2. ПРЯМАЯ ТЕХНОЛОГИЯ

Приступая к решению новой системной задачи, человек использует прямую технологию. Она заключается в разбиении всех работ на этапы с последующим выполнением этапов, от первого до последнего. Прямая технология успешно применяется при решении множества несистемных задач, с которыми человек сталкивается ежедневно, и по этой причине представляется ему вполне проверенной на практике. Ее логика сводится к следующему. Если система состоит из элементов, то изучение анатомии и физиологии каждого элемента с последующим обобщением результатов даст нам полную модель системы. Приняв такое интуитивно очевидное решение, разработчик направляет всю свою энергию на изучение элементов системы. Обобщение полученной информации сводится при этом к организованному складированию локальных схем и описаний.

2.1 Исходные представления

Зафиксируем исходные представления разработчика о методах построения АСУП, чтобы потом сравнить их с реальной технологией и сделать соответствующие выводы.

Конструкция объекта

Объект, для которого строится АСУП – это предприятие, а точнее – административная система управления предприятием. В ее недрах решается множество взаимосвязанных задач, распределенных между подразделениями и работниками. Часть работников занята творческим трудом, который не поддается очевидной формализации. Другая часть, и довольно многочисленная, выполняет повторяющиеся рутинные операции, которые вполне могли бы быть поручены вычислительной машине.

Цель разработки заключается в автоматизации рутинных задач, что существенно повышает скорость и качество вычислений и высвобождает людей для творческого труда.

Конструкция АСУП

Автоматизированная система представляет собой множество программно–технических комплексов, пред-

назначенных для решения задач обработки данных. Взаимодействие комплексов осуществляется на основе общей схемы, которая во многом повторяет административную схему управления предприятием.

Модель каждой задачи представляет собой зафиксированную структуру данных и алгоритмов их обработки. Каждый алгоритм имеет однозначное описание, из которого ясно, какие переменные используются на входе, как они обрабатываются, и куда выдается результат.

Порядок работ

Основные технологические этапы показаны на рис. 8. На фазе обследования уточняется список задач, подлежащих автоматизации. По каждой задаче определяются структуры данных, алгоритмы и перечень выходных документов. Все эти сведения можно получить у будущего пользователя задачи, который сегодня решает ее вручную.

Рис. 8. Этапы прямой технологии

Полученная информация формализуется, выверяется и оптимизируется. Появляется математическая модель системы. Параллельно определяется архитектура системы – схема расстановки технических средств, принципы передачи информации между компьютерами, организации хранилищ информации. Затем появляется рабочий проект, конкретизирующий модель с учетом выбранных технических средств и программного инструментария.

На стадии программирования модель реализуется в виде баз данных и прикладных программ. Этап завершается тестированием программы, то есть проверкой ее на работоспособность, на соответствие модели и проекту.

Внедрение заключается в передаче программы пользователю и проведении опытной эксплуатации. На этой стадии программа работает в реальных условиях подразделения, использует реальные входные данные, что позволяет выявить ошибки моделирования и проектирования, а также увидеть недостатки эксплуатационного характера (скажем, оценить длительность поиска информации).

Эксплуатация программы как правило предполагает надзор за ее работой со стороны программиста (сопровождение) с целью устранения ошибок, не обнаруженных ранее, а также адаптацию программы к изменениям во внешнем окружении предприятия и к появлению в системе новых комплексов и подсистем, с которыми она должна взаимодействовать.

Команда

В отделе АСУ появляются группы проектирования, программирования и сопровождения, каждая из которых берет на себя выполнение соответствующих этапов.

Задача проходит по конвейеру от группы к группе, поэтому результаты проектирования и программирования довольно подробно и тщательно оформляются. Существуют различные документы (акты, протоколы замечаний...), регламентирующие отношения между группами разработчиков, а также их взаимодействие с пользователями.

Ожидаемый результат

Предполагается, что автоматизация всех рутинных задач позволит построить законченную АСУП. Разработка системы планируется как одноразовая кампания, имеющая начало и конец, после чего наступает период длительной эксплуатации, практически не требующий привлечения разработчиков высокой квалификации. С учетом обычных мероприятий по сопровождению, система отработает весь проектный срок, до тех пор, пока окончательно не устареет физически и морально. Затем ей на смену придет новая система.

2.2 Фактические результаты

При решении простых задач технология работает удовлетворительно. Но с их усложнением она все дальше отходит от первоначальной схемы и подвергается корректировке.

Уровень формализации

В принципе, сложность задачи определяется по целому ряду признаков, среди которых объем вычислений, степень разветвленности алгоритмов, количество переменных. Но оказывается, что нарушение технологии зависит в основном от степени формализации задачи. Обращаясь к пользователю, разработчик в одних случаях получает точное описание структуры данных и однозначный алгоритм расчетов, а в других случаях ему приходится проводить целое расследование.

Как правило, система стартует с задач, которые давно устоялись и определились. Для их окончательной формализации требуется лишь уточнить некоторые детали. Компьютерная программа копирует имеющиеся алгоритмы, поэтому пользователь не видит в ней существенных отличий от тех представлений, которые у него уже сложились, и внедрение проходит гладко.

Затем разработчики переходят к анализу более сложных процессов, которые нужно предварительно уточнять и дорабатывать. Разработчику необходимо выяснить, какие ситуации могут возникнуть на входе программы, какие алгоритмы должны быть применены для их обработки. К сожалению, сложные комплексные задачи как правило не имеют четкого описания. Одни режимы работы детально формализованы, другие вообще не учтены.

Творчество разработчика

В этих условиях разработчик вынужден самостоятельно дорабатывать структуру данных и составлять дополнительные алгоритмы, опираясь на логический анализ добытых им сведений и свой прошлый опыт. Отыскать новые структуры и алгоритмы несложно, но проблема заключается в выборе наиболее эффективного решения из множества возможных, а объективный критерий выбора отсутствует.

Чем меньше формализована задача, тем шире творче-

ство разработчика. Однако ответственность за эксплуатацию комплекса лежит на пользователе, поэтому именно он может дать санкцию на использование новых алгоритмов. Решения разработчика, даже если они выглядят привлекательно, требуется тщательно проверить на практике. Проверка требует времени, а ускорить естественный процесс врастания решений в существующую систему очень трудно.

Внедрение

Таким образом, на этапе проектирования некоторые алгоритмы исходной задачи вошли в компьютерную модель без изменений, а другие подверглись доработке. Пользователь без проблем принимает старые алгоритмы, а по отношению к новым ведет себя очень осторожно. Мы видим, что часть исходной задачи проходит все этапы технологии и приводит к конечному результату, а другая часть «застревает» на этапе внедрения и требует серьезной доработки (рис. 9).

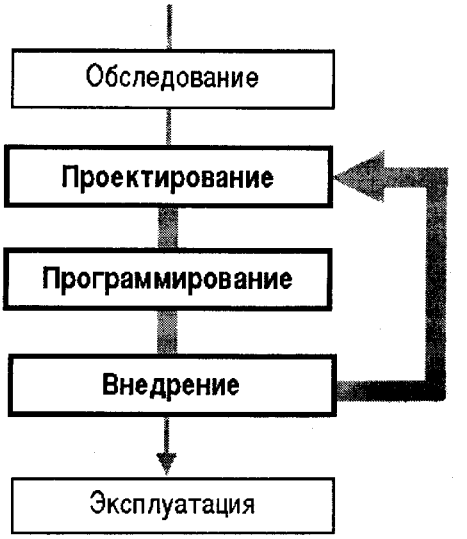

Рис. 9. Изменение технологии после перехода к сложным задачам

Другими словами, пользователь принимает комплекс к эксплуатации только частично. Обсуждение спорных режимов откладывается на дополнительную проработку. Внедрение завершается приемом первой версии в эксплуатацию и составлением списка замечаний, который требует возврата к проектированию и порождает вторую версию комплекса.

Циклы

Выполнение работ зацикливается, на этапе внедрения задача раздваивается, одна ее часть уходит на сопровождение, а другая возвращается на проектирование.

Разработка второй версии порождает появление третьей, и так продолжается довольно долго.

При усложнении задач обратная связь от внедрения к проектированию приобретает все большее наполнение, и со временем становится основной. Передача в рабочую эксплуатацию осуществляется условно, с замечаниями, а следующим этапом становится не завершение работ, а переход к разработке новой, улучшенной версии программы.

Внедрение и сопровождение смыкаются в одно целое, и в результате программист поддерживает развитие программы, постоянно корректируя ее на ходу, в процессе реальной эксплуатации.

Неопределенность результата

Первоначально предполагалось, что последовательное решение рутинных задач приведет к построению системы или ее первой очереди, которая полностью перейдет на сопровождение. На самом деле внедрение программ приводит к возникновению новой волны разработки, и процесс становится непрерывным и циклическим.

С одной стороны, внедренные задачи точно соответствуют техническому заданию. С другой стороны, пользователь выставляет все новые и новые требования к их модификации. Программисту приходится вновь и вновь возвращаться к сделанному и проводить корректировку и модификацию. План разработки программ выполняется, но с той же интенсивностью возникают новые задачи, поэтому идея разработки и внедрения всей автоматизированной системы становится все более сомнительной. Разработка превращается в непрерывный процесс, у которого есть начало, но нет конца.

2.3 Традиции управления

Причина зацикливания прямой технологии заключается в том, что она вступает во взаимодействие с традиционными механизмами управления.

Взаимодействие задач

Каждая задача, которая в прямой технологии рассматривается как законченная и самостоятельная, на самом деле находится в развитии и зависит от изменений во внешней среде. Вся система пребывает в состоянии равновесия до тех пор, пока не появляется внешнее воз-

мущение, приводящее к волне адаптации, распространяющейся по принципу домино. Точнее, система могла бы находиться в состоянии равновесия, если бы по ней постоянно не прокатывались многочисленные волны адаптации.

Первичное возмущение приходит из внешней среды. На входе системы появляется ситуация, не учтенная в алгоритме обработки. Работник, выполняющий задачу, пытается самостоятельно изменить алгоритм и справиться с проблемой. Иногда ему это удается, но в целом ряде других случаев он вынужден обращаться за помощью к руководству, а также предлагать решения, затрагивающие алгоритмы соседних задач, что отражено на рис. 10.

Из рисунка видно, что первичное возмущение затрагивает задачу непосредственно, но это вызывает вторичную волну, распространяющуюся уже внутри системы.

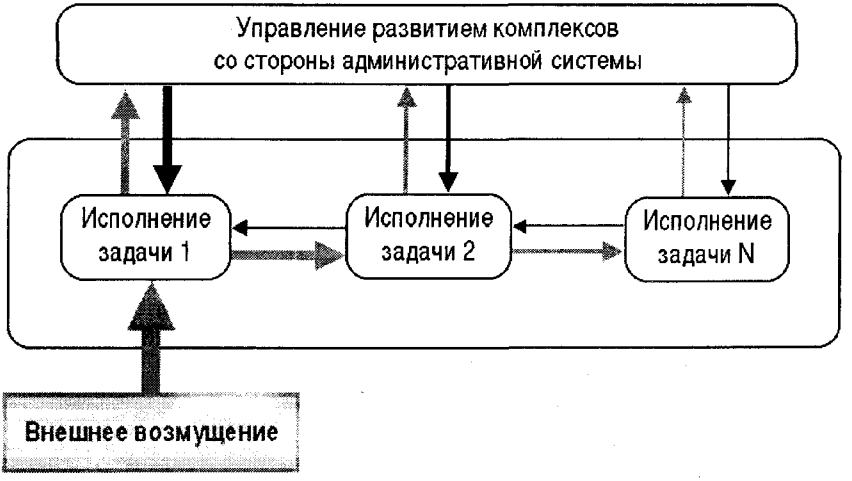

Рис. 10. Управление задачей в контексте административной системы

Любая задача корректируется не только по входной информации, но и в процессе взаимодействия с другими задачами. Очень часто проблема имеет несколько возможных решений, поэтому работники соседних подразделений не всегда приходят к единому варианту, и в этом случае согласованность решений обеспечивает руководитель.

Процесс адаптации

Несмотря на то, что руководитель подразделения не занимается непосредственной обработкой информации и не всегда вникает в тонкости рабочих алгоритмов, он играет определяющую роль в развитии задачи.

Во-первых, руководитель решает, подлежит ли новая входная ситуация решению в рамках данной задачи, или ее следует решать где-то в другом месте. Во-вторых, именно руководитель в одних случаях поддерживает предложения работника, а в других случаях – альтернативные предложения других подразделений.

Если задача давно находится в работе, она проходит адаптацию к самым разным изменениям на входе, и новые изменения практически не приводят к возникновению каких-либо вопросов. Волны не возникает, руководитель не вмешивается в процесс. Можно считать, что алгоритмы и структуры задачи в этом случае являются самодостаточными.

Если же задача эксплуатируется недавно, вероятность появления возмущающего воздействия довольно высока, но в алгоритмах и структурах еще нет ответа на вопрос, как следует реагировать на изменения. Задача не готова действовать в условиях возмущения, в ней отсутствует соответствующее формализованное описанние алгоритма и структуры. Оно появится после прохождения адаптационных волн, подключения руководителя и изучения реакции соседей.

Стихийное развитие

В целом развитие традиционного управления выглядит следующим образом. Система создается путем копирования готовых задач управления одного или нескольких заводов. Так называемое проектирование и формирование новой структуры представляют собой создание некоторой весьма простой компоновки из существующих блоков, уже проверенных на других предприятиях. Специалисты обобщают опыт, печатают книги, читают лекции в ВУЗах, но любое знание такого рода так или иначе возникло на действующем предприятии.

Затем включается механизм адаптации, который пытается снять нестыковки между задачами и погасить любую волну возмущений. Иногда это удается сделать без

особых проблем. Бывает и так, что волна не гаснет, а разрастается, усиливается. В этом случае административная надстройка реагирует на ситуацию радикально, вплоть до замены задач и работников, которые их выполняют. В этом случае задача заменяется прототипом, внесенным с другого предприятия либо непосредственно, либо через книги, стандарты и так далее.

Важно то, что в традиционной системе отсутствует механизм проектирования, критерий качества, контроль оптимальности алгоритмов, то есть элементы осознанной разработки. Вместо этого присутствуют наследственность, изменчивость и борьба за выживание. Традиционная схема рассматривает компьютерные комплексы как одну из реализаций решения задачи и проделывает с ними те же процедуры, что и с обычными прототипами. Отсюда и возникают циклы.

2.4 Автоматизация задач

Прямая технология не учитывает тех условий, в которые попадает рабочий комплекс, поэтому у него возникают проблемы с адаптацией.

Исходная информация

На этапе обследования разработчик не получает полных сведений о структурах и алгоритмах. Считается, что пользователь не может представить полный алгоритм задачи из-за неглубокого ее видения или из-за отсутствия необходимых навыков логического мышления. Однако пользователь ориентирован на циклическую технологию, поэтому он не составляет и не хранит полный алгоритм, учитывающий те ситуации, которые могут произойти, но еще не произошли.

В результате разработанная программа «правильно» выполняет ту часть алгоритмов, которая уже устоялась, а что касается другой части, то здесь все зависит от того, в какой степени разработчик смог предвидеть последующие события. Учитывая отсутствие прогноза по внешним возмущениям, отсутствие сведений о представлениях руководства, о составе и реализации других задач в системе, шансы угадать развитие событий близки к нулю.

Развитие задачи

Циклический процесс неминуем, но с появлением компьютерного комплекса он существенно усложняется,

так как в схеме появляется еще одна петля, связанная с разработкой программы (рис. 11). Мы видим, что помимо административной системы гашением волны теперь занят и разработчик. Административная система использует традиционные представления и технологию постепенного развития, в то время как разработчик опирается на информационную модель и стремится внедрить комплекс в один прием.

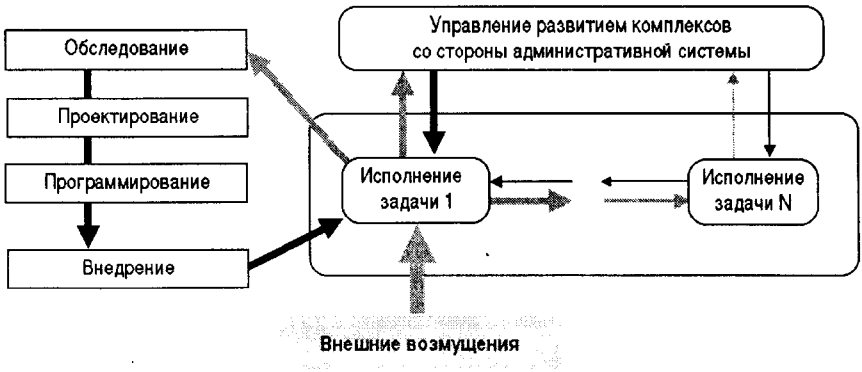

Рис. 11. Разработка и внедрение программного комплекса

По приходу внешних возмущений одна волна, как и прежде, распространяется в сторону руководителя, который принимает соответствующее решение. Другая волна попадает к разработчику и заставляет его корректировать информационную модель. Заметим, что представления руководителя, его понятийная платформа также являются своеобразной информационной моделью, отражающей структуру данных, алгоритмы и их взаимодействие.

Теперь в системе имеется две модели, построенные по различным технологиям, и каждая из них адаптирует задачу к изменениям во внешней среде.

Проблемы внедрения

Приступая к автоматизации рутинных задач, мы исходили из того, что компьютер справляется с их решением гораздо лучше, чем человек. Но оказалось, что задача должна не только выполняться, но и адаптироваться к

постоянным изменениям, и здесь компьютер оказывается не на высоте. Человек, обладающий прекрасными адаптационными возможностями, легко следует технологии постепенного наращивания задачи. Если это необходимо, он модифицирует структуру данных или придумывает алгоритмы на ходу, и тут же внедряет их в эксплуатацию.

Модификация компьютерного комплекса протекает гораздо сложнее, поэтому разработчик стремится решить всю задачу сразу. Он старается предвидеть изменения во внешней среде и придать своему комплексу необходимый запас прочности. Как правило, это обеспечивается за счет введения более сложной схемы, за счет введения новых понятий. В результате в системе управления внезапно появляется программный комплекс, содержащий множество новых решений, который вызывает «большую волну» в адаптационных цепочках.

Аналитик и пользователь делают одну и ту же работу, но со смещением во времени. Пользователь раскручивает процесс постепенно, мелкими шагами, наращивая новые режимы на действующую версию, выверяя каждый шаг на практике. Аналитик создает комплекс в один прием, сразу ориентируясь на целевой результат.

Его решения ориентированы на завтрашний день, поэтому они отличаются от сегодняшних решений и создают в системе «внутреннее возмущение». Оно прокатывается по системе и возвращается к комплексу в виде целого списка возражений, вопросов, замечаний и дополнений. Пользователю непонятно, каким образом будет осуществлен переход всей системы к новому решению, каким образом в переходный период должны работать другие задачи. Разработчику приходится возвращаться к задаче и обдумывать ее заново.

Внедрение системы

Учитывая взаимосвязь между задачами, АСУП в полном объеме будет закончена тогда, когда все они завершат итерационную адаптацию. Но появление каждой новой задачи заставляет систему пройти еще несколько циклов, полное завершение работ снова откладывается, а для поддержания действующих комплексов в удовлетво-

рительном состоянии затрачиваются средства, адекватные интенсивной разработке.

Положение усугубляется тем, что компьютер позволяет автоматизировать не только решаемые сегодня задачи, но и те, которые до сих пор не решались по причине сложности обработки информации в ручном варианте. Автоматизация расширяет перечень возможных алгоритмов и процессов, которые могут быть применены в системе управления. Поэтому, несмотря на трудности внедрения, пользователь активно требует продолжения работ, его аппетиты растут, первоначальные требования к задачам постоянно расширяются.

Учитывая сказанное, прямая технология уже не отвечает целям и задачам разработки АСУП. Планы, построенные с ее помощью, не выполняются, модели не соответствуют действительности, организация работ становится неэффективной.

2.5 Модификация технологии

Столкнувшись с зацикливанием, разработчики пытаются бороться с ним самыми различными способами. Исходная технология дополняется новыми идеями.

Пассивное копирование

Разработчик невысокой квалификации приходит к выводу, что зацикливание является неизбежным злом, и по этой причине не пытается с ним бороться. Приступая к автоматизации задачи, он ставит перед собой скромную цель, отталкиваясь от существующей схемы управления, и прежде всего от ее устоявшейся части. Модель разработчика в этом случае представляет собой существующую схему, переведенную на язык компьютера, без каких-либо изменений или дополнений новыми понятиями.

Пассивное поведение разработчика, соглашающегося с решениями руководителя, максимально сглаживает возмущающие воздействия. Осторожность в выборе задач также способствует этому. Но к сожалению, результат оказывается более чем скромным. Руководитель может убедиться, что компьютерный комплекс решает поставленную задачу, но адаптировать его к изменениям оказывается очень сложно. Если указание подчиненному

работнику внедряется в течение нескольких часов или дней, то модификация комплекса требует недель и месяцев.

Учитывая это, руководитель прибегает к модификации комплекса только в тех случаях, когда другие варианты оказываются явно неудовлетворительными. Со временем развитие подразделения все больше проводится путем адаптации неавтоматизированных задач. Появляются альтернативы, а компьютерный комплекс утрачивает свою актуальность, хотя и выполняет кое-какую полезную работу. Автоматизированной системы как таковой не возникает, отдельные локальные комплексы не приводят к ощутимому прорыву в управлении.

Использование полуфабрикатов

Другой подход к автоматизации связан с использованием прототипов, то есть готовых комплексов, уже внедренных на других предприятиях. Существует несколько вариантов копирования прототипов – от непосредственного изучения и повтора разработки до поставки готовых модулей и систем силами специализированных фирм. Суть от этого не меняется. Поскольку прототип уже прошел несколько циклов, его проще довести до внедрения, чем новую разработку.

Но есть и некоторые проблемы. Во-первых, такой прототип нужно найти. Если речь идет об автоматизации склада, то вариантов множество. Но если задача связана с конструированием уникальных изделий, то здесь возможностей гораздо меньше. Кроме того, несколько удачных прототипов, собранных с различных предприятий, очень часто не стыкуются между собой, поскольку они были изготовлены и внедрены в условиях различных концепций.

Пользоваться прототипами может только то предприятие, которое не претендует на лидерство в своей области. На их отработку требуется время, и такая стратегия позволяет использовать в АСУП методы вчерашнего дня, хотя для многих предприятий это допустимо.

Вывод – метод прототипов нередко приводит к хорошим результатам, но его внедрение обусловлено целым рядом требований, поэтому во многих случаях он оказывается неэффективным.

Жесткая реорганизация

Нередко разработчик заранее приходит к выводу, что существующую схему управления, при всех ее достоинствах, следует кардинально изменить. Использование компьютеров резко расширяет множество возможных вариантов, и в результате можно реализовать такие решения, которые в принципе не могут быть выполнены вручную.

Активная позиция предполагает серьезную реорганизацию существующей схемы управления [18]. Разработчик тщательно изучает ее и проектирует свой вариант, в котором помимо существующих режимов учтены и другие, расширяющие функциональные возможности схемы, улучшающие качество результата. В целом схема оптимизируется и получает логическое обоснование, из нее исключаются все неясности.

Внедрение превращается в битву с пользователем, поскольку речь идет о кардинальной замене его представлений. Разработчик уже не пытается разрабатывать одну версию программы за другой, он с самого начала активно агитирует пользователя, склоняя его к переходу на принципиально новую схему управления. При этом есть надежда, что перестройка не только упростит решение задачи, но и приведет к улучшению работы всей системы управления (скажем, увеличится оборачиваемость средств).

Характерной особенностью активного подхода является существование информационной модели, которая не столько отражает существующую ситуацию, сколько проектирует будущую систему, построенную на основе формальной логики и информационных технологий. Но разработчик даже не пытается показывать эту модель пользователю, поскольку ее чтение требует специальной подготовки. Работа с пользователем планируется отдельно от модели.

Позиция руководителя

Заметим, что окончательный успех разработки так или иначе связан с позицией руководителя и зависит от того, в какой степени он согласен с проводимыми мероприятиями, в какой степени они его устраивают, и как он реагирует на них. Поэтому рано или поздно разра-

ботчик выходит на прямой контакт с руководителем и *пытается согласовывать свои действия с ним.*

Рассмотрим рис 12. Руководитель получает информацию об изменении внешней среды и проблемах, возникающих при эксплуатации задач. Ему требуется скорректировать задачи, адаптировать данные и алгоритмы. Сделать это будет гораздо проще, если руководитель предварительно согласует свои представления о задаче с компьютерной моделью. В этом случае циклы все равно остаются, но в них уже не участвуют внешняя среда, программист и расчетчик.

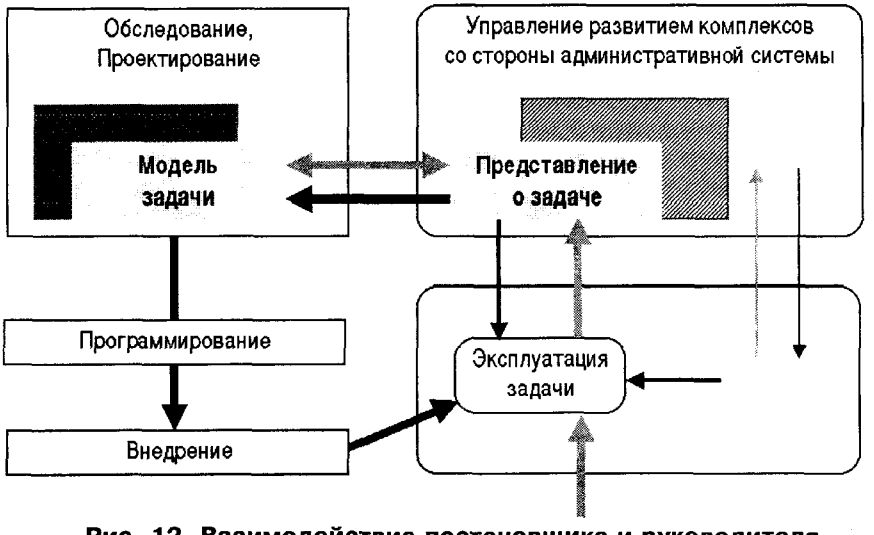

Рис. 12. Взаимодействие постановщика и руководителя

Обсуждение ведется непосредственно между руководителем и постановщиком, то есть разработчиком, который ведет компьютерную модель. Возникают все те же проблемы, предлагаются все те же решения, но все это происходит быстро, в процессе беседы. Вместо разработки и ликвидации программ «разрабатываются» и ликвидируются неудачные рисунки на бумаге.

В момент, когда приходят внешние возмущения, руководитель принимает решение с учетом результатов обсуждения компьютерной модели и с учетом плана раз-

работки программ. Другими словами, он либо принимает временное компромиссное решение и дожидается появления запланированной программы, либо понимает, что такая программа в ближайшее время не появится, и ищет альтернативное решение. Самое главное, что с появлением программы она не воспринимается как нечто новое, подлежащее длительной проверке.

Результат модификации

В рамках прямой технологии система управления стартовала от задач, исполняемых на нижнем уровне административной пирамиды. Но постепенно разработчики убедились, что успех внедрения определяется тем, в какой степени информационная модель, охватывающая сразу несколько задач, будет принята руководителем, направляющим работу расчетчика.

Как видим, ситуация полностью изменилась. Изменились субъекты, представление об элементе системы, появилась модель, а вместе с нею – новые методы обследования и согласования с руководством, о которых ничего не говорится в рамках прямой технологии.

Вывод – постепенная модификация изменила прямую технологию до такой степени, что от первоначальной концепции мало что осталось. Необходимо вернуться к самому началу и сформулировать новую концепцию, которая складывается по факту. Прежде чем это сделать, попробуем определить основные ее моменты.

2.6 Функциональный подход

В новой технологии важное место будут занимать представления руководителя о программном комплексе, который требуется разработать для управления подразделением.

Предварительное согласование

Разработка системы начинается с диалога между постановщиком и руководителем, в котором они согласовывают свои представления о задачах. Руководителю демонстрируют целый ряд новых представлений и решений, складывающихся в единую логическую схему. Демонстрационная версия системы, собранная на живую нитку, выполняет роль иллюстрации решений разработчика, а о внедрении даже не упоминается.

Получив новую информацию, руководитель приступает к ее осмыслению. Он пробует приложить решения разработчика к старой системе, постепенно осуществляя их адаптацию цикл за циклом. Все возникающие при этом вопросы периодически согласовываются. Система развивается в теоретических представлениях разработчика и пользователя, а до реального внедрения дело не доходит. Это существенно экономит усилия.

Стабильная основа

Отношения разработчика с руководителем оказываются более стабильными. Изменения во внешней среде уже не оказывают на них дестабилизирующего воздействия, поскольку разработчик с самого начала опирается на полную модель задачи, а не на ее конкретную реализацию, приспособленную к сложившимся условиям. Компьютерный комплекс, который слишком медленно адаптируется к изменениям, теперь уже не нужно переделывать ежедневно, что существенно повышает его привлекательность для пользователя.

Но тогда меняется наше представление о системе. Если раньше в качестве элемента рассматривалась задача, то теперь нужно взять за основу представления руководителя. В них данные и алгоритмы появляются как привязка к конкретным входным условиям некоторой более общей и более стабильной модели. В ее основе лежат функции и сущности (entity).

Функциональная устойчивость

Обычно задача управления, которую требуется автоматизировать, заключается в получении определенного результата. Способы, которыми будет организован сбор информации, а также алгоритмы, по которым будет рассчитан этот результат, могут меняться, на них не накладывается ограничений. Главное – результат. В качестве иллюстрации приведем пример конструкторской документации, плана производства или бухгалтерского баланса. Для предприятия важен сам факт их своевременного появления, а способы расчетов могут быть различными.

Такая устойчивость имеет функциональный характер. Управление осуществляет прямые или косвенные действия с ресурсами, участвующими в производственном

процессе (продукция, сырье, труд, оборудование, инструмент). По отношению к этим ресурсам должны быть выполнены типичные функции (прием, хранение, переработка, отправка). Отсюда появляются и косвенные понятия (технологический процесс, план, баланс, нормативный справочник).

На предприятии может появиться новый материал в группе сырья, может измениться упаковка изделия, может скорректироваться реестр аналитических счетов, но такие действия как покупка материала, упаковка продукции, проведение бухучета останутся. Следовательно, они являются наиболее устойчивыми, и на них следует ориентироваться при создании информационной модели и программных комплексов.

Технологии

Примерно из этих соображений выросли такие технологии, как структурный анализ [26], функциональный анализ [13] и множество других подходов, базирующихся на функциональном моделировании. Раньше мы базировали модель на структурах данных, под которыми можно понимать все что угодно, от простой таблицы до объектной схемы всего завода, а также на алгоритмах, которые также можно трактовать широко и неопределенно.

С переходом к функциональным представлениям мы начинаем работать с функциями, выстроенными в иерархическую систему. Каждая крупная функция расшифровывается с помощью более мелких. Каждая функция опирается на понятие сущностей, с которыми она работает [16]. Появляется возможность разговаривать с пользователем на языке более высокого уровня, а самое главное – этот язык ближе к языку пользователя.

Комплексная проработка функции

Теперь основной цикл переместился выше, на уровень «аналитик–руководитель» (рис. 13). Предметом обсуждения становится полностью завершенная функция, описанная в проекте и ориентированная на определенный вид входной или выходной информации.

Теперь колебания внешней среды, если они укладываются в исходные представления разработчика и руководителя, никак не могут нанести ущерб программе. Есть ответы на все вопросы – значит, программа будет рабо-

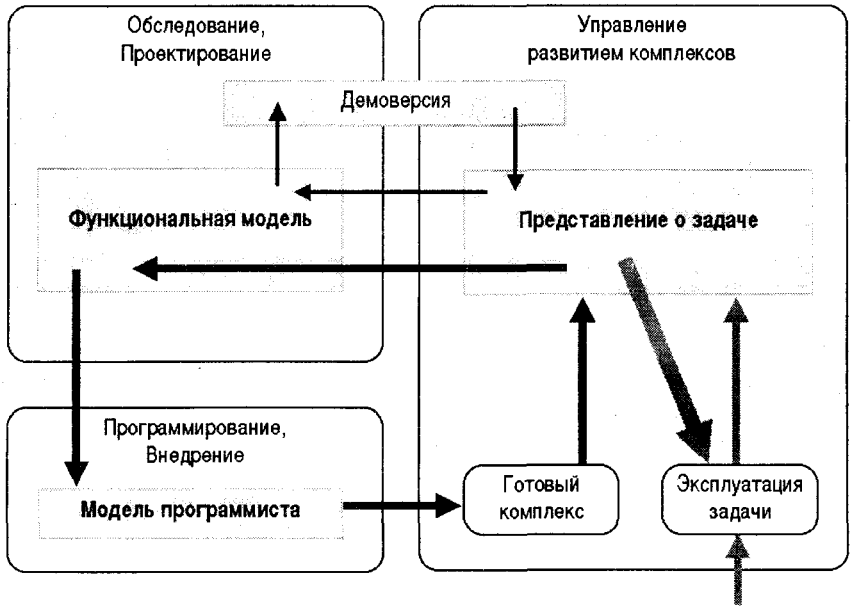

Рис. 13. Автоматизация функций административного подразделения

тоспособной при изменении внешних условий в довольно широких пределах. Более того, есть понимание, как можно объединять разрабатываемые комплексы в большие системы без дополнительных циклов адаптации.

Заметим, что руководитель и раньше понимал, как можно реализовать функцию в полном объеме, но без компьютера такая реализация представлялась ему невозможной или неэффективной. Поэтому ведением функции руководитель занимался сам, а непосредственные вычисления выполняли его подчиненные, которым сообщался упрощенный вариант расчетов, ориентированный на текущую внешнюю ситуацию. Компьютер позволяет реализовать функцию сразу, без перенастройки на частный случай, поэтому работать с ним гораздо проще, хотя требуется серьезно потрудиться на этапе внедрения.

Программирование

Мы видим, как технология постепенно разделилась на две фазы – составление аналитической модели и собственно программирование. О модели мы еще будем го-

ворить в следующей главе. Что же касается программирования, то оно теперь опирается на точную и определенную модель, не подверженную каждодневным изменениям. Программирование выполняется по прямой технологии. Этапы обследования и моделирования теперь выполняет системный аналитик, а программист получает схему или модель базы данных.

Несколько позже в инструментарии появляются генераторы экранных форм и отчетов. Поскольку задача программирования довольно точно специфицирована и формализована, возникают серьезные предпосылки для ее автоматизации. Программист постепенно переходит от кодирования к спецификации задачи, к настройке ее дизайна и интерфейса.

2.7 Некоторые итоги

Рассмотрев прямую технологию, выделим главные тенденции, которые проявляются на этапе перехода от алгоритмов к функциям.

Конструкция системы

Действительно, непосредственное управление предприятием осуществляется силами исполнителей, а их действия базируются на хорошо формализованных алгоритмах. Именно здесь и находится большинство рутинных задач, которые имеет смысл автоматизировать.

Вместе с тем, под влиянием изменений во внешней среде рутинные алгоритмы приходится постоянно модифицировать и согласовывать. Этим занят слой руководителей, действия которых базируются на функциональных представлениях.

Таким образом, система управления представляет собой двухуровневую конструкцию. Ее нижний слой обеспечивает функционирование системы, а верхний слой предназначен для решения вопросов развития (рис. 14). Автоматизация предприятия предполагает полную или частичную замену исполнителей на программные комплексы и автоматические процедуры.

Результаты внедрения

Попытка возложить на компьютер задачи исполнителя оказалась весьма удачной, поскольку человек не может конкурировать с компьютером по скорости и точности

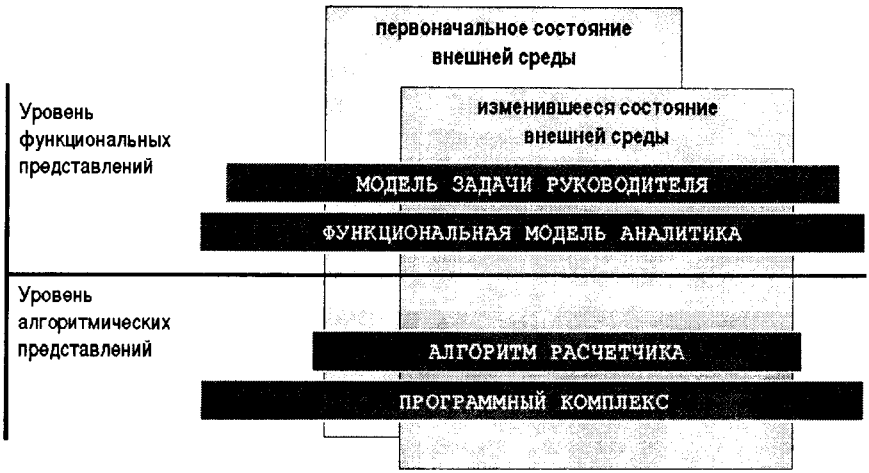

Рис. 14. Уровни моделей в административной и компьютерной системе

вычислений, а также по вопросам хранения больших массивов информации.

Но неожиданно выяснилось, что процессы развития системы с появлением компьютера существенно усложнились. Механизм развития, принятый в системе управления, был рассчитан на исполнителя, а он гораздо быстрее адаптируется к изменениям в окружающей обстановке, чем компьютерный комплекс.

В результате АСУП, построенная по прямой технологии, оказалась весьма привлекательной с точки зрения функционирования, но совершенно не приспособленной к развитию. Попытки исправить ситуацию не увенчались успехом, и тогда разработчики пришли к выводу, что для обеспечения развития АСУП нужно изменить конструкцию и технологию ее построения.

Человек и компьютер

Заметим, что любая задача рано или поздно дорастает до уровня функции. В старой системе этот процесс был организован на базе стихийного развития методом проб и ошибок, то есть методом восходящего синтетического развития. Учитывая высокие адаптационные способности человека, такой подход является естественным и наиболее рациональным.

Компьютер вообще не обладает способностью к адаптации, поэтому для него наиболее рациональным является аналитический путь развития. Это означает, что задача должна быть однократно осмыслена в полном объеме, а затем однократно реализована.

Руководитель, который привык иметь дело с людьми, осуществлял координацию задач синтетическими методами, передавая исполнителям оперативную информацию, поправки к существующей схеме, корректируя их поведение в частностях. Руководитель не был готов работать с компьютером, которому потребовалось передать всю информацию о задаче в виде функциональной схемы и полного описания всех режимов работы.

Роль постановщика

Тем не менее, такая информация у руководителя есть, хотя бы и в специфическом виде. Появляется постановщик, который выходит на уровень руководителя и формализует его представления в виде программно-ориентированной модели. Таким образом реализовалась аналитическая схема разработки, наиболее приемлемая для компьютера.

Постановщик активно работает на этапе составления модели и отработки программного комплекса, а затем ему остается осуществлять надзор за развитием событий. В идеальном случае, если представления руководителя действительно полны, а постановщик формализовал их без искажений, функциональная модель и программный комплекс вообще не требуют доработки.

В результате двухуровневая система управления сохраняется, руководитель продолжает управлять исполнителями, а что касается компьютерного комплекса, то он изначально готов к выполнению функции в полном масштабе. Со временем компьютерных комплексов становится все больше, а объем работы руководителя постепенно снижается.

Ниша

Несмотря на указанные недостатки, прямая технология не исчезла совсем. С нее начинают молодые специалисты, с нее начинаются новые системы. Подобно тому, как каждый малыш учится ходить, каждый молодой специалист начинает строить систему с прямой технологии.

Максимум, что здесь можно сделать – это организовать ускоренное прохождение прямой технологии на лабораторных занятиях в ВУЗе, чтобы не выносить ее на этап практической деятельности на предприятии.

Отметим также, что существуют автономные задачи (даже в рамках АСУП, не говоря уже о других системах), в которых взаимодействие с партнерами и проблемы координации этого взаимодействия сведены к минимуму. В этом случае применение прямой технологии вполне возможно и даже в какой–то степени оправдано.

Характерные признаки

В силу различных причин прямая технология иногда ошибочно применяется и для разработки больших систем, хотя ее время уже прошло. Нам важно различать внешние признаки, по которым мы всегда узнаем прямую технологию.

Первый признак – разделение команды исполнителей на постановщиков и программистов и четко оформляемая документация на стыке между ними.

Второй признак – отношения с пользователем устанавливаются на уровне исполнителей, а не руководителей.

Третий признак – быстрая смена версий программного обеспечения. Точнее, настолько быстрая смена версий, когда понятие версии исчезает совсем, а программа изменяется ежедневно, постепенно «переползая» из старого состояния в новое.

Вывод

На смену прямой технологии, в основе которой лежит идея изучения отдельных задач, приходит функциональный подход. В нем каждая задача рассматривается как одна из возможных реализаций некоторой устойчивой функции, входящей в структуру системы. Задачи появляются и уходят, изменяются и адаптируются, а функциональный скелет системы остается практически неизменным.