Юрий Дубровский

Как написать бизнес-требования?

Бизнес-специалисту - как разговаривать на одном языке с ИТ

12+

Юрий Дубровский

Как написать бизнес-требования? Бизнес-специалисту – как разговаривать на одном языке с ИТ

«ЛитРес: Самиздат»

2021

Дубровский Ю.

Как написать бизнес-требования? Бизнес-специалисту – как разговаривать на одном языке с ИТ / Ю. Дубровский — «ЛитРес: Самиздат», 2021

Эта книга поможет Вам создать уютный оазис определенности в жестокой пустыне проектной деятельности.Палящие лучи неопределенности, выжигающие все живое вокруг, остановятся, когда на их пути возникнут нормальные, понятные, продуманные бизнес-требования к проекту.Вы внесете в проект атмосферу взаимопонимания и управляемости, если будете руководствоваться изложенными в книге подходами и шаблонами.Ваши контрагенты будут приятно удивлены и станут стремиться только к Вам, потому как Вы станете тем редким "заказчиком, который точно знает, что он хочет".Книга поможет вам задать правильные вопросы и найти ответы на старте проекта, заранее, до того, как все сроки прошли, а работа сделана.И не важно, кто вы - бизнес-специалист или аналитик, в любом случае требования, вышедшие из под Вашего пера после прочтения этой книги будут вызывать диалог по существу, а не ужас в глазах ваших потенциальных исполнителей.Это дорогого стоит! Добро пожаловать в мир правильных бизнес-требований!

Содержание

Юрий Дубровский
Как написать бизнес-требования? Бизнес-специалисту – как разговаривать на одном языке с ИТ

Эта книга посвящается:

Функциональным заказчикам, бизнес-заказчикам нашей ИТ-отрасли, которые своим великим терпением и энтузиазмом двигают эту отрасль вперед, несмотря на то, что разработчики и заказчики живут в параллельных мирах и говорят на разных языках. Без них не было бы этой книги, да и автоматизации бизнеса как отрасли тоже.

Введение

Вы функциональный специалист бизнеса и решили автоматизировать свой бизнес-процесс? Вы уже делали это и знаете по собственному опыту, как порой нелегко донести до разработчика свои потребности? Вы аналитик, которому поручено выяснить бизнес-потребности? Вы разработчик, который хочет понять, что него хочет его заказчик?

Значит Вам нужно определить бизнес-требования.

Именно они станут общим языком, на котором смогут разговаривать разработчик и заказчик.

В этой книге обобщен опыт автора и его коллег по взаимодействию с заказчиком в крупных и небольших проектах, осмыслены ошибки и их последствия и выстроен подход к написанию бизнес-требований, который позволяет минимизировать непонимание заказчика и разработчика.

Это приводит, к лучшим из возможных результатов проекта с точки зрения удовлетворения потребностей заказчика.

Материал этой книги – попытка поделиться опытом и его систематизация, что всегда полезно. Это дает уверенность, что в какой бы роли вы ни вступали, описанные в этой книге подходы и примеры расширят ваш профессиональный кругозор, придадут больше системности вашим знаниям вопроса. И даже дадут повод задуматься и пересмотреть что-то в своей повседневной деятельности.

Вместе мы создадим еще более профессиональную и эффективно помогающую бизнесу отрасль ИТ!

Благодарим за выбор этой книги и желаем интересного чтения!

ЧАСТЬ I. Что такое бизнес-требования и зачем они нужны

Глава 1. Как начинается разработка

Вместо эпиграфа

Заказчик на объекте принимает работу у подрядчика. Заказчик смотрит: вырыта глубокая узкая круглая яма, на дне которой горит прожектор. Он гневно спрашивает:

– Что за чёрт?!

Подрядчик:

– Всё сделано по чертежу, вот, посмотрите.

Заказчик, переворачивая чертеж вверх ногами:

– Это должен был быть маяк!

Все знают этот старый анекдот. Но вот вопрос, почему подрядчик копал яму по чертежу, и даже не подумал его перевернуть? Хочется ответить, что просто неграмотный и не перевернул чертеж.

Да, возможно, это и так. Ну, а, если ошибся чертежник и переворачивать чертеж по правилам не было нужно? Почему возможна такая дикая нестыковка с ожиданиями заказчика?

Ответ простой – подрядчик исполнял задание, не задумываясь о бизнес-задачах заказчика! Он просто не знал, зачем заказчику то, что он делает. Возможно, заказчик собирался указывать путь кораблям? Но это предположение. Может быть, маяк ему нужен был совсем для другого, и тогда нужно было бы строить совсем другой маяк, или вообще не маяк?

Подрядчик всего этого не знал и не спросил, вот и получился анекдот вместо результата.

К сожалению, непопадание в ожидания и потребности заказчика – одна из животрепещущих проблем создания ИТ-решений. Такие промахи в состоянии погубить проект целиком, испортить репутацию и расшатать нервы всем участникам.

Давайте посмотрим, почему это происходит и как это можно улучшить.

Как начинается разработка

Как начинается разработка ИТ-решения?

Обычно у кого-то из ответственных лиц возникает идея «пора бы это автоматизировать». Подтолкнуть к этому жизнь может с разной стороны, например, побуждающими факторами могут быть:

– Конкуренция и то, что у других компаний данный процесс работает лучше.

– Время на принятие решений и выполнение операций превышает желаемое.

– Оптимизации и прочие внутрикорпоративные движения повышения эффективности.

– Желание стать более современным и цифровизированным подразделением.

– Хайп на рынке, влияние внешнего маркетинга.

– Вера в чудесное новое решение всех проблем.

– Новости законодательства и отраслевых регуляторов.

– Воля вышестоящего руководства.

Кроме того, может быть и множество других причин. Здесь важно, что появляется воля лица, принимающего решения, которая может быть выражена в выделении на создание ИТ-

решения некоторого количества ресурсов (финансовых, временных, экспертных и др.), что должно привести к автоматизации какого-то участка бизнеса и изменить показатели, характеризующие этот участок, соответственно ожиданиям.

Появляется критерий «соответствие ожиданиям», то есть:

– ожидания должны быть сформированы;

– ожидания должны быть формализованы до метрик или категорий, которые позволили ли бы определить меру соответствия ситуации этим ожиданиям;

– для метрик и категорий определены целевые значения.

В реальности так бывает редко. Обычно ожидания сродни мечте или иллюзорному образу «светлого будущего». При этом воля (или драйв, как нынче модно говорить) имеется, а, значит, есть и движение к мечте.

Особенность мечты, как мы все знаем из многочисленных трудов по мотивации, в том, что пока не выстроен более-менее измеримый и непротиворечивый образ этой мечты, движение носит хаотический характер, а шансов достичь цели немного.

Более того, велик шанс не понять, что цель уже достигнута и разочароваться. Увы, все тоже самое, часто с куда большими затратами, верно и для создания ИТ-решений.

Чем более размыты и абстрактны ожидания, тем меньше шансов на удовлетворенность результатами проекта.

Итак, ключевые факторы начала разработки это:

– Желание, или даже мечта, драйв;

– Возможность привлечь для создания решения необходимые ресурсы;

– Понимание целей того, что собрались сделать, и вера в бизнес-полезность этого.

Пока мы говорили о желании, мечте, но всем известно, что разработка ведется на основе требований. И если мечты и желания исследуются поэтами, писателями, психологами, то требования – область деятельности аналитиков, менеджеров продуктов и разработчиков.

Немного остановимся, на том, что такое требование и каковы его свойства.

Глава 2. Немного о требованиях

Основные виды требований

Детально требования и все аспекты работы с ними проработаны книгах по управлению требованиями, например, Карла Вигерса и соавторов. Приведем необходимый для дальнейшего изложения минимум, а за остальным предлагаем обратиться к упомянутой классике.

Итак, выделяются следующие виды требований:

Бизнес-требование – высокоуровневая бизнес-цель организации или заказчиков системы, в то же время под бизнес-требованием может пониматься достаточно детализированное требование любой другой категории, если его выполнение прямо связано с достижением бизнес-цели заказчика, а отклонение от него ведет к не достижению или неполному достижению этой цели. Обычно, говоря о подготовке качественных бизнес-требований, имеют в виду эту расширенную трактовку.

Бизнес-правило – политика, предписание, стандарт или правило, определяющее или ограничивающее некоторые стороны бизнес-процессов. По своей сути это не требование к программному обеспечению (ПО), но оно служит источником нескольких типов требований к ПО.

Ограничение – ограничение на выбор вариантов, доступных разработчику при проектировании и разработке решения

Требование к взаимодействию – описание взаимодействия с пользователем, другой программной системой или устройством.

Функциональное требование – описание требуемого поведения системы в определенных условиях. Функциональные требования описываются в форме традиционных утверждений со словами «*должен*» или «*должна*».

Нефункциональное требование – описание свойства или особенности, которым должна обладать система, или ограничение, которое должна соблюдать система.

Среди нефункциональных требований обычно выделяют:

Требования по удобству использования – достаточно сложно формализуемый пункт, но часто важный, например, «размещение заказа из корзины должно выполняться не более, чем двумя кликами мыши».

Требования к производительности – нагрузочные характеристики системы по количеству пользователей, запросов, профилю нагрузки, объему данных и т.п.

Требования к безопасности – все, что касается обеспечения информационной безопасности и защиты информации, включая требования по криптографической защите.

Требование к доступности и надежности – необходимые параметры, определяющие доступность системы, простои (например, по рабочим дням с 9:00 до 18:00 московского времени, с не более, чем двумя перерывами обслуживания в день продолжительностью до 5 минут), резервирование данных и вычислительных средств.

Далее покажем основные пути, как из мечты или представлений формируются требования.

Способы сбора требований

Источников требований к ИТ-решению не так много. Вот основные из них:

– Лицо, принимающее решения по проекту и эксперты его команды, то есть те, кого абстрактно именуют заказчиком. Они имеют мечту и представление о том, чего хотят добиться ИТ-решением и потому являются основным источником требований.

– Нормативно-регулятивное окружение бизнеса. Это законодательство, регуляторные требования, обычаи делового оборота, внутрикорпоративные и даже этические нормы, в пределах которых должно строиться ИТ-решение.

– Ресурсно-технологическое окружение бизнеса. Это наличие и возможность использования ресурсов, технических средств, технологий, которые могут быть применены для ИТ-решения.

Среди традиционных способов сбора требований упомянем такие как:

– Проведение фокус-групп типичных пользователей.

– Работа с пользователями для выяснения назначения продукта.

– Проведение интервью для выявления требований.

– Проведение совместных семинаров.

– Наблюдение за пользователями на рабочих местах.

– Разработка и раздача опросных листов.

– Анализ документов.

Здесь важно отметить, что многое можно узнать анализом документов, однако их актуальность, понимание, трактовка, практика применения и степень влияния определяется лицами, принимающими решения, и экспертами команды.

Если говорить о ведении разработки в зависимости от формы требований, стоит выделить следующие подходы:

– со слов, без документирования;

– с использованием спецификаций требований, то есть с предварительным документированием и согласованием требований до разработки;

– гибкий итерационный подход (Agile), когда требования формируются с минимальной формализацией, последовательной реализацией, уточнением и доработкой.

Ознакомимся детальнее с каждым из перечисленных подходов.

Разработка «со слов»

Желание начать разработку немедленно, не тратя времени и ресурсов на подготовку, потребность обойти конкурентов, сократив до минимума время, проходящее от идеи до работающего ИТ-решения – все это породило разработку «со слов». Сюда же добавляется отношение к документации, как пережитку забюрократизированных старорежимных контор.

Так в чем же сильные и слабые стороны такого подхода? Какие условия необходимы для достижения успеха проекта при его реализации?

Сначала определим, что такое разработка «со слов». Это метод ведения разработки ИТ-решений, когда не разрабатывается никаких технических спецификаций требований в письменном виде (схемы и эскизы – тоже письменные документы), а все необходимые требования излагаются разработчику заказчиком устно.

Как это может выглядеть? Например, так: заказчик вызывает к себе разработчика и рассказывает ему, что хотел бы видеть в ИТ-решении. Разработчик его выслушивает, возможно конспектирует, и после этого начинает разработку.

Важно, что этапа согласования письменных требований нет – как понял разработчик, так и реализует.

Это может быть реализовано в рамках небольшой компании или функционального подразделения корпорации, когда ключевое лицо заказчика и разработчика работают в нем.

Возможно, что разработчик и заказчик настолько глубоко погружены в бизнес-процесс, что понимают друг друга с полуслова, а, может быть, наоборот, совершенно не понимают друг друга. Возможны, конечно, и не столь крайние вариант – в любом случае, получив информацию устно, разработчик будет руководствоваться ею сразу как требованиями, минуя согласование их с заказчиком.

Преимущества метода очевидны – скорость перехода от идеи к реализации, отсутствие для заказчика необходимости глубоко продумывать и детализировать свою идею, отсутствие трудозатрат на разработку и документирование требований.

Также ценным является ощущение отсутствия потери информации, передаваемой заказчиком разработчику – что рассказал, то и идет сразу в разработку, а не перерабатывается в требования. Поэтому кажется, что информация не может быть искажена по ходу переработки.

Ценна и возможность оперативной обратной связи разработчика с заказчиком – можно просто быстро устно спросить, получить ответ и сразу брать его в разработку.

Недостатки не столь очевидны, но они есть:

– существенные искажения, если они случились при восприятии разработчиком требований, будут выявлены только при демонстрации результатов разработки. То есть высок риски доработок и переделок (причем многократных, так как искажения возможны и в восприятии замечаний);

– перенос единолично на разработчика задачи детализации требований и разрешения противоречий, существующих в требованиях, так как не делается детальной проработки и согласования требований с заказчиком;

– отсутствие возможности восстановления истории требований. Это негативно влияет на ведение изменений, может привести к «хождению по кругу», когда одно и то же требование то возникает, то пропадает в последовательных правках решения;

– практическая сложность и высокая стоимость поддержки и развития как сторонними разработчиками, так и самим автором по истечении времени. Это произойдет потому, что требования, под которые построена система, забудутся или просто будут неизвестны новому разработчику, принятые технические решения и их обоснование, известное только первоначальному разработчику, будет непросто понять и развить (в том числе, возможны ошибочные представления, ведущие к нарушению работоспособности ИТ-решения при попытках его модификации и развития);

– неконтролируемость движения к результату, невозможность планирования на основе исполнения требований, декомпозиции процессов и частей системы. Невозможно составить письменный план работ по исполнению требований, если нет письменно зафиксированных требований;

– потребность в многократном объяснении запутанных моментов со стороны заказчика при доведении информации до разработчика – излишние временные затраты и все тот же риск недопонимания и искажения восприятия;

Таким образом, решение разрабатывать «со слов» целесообразно для небольших задач без необходимости сопровождения и развития, выполняемых в хорошо сработанных командах, где заказчик и разработчик отлично понимают детали бизнес-процесса и идеи друг друга с полуслова.

В иных случаях шанс достижения цели разработки «со слов» без разочарований невелик.

Письменная спецификация требований к разработке

Другой крайностью является старт проектирования и разработки только по окончании детальной проработки требований, оформленных и согласованных письменно, в виде документа.

Называться он может по-разному, в зависимости от принятых у заказчика или разработчика стандартов. Это могут быть и бизнес-требования, и требования к системе, и программная спецификация – перечень не полный, встречаются разные креативные варианты.

Первое ощущение, которое возникает при описании такого подхода – долго, нудно и, скорее всего, дорого. И это ощущение имеет под собой реальную почву. Так почему же и в каких случаях имеет смысл использовать такой подход?

Что дает нам письменная согласованная специалистами заказчика спецификация требований (пропустим про спокойный сон руководителя проекта, поскольку у него юридически ограничены рамки проекта, хотя это тоже есть)?

Хорошая спецификация дает нам предварительно построенную достаточно детальную и непротиворечивую картину, что же мы должны построить.

При этом еще и определены приоритеты, как это строить, проработаны и сняты основные нестыковки, с которыми при построении решения столкнется разработчик.

То есть, хорошая спецификация существенно повышает шансы достигнуть цель разработки в плановые сроки и бюджет. Также проще планировать путь и ресурсы исполнения проекта.

Здесь важно отметить, что речь идет о хорошей спецификации, а не о томах сказок и легенд, которые нередко выдаются за бизнес-требования. Только согласованные, достоверно отражающие реальность, однозначно толкуемые, проверяемые и непротиворечивые требования, определяющие искомую цель автоматизации и изложенные в спецификации обладают названными достоинствами.

Некоторые недостатки подхода очевидны – сравнительно долго и трудозатратно – нужно время, чтобы формализовать замысел, записать и оформить требования, согласовать их. Это явно дольше, чем просто рассказать.

Из неочевидного – если разработчик неправильно поймет спецификацию или она по содержанию не несет в себе требований, то может возникнуть опасная иллюзия, что все под контролем и работа идет к цели по плану, хотя на самом деле разработка ведется даже не «со слов», а из предположений разработчика как это могло бы быть в спецификации.

Письменные спецификации требований характерны для сложных проектов с широким охватом бизнес-процессов, интеграционными взаимодействиями, длительным жизненным циклом решения и, соответственно, сроком сопровождения. Применяются чаще в крупных организациях, склонных к тяжелым, например, водопадным, методологиям внедрения.

Еще один момент – время разработки спецификации. Мир сейчас меняется быстро, и бизнес-среда тоже, поэтому время создания решения, в том числе требований к нему, часто критично.

Проработка хорошей спецификации требований требует времени, которого может и не быть – это следует сопоставить с бизнес-окружением, конкурентами, сроком жизни разрабатываемого решения и вовремя остановиться с детализацией.

Случаи, когда требования к моменту завершения их подготовки уже устарели – увы, обычная история в современном корпоративном мире.

Поэтому хорошая спецификация требований – отличное подспорье проекту разработки ИТ-решений, но нужно чтобы она была действительно хорошей и завершалась до того момента, когда ее содержание устареет.

Agile-подход

Нужны ли письменные спецификации в Agile? Недальновидные поклонники методологии думают, что ее придумали, как отказ от документирования. Гуру указывают, что они такого

не замышляли, и вот почему. Посмотрим на Agile манифест, широко известный в русском переводе, где говорится, что:

– Люди и взаимодействие важнее процессов и инструментов.

– Работающий продукт важнее исчерпывающей документации.

– Сотрудничество с заказчиком важнее согласования условий контракта.

– Готовность к изменениям важнее следования первоначальному плану.

То есть, не отрицая важности того, что справа, мы всё-таки больше ценим то, что слева.

Все заявления затрагивают требования, попробуем понять, как.

«Люди и взаимодействие важнее процессов и инструментов» говорит нам о том, что нужно выбирать тот объем и характер взаимодействия специалистов заказчика и разработчика, который позволил бы им максимально быстро и комфортно сформировать единое мнение о том, какое ИТ-решение нужно разработать. Для хорошо сыгранной команды и простого решения это может быть разработка «со слов», для множества распределенных команд и сложной корпоративной системы – объемная согласованная стандартизованная документация. Именно выбор людьми команды подходящего ситуации инструмента и процесса декларируется здесь.

«Работающий продукт важнее исчерпывающей документации» не отрицает документирование, а лишь указывает на ее вторичную, поддерживающую роль. Пока продукт работает корректно и не развивается, принцип может трактоваться как отсутствие документации. Как только заходит речь о поддержке и развитии – сразу требуется наличие минимально необходимой документации, в которую обычно входят и бизнес-требования «для чего и почему именно так мы действуем».

«Сотрудничество с заказчиком важнее согласования условий контракта» говорит о готовности к изменениям даже если законтрактованы жесткие требования. Увы, жизнь сложнее и многообразнее любой спецификации требований, однако контрактные рамки позволяют цивилизованно и к обоюдной выгоде разрешать такие отклонения. Документирование требований, в частности, являющееся приложением к контракту, эти рамки создает.

«Готовность к изменениям важнее следования первоначальному плану» указывает, как и предыдущее, что изменения неизбежны и к ним нужно быть готовыми. Тем не менее, чтобы изменить первоначальный план, его нужно сначала выстроить. А если изменять планы и не отслеживать (документировать) отклонения, вполне возможно начать ходить по кругу, зациклиться в изменениях, и цель не будет достигнута.

Таким образом, Agile придерживается позиции постоянной готовности к изменениям ради движения к цели в целом, и, для обеспечения этого, предлагает использование разумного объема документирования, исходя из текущей ситуации (и этот объем может меняться по ходу работ вместе с ситуацией).

Agile позволяет совместить преимущества ранее описанных подходов «со слов» и письменной спецификации, устранив часть недостатков.

Однако, существенным ограничением применения Agile является существование факторов, влияющих на него, которые невозможно изменить в ходе проекта. Это могут быть как внутренние ограничения, например, размер бюджета или нормативные акты внутри компании, так и внешние, например, требования законодательства.

Выявление таких требований на поздних этапах разработки может привести к необходимости начать ее полностью или в значительной части с начала, что никак не приближает к цели, поэтому «полный Agile», исключающий такие факторы из рассмотрения может быть успешен лишь на небольших обособленных задачах.

Наш опыт свидетельствует, что оптимальным является подход письменной спецификации общей структуры решения с последующими Agile итерациями в реализации отдельных задач. В этом случае минимизируется риск не учета в требованиях существенных факторов,

влияющих на процесс, но сохраняются преимущества и гибкость Agile подхода при решении частных задач.

Когда же разумно начинать разработку?

Привести формальный критерий начала разработки и просто, и сложно одновременно. Кажется, что чем быстрее начнем разработку, тем быстрее придем к цели.

И это было бы верно, если бы путь к цели был фиксирован, а его длительность и затраты не зависели от исходной постановки требований. Увы, зависимость есть, и она не линейна. Вплоть до того, что при существенных неточностях можно двигаться кругами сколь угодно долго.

Поэтому нужен какой-то критерий. Сформулировать его нам поможет представление о рисках проекта, которое поверхностно сводится к тому, что:

– выполняя какую-то работу для снижения рисков, мы тратим дополнительно ресурсы и время, но верим, что возможные затраты ресурсов и времени на ликвидацию последствий срабатывания риска существенно выше, а вероятность срабатывания не позволяет риском пренебречь.

– не делая ничего для снижения риска, мы понимаем и принимаем необходимые затраты ресурсов и времени на устранение последствий срабатывания риска с учетом его вероятности и готовы в случае срабатывания эти затраты понести или смириться с тем, что проект не достигнет цели. Причины могут быть самые разные, начиная от реальной невозможности что-то сделать и до обычной лени и самоуверенного игнорирования риска, но важен результат – не делаем.

Начать разработку означает «начать что-то делать» для снижения риска «не разработать никогда, потому что не начали», а этот риск реален. Для проекта этот риск выглядит скорее с временным ограничением – «не успеть разработать вовремя».

Не начать разработку, но уточнять требования, означает «делать что-то» для снижения риска «разработать не то, что нужно, но потратить ресурсы и время».

В каждый момент можно оценить эти риски, оценить затраты ресурсов и времени на выполнение шага «начать разработку» и шага «не начинать пока, продолжить уточнение требований» и принять вполне взвешенное решение, пора или еще нет.

Интересный факт состоит в том, что по мере разработки и погружения в детали оба риска будут меняться в своих оценках. Натыкаясь на «грабли» по ходу разработки будет вырастать риск «не успеть разработать вовремя», а риск «разработать не то» от приостановки и уточнения будет снижаться. Наоборот, если все оказалось вполне понятно, риск «не успеть разработать вовремя» не будет расти по ходу разработки. Таким образом, оба риска стоит переоценивать по ходу проекта и, если выявляются неясности, принимать взвешенные решения, вплоть до приостановки разработки до уточнения требований, если вероятность «разработать не то» становится угрожающе высокой.

Таким образом, точкой начала разработки является такой момент, когда затраченные на ближайший шаг разработки время и ресурсы приемлемы по результату и ожидаемой его полезности для достижения цели. При этом полезность может быть разная, например:

– Получение части функций решения (непосредственно приближает к цели);

– Получение прототипа или макета для обсуждения и принятия более обоснованных требований (приближает к цели, но часть затрат может оказаться «выброшенной» – придется переделать, речь идет о прототипах и макетах, но полученная информация приближает проект к цели быстрее и точнее альтернативных вариантов);

– Апробация технологических решений (снижает технологические риски не достижения цели, уточняет возможность и соответствие показателям назначения используемых технологических решений, алгоритмов, методик).

Рекомендация по моменту начала разработки может быть сформулирована так: как можно раньше, но, когда уже осознана и формализована польза для достижения цели от ближайшего шага разработки, взвешены, соизмерены с этой пользой и приняты затраты ресурсов времени на этот шаг.

Ясно, что такой подход наводит на мысль о целесообразности начинать с малых по затратам шагов, дающих максимальное продвижение.

И это действительно так – пока неопределенность высока, лучше быстрыми короткими продвижениями уточнить потребности по ходу разработки, чем завязнуть в сложных деталях, имея большой риск переделки.

Глава 3. Зачем бизнес-требования?

Строить или перестраивать?

Мне нравится подход, восходящий к ТРИЗ (Теория решения изобретательских задач, разработанная Г.Альтшуллером в середине 20 века), который предполагает, что идеальное решение задачи, когда она сама себя решает. В рамках этого подхода при появлении любого нового компонента стоит спросить себя, а зачем он? Нельзя ли прийти к цели без него?

Зададим этот вопрос к бизнес-требованиям. Нельзя ли получить решение без них? И правильный ответ будет «да, это возможно».

Так зачем же нам тратить усилия на них? Все дело в затратах и вероятности. Как и в когда-то популярной вероятностной задаче о вероятности того, что обезьяна, сидящая за пишущей машинкой, напечатает роман «Война и Мир», в нашем ответе также нужно указать вероятность того, что полученное решение без требований совпадет с задачей.

Вероятность эта может быть разной. Например, в случае типового отраслевого решения совпадение с задачей весьма вероятно даже без детальной постановки. В случае же достаточно уникального и сложного процесса такое совпадение маловероятно.

Еще важным фактором является желание и способность бизнеса изменить свои процессы под имеющееся ИТ-решение. Если бизнес проявляет существенную гибкость в этом, то вполне возможен вариант использования типового решения с типовыми его настройками и «натягиванием» на него, как на каркас, бизнес-процессов.

Такое решение обычно существенно дешевле по стоимости внедрения, но может нести риски потери части бизнес-процессов, ухода персонала при переучивании на новые процессы (изменения не всем по душе) и утрате каких-то важных «ноу-хау», запрятанных в процессе и составлявших конкурентные преимущества.

Таким образом, требования тем более нужны, чем сильнее нам требуется отступить от имеющегося типового решения. Фактически, при использовании типового решения, мы соглашаемся с требованиями, заложенными в нем (даже если они формально не описаны или мы о них не знаем), и не разрабатываем свои.

Вопрос, нужны ли требования, еще и сводится к тому, хотим ли мы максимально быстро построить целевое решение, или готовы постепенно и последовательно приближаться к целевому, перестраивая и перестраивая?

Вы спросите: «Будет ли кто-то в здравом уме подписываться на многократную перестройку решения без уверенности в достижении цели?» – и будете правы, что это странный выбор. Но есть обстоятельства, когда такой выбор разумен:

– это исследования и уточнения требований, когда в некоторый момент можно просто остановиться, сказать себе «стоп, теперь нам все понятно» и приступить к разработке решения с понятными требованиями или просто получить необходимые выводы и остановиться (например, CusDev прототипы);

– когда задача кратковременна и ее как-нибудь, но удается решить, а требования собрать равносильно решению задачи (часто так выглядят задачи миграции данных из долго эксплуатируемых, существенно измененных и не очень задокументированных систем).

Завершая этот раздел, отметим, что требования нужны и есть они всегда. Они могут быть уже готовы до нас и «упакованы» в типовое решение, или разработаны нами с разной степенью детальности. Совсем без них не получится, надо же знать, какую цель мы хотим достичь.

Проект или прототип?

Любое дело (конечно, не приводящее к разрушениям) можно делать, по крайней мере, двумя способами:

– сначала подумать, потом сделать;

– сначала сделать потом посмотреть на результат, подумать и переделать.

Разработка программного обеспечения не является исключением, поэтому есть способ разработки с предварительным проектированием (сначала подумать), и с прототипированием (сначала сделать, потом поправить).

Не впадая в крайности, отметим, что на практике применяется сочетание способов: прототипирование начинают, когда частично понятно, что делать, но дальнейшие размышления не позволяют быстро продвинуться в разрешении оставшихся неясностей.

Конец ознакомительного фрагмента.