12.08.2023

Как стартап теряет искру

На хорошо организованном стартапе на начальной стадии инженеры часто описывают опыт работы как опьяняющий.

В больших компаниях лучшее, на что можно рассчитывать - это "приятно".

Почему так происходит? Это неизбежно?

Давайте рассмотрим, что делает стартап опьяняющим. Инженер должен проводить большую часть своего времени в этом основном цикле:

1. Если нужно, общайтесь с пользователями, выясняйте их проблемы.
2. Придумайте идею для реализации.
3. При необходимости обсудите идею с коллегами.
4. Воплотите идею в жизнь.
5. Перекрестите пальцы и запустите продукт. Отпразднуйте успех или проведите анализ ошибок. Затем вернитесь к шагу 1.

Когда в команде менее 10 человек, каждый из этих этапов может быть веселым.

1. Вы можете напрямую связаться с пользователями, которые вас интересуют, и выпить с ними пиво.
2. Если вы находите идею, которая, по вашему мнению, полезна для компании и интересна вам, вы можете просто бросить все и начать работать над ней.
3. Почти каждый коллега будет заинтересован в обсуждении этой идеи с вами, потому что:
    1. У них есть личный интерес. Если ваша идея хороша, они получают выгоду, поддерживая вас. Если ваша идея плоха, им выгодно объяснить, в чем проблема. Они даже могут быть заинтересованы как пользователи.
    2. Они могут захотеть работать с вами над этим.
    3. Они, вероятно, могут предложить полезные замечания, так как каждый знаком с большой частью кодовой базы.
4. Реализация проходит быстро.
5. Выбирайте любые инструменты, без проверки безопасности.
6. Маленькая кодовая база. Можно удерживать все в голове, поэтому вы чувствуете себя комфортно, делая глобальные изменения. Быстрая перезагрузка, поэтому можно быстро отлаживать методом проб и ошибок.
7. Быстро вносите изменения. Нет медленных тестов в CI. Нет необходимости в проверке PR для безопасных изменений. Для небезопасных изменений просто попросите коллегу ознакомиться с PR.
8. Если пользователей не так много, можно допустить простои, чтобы быстро вносить изменения в базу данных.
9. Все возможно! Может быть, это сделает вашу долю в компании действительно ценной. Может быть, люди будут развивать эту функцию в течение многих лет, возможно, вы будете считаться видным деятелем в истории. Может быть, вы посетите вечеринку с вашими пользователями, и услышите их слова: "это была ваша идея? О мой Бог, это спасло мою жизнь". Так как у всех в команде есть личный интерес, все болеют за вас. Ваши сердца бьются в унисон.

По мере роста компании, удовольствие от каждого из этих этапов уменьшается. При более чем 100 сотрудниках цикл выглядит примерно так:

1. Общение с пользователями - забота менеджеров по продукту, дурачок! Делай то, что у тебя хорошо получается. В лучшем случае ты получаешь краткое изложение мнений пользователей и адекватный список приоритетов задач на основе этого. В худшем случае у тебя есть запутанный список задач, основанный на неправильном понимании пользователей и эгоистичном видении менеджера, и никто не может объяснить, почему каждая задача важна.

2. Ты не можешь просто работать над тем, что хочешь; это создало бы хаос в коммуникациях O(N^2). У твоего менеджера для тебя есть планы, которые он разработал, учитывая все остальные происходящие события. Возможно, ты сможешь работать над своей идеей после того, как закончишь все остальное. Или можешь работать над этим в свободное время наедине.
3. Ты можешь обсудить свою идею с коллегами, но скорее всего они не заинтересованы. Им это не приносит особых преимуществ; на самом деле они, возможно, конкурируют с тобой за повышение. У них длинный список задач; у них нет свободы бросить все и работать с тобой. И они, вероятно, плохо помнят код, который ты касаешься.
4. Реализация медленная.
5. Все инструменты уже были выбраны. Возможно, есть лучшие новые инструменты, но они не стоят внедрения несоответствия. Ни в коем случае не создавайте несоответствия. Если вы хотите использовать что-то новое, спросите у вашего менеджера и напишите письмо команде по проверке безопасности.
6. Большая кодовая база, в основном по историческим причинам. Много кода, которого ты боишься коснуться, много кода, который уже никто не понимает. Тщательное тестирование - ваша единственная уверенность в том, что вы ничего не ломаете. Вам придется полагаться на многочисленные отладки методом проб и ошибок, но каждый цикл проб и ошибок медленный, потому что код компилируется долго.
7. На небольшой PR уходит 2 недели. Подождите 20 минут, пока CI пройдет, возникнут ошибки, перезапустите тесты. Конфликты слияния накапливаются, пока вы ждете обратной связи от рецензента. Рецензент просто указывает на мелочи, потому что у него недостаточно контекста или стимулов, чтобы делать важные предложения. Вы вносите запрошенные изменения, отправляете изменения, еще 20 минут. Повторяйте 3 раза на каждый PR.
8. Любое изменение инфраструктуры - это процесс из 14 шагов, чтобы не вызвать простои или потерю данных для 10 миллионов пользователей.
9. После 3 месяцев тихой работы вы что-то выпускаете. Вы демонстрируете это на пятничном собрании, ваш единственный шанс, чтобы генеральный директор увидел, над чем вы работали. Только несколько коллег действительно обращают внимание; ваша работа просто не затрагивает большинство из них. Вас похлопают по плечу. Ваш менеджер говорит, что это будет большим прорывом для вас на следующем этапе оценки производительности через 5 месяцев. В лучшем случае вы получаете увеличение зарплаты на 20% и изменение должности. В худшем случае, через 4 месяца происходит реорганизация, вы получаете нового менеджера, ваш проект теряет приоритет или отменяется. Вам нужно убедить нового менеджера, что вы многое сделали в этой компании.

Это можно предотвратить? Я думаю, что нет.

Если вы внимательно посмотрите, все эти проблемы в основном происходят из-за:

1. Уменьшенного личного интереса, что снижает сплоченность команды.
2. Квадратичной коммуникации, что создает необходимость в менеджерах и специализации, что снижает индивидуальные полномочия и объем обучения.
3. Уменьшенной склонности к риску, что замедляет все процессы.

Пункты 1 и 2 являются неизбежным результатом увеличения числа сотрудников. Пункт 3 является неизбежным результатом увеличения числа пользователей, частично из-за государственного регулирования.

Однако существуют способы, которые могут замедлить потерю радости от работы. Прежде всего, не ускоряйте этот процесс. Большинство стартапов необоснованно ускоряют свою корпоратизацию, копируя процессы крупных компаний, обычно путем завлечения менеджеров из крупных компаний, которые приносят с собой свои методики. Например, многие стартапы используют Jira потому, что крупные компании используют Jira. Не используйте Jira. Y Combinator помог миру понять, что вдохновение должно идти в другую сторону — крупные компании должны стараться работать больше как стартапы.

Не только у крупных компаний есть свои сценарии для больших компаний, они также устарели. Когда ваш стартап растет, появятся лучшие решения многих проблем масштабирования. Поэтому, когда вы

сталкиваетесь с проблемами масштабирования, попробуйте решить их, исходя из первоначальных принципов, или вдохновитесь стартапами, которые двигаются быстрее вас, несмотря на одинаковый размер.

Вы можете попробовать структурировать вашу компанию как ряд независимых стартапов. И на самом деле, вы должны делать это настолько, насколько можете; Rippling это в какой-то мере осуществил. Но, как правило, вы не можете зайти очень далеко, потому что компоненты вашего продукта слишком тесно связаны, и большинство компонентов самостоятельно не приносят дохода.

Можно заметно замедлить пункт #1, умно разрабатывая вашу систему стимулирования. Я рассмотрю это в другой статье блога. Но никакое стимулирование не сравнится с большим пакетом акций + возможностью влиять на его стоимость.

Помимо этого, лучший способ — нанимать меньше сотрудников. Я убежден, что большинство компаний слишком быстро нанимают из-за несоответствующих стимулов для менеджеров, неосведомленных инвесторов, которые подталкивают вас быть больше похожими на другие быстро растущие стартапы, и не понимая истинной стоимости работника. И по мере того как инструменты (особенно ИИ) становятся лучше, маленькая команда может обслуживать больше пользователей. Основателям будущего, возможно, не придется так много беспокоиться о проблемах масштабирования, и работа может стать веселее для всех.