© Чугреев В.Л.

ОРГАНИЗАЦИОННЫЕ АСПЕКТЫ УПРАВЛЕНИЯ ЗНАНИЯМИ В IT-КОМПАНИИ

В статье рассматриваются вопросы управления знаниями в IT-компаниях, занимающихся разработкой программного обеспечения. Обсуждаются производственные особенности таких компаний, а также их потребности в знаниях. Предлагаются организационно-методические формы обмена знаниями, которые могут быть использованы в данных компаниях.

IT-компания, софтверная компания, программные разработки, разработка программного обеспечения, управление знаниями, системы управления знаниями, просмотр кода, парное программирование.

Вряд ли кто-то будет спорить с тем фактом, что ценность и потенциал инновационной, высокотехнологичной компании определяется прежде всего тем кадровым ресурсом, которым она располагает. В IT-компании, занимающейся разработкой программного обеспечения (т. н. софтверной компании), именно кадровый состав, а точнее компетенции и знания сотрудников, определяют её жизнеспособность и рыночную эффективность. О таких компаниях и пойдёт речь далее.

Если говорить о технологиях, то это главным образом программные технологии. Разработка новых систем, а также сопровождение существующих опирается на применяемые компанией программные решения. К ним можно отнести: используемые языки и среды программирования, ранее написанный код, принципы и методы написания кода, требования к нему, архитектурные решения.

ЧУГРЕЕВ Валерий Леонидович
кандидат технических наук,
научный сотрудник ИСЭРТ РАН
chugreev10@mail.ru

Таким образом, центральным, стержневым элементом IT-компании является программный код. От того, каким он будет и будет ли вообще, зависит успех компании на рынке. Безусловно, важную роль также играет и маркетинговая составляющая, но сейчас мы не будем её рассматривать. Сосредоточимся на программном аспекте, а точнее на тех людях, которые создают код или оказывают существенное влияние на его создание. Именно их знания, опыт и компетенции будут объектом управления в контексте рассматриваемой системы управления знаниями.

Термин «управление знаниями» имеет разное толкование. С точки зрения организации практической деятельности наиболее удачное, на наш взгляд,

следующее: **«управление знаниями (knowledge management)** – процесс создания условий для выявления, сохранения и эффективного использования знаний и информации в сообществе; стратегия, направленная на предоставление вовремя нужных знаний тем членам сообщества, которым эти знания необходимы для того, чтобы повысить эффективность деятельности сообщества» [5].

Термин «знание» также неоднозначен, в данной статье он используется в следующем значении: «**знания** – это закономерности предметной области (принципы, связи, законы), полученные в результате практической деятельности и профессионального опыта, позволяющие специалистам ставить и решать задачи в этой области» [9].

Стоит отметить, что управление знаниями – сформировавшаяся область научных и практических интересов. Существует ряд теоретических моделей управления знаниями и исследователей, внесших значительный вклад в развитие этого направления. Перечислим некоторых из них: Икуджиро Нонака, Гуннар Хедлунд, Майкл Эрл, Эллис Караянис, Карл Вииг, Лейф Эдвинссон, Дэвид Сноуден, Эндрю Инкпен и Адва Динур, Ван Бурен, Деспре и Шаувель [3]. Рассмотрение предложенных авторами моделей выходит за рамки данной статьи, здесь мы сосредоточим внимание на организационно-методических аспектах управления знаниями в IT-компаниях, занимающихся программными разработками.

Можно выделить две главные составляющие систем управления знаниями: 1) техническое решение, обеспечивающее накопление знаний, а также доступ к ним, 2) корпоративная культура, способствующая обмену знаниями, их накоплению и использованию.

1. Под техническим решением понимается программно-аппаратная система, которая позволяет сохранять, искать и извлекать информацию из специального хранилища. Информация в данном случае – это документы, статьи, книги, аудио- и видеоматериалы, условно называемые знаниями. Условный характер такого именования объясняется тем, что знания не являются информацией, поэтому ставить между ними знак равенства было бы некорректно. Однако знания необходимо каким-то образом оформить и сохранить, задача технической системы как раз в этом и состоит. Естественно, речь идёт о явных, формализованных знаниях. Их формализация, перевод в явную форму – это отдельная, важная задача, решение которой осуществляется с помощью методов обмена знаниями (об этом мы поговорим позже).

Такие системы на рынке есть, они активно продвигаются и зачастую совершенно неоправданно, на наш взгляд, ставятся во главу угла. Какой бы гибкой и удобной ни была техническая система, она не способна компенсировать отсутствие корпоративной культуры обмена знаниями. Условно говоря, если нет заинтересованных в управлении знаниями людей или их мало, никакая техническая система не поможет. Она просто не будет использоваться. При этом наличие таких людей делает возможным внедрение системы управления знаниями даже без каких-то серьёзных вложений в техническую составляющую.

В самом простом варианте можно обойтись хорошо структурированным файловым хранилищем, на котором будет аккумулироваться полезная информация. В дополнение к этому целесообразно завести сводный документ, в котором будет указано, что и где находится (какой файл, в какой папке), а также приведена краткая аннотация – справка по содержанию. Реализация такого варианта хранения знаний не представляет сложностей и не требует дорогостоящих вложений.

Компаниям, не имеющим опыта по внедрению систем управления знаниями, было бы разумным начинать с такого упрощённого решения и затем уже при наличии обоснованной необходимости и чётко сформулированных технических требований переходить на более сложные, специализированные системы.

2. Под корпоративной культурой понимается совокупность социальных механизмов и ценностей, способствующих (или препятствующих) обмену знаниями, их сохранению, использованию и созданию новых знаний.

Культура проявляет себя в форме традиций и норм поведения, культура управления знаниями также может и должна проявлять себя в виде соответствующих норм и традиций, ориентированных на обмен знаниями. Очевидно, что их нужно целенаправленно развивать, т. е. последовательно и системно внедрять организационно-методические мероприятия по обмену знаниями.

Однако в отрыве от ежедневной профессиональной деятельности компании сами по себе традиции и нормы, не несущие какого-либо практического смысла и пользы, нежизнеспособны. В коммерческой компании культура не может быть самоцелью. Компания решает свои, вполне конкретные задачи на рынке, и культура должна ей в этом помогать. В софтверной IT-компании культура должна помогать прежде всего в создании программного кода.

Скорость и качество. Есть два важных показателя создаваемого кода: скорость его написания и качество (здесь имеется в виду качество исходного кода, а не конечной программы). Существуют различные метрики [15] для оценки этих показателей. При этом скорость разработки традиционно ставится на 1-е место. Это обусловлено как маркетинговыми соображениями (рынок и клиенты не готовы долго ждать), так и экономическими (стоимость разработки программной системы зависит от сроков её разработки). Иначе говоря, платят за написанный код: чем быстрее он будет написан, тем меньше нужно будет платить программистам и, соответственно, тем дешевле обойдётся разработка программы. Если главное и единственное требование к коду – его работоспособность, то качество кода будет игнорироваться, а работа по его улучшению не будет оплачиваться. Т. е. компания экономит на внутреннем качестве, которое незаметно (или малозаметно) для конечного пользователя.

Естественно, такой подход имеет свои недостатки, наиболее точно ситуация описывается через концепцию технического долга. Авторство данной концепции принадлежит Уорду Каннингему, он сформулировал её в виде метафоры финансового долга. Суть заключается в следующем: так же, как при финансовом займе, разработчики программного обеспечения (ПО) могут получить дополнительные средства на начальном этапе в виде ускорения разработки ПО, получают ускорение за счёт использования быстроразрабатываемых, но неоптимальных технических решений (алгоритмов, архитектуры). Впоследствии в процессе доработки и модификации ПО за такие решения приходится расплачиваться дополнительными трудо- и времязатратами, т. е. платить своего рода проценты по долгу. Погасить долг можно с помощью рефакторинга [16].

Таким образом, в проектах, требующих быстрого выхода на рынок, может быть принято осознанное решение пожертвовать качеством на начальном этапе ради скорости разработки. После занятия рыночной ниши, получения прибыли можно вернуться к проблеме некачественного кода и «погасить долг».

Очевидно, что использование такого подхода как базовой стратегии во всех случаях нецелесообразно. Некачественный код служит не только причиной ошибок, которые весьма трудно обнаружить, но и причиной сложности дальнейшего сопровождения проекта. При уходе специалиста, изначально разрабатывавшего программу, при передаче проекта другому программисту возникают вполне ожидаемые проблемы: в плохом коде сложнее разбираться, его сложнее модифицировать, изменения влекут за собой труднопрогнозируемые последствия и ошибки в работе программы. На всё это уходит очень много времени и сил. В крайнем случае может возникнуть такая ситуация, когда проще переписать программу заново, чем исправлять то, что есть. Таким образом, начальная экономия оборачивается существенными потерями в будущем.

Кадровые проблемы. Есть ещё один немаловажный фактор, с которым неизбежно столкнётся компания, – демотивация вновь набираемого персонала. Разбираться в чужом, плохонаписанном коде опытному, ценящему свои навыки и труд программисту не так-то интересно. Если руководство при этом ориентируется на скорость написания плохого кода, которое демонстрируют «опытные разработчики» (те программисты, которые уже давно работают в компании) и ожидает такую же скорость от вновь принятого на работу программиста при доработке программы, то естественно, что это нереалистичные ожидания. Вполне может оказаться так, что опытный и квалифицированный программист, вынужденный работать с чужим, некачественным кодом, будет демонстрировать худшие результаты, чем его коллеги с такой же и даже меньшей квалификацией, работавшие с этим кодом раньше.

Волнение, вызванное адаптацией, непривычность нового рабочего места, неизбежные потери времени на выстраивание отношений с коллегами, недостаток внимания со стороны руководителя проекта – всё это может сказаться на эффективности работы специалиста.

В совокупности эти факторы могут сформировать идею о мнимой неэффективности принятого работника. В итоге – разочарование программиста в работе, а также недовольство руководства и, как следствие, – его увольнение. Если такой сценарий воспроизводится регулярно, то фирма будет вынуждена постоянно искать новых сотрудников, т. е. фактически заниматься перебором кадров на рынке труда, что влечёт за собой дополнительные временные и финансовые потери.

Объект управления. Исходя из вышесказанного, мы приходим к пониманию важности качества кода, а также своевременной и систематической работе по его повышению. Корпоративная культура управления знаниями, безусловно, должна учитывать этот момент.

Отвечая на вопрос «Какими знаниями необходимо управлять?», мы можем сказать, что прежде всего необходимо управлять знаниями, затрагивающими качество программного кода. Это рефакторинг, принципы объектно-ориентированного программирования и проектирования [4], использование паттернов [1]. Помимо качества, несомненно, важны и другие аспекты программной разработки: алгоритмы, их построение и анализ, профилирование (оценка времени выполнения отдельных участков программы), тестирование, разработка интерфейсной части и др. Все эти знания, а точнее их обмен, передача, сохранение могут стать объектом управления.

Методы обмена знаниями. Существует ряд устоявшихся к настоящему времени мероприятий, проводимых в софтверных IT-компаниях. Данные мероприятия являются как техническими (они нацелены на улучшение кода), так и учебными (они позволяют обмениваться знаниями). Перечислим некоторые из них.

Просмотр кода (англ. *code review*). Этот метод ещё называют инспекцией или ревизией кода. Его смысл заключается в регулярном просмотре и обсуждении написанного кода, т. е. выбирается чей-то код и анализируется. На обсуждении присутствует автор и другие программисты, выступающие в роли рецензентов. Такая процедура помимо выявления ошибок и устранения неоптимальных решений позволяет обмениваться опытом, удачными решениями и подходами, а также приводит код проекта к единообразию. Если проект большой и программисты специализируются на отдельных его частях, просмотр и обсуждение чужого кода позволяет лучше понять логику и особенности работы всей системы [11].

При просмотре кода участники процесса высказывают мнения относительно целесообразности тех или иных решений, предлагают свои и получают обратную связь от коллег. Очевидно, что такое обсуждение должно быть хорошо организованным и поначалу (первые несколько мероприятий) модерируемым, чтобы критика подавалась конструктивно и доброжелательно. Позже, когда мероприятие станет частью корпоративной культуры, люди привыкнут к нему и найдут приемлемые формы критики, от модерирования можно отказаться.

Один из вариантов просмотра кода исключает публичное обсуждение, работа идёт в паре: автор – рецензент. Такой вариант снижает вероятность стресса для автора, т. к. возможные ошибки не выносятся на публику, но, соответственно, уменьшает и обучающий эффект: другие программисты компании не могут учиться на чужих ошибках, а сам автор получает неполный спектр мнений.

«*Парное программирование* – техника программирования, при которой весь исходный код создаётся парами людей, программирующих одну задачу, сидя за одним рабочим местом. Один программист управляет компьютером и в основном думает над кодированием в деталях. Другой программист сосредоточен на картине в целом и непрерывно просматривает код, производимый первым программистом. Время от времени они меняются ролями, обычно каждые полчаса» [10].

Такое взаимодействие позволяет не только повысить качество кода, но и организовать неформальный обмен знаниями. В случае значительной разницы в опыте и квалификации участников процесса эта методика трансформируется в наставничество.

Её применение может дать ещё один важный результат – совместное владение кодом. Данное понятие означает, что зона ответственности программиста расширяется на код, написанный другими. Такая практика приучает править чужой код: исправлять ошибки, улучшать и оптимизировать его. Впоследствии вероятность того, что программист исправит чужой код, даже не работая в данный момент в паре с его автором, значительно повышается.

Необязательно работать в паре всё время, такую деятельность можно чередовать с традиционным одиночным программированием. Также периодически полезно менять состав пар, чтобы в идеале каждый программист поработал со всеми другими участниками проекта. Такой подход может значительно усилить рабочие и межличностные отношения, способствовать т. н. срабатыванию команды.

Лекции и семинары. Здесь как нельзя лучше подходит поговорка «всё новое – это хорошо забытое старое». Такие привычные любому студенту формы обучения мало распространены в российских IT-компаниях. Вместе с тем, на наш взгляд, они могли бы успешно дополнить перечисленные выше методы. Нельзя сказать, что такие формы передачи знаний совершенно чужды IT-среде. Практически ни одна тематическая IT-конференция не обходится без специально подготовленных докладов в форме мини-лекций или семинаров. Было бы уместным внедрять такие формы и в компаниях. Однако это сопряжено с различными сложностями.

Первая и наиболее очевидная проблема – большие время- и трудозатраты докладчика. К таким мероприятиям нужно готовиться. Вряд ли руководство компании пойдёт на то, чтобы освободить программиста от его основной работы ради подготовки к выступлению. Скорее всего, ему придётся готовиться в свободное от работы время или выискивать его в перерывах. Очевидно, что таких высокомотивированных специалистов, готовых делиться своими знаниями и тратить на подготовку своё личное время, не так уж и много. Поэтому для успешного использования подобной формы обучения необходимо разрабатывать и внедрять систему поощрений, т. е. докладчик должен вознаграждаться за свой труд.

На первый взгляд, такая форма управления знаниями затратна для компании, но не будем забывать о возможности видеофиксации выступления и последующего его транслирования без каких либо дополнительных вложений. Один раз создав такую видеолекцию, её можно показывать много раз неограниченному числу сотрудников. Наверное, наиболее актуально это для крупных компаний, где происходит регулярное пополнение сотрудников.

Вместе с тем живое выступление, возможность получать от докладчика ответы на вопросы всё же предпочтительнее.

Вторая, менее очевидная проблема – это отсутствие навыков публичного выступления у тех специалистов, которые могли бы поделиться своими знаниями. Здесь сложно предложить какое-то универсальное решение, максимум, что можно сделать, – это создать благоприятные психологические условия для выступления: предоставить уютное, комфортабельное помещение, задать неформальный характер встречи и др.

Таким образом, эта форма обучения в значительной мере зависит от желания потенциальных лекторов. Навязывать её было бы неправильно, но поощрять можно и нужно.

Подводя итог данному исследованию, можно сказать, что система управления знаниями в IT-компании должна включать в себя две составляющие: техническую и организационную. При этом, на наш взгляд, организационная составляющая должна иметь приоритет над технической. Внедрение системы управления знаниями должно начинаться с организационно-методических мероприятий, нацеленных на развитие корпоративной культуры обучения и самообучения. Развитие образовательных традиций будет способствовать повышению компетентности сотрудников и, как следствие, – эффективности компании.

В крупных компаниях со значительным числом программистов (несколько десятков и более) имеет смысл вводить должность для специалиста, отвечающего исключительно за развитие корпоративной культуры управления знаниями. Предпочтительно, если этот человек

будет иметь знания не только по программированию, но и по психологии, т. к. его сфера ответственности затрагивает и технические знания, и психологию человека. Если говорить о технических знаниях, то он должен уметь оценивать их качество и ценность, т. е. хорошо владеть предметной областью. Если говорить о психологии, он должен уметь общаться с людьми, находить с ними общий язык, а также мотивировать их. Такой специалист сможет оценивать знания и компетенции сотрудников, следить за их профессиональным ростом и поддерживать их.

Искать подобного специалиста среди менеджеров по работе с персоналом бессмысленно, у них нет специальных технических знаний, искать нужно среди программистов. Недостаток базовых психологических знаний можно компенсировать с помощью краткосрочных курсов, специальных тренингов и самообразования, т. е. программиста необходимо дообучить.

Ориентируясь на современный опыт отечественных компаний, отметим, что некоторые функции такого специалиста выполняет менеджер проекта. В небольших IT-компаниях в силу ограниченности их финансовых возможностей этот подход вполне оправдан, в средних (ориентировочно от 20 до 30 программистов) и крупных компаниях лучше вводить специальную штатную должность.

ЛИТЕРАТУРА

1. Гамма, Э. Приёмы объектно-ориентированного проектирования. Паттерны проектирования [Текст]: пер. с англ. / Э. Гамма, Р. Хелм, Р. Джонсон, Дж. Влиссидес. – СПб.: Питер, 2007. – 366 с.
2. Гапоненко, А.Л. Управление знаниями [Текст] / А.Л. Гапоненко. – М.: ИПК Госслужбы, 2001. – 52 с.
3. Гретченко, А.А. Современные концепции управления знаниями [Текст] / А.А. Гретченко // Сборник научных трудов РЭА им. Г.В. Плеханова. – М.: РЭА, 2009.
4. Ларман, К. Применение UML 2.0 и шаблонов проектирования [Текст]: пер. с англ. / К. Ларман. – М.: Вильямс, 2006. – 736 с.
5. Мариничева, М. 10 общепринятых заблуждений об управлении знаниями (Knowledge Management) [Электронный ресурс]. – Режим доступа: http://www.iteam.ru/publications/human/section_55/article_3080/
6. Маринко, Г.И. Современные модели и школы в управлении знаниями [Текст] / Г.И. Маринко // Вестник Московского университета. – 2004. – № 2. – С. 45-65.
7. Мильнер, Б.З. Управление знаниями. Эволюция и революция в организации [Текст] / Б.З. Мильнер. – М.: Инфра-М, 2003. – 177 с.
8. Нонака, И. Компания – создатель знания. Зарождение и развитие инноваций в японских фирмах [Текст]: пер. с англ. / И. Нонака, Х. Такеучи. – М.: Олимп-Бизнес, 2003. – 384 с.
9. Основные понятия представления знаний [Электронный ресурс]. – Режим доступа: http://itteach.ru/predstavlenie-znaniy/osnovnie-ponyatiya-predstavleniya-znaniy
10. Парное программирование [Электронный ресурс]. – Режим доступа: http://ru.wikipedia.org/wiki/Парное_программирование
11. Пахунов, А. Рецензирование кода (code review) [Электронный ресурс]. – Режим доступа: http://blog.not-a-kernel-guy.com/2007/02/21/151
12. Петелин, Д. «Свалки данных» и системы управления знаниями [Электронный ресурс]. – Режим доступа: http://www.pcweek.ru/themes/detail.php?ID=72524
13. Подолякин, О.В. Внедрение информационных систем управления на предприятии [Текст] / О.В. Подолякин // Проблемы развития территории. – 2012. – № 4 (60). – С. 20-28.

14. Ригин, В.А. Информатизация в аспекте процессно-ориентированного подхода к управлению предприятием [Текст] / В.А. Ригин // Проблемы развития территории. – 2012. – № 2 (58). – С. 86-91.
15. Рыжков, Е. Программный код и его метрики [Электронный ресурс]. – Режим доступа: http://habrahabr.ru/company/intel/blog/106082/
16. Фаулер, М. Рефакторинг: улучшение существующего кода [Текст]: пер. с англ. / М. Фаулер. – СПб.: Символ-Плюс, 2003. – 432 с.
17. Li-Su Huang, Mohammed Quaddus, Anna L Rowe and Cheng-Po Lai. An investigation into the factors affecting knowledge management adoption and practice in the life insurance business / Knowledge Management Research & Practice (2011) 9, 58–72.
18. Thomas H. Davenport, Laurence Prusak. Working Knowledge: How Organizations Manage What They Know, Harvard Business Press, 1998.