РУКОВОДСТВО ДЖОЭЛА СПОЛЬСКИ

по подбору программистов и управлению ими

SMART AND GETS THINGS DONE

Joel Spolsky's Concise Guide to Finding the best Technical Talent

Apress®

РУКОВОДСТВО ДЖОЭЛА СПОЛЬСКИ

по подбору программистов и управлению ими

Джоэл Спольски

Издательский дом "Вильямс"
Москва • Санкт-Петербург • Киев
2008

ББК 65.245
С73
УДК 331.108.54

Издательский дом "Вильямс"
Главный редактор *С.Н. Тригуб*
Зав. редакцией *А.В. Назаренко*
Перевод с английского *И.В. Константинова*
Под редакцией *А.В. Назаренко*

По общим вопросам обращайтесь
в Издательский дом "Вильямс" по адресу:
info@williamspublishing.com, http://www.williamspublishing.com
115419, Москва, а/я 783; 03150, Киев, а/я 152

Спольски, Джоэл.

С73 Руководство Джоэла Спольски по подбору программистов и управлению ими. : Пер. с англ. — М. : ООО "И.Д. Вильямс", 2008. — 144 с. : ил. — Парал. тит. англ.

ISBN 978-5-8459-1354-8 (рус.)

Известный эксперт по вопросам организации работы компаний по разработке программного обеспечения Джоэл Спольски легко и понятно объясняет, как найти, отобрать, увлечь и нанять на работу лучших технических специалистов. Кроме того, много внимания уделяется созданию хороших условий труда для людей, которые этого действительно заслуживают. Книга будет полезна всем, кто нанимает на работу программистов или руководит ими в организации.

ББК 65.245

ISBN 978-5-8459-1354-8 (рус.)
ISBN 978-1-59059-838-2 (англ.)

Оглавление

Содержание

Посвящается Джареду

Об авторе

Джоэл Спольски — это всемирно признанный эксперт по разработке программного обеспечения. Его Web-сайт Joel on Software (`www.joelonsoftware.com`; на русском языке — `russian.joelonsoftware.com`) популярен среди разработчиков программ со всего мира и переведен более чем на 30 языков. Основав в Нью-Йорке компанию *Fog Creek Software*, он создал FogBugs — популярную систему управления проектами, предназначенную для команд программистов. Джоэл работал в компании *Microsoft*, где, входя в команду разработчиков Excel, он проектировал VBA, а работая в компании *Juno Online Services*, проектировал известный Интернет-клиент, используемый миллионами. Он написал две книги — *User Interface Design for Programmers* (Apress, 2001) и *Joel on Software* (Apress, 2004)[1], а также был редактором книги *The Best Software Writing I* (Apress, 2005)[2]. Степень бакалавра наук по программированию Джоэл получил в Йельском университете. Он служил парашютистом в Армии Израиля и был одним из основателей киббуца Ханатон.

[1] *Джоэл о программировании*, Символ-Плюс. — 2006.

[2] *Лучшие примеры разработки ПО*, Питер. — 2006.

Введение

Представьте себе маршрут с десятью препятствиями. В самом начале была целая группа тех, кому предстояло по нему бежать. Впереди их ждет двухэтажная стена, через которую надо перелезть, а также что-то вроде подвесного моста и равнины со змеями.

Предположим для простоты, что каждое препятствие останавливает 50% бегущих.

Итак, если бег начали 12 человек, то шесть из них после самого первого препятствия поймут, что проиграли. Можно будет увидеть, как они свалились в кучу у самого подножия двухэтажной стены.

Что касается оставшихся шести, то они будут двигаться по подвесному мосту, с которого трое из них уморительно вывалятся через веревочные перила. Через какое-то время эти вывалившиеся обнаружат, что висят на одной ноге, а из их карманов сыпется всякая ерунда: ключи, кошельки, монеты, а также резиновые утята, казу[1]. Ну, вы знаете, что еще может сыпаться, — ерунда, одним словом.

В действительности никто из них не преодолеет все десять препятствий.

На самом деле, чтобы это удалось хотя бы одному человеку, на старт необходимо вывести 2^{10}, т.е. 1024 бегуна.

Процесс найма прекрасного технического таланта — это отборочное соревнование по бегу. Многие слыхом не слыхивали о вашей компании. А многие пребывают в неведении, что вы сейчас заняты наймом. Другие вообще живут не в вашем городе. Есть и такие, у кого визы не в порядке. Другие же присылают свои резюме, но Джон, которые их все читает, обычно выбрасывает резюме от выпускников тех школ, чьи команды игроков в

[1] Крохотные музыкальные инструменты. — *Примеч. пер.*

американский футбол побеждали команду из округа Колумбия, в которой играл Джон. Тем не менее другие все же приходят на собеседование, но уходят ни с чем. Среди прочих бывают те, кто выше всяких похвал, но им уже предложили другую работу. Ну а те, кому другую работу не предложили, бывают так огорчены серой облупившейся краской на стенах и ужасным флуоресцентным освещением, что остаются на своей прежней работе. Небольшое количество людей, не возражающих против работы в каморках, решили, что на самом деле они не хотят зарабатывать на жизнь, программируя бомбы для уничтожения бункеров. Не то чтобы эти люди были против таких бомб, просто у них другое призвание. Препятствия, одним словом...

Такова вызывающая депрессию реальность найма, похожая на бег с препятствиями (и это действительно реальность). Следовательно, чтобы нанять троих программистов, надо заняться тремя тысячами кандидатов?

Во всей этой математической путанице есть рациональное зерно или, если хотите, светлая сторона. Она заключается в том, что если убрать *одно* препятствие — только одно! — то численность нанятых людей можно *удвоить*. Уберите два препятствия, и вы вчетверо увеличите количество нанятых, прошедших через остальные испытания. Ну и так далее, и тому подобное.

Однако это решение — не палочка-выручалочка. Нет такого волшебного предмета, с помощью которого вы решите все свои проблемы по найму и получите прекрасных специалистов, готовых работать на вас хоть завтра. Что вам нужно сделать, так это взглянуть на весь процесс как на маршрут с препятствиями и начать думать, как убрать их, чем побольше. Работайте над ними всеми, так как они одинаково важны.

Имея должность, называемую "технический рекрутер" или что-то вроде "менеджер по работе с персоналом", вы, возможно, начнете сталкиваться с небольшой проблемой. Действительно, если ваша должность не "Лорд-Суперверховный-Правитель, Командор-Империи и Матка-всех-пчел", то, вероятно, вы столкнетесь с тем, что иногда препятствия, связанные с наймом, не будут вашей виной, и повлиять на них вы не можете. Если люди не хотят работать на вашу компанию из-за того, что не желают сидеть в разделенной на каморки шумной,

темной комнате без окон, с мерцающим освещением, старыми протертыми коврами и явственным запахом плесени, то это проблема уже не найма, а офисного оборудования, не так ли? Да, я знаю, что вашей *вины* здесь нет, однако ваша проблема здесь *есть*. Я буду говорить о многом, что вам нужно делать для найма прекрасных разработчиков, но, тем не менее, это лежит за пределами возможностей нормального рекрутера. Да и со значительной частью этого "многого" не справится даже сам генеральный директор.

Впрочем, еще не все потеряно. Вам надо хотя бы иметь представление об этих проблемах, чтобы знать, чего добиваться. И если вам трудно нанимать специалистов из-за ужасного состояния офисной площади, то (хотя традиционно это не дело рекрутера) вам надо будет насильно пробиться на следующее заседание по планированию офиса. Если ваша компания расположена в месте, малопривлекательном для незаурядных выпускников колледжей, тогда об этом следует поговорить с вашим генеральным директором, как только он поинтересуется, почему еще не заполнены предназначенные для них вакансии. Обычно же вам придется работать в тех частях маршрута с препятствиями, которые вы *можете* контролировать, и в таких случаях вам гарантированы положительные результаты.

История, не относящаяся к этой книге

Давным-давно (а если точно, то в 2000 году) я написал для своего Web-сайта путанную статью, озаглавленную "Ты что, не можешь найти программистов?"[2].

В те времена первый бум "дот-комов" был в самом разгаре и спрос на Web-программистов был таким высоким, что многие большие компании вынуждены были обращаться к фирмам, за-

[2] Обычно я указываю URL-адрес статьи, по которому ее можно просмотреть. Но эта старая статья, написанная мною еще в юности, просто нелепая, поэтому, если она еще где-то есть во Всемирной паутине, ищите ее сами, но этого я делать не советую. В оригинале она называется *Whaddaya mean, you can't find programmers*?

нимающимся IT-консалтингом, и платить по 200–300 долл. в час за не слишком хороших HTML-кодеров. Возникали целые армии Интернет-консультантов, в которые брали 22-летних выпускников колледжей, — лишь бы эти выпускники знали, как пользоваться FrontPage. Их потенциальный почасовой заработок составлял примерно 40 долл., но по выставленным за их работу счетам надо было платить уже 250 долл. в час.

Смысл статьи был в том, что достаточно хорошо платить, предоставить для работы хорошо освещенные офисы, организовать бесплатный массаж, и тогда проблем с наймом не будет.

Однако вскоре после того, как статья была написана, первый "Интернет-пузырь" лопнул, и в технической индустрии наступило что-то вроде ядерной зимы. На улицу была бесцеремонно выброшена масса программистов, разработчиков и Web-дизайнеров. Многие из них даже не понимали, что на самом деле несправедливо, когда начальную зарплату в 60 тыс. долл. получали те выпускники колледжа, чьей основной специальностью был английский язык, а единственным техническим навыком — умение создавать в Geocites личную страницу. Программисты были рады хоть какой-то работе, даже (*зажмите нос*) на эту отвратительную *Microsoft* (*прямо мороз по коже*). Следующие три–пять лет я вообще стеснялся писать подобные бестактности, когда столько талантливых (и бесталанных тоже) программистов вернулись жить к своим родителям и устроились работать в магазинах канцтоваров.

Когда я недавно перечитывал эту статью, то понял, что написал ее о том, в чем совсем не разбирался. Имея несколько идей о том, как нанимать программистов, я совсем не обладал опытом работы в этой области.

Последние шесть лет мы с моим другом Майклом создаем компанию по разработке ПО. С самого начала, когда мы даже не знали, какими именно будут эти программы, нас не покидает *полная* уверенность, что приоритет *номер один* — это найм прекрасных людей. Все эти годы, даже до того как появилась хоть какая-то возможность кого-либо нанять, мы постоянно следили за тем, чтобы у нас хотели работать прекрасные люди. Мы допустили некоторые ошибки, а также многому научились, и теперь я чувствую, что привлечение прекрасных людей

для работы в *Fog Creek* — это практически единственная проблема, с которой мы *не сталкиваемся*. Теперь весь процесс привлечения замечательного таланта мне знаком намного больше, чем во время написания своей первой статьи, при виде которой меня передергивает.

Да и за последний год-другой кое-что изменилось — закончилось тяжелое время, снова пошли инвестиции в технологию, здесь и там появляются новые фирмочки, а многие относительно признанные технологические компании опять бросились нанимать сотрудников, как сумасшедшие. Почти в каждой компании, нанимающей разработчиков программ, имеется длинный список вакансий. А компании, где этого списка нет, являются по большей части проворными, созданными с нуля, фирмами, которые стараются прожить за счет денег, реально заработанных на продаже своей продукции заказчикам; это намного более здоровая ситуация, чем в 2000 году. В целом я замечаю, что наличие вакансий характерно для компаний с большим количеством денег и заказчиков, а эти компании заняты наймом просто потому, что снова хотят инвестировать в последующие прибыли и в дальнейший рост. А чего я *не вижу*, так это доминирования на рынке труда множества фирм, созданных сумасшедшими, которые, вполне естественно полагая, что в новой компании обязательно должно быть примерно 170 сотрудников, первое, что делают, — нанимают 170 человек.

Как и прежде, менеджеры, посредники по найму и рекрутинговые директора, которые долгое время были полностью убеждены, что самое важное — это нанять прекрасных разработчиков программ, начинают удивляться, почему так тяжело заполнить все эти имеющиеся у них вакансии. Видя такую ситуацию, я и написал эту книгу, надеясь, что она станет в некотором роде искуплением за мою первую неловкую попытку.

Еще я надеюсь, что книга поможет сделать вашу организацию прекрасным местом работы и, помогая вам, хоть немного увеличит количество счастья в мире.


50
LBS

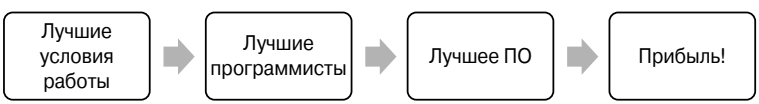
Глава 1

Достигая тех высот

В марте 2000 года я запустил Web-сайт Joel on Software[1], сделав довольно громкое заявление: большинство людей не правы, думая, что для создания успешной компании по написанию программ нужна идея. Вот мои слова.

Довольно распространенно убеждение, что вы создаете компанию по написанию программ, ставя цель найти разумную идею, которая справляется с некоторой еще нерешенной проблемой, затем реализовать эту идею и сколотить состояние. Мы называем это "верой в создание еще лучшей мышеловки". Но настоящей целью компании по написанию программ должен быть перевод капитала в работающие программы[2].

Последние пять лет я испытывал эту теорию на практике. Вот формула, по которой работает компания *Fog Creek Software*, основанная мною вместе с Майклом Прайором в сентябре 2000 года. Вкратце эту формулу можно показать в виде четырех этапов.



Это очень удобная формула, особенно потому, что настоящей целью основания *Fog Creek* была компании по созданию программ, где *нам хотелось бы работать*. Еще тогда я объявил, что хорошие условия труда (или более коряво, "создание компании, где хотелось бы работать лучшим в мире разработчикам программ") должны привести к прибылям, причем так

---

[1] `www.joelonsoftware.com`.

[2] Joel Spolsky, "Converting Capital Into Software That Works", опубликованная на сайте `www.joelonsoftware.com`, 21 марта 2000 года (поиск по словам "Converting Capital").

же естественно, как шоколад — к полноте и мультяшный секс в видеоиграх — к разгулу бандитизма.

Впрочем, сейчас мне хотелось бы ответить всего на один вопрос, ведь если эта часть теории неверная, тогда и вся теория разваливается. Этот вопрос состоит в следующем: есть ли какой-то смысл говорить, что надо заполучить "самых лучших программистов"? Не слишком ли много разновидностей программистов, чтобы предыдущий вопрос имел какой-то смысл?

Может, для нас это утверждение и очевидно, но для многих оно все равно нуждается в доказательстве.

Несколько лет назад сравнительно крупная компания собиралась купить *Fog Creek*, но я знал, что это невозможно, так как генеральный директор этой компании говорил, что в действительности он не согласен с моей теорией о найме самых лучших программистов. Аргументируя свою позицию, он использовал библейскую метафору: вам нужны только царь Давид вместе с армией солдат, которые просто должны выполнять его приказы. Впоследствии акции его компании заметно упали в цене — с 20 долл. до 5 долл. за штуку; поэтому хорошо, что мы не приняли предложение.

На самом же деле, если взять мир бизнес-журналистов (мастеров списывать друг у друга) и больших компаний (тех, в которых полагаются на менеджеров-консультантов со сверхоплатой, нанятых думать за компанию, жевать за нее и т.д.), то для здравого смысла этого мира, как мне кажется, самым важным является уменьшение *стоимости* программистов.

В некоторых других отраслях дешевое *важнее* хорошего. *Wal-Mart* выросла в самую большую корпорацию на земле, продавая не хорошие, а дешевые продукты. Если бы эта компания попыталась продавать высококачественные товары, то их цена была бы более высокой, и преимущество, достигаемое благодаря дешевизне, было бы потеряно. Например, если бы эта компания пыталась продавать носки, которые могут выдерживать, например, очень интенсивную стирку в стиральной машине, то в них пришлось бы использовать всевозможные дорогие компоненты, например хлопок, и тогда стоимость каждого носка должна бы была возрасти.

Итак, почему в отрасли по созданию программного обеспечения нет места поставщикам дешевизны — тем, кто старается использовать самых дешевых программистов? (Напомните, чтобы я не забыл спросить у *Quark*, как работает план "Выжми из каждого все, затем найди ему дешевую замену".)

Ответ кроется в том, что дублирование программ обходится бесплатно. Это значит, что цена программистов распределяется на все копии продаваемой вами программы. А если говорить о качестве программ, то его можно поднять, не наращивая цену продаваемой единицы продукции.

В сущности, разработка скорее увеличивает ценность, а не цену.

Или, попросту говоря, проявив скупость к программистам, вы создадите дорогие программы, которые даже не окупят своего производства.

То же самое применимо и к индустрии развлечений. Вполне оправданно нанять (даже за большие деньги) для вашего последнего блокбастера Брэда Питта, поскольку затраченные средства можно будет разделить между всеми теми миллионами зрителей, которые придут смотреть фильм только потому, что Брэд такой *чертовски крутой*.

Или, с другой стороны, для вашего последнего блокбастера вполне оправданно нанять (даже за большие деньги) Анджелину Джоли, так как и в этом случае затраченные средства можно будет разделить между всеми теми миллионами зрителей, которые придут смотреть фильм только потому, что Джоли такая *чертовски крутая*.

Но я пока еще ничего не доказал. Что значит быть "самым лучшим программистом" и действительно ли программы, созданные разными программистами, так сильно отличаются по качеству?

Давайте начнем со старой доброй производительности. Измерить производительность программиста довольно трудно. Почти любая мера, какую можно применить (количество строк отлаженного кода, аргументов командной строки, количество точек входа функций), является слишком примитивной, да и очень трудно получить конкретные данные по большим проектам, ведь крайне редко двум программистам поручают сделать одно и то же.

Что касается меня, то я пользуюсь данными профессора Стэнли Айзенштата из Йельского университета. Каждый год он ведет интенсивный курс программирования (CS 323), по большей части состоящий из пяти заданий по программированию или что-то вроде этого. На выполнение каждого из них дается примерно две недели. Для вузовского курса обучения это очень серьезные задания: реализовать для системы Unix оболочку командной строки, реализовать программу ZLW-сжатия файлов и т.д.

Студенты были так недовольны слишком большой работой, требуемой на освоение курса, что профессор Айзенштат стал просить их сообщать, сколько времени они потратили на каждое задание. Эти данные он тщательно собирал в течение нескольких лет.

Я потратил некоторое время, обрабатывая его цифры; пожалуй, это единственный известный мне набор данных, который позволяет сравнить десятки студентов, одновременно работавших над одинаковыми заданиями с помощью одной и той же технологии.

Первое, что я сделал с этими данными, — подсчитал для часов, потраченных на каждое из заданий, среднее, минимум, максимум и стандартное отклонение. Результаты моих расчетов приведены ниже.

Сильнее всего здесь бросается в глаза громадный разброс значений. Самые быстрые студенты справлялись с работой в три-четыре раза быстрее, чем средние, и в десять раз быстрее, чем самые медленные студенты. Стандартное отклонение оказалось крайне большим. Поэтому я подумал, что, скорее всего, для некоторых студентов задание оказалось ужасно сложным. Мне не хотелось учитывать тех студентов, которые, потратив на задание четыре часа, не смогли создать работающей программы. Поэтому я рассматривал данные только по студентам, находящимся в верхнем квартиле успеваемости, ...т.е. среди лучших 25% по качеству кода. Должен сказать, что оценки на курсе профессора Айзенштата *полностью* объективны: они вычисляются по формулам, где используется только количество пройденных автоматических тестов кода. За плохой стиль никакие баллы не вычитались.