работа в команде — пособие по выживанию

крис касперски no-email

почему-то все преимущественно говорят о технике _создания_ команды, чему посвящено множество статей, книг, семинаров и на выбор руководителя предлагается ряд методик, отработанных годами. но вот как _выжить_ в команде, реализовав свой творческий потенциал, обычно предпочитают умалчивать

"*every single employee understand that they are part of a whole. thus, if an employee has a problem, the company has a problem*"
(c) the matrix

введение

Время, когда один-единственный программист мог вести весь проект целиком, безвозвратно прошло. Рынок наводнили акулы бизнеса, ожесточенно конкурирующие между собой, и требование к качеству продукта резко возросло. "Голая" программа без красивого графического интерфейса, многостраничной документации, официального сайта, службы поддержки уже никому не нужна. Исключение составляют утилиты, ориентированные на узкий круг профессионалов которым достаточно развитого функционала, но таких — единицы и реальные деньги крутятся там, где обитают полуграмотные пользователи, популяция которых исчисляется сотнями миллионов. Даже примитивный каталогизатор дисков, будучи достойно оформленным и поданным, может приносить неплохой доход, не говоря уже про автоматизацию предприятий или компьютерные игры. Было бы заманчиво написать такой продукт в одиночку, но увы… это практически невозможно. Даже гениальный программист, талантливый абсолютно во всем, навряд ли захочет отвечать на идиотские письма дебильных пользователей, а писем таких будет очень много и ведь каждый покупатель считает, что вместе с продуктом он покупает еще и внимание. А ну-ка, мать вашу за ногу так, быстро разберитесь почему ваша программа не работает у меня!

Десять лет назад можно было программировать зная всего лишь Turbo Pascal и MS-DOS, теперь же количество обязательных языков/библиотек/технологий исчисляется десятками. Писать интерфейс на Си — это самоубийство, причем крайне болезненное и мучительное, но еще хуже пытаться реализовать вычислительное ядро на DELPHI. Поэтому, неизбежно встает вопрос о создании команды, а, быть может, и целой фирмы.

Работать в команде и сложно, и просто одновременно. С одной стороны, без обмена идей и знаний программист, находящийся в изоляции, никогда не сможет реализовать свой потенциал на 100%. Великие находки и грандиозные открытия, как правило, рождаются в спорах. С другой стороны, если ты один — тебе приходится рассчитывать только на самого себя и никто не придет на помощь в трудную минуту (про то, что некоторые пройдохи ухитряются перекладывать свои проблемы на чужие плечи, мы знаем, но скоромно промолчим).

Это были плюсы. Теперь о минусах. Командная работа требует намного большей самоотдачи, заставляя заниматься не тем, чем тебе интересно, а тем, что нужно команде (как говорится, что полезно для вида, вредно для индивида), при этом неудача одного из участников ударяет рикошетом по всем остальным, и сотни часов, проведенных за монитором, летят впустую. Сплоченные команды — большая редкость (если они вообще существуют в природе) и интересы участников чаще всего оказываются взаимопротиворечивыми. Кто-то рвется вперед, выдвигая одну сумасбродную идею за другой, кто-то топчется на месте, вылизывая каждую строку кода, радуясь, что ему ценой месячного траха удалось сократить длину функции на шесть байт. Как следствие — возникают расколы и конфликты. Атмосфера внутри коллектива становится душной, а работа — непроизводительной.

Хуже всего приходится творчески активным индивидуалистам — необщительным, неуживчивым, неконтактным, привыкшим работать удаленно, каким, собственно, и является мыщъх, имеющий богатый опыт командной работы (хоть большей частью и отрицательный), но ведь… тут вот какой момент. Тот, кто умеет что-то делать неосознанно, кто наделен этим талантом от рождения — никогда не сможет объяснит как именно и что именно он делает, почему поступает так, а не эдак и т.д. Мыщъх же представляет собой довольно аутичный тип от

природы и общению учился сам, действуя методом проб и ошибках, о которых и хочет рассказать, адресую статью таким же как он сам.

вхождение в команду

Примкнуть к известной команде, стать ее частью — мечта каждого второго, а, быть может быть, и двух третий всех программистов. Точной статистики у мыщъх'а нет, но и без нее понятно, что сильная команда — это хорошо, однако, тут все не так гладко, как кажется.

Начнем с того, что "не бойся показаться дураком, бойся показаться очень умным, потому что тогда немедленно возникнет вопрос: если ты такой умный, то почему ты нанимаешься на работу, а не нанимаешь на нее?" (с) Виктор Пелевин "Поколение П". Сильные команды в своей массе довольно скептически (если не сказать брезгливо) относятся к новичкам, особенно если те ведут себя самоуверенно, как будто любая работа им по плечу (это, кстати сказать, первый признак непрофессионализма, чем больше знает человек, тем сильнее он начинает комплексовать по поводу своей тупости, потому что осознает разрыв, отделяющий его от истинных знаний, к тому же, во всякой предметной области есть куча тонких нюансов и ее нельзя освоить с криком "ура", бросая свое тело на амбразуру технической документации и литературы).

Лучше занизить планку своих знаний, чем завысить ее, поскольку, в первом случае остается "запас прочности", а во втором — легко попасть в тупик не сумев ответить на поставленный вопрос. Опять-таки следует различать понятия "умею" и "видел". Например, мыщъх краем глаза видел Perl, но _опыта_ работы с ним не имеет, поэтому зачастую идет не тем путем который нужно, а тем который знает. Тоже самое можно сказать и о DELPHI, Си++ и многих других языках.

Любой нормальный программист имеет представление как минимум о паре десятков языков, но активно программирует на двух-трех, на которые и следует делать упор, а остальные можно даже не упоминать, особенно, если вы работали с ними давно и уже успели подзабыть. Однажды с мыщъх'ем случился такой конфуз: желая блеснуть своим хвостом, он настрочил длинный список языков, с которыми когда-либо имел дело. Естественно, когда его начали проверять, выяснилось, что даже простейшую программу для вычисления факториала ни на Форте, ни на Лиспе мыщъх (без обращения к документации) составить не в состоянии. Доверие сразу же было подорвано, а сама компетентность поставлена под сомнение.

В идеале, следует искать только те команды, которые реально нуждаются в вас, в ваших знаниях и умениях, а не в чем-нибудь другом, но в жизни очень часто случается так, что вам рекомендуют подучиться, а то и переучиться полностью! "ОК, вы классно программируете на Си, но мы ведем несколько проектов на Java, поэтому вы должны быть знакомы и с этим языком". Соглашаться или нет? Спорный вопрос. С одной стороны, если отвечать "да пошли вы все!", в поисках можно провести всю оставшуюся жизнь, к концу которой Си рискует стать неактуальным, так что переучиваться все равно придется. Не сейчас, так потом. Но! Нормальный лидер всякой преуспевающей команды всегда стремится использовать навыки и наклонности каждого из программистов по максимуму, то есть по назначению. Собственно именно это и обеспечивает команде процветание. Если же брать кого попало, распределяя задания хаотичным образом по принципу: не умеет — научится, то это не команда будет, а просто сброд, в компании которого хорошо пить пиво и обсуждать округлости женщин, но отнюдь не выпускать конкурентоспособные продукты.

Подытоживая сказанное, сформулируем правило номер один: **всегда оставайтесь самим собой**. Если вы влюблены в ассемблер — программируйте на ассемблере, если вы небрежны, небриты, неаккуратны — вот таким и приходите на собеседование. Даже если у вас рваный график, плавно перетекающий из совы в жаворонка, а потом обратно — не берите на себя обязательств приходить на работу ровно в восемь или быть в это время как штык на IRC-канале. Поверьте, существует множество команд, работающих по такому же принципу. Внешний вид — не помеха. Главное — результат.

Новички панически бояться ответить "нет" на любой поставленный им вопрос, опасаясь, что за этим последует категорический отказ (и хорошо если не пинок ногой), вот и "подписываются" на любые условия, которые им только предложат. На самом деле, в слове "нет" ничего ужасного нет. Употребляйте его почаще. Оно поможет вам продать свои умения и навыки как можно дороже.

в команде

Поздравляю! Вы в яйце. С этого момента ваши личные интересы задвинуты на задний план и все свободное (и несвободное) время посвящено команде. Некоторые ведут сразу несколько проектов в различных командах, но, во-первых, это страшно выматывает, а, во-вторых, отсутствие концентрации на одной-единственной цели часто приводит к провалу.

В команде намного сложнее реализовать свой творческий потенциал, свои идеи (особенно рискованные и непроверенные), поскольку, в случае провала страдают все участники команды (а у многих жена, дети…). Конечно, без дерзости, без творческих порывов не бывает и прорывов, но позволить себе втягиваться в работу над рискованным проектом может либо только что сформировавшаяся команда, либо команда уже имеющая за плечами несколько успешных проектов и солидный капитал, но даже в этом случае, "раскрутить" остальных членов, заинтересовать их своим проектом будет нелегко, поскольку у каждого из них имеются свои собственные идеи и чем команда больше, тем труднее выделиться на фоне остальных.

С другой стороны, в мелких командах никакая исследовательская деятельность невозможна в принципе. Главное здесь — получить рабочий продукт в отведенные строки, не отвлекаясь на посторонние вопросы. Вы хотите экспериментов, вы чувствуете, что можете значительно увеличить производительность и функционал кода, если сменить парадигму программирования, библиотеку, компилятор и т. д. Вам интересно посидеть с Кнутом, осваивания новые алгоритмы, чтобы в конечном счете выбрать самый эффективный из них, но… команду это не интересует. Код работает — и это главное! Навряд ли оптимизация увеличит объемы продаж. Лучше выпустить еще один минимально работающий проект. А потом еще один и еще… Постепенно это гонка превращается в настоящий марафон, требующий только тех знаний, которые уже есть и никак не способствующий творческому развитию.

Большие команды могут позволить талантливому программисту совершенствовать свое мастерство и годами вести исследовательскую деятельность, которая, возможно, не принесет никакой отдачи. А, возможно, и принесет. В девятнадцатом веке лишь немногие люди верили в перспективу электричества, остальные же считали, что оно пригодно лишь для игрушек и фокусов. Графический интерфейс тоже поначалу не вызвал большого энтузиазма, а сейчас это основной способ общения работы с компьютером. Никто и никогда не сможет заранее сказать какая технология окажется перспективной, а какая нет, однако, это не значит, что любая бредовая идея будет щедро финансироваться.

В принципе, обладая достаточной напористостью (и авторитетом) можно "раскрутить" лидера команды на _любой_ проект, убедив его, что все будет ОК. Это самая легкая часть, а вот потом… приходится сложнее. Самые замечательные и элегантные (на бумаге) конструкции зачастую рассыпаются в прах при столкновению с реальностью. На поздней стадии нередко вскрываются непредвиденные трудности, которые без дополнительных (и весьма крупномасштабных) исследований никак не удается обойти. Проект затягивается, требуя все новых вложений, но бросить его, похоронив тысячи строк кода — жалко. Команда высаживается и нервничает, теряя время и деньги, а вы — авторитет.

Правило номер два: **продвигая свои идеи, никогда не выдавайте желаемое за действительное и не убеждайте других в том, в чем сами до конца не уверены**. Если проект провалится это не принес пользы ни вам, ни команде. Конечно, от ошибок и провалов никто не застрахован, но если решение о новом проекте принималось коллегиально, то в случае провала ответственность равномерно распределяется между всеми участниками. Если же вы агрессивно лоббировали свою идею, вам это обязательно припомнят.

соображения о стиле программирования

Стиль программирования — весьма щекотливый вопрос. В одних командах имеется свод правил оформления листинга (учитывающий буквально все: от расстановки фигурных скобок до системы наименования переменных), в других — ничего подобного нет. Пиши как хочешь и на чем хочешь! У каждой практики есть свои плюсы и минусы. Даже плохой стандарт кодирования, поддерживаемый всеми участниками, лучше чем совсем никакого. Команда на то и команда, чтобы работать _совместно_ попеременно заглядывая то в свои, то в чужие листинги. И, если каждый начнет следовать своим привычкам, образуется форменный балаган, в котором через некоторое время будет совершенно невозможно разобраться.

А вот и другая сторона медали (очень часто упускаемая маркетоидами из виду). Привычка — это вторая натура и от нее никуда не уйти. Кодирование в "неестественном" стиле _значительно_ снижает производительность труда и одновременно с этим затрудняет поиск ошибок "глазами". К тому же, если программист использует фрагменты ранее написанных

листингов, их придется реконструировать в соответствии с правилами кодирования, а это не только впустую потраченное время, но еще и вероятность внесения ошибок в уже отлаженный код. А что если один и тот же исходный файл используется в нескольких независимых проектах, каждый из которых велит придерживаться своих правил кодирования?! Как развивать и сопровождать такой файл? Это же сдохнуть можно если вручную синхронизовать несколько проектов, а подходящих средств оптимизации не было и нет!

Правило номер три — если вас не устраивает стандарт кодирования, принятый в команде, пишите в своем стиле, пытаясь убедить остальных в том, что от этого никому хуже не станет. Как показывает мыщъх'ный опыт, в большинстве случаев это срабатывает. Стиль программирования — это вообще-то, так, ерунда. Бюрократическая волокита. Гораздо важнее сразу определиться с используемыми языками и компиляторами. Допустим, 9 из 10 членов команды пишут на Си, а один — на DELPHI. Состыковать откомпилированные модули не проблема, но вот для эффективной работы каждый участник проекта должен понимать любого другого. Мы не можем просто сказать — возьми DELPHI и напиши крутой интерфейс для нашей программы, поскольку сразу же возникает вопрос: а, что, собственно, писать? Какими функциями обладает программа на данный момент и какими она (возможно) будет обладать? К тому же, интерфейс полностью отделен от вычислительной части лишь в книжках по программированию, а в реальной жизни все компоненты программы связаны друг с другом и если DELPHI-программист абсолютно не разбирается в Си, остальные члены команды вынуждены постоянно отвлекаться от своей работы, объясняя ему что и как. И наоборот. Си-программисты обязаны знать DELPHI хотя бы на минимальном уровне, чтобы самостоятельно собрать проект без посторонней помощи.

Крайне нежелательно использовать в проекте языки которые кроме вас никто не знает (например, Haskell или Ruby), поскольку это ставит в зависимость остальных членов проекта. Вы становитесь незаменимым носителем знания. В команде, состоящей из нескольких человек, с этим еще можно как-то смириться, но вот крупные коллективы вас просто попрут. И правильно сделают! А вдруг вы решите уйти или забросите проект, занявшись другими неотложными делами? Кто разберется в ваших исходных текстах, если потребуется исправить баг или добавить новую фичу? Какой смысл ставить под удар труд десятков людей только потому, что на таком-то языке задача решаться чуть-чуть более эффективно, чем на общепринятых?

Но даже в рамках одного языка (например, того же Си) необходимо согласовать используемые компиляторы, поскольку "смешанное" программирование еще никого не доводило до добра. Достаточно привести один пример: в Borland C++ тип char по умолчанию unsigned, а у Microsoft Visual C++ signed, поэтому, программа, разработанная на Borland C++, может разваливаться при компиляции под Microsoft Visual C++ и наоборот. Но с этим еще можно хоть как-то бороться. Скажем, взять за правило компилировать программу несколькими компиляторами и давить багов еще в зародыше. При переносе на другие платформы это очень помогает! Но никогда не надо собирать программу из объектных файлов, откомпилированных разными компиляторами, поскольку каждый из них завязан на свой RTL, а RTL должен быть только один! В крайнем случае, транслируйте объектные файлы, созданные различными компиляторами, в DLL модули. Это снижает производительность (и подчас довольно значительно), но исключает конфликт RTL.

Другой больной вопрос — это хаки, то есть нестандартные приемы программирования, зачастую завязанные на малоизвестные или недокументированные возможности операционной системы, языка программирования или компилятора. Их следует избегать любой ценой! И это не обсуждается!

размер имеет значение! (или про то как программисты меряются писками и что из этого обычно выходит)

Как говорится: "тормозит обычно тот, кто за клавиатурой, да и то лишь по мнению тех, кто не за ней". Критиковать чужие ошибки и просчеты всегда легко, но… зачем?! Только чтобы почувствовать превосходство над остальными? Все мол, кретины и дураки, один я умный, весь проект держится на мне, чтобы вы, идиоты, без меня делали и т. д. Да, в команде могут (и должны!) встречаться программисты различного уровня квалификации. Тезис о том, что все члены должны стоять на одной ступеньке в корне неверен и вот почему: программное обеспечение крайне неоднородно по своей структуре и состоит из модулей различной

сложности. С одними справится даже девушка, закончившая двухнедельные курсы, другие же потребуют консолидации усилий нескольких матерых спецов. Какой смысл поручать спецам писать простой код? Не лучше ли использовать их навыки там, где они действительно необходимы? И потом, специалистами не рождаются. Старшие передают опыт младшим и так испокон веков.

Профессионалы, кстати говоря, встречаются самые разные. Одни смотрят на салаг свысока, другие же общаются с ними как с равными, зачастую вступая в разборки по поводу и без. Но подробных объяснений ждать в любом случае не стоит. Вы не в институте и за ваше обучение денег никто не получает. А объяснять очередному пионеру в сотый раз одни и те же истины — скучно, неинтересно и непродуктивно.

Правило номер четыре: профессию не получают, ее воруют. Присматривайтесь к остальным членам команды и берите на заметку все полезные приемы программирования, обращая внимание даже на незначительные нюансы. Не бойтесь открыто обсуждать свои идеи с коллегами. Даже если вас раскритикуют и поднимут на смех, по крайней мере, укажут на ошибки, а это дорого стоит! Так же, не пытайтесь завоевать дешевый авторитет, руководствуясь принципом: кто ничего не делает — тот никогда не ошибается. Пускай авторитет позаботиться о себе сам. Ваша основная задача — совершить максимум ошибок, наступив на все грабли, которые только находятся в пределах досягаемости. Шишки — заживут, ошибки — забудутся, а знания — останутся и будут работать на авторитет. На настоящий авторитет, который нельзя ни пропить, ни скомпрометировать, ни потерять, потому что он есть.

Гораздо лучше заслужить репутацию человека, не боящегося признаться в собственном незнании, чем выглядеть снобом. Впрочем, в руководители высшего звена пробиваются именно снобы и губят всех тех, кто находится под ними. Сравните какой была Microsoft при Билле Гейтсе и какой она стала при Стиве Балмере.

заключение

Прежде чем закончить свое сумбурное "пособие по выживанию", мыщъх хотел бы напомнить, что в таких вопросах никакой истины не существует и "сколько людей — столько же и мнений". Мыщъх высказал свое. Это не означает, что все остальные мнения порочны и неверны. Вовсе нет! Главное ведь результат, а не пути его достижения. Чужой опыт редко бывает полезен остальным, поскольку зависит от психотипа личности и стоящих перед ней проблем, а проблемы можно решать по-разному. Они предпочитают наклонять голову, другие — идут напролом. Третьи же комбинируют два первых способа, помня о мудрой восточной поговорке: "не будь слишком мягким — сомнут, не будет слишком твердым — сломают".