

Л.А. Лыноградский

АНАЛИЗ И РАЗВИТИЕ ТРАДИЦИОННОГО ПОДХОДА К АВТОМАТИЗАЦИИ УПРАВЛЕНИЯ ПРЕДПРИЯТИЕМ

Рассматривается задача целенаправленного развития структуры предприятия в контексте внешнего окружения. Показаны слабые места традиционного подхода, опирающегося на административную иерархию процессов управления. Предложен подход, позволяющий эффективнее использовать возможности вычислительной техники.

На протяжении нескольких десятков лет во всем мире ведутся работы по созданию и развитию автоматизированных систем управления предприятием. Несмотря на огромные вложения в решение этой проблемы, сегодня не имеется технологии, позволяющей гарантировать успешное выполнение очередного проекта. Треть проектов оказывается проваленной, другая треть выполняется с существенными нарушениями сроков, сметы и технического задания, и только оставшаяся треть завершается успешно [1].

Внешние факторы. Анализируя причины провалов, мы не можем свести их исключительно к низкой квалификации разработчиков или недостаткам технологии. Имеется немало случаев, когда одна и та же команда успешно выполняет некоторый проект, а затем проваливает следующий, совершенно аналогичный. Это означает, что на результат проекта влияют не только факторы, учтенные в технологии разработки, но и другие, возникающие во внешнем окружении, которые способствуют или препятствуют успешному ходу работ.

Для повышения качества проекта, для уменьшения риска его выполнения разработчик должен отразить в модели объекта управления все существенные факторы, влияющие на него, а также оценить их количественно. Это позволяет строить точные прогнозы и подбирать технологии, наилучшим образом соответствующие поставленной задаче.

К сожалению, каждый опытный разработчик рано или поздно осознает несовершенство методов, которыми он пользуется в процессе моделирования. Несмотря на тщательное изучение связей объекта с другими объектами инфраструктуры, избежать непредсказуемых внешних возмущений со стороны последних все-таки не удается. Возникают гипотезы, объясняющие механизм действия внешних факторов, в основе которых лежит широкий спектр моделей – от технических до социально-психологических [2]. К сожалению, гипотезы не подтверждаются, мероприятия на их основе оказываются неэффективными, а риск провала проектов остается недопустимо высоким.

Системная модель. Предположим для определенности, что проект направлен на оптимизацию работы предприятия. Заказчик, исходя из выбранного им критерия, ставит задачу перевода предприятия из текущего состояния S_0 в некоторую область, расположенную вблизи целевого состояния S_g . Для простоты изложения допустим, что качество результата обратно пропорционально расстоянию до S_g . Поведение объекта

$$\Delta S = S_g - S_0 = F(a, b, c, \dots, l, m, n, \dots) \quad (1)$$

зависит от ряда факторов, только часть из которых - a, b, c, \dots - может меняться по желанию разработчика. Другая часть - l, m, n, \dots - отражает состояние внешних объектов за пределами предприятия, а потому ее можно лишь пассивно наблюдать или прогнозировать.

Несмотря на то, что факторами l, m, n, \dots разработчик не управляет, он может учесть их нежелательное влияние на общий результат и скомпенсировать его путем маневрирования решениями a, b, c, \dots . Тогда для выбора конкретных решений используется зависимость

$$\{a, b, c, \dots\} = Q(S_g, S_0, \Delta l, \Delta m, \Delta n, \dots) \quad (2)$$

Эта зависимость позволяет увидеть, как относятся различные разработчики к проблеме системного моделирования и какие при этом возникают результаты. Наиболее простой подход – игнорирование всех $\Delta l, \Delta m, \Delta n, \dots$, фиксация текущих условий без какой-либо попытки прогнозирования. Другой подход – экспертиза возможных отклонений по каждому фактору, попытка построения системы под условия конечного состояния. Третий подход – обеспечение гибкости системы, ее способности реагировать на изменение того или иного фактора путем заранее предусмотренной настройки.

Не обсуждая достоинств и недостатков этих подходов, отметим, что на практике чаще всего используется комбинированный вариант, в котором предполагается, что часть факторов l, \dots в процессе проектирования останется неизменной, другая часть m, \dots изменится определенным образом, а третья n, \dots будет изменяться непредсказуемо. Экспертиза и прогнозирования факторов в большинстве случаев выполняются силами разработчика на этапе обследования объекта и ставят его в сложное положение.

В самом деле, прямое изучение и прогнозирование всех внешних факторов в силу своей сложности оказывается неприемлемым. Теоретические представления, позволяющие априорно исключить те или иные факторы, у разработчика отсутствуют. Интуиция разработчика (а возможно, и случай) приводят к выбору тех или иных вариантов модели (2). От этого выбора в конце концов и зависит судьба проекта. Если в процессе выполнения работ активизируются как раз те факторы, которые были учтены в модели, проект выполняется успешно. Если активизируются факторы, не вошедшие в модель, проект ждет та или иная степень провала либо, в лучшем случае, необходимость коренной переработки.

Эволюция структуры. Проблема исследования внешних факторов заставляет нас вернуться к первоначальной постановке задачи и «прочитать» ее глубже. В самом деле, необходимость перевода предприятия из состояния S_0 в состояние S_g уже содержит некоторую информацию. Очевидно, здесь идет речь об укреплении структуры, о повышении конкурентоспособности, об увеличении устойчивости. Все эти частные задачи заказчик намерен решать сам, без помощи разработчика. Другими словами, он намерен заниматься параметрической оптимизацией по известному ему критерию. Разработчику необходимо лишь реорганизовать структуру управления под требования заказчика.

Действительно, текущее состояние предприятия определяется его структурой, то есть совокупностью состояний элементов $\{E_i\}$ и связей $\{R_{ij}\}$. Предположим, разработчик отказывается от каких-либо активных действий. Тогда под влиянием внешних факторов объект перейдет в некоторое состояние S^* , которое определяется новым состоянием элементов $\{E_i^*\}$ и связей $\{R_{ij}^*\}$, а его поведение будет выглядеть следующим образом:

$$\Delta S'' = S^* - S_0 = S(E_1^*, E_2^*, \dots, R_{12}^*, R_{13}^*, \dots) - S(E_1, E_2, \dots, R_{12}, R_{13}, \dots) = F(l, m, n, \dots) \quad (3)$$

Мы видим, что структура объекта изменяется. Это выражается в потере одних элементов, приобретении других, а также в развитии их взаимодействия, т.е. изменении состава и характера связей. Можно сказать, что поведение i -го элемента в составе системы S при определенном внешнем окружении (исключая форс-мажор), выглядит следующим образом:

$$\Delta E_i = [E_i^* - E_i] = F_i(l, m, n, \dots) \approx U_i(m, n, \dots) \quad (4)$$

Заметим, что из всего множества внешних факторов только часть из них реально действует на i -й элемент. Это позволяет использовать вместо соответствующей функции F_i упрощенную функцию U_i практически без потери точности. Заметим также, что новое состояние элемента E_i^* может означать его ликвидацию или переход в другую систему. Для других элементов исходное состояние E_j может означать существование вне системы.

Действительно, под воздействием внешних факторов элемент не только меняет свое состояние и связи, но и может переходить от одного объекта к другому. В тех случаях, когда речь идет о движении ресурсов по цепочке «сырье-продукция», обмен элементами проводится по инициативе предприятия и не влияет на качество структуры. В других случаях потеря важного элемента или приобретение «балласта» являются нежелательными.

Разработчик, изучив механизмы стихийного развития структуры, вмешивается в них и принимает решения, позволяющие удерживать важные элементы и избавляться от бесполезных. Эти решения заключаются в выборе определенных воздействий на элемент:

$$\{ a, c, \dots \} = Q_i [E^*_i, E_i, \Delta m, \Delta n, \dots] \quad (5)$$

Характер решений, принимаемых разработчиком, существенно зависит от природы элемента и находится в широком спектре между утилизацией отходов производства и премированием лучших работников.

Устойчивость системы. Для того, чтобы система была устойчивой, она должна удерживать определенные позиции во внешней среде, возвращаться в состояние равновесия после возмущающего воздействия со стороны других систем [3]. Уже на этапе постановки заказчик, анализируя текущее состояние объекта в сравнении с аналогичными объектами, отмечает отсутствие некоторых элементов, компонент, связей и алгоритмов, которые есть у конкурентов и которые желательно ввести в структуру объекта. И только потом проводится сравнительный анализ параметров, характеризующих качество работы элементов и связей.

Таким образом, проект решает две главные задачи: формирование нового состава элементов и обеспечение их эффективной работы. Вторая задача проще, поскольку в ней точно определено множество элементов. Сложность первой задачи связана с анализом и выбором вариантов из бесконечного множества возможных. Поэтому формирование обновленного состава элементов – это необходимое, хотя и не достаточное условие успеха проекта.

Решая структурную задачу, разработчик составляет множество уравнений для каждого элемента системы. Пусть, для простоты, он имеет два уравнения:

$$\begin{aligned} \{ a, c, \dots \} &= Q_i [E^*_i, E_i, \Delta m, \Delta n, \dots] \\ \{ b, c, \dots \} &= Q_j [E^*_j, E_j, \Delta l, \Delta p, \dots] \end{aligned} \quad (6)$$

Любой комплект решений a, b, c, \dots , удовлетворяющий системе уравнений, теоретически позволяет обеспечить развитие элементов в нужную сторону. Но решения должны находиться в диапазоне физически допустимых и экономически оправданных значений.

Выбор общего решения c проводится таким образом, чтобы обеспечить возможность маневрирования решениями a и b , то есть возможность их изменения в ту или иную сторону в случае необходимости. Тогда для управления любым элементом системы необходимо рассчитать и реализовать соответствующее частное решение:

$$\begin{aligned} a &= H_i [E^*_i, E_i, \Delta m, \Delta n, \dots, c] \\ b &= H_j [E^*_j, E_j, \Delta l, \Delta p, \dots, c] \end{aligned} \quad (7)$$

Разумеется, с развитием системы S выражения в правых частях постепенно изменяются, но если решения a и b могут варьироваться в широких пределах, то контроль над поведением элементов остается в руках разработчика.

Административное управление. В реальной системе количество элементов огромно, поэтому комплексное решение задачи оказывается невозможным. Проект направлен на реорганизацию наиболее слабой части структуры, выявленной в результате предварительного анализа. В нее входят только некоторые из элементов (в рассматриваемом примере - один):

$$\{ a, c, \dots \} = Q_i [E^*_i, E_i, \Delta m, \Delta n, \dots] \quad (8)$$

Выбирая и реализуя решения a, c, \dots , разработчик невольно усложняет задачу выбора b :

$$b = H_j [E^*_j, E_j, \Delta l, \Delta p, \dots, c] \quad (9)$$

При выполнении следующего проекта становится ясно, что можно было бы несколько изменить a, c , и тогда комплексное решение было бы более эффективным. Но к этому моменту первый проект уже завершен, а изменение его решений связано с серьезными затратами средств и времени. В результате последовательного выполнения ряда проектов совокуп-

ность локальных решений уходит далеко от оптимума, который мог бы быть найден в случае комплексного подхода.

В идеальном случае каждое решение должно находиться в середине допустимого диапазона, чтобы разработчик в случае необходимости имел свободу маневра. Более точно, изучение технических и экономических характеристик каждого решения позволяет определить его предпочтительное значение, а также штраф Z за отклонение от оптимума, приведенный к некоторой общей единице измерения. Можно было бы определить качество комплексного решения в виде

$$K = (Za, Zb, Zc...) \rightarrow \min \quad (10)$$

Проблема в том, что в силу сложности комплексной задачи мы вынуждены выполнять локальные проекты, а они не позволяют принимать полный спектр решений одновременно. Решения принимаются последовательно, одно за другим, что со временем ухудшает значение K . Здесь просматривается определенная аналогия с методом покоординатного спуска со всеми его недостатками.

Для решения этой проблемы существует административная пирамида, в которой частные решения принимаются на нижних уровнях, а глобальные – на верхних. Для локального проекта оптимизируется значение критерия

$$K_i = (Za, Zc_i...) \rightarrow \min \quad (11)$$

Другими словами, каждый локальный разработчик волен выбирать общее решение c так, как ему удобно, не задумываясь над последствиями. Но после этого он должен согласовать его с руководством.

Руководители верхнего звена уже не занимаются выбором частных решений a и b . Они получают множество значений c_i, c_j, \dots и начинают их согласование, то есть выбор единого решения c . При этом преследуется цель оптимизации значения K . Процедура согласования решений выходит за рамки рассматриваемой задачи. Важно, что административная пирамида в основном занимается задачей согласования решений, что прямо влияет на качество функционирования предприятия [4].

Автоматизация управления. Выполнение информационного проекта связано с развитием подсистемы, которую мы обозначим P_i . Однако и для нее справедливы рассуждения, которые были приведены выше. В процессе создания новой подсистемы принимаются решения, часть которых a, c, \dots мы отнесем к административным, поскольку они связаны с сущностью выполняемых функций. Другую часть x, y, \dots отнесем к технологическим, определяющим архитектуру и аппаратно-программную реализацию подсистемы:

$$\{ a, c, \dots, x, y, \dots \} = Q_i [P_i^*, P_i, \Delta m, \Delta n, \dots] \quad (12)$$

Пусть решения a и x являются локальными и принимаются в рамках данной подсистемы. Решения c, \dots и y, \dots являются общими для нескольких подсистем. После переноса их в правую часть получим локальные решения:

$$\begin{aligned} a &= H_i [P_i^*, P_i, \Delta m, \Delta n, \dots, c, \dots, x, y, \dots] \\ x &= H_i [P_i^*, P_i, \Delta m, \Delta n, \dots, a, c, \dots, y, \dots] \end{aligned} \quad (13)$$

Заметим, что решение данной системы осуществляется двумя субъектами. За решение первого уравнения отвечает заказчик (пользователь), за решение второго – разработчик. Традиционный подход к построению подсистем предполагает ведущую роль заказчика, который и должен проводить оценку внешних факторов. Если принять первое уравнение за главное, а второе – за вспомогательное, получим упрощенную схему поиска решения:

$$\begin{aligned} a &= H_i [P_i^*, P_i, \Delta m, \Delta n, \dots, c, \dots, x] \\ x &= H_i [P_i^*, P_i, a, y, \dots] \end{aligned} \quad (14)$$

Полученная система описывает качественно процесс разработки нового программного продукта. Но проходит время, внешние факторы изменяют свое состояние, меняются и об-

шие решения (как административные, так и технологические). Возникает необходимость адаптировать подсистему к новым условиям. Здесь выясняется, что адаптация автоматизированной подсистемы оказывается более сложной, чем неавтоматизированной. Если раньше работнику требовалось лишь уточнить новую задачу, то теперь нужно еще переделать программу, что выливается в серьезные затраты средств и времени.

Усложняется и процесс согласования решений. Теперь как на нижнем, так и на верхнем уровне имеются уже по два субъекта, каждый из которых отвечает за соответствующие решения. Руководитель верхнего уровня согласовывает варианты c_i, c_j, \dots общих решений. Работник нижнего уровня занимается поиском частных решений a и согласованием их с общими решениями, а теперь еще и с решениями программиста. Программист ищет частные технологические решения x , согласуя их с общими технологическими, а главный архитектор информационной системы согласовывает частные решения программистов y_i, y_j, \dots :

$$c_i, c_j, \dots \leftrightarrow a \leftrightarrow x \leftrightarrow y_i, y_j, \dots \quad (15)$$

Возникают две архитектуры, которые взаимодействуют через множество частных решений нижнего уровня, а потому развиваются в каком-то смысле независимо друг от друга. До тех пор, пока автоматизация еще не распространилась на значительную часть рабочих процессов, технологическая архитектура является слабой и не может серьезно влиять на результаты работы предприятия. В перспективе тотальной автоматизации она должна стать основной. Где-то на этом пути возникает состояние паритета и кризиса, когда старая архитектура уже потеряла свою монополию, а новая еще не захватила ее [5].

Разрешение кризиса. Заметим, что мы стартовали от системы уравнений, в которой административная и технологическая части были равноправны. Затем мы отдали инициативу заказчику и поставили программиста в зависимое от него положение. Попробуем поступить наоборот, то есть поставить заказчика в положение, зависимое от программиста. Упрощая соответствующие уравнения, получим:

$$\begin{aligned} x &= H_i [P^*_i, P_i, \Delta m, \Delta n, \dots, a, c, \dots, y, \dots] \\ a &= H_i [P^*_i, P_i, x] \end{aligned} \quad (16)$$

Теперь заказчик, который раньше выполнял роль постановщика и занимался согласованием решений с руководством, превратился в простого оператора. Разумеется, речь не идет об операторе, способном вводить только тексты и числа. Возможно, это эксперт, прогнозирующий развитие курса валют или технолог, выбирающий тот или иной стандарт оформления документации. Важно то, что теперь программист (точнее, разработчик) взял на себя функции согласования решений с верхним уровнем управления. Учитывая то, что на верхнем уровне произошло то же самое, получим новую схему согласования.

$$a \leftrightarrow x \leftrightarrow y_i, y_j, \dots \leftrightarrow c_i, c_j, \dots \quad (17)$$

Теперь обеспечен непосредственный контакт между архитектурами верхних уровней, а вертикальное согласование проводится в рамках информационной службы между разработчиком и главным системным аналитиком (рис. 1). Для этого необходимо расширить понятие технологической архитектуры далеко за рамки аппаратно-программных решений. Те-

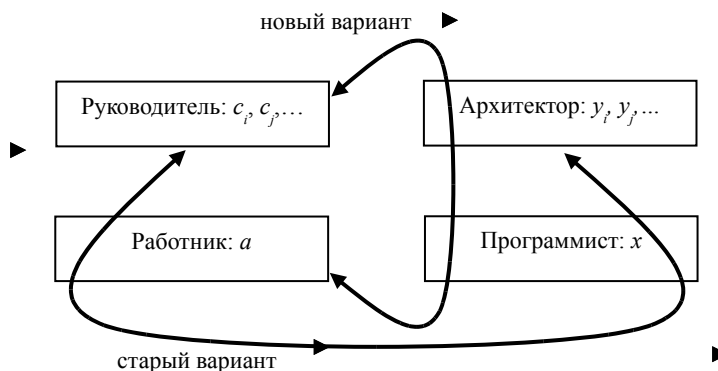


Рис. 1. Схема согласования проектных решений

перь решения информационной системы базируются на теории и технологии управления предприятием.

Заметим, что эта схема показывает, что руководство предприятия не потеряло власть, но получило возможность заниматься творческой работой, искать решения, проводить наполнение реальным содержанием того стандарта автоматизированного управления, которое предложено со стороны информационной службы.

Управление проектами. Отметим, что в ходе разработки схем вскрыта причина провала проектов. Она связана с тем, что процессы их создания находятся в руках программистов, а процессы развития переключаются на администрацию. Администрация могла бы управлять развитием автоматизированной системы, но это не предусмотрено конструкцией комплексов, выполненных по жесткой схеме.

Конкретные заказы, которые формирует административная система, предполагают создание гибкого продукта, а потому предлагается строить информационную систему на основе теории и технологии управления, и только потом настраивать ее на конкретные требования заказчика. Разумеется, для разработки теории и технологии потребуются неординарные усилия, но в этом случае мы получаем универсальный результат, который можно применять на всех или на многих предприятиях.

Для того чтобы реализовать эту идею, осталось сделать еще два шага. Первый связан с необходимостью сращения технологической структуры с административной. В [4,5] показано, что каждый уровень административной пирамиды использует свою технологию управления. Три нижних уровня уже нашли свое отражение в простейших алгоритмах, функциональной и объектно-ориентированной технологии разработки. Но существуют еще два уровня, где применяются технологии, отсутствующие в структуре информационной системы. Именно они связаны со спецификацией архитектурных решений и согласованием их с административными понятиями.

На верхних уровнях идет постоянный процесс обеспечения баланса структуры. Он определяет подчиненность процессов нижних уровней, структурирует их в виде своеобразной пирамиды, а также следит за наполнением уровней соответствующими средствами. Увлечение верхними уровнями управления позволяет создать высокоорганизованную, но неустойчивую схему управления. Увлечение нижними ведет к потере высокой эффективности, хотя устойчивость в этом случае даже избыточна. Необходимость обеспечения баланса подтверждается и общесистемными соображениями [6].

Второй шаг связан с тем, что компьютер обладает вычислительной мощностью, существенно превышающей мощность человека-вычислителя. Комплексы, построение которых ориентировано на человека, оптимальны с точки зрения вычислительных усилий, но сложны в построении и развитии. Переход к сборным конструкциям из простых типовых модулей может привести к увеличению загрузки компьютера (который сегодня на 90 процентов простаивает), но при этом упрощаются процессы адаптации такой конструкции.

Идея перехода от сложных комплексов к конструкциям заключается в следующем. Уравнение (16) может быть преобразовано в вычислительный конвейер, где на каждом шаге обрабатываются вопросы в плоскости, связанной с соответствующим внешним фактором. В результате получаем целый ряд процессов

$$\begin{aligned} x'' &= H_i [P''_i, P_i, \Delta m, a, c, \dots, y, \dots] \\ x''' &= H_i [P'''_i, P''_i, \Delta n, \dots, a, c, \dots, y, \dots, x''] \\ x'''' &= H_i [P^*_i, P'''_i, \dots, a, c, \dots, y, \dots, x'''] \end{aligned} \quad (18)$$

Сложный вектор $[P^*_i, P_i]$ раскладывается на составляющие, соответствующие направлениям осей координат, которые, в свою очередь, соответствуют отдельным внешним факторам. Каждое из решений x'' , x''' и x'''' позволяет управлять подсистемой по одному из направлений, а их композиция с определенными весами и приводит к появлению необходимого результирующего вектора.

Такой подход усложняет описание подсистемы, существенно увеличивает загрузку компьютера, но упрощает процедуры развития. В самом деле, при появлении нового фактора достаточно добавить в конвейер еще одну строку и разработать еще один простой комплекс. В том случае, когда фактор перестает действовать, достаточно вычеркнуть соответствующую строку и удалить комплекс.

Идея создания типового программного комплекса, заранее предназначенного для взаимодействия с другими типовыми комплексами, кажется заманчивой. Эволюция таких элементов ведется под руководством аналитиков, снабженных специальными инструментальными средствами ведения модели системы. Рабочие комплексы получают спецификацию непосредственно из комплексов управления проектами, а также сообщают последним некоторые характеристики, касающиеся их работы (объемы данных, скорость обслуживания заявки и многое другое, необходимое для построения эффективной структуры).

Учитывая тенденции в общем развитии живой природы и технических средств, мы можем полагать, что в недалеком будущем такие универсальные комплексы могут появиться и постепенно вытеснить традиционных «монстров», которые уже достаточно убедительно показали свою неспособность решить задачу автоматизации управления предприятием.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Павлов В.Л. Microsoft Solutions Framework на предприятиях и в вузах. <http://research.microsoft.com/collaboration/university/europe/events/rcc/russsia>
2. Йордон Э. Путь камикадзе. Как разработчику программного обеспечения выжить в безнадежном проекте. Пер. с англ. – М.: ЛОРИ, 2001. – 255 с.
3. Волкова В.Н., Денисов А.А. Основы теории систем и системного анализа. – СПб: Издательство СПбГПУ, 2003. – 520 с.
4. Лыноградский Л.А. Концепция системного проектирования. – Самара: СамГТУ, 2005. – 180 с.
5. Лыноградский Л.А. Горизонты системного анализа. – Самара: ИЭКА Поволжье, 2000. – 244 с.
6. Кудрин Б.И. Применение понятий биологии для описания и прогнозирования больших систем, формирующихся технологически // Электрификация металлургических предприятий Сибири. – Томск, 1976. – Вып. 3. – С. 171-204.