Руслан Раянов

Как создать собственную CRM

Руководство по созданию CRM для вашего бизнеса

www.web-automation.ru 2015

Оглавление

Введение

Итак, вы наконец решили внедрить в своей организации CRM. В этом небольшом руководстве Вы найдете методику создания собственной CRM, адаптированной под ваш бизнес.

Тем, кто будет использовать готовую CRM, данное руководство поможет лучше понять собственные потребности в плане организации бизнес-процессов вашей компании (продажи, контрагенты, финансы, проекты и т.д.).

Давайте начинать, и в первой главе мы рассмотрим, что такое CRM и зачем она нужна.

Глава 1. Что такое CRM и зачем она нужна.

Если кратко, то CRM – это программа, предназначенная для учета, обработки и хранения информации о взаимоотношениях с вашими клиентами. Мы это понятие слегка расширим и будем считать, что это программа, позволяющая упростить управление вашими бизнес-процессами.

Допустим, вы продаете некоторые продукты. Сразу можно выделить процессы, которые возможно автоматизировать с помощью CRM:

- учет Поставщиков;
- закупка у Поставщиков;
- продажи;
- база Клиентов;
- база Сотрудников;
- отделы;
- финансовые транзакции (счета);
- система мотивации;
- брак;
- каналы сбыта.

В итоге CRM позволяет добиться следующих целей:

- уменьшение издержек. Автоматизация сложных вычислений и других операций. Уменьшение лишней коммуникации между сотрудниками и формализация бизнес-процессов.
- ускорение бизнес-процессов. Уведомления (почта, СМС), исключение ненужных рутинных задач из бизнес-процесса и упрощение операций обработки данных (например, поиск по базе клиентов).
- улучшение качества продукции. В основном за счет ведения метрик (рассмотрим далее).
- повышение прозрачности бизнеса. Данные по работе всех сотрудников заносятся в систему. Есть возможность делать на основании этих данных отчеты руководству для понимания ситуации на местах.
- управление на основе KPI. Для каждого процесса можно выделить показатели эффективности (метрики) и начать их отслеживать с помощью CRM (через панель метрик в личном кабинете руководителя). Внедрили инновацию – оцениваете изменение метрик.
- стандартизация бизнес-процессов. Все сотрудники работают по определенной схеме, которая заложена в CRM. Плюс работают правила отфильтровки неверных данных, что способствует унификации данных.

Теперь давайте определимся с выбором вашей CRM.

Возможны три варианта:

1. Использовать готовую CRM общего назначения.
2. Использовать готовое CRM - решение для вашей отрасли.
3. Создать собственную CRM.

Если ваши бизнес-процессы не имеют уникальных особенностей и довольно типичны, то лучшим для вас вариантом будет установка CRM общего назначения. Ищите готовую CRM – она стоит относительно недорого (в некоторых случаях даже бесплатна) и имеет довольно внушительное количество функций. Однако вам нужно будет подстроиться под систему и использовать ее как есть. Поэтому при установке такой CRM лучше заранее представить свои будущие потребности в автоматизации и понять, сможет ли система позволить покрыть эти потребности.

Хорошо, если для вашей отрасли есть готовая CRM, учитывающая особенности этой отрасли. По большому счету, вы можете подстроить свои процессы под эту CRM и начать действовать в рамках ее функциональности. Плюс этого подхода – вы сразу и за небольшие деньги получаете готовую методику работы в вашей отрасли. Минус – при развитии вашего бизнеса на развитие CRM вы не можете влиять. Т.е. ваши возможности будут ограничены тем, что предусмотрели для вас разработчики данной CRM.

В любом случае этот вариант достоин рассмотрения. Как минимум необходимо изучить 1-2 программы данного рода и посмотреть, подходят ли она вам.

Следующий вариант – **разработка CRM на заказ**. Вы сами решаете, что будет в вашей системе и как именно будут протекать процессы. Для этого вам придется полностью погрузиться в свои бизнес-процессы и разработать их от начала до конца.

Этот подход предпочтителен для компаний со сложной бизнес-логикой процессов. Такой компании нужен свой продукт, который будет учитывать особенности и специфику бизнеса.

Важный момент – развитие и масштабирование бизнеса. Процессы со временем меняются, оптимизируются. В своей системе вы можете развивать функциональность системы в нужном направлении. В CRM общего назначения это довольно проблематично, поэтому об этом лучше подумать заранее.

Получается, что собственная CRM нужна компаниям, у которых либо сложные бизнес-процессы, либо тем, кто собирается оптимизировать свой бизнес в процессе эксплуатации CRM.

Конечно, разработка собственной CRM стоит дороже, чем внедрение готовой, т.к. программное приложение по сути пишется под

потребности вашего бизнеса. Также сроки внедрения собственной CRM больше, чем готовой системы. Здесь важно определиться с краткосрочными и долгосрочными приоритетами. Что важнее? Скорость внедрения, стоимость, учет специфики бизнес-процессов, безопасность, удобство и др.

В дальнейшем мы рассмотрим третий вариант внедрения CRM – создание CRM под требования вашего бизнеса.

В этом случае необходимо будет пройти все стадии разработки программного продукта, а именно:

- концепция;
- техническое задание;
- проектирование и кодирование;
- тестирование и доработки;
- внедрение в эксплуатацию.

В следующей главе мы кратко поговорим о концепции проекта и как ее правильно составить.

Глава 2. Концепция проекта CRM.

Допустим, вы решили разработать собственную CRM. Для этого необходимо написать концепцию будущего проекта.

По сути, концепция – это документ, на основании которого вы будете общаться с подрядчиками-кандидатами.Очень важно составить ее правильно, чтобы на ранней стадии отсеять неподходящих подрядчиков.

Что должна содержать концепция проекта:

- ➢ Краткое описание проекта (1 предложение).
- ➢ Состав модулей (определение ролей и их функций).
- ➢ Требования к технологиям.
- ➢ Параметры проекта (сроки, стоимость).
- ➢ Требования к команде разработки + состав брифа для нее.

Пройдемся подробнее по каждому пункту.

Краткое описание проекта.

Вы должны описать одним предложением точную суть вашего проекта. Тем самым вы сэкономите кучу времени другим людям, которые будут просматривать вашу концепцию. Уже по заголовку будет понятно – подходит ли это подрядчику или нет.

Дело в том, что при неясном описании вам будут задавать одни и те же вопросы – и вы будете тратить время на их обработку. Поэтому исключите интригу (создание очень большой программы для суперкомпании), пишите просто и понятно.

Если есть критичные требования, то их тоже лучше указать в заголовке. Это могут быть технологии, территориальная расположенность подрядчика, сроки, бюджет.

Плохие примеры:
Создание сайта
Нужна программа
CRM

Хорошие примеры:
Нужна студия на разработку движка CRM
Разработка CRM на заказ
Разработка CRM на заказ, только Москва.

Состав модулей (определение ролей пользователей и их функций). Для начала хотя бы в общих чертах определитесь, какие будут в вашей системе роли. Роль – это типовой пользователь в вашей системе. Например, продавец, администратор, оператор, менеджер, работник склада и т.д.

После того, как вы определили роли в системе, кратко опишите их. Например, роль - продавец. Продавец имеет следующие функции: создание коммерческого предложения, просмотр базы клиентов, обработка потенциальных клиентов и их предзаказов, печать документов для заказов.

Требования к технологиям

Если у вас на предприятии уже используются сервера, то подумайте о том, чтобы будущая CRM была совместима по технологиям с существующей инфраструктурой. Это необязательное требование, но лучше заранее продумать этот момент. К примеру, если у вас на сервере используется ПО MS Windows, то нет смысла делать свою CRM на технологиях, которые подразумевают использование серверов Linux. Если все ваши системы будут сделаны на одних и тех же технологиях, это снизит стоимость поддержки IT-решений.

Если нет никакой привязки к технологиям, то тоже лучше кратко укажите этот момент к концепции. Будет меньше вопросов.

Параметры проекта.

Указывайте всегда вилки оценки стоимости и сроков.

Точную стоимость указывать бессмысленно, т.к. в ходе проекта будут новые дополнения и развитие функционала. Нужен просто порядок цен и сроков, чтобы подрядчик это соотносил со своими возможностями / ресурсами.

Т.е. сейчас задача не стоит в том, чтобы точно оценить бюджет и сроки. Нужно просто понимание контуров проекта. Для заказчика и для подрядчика.

Есть еще один параметр проекта – качество. Но его довольно сложно описать кратко в концепции. Как вариант, вы можете требовать некоторых гарантий. С другой стороны, гарантии вам выдают продавцы от IT компаний, а обеспечить их будут разработчики. И здесь иногда возникает некоторое рассогласование обещаний продавцов и фактического сервиса компании.

Требования к команде разработки.

Укажите, кто именно вам нужен. Если у вас есть IT-директор или технический специалист, то привлеките его к этой работе.

Какой должен быть состав команды разработки?

Какой компетенцией они должны обладать?

Есть ли пожелания по взаимодействию с командой? (например, условие – хотим иметь возможность напрямую общаться с разработчиками, а не только через менеджера).
В какие часы/дни специалисты должны быть доступны?
Какие средства связи использовать (скайп, телефон)?
Личные встречи?

После всех вопросов напишите, что конкретно вы хотите получить от исполнителя, если его заинтересовал проект.
Вообщем, вам нужен от него заполненный бриф.

Что может включать бриф:
- ваша оценка проекта (сроки, бюджет)
- ответы на вопросы
- технологии
- порядок работ
- город, ФИО, контакты
- портфолио, кейсы
- предложения по созданию CRM

А теперь давайте рассмотрим, какими качествами должна обладать хорошая концепция:

- краткость и лаконичность. Без «воды». Если написано слишком много и беспорядочно, то есть вероятность, что многие даже не прочтут вашу концепцию, либо поймут ее неверно.

- структурированность. Структура рассмотрена выше.
- однозначно определяет порядок действий. Желательно точно и понятно описать, что должен сделать потенциальный подрядчик для начала взаимодействия по проекту.

После того, как вы написали концепцию, покажите ее своим коллегам, друзьям и родственникам. Им понятно, что там написано? Возникают ли встречные вопросы по концепции, которые надо осветить в ней? Если вопросов нет, то либо можно переходить к следующему шагу, либо им совсем ничего не понятно. Задайте вопросы по концепции и добейтесь четкого однозначного ее понимания.

Проработав свою концепцию и протестировав ее на своем окружении, приступайте к поиску подрядчика для реализации своего проекта и написанию технического задания.

Об этом мы и поговорим в следующей главе.

Глава 3. Пишем техническое задание и выбираем исполнителя на проект

Наверное, каждый понимает, что качество его будущего продукта зависит в большей степени от качества работ исполнителя, чем от технологий, которые он выберет. В этой главе мы сосредоточимся на том, как подобрать исполнителя и написать совместно с ним техническое задание (ТЗ).

Почему именно в такой последовательности? Дело в том, что при написании ТЗ исполнитель сразу будет его оценивать как сторона, которой все, что написано в ТЗ, необходимо будет реализовать. Еще одним плюсом этого подхода является то, что в процессе написания ТЗ исполнитель глубже погрузится в вашу предметную область и более четко представит ваши цели и задачи.

В прошлой главе мы составили концепцию проекта. Этот документ мы будем использовать как основной для поиска исполнителя.

Я бы выделил три канала поиска кандидатов на разработку:

1. Поисковики. Ищем подрядчика через поисковые системы. Сразу попадаем на его сайт и изучаем его предложение, портфолио и т.д.
2. Биржи. Заходим на биржу фрилансеров типа fl.ru и выкладываем свою концепцию. Исполнители сами пишут вам свои предложения.
3. Рейтинги. Вы находите рейтинг поставщиков CRM и исходя из него выбираете. Понятно, что выбирая поставщика с высоким местом в рейтинге, вы имеете дело с высоким ценовым сектором.

Более простой способ - опубликовать проект на бирже. Исполнители сами вас найдут.

Важно, чтобы в концепции вы сразу написали о том, кто вам подходит, а кто - нет. Также укажите максимально конкретно, какую информацию и в каком виде вы хотите получить от кандидата.

После того как вы получите 3-10 предложений от подходящих кандидатов, вам предстоит выбрать из них одного. Хотя в некоторых случаях, если ваш бюджет позволяет, вы можете начать работу над ТЗ параллельно несколькими кандидатами. Через некоторое время будет понятно с кем проще и продуктивнее работать. При этом вы никого не обманываете, т.к. другие исполнители получат деньги за проделанную часть работы.

Какие критерии отбора я бы выделил в первую очередь:

- ➢ Скорость и полнота ответа. Это показывает, насколько заинтересован исполнитель в проекте. Нет заведомо пустых обещаний (например, сделаем и внедрим за 1-2 недели).
- ➢ Демо и портфолио. Вам необходимо понять какой backend есть у исполнителя. Делал ли он подобные проекты?
- ➢ Ценовой диапазон и сроки. Насколько их оценки соотносятся с вашей оценкой? Они не должны отличаться более чем в 3 раза.

Какие действия рекомендуется сделать при выборе исполнителей:

- Просмотреть их сайт, портфолио, кейсы, демонстрации.
- Проверить отзывы исполнителя. Свяжитесь с людьми, которые оставили эти отзывы.Поговорите с ними по скайпу, телефону.
- Дают ли они сразу полезную информацию по проекту? Есть ли у них четкий план по созданию вашей CRM?
- Поищите информацию об исполнителе через поисковые системы и социальные сети. Соотносятся найденные данные с тем образом, который строит исполнитель на своем сайте?

Допустим, вы выбрали одного исполнителя и решили с ним работать. Что дальше? Настало время написать ТЗ.

Некоторые заказчики считают, что ТЗ - это просто формальность и исполнители должны делать его бесплатно. Такое может быть, когда

продукт настолько типовой, что ТЗ получается при изменении нескольких параметров в шаблоне документа.

Разработка ТЗ для создания CRM на заказ – это довольно большая и объемная работа, которая требует не только фиксации требований заказчика, но и проработки предметной области клиента, предпроектирования некоторых модулей системы, создания тестовой функциональности с учетом специфики процессов.

Отнеситесь к написанию ТЗ с полной серьезностью. От качества ТЗ в дальнейшем зависит качество реализации системы. К тому же ТЗ может служить прототипом для будущей документации по системе.

Что же должно включать полноценное ТЗ?

- Структуру будущего продукта. Для CRM – это состав кабинетов пользователей и их функционала (страниц). Структура – это исходный элемент для всех остальных работ по ТЗ.
- Макеты и текстовое описание всего функционала по структуре. Макеты визуализируют ТЗ. Это снижает риски недопонимания между заказчиком и исполнителем Чаще всего исполнители тяготеют к техническому языку, а заказчики – к бизнес-целям. Макеты позволяют общаться на одном языке – все видят один и тот же интерфейс и общаются не на абстрактном уровне, а на уровне кнопок, таблиц, меню и т.д.

- Требования по интеграции. Если система связана с внешними ресурсами, то вы должны прописать, как именно система будет взаимодействовать с ними и в каком объеме. Например, интеграция с 1С. Не пишите в ТЗ «Сделать интеграцию с 1С» - 1С содержит сотни таблиц в базе данных. Очень трудоемко сделать интеграцию всех объектов 1С с приложением, да вам это и не нужно в большинстве случаев. Лучше указать, какие именно объекты надо синхронизировать с приложением (клиенты, заказы, единицы измерения, товары и т.д.). Указывайте также предпочтительный способ реализации (например, через веб-сервисы) – это позволит избежать недоразумений на стадии разработки.
- Согласен, вы не можете знать всех нюансов. В этом плане вам должен помогать исполнитель, который пишет ТЗ. Задавайте ему максимум вопросов по реализации. Его задача – предоставить вам информацию, а вы на ее основании уже решаете какой вариант выбрать.
- Особые требования. Сюда можно отнести требования по производительности (хотя в некоторых случаях это сложно спрогнозировать, т.к. приложение работает не изолированно, а в некоторой среде), требования к браузерам, требования по дизайну и др.
- При составлении требований указывайте список браузеров и их версии, для которых должно работать приложение. Некоторые

версии (например, IE8) не поддерживают такие порталы как ВКонтакте.

- ➢ Если говорить о дизайне CRM, то главная его задача – это не мешать пользователю. На мой взгляд, здесь не нужно придумывать велосипед – можно взять готовый шаблон и использовать его.
- ➢ Проработка «узких мест». Если в вашем проекте есть сложные механизмы, то лучше их проработать на этапе ТЗ. Это снижает риски проекта. В случае если не найдется приемлемого решения, то лучше изменить что-то на этапе ТЗ, а не на этапе разработки системы. Почему? Это дешевле. Раньше обнаружена ошибка – дешевле ее исправить. Самые дорогие ошибки – на этапе эксплуатации. Пример: автоматическая генерация пароля. В начале проекта вы сделали так, что пароль генерируется слишком простой. Сейчас изменить это несложно. Теперь представьте, что у вас система запущена, и в ней работают 100 пользователей. Будет довольно непросто сменить всем пароли (поменять данные в базе, оповестить пользователей, часть пользователей не поймет что произошло и у вас появятся дополнительные расходы на техподдержку).

Как лучше всего организовать процесс написания ТЗ?

Во-первых, не думайте что исполнитель полностью напишет за вас ТЗ. Вернее он напишет, но это будет не совсем то, что вам нужно.

Во-вторых, пытаясь сэкономить на ТЗ, некоторые заказчики делают ТЗ полностью самостоятельно. Тем самым они полностью полагаются на свои решения, которые могут быть неоптимальными с технической точки зрения или с точки зрения удобства пользования (юзабилити).

На мой взгляд, наиболее предпочтительна следующая схема взаимодействия: исполнитель и заказчик работают совместно над ТЗ (например, через сессии по скайпу). Причем исполнитель руководит этим процессом, а заказчик выступает в роли источника знаний о системе. Иными словами, исполнитель спрашивает, а заказчик отвечает. Только исполнитель фиксирует изменения ТЗ. Заказчик периодически анализирует ТЗ. Все понятно? Ничего не упущено? Надо что-то дополнить?

Используя такой подход, мы достигаем того, что обе стороны держат под контролем содержимое и форму ТЗ.

Также стоит упомянуть о случаях, когда очень трудоемко написать ТЗ сразу. Система настолько сложна и быстро меняется, что нет смысла писать сразу большое ТЗ. В этом случае необходимо определить область действия ТЗ. Делайте ТЗ только на конкретную часть системы, например, на блок «База клиентов» или процесс «Обработка заказа от клиента».

При этом желательно иметь план внедрения модулей системы. Например, в этом году мы внедрим модули 1-5, а в следующем – 6-9 модули.

Из опыта вспоминаются ситуации, когда некоторые заказчики хотят выполнить очень большой проект за один этап, например, реализовать аналог Avito.ru. Желание заказчика понятно – он хочет избежать риска получить часть продукта, ведь ему нужен целостный продукт. Сам не понимая того, он движет проект к катастрофе, т.к. подобные проекты невозможно сделать одним куском за один раз. Здесь действует цикл «задумка – реализация – получение обратной связи - анализ». Вспомните, Windows не сразу стал тем, что он есть сейчас.

Есть и обратные примеры, когда разработка ведется фрагментарно. Новые функции появляются без всякого плана, спонтанно. Это тоже плохой вариант развития ситуации, поскольку теряется фокус и функции добавляются хаотично.

Очень важный момент. Обе эти ситуации порождает заказчик, а не исполнитель! Именно заказчик определяет стиль стратегического управления продуктом. Исполнителю остается только разве что пожимать плечами и подчиниться пожеланию заказчика. Если у вас большой проект, то имейте стратегический план его развития и не пишите большое ТЗ сразу на всю систему.

Итак, вы нашли исполнителя, написали совместно с ним ТЗ. Пора приступать к разработке CRM. В следующей главе мы рассмотрим, как управлять проектом со стороны заказчика, какие бывают подводные камни и как построить взаимодействие с командой разработки.

Глава 4. Реализация проекта и внедрение системы

Итак, начинаем проект по созданию CRM-системы. Что нужно сделать в первую очередь? Заключить договор. Заключение договора более предпочтительный способ, чем другие.

Работа по безопасной сделке, например, через fl.ru подразумевает использование третьей стороны в качестве арбитража между вами и исполнителем. Тем самым вы предоставляте доступ к своим коммерческим данным третьей стороне. На мой взгляд, подобные сервисы – это некоторый компромисс между официальными договорами и работой без договора.

Работа без договора. Здесь нюанс в том, что разработка системы предполагает длительное сотрудничество. В случае создания простого сайта визитки, возможно, лучше работать без договора (или по договору оферты). Но в случае создания большой серьезной системы у вас должны быть гарантии, что исполнитель не испарится внезапно.

Что прописывать в договоре? В первую очередь - это права на систему, сроки, бюджет и порядок приемки. Обычно такой договор

составляется подрядчиком (скорее всего, у него есть типовой договор). Внимательно читайте эти пункты, особенно порядок приемки.

По цене и срокам. Если разработка системы делится на несколько этапов, то бессмысленно писать сроки и стоимость в основном договоре, т.к. ни одна из сторон не может дать точной оценки сроков и бюджета проекта. Это будет лишь вносить элемент неопределенности и нервотрепки в ваши взаимоотношения с исполнителем. Лучше писать стоимость по каждому этапу в очередном дополнительном соглашении к договору.

Аналогично и с техническим заданием. Если у вас ТЗ разбивается по этапам, то оформляйте его как часть дополнительного соглашения к договору.

Авторские права на систему должны принадлежать вам. При этом исполнитель сохраняет за собой право использования компонентов общего назначения и ядра системы, т.к. система обычно разрабатывается на базе платформы. Причем это может быть платформа третьей стороны, например, 1С-Битрикс, или являться собственностью исполнителя, как в нашем случае, arkAS.

Теперь перейдем к процессу взаимодействия с исполнителем.

Обычно от исполнителя выделяется человек, который будет взаимодействовать с вами. В его задачи обычно входит:

- обработка ваших запросов и пожеланий;
- подготовка отчетов по ходу проекта;
- решение внештатных ситуаций.

С вашей стороны хорошо бы иметь доступ не только к этому человеку, но и к команде разработки. Обычно разработчики менее подкованы в плане дозированный выдачи информации клиенту – вы сможете узнать более реалистичную картинку. Однако разработчики разговаривают на довольно специфичном языке. Будьте готовы к тому, что не всегда получится полное взаимопонимание с ними. Именно для этого и придумали должность бизнес-аналитика.

Важный момент. Не думайте, что вы сделали ТЗ, отдали группе разработки и можно забыть о проекте на пару месяцев. При таком подходе вы не получите нужного вам результата. Контролируйте минимум 1 раз в неделю как идут дела по проекту. При этом изучайте не только отчеты команды разработки, но и смотрите своими глазами тестовое приложение и пробуйте использовать тестовые функции. Тем самым вы сможете вовремя вносить корректировки в процесс развития функционала проекта. Ваша задача – постоянно давать обратную связь по разработанному функционалу.

Следующий важный момент – при планировании проекта менеджером вы должны четко обозначить свои приоритеты. Что нужно получить сначала? Что можно сделать уже на поздних этапах. Понимания ваши приоритеты, менеджер проекта сможет транслировать их на команду разработки.

Некоторые заказчики определяют все свои задачи важными и необходимыми для реализации. Это управленческий просчет. В проекте может быть много разных рисков (конфликт с исполнителем, проблемы с бюджетированием, выход за сроки, переработка уже сделанного функционала и многое другое). Желательно, чтобы при резком прерывании проекта у вас был минимально необходимый работающий функционал, который можно использовать в работе. Стремитесь к тому, чтобы ваша система уже через месяц работала хотя бы в тестовом режиме.

Документация. Она помогает понять заказчику, правильно ли его понимает исполнитель.

Документацию надо писать сразу. Одна из причин - резкая смена исполнителя. Исполнитель чем-то вас не устраивает или произошел конфликт. Пока у вас нет документации и исходного кода – считайте себя заложником.

Писать документацию проще по горячим следам. Требуйте документацию сразу же после тестирования. Причем требуйте документацию не только для пользователя, но и для программиста. С другой стороны, на оформление документации нужно выделить время. Поэтому не спрашивайте подробно написанных и оформленных документов. Подойдет просто текстовое описание и скриншоты.

Разработчики в первую очередь должны разрабатывать, а не писать документы. Лучше всего возложить оформление документации по использованию системы на тестировщика со своей стороны. Таким образом, вы сможете лучше контролировать процесс, а разработчики при этом будут минимум времени тратить на документацию и ее оформление.

Мелкие и крупные правки. В проекте всегда бывают правки. Написание вами подробного ТЗ не значит, что все будет именно так, как написано в ТЗ. По мере развития проекта вы получаете дополнительную информацию, у вас появляются новые идеи. Возникает желание что-то улучшить. Как быть в этом случае? Если в договоре стоит фиксированная сумма, то вносить большие доработки в приложение без соответствия ТЗ будет некорректным.

При малых правках в большинстве случаев исполнитель идет навстречу заказчику и вносит их в систему.

Если грядут большие перемены, то следует составить новое ТЗ как дополнительное соглашение к договору и изменить план разработки.

Если вы работаете по нефиксированной сумме оплаты, то вы можете сразу менять свои требования к системе. Исполнитель вносит на очередной итерации нововведения по системе.

Старайтесь как можно меньше делать крупных необдуманных изменений в системе. Напишите небольшую концепцию на это изменение и отложите на некоторое время. Если это изменение действительно важно, то вы обязательно вернетесь к нему в будущем.

Всем хотелось бы, чтобы проекты выполнялись четко, быстро, качественно и полном объеме. Но в любом проекте бывают различные проблемы.

Как владельцу проекта вам следует помнить о параметрах проекта: стоимость, сроки, качество, объем работ.

При возникновении проблемы по одному из параметров, вы можете изменять другие и тем самым влиять на проект, чтобы довести его до конечного состояния,
Например, вы не можете закрыть проект в ранее планируемые сроки. В таком случае можно где-то пожертвовать качеством, нанять дополнительно программистов, часть функций убрать.

Не зная о параметрах проекта, очень просто впасть в состояние «Хочу сделать CRM очень качественную, за 50 тыс руб, желательно за 1-2

недели и при этом одна должна содержать все функции 1С». Самое печальное – это не шутка, такие проекты (чаще социальные сети и доски объявлений) встречаются на биржах .

Также важно учитывать человеческий фактор. Нередко проблема именно в личном конфликте участников проекта. Важно знать цели участников проекта. Заказчик должен понимать, что требуется для работы исполнителю. Исполнитель должен понимать, зачем вам нужен проект. Дайте исполнителю всю необходимую информацию и расскажите, для чего вам нужен проект и как вы будете использовать его результаты.

Общение – важная часть взаимодействия. Не скатывайтесь полностью на переписку. Да, в некотором смысле она удобнее, но отсутствие живого контакта влияет на понимание между сторонами. Бывает, что заказчик «садится на уши» исполнителю и перегружает его несущественной информацией. Это заставляет исполнителя избегать заказчика и ограничивать с ним общение.

В нашей практике были разные проекты. Кто-то из заказчиков предпочитал связываться с нами только по электронной почте. А с некоторыми мы на протяжении всего проекта много общались и занимались подготовкой большого количества различных документов. С точки зрения рисков проекта, второй вариант предпочтительнее, но все же надо стремиться к золотой середине.

Также стоит упомянуть о личных встречах. Для людей, привыкших вести бизнес по старой схеме, это является необходимым шагом в ведении проекта.

Мое мнение – это часто бывает потерей времени. Обычно цель такой встречи – получить ощущение, что исполнитель - реальный человек и, в некоторых случаях, «прощупать» его в плане адекватности и подтвердить, что он не кинет заказчика. Но проблема в том, что именно «кидала» больше готов к подобным встречам, чем команда технарей-разработчиков.

Есть ли информация, которую вы можете получить только при личной встрече и не можете получить при конференц-связи по скайпу?

Используйте личную встречу, если вы точно знаете, что она может вам дать и у вас есть некая методика получения этого.

Теперь давайте поговорим о предметной области вашего бизнеса. Очень хорошо, если у вас есть небольшой мануал по вашей предметной области, который раскрывает основные понятия и процессы вашего бизнеса. Будет полезно объяснить разработчикам, зачем нужен тот или иной функционал разрабатываемой CRM.

Без понимания предметной области разработчики будут ваять, сами не зная что. Они будут просто механически делать то, что им скажут. Этот подход чреват ошибками. Разработчик должен хотя бы частично представлять как будет работать заказчик, какие у него будут

потребности. Это значительно улучшит качество продукта. При этом, зная потребности конечного клиента, исполнитель сможет предложить лучшие решения реализации в вашей системе.

Мы уже упоминали о макетировании. При ведении проекта старайтесь общаться с командой разработки визуальным языком. Если оказалось, что в ТЗ было описано недостаточно, то рисуйте наброски того, что нужно получить. Если что-то не работает как планировалось, то скиньте исполнителю скрин с комментариями для поправки. Сейчас довольно много хороших программ для создания скриншотов с возможностью вставки текста и стрелки на скриншоте (примеры программ – Яндекс.Диск, clip2net). Также при ошибках лучше указывать URL проблемы и дополнительные данных (пользователь, какая операция выполнялась и когда зафиксирована ошибка).

Теперь о процессе внедрения системы.
На этом этапе система делится на DEV и PROD версию. Первая нужна для тестирования разработчиками, вторая – для промышленной эксплуатации системы. Такое разделение позволяет избежать внесения случайных ошибок при сопровождении системы.

При внедрении системы:

- получите постоянный доступ к исходному коду. Пусть исполнитель настроит для вас такой доступ и даст инструкцию как его использовать. В качестве средства такого доступа может

быть FTP (протокол для доступа к файлам) или SVN (средство командной разработки).

- делайте резервное копирование для сохранности ваших данных. При этом данные лучше хранить в удаленном хранилище (т.е. не на сервере, где работает приложение). Делать бекап базы лучше не менее 1 раза в день. Копирование на удаленное хранилище можно делать 1 раз в неделю. В качестве такого хранилища подойдет Dropbox или Яндекс Диск.
- проверяйте работу периодически запускаемых на сервере скриптов (например, скрипт создания бекапов).
- делайте мониторинг доступности веб-приложения. Есть замечательные бесплатные сервисы вроде Uptime Robot.
- проверяйте работу почты и уведомлений. У почты бывают сбои, поэтому обязательно проконтролируйте работу почтовых уведомлений на основной версии приложения.

Перед тем как мы перейдем к следующей главе, хотелось бы рассказать о тех принципах, которыми руководствуются исполнители при разработке программного обеспечения. Я считаю, что если у заказчика есть представление об этих принципах и он ими пользуется при принятии решений по системе, то это значительно улучшает результат.

Принципы разработки программного обеспечения:

- Простота – это хорошо. Стремитесь выбирать простые решения.

- Декомпозиция. Сложное раскладывайте на простое.
- Быстрая обратная связь. Сделайте быстро функционал, получите обратную связь и доработайте его на основании этой обратный связи. Лучше короткие шаги, чем затяжной прыжок.
- Код важнее документации.
- Нет ничего постоянного. Делайте так, чтобы потом было просто менять.
- Главный ориентир – это цели клиента на проект. Помните, зачем вам нужен проект.
- Ничего лишнего. Сравните свой проект с самолетом. Убирайте все лишнее, что не дает пользу проекту.

В следующей главе мы поговорим об этапе сопровождения проекта. После ввода в эксплуатацию проект на самом не деле не заканчивается, а только начинает свое движение в реальной среде. Мы поговорим о развитии функционала проекта, получении обратной связи от пользователей системы.

Глава 5. Сопровождение и развитие CRM

Вы ввели свою CRM-систему в эксплуатацию. Пользователи начали в ней работать. Проект по внедрению закончен.

Наступает этап сопровождения системы, который состоит из следующих действий:

- ➢ Разработка новых функций и модулей системы.
- ➢ Повышение удобства работы с системой.
- ➢ Обслуживание системы.
- ➢ Анализ работы системы.

Рассмотрим их более подробно.

Разработка новых модулей подразумевает развитие системы с помощью команды разработки. Хорошо, если компания-разработчик поддерживает продукт на этом этапе. Если нет – не беда при наличии проработанной документации.

При разработке новой функциональности помните, что вы должны критически подходить к каждой новой функции: действительно ли она так необходима? Как она соотносится общими планами на развитие продукта?Проработав эти вопросы, вы в итоге получите более сбалансированную систему.

Откуда появляются доработки по системе?

Во-первых, это пожелания пользователей и потребности бизнеса. Процессы не стоят на месте, они развиваются, следовательно, должна меняться и ваша система.

Во-вторых, вы можете автоматизировать те бизнес-процессы, которые ваша система ранее не охватывала. Например, сначала вы автоматизировали процесс продажи, затем вам захотелось упростить работу HR отдела, чуть позже – внедрить систему мотивации на основе показателей, вычисляемых из данных, хранящихся в системе.

В-третьих, это может быть исправление ошибок. Не бывает идеального софта. И чем сложнее софт, тем больше он подвержен ошибкам. Ошибки в любом случае будут всплывать по ходу работы системы. И их надо оперативно исправлять.

Повышение удобства работы с системой подразумевает тесное общение с пользователями системы. Если с системой неудобно работать, то это может привести к следующим последствиям:

- возникновение ошибок пользователей;
- замедление бизнес-процессов;
- искажение метрик по процессам;
- репутационные риски, если к вашей системе есть доступ у клиента;
- тихий бунт сотрудников против системы. Они будут вести работу по старинке, а в систему будут добавлять фиктивные данные (для руководства).

Простейшими средствами повышения удобства пользования системой можно считать:

- Скорость работы программы
- «Защита от дурака»
- Возможность отменить некоторые операции
- Минимум кликов и переходов до цели
- Отображение статусов любой операции
- Приятный дизайн
- Четкий фокус на конкретной операции

Попробуйте сами поработать от каждой роли и, вероятнее всего, у вас уже появится список возможных доработок по системе.
Слушайте своих пользователей – они могут дать ценную информацию как улучшить процесс работы с системой.

Обслуживание системы включает в себя следующие работы:

- Оказание технической поддержки пользователей. В процессе работы у пользователей возникают вопросы по работе с системой. Важно оперативно решать такие вопросы. Техническую поддержку обычно оказывают разработчики, сопровождающие проект. Конечно, это не сами разработчики продукта, но это те лица, которые специализируются на решении вопросов по конкретному продукту.
- Обучение пользователей. Новых пользователей необходимо обучать. Если система сложная, то можно и не обойтись только

чтением текстовой документации. Поэтому проводятся семинары по изучению продукта.

- Обслуживание сервера. Эта задача ложится на системного администратора. Основные задачи, которые решает системный администратор (или «сисадмин») это:
- Обслуживание и проверка бекапов
- Мониторинг ресурсов сервера
- Мониторинг доступности приложения
- Профилактические работы на сервере
- Если есть внешние системы, то необходимо оплачивать их услуги. Например, это могут быть СМС агрегаторы, продление домена или аренды сервера и т.д. Рекомендую занести важные события в Google Calendar и настроить уведомления для самых критичных событий на телефон.
- Периодическая проверка работы форм на сайте, общего тестирования, работы рассылок. Рекомендуется хотя бы 1 раз в 2 недели проводить такие работы, потому что при взаимодействии с внешними системами случаются различные сбои.

И, наконец, **анализ работы системы**. Вы анализируете действия пользователей с целью понять как они используют ваш функционал. Возможно, они выполняют свои задачи не так как вы предполагали или какие-то функции игнорируют из-за того, что не знают о них или им просто неудобно с ними работать.

Какие средства использовать для анализа?

Можно просто понаблюдать за работой операторов. Попросите их выполнить простые типовые задачи и оцените насколько хорошо они владеют системой.

Другой вариант – это использование веб-аналитики. Вы ставите специальный код в веб-приложение и отслеживаете действия пользователей в системе. Для этого можно использовать средства общего назначения такие как Яндекс Метрика или Google Analytics, либо использовать встроенные в CRM средства отслеживания действий пользователя.

Как вы понимаете, после ввода в эксплуатацию все только начинается, и именно здесь проявляются основные достоинства выбранного способа – создания CRM под свои требования. Вы можете ее развивать как вам угодно, при этом вы полностью контролируете ход ее совершенствования. С развитием вашего бизнеса, ваша система развивается параллельно, что служит хорошим фундаментом для дальнейшего роста бизнеса.

Заключение

Вот мы и подошли к концу этого небольшого руководства. Если вы не начали свой проект по ходу чтения этой книги, предлагаю вам приступить к написанию концепции вашего проекта.

Мы подготовили небольшой бриф, чтобы упростить задачу составления концепции.
Вы можете найти бриф по адресу (в формате Excel) либо взять из приложения 2 к данной книге.

Остались вопросы по материалу? Задавайте свои вопросы на нашем сайте (http://web-automation.ru) или через профиль в социальной сети (http://vk.com/hecrus). Также вы можете присылать свои концепции для получения обратной связи.

Последнее –Если вам понравилась книга – напишите пожалуйста отзыв http://web-automation.ru/book/how-create-crm/. Ну а если очень понравилась - то сделайте пожалуйста пост в социальной сети с ссылкой на книгу - книга бесплатная и мы не продвигаем ее через какие-то рекламные площадки. Поэтому ваш пост очень поможет доставить эту книгу тем, кому она действительно нужна.
Всего доброго и удачных вам проектов!

Приложение 1. Основные термины и понятия в веб-разработке.

Доступно по адресу www.web-automation.ru/glossary/. Для удобства Вы можете сохранить отдельно этот список (либо на диск, либо в социальной сети).http://www.web-automation.ru/books/glossary/

Авторизация – процесс входа пользователя на сайт (для гурманов – процесс проверки прав на выполнение определенного действия).

Архитектура – документ, определяющий структуру и организацию вашей системы.

База данных - место где хранятся данные веб-приложения (сайта).

Бекап, резервная копия – копия вашей базы данных или приложения. Делается для того, чтобы снизить риски потери важных данных.

Бизнес-процесс – последовательность действий, направленных на определенный результат. Имеет следующий параметры: действия, участники, ресурсы, цель и результат, триггер.

Биржа – сайт, на котором могут встретиться Заказчики и Исполнители.

Браузер – программа, с помощью которой просматривают страницы в интернете.

Бриф – краткая анкета для получения первичной информации по проекту.

Веб-приложение – программа работающая в браузере.

Веб-сервер – сервер, который обрабатывает запросы вашего веб-приложения.

Веб-сервис – веб-приложение, предназначенное для обработки специфических программных запросов. Например, это может быть программное извлечение данных из 1С.

Веб-аналитика – инструмент для изучения поведения пользователей на сайте.

Верстка – способ организации страницы сайта.

Воронка продаж – визуальное распределение базы клиентов по статусам (заявки, лиды, горячие лиды, клиенты).

Дизайн – средство для правильной подачи информации.

Интерфейс – совокупность возможностей, элементов управления и графеских элементов для пользователя или роли. Интерфейс – это способ взаимодействия пользователя с системой.

Конверсия – процент посетителей, достигших определенной цели на сайте к общему числу посетителей. Целью может быть заполнение формы, щелчок на кнопке или переход на определенную страницу.

Контрагент – лицо (или компания), с которой взаимодействует ваша компания.

Контент – текстовое или визуальное содержание вашего сайта.

Концепция – совокупность основных данных по вашей системе (система взглядов на систему).

Логин – идентификатор человека в системе.

Личный кабинет – совокупность страниц, предназначенных для определенной роли в системе. Закрытый раздел сайта.

Макетирование, макет – графическое отображение некоторого объекта (страницы) для более простого его понимания.

Платежный шлюз – система, позволяющая принимать платежи через различные каналы связи (Яндекс Деньги, Веб-мани, Карты, SMS и др.)

Проектирование – процесс решения, как именно внутри будет работать ваша система.

Прототип – частично работающий продукт, направленный на проработку некоторой задачи.

Пользователь – зарегистрированный член вашей системы, имеющий свой логин и пароль.

Рассылка – часть системы, предназначенная для массовой отправки пользователям системы некоторых сообщений по СМС или email.

Роль – совокупность пользователей, имеющих сходный интерфейс работы в системе. У каждой роли есть свои функции и права в системе. Примеры: продавец, оператор, администратор, контролер.

Сайт – программа, которая позволяет отображать информацию в интернете и обрабатывать различные данные.

Синхронизация – процесс обновления данных в двух источниках данных (базах данных) с целью передачи данных из одного в другой.

Скриншот – снимок экрана. Делается при помощи клавиши Print Screen либо с помощью специализированных программ (Яндекс.Диск, clip2net и др.)

ТЗ (Техническое задание) – документ, закрепляющий требования на разработку продукта.

Тонкий клиент – это по сути браузер. Т.е. это программа, которая отправляет запросы на сервер и выводит данные пользователю. На компьютере не надо ничего лишнего ставить, чтобы работала такая система. Именно поэтому такой клиент называется тонким.

Толстый клиент – в отличии от тонкого клиента, для работы пользователя нужно уставить программу на компьютер.

Функционал – перечень функций определенной страницы или модуля.

Хостинг – место на диске, хостинг компании. Это место используется для хранения вашего веб приложения. Является упрощенным вариантом размещения вашего веб приложения.

Юзабилити – область знаний об удобстве использования сайтом.

1С – ERP система, включающая в себя большой комплекс различного функционала (Бухгалтерия, склад и др.)

API – программный интерфейс некоторой системы. Используется для взаимодействия с другими системами через программный код.

ASP.NET – средство для разработки веб-приложений.

arkAS – платформа для разработки веб-приложений, ориентированную на бизнес-среду. Разработана на ASP.NET.

Bootstrap – технология для визуального отображения содержания вашего сайта.

CMS – система управления контентом. Позволяет создавать и редактировать содержимое вашего сайта, не применяя навыки программирования или верстки.

CRM – система по работе к клиентами (в узком смысле).

DEV версия – тестовая версия продукта. Предназначена для тестирования и разработки продукта.

HTML разметка – тело страницы, которая отображается в браузере.

Internet Information Service – веб-сервер от Microsoft.

jQuery - средство для создания интерактивного интерфейса веб-приложения.

Prod версия – основная версия продукта. Предназначена для использования конечными клиентами системы.

SMS агрегатор – платный сервис, позволяющий отправлять СМС.

SQL Server - Система управления базами данных. Разработчик – компания Microsoft.

SSL (HTTPS) – защищенный протокол доступа к секретной информации, обычно используется для доступа к личному кабинету пользователя в системе.

VPS – сервер, на котором работает ваше приложение. Обычно это Windows или Linux. ASP.NET работает только на Windows Server.

Приложение 2. Бриф на создание концепции проекта

Название проекта	
Концепция	
Суть проекта	
Кабинеты (роли) и их функции	
Роль 1	Функция 1
	Функция 2
	Функция 3
Роль 2	Функция 1
	Функция 2
	Функция 3
Технологии	
База данных	
Сервер	
Язык программирования и платформа	
Сроки	
Бюджет	

Приоритеты	
Требования к исполнителю	
Порядок подачи заявки на проект	

Бонус. Памятка по книге «Как создать свою CRM»

Доступно по адресу:

http://web-automation.ru/?p=1362

Для удобства Вы можете сохранить отдельно эту схему (на диск или в социальной сети).