

12+

Кирилл Ледовский

ТОМ

4

ERP внедрили. Что дальше?

От работающей ERP к цифровой реплике
и вычислимой модели предприятия



МОЯ ERP ДЛЯ АНАЛИТИКА

Кирилл Ледовский

ERP внедрили. Что дальше?

«Издательские решения»

Ледовский К.

ERP внедрили. Что дальше? / К. Ледовский — «Издательские решения»,

ERP внедрили. Что дальше? Этот вопрос возникает у любой команды после запуска системы в промышленную эксплуатацию. Формально ERP уже работает, документы проводятся, пользователи вошли в систему, отчёты строятся. Но означает ли это, что предприятие действительно получило управляемую цифровую среду?

Содержание

Введение	6
I. Устав проекта, который неожиданно оказался технологией	10
II. Сначала нужно понять, чем именно управляет предприятие	14
Конец ознакомительного фрагмента.	19

ERP внедрили. Что дальше?

Кирилл Ледовский

© Кирилл Ледовский, 2026

ISBN 978-5-0071-3324-1

Создано в интеллектуальной издательской системе Ridero

Введение

Когда-то, много лет назад, значительная часть моих публикаций на Infostart так или иначе вращалась вокруг довольно простого вопроса: **«Купили ERP — что дальше?»** Предприятие приобрело большую систему, руководство ждёт от неё результата, проектная команда собрана, консультанты пришли, пользователи продолжают работать — и в этот момент выясняется, что сама покупка программного продукта практически ничего не решает. Нужно понять, что именно внедрять, как организовать проект, как обследовать предприятие, где заканчивается методология и начинается автоматизация, кто принимает решения и каким образом всё это в конечном счёте должно превратиться в работающую систему.

С тех публикаций прошло много лет. За это время через нашу практику прошло больше сотни проектов — очень разных по масштабу, состоянию предприятия, отрасли, составу команды и исходной культуре управления. Были действительно удачные проекты, после которых оставалось ощущение серьёзного профессионального продвижения. Были проекты, где отдельное решение становилось настоящим прорывом и потом переходило в следующие внедрения. Были и совершенно другие истории, когда месяцами двигаешься, проводишь совещания, выпускаешь документы, закрываешь этапы, а потом честно признаёшь самому себе, что по существу находишься примерно в том же месте, с которого начинал.

Это не научная статистическая выборка, и я не пытаюсь её так называть. Но когда за спиной уже не несколько кейсов, а больше сотни проектов, постепенно начинаешь отличать случайность конкретного внедрения от проблемы, которая повторяется при разных заказчиках, командах и информационных системах. Если один и тот же вопрос возвращается снова и снова, значит, дело, скорее всего, уже не в особенностях отдельного предприятия. Возможно, чего-то не хватает в самой технологии проекта.

Именно так на протяжении многих лет у меня накапливались заметки. Практически из проекта в проект я записывал вещи, которые хотелось потом обдумать: почему здесь получилось, а там нет; почему одна и та же методика в одной команде дала результат, а в другой превратилась в ритуал; почему прекрасно описанный процесс неожиданно ничего не объясняет программисту; почему после нескольких месяцев обследования новый специалист снова начинает задавать заказчику те же самые вопросы; почему система работает, но через несколько лет никто уже не может уверенно объяснить происхождение конкретной настройки или доработки.

Отдельная история — чужие методологии. В реальных проектах далеко не всегда работаешь так, как считаешь правильным сам. Иногда методология приходит от заказчика, иногда от крупного партнёра, иногда она закреплена корпоративным стандартом, а иногда просто считается обязательной потому, что «так принято». И здесь у тебя два варианта: формально выполнить требования или попытаться действительно понять, зачем вся эта конструкция существует и в каком месте она должна давать результат.

Я обычно выбирал второе. Где-то после такого разбора чужая технология становилась понятнее и в ней обнаруживались действительно сильные решения. Где-то, наоборот, становилось видно, что красивый документ отлично существует сам по себе, но почти не связан со следующим этапом проекта. А какие-то вещи, первоначально казавшиеся избыточными, начинали выглядеть совершенно иначе после нескольких неудачных собственных попыток решить ту же проблему более простым способом.

Наверное, поэтому мне всегда было трудно принять методологию просто как набор обязательных документов. Если я не понимаю, какой следующий результат становится возможным благодаря этому документу, у меня довольно быстро возникает вопрос: зачем мы вообще его делаем?

Постепенно отдельные наблюдения начали соединяться. Оказалось, что многие проблемы, которые на проекте выглядят совершенно разными, имеют похожее происхождение. Процесс описан, но непонятно, какой предмет в нём изменяет состояние. Требование сформулировано, но невозможно восстановить его профессиональное основание. Доработка реализована, но неясно, какой именно доказанный разрыв типовой функциональности она закрывает. Тест выполнен, но непонятно, что именно этим тестом доказано. Роль назначена, но неизвестно, каким действием над каким результатом она обладает. Проект закончен, но через несколько лет его причинная конструкция снова рассыпалась между документами, электронными таблицами, настройками, программным кодом и памятью нескольких специалистов.

Из этих вопросов постепенно и выросла собственная методология, а затем и технология ERP-проекта. Не в один момент и не из одной красивой идеи. Скорее наоборот: большинство её элементов сначала появлялось как решение вполне конкретной практической проблемы, потом проверялось в других проектах, что-то отбрасывалось, что-то перестраивалось, а отдельные решения спустя годы неожиданно соединялись между собой.

За последние годы я начал приводить весь этот накопленный материал в порядок уже как единую работу. Пришлось возвращаться к старым проектным записям, схемам, методическим материалам, решениям и собственным выводам, отделять то, что было связано только с обстоятельствами конкретного внедрения, от того, что воспроизводилось в разных условиях. Постепенно из этого выросла довольно большая монография, в которой удалось связать в одну траекторию Устав проекта, обследование, предметную инженерию, профессиональные модели, прикладную реализацию, проверку, разработку, миграцию, переход, сопровождение, граф, цифровую нить и цифровую реплику предприятия.

И когда эта конструкция наконец стала видна целиком, первоначальный вопрос изменился.

Раньше он звучал так:

Купили ERP — что дальше?

Теперь мне гораздо интереснее другой:

ERP внедрили. Что дальше?

Это уже принципиально иная исходная точка. ERP работает. В ней существуют пользователи, данные, интеграции, настройки и доработки. Через неё проходят реальные хозяйственные события. Вокруг неё за годы сформировались инструкции, организационные привычки, ручные компенсаторы, неформальные правила и большой объём профессионального знания. Но вместе с этим внутри системы и вокруг неё накопилась история решений, происхождение которых далеко не всегда удаётся восстановить.

Поэтому нынешний вопрос «что дальше?» для меня означает не просто «какой модуль внедрять следующим» и не «какую новую технологию теперь подключить». Сначала нужно научиться **приводить в порядок уже существующее состояние предприятия**: восстановить, чем именно оно управляет, какие предметы существуют в его деятельности, в каких состояниях они находятся, какие процессы меняют эти состояния, какие профессиональные нормы ограничивают действия, какие решения реализованы в ERP, почему они появились и какими доказательствами подтверждаются.

На первый взгляд это похоже на ретроспективную работу. Мы как будто смотрим назад, восстанавливаем происхождение системы и разбираемся с накопленным наследием. Но именно здесь находится главный поворот всей конструкции. Мне интересна эта работа не потому, что я хочу создать идеальный архив прошлого.

Она нужна для того, чтобы **получить возможность двигаться дальше**.

Если предприятие сумело восстановить собственную предметную и причинную модель, связать её с фактической ERP, научилось сохранять основания решений, состояния, зависи-

мости, полномочия и доказательства, тогда эта модель начинает работать уже не только как объяснение того, почему система сегодня устроена именно так.

Она постепенно становится средой, в которой можно исследовать изменения до того, как они произошли в реальном предприятии.

Сначала нужно восстановить причинность прошлого и управляемость настоящего, чтобы получить право серьёзно говорить о вычислимом будущем.

Отсюда и возникает дальняя траектория этого текста. Мы начнём с вещей довольно привычных — Устава проекта, обследования, бизнес-процессов, профессиональных ролей, ERP-объектов. Затем постепенно выяснится, почему этого недостаточно и зачем понадобились бизнес-предметы, состояния, точки передачи, профессиональные модели, доказательный GAP, исполняемые сценарии, цифровая нить и граф.

А в конце вопрос окажется уже совсем другим: если мы действительно знаем текущее состояние предприятия, понимаем причинные связи, видим ограничения и допустимые переходы, можем ли мы исследовать **множество возможных будущих состояний** и заранее смотреть, какие решения сохраняют жизнеспособность предприятия?

Именно поэтому в названии стоит не «как правильно внедрить ERP», а **«ERP внедрили. Что дальше?»**

Ретроспектива здесь необходима, но она не является конечной целью. Нам приходится вернуться назад и привести в порядок проектную причинность только для того, чтобы затем развернуть модель вперёд.

Наверное, это и есть главный вопрос, который за последние годы стал для меня гораздо интереснее самого внедрения. Хорошо, систему мы запустили. А дальше предприятие опять будет годами накапливать решения, происхождение которых постепенно забудется? Или мы всё-таки можем научиться использовать уже созданную ERP как одну из основ следующего уровня управления?

На этом месте и возникла идея сделать **автоконспект собственной монографии**.

Сначала само сочетание показалось мне немного странным. Зачем автору конспектировать самого себя? Но оказалось, что автоконспект позволяет сделать совсем другую работу. Монография должна последовательно строить технологию, вводить понятия, показывать происхождение решений, разбирать детали и удерживать довольно большую архитектуру. Здесь же я могу пройти тот же маршрут значительно быстрее и остановиться прежде всего в тех местах, которые лично для меня оказались профессиональными поворотами.

Поэтому этот текст я воспринимаю ещё и как **саморефлексию над собственной технологией**. Я как будто заново прохожу то, что уже написал подробно, и спрашиваю себя: почему именно здесь когда-то возникло это решение? Из какой проектной проблемы оно выросло? Что сегодня я считаю в нём самым важным? Где несколько разных многолетних наблюдений наконец соединились в одну конструкцию?

Отсюда и немного необычный способ изложения. Основной текст — собственно автоконспект. *Курсивом я иногда оставляю непосредственную профессиональную реакцию*: не восхищение самим собой, а тот внутренний комментарий, который возникает, когда в собственной уже собранной конструкции вдруг узнаёшь старую проектную проблему или понимаешь происхождение решения, которое когда-то появилось почти интуитивно.

Отдельными вертикальными блоками я выношу наиболее жёсткие формулы. Это не академические цитаты и не попытка придать тексту искусственную значительность. Просто некоторые выводы полезно отделить от окружающего рассуждения, чтобы потом к ним можно было быстро вернуться.

Когда-то на Infostart оставались мои старые публикации, и в этом смысле нынешний материал можно рассматривать ещё и как своеобразную контрольную точку. При желании можно поднять старые записи и посмотреть, с каких вопросов всё начиналось. Потом меня на пло-

щадке довольно долго не было. А теперь я возвращаюсь уже с другим вопросом и могу честно показать: **вот до чего мы дошли за эти годы и вот так сегодня я действительно делаю ERP-проекты.**

Мне не хочется превращать этот текст в заявление о том, что найден единственно правильный путь. Скорее я предлагаю коллегам **сверить часы**. Возможно, в каком-то месте вы узнаете собственную проектную боль, которую давно считаете неизбежной. Возможно, с чем-то категорически не согласитесь. Возможно, какая-то проблема, которую раньше трудно было сформулировать, неожиданно получит для вас понятную конструкцию. Мне кажется, именно ради такого профессионального узнавания автоконспект и имеет смысл публиковать отдельно.

Сама монография значительно подробнее. Делать из неё коммерческий продукт я сейчас не вижу большого смысла: настолько глубокое погружение действительно нужно сравнительно небольшому кругу специалистов. Если кому-то после этого обзора понадобится пройти технологию целиком, можно написать мне лично — предоставлю материал.

Есть ещё одна оговорка, которую лучше сделать сразу. По ходу текста я иногда буду называть конкретные инструменты, которыми действительно пользовался в работе: **Microsoft Word, Microsoft Excel, Vanessa Automation, ChatGPT**, отдельные RPA-средства и некоторые другие продукты. Я долго думал, стоит ли вообще оставлять коммерческие названия, но полное их удаление создаёт лишнюю неопределённость: коллега перестаёт понимать, каким именно инструментарием был получен описываемый результат. Поэтому конкретное название будет появляться там, где оно действительно нужно для понимания практического контекста, но это не реклама и не утверждение, что данный продукт является единственно возможным. После первого упоминания мне важнее уже функция инструмента внутри технологии, а не его торговое название.

В итоге этот автоконспект — не история о том, как я придумал ещё одну методологию внедрения ERP. Скорее это попытка показать длинную профессиональную траекторию: от вопроса **«Купили ERP — что дальше?»**, через множество реальных проектов и постепенно сложившуюся технологию, к новому вопросу — **«ERP внедрили. Что дальше?»**

Ответ, к которому ведёт весь текст, для меня сегодня выглядит примерно так: сначала предприятие должно научиться не терять собственную причинность, затем превратить накопленное профессиональное знание в управляемую цифровую модель, а после этого попробовать использовать эту модель уже не только для объяснения прошлого и настоящего, но и для исследования будущего.

ERP фиксирует значительную часть того, что предприятие уже сделало. Следующий шаг — построить модель, которая позволит понять, что предприятие ещё способно сделать и какие из возможных переходов сохраняют его жизнеспособность.

До этого последнего вопроса, однако, ещё нужно дойти. Нельзя начать с цифровой реплики, графа или прогнозирования будущего, если мы не понимаем, каким образом вообще получаем достоверное знание о предприятии, кто имеет право подтверждать это знание и когда результат работы одной команды становится законным основанием для работы следующей.

И поэтому путь начинается с документа, который я сам долгое время воспринимал главным образом как организационную формальность.

С Устава проекта.

I. Устав проекта, который неожиданно оказался технологией

Если смотреть на Устав привычным взглядом, ничего особенно интересного в нём нет. Цели проекта, сроки, состав команды, роли, ответственность, основные этапы, иногда верхнеуровневые результаты — стандартный набор, без которого крупное внедрение формально трудно запустить. Проблема только в том, что реальная жизнь проекта очень быстро отделяется от этого документа. Устав продолжает существовать, но ответы на большинство практических вопросов команда ищет уже на совещаниях, в переписке, в отдельных регламентах и, хуже всего, в головах нескольких людей.

Именно поэтому со временем для меня изменился сам вопрос к Уставу. Меня перестало интересовать, насколько полно он описывает организацию проекта. Гораздо важнее стало другое: **можно ли из Устава понять, каким способом исполнитель и заказчик вместе будут получать профессиональный результат?**

Это различие кажется небольшим только на первый взгляд. Можно прекрасно описать фазы проекта и назначить на каждую из них ответственных, но по-прежнему не объяснить специалисту заказчика, в какой конкретной точке он должен подключиться к работе, какой результат ему предъявят, что именно от него потребуется и почему без его действия следующий участок проекта не должен начинаться.

А ведь специалисты заказчика не являются внешними наблюдателями технологии внедрения. Они находятся внутри неё.

Когда проект входит в обследование, предметную инженерию, профессиональное моделирование, проверку сценариев, приёмку результатов или подготовку перехода, на разных участках требуется разное участие людей со стороны предприятия. В одной точке специалист подтверждает фактическое состояние дел, в другой проверяет профессиональный смысл, в третьей разрешает противоречие между двумя позициями, в четвёртой оценивает результат относительно принятого критерия, а где-то уже обладает полномочием согласовать его, принять или отклонить.

Если этого маршрута участия нет, фраза «заказчик предоставляет экспертов» почти ничего не означает. Люди действительно приходят на встречи, отвечают на вопросы, получают документы, оставляют замечания, но само их участие технологически не определено. В результате проект легко превращается в бесконечный обмен мнениями, в котором никто до конца не понимает, какое решение сейчас должно быть получено и чьё действие делает его действующим.

Вот это для меня постепенно и стало одним из главных отличий технологии от обычной организации проекта. Назначить человека на роль недостаточно. Нужно ещё встроить его профессиональное действие в маршрут получения результата.

Роль становится частью технологии только тогда, когда определено, какой результат к ней приходит, что именно она имеет право с ним сделать и какое следующее действие проекта разрешается после этого.

Из этого достаточно естественно вырастает **граница обязательств исполнителя и заказчика**. Исполнитель может исследовать источники, подготавливать гипотезу, строить модель, сопоставлять данные, формировать результат и собирать доказательства. Но он не может самостоятельно подтвердить факт, который существует только внутри предприятия, сам себе выдать профессиональное заключение или от имени заказчика принять решение, для которого у него нет соответствующих полномочий.

Ровно так же участие заказчика нельзя описывать универсальной фразой «предоставляет информацию по запросу». Специалисту нужно заранее понимать, где он понадобится технологии, что будет подготовлено до его включения, какой промежуточный результат он получит, по какому критерию должен его рассмотреть и что произойдёт после его заключения. В этом месте технология перестаёт быть внутренним способом работы подрядчика и становится **общим маршрутом двух сторон**.

Мне здесь очень понятна производственная аналогия. Если после технологической операции предусмотрено испытание или приёмка, никого не удивляет, что изделие нельзя просто передать дальше со словами «мы всё сделали». Есть результат операции, есть определённый контроль, есть полномочие принимающей стороны и есть состояние, которое результат получает после проверки. Только после этого маршрут продолжается.

В ERP-проекте объектами такой проверки становятся уже не детали, а профессиональные результаты: фактическая модель предприятия, предметная модель, процесс, профессиональное решение, сценарная цепочка, подтверждённый разрыв типовой функциональности, программный выпуск, результат миграции или готовность предприятия к переходу. Для каждого такого результата должно быть понятно не только, кто его создал, но и **кто имеет право подтвердить, что проект действительно может на него опереться дальше**.

Если обязательная проверка, экспертное заключение или принятие со стороны заказчика не состоялись, проблема не в том, что не хватает подписи. В технологическом маршруте отсутствует необходимое действие, а значит, следующий результат пока не имеет достаточного основания.

Именно поэтому технологический маршрут должен быть зафиксирован уже в Уставе, а не появляться по мере того, как проект сталкивается с очередной проблемой. Специалисты заказчика должны заранее видеть собственный будущий путь внутри проекта: где понадобятся их знания, какой результат к этому моменту подготовит исполнитель, что именно нужно будет проверить и какое полномочие потребуется для дальнейшего движения.

Тогда становится понятна и сама логика фаз проекта. Их смысл не в том, чтобы разделить несколько лет работы на удобные календарные куски. Фаза имеет значение потому, что внутри неё производится определённый результат, а в завершении происходит контролируемый переход к следующему состоянию проекта.

И здесь продуктовая логика начинает выглядеть совершенно иначе. В ERP-проектах мы очень любим глаголы: обследовали, описали, смоделировали, разработали, обучили, протестировали. В календарном плане всё это выглядит вполне убедительно, но глагол сообщает только о выполненной активности и почти ничего не говорит следующему участнику проекта.

Фраза «обследование закончено» сама по себе не отвечает на главный вопрос: **что теперь существует такого, чего раньше не существовало и что следующая команда имеет право использовать как основание своей работы?**

Если обследование не сформировало подтверждённую фактическую основу, следующий участок проекта не должен начинать исследование предприятия заново. Если предметная модель не определила, чем именно предприятие управляет, прикладное проектирование не должно угадывать этот смысл по структуре ERP. Если профессиональная модель не установила хозяйственную или нормативную логику, технический специалист не должен извлекать её из состава реквизитов и поведения формы.

Поэтому значимый шаг проекта должен заканчиваться не просто выполненной работой, а **определённым продуктом**, который имеет понятное происхождение, критерии, состояние и принимающую сторону. Следующий продукт оказывается связан с предыдущим уже не календарно, а содержательно: он использует его как основание и сохраняет эту связь дальше.

На этом месте я перестал воспринимать продукт проекта как очередной документ, который нужно передать по договору. Если следующий участок технологии не зависит от

результата предыдущего, возникает вполне неприятный вопрос: а зачем вообще этот результат был нужен?

Именно здесь появляется ещё одно важное различие, которое потом проходит через всю технологию. В повседневной проектной речи слово «готово» используется слишком легко. Аналитик закончил документ — готово. Разработчик передал сборку — готово. Тестировщик выполнил сценарий — тоже готово. Но в профессиональной технологии между тем, что исполнитель закончил свою работу, и тем, что результат разрешено использовать дальше, существует несколько разных состояний.

Результат сначала должен быть **создан**. После этого он становится **наблюдаемым** для стороны, которая должна его проверить. Затем он **оценивается** относительно заранее понятного критерия. И только после этого сторона, обладающая соответствующим полномочием, может его **принять** либо вернуть на доработку.

А, вот ты о чём, Семён Семёныч. Получается, наше вечное проектное «ну мы же это сделали» на самом деле ещё ничего не гарантирует.

Сделано не равно принято. Результат становится основанием следующего шага только после того, как прошёл предусмотренный технологией контур наблюдения, оценки и принятия.

Причём принятие результата не означает, что он навсегда становится неприкосновенным. Если позже появляется новый источник, выявляется противоречие, меняется профессиональная норма или обнаруживается ошибка исходного решения, старый результат нельзя молча переписать и продолжить работу так, будто ничего не произошло. Должна появиться новая версия, а по связям проекта нужно понять, какие уже созданные продукты теперь затронуты этим изменением и требуют повторной проверки.

Так довольно рано возникает **цифровая нить**: основание связано с решением, решение — с результатом, результат — с последующими зависимостями, а изменение основания позволяет пройти по этой цепочке вперёд и определить последствия.

Это выглядит как техническая дисциплина ведения проекта, но на самом деле здесь решается одна из самых болезненных проблем зрелой ERP. Через несколько лет после внедрения предприятие часто прекрасно знает, **что** у него настроено, но значительно хуже понимает, **почему** это было настроено именно так. Причинность постепенно исчезает, и каждая новая модернизация вынуждена часть прошлого восстанавливать заново.

Именно поэтому Устав в этой конструкции неожиданно оказывается связан не только с началом проекта, но и с тем самым вопросом, который был поставлен во введении: **«ERP внедрили. Что дальше?»**

Если проект изначально строится как связанная технология получения, проверки и изменения результатов, после его завершения остаётся не только работающая система. Остаётся возможность восстановить происхождение решения, увидеть его профессиональное основание, определить зависимые результаты и понять, что именно понадобится перепроверить при следующем изменении.

Устав тем самым начинает задавать не только способ завершить текущий проект, но и **качество будущей изменяемости предприятия.**

Есть здесь и ещё один результат, который я долго недооценивал. Хорошо построенная технология меняет не только систему и процессы; она постепенно меняет самих специалистов заказчика. Человек, который раньше просто хорошо знал собственный участок, в ходе проекта начинает различать бизнес-предмет, состояние, событие, основание, критерий, доказательство, зависимость, точку передачи и границу собственного полномочия.

Это особенно важно после завершения внедрения. Если профессиональная логика проекта осталась только у консультантов, предприятие вместе с ними теряет часть способности понимать собственную систему. Если же специалисты заказчика были встроены в технологию

ческий маршрут и действительно участвовали в создании, проверке и принятии результатов, после проекта внутри предприятия остаётся не только ERP, но и более зрелый способ работать с её дальнейшими изменениями.

И вот здесь для меня вопрос «что остаётся после проекта?» перестал сводиться к системе, документации и обученным пользователям. Остаётся — или не остаётся — способность предприятия самостоятельно сохранять причинность собственных решений.

Отсюда становится понятна и последняя для этого раздела мысль. Профессиональный смысл не должен после проекта снова распасться между Microsoft Word, Microsoft Excel, схемами, настройками, программным кодом, трекером задач и памятью нескольких специалистов. Все эти инструменты полезны и реально используются в работе, но каждый из них хранит только свою проекцию результата.

Следовательно, где-то глубже должно существовать **структурированное цифровое ядро**, в котором сохраняются сами сущности, их состояния, связи, основания, версии, зависимости и доказательства. Документ, таблица, схема, пользовательская инструкция или сценарий проверки тогда становятся уже разными представлениями одного и того же принятого содержания.

ERP при таком подходе не теряет своего значения. Она по-прежнему остаётся одной из важнейших систем предприятия, но постепенно перестаёт быть единственным местом, из которого пытаются восстановить всю профессиональную модель бизнеса.

И вот после этого возникает следующий вопрос.

Если не начинать с объектов ERP и не заставлять информационную систему самой объяснять нам, чем живёт предприятие, то **с чего вообще начинать восстановление его модели?**

С этого места технология переходит к обследованию, вопросам Q0—Q1—Q2 и предметной инженерии.

II. Сначала нужно понять, чем именно управляет предприятие

В конце предыдущего раздела мы пришли к довольно неудобному вопросу. Если не пытаться восстанавливать предприятие непосредственно из структуры уже работающей ERP, то откуда вообще брать модель? Система перед глазами есть, документы есть, отчёты есть, пользователи прекрасно знают свои экраны, и самый естественный ход аналитика — открыть всё это хозяйство и начать разбираться.

Именно так я сам много раз и делал.

Берёшь подразделение, разговариваешь с руководителем, потом с ключевыми пользователями, смотришь документы, отчёты, настройки, постепенно рисуешь процессы. Через некоторое время появляется вполне убедительная схема действующего предприятия. Проблема обнаруживается позже, когда выясняется, что значительная часть этой схемы на самом деле описывает не деятельность предприятия, а **тот способ, которым деятельность сегодня отражена в информационной системе.**

Это не одно и то же.

На старых проектах я много раз попадал в эту ловушку. Кажется, что ты обследуешь предприятие, а на самом деле всё глубже обследуешь его ERP. Причём чем лучше знаешь систему, тем легче туда провалиться.

Отсюда и появился принцип, который сегодня кажется мне почти очевидным, хотя пришёл я к нему далеко не сразу.

Структура ERP может быть важнейшим источником обследования, но она не имеет права заранее определять онтологию предприятия. Сначала нужно восстановить профессиональную реальность, а уже потом выяснять, каким образом эта реальность представлена в системе.

Именно поэтому глубокое обследование не начинается сразу с бесконечной серии интервью. До разговора с дорогим специалистом предприятие и проектная команда должны выполнить свою домашнюю работу.

Первый уровень я обозначаю как **Q0**. Буква Q здесь — просто сокращение от английского *question*, «вопрос». Q0 — это не вопросник, который посылают пользователю, а предварительное исследование, в котором проект сам собирает доступные источники и строит гипотезу будущего обследования. Мы пытаемся понять структуру предприятия, предполагаемые направления деятельности, предметные области, действующие системы, существующие носители информации и круг специалистов, с которыми вообще имеет смысл разговаривать.

На этом уровне ничего ещё не объявляется истиной. Смысл Q0 как раз в обратном: заранее собрать пространство **гипотез**, чтобы не расходовать время специалистов заказчика на вопросы, ответы на которые проект мог получить самостоятельно.

Следующий уровень — **Q1**. Теперь подготовленная карта предъявляется людям, которые действительно способны видеть предприятие шире собственного рабочего места. Здесь уточняется периметр: какие предметные области реально применимы, какие предполагаемые участки деятельности отсутствуют, где проходит организационная граница, какие системы и внешние источники действительно участвуют в работе, кого ещё необходимо подключить к глубокому обследованию.

Q1 поэтому не является просто «более подробным Q0». Его продукт другой. После Q0 у нас есть исследовательская гипотеза, после Q1 — **подтверждённый периметр дальнейшего исследования.**

И лишь после этого имеет смысл запускать **Q2** — предметное и процессное профессиональное обследование уже внутри подтверждённых областей.

Здесь меняется сама точность вопроса. Специалиста больше не просят рассказать, «как у вас устроены закупки» или «чем занимается ваше подразделение». Вопросы начинают строиться вокруг конкретной профессиональной реальности: каким предметом вы здесь управляете, как отличаете один экземпляр этого предмета от другого, в каких состояниях он бывает, что переводит его из состояния в состояние, кто имеет право совершить этот переход, чем он подтверждается, что передаётся следующему участнику и где сегодня фиксируется соответствующая информация.

В результате $Q0 \rightarrow Q1 \rightarrow Q2$ — это не накопление всё большего количества интервью. Это последовательное **сужение неопределённости**. Сначала пространство возможно очень широкое, затем подтверждается применимый периметр, а внутри него уже восстанавливается фактическая профессиональная модель.

При этом ответ человека не становится нормой предприятия только потому, что этот человек опытный и уверенно говорит. Из обследования отдельно выделяются подтверждённые факты, гипотезы, противоречия, уже принятые решения и открытые вопросы. Если два сильных специалиста описывают один и тот же переход по-разному, нормальная реакция аналитика — не выбрать наиболее убедительного собеседника, а сохранить само противоречие как результат обследования, который требует отдельного разрешения.

Это тоже пришло не сразу. Очень хочется закончить интервью с красивой непротиворечивой картиной. Но иногда честно зафиксированное противоречие значительно ценнее, чем аккуратная схема, в которую аналитик незаметно добавил собственное решение.

Обследование не обязано немедленно производить правильный ответ. Его первая обязанность — отделить подтверждённое знание от гипотезы и не спрятать неопределённость внутри красивой модели.

Внутри **Q2** полезно различать два типа вопросов, хотя в реальном интервью они, конечно, пересекаются. Один тип восстанавливает сам **бизнес-предмет**: его границы, признаки идентичности, характеристики, источники данных, носители контекста и правила качества информации. Второй тип восстанавливает его жизнь в деятельности: состояния, события, роли, процедуры, операции, передачи, обратные связи и существующую автоматизацию.

Это разделение оказалось важным потому, что предмет и процесс — тоже не одно и то же. Предмет существует и сохраняет идентичность, а процесс объясняет, **какое существенное изменение этого предмета предприятие пытается получить**.

До этого, однако, нужно определить пространство, внутри которого вообще ищутся предметы. Так появляется **предметная область**.

Под предметной областью здесь понимается не подразделение и не функциональная подсистема ERP, а ограниченный профессиональный мир со своими управляемыми сущностями, состояниями, событиями, правилами и связями. Такая область может организационно пересекать несколько подразделений и технически использовать несколько систем, потому что её граница определяется не оргструктурой и не программой, а содержанием деятельности.

В универсальном доноре, который используется технологией, на текущей версии выделены 23 предметных кластера и 69 предметных областей. Это не означает, что каждое предприятие обязано немедленно получить все 69. Универсальная библиотека вообще не является моделью конкретного предприятия. Она отвечает на вопрос: **что уже известно технологии как возможное профессиональное содержание?**

Проектная модель отвечает на совершенно другой вопрос: **что из этого действительно существует, применимо и подтверждено здесь?**

Та же логика относится и к предметной библиотеке. Для версии, описываемой в монографии, приводятся 836 канонических бизнес-предметов, 844 доменные проекции, 637 меж-

доменных интерфейсов и 5 044 записи влияния. Эти цифры я привожу так, как они зафиксированы в материалах; важна здесь не сама величина, а принцип. Универсальная библиотека уже достаточно велика, чтобы работать как профессиональный донор, но конкретный проект не копирует её содержимое целиком. Он формирует **собственную проектную библиотеку**, в которой остаются только применимые предметы, их фактические проекции, связи и проектные уточнения.

Когда количество канонических предметов перевалило далеко за несколько сотен, для меня окончательно закончилась история про «давайте просто составим хороший словарь терминов». Это уже не словарь. Это материал, из которого можно собирать модель конкретного предприятия, не начиная каждый раз с чистого листа.

И вот здесь появляется ключевое понятие раздела — **бизнес-предмет**.

Для аналитика прикладной системы очень естественно мыслить документами. Услышал «закупка» — вспоминаешь заявку, заказ поставщику, поступление. Услышал «продажи» — заказ клиента и реализацию. Услышал «ремонт» — заказ на ремонт. В системе это действительно основные точки работы пользователя, поэтому постепенно возникает почти автоматическая подмена: если есть документ, значит, перед нами и есть тот самый бизнес-объект.

Но профессиональная реальность обычно устроена сложнее.

Закупочная потребность не равна заявке. Заявка не равна закупочной процедуре. Закупочная процедура не равна решению о выборе поставщика. Решение не равно договорному обязательству. Обязательство не равно заказу поставщику, а заказ не равен самой поставке.

Система вполне может объединять несколько этих смыслов в одном техническом объекте либо, наоборот, разносить один профессиональный предмет по нескольким объектам. Но техническая форма не должна определять, сколько самостоятельных предметов существует в деятельности.

Бизнес-предмет в этой конструкции — самостоятельная управляемая сущность, которая обладает собственной идентичностью, смысловой границей и жизненным циклом.

Идентичность отвечает на вопрос, по каким признакам мы понимаем, что перед нами всё ещё тот же самый предмет, несмотря на изменение его характеристик и состояния. Смысловая граница определяет, что относится к этому предмету, а что уже является другим предметом или только его характеристикой. Жизненный цикл показывает, какие значимые состояния предмет может проходить и какие события способны эти состояния изменить.

Вот здесь я бы сегодня остановил аналитика перед открытым экраном ERP и сначала задал ему один вопрос: что именно существует в предприятии независимо от того, каким документом или набором регистров это сейчас представлено? Если на этот вопрос нет ответа, прикладное сопоставление ещё рано начинать.

Бизнес-предмет существует в профессиональной реальности предприятия. ERP-объект — только одна из возможных технических проекций этого предмета.

Чтобы библиотека таких предметов не превратилась со временем в ещё один бесконтрольный словарь, появляется **таксон**. Здесь я использую это слово в довольно инженерном смысле: таксон — нормализованное описание класса бизнес-предметов, позволяющее одинаково понимать его в разных проектах и затем корректно проектировать конкретные экземпляры и проекции.

У таксона есть паспорт. В нём фиксируются нормативное имя и определение, критерии включения и исключения, признаки идентичности, значимые характеристики, возможные состояния и типы событий, типовые связи, источники, носители контекста и доказательства. Отдельно важно помнить, что паспорт описывает **возможную жизнь предмета**, но не назначает ему заранее процессную роль.

Это принципиально. Предмет сам по себе не является «входом», «выходом», «доказательством» или «драйвером». Такие роли возникают только тогда, когда предмет включён в конкретный процесс.

То же относится и к месту, где сегодня хранится информация. Я использую термин **«носитель контекста»** для того места, в котором на текущий момент зафиксирована информация о предмете: это может быть ERP, электронная таблица, текстовый документ, специализированная производственная система, бумажный документ, программный интерфейс обмена или другой носитель. Носитель важен для восстановления фактического состояния, но сам по себе он тоже не становится предметом.

Здесь особенно легко смешать разные сущности. Состояние бизнес-предмета — не то же самое, что событие; событие — не то же самое, что подтверждающий факт; факт — не документ; документ — не носитель; носитель — не поле; поле — не характеристика предмета. Пока эти различия не проведены, модель выглядит проще, но дальше практически неизбежно начинается ломаться трассировка.

Предмет без состояний вообще трудно использовать для управления. Поэтому следующим шагом восстанавливается его **жизненный цикл**. Состояние описывает профессионально значимое положение предмета в некоторый момент времени, а событие либо переводит его в другое состояние, либо подтверждает существенный факт, связанный с этим состоянием.

В простейшем виде это можно записать как: исходное состояние → событие → новое состояние. Но для проекта важна не сама стрелка, а всё, что стоит вокруг неё: кто имел право вызвать событие, на каком основании, какие условия должны быть выполнены, чем подтверждается переход и какие действия становятся допустимы после него.

Эта точка очень важна для уже внедрённой ERP. В системе обычно имеется много технических статусов, признаков проведения, состояний обмена и служебных флагов. Они полезны, но не всякий системный статус является профессиональным состоянием предмета. И наоборот, профессионально значимое состояние иногда вообще не представлено одним отдельным полем и вычисляется из совокупности данных.

Поэтому задача обследования — не переписать значения поля «Статус», а восстановить **профессиональную машину состояний**: что действительно изменяется в деятельности предприятия и каким наблюдаемым событием это изменение подтверждается.

И только после этого появляется процесс.

Это один из тех моментов, где у меня за годы работы действительно поменялся порядок мышления. Раньше естественно было начинать с вопроса «какие у вас бизнес-процессы?». Сегодня я сначала спрашиваю: **какими предметами предприятие управляет и какие существенные изменения их состояния оно пытается получить?**

Процесс в такой конструкции выводится из предмета. Для каждого самостоятельного бизнес-процесса должен существовать один **предмет-драйвер** — предмет, существенное изменение которого удерживает смысловую границу процесса.

Это сразу решает довольно старую проблему гигантских сквозных процессов. Например, под одним названием «Закупка» легко объединить возникновение потребности, подготовку заявки, проведение закупочной процедуры, выбор поставщика, возникновение обязательства, заказ, поставку и расчёты. На картинке это выглядит удобно, но внутри такой схемы одновременно живут несколько разных предметов с разными жизненными циклами.

Если драйвер меняется, значит, очень возможно, что закончился один процесс и начинается другой.

Вокруг предмета-драйвера находятся другие предметы, которые создают **предметное окружение процесса**. Они могут обеспечивать работу данными и ресурсами, служить доказательствами, задавать ограничения, использоваться в расчётах или выполнять контрольную функцию. При этом обеспечивающий или доказательный предмет не становится автоматиче-

ски самостоятельным процессом и не образует собственную горизонтальную линию передачи только потому, что участвует в деятельности.

Конец ознакомительного фрагмента.

Текст предоставлен ООО «Литрес».

Прочитайте эту книгу целиком, [купив полную легальную версию](#) на Литрес.

Безопасно оплатить книгу можно банковской картой Visa, MasterCard, Maestro, со счета мобильного телефона, с платежного терминала, в салоне МТС или Связной, через PayPal, WebMoney, Яндекс.Деньги, QIWI Кошелек, бонусными картами или другим удобным Вам способом.