

Автоматизация хаоса

Автор: Андрей Акопянц

Названием я обязан Георгию Башилову, который, сетуя на беспорядок в наших организациях, мешающий применять "правильные" информационные технологии, написал, что "Нельзя автоматизировать хаос. Это приводит только к его увеличению". Действительно - что и как можно автоматизировать в ситуации, когда стремительно меняется экономическая и законодательная реальность, когда в считанные годы возникают принципиально новые индустрии (например, банковская или ценнобумажная), когда не пишутся должностные инструкции, поскольку они устаревают к моменту их написания, а уже написанным никто не следует? Но ведь выбора нет - "хаос" объективен, и автоматизировать его нужно, а значит можно. И за 15 лет автоматизации всего, что шевелится, я накопил некоторый опыт разработки систем, наиболее сильно испытывающих влияние организационного "хаоса" - корпоративных информационных систем. Может быть, кому-то мои наблюдения покажутся тривиальными, а кто-то сочтет их ошибочными, но в конце концов, эти они не претендуют на универсальность и отражают мой ограниченный личный опыт.

Ты скажи, че те надо...

Что требуется от корпоративной информационной системы? Заказчики формулируют это по разному. Как правило, называю самые наболевшие проблемы - где налоговая наезжает из-за ошибок в отчетности, где себестоимость неизвестна, где при затаренном складе покупателей не удастся обслужить из-за отсутствия запасов нужной им продукции. Указывают на необходимость планирования финансовых потоков, потерянные документы, сорванные поставки... Часто начальнику хочется иметь возможность постоянно видеть результаты работы подчиненных. Некоторые особо продвинутые пользователи жаждут трехуровневое приложение с монитором транзакций и Java-сервлетами вместо их старенькой программы на FoxPro.

Называется и много других потребностей, которые должна удовлетворять система. И все они (разве что кроме трехуровневости) как правило, действительно важны для деятельности заказчика. Но любой системный аналитик знает старый афоризм: "То, что говорит пользователь о своих потребностях - это не совсем то, что он думает, а то, что он думает - это совсем не то, что ему нужно на самом деле".

На самом деле все вышеперечисленное - это верхушка айсберга. Если достаточно долго и последовательно задавать вопросы типа "А зачем Вам это надо", то в конце концов всплывают истинные проблемы, одинаковые для подавляющего большинства российских компаний. В сложных и быстро меняющихся экономических условиях предприятия оказываются фактически неуправляемыми из-за того, что у руководства отсутствует информация, необходимая для принятия решений.

Имеющаяся информация фрагментарна и рассредоточена по текстовым и excel-файлам, бумажным документам, записным книжкам и головами сотрудников. Данные бухгалтерии как правило, безнадежно (на отчетный период) отстают от жизни. Иногда имеются какие-либо системы, разработанные "на коленке" несколько лет назад, охватывающие малую часть деятельности компании, не с чем не интегрированные и не интегрируемые, и, в завершение всех бед, основательно искорежившие бизнес-процесс "под себя". Такая ситуация не только не позволяет эффективно управлять компанией, но и ведет к еще одному очень опасному последствию - появлению большого количества незаменимых людей, владеющих "тайными знаниями", жизненно необходимыми для функционирования предприятия.

Поэтому **первая и самая важная задача** корпоративной системы - это отчуждение и обобществление информации, вырабатываемой и используемой различными подразделениями и сотрудниками фирмы, или говоря иными словами, создание общего информационного пространства фирмы, в идеале охватывающего всю корпоративную информацию.

Для этого недостаточно спроектировать "честную" структуру данных. Необходимо сделать так, чтобы ее заполнение и актуализация выполнялись естественным образом, в ходе повседневной деятельности сотрудников. Выполнение операций с помощью системы должно быть в подавляющем большинстве случаев не более трудоемко, чем "по старинке", а в исключительных случаях - трудоемко, но возможно. Несмотря на тривиальность этого тезиса, добиться этого в реальности оказывается сложно, поскольку полный перечень выполняемых операций составить не удастся практически никогда.

Тем не менее нужно стремиться к решению **второй задачи** - повышению эффективности и надежности выполнения рутинных операций, включая подготовку выходных бумажных документов.

Третья задача любой корпоративной информационной системы - это организация удобного доступа к

накопленной информации как в виде отчетов, так и в режиме онлайн-фильтрации и просмотра. "Хаос" тут привносит свою лепту - по аналогии с 1 законом Мэрфи (все, что может испортиться, портится) - любой запрос, который можно придумать, кому-нибудь обязательно понадобится. Поэтому здесь, как и в случае поддержки операций, нужно стремиться к тому, чтобы типичные запросы задавались удобно и отрабатывались быстро, а нетипичные - как-то, но отрабатывались.

Собственно, способ организации информации в системе и интерфейсы для работы с ней в значительной степени определяется потребностями доступа к информации - нужно, чтобы легко можно было находить нужные записи как по их атрибутам, так и по связям с другими записями.

И наконец, **четвертая задача** связана с тем, что кроме информации о происшедших событиях система должна хранить информацию о событиях ожидаемых, и вовремя напоминать о них. Это своего рода функция контроля исполнения.

Предельным выражением этой функции является технологии WorkFlow, когда система наделяется знанием не только о том, что и когда должно быть сделано, но и кем, и в какой последовательности... Правда, в условиях "хаоса" WorkFlow умирает первым, и в реальной российской жизни эти технологии мало применимы.

Кроме контроля исполнения, знание ожиданий позволяет строить прогнозы - например, денежного и товарного потока. Развитием этой (прогностической) функции является возможность исследования вероятного будущего ("What If" анализ), когда система позволяет "играть" с ожиданиями, и анализировать последствия их реализации.

В идеале, корпоративная информационная система должна замыкать на себя всю деятельность всех сотрудников компании. Мерой полноты системы может служить соотношении времени, которое сотрудники проводят в среде системы, и в офисных приложениях. "Приватная" Excel-табличка, в которой ведется учет чего-то отдельным сотрудником, вручную (непосредственно в Word-e) формируемые документы, распечатки, расчерченные цветным маркерами - все это свидетельства того, что систему есть куда развивать.

При этом корпоративная система не должна подменять специализированные приложения - бухгалтерские, аналитические и др. Последние, по идее, должны "питаться" экспортированной из нее информацией.

Бухгалтер, милый мой бухгалтер...

Подавляющее большинство заказчиков в качестве основной цели внедрения информационных технологий указывают автоматизацию учета. Здесь очень важно понять, что учет в организации, как правило, делится на две неравных части - учет, связанный с ее основной производственной деятельностью, и учет всего остального, куда относится начисление зарплаты, основные фонды, представительские расходы и прочие факторы и издержки, не связанные напрямую со спецификой организации.

По целям учета также имеются две разновидности:

Управленческий учет, задача которого - предоставление лицам, принимающим решения, необходимых им показателей, и как можно быстрее - сколько сегодня отгрузили продукции, сколько запасов на складе, и как это соотносится с объемом заказов на завтра, какова текущая себестоимость продукции, от кого мы ждем денег и сколько, и т.д.

Бухгалтерский, или фискальный учет, задача которого - формирование официальной финансовой отчетности организации к окончанию отчетных периодов. Бухгалтерский учет имеет дело со всей финансовой и хозяйственной деятельностью компании, но от него не требуется оперативность.

Хотя и управленческий, и бухгалтерский учет имеют дело в основном с одними и теми же первичными хозяйственными операциями, но способы их отражения существенно различаются. Управленческий учет отличается от бухгалтерского так же, как финансовый директор от главного бухгалтера.

Различия управленческого и бухгалтерского учета

Различная учетная политика.

В управленческом учете расчет, например, балансовой стоимости и прибыли зачастую ведется по факту возникновения обязательств, в российском бухгалтерском учете как правило, по факту оплаты. Во всяком случае, учет обязательств, и показателей, связанных с ними, является важнейшей частью управленческого учета, и почти никогда не отражается в бухгалтерском учете.

Различная степень детализации.

В управленческом учете операции, связанные с основной производственной деятельностью отражаются максимально детально, а издержки управления как правило, грубо и интегрировано, так как они квазипостоянны, и слабо влияют на принятие решений.

В бухгалтерском же все наоборот - издержки управления отражаются детально, а движение средств, связанное с основной производственной деятельностью в нем зачастую может учитываться интегрировано за периоды, особенно при большом объеме транзакций.

Различная область охвата хозяйственной деятельности

В реальности у любого предприятия кроме "белой", официальной составляющей его деятельности есть "черная" - работа с неучтенными товарами за наличный расчет, и "серая" - работа через квазииностранные (оффшорные) компании, и российские компании - однодневки. Понятно, что в управленческом учете должна отражаться вся хозяйственная активность, а в бухгалтерском - только белая.

При этом зачастую решение о том, как проводить конкретные операции, принимается потом - по концу отчетного периода, и сопровождается рядом заключаемых задним числом сделок по переброске товаров и денег между различными составляющими оборота.

Имеется и ряд других различий, например, для управленческого и бухгалтерского учета используются разные планы счетов и наборы проводок при отражении одних и тех же хозяйственных операций.

И очень много проблем при автоматизации учета происходит от того, что этого различия понятий часто нет в голове у заказчика. При отсутствии управленческого учета именно бухгалтерия является не очень качественным, но единственным источником финансовой информации. Поэтому руководители, испытывая недостаток оперативных данных, требуют их у бухгалтерии. Бухгалтерия дать их не может, и соответственно, задача постановки управленческого учета формулируется как улучшение работы бухгалтерии.

И во многих случаях разработчики на это покупаются, и начинают автоматизировать бухгалтерию. И на объективную сложность создания системы накопления первичной информации накладывается задача отображения хозяйственных операций в бухгалтерские проводки, которая сама по себе достаточно сложна, плохо формализуема и подвержена быстрому изменению постановки задачи.

В результате разработка начинает буксовать, и за разборками с главным бухгалтером приходится забыть про высокую цель создания общего информационного пространства, и отстреливать все, что не относится к выходу на баланс.

А главный парадокс этой ситуации заключается в том, что, даже если героическими усилиями бухгалтерию удастся автоматизировать, то руководитель все равно недоволен, так как не получает того, что ему на самом деле было нужно!

Таким образом похоже, что управленческий учет должен являться органической составной частью корпоративной информационной системы, в то время как бухгалтерский не может быть успешно реализован в этих рамках по одной причине - слишком велик уровень сложности при таком совмещении. По крайней мере, мне не известен не один успешный пример такого рода.

Даже там, где это возможно с точки зрения используемого инструментария (например, 1С:Предприятие допускает такое конфигурирование), оперативный и бухгалтерский учет разнесены по разным подсистемам, настраивать которые нужно отдельно. И я не уверен, что при этом не возникает неразрешимых противоречий в учетной политике - например, когда считать документ проведенным.

Разумным подходом является реализация корпоративной системы, реализующей накопление всей первичной информации с выходом на управленческий учет, и последующий экспорт данных в бухгалтерскую программу.

Но на практике таким образом удастся автоматизировать отнюдь не весь бухгалтерский учет, а только рутинную часть работы бухгалтерии, связанную с отражением основной производственной деятельности. Эта доля может составлять 80% по количеству проводок, но основное их разнообразие дает учет всех остальных многочисленных факторов, влияющих на финансовую отчетность.

Последние, даже если и данные о них попадают в информационное пространство корпоративной системы, проблематично автоматически отразить в бухгалтерском учете, поскольку на способ их отражения сильно влияют быстро меняющиеся и трудно формализуемые соображения типа минимизации налогов в свете новой инструкции о переоценке основных фондов.

Покажи мне свои данные, и я скажу, что тебе надо

Существует много толстых книжек, описывающих, как "по науке" надо разрабатывать информационные системы. Как правило, там описано, что нужно проанализировать функции автоматизируемого объекта и отдельных подсистем. Далее определяются функции конкретных типов рабочих мест и потоки данных и управления между ними. Затем специфицируются хранилища данных и способы доступа к ним для каждого рабочего места, фиксируется в некотором формализме и уже исходя из всего этого осуществляется кодирование.

Этот подход, сложившийся на Западе в 70-тые годы, и успешно применялся для автоматизации больших консервативных корпораций, работающих в стабильных условиях,

Более того на том уровне развития инструментария, когда любой элемент пользовательского интерфейса и любой запрос к данным нуждался в кропотливом и трудоемком программировании иначе действовать, вероятно, было нельзя.

А более новых книжек на эту тему что-то не видно - то ли их писать перестали, то ли переводить...

Последним отслеженным мной идеологическим всплеском стала идеология быстрого прототипирования,

предлагавшая на базе языка 4-того поколения (!) быстренько создать действующий макет системы, отладить его функциональность на пользователях, а затем кодировать систему "по честному".

Но с тех пор изменилось довольно многое. Жизнь стала гораздо более динамичной. Сроки разработки в год и более, бывшие нормальным явлением в 70-80 годы, в современных условиях недопустимы. Вырос уровень инструментальных средств. Все современные CASE и RAD - средства давно перешагнули уровень "языков 4-того поколения". Реляционные СУБД кардинально упростили реализацию запросов к данным. Фактически, все современные системы являются именно макетами в понимании "быстрого прототипирования" 80-тых.

Во всяком случае, этот традиционный подход "от функций" в нашем сегодняшнем хаосе работает плохо. На это есть несколько причин:

- как правило, функции плохо формализованы

- структура организации, функции людей и распределение обязанностей зачастую меняются быстрее, чем разрабатывается система.

Очень часто также в процессе обследования становится ясно, что автоматизировать существующие способы выполнения операций нельзя, но при этом непонятно, удастся ли реорганизовать работу организации.

А для разных вариантов будущей организации бизнес-процессов компании мы получим различные проекты информационной системы. Выбирая из них какой-либо один, мы рискуем получить неработоспособную систему при малейшем изменении ситуации.

Но существует подход, который успешно работает даже в таких условиях. Если попытаться выделить "общий знаменатель" для различных вариантов организации бизнес-процессов, то им, как правило, окажется та совокупность данных, с которой, собственно, работает компания - собственно, то самое информационное пространство, обеспечение которого мы постулировали как первую, и самую важную, задачу корпоративной системы.

Выясняется, что данные, с которыми идет работа, их внутренняя логика и взаимосвязь гораздо более консервативны и менее чувствительны к изменениям организационной ситуации, чем функции. Оказывается так же, что огромное большинство функциональных операций, выполняемых сотрудниками организации, сводится к вводу, поиску и достаточно тривиальному редактированию данных.

Этот факт позволяет использовать проектирование "от данных". При этом подходе на базе анализа функций и информационных потоков организации создается то, что я называю "информационной моделью предметной области" - перечень объектов предметной области и их взаимосвязей, выделенных таким образом, чтобы как можно большая часть содержательных операций могла быть описана в терминах ввода, поиска и редактирования объектов и отношений между ними.

Такой подход к автоматизации отнюдь не нов.

Именно на он обусловил колоссальную популярность Excel или Lotus Notes как механизмов "малой автоматизации". Причем в подавляющем большинстве случаев имеющиеся в этих продуктах средства программирования не используются - хватает возможности вводить, просматривать, искать и печатать данные.

Правда, реальные данные имеют, как правило, более сложную структуру, чем таблицы Excel или плоские файлы Notes. В "честной" модели предметной области появляется много разных типов объектов и отношений между ними, причем свойства объектов и отношения изменяются во времени. Поэтому требуются более сложные средства моделирования.

В начале 80-тых на реализацию подобной концепции стали претендовать реляционные СУБД. Крайне популярны были декларации о "Программировании без программиста", основанные на использовании реляционных моделей и QBE (Query by example) в качестве среды для работы пользователя. Все оболочки популярных настольных систем (от DBase-II до Access-a) разрабатывались именно исходя из этой парадигмы.

Если вспомнить, так даже SQL разрабатывался не как стандартный способ взаимодействия с серверами баз данных, а как язык запросов, ориентированный на неквалифицированного пользователя!

Но жизнь показала, что рядовой пользователь не способен воспринять сколько-нибудь сложную реляционную модель данных. Конечно, существует класс задач, для которых база данных в стандартной оболочке Access-a может использоваться достаточно квалифицированными пользователями как информационная модель предметной области, но разрабатывать таким способом корпоративную информационную систему может прийти в голову только в кошмарном сне.

При таком подходе ядро системы проектируется как специализированный редактор данных. При этом значительная часть бизнес-логики системы уходит на уровень поддерживаемых системой связей между данными, специфических ограничений целостности и специальных операций редактирования.

При таком подходе выясняется, что значительная часть содержательных операций при этом может выполняться стандартными методами, не привязанными к специфике предметной области. Например, большинство поисков вполне успешно реализуется вполне универсальным механизмом QBF (Query by form

- способ задания поисковых запросов, при которых пользователь вводит условия запроса непосредственно в полях форм просмотра/редактирования данных). А возможности настройки состава и порядка полей в таблицах в сочетании с QBF значительно уменьшают потребность в специально разработанных отчетах. Такая простая функция, как суммирование указанных полей по текущей выборке покрывает огромное множество запросов типа "А сколько у нас...", каждый из которых, в функциональном подходе, пришлось бы прописывать отдельно.

Разделение типов рабочих мест при этом выполняется путем разграничения прав по доступу и выполнению различных операций на уровне объектов модели предметной области.

Конечно, нельзя надеяться, что при разработке корпоративной информационной системы удастся ограничиться реализацией интерфейсных решений, как правило, не очень удобны для каждого конкретного случая. Кроме того, некоторые типичные операции могут потребовать более одной операции ввода/редактирования данных. Поэтому для наиболее массовых или критических по времени операций и запросов, как правило, потребуются специальные интерфейсы. Как правило, не удастся избежать также реализации специализированных отчетов. Рано или поздно придется наделять систему активностью, что бы обеспечивать контроль исполнения и отработку отложенных или массовых операций.

Но прелесть этого подхода заключается в том, что ядро системы в виде модели предметной области уже готово к работе, при этом дополнительную бизнес-логику можно подключать после запуска системы в эксплуатацию. Кроме того, разумная организация данных стимулирует возникновение разумных способов работы с ними, и позволяет осуществлять некий "ползучий" реинжиниринг бизнес-процессов, когда старые процессы отмирают за их очевидной абсурдностью, а новые возникают как бы сами собой.

В конце концов, если система проживет достаточно долго, и корпорация формализует свои внутренние технологии, то разрабатываемая "От данных" система придет к тому же состоянию, что и система, которая разрабатывалась бы "от функций". Только "от данных" к этому состоянию прийти возможно, а "от функций" - практически нет.

Существует проблема инструментария для реализации таких моделей. Современные инструменты быстрой разработки позволяют реализовывать такие модели, но требуют для этого довольно много программирования, так как базовые элементы интерфейса, их поведение, способы их связи с данными и друг с другом в современных инструментальных средствах не соответствуют содержательным понятиям модели предметной области.

Ближе всего из известных мне систем к платформе для реализации информационных моделей подошла система DataEase. Она была в свое время очень популярна на DOS-овской платформе, и неоднократно бравшая призы на конкурсах по скорости разработки приложений.

Именно из DataEase я позаимствовал идею двойственности табличного и "формового" представлений данных, QBF как основной способ задания поисковых запросов, и очень важную концепцию связанных форм. Эти механизмы я успешно использовал впоследствии во всех своих проектах. А мощностю и удобством системы разработки отчетов DataEase до сих пор заставляет вспоминать о ней с сожалением.

Но DataEase отличалась малой гибкостью - скелет приложения на ней создавался очень быстро, а довести его до требуемой пользователем функциональности удавалось далеко не всегда. Эта система исчезла с горизонта при массовом переходе на Windows, привнесшем новые интерфейсные концепции, и поменявшем расстановку сил на рынке настольных СУБД.

С точки зрения современных представлений, средства описания информационной модели предметной области должны быть достаточно близки к реляционным, но с рядом существенных отличий.

Обязательно должны учитываться отношения наследования между объектами. Объекты могут не однозначно соответствовать таблицам - некоторые объекты могут представляться несколькими таблицами, и некоторые таблицы могут относиться к нескольким объектам. Имеются и другие отличия. Идеально было бы иметь такой CASE, в который можно было бы заложить описание модели, и получить работающую систему, которую дальше можно было бы дорабатывать, но такой инструментарий мне неизвестен. Попытки его создания предпринимались мною в течении ряда лет, пока не стало ясно, что в современных условиях бессмысленно разрабатывать сколько-нибудь масштабный инструментарий, не имея поддержки крупных компаний.

Автоматизируем учет

Отдельно стоит рассмотреть реализацию объектов, характерных для учетных задач. Как правило, в процессе выполнения операций нам нужно накапливать некоторые итоговые числовые данные. Это может быть количество товаров на складе, начисленная работнику зарплата, количество акций, принадлежащих акционеру, остаток на расчетном счете предприятия, количество пенсионеров, направивших свои обращения в городскую администрацию, и т.д.

В принципе, все эти цифры (следуя бухгалтерской лексике, будем называть их остатками) однозначно следуют из имеющихся у нас первичных данных, и могут быть получены путем генерации соответствующих отчетов, но в реальных системах это обычно неприемлемо из-за того, что смотреть на эти итоговые цифры нужно постоянно, а генерация отчетов требует значительного времени.

Потому типичным является решение, когда ввод и редактирование данных об операциях нагружается изменением значений остатков.

Относительно остатков нас, как правило, интересует не только текущее их значение, но и их значения на прошлые даты, а так же динамика их изменения. Учитывая, что логика формирования остатков может быть достаточно прихотливой, и различается для разных типов операций, "откатить" остаток на основании данных об операциях может быть очень непросто. Поэтому обычно для развязывания логики операций и динамики остатков вводят понятие, которое по аналогии с бухгалтерией, будем называть проводками.

Проводка - это датированная запись, описывающее элементарное изменение одного (полу проводка) или двух (полная проводка) остатков, содержащая ссылку на инициировавшую данную проводку операцию.

При этом логика получается следующая - остатки изменяются только проводками, а операции, желающие изменить остатки, делают это, формируя соответствующие проводки. При такой конструкции откатка/накатка остатков, анализ их динамики за период не вообще требуют знания логики операций, и выполняются чисто на уровне остатков и проводок.

Хорошей практикой является структурное отделение подсистемы, работающей с остатками и проводками, от остальной системы. Такую подсистему я называю машиной проводок. Примером машины проводок является план счетов в любой системе бухгалтерского учета или регистры в "1С:Предприятие".

Но машина проводок связана с остальной системой не только через операции. Связь проявляется и в том, что остатки "висят" на каких-то объектах информационной модели (задолженность контрагента), или на их пересечении (остаток данного товара на складе N1, например). В бухгалтерском учете эта связь отражается с помощью аналитических параметров счетов, указывающих на соответствующие объекты.

Организация системы остатков в машине провод как правило, очень хорошо ложится на характерную для OLAP модель многомерной базы данных, где измерениями являются "план счетов" и аналитические параметры. Проводки также ложатся в эту модель, только у них поворачивается дополнительное измерение - время. OLAP-модель запросов, выраженная в терминах срезов, поворотов и "схлопывания" измерений метакуба покрывает все возможные запросы к машине проводок, предоставляя при этом удобную концептуальную модель интерфейса.

Сейчас появилось достаточно много модулей, обеспечивающих OLAP-подобные интерфейсы, и я очень рекомендую ими пользоваться для визуализации данных машины проводок.

Отец реляционной алгебры Кодд в начале 90-тых выдвинул новую идеологию, названную им OLAP (on-line analytical proceeding). Моделью данных для поддержки OLAP является многомерный куб, где на измерениях определены некоторые иерархии, а в клетках этого куба находятся числовые значения.

Например (близкий мне пример), динамика рынка ценных бумаг хорошо описывается измерениями Инструменты, Торговые площадки, Дата-время и Параметр (цена спроса, цена предложения, цена последней сделки, объем и др.), а значение параметра находится на пересечении этих измерений.

Единичный факт (точка) выглядит, например, как "Обыкновенные акции МосЭнерго на РТС имели в 8 августа 1998 г. в 15:00:00 bid (цену покупки) 1 цент".

Операции извлечения данных из такого куба описываются в терминах поворотов, срезов, и иерархического "схлопывания" измерений с агрегированием значений (суммирование, взятие среднего и др.). Эта метафора хорошо ложится на табличную организацию пользовательского интерфейса.

Имеется класс программ для поддержки такой работы с данными. Они обычно используются для создания так называемых "хранилищ данных". Наиболее известный у нас в стране программный продукт этого класса - это Oracle Express Server.

Имеется также вариант этого подхода, называемый ROLAP - Relational OLAP, когда данные хранятся в реляционных базах данных, а работа с ними идет в OLAP-методологии и интерфейсе. Более того, последние версии SQL-серверов, например, Oracle8i, содержат специальную поддержку характерных OLAP способов организации данных, за счет чего их производительность на OLAP-задачах приближается к производительности специализированных хранилищ данных.

Внедрение

"все на свете должно происходить медленно и неправильно...". В Ерофеев.

Я являюсь убежденным сторонником поэтапной (инкрементальной) разработки и внедрения. Я не верю в "метод большого скачка", когда система долго делается в отрыве от заказчика, затем приносится, устанавливается, и начинает эксплуатироваться.

То есть она конечно, приносится, но тут же выясняется, что чего-то забыли, что-то поменялось, и вообще -

чтобы набить ее данными, обучить персонал, и хоть как-то запустить ее, нужно на неделю прекратить всю остальную деятельность компании.

Я помню, как в одной очень крупной брокерской компании устанавливали систему автоматизации. Это событие было приурочено к праздникам, тем не менее компании пришлось остановить трейдинг на три дня (сводили остатки). Убытки от этого, по скромным подсчетам, в два раза превысили стоимость самой системы.

Я для себя даже вывел цифру - три месяца после начала разработки. Если через этот срок первые модули системы не начинают реально эксплуатироваться и приносить пользу, то шансы проекта на успех резко уменьшаются.

Конечно, такой подход заметно увеличивает трудоемкость разработки по сравнению с "офф-лайновой" реализацией. Для того, чтобы запустить в эксплуатацию кусок системы, требуется выполнить много программистской работы, которую вообще говоря, разумно было бы выполнять на более поздних этапах, и которую все равно придется повторять.

Дело в том, что имеются две разных логики - логика реализации и логика внедрения системы.

Разработчики мыслят в логике реализации, и им трудно объяснить, зачем нужен с их точки зрения "мартышкин труд" по реализации аккуратных заглушек, замещающих функции, которые все равно будут реализовываться позднее, и разработка аккуратных интерфейсов к структурам данных, которые все равно будут меняться.

При этом должен быть кто-то, мыслящий в логике внедрения, и способный понять, что, например, если сейчас, срочно (а не через два месяца) не сделать простейшие отчеты для начальства, то с таким трудом запущенный процесс ввода данных в систему по факту прекратится, будут обесмыслены уже введенные данные и процесс внедрения растянется на неопределенное время.

Поэтому задача постановщика (близкая к искусству) - это таким образом спланировать процесс, чтобы, при инкрементальной разработке и внедрении количество лишней и "авральной" работы было минимально.

Есть еще один выведенный мной критический срок - проект не должен длиться дольше 8 месяцев. Конечно, взаимодействие заказчика и разработчика может длиться годами, но длительность каждого отдельного цикла проект-реализация-внедрение не должна превышать эти самые 8 месяцев. Проекты с большими плановыми сроками заставляют вспомнить известную историю про Ходжу Насредина, который взялся за 20 лет выучить ишака разговаривать, рассудив, что за это время умрет либо шах, либо ишак, либо он сам.

Заключение

Имеется интересный психологический феномен. Часто человек, успешно сделавший нечто объемное и значимое, извлекает из этого своего опыта некоторые очень общие и важные для него принципы, и именно их воспринимает как главный результат своей деятельности. Но когда он пытается пересказать их окружающим, его энтузиазм вызывает легкое удивление, так как для подавляющего большинства эти принципы очевидны. Проблема состоит в том, что для говорящего они наполнены содержанием в силу того, что они пропущены через его жизненный опыт, которого нет у окружающих, поэтому для них эти принципы столь же тривиальны, сколь и бесполезны. Но я надеюсь, что в случае данной статьи какие-то из изложенных мною соображений могут оказаться полезными и без опоры на аналогичный моему профессиональный опыт.