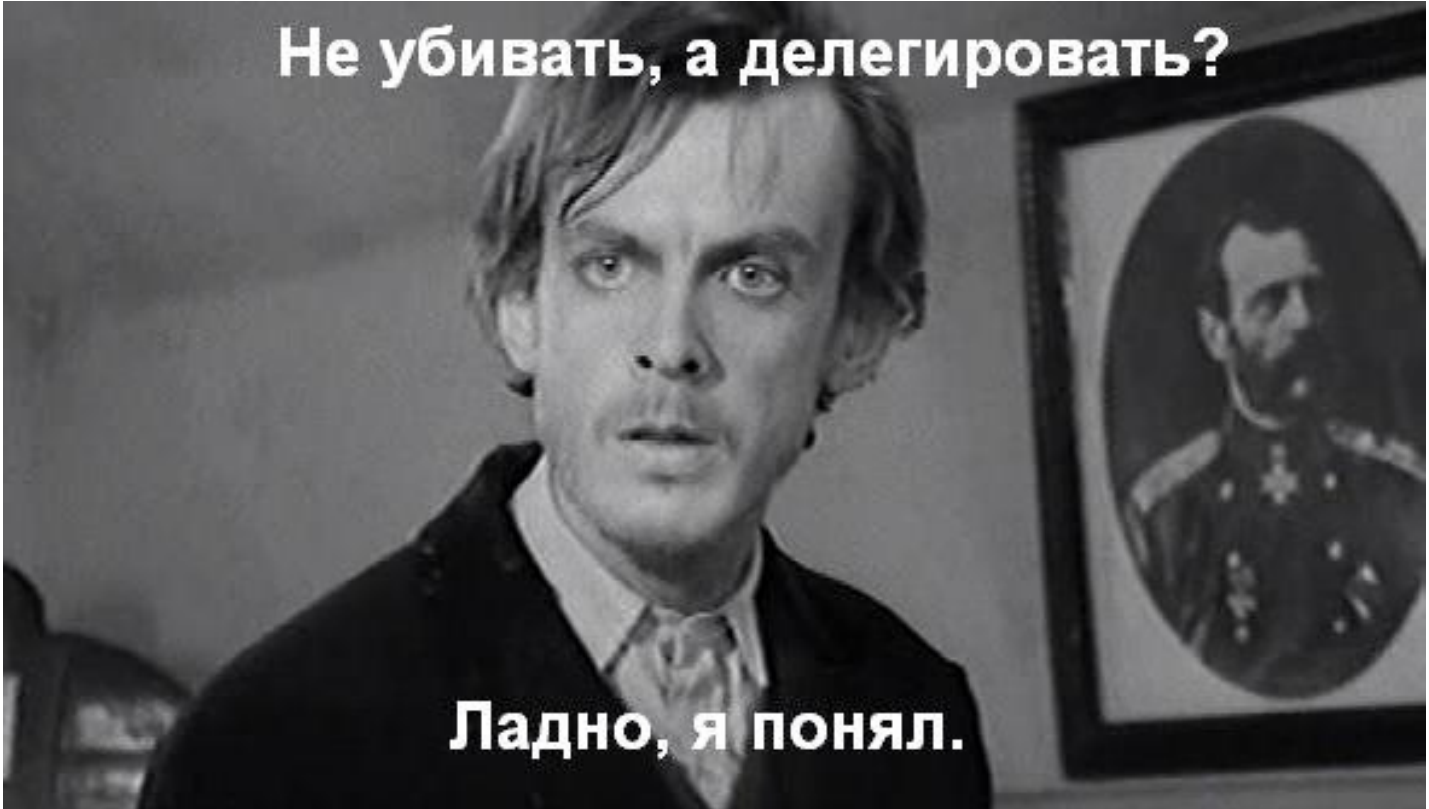
задачи за счет простого делегирования и можем выполнять задачи, которые и входят в наш план.

Автоматизируйте. Плюс различного рода оптимизации в виде CI/CD/статических анализаторов, кодогенераторов, базовых либ и так далее. Всё это экономит нам время в будущем.

Тайм-менеджмент и делегирование

Кто-то из разработчиков уходит в программный запой на неделю и возвращается с готовой фичей, а кому-то требуется ежедневный контроль. Наблюдение, анализ и логическое мышление помогут вам разобраться, кто есть кто.

Делегируйте. Главное — это понимание того, что большинство задач можно и нужно делегировать. На первом этапе может показаться, что дел стало больше. Особенно, если вы были разработчиком с ключевыми обязанностями вроде написания архитектурных решений. Вполне может оказаться, что рядом не будет человека с нужными компетенциями. Значит, компетенции надо выращивать. Возможно это будет трудно принять на первом этапе, но это единственно правильный путь в вашем положении.



Не убивать, а делегировать?

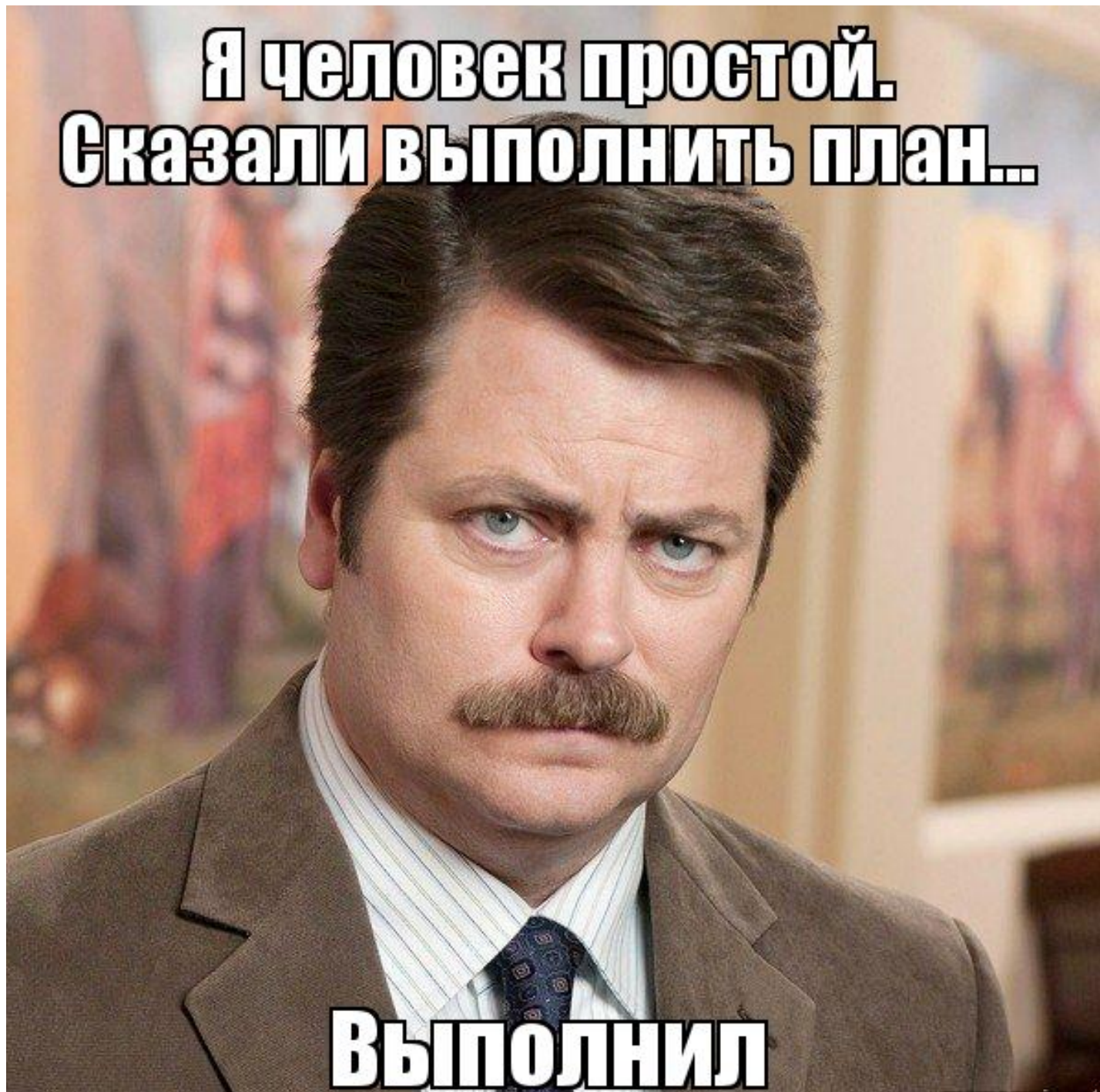
Ладно, я понял.

Контролируйте время. Следите за временем, отведенным на задачу. Речь о тактических и стратегических задачах. Говорю не о ежедневной работе, хотя она и будет являться основным фактором успеха. Под тактическими задачами подразумевается выполнение конкретных больших задач в виде создания базовых библиотек, задокументированных процессов или процесса обучения. Под стратегическими — определения вектора развития. В какой момент уйти с objective-C? Как сформировать экосистему, чтобы она работала эффективнее? Что надо сделать, чтобы эффективность решения задач бизнеса возросла? Это и есть примеры стратегических вопросов. Они наиболее сложные и наиболее длительные, но именно от них зависит ваше завтра.

Важным моментом будет являться, насколько вы понимаете, что интересно людям. Это можно использовать как дополнительный фактор и роста разработчика и проработки какого-то отдельного

компонента вашей экосистемы. Разработчики, как правило, любят заниматься интересными вещами и в свободное время. Вы можете им помочь в этом. Важно лишь понять, что действительно интересно человеку.

**Я человек простой.
Сказали выполнить план...**



Ошибки, негатив и минусы

Можно долго и красиво рассуждать о том, какой вы замечательный человек и специалист. Но факт остается фактом — не ошибается

лишь тот, кто ничего не делает. И чем больше человек начинает отходить от своей зоны комфорта, тем выше вероятность ошибки. Конкретный отход от фул-тайм разработчика к более менеджерской роли предполагает сильное взаимодействие с командой и другими отделами в компании. Вполне логично, что появляются новые нестандартные для человека ситуации. И он не всегда знает, как их решать.

При встрече с неприятной ситуацией первое, что нужно сделать — понять, что только вы можете ее исправить и никто другой. И чем дольше вы с ней затягиваете, тем сильнее в конце выстрелит пушка времени. А теперь перейдём от абстракций к конкретным примерам.

Принятие решений. Бывает, что людей в команду спускают сверху. Вам необходимо сразу четко обозначить здесь правила: либо у вас есть право вето на абсолютно все решения по построению команды, либо необходимо объяснить руководству, почему будет работать именно так. В идеале и самое часто встречающееся на практике — когда team lead сам формирует себе команду. Повышается моральная ответственность так как решения принимал сам lead.

Личностные качества. В команде есть человек, который по каким-то причинам вас не устраивает. При этом абсолютно не важно, наняли его вы, или он уже был, или дали со стороны. Все люди ошибаются и вы не исключение. Особенно на первом этапе, когда всё ваше собеседование строится не на том, чтобы понять насколько человек вообще впишется в команду, а знает ли он, как

решить алгоритмическую задачу по поиску элемента в бинарном дереве. Вы должны понимать, что любой алгоритм можно выучить за один день, любой framework при должном усердии от начального применения в первый же день до глубокого погружения в течение месяца. А личностные качества, некое «абстрактное чувство кода» за день-неделю-месяц или год не исправишь. В этом вопросе тем более не нужно ориентироваться на hr и считать, что это их работа. Потому что в итоге человек будет работать большую часть своего времени именно с вами в период работы в компании, а не с hr.

К минусам можно также отнести то, что со временем ваши технические навыки будут падать. Это и миф, и правда одновременно. Роль лида позволяет более широко взглянуть на некоторые технические аспекты, на мета-принципы программирования. А то, что вы не будете знать как запрограммировать в iOS 10 новый фреймворк CallKit и какие интерфейсные методы в нём есть — это пережить будет тяжело, но в целом можно.

Coming out

Я долгое время испытывал чувство гордости за то, что никто из членов моей команды не уволился, а уж тем более его не уволили за низкую производительность/крупные косяки в работе. Считал неким долгом помочь человеку, даже если это занимало некоторое количество моего свободного времени на выходных. Но необходимо раз и навсегда понять, что всё, что должен делать менеджер любого уровня — это направлять и помогать человеку, а

не делать за него. Проведите несколько личных бесед для того, чтобы обозначить проблему. Чтобы не было конфликта ожиданий и что вы оба понимаете, проблема нужно решать. Постройте стратегию решения проблемы и жестко её контролируйте. Это может не сработать. Чтобы удостовериться, что проблема не в вас, то попросите другого разработчика в команде выступить неким независимым экспертом по оценке производительности/квалификации или просто моральной работы в команде. Если ваши мнения сошлись, то расставайтесь без тени сомнения. Так вы убережете и свои нервы, и нервы членов команды. И даже сам человек будет благодарен спустя какое-то время, что именно так всё закончилось. Чем больше времени проходит и больше опыта я получаю, тем меньше нужно времени, чтобы понять, совершил я ошибку или нет.

Вам необходимо заранее понимать, что бывают и конфликтные ситуации. Но чаще всего это решается через личную беседу. Просто говорите прямо, хотя это бывает крайне сложно.

Построение масштабируемой схемы. Отход от роли «кольца всевластия»

Одна из самых частых негативных историй, которые я слышу от разработчиков про lead'ов — все интересные куски лид забирает себе. От лидов же получаю другой фидбэк, что самая частая проблема — есть задача и её **ВООБЩЕ НИКАК!!!** нельзя делегировать. Отсюда вытекает ряд проблем.

- Время — ресурс ограниченный. Если лид будет делать интересные задачи, а не свою работу, то работа будет попросту не сделана.
- Закладывая большинство компетенций внутри одного человека, вы получаете абсолютно немасштабируемую систему. А что самое интересное, получаете фактор автобуса. Конечно, этим можно оперировать при обсуждении зарплаты, но по факту вы не становитесь лучше как лид, а лишь узурпируете власть.
- Ваша команда не растёт. Никто не понимает установленных процессов, потому что их нет.

Весь ваш ориентир или хайповое слово KPI заключается только в вашей команде. Если есть крутая команда, которая сплоченно, быстро делает продукт и соблюдает процессы — вы хороший лид. Старайтесь постепенно повышать различного рода компетенции в других людях. Крайне важно, чтобы это было осознанным выбором самого человека, а не навязанной сверху практикой.

Варианты будущего роста

Раз у вас уже возник вопрос «а что дальше?» в бытность разработчиком, то он у вас возникнет и на этапе team lead. А здесь всё также можно следовать разобранной ранее схеме. Вопрос лишь в том, стоит ли развиваться как менеджер или всё-таки уйти еще глубже в сторону разработки в роли архитектора. Попробуйте, и вы сможете четко ответить на этот вопрос. Но как я говорил ранее, не задавайте его себе в первые несколько месяцев. Потому что находясь вне привычной зоны комфорта человек по умолчанию склонен негативно

реагировать на любые стимулы. Разберитесь хотя бы в базовых вещах, потом принимайте решение.

Что делать, если стали лидером

Роль team lead, как и роль менеджера, имеет ряд специфических особенностей, зависящих от внешних и внутренних факторов бизнеса, времени, технологий и видения самой компании. Если стали лидером, первым делом:

- **Настройте процессы работы в отделе.** При этом максимально их задокументируйте.
- **Продумайте систему обучения сотрудников.**
- **Настройте правильную систему делегирования** различного рода заданий и лишь направляйте и контролируйте их выполнение.
- **Вступайте в диалог в любой непонятной ситуации.** В споре рождается истина. Лишь так вы поймете людей и получите подтверждение, что они понимают вас.

Что сработало в моем случае может сработать и в вашем. Главная же доктрина заключается в том, что всё приходит с опытом. Если вы работаете в этом направлении, конечно. Проблемы со временем не исчезнут, вы просто научитесь их решать.

Помните, что учиться можно не только на своём опыте, но и на чужих ошибках. Teamleadство — это круто.

Список источников

С. Макконнелл. Сколько стоит программный проект
Дж. Ханк Рейнвотер Как пасти котов
Давид Хейнемейер Ханссон и Джейсон Фрид. Rework

Проголосовать:



+86



Поделиться:



Сохранить:



Комментарии (24)

Похожие публикации

RevealApp — Firebug для iOS приложений

junk • 17 октября 2013 в 13:57

16

iOS 7 и Xamarin

ПЕРЕВОД

junk • 19 сентября 2013 в 15:43

14

C# async для iOS и Android

ПЕРЕВОД

junk • 5 сентября 2013 в 14:05

4

Популярное за сутки

**Яндекс открывает Алису для всех разработчиков.
Платформа Яндекс.Диалоги (бета)**

69

BarakAdama • вчера в 10:52

**Почему следует игнорировать истории
основателей успешных стартапов**

20

ПЕРЕВОД

m1rko • вчера в 10:44

**Как получить телефон (почти) любой красотки в
Москве, или интересная особенность MT_FREE**

24

ИЗ ПЕСОЧНИЦЫ

cab404 • вчера в 20:27

Java и Project Reactor

10

zealot_and_frenzy • вчера в 10:56

**Пользовательские агрегатные и оконные функции
в PostgreSQL и Oracle**

6

erogov • вчера в 12:46

Лучшее на Geektimes

Как фермеры Дикого Запада организовали телефонную сеть на колючей проволоке

NAGru • вчера в 10:10

31

Энтузиаст сделал новую материнскую плату для ThinkPad X200s

alizar • вчера в 15:32

49

Кто-то посылает секс-игрушки с Amazon незнакомцам. Amazon не знает, как их остановить

Pochtoycom • вчера в 13:06

85

Илон Маск продолжает убеждать в необходимости создания колонии людей на Марсе

marks • вчера в 14:19

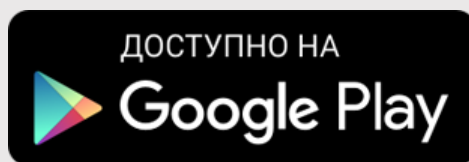
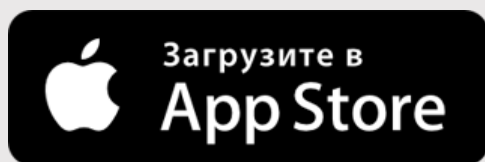
140

Дела шпионские (часть 1)

TashaFridrih • вчера в 13:16

16

Мобильное приложение



Полная версия

