Министерство образования и науки Российской Федерации
Уральский федеральный университет
имени первого Президента России Б.Н. Ельцина

О.Г. Инюшкина

Проектирование информационных систем
(на примере методов структурного системного анализа)

Учебное пособие

Научный редактор Матвеева Татьяна Анатольевна

Екатеринбург
Издательство «Форт-Диалог Исеть»
2014

УДК 519:6 + 658.01 + 681.3 + 004.8
ББК 65.290 – 2я73 + 65.5ф
И 748

Рецензенты:
проф., д-р пед. наук Л.И. Долинер (Уральский технический институт связи и информатики (филиал) ФГОБУ ВПО «Сибирский государственный университет телекоммуникаций и информатики», г. Екатеринбург);
д-р физ.-мат. наук В.В. Прохоров (НПЦ «Видикор», г. Екатеринбург)

Инюшкина О.Г.

И 748 Проектирование информационных систем (на примере методов структурного системного анализа): учебное пособие / О.Г. Инюшкина, Екатеринбург: «Форт-Диалог Исеть», 2014. 240 с.

ISBN 978-5-91128-072-7

Учебное пособие предназначено для теоретического освоения методов и средств проектирования информационных систем, применения на практике методологий структурного анализа и проектирования. Пособие содержит теоретический материал по дисциплинам направления 230400 «Информационные системы и технологии».

Библ.: 20 назв., табл. 9, рис. 59

УДК 519:6 + 658.01 + 681.3 + 004.8
ББК 65.290 – 2я73 + 65.5ф

ISBN 978-5-91128-072-7

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Учебное пособие посвящено методам и средствам проектирования информационных систем.

Создание ИС – это логически сложная, трудоемкая и длительная работа, требующая высокой квалификации участвующих в ней специалистов. Но нередко создание таких систем выполняется на интуитивном уровне с применением неформализованных методов, основанных на искусстве, практическом опыте, экспертных оценках и дорогостоящих экспериментальных проверках качества функционирования системы. Эксплуатационные расходы, возникающие после сдачи таких систем, могут существенно превышать расходы на их создание. Исследования показывают, что на обнаружение ошибок, допущенных на стадии проектирования, расходуется примерно в два раза больше времени, чем на исправление ошибок, допущенных на последующих фазах. При этом исправление ошибки на стадии проектирования стоит в 2 раза, на стадии тестирования – в 10 раз, а на стадии эксплуатации системы – в 100 раз дороже, чем на стадии анализа. Кроме того, ошибки анализа и проектирования обнаруживаются часто самими пользователями, что вызывает их недовольство и осложняет сопровождение ИС.

При этом выбор методологии проектирования является далеко не простой задачей. В гл. 1.6. дан сравнительный анализ структурных (процессно-ориентированных) и объектно-ориентированных парадигм. Структурные методологии и их применение на практике подробно описаны в разделе 4.

Учебное пособие предназначено для формирования соответствующих компетенций, необходимых для успешной реализации, внедрения и эксплуатации информационных систем. Материалы книги могут быть использованы студентами и преподавателями в рамках направления 230400 «Информационные системы и технологии» высшего профессионального образования.

1. ОСНОВНЫЕ ПОНЯТИЯ ПРОЕКТИРОВАНИЯ ИНФОРМАЦИОННЫХ СИСТЕМ

1.1. Основные понятия общей теории систем (ОТС)

История возникновения ОТС

В настоящее время знания человека о природе разрослись до такой степени, что не представляется возможным охватить не только весь их объем, но и отдельные области. При этом для создания полезных знаний и продуктов необходимы сведения из смежных областей. Теория систем призвана помочь человечеству в преодолении недостатков узкой специализации, усилении междисциплинарных связей, развитии диалектического видения мира, системного мышления.

Теория систем впервые была применена в точных науках и технике как вклад школы науки управления. Как самостоятельная дисциплина теория систем оформилась в 40-50-х годах XX века. Системный анализ со временем стал меж- и наддисциплинарным курсом, обобщающим методологию исследования сложных технических и социальных систем, а также представляет собой наиболее надежную концептуальную основу современного менеджмента.

Специфической чертой социальной роли науки в настоящее время является направленность научного познания в целом на создание эффективных средств управления как природными, так и социальными процессами.

Одно из первых открытий, сделанных философами (Б.Трентовский), заключалось в том, что действительно эффективное управление должно учитывать все важнейшие внешние и внутренние факторы, влияющие на объект управления. При этом главная сложность управления связана, по со сложностью поведения людей.

Следующее открытие (А.А. Богданов) заключалось в следующем. Все

существующие объекты и процессы имеют определенную степень, уровень организованности. Все явления рассматриваются, как непрерывные процессы организации и дезорганизации. При этом уровень организации тем выше, чем сильнее свойства целого отличаются от простой суммы свойств его частей.

Широкое признание теории, осознание системности мира началось в 1948 году после публикации американским математиком Н.Винером книги «Кибернетика». Первоначально он определяет кибернетику как «науку об управлении и связи в животных и машинах» (аналогии процессов в живых организмах и машинах), позже анализирует с позиций кибернетики процессы, происходящие в обществе.

С кибернетикой Винера связаны такие продвижения, как типизация моделей систем, выявление особого значения обратных связей в системе, подчеркивание принципа оптимальности в управлении и синтезе в целом, осознание информации как всеобщего свойства материи и возможности ее количественного описания, развитие методологии моделирования вообще и, в особенности идеи математического эксперимента с помощью ЭВМ.

Параллельно и независимо от кибернетики прокладывается еще один подход к науке о системах – общая теория систем. Выдвигается идея построения теории, применимой к системам любой природы (австрийский биолог Л.Берталанфи). Один из путей реализации этой идеи – поиск и обобщение структурного сходства законов, установленных в различных дисциплинах. В отличие от предыдущего подхода (Винер), где изучаются внутрисистемные обратные связи, а функционирование систем рассматривается просто как отклик на внешнее воздействие, данный подход подчеркивает особое значение обмена веществом, энергией и информацией с открытой средой.

Отправной точкой общей теории систем как самостоятельной науки можно считать 1954 год, когда было организовано общество содействия развитию общей теории систем. Указывается основная причина появления новой отрасли знания:

Существует общая тенденция к достижению единства различных есте-

ственных и общественных наук. Такое единство может быть предметом изучения ОТС. Эта теория может быть важным средством формирования строгих теорий в науках о живой природе и обществе. Все это может приблизить к достижению единства науки и единства научного образования.

Развитием системного анализа занимались ученые самых различных специальностей (физики, философы, геологи, медики, биологи). Это указывает на то, что положение ОТС находится в центре человеческих знаний.

По степени общности ОТС ставят на один уровень с математикой и философией (Дж.ван Гиг). Близко к ОТС расположены другие науки, занимающиеся изучением систем: кибернетика, теология, теория информации, инженерная теория связи, теория ЭВМ, системотехника, исследование операций и связанные с ними научные и инженерные направления.

ОТС как дисциплина, претендующая на роль «скелета науки» возложила на себя также функцию разработки системы основных понятий.

Определения понятий общей теории систем

Рассмотрим основные понятия ОТС, без оперирования которыми невозможно ни структурирование научного знания, ни анализ организаций.

Понятие **системы** является центральным в кибернетике и теории систем, многие авторы давали этому понятию различные определения. На основе анализа тридцати пяти различных определений понятия «система», были выбраны следующие (А.И.Уемов):

- система – множество объектов, на котором реализуется определенное отношение с фиксированными свойствами;
- система – множество объектов, которые обладают заранее определенными свойствами с фиксированными между ними отношениями.

Эти определения, несмотря на краткость, достаточно полны, но слишком тяжелы для восприятия. Более понятное, но в контексте кибернетики, определение Р.Эшби: «система – любая совокупность переменных, которую наблюда-

тель выбирает из числа переменных, свойственных реальной «машине»».

Определение Аккофа и Эмери (понятное и наиболее часто встречающееся в литературе): «Система – множество взаимосвязанных элементов, каждый из которых связан прямо или косвенно с каждым другим элементом, а два любые подмножества этого множества не могут быть независимыми».

Значения понятия «система» в греческом языке: сочетание, организм, устройство, организация, союз, строй, руководящий орган. Первенство в использовании этого понятия приписывается стоикам. Также это понятие прослеживается у Аристотеля.

Некоторые идеи, лежащие в основе общей теории систем (встречаются уже у Гегеля):

- целое есть нечто большее чем сумма частей;
- целое определяет природу частей;
- части не могут быть познаны при рассмотрении их вне целого;
- части находятся в постоянной взаимосвязи и взаимозависимости.

Система – это некоторая целостность, состоящая из взаимозависимых элементов и удовлетворяющая следующим двум требованиям:

1) поведение каждого элемента системы влияет на поведение системы в целом, и существенные свойства системы теряются, когда она расчленяется;
2) поведение элементов системы и их воздействие на целое взаимозависимы, и существенные свойства элементов системы при их отделении от системы также теряются (Гегель писал о том, что рука, отделенная от организма, перестает быть рукой, потому что она не живая).

Таким образом, свойства, поведение или состояние, которыми обладает система, отличаются от свойств, поведения или состояния образующих ее элементов (подсистем).

Кортежное определение системы: S : { {E}, {A}, F }, где S – система, {E} – совокупность элементов, {A} – совокупность связей, F – свойство (функция).

Основные характеристики системы:

- свойство (назначение, функция) системы отлично от суммы свойств ее элементов (эмерджентность); совокупность свойств элементов системы не представляет собой общего свойства системы, а дает некоторое новое свойство;
- для любой системы характерно наличие собственной, специфической закономерности действия, не выводимой непосредственно из одних лишь способов действия образующих ее элементов;
- всякая система является развивающейся, она имеет свое начало в прошлом и продолжение в будущем.

Элемент представляет собой далее не делимый компонент системы при данном способе расчленения, это компонент, обладающий рядом важных для целей рассмотрения свойств, при этом его внутреннее содержание (строение) безотносительно к цели рассмотрения.

При определении этого понятия нет такого большого количества мнений, как в случае с понятием «система». Все авторы дают схожие определения, но при этом часто упоминают, что элементы могут в свою очередь представлять собой системы, т.е. быть *подсистемами*. С этой точки зрения при анализе организации (составлении модели) основной сложностью является разбиение цельной системы на конечное число элементов, чтобы избежать излишней сложности и при этом не потерять в адекватности модели исследуемому объекту.

Элементы (Ван Гиг) делятся на *живые* и *неживые*, *входные* и *выходные*.

Различие между входными элементами и ресурсами очень незначительно и зависит от *точки зрения* и условий. В процессе преобразования входные элементы – это те элементы, которые потребляют ресурсы. Определяя входные элементы и ресурсы систем, важно указать, контролируются ли они проектировщиком системы, т.е. следует их рассматривать как часть системы или как часть окружающей среды.

При оценке эффективности системы *входные элементы* и *ресурсы* обычно относят к затратам.

Выходные элементы представляют собой результат процесса преобразо-

вания в системе и рассматриваются как выходы (например, прибыль).

Окружающая среда. *Окружающую среду* можно противопоставить *элементу*. Если элемент определяет (ограничивает) уровень детализации системы в рамках определенных границ, то окружающая среда эти границы устанавливает. Если система – это целенаправленное множество взаимосвязанных элементов любой природы, то внешняя (окружающая) среда – это множество существующих вне системы элементов любой природы, оказывающих влияние на систему или находящихся под ее воздействием.

Связь – отношения между элементами.

Структура. Понятие *структуры* связано с упорядоченностью отношений, которые связывают элементы системы. «Чтобы получить велосипед, недостаточно получить коробку со всеми его деталями. Необходимо еще правильно соединить детали между собой».

Структура – это совокупность необходимых и достаточных для достижения цели отношений между элементами (Перегудов и Тарасенко).

Другие определения структуры:

- совокупности связей между элементами системы, отражающих их взаимодействие;
- устойчивая упорядоченность ее элементов и связей;
- форма представления некоторого объекта в виде составных частей;
- множество всех возможных отношений между подсистемами и элементами внутри системы;
- совокупность элементов и связей между ними, которые определяются, исходя из распределения функций и целей, поставленных перед системой;
- то, что остается неизменным в системе при изменении ее состояния, при реализации различных форм поведения, при совершении системой операций и т.п.;
- совокупность элементов и связей между ними, определяющих целостность системы.

Структура определяет *целостность*, *внутреннее содержание* и основные *свойства системы*, изменение структуры ведет к изменению свойств системы.

Энтропия. *Энтропией* называется степень неупорядоченности.

В термодинамике, откуда заимствовано это понятие, энтропия связывается с вероятностью возникновения определенного расположения молекул.

В кибернетике и ОТС **энтропия** означает величину разнообразия системы, где под разнообразием понимается степень неопределенности, возникающей при выборе из большого числа возможных вариантов.

Для уменьшения энтропии необходимо уменьшить существующую неопределенность, что обеспечивается путем получения информации.

Понятия энтропии и количества информации можно использовать для того, чтобы дать характеристику живым и неживым системам.

Неживые системы (рассматриваются обычно как закрытые) имеют тенденцию развиваться по направлению к состоянию максимальной неупорядоченности и энтропии.

Отличительной чертой живых (открытых) систем является их сопротивляемость процессу разупорядочения и их развитие по направлению к состоянию более высокой организации.

ОТС объясняет эти тенденции, основываясь не следующих фактах:

- обработка информации приводит к соответствующему уменьшению положительной энтропии;
- получение энергии из внешней среды (увеличение отрицательной энтропии) противодействует ослабевающим тенденциям неотвратимого естественного процесса (увеличению положительной энтропии).

Кибернетика установила, что управление присуще только системным объектам, а общим в процессах (управления) является его антиэнтропийный характер, направленность на упорядочение системы. Конечной целью теории управления является универсализация, а значит согласованность, оптимизация и наибольшая эффективность функционирования систем.

Состав системы

Основными частями системы являются *вход, процесс (или операция) и выход*.

Первая часть системы – *вход*. У любой системы вход состоит из элементов, классифицируемых по их роли в процессах, протекающих в системе.

Первый элемент входа – тот, над которым осуществляется некоторый процесс, или операция. Этот вход есть или будет «нагрузкой» системы (сырье, материалы, энергия, информация и др.).

Вторым элементом входа системы является внешняя (окружающая) среда, под которой понимается совокупность факторов и явлений, воздействующих на процессы системы и не поддающихся прямому управлению со стороны ее руководителей.

Не контролируемые системами внешние факторы обычно можно разбить на ***две категории***:

- случайные, характеризуемые законами распределения, неизвестными законами или действующие без всяких законов (например, природные условия);
- факторы, находящиеся в распоряжении системы, являющейся внешней и активно, разумно действующей по отношению к рассматриваемой системе (например, нормативно-правовые документы, целевые установки).

Цели внешней системы могут быть известны, известны не точно, вовсе не известны.

Третий элемент входа обеспечивает размещение и перемещение компонентов системы, например различные инструкции, положения, приказы, то есть задает законы ее организации и функционирования, цели, ограничительные условия и др.

По содержанию *входы* классифицируются на *материальные, энергетические, информационные*, могут представлять собой любую комбинацию вышеперечисленных.

Вторая часть системы – это операции, процессы или каналы, через которые проходят элементы входа. Система должна быть устроена таким образом, чтобы необходимые процессы (производственные, подготовки кадров, материально-технического снабжения и др.) воздействовали по определенному закону на каждый вход, в соответствующее время для достижения желаемого выхода.

Третья часть системы – выход, являющийся продуктом или результатом ее деятельности. Система на выходе должна удовлетворять ряду критериев, важнейшие из которых – стабильность и надежность. По выходу судят о степени достижения целей, поставленных перед системой.

Обратная связь. Управляющий механизм любой системы основан на принципе подачи выходного сигнала обратно на вход. Поэтому *обратная связь* – понятие, лежащее в основе основных принципов *управления системами*. Именно оно способствовало установлению принципиальных аналогий между организацией управления в таких качественно различных системах, как машины, живые организмы и коллективы людей. *Обратная связь* означает соединение между выходом и входом системы, осуществляемое либо непосредственно, либо через другие элементы. Основными *функциями обратной связи* являются:

- противодействие тому, что делает сама система, когда она выходит за установленные пределы (например, реагирование на снижение качества);
- компенсация возмущений и поддержание состояния устойчивого равновесия системы (например, неполадки в работе оборудования);
- синтезирование внешних и внутренних возмущений, стремящихся вывести систему из состояния устойчивого равновесия, сведение этих возмущений к отклонениям одной или нескольких управляемых величин (например, выработка управляющих команд на одновременное появление нового конкурента и снижение качества выпускаемой продукции);
- выработка управляющих воздействий на объект по плохо формализуемому закону (например, установление более высокой цены на энергоносители вызывает в деятельности различных организаций сложные измене-

ния, которые меняют конечные результаты их функционирования, требуют внесения изменений в производственно-хозяйственный процесс путем воздействий, которые невозможно описать с помощью аналитических выражений).

Нарушение обратных связей в социально-экономических системах по различным причинам ведет к тяжелым последствиям. Отдельные локальные системы утрачивают способность к эволюции и тонкому восприятию намечающихся новых тенденций, перспективному развитию и научно обоснованному прогнозированию своей деятельности на длительный период времени, эффективному приспособлению к постоянно меняющимся условиям внешней среды.

Особенностью *социально-экономических систем* является трудность четкого определения обратных связей, которые в них, как правило, длинные и проходят через целый ряд промежуточных звеньев. Сами управляемые величины часто не поддаются ясному определению, и трудно установить множество ограничений, накладываемых на параметры управляемых величин. Не всегда известны также действительные причины выхода управляемых переменных за установленные пределы.

Типология систем

Системы бывают *живыми* (обладающими биологическими функциями) и *неживыми*. Различают *физические (конкретные)* и *абстрактные* системы. Систему относят к *конкретным (физическим),* если по крайней мере два ее элемента являются объектами, субъектами, либо теми и другими. Физические системы состоят из людей, изделий, оборудования, машин и прочих реальных или искусственных объектов. Система называется *абстрактной*, если ее элементы являются понятиями. В абстрактных системах свойства объектов, существование которых может быть неизвестным, за исключением их существования в уме исследователя, представляют символы. Идеи, планы, гипотезы и понятия, находящиеся в поле зрения исследователя, могут быть описаны как абстрактные си-

стемы. Все *абстрактные системы* являются *неживыми*, в то время как *конкретные системы* могут быть и *живыми*, и *неживыми*. *Виртуальные системы* – разновидность *конкретных*.

В зависимости от происхождения выделяют: *естественные* системы (например, климат, почва) и *сделанные человеком*.

По типу составных частей, входящих в систему, последние можно классифицировать на *машинные* (автомобиль, станок), *человеко-машинные* (самолет – пилот) и по типу «*социальные*» (коллектив организации).

По целевым признакам различают: *одноцелевые* системы, то есть предназначенные для решения одной единственной целевой задачи и *многоцелевые*.

Кроме того, можно выделить *функциональные* системы, обеспечивающие решение или рассмотрение отдельной стороны или аспекта задачи (планирование, снабжение и т. п.).

По степени связи с внешней средой системы классифицируют на *открытые* и *закрытые (замкнутые)*. *Открытые системы* – это системы, которые обмениваются материально-информационными ресурсами или энергией с окружающей средой регулярным и понятным образом. Противоположностью открытым системам являются *закрытые (замкнутые)*. *Закрытые системы* действуют с относительно небольшим обменом энергией или материалами с окружающей средой, например химическая реакция, протекающая в герметически закрытом сосуде. Все *живые системы – открытые*. *Неживые системы* являются относительно *замкнутыми*; наличие обратной связи наделяет их некоторыми неполными свойствами живых систем, связанными с состоянием *равновесия*. В деловом мире *закрытые системы* практически отсутствуют, и считается, что *окружающая среда* является главным фактором успехов и неудач деятельности различных организаций. Однако представителей различных школ управления первых 60 лет прошлого века, как правило, не волновали проблемы внешней среды, конкуренции и всего остального, что носит внешний для организации характер. Подход с точки зрения закрытой системы предполагал то, что следует делать, чтобы оптимизировать использование ресурсов, принимая во внимание

только происходящее внутри организации. Реалии окружающего мира заставили исследователей и практиков прийти к выводу, что любая попытка понять социально-экономическую систему, рассматривая ее закрытой, обречена на провал, поскольку главную роль в окружающем нас мире играет *неустойчивость* и *неравновесность*. С этой точки зрения системы можно классифицировать на: *равновесные, слабо равновесные,сильно неравновесные*.

Для социально-экономических систем состояние *равновесия* может наблюдаться на относительно коротком промежутке времени. Для *слабо равновесных* систем небольшие изменения внешней среды дают возможность системе в новых условиях достичь состояния нового равновесия. *Сильно неравновесные системы*, которые весьма чувствительны к внешним воздействиям, под влиянием внешних сигналов, даже небольших по величине, могут перестраиваться непредсказуемым образом.

Система может быть *устойчивой* и *неустойчивой*. *Устойчивость системы* – это состояние, означающее неизменность ее существенных переменных. *Неустойчивость* выражается в том, что система, организованная для выполнения определенных функций, перестает их выполнять под влиянием каких-либо причин.

В изменяющейся среде или под воздействием различных «возмущений», которые достигают порога *устойчивости*, система может прекратить существование, превращаться в другую систему или распадаться на составные элементы (пример – банкротство предприятия). Способность системы оставаться *устойчивой* через *изменения своей структуры* и поведения называется *ультрастабильностью*. Компании обеспечивают высокий уровень своей *стабильности* за счет высокой приспособляемости к внешним и внутренним условиям своего функционирования. Такие компании своевременно прекращают одни направления своей деятельности и начинают другие, вовремя выходят на новые рынки и покидают бесперспективные.

Адаптивные системы

Адаптивные системы – это системы, которые могут приспосабливаться к изменениям как внутренних, так и внешних условий, т.е. способные к адаптации.

Различают *пассивную* адаптацию (реагирование системы на изменение окружающей среды) и *активную* (воздействие системы на окружающую среду).

Адаптивные системы подразделяются на *самонастраивающиеся* и *самоорганизующиеся* системы. В первом случае в соответствии с изменениями внешней среды меняется способ функционирования системы, а во втором – меняется структура, организация системы.

Самоорганизующаяся система – кибернетическая (или динамическая) адаптивная система, в которой запоминание информации (накоплении опыта) выражается в изменении структуры системы.

Самоорганизующиеся и высокоорганизованные адаптивные системы обладают, кроме того, способностью так изменять внешнюю среду, чтобы изменение собственного поведения системы не являлось необходимым. Они в состоянии изменять (адаптировать) внешние условия для достижения собственных целей.

Если управляемая система и (окружающая) среда стационарны, то адаптивная управляющая система по истечении определенного периода времени накапливает необходимую информацию, устраняет неопределенность, и качество адаптивного управления приближается к качеству оптимального управления в условиях полной информации.

В самоорганизующихся системах характеристики объекта управления меняются во времени и устранить неопределенность полностью не удается. Однако в тех случаях, когда процесс адаптации быстро сходится к оптимальному процессу, качество адаптивного управления может мало отличаться от оптимального.

Поведение адаптивных систем является дуальным. С одной стороны, невозможно осуществить эффективное управление, не зная характеристик управляемой системы, с другой – можно изучать эти характеристики в процессе

управления и тем самым улучшать управление, стремясь к оптимальному. В этом случае управляющие воздействия носят двойственный характер: они служат средством как активного изучения, познания управляемой системы для будущего, так и непосредственного управления в текущий момент. В адаптивных системах (управления) всегда существует известное противоречие между познавательной и направляющей функциями управляющих воздействий.

Большие и сложные системы

Большая система – система, состоящая из значительного числа однотипных элементов и связей. Особенность *больших систем* – сложная иерархическая структура организации системы, предусматривающая сочетание централизованного управления с автономностью частей. Примеры больших систем: крупные производственно-экономические системы (например, холдинги), города, строительные и научно-исследовательские комплексы.

Системный анализ предусматривает специальные приемы, с помощью которых большую систему, трудную для рассмотрения исследователем, можно было бы разделить на ряд малых взаимодействующих систем или подсистем.

Таким образом, *большой системой* целесообразно назвать такую, которую невозможно исследовать иначе, как по *подсистемам*.

Помимо больших систем в задачах управления экономикой выделяют *сложные системы*.

Сложная система – система, состоящая из элементов разных типов и обладающая разнородными связями между ними. Особенности сложны систем:

1. наличие сложной, составной цели, параллельное существование разных целей или последовательная смена целей;
2. наличие одновременно многих структур у одной системы (например, технологической, административной, функциональной и т. д.);
3. невозможность описания системы на одном языке, т.е. необходимость использования разных языков для анализа и проектирования отдельных

ее подсистем (например, технологическая схема изготовления продукции; нормативно-юридические акты, устанавливающие распределение обязанностей и прав; схема документооборота и программа совещаний; порядок взаимодействия служб и отделов при разработке проекта плана).

Справиться с задачами анализа больших сложных систем можно лишь тогда, когда в нашем распоряжении будет надлежащим образом организованная система исследования, элементы которой подчинены общей цели.

Один и тот же объект может иметь множество разных типов систем. Если рассматривать производственное предприятие как совокупность машин, технологических процессов, материалов и изделий, которые обрабатываются на машинах, то предприятие представляется как *технологическая система*. Можно рассмотреть предприятие и с другой стороны: какие люди на нем работают, каково их отношение к производству, друг к другу и т. д. Тогда это же предприятие представляется в качестве *социальной системы*. При исследовании отношений руководителей и сотрудников предприятия к средствам производства, их участие в процессе труда и распределении его результатов и т. д. предприятие рассматривается как *экономическая система*.

Система управления – совокупность взаимосвязанных элементов, способ реализации технологии управления, предполагающий воздействие на объект с целью изменения его состояния и процессных характеристик. Система управления включает следующие основные элементы:

- датчики информации о состоянии объекта управления;
- подсистема сбора и передачи этой информации;
- подсистема обработки и отображения этой информации;
- подсистема выработки управляющих воздействий;
- подсистема передачи управляющих воздействий;
- исполнительные устройства.

Системный анализ (СА) и системный подход (СП)

Определение СА

Системный анализ как дисциплина сформировался в результате возникновения необходимости исследовать и проектировать сложные системы, управлять ими в условиях неполноты информации, ограниченности ресурсов и дефицита времени.

В системном анализе исследования строятся на использовании категории «система» и «системность рассмотрения». Под *системой* понимается единство взаимосвязанных и взаимовлияющих элементов, расположенных в определенной закономерности в пространстве и во времени, совместно действующих для достижения общей цели. Под *системностью рассмотрения* понимается учет всех важнейших факторов и взаимосвязей, влияющих на решение проблемы, использование определенной логики поиска лучшего решения и т.д.

Существуют различные точки зрения на содержание понятия «системный анализ» и область его применения. Согласно наиболее распространенной трактовке *системный анализ* – это конкретное теоретико-прикладное направление исследований, основанное на *системной методологии* и характеризующееся определенными принципами, методами и областью применения, включее в свой состав как методы *анализа*, так и методы *синтеза*.

Таким образом, **системный анализ** – это совокупность определенных научных методов и практических приемов решения проблем, возникающих во всех сферах целенаправленной деятельности общества, на основе *системного подхода* и представления *объекта исследования* в виде *системы*.

История СА

Системный анализ был разработан в США для военно-промышленного комплекса и считался наиболее ценным побочным достижением в области обороны и изучения космического пространства. В 1960-е годы в обеих палатах

конгресса США были внесены законопроекты «о мобилизации и использовании научно-технических сил страны для применения системного анализа и системотехники в целях наиболее полного использования людских ресурсов для решения национальных проблем». Позже системный анализ начал использоваться руководителями и инженерами в крупных предприятиях промышленности и в коммерческой области для поиска путей получения высокой прибыли и выявления возможных источников роста промышленной фирмы. В 1970-80-х годы на роль высших управляющих крупнейших американских корпораций начали приглашать людей, обладающих квалификацией в области системных дисциплин и системного анализа.

Характеристики СА

Системный анализ имеет следующие особенности:

- предназначен для решения слабо структурированных проблем, которые не могут быть поставлены и решены отдельными формальными методами математики;
- опирается на научное мировоззрение, в частности, на диалектическую логику;
- использует различные методы решения, которые могут быть разработаны как в рамках самого системного анализа, так и в рамках других наук;
- для получения квалифицированного суждения по проблемам использует не только формальные аналитические методы, но и методы качественного анализа;
- основное внимание уделяет целям и целеобразованию.

Системный подход

Системный подход – это набор правил применения системного анализа, подход к исследованию объекта (проблемы, явления, процесса) как к системе, в которой выделены элементы, внутренние и внешние связи, наиболее существенным образом влияющие на исследуемые результаты его функционирования, а цели каждого из элементов определены исходя из общего предназначения объекта. Можно также сказать, что системный подход – это такое направление методологии научного познания и практической деятельности, в основе которого лежит исследование любого объекта как сложной системы.

Системный подход подразумевает прохождение трех ступеней анализа:

1) определение целого (системы), частью которого является интересующий нас объект;
2) объяснение поведения или свойств этого целого (системы);
3) объяснение поведения или свойств интересующего нас предмета с точки зрения его функций в этом целом, частью которого он является.

При системном подходе *синтез* (объединение частей в целое) предшествует *анализу* (расчленение целого на части), а объясняемый предмет рассматривается как часть некоторого целого.

Ниже приведены некоторые *правила системного подхода*.

1. Не компоненты сами по себе составляют суть целого (системы), а наоборот, целое как первичное порождает при своем членении или формировании компоненты системы.
2. Сумма свойств (параметров) или отдельное свойство системы не равны сумме свойств ее компонентов, а из свойств системы нельзя вывести свойства ее компонентов.
3. Количество компонентов системы, определяющих ее размер, должно быть минимальным, но достаточным для реализации целей системы.
4. Для упрощения структуры системы следует сокращать количество уровней управления, число связей между компонентами системы и число па-

раметров модели управления, автоматизировать процессы.

5. Структура системы должна быть гибкой, с наименьшим числом жестких связей, способной быстро перстраиваться на выполнение новых задач, оказание новых услуг и т. п.
6. Структура системы должна быть такой, чтобы изменения в вертикальных связях компонентов системы оказывали минимальное влияние на функционирование системы. В социо-организационных структурах для этого следует обеспечивать: делегирование полномочий субъектам управления, оптимальную самостоятельность и независимость объектов управления.
7. Число горизонтальных связей между компонентами одного уровня системы должно быть минимальным, но достаточным для нормального функционирования системы. Установление горизонтальных связей способствует передаче знаний и навыков, обеспечивает координацию действий компонентов одного уровня по достижению целей системы. В то же время, уменьшение числа связей ведет к повышению устойчивости и оперативности функционирования системы.
8. При установлении взаимосвязей и взаимодействия системы с внешней средой следует строить «черный ящик» и формулировать сначала параметры «выхода», затем определять воздействие факторов макро- и микросреды, требования к «входу», каналы обратной связи и в последнюю очередь проектировать параметры процесса в системе.
9. В силу сложности, не следует пытаться познать все свойства и параметры системы.
10. Число связей системы с внешней средой должно быть минимальным, но достаточным для нормального функционирования системы. Чрезмерный рост числа связей усложняет управляемость системы, а их недостаточность снижает качество управления. При этом должна быть обеспечена необходимая самостоятельность компонентов системы, поскольку для обеспечения адаптивности система должна иметь возможность быстрого изменения своей структуры.

11. В условиях глобальной конкуренции и международной интеграции следует стремиться к росту степени открытости системы при обеспечении ее экономической, технической, информационной, правовой безопасности; обеспечивать совместимость с другими системами на основе стандартизации.
12. Для определения стратегии развития системы следует строить дерево целей. При этом следует помнить, что цели системы и цели ее компонентов, как правило, не совпадают.
13. При формировании миссии и целей системы следует отдавать приоритет интересам системы более высокого уровня как гарантии решения глобальных проблем.
14. При формулировании целей и стратегии развития системы следует учитывать неопределенность информации и вероятностный характер ситуаций.
15. Все компоненты должны выполнять конкретную задачу по достижению цели системы. Если без какого-либо компонента можно достичь цели системы, значит, этот компонент – лишний.
16. При построении дерева целей системы и оптимизации ее функционирования следует изучать проявление свойства ее мультипликативности. Например, безотказность системы определяется не сложением, а умножением коэффициентов безотказности ее компонентов.
17. При построении структуры системы и организации ее функционирования следует учитывать, что все процессы взаимообусловленны.
18. При формировании стратегии системы следует обеспечивать альтернативность путей ее функционирования и развития на основе прогнозирования различных ситуаций. Наиболее непредсказуемые фрагменты стратегии следует планировать по нескольким вариантам.
19. При организации функционирования системы следует учитывать, что ее эффективность не равна сумме эффективностей функционирования подсистем (компонентов). При взаимодействии компонентов возникает по-

ложительный (дополнительный) или отрицательный эффект синергии. Для получения положительного эффекта синергии необходимо иметь высокий уровень организованности системы.

20. Для снижения инерционности функционирования системы, т. е. увеличения скорости изменения выходных параметров при изменении входных параметров или параметров функционирования системы, следует ориентировать производство на интегрированные автоматизированные модули и системы, обеспечивающие мобильность производства и быстрое реагирование на изменения.
21. При обосновании инвестиций в проекты следует изучать доминантные (преобладающие, наиболее сильные) и рецессивные признаки системы и вкладывать средства в развитие первых.
22. Из всех показателей качества систем приоритет следует отдавать надежности как совокупности проявляющихся свойств безотказности, долговечности, ремонтопригодности и сохраняемости.
23. Структура и содержание системы формируются на идеях и принципах стандартизации, без соблюдения которых она не может функционировать. Глобальная конкуренция повышает удельный вес стандартизованных систем и их компонентов, особенно в международном масштабе.
24. Единственным путем развития организационно-экономических и производственных систем является инновационный.

Применение СА

Для исследования сложных систем применяются различные подходы к сочетанию процедур *анализа* и *синтеза*.

В общих чертах в рамках СА исследуются *факторы внутренней среды* организации, *факторы внешней среды организации*, а сама она рассматривается как открытая, динамично развивающаяся система.

Первая задача СА – *формулировка целей* системы и *критериев* их дости-

жения, которые должны быть выражены в виде конкретных показателей. Имея конкретные, четко сформулированные цели, можно выявить и проанализировать факторы, способствующие либо препятствующие достижению этих целей. Следующая задача системного анализа – *определение глобальной цели развития системы.*

Далее исследуются параметры «*выхода*» (товары или услуги). Затем определяют параметры «входа», т.е. исследуется *потребность в ресурсах, организационно-технический уровень системы, параметры внешней среды, параметры процессов*.

При этом нужно отметить, что не существует универсальной методики проведения системного анализа, и руководящие принципы также не являются универсальными. Каждое исследование имеет свои особенности и требует от исполнителей интуиции, инициативы и воображения, чтобы, в частности, определить цели проекта и добиться успеха в их достижении. При этом, выбрав конкретный алгоритм выполнения работ по СА, необходимо следовать предписаниям именно этого алгоритма.

В качестве примера ниже приведены примерные процедуры СА при *принятии решения* и *исследовании системы управления*.

СА при принятии решения. Основные процедуры системного анализа при принятии решениия:

1) изучение структуры системы, анализ ее компонентов, выявление взаимосвязей между отдельными элементами;
2) сбор данных о функционировании системы, исследование информационных потоков, наблюдения и эксперименты над анализируемой системой;
3) построение моделей;
4) проверка адекватности моделей, анализ неопределенности и чувствительности;
5) исследование ресурсных возможностей;
6) определение целей системного анализа;

7) формирование критериев;
8) генерирование альтернатив;
9) реализация выбора и принятие решений; внедрение результатов анализа.

СА при исследовании системы управления. Для исследования сложных систем необходимо особое единство процедур синтеза и анализа. Существуют разные подходы к их сочетанию. Ниже приведен один из перечней процедур СА, который может быть эффективно применен к исследованию таких систем, как предприятие:

1. Определить границы исследуемой системы. Эти границы условны и диктуются конкретной задачей исследования. Например, границы системы «корпорация» в одном случае могут быть определены списочным составом постоянного персонала, в другой задаче – постоянным персоналом плюс всеми акционерами компании, в третьем случае эти пределы расширяются за счет всех временно привлекаемых специалистов, экспертов, консультантов и т.д. Затем можно расширить эти границы за счет всех поставщиков компании, ее потребителей и любых иных субъектов каким-либо образом с ней связанных.
2. Определить все надсистемы, в которые входит исследуемая система в качестве части. При этом необходимо ограничиться лишь кругом наиболее значимых надсистем. Так, если мы выясняем воздействие на предприятие экономической среды, то именно она и будет той надсистемой, в которой следует рассматривать его функции. Предприятие следует также изучать в составе политических, социальных, экологических, государственных, международных и других надсистем. становится ясной необходимость сознательного изучения среды, окружающей предприятие.
3. Определить основные черты и направления развития всех надсистем, которым принадлежит данная система, в частности, сформулировать их цели и противоречия между ними;

4. Определить роль исследуемой системы в каждой надсистеме, рассматривая эту роль как средство достижения целей надсистемы. Следует рассматривать как ожидаемую, так и реальную роль системы с точки зрения надсистемы; пример: оценка потребностей покупателей в конкретном виде товаров, их количестве и качестве, и с другой стороны – оценка параметров реально выпускаемого товара.
5. Выявить состав системы, т.е. определить части, из которых она состоит. Нередко исследовательская задача требует не только расчленения системы на составные части, но и расчленения компонентов, из которых состоят сами части. Глубина декомпозиции определяется потребностями конкретной задачи.
6. Определить структуру системы, представляющую собой совокупность связей между компонентами. Здесь следует отметить многоструктурность любой системы.
7. Определить функции компонентов системы, т.е. целенаправленные действия компонентов, их вклад в реализацию роли системы в целом.
8. Выявить причины, объединяющие отдельные части в систему, в целостность (интегрирующие факторы). Основным интегрирующим фактором при создании социо-организационных систем является человеческая деятельность. Первичным интегрирующим фактором является цель. Определение реальной цели, послужившей причиной создания той или иной системы, является непростой задачей, поскольку цель в любой сфере деятельности представляет собой сложное сочетание различных противоречивых интересов.
9. Определить связи системы с внешней средой. Здаесь также необходимо также определить внешние системы, которым принадлежат компоненты исследуемой системы. Так, например, следует определить все системы, которым принадлежат работники предприятия – профсоюзы, политические партии, семьи, системы социокультурных ценностей и этических норм, этнические группы и т.д. Необходимо также хорошо знать связи

структурных подразделений и работников предприятия с системами интересов и целей потребителей, конкурентов, поставщиков, зарубежных партнеров и пр. Нужно также видеть связь между используемыми на предприятии технологиями и «пространством» научно-технического процесса и т.п. Осознание органического, хотя и противоречивого единства всех систем, окружающих предприятие позволяет понимать причины его целостности, предотвращать процессы, ведущие к дезинтеграции.

10. Рассмотреть исследуемую систему в динамике (развитии). Это означает сформулировать историю системы, источник ее возникновения, периоды становления, тенденции и перспективы развития, переходы к качественно новым состояниям. Необходимость динамического подхода к исследованию систем легко показать на сравнении двух предприятий, у которых в какой-то момент времени совпали значения одного из параметров, например, объема продаж. Из этого совпадения не вытекает, что предприятия занимают одинаковое положение на рынке: одно из них может набирать силу, двигаться к расцвету, а другое, наоборот, переживать спад.

При проектировании информационных систем СА включает следующие этапы предпоектного исследования системы управления:

1) сбор сведений об объекте автоматизации (организации или ее части);
2) анализ сведений, который включает следующие подэтапы:
   - описание предметной области (организации или ее части) (см. гл. 3.2);
   - моделирование предметной области (см. гл. 3.3).

Методы и средства выполенения процедур СА выбираются в процессе исследования / создания конкретной системы.

1.2. Организация как система управления

Объект управления

Рассмотрим определение понятия и общие характеристики организации как объекта управления.

Организация – это группа людей, деятельность которых сознательно координируется для достижения общей цели или целей. **Характеристики** любой организации с точки зрения системного подхода к управлению – это: *цели, ресурсы, зависимость от внешней среды, внутренние переменные, наличие подсистем и разделение труда*.

1. Цели – конкретные конечные состояния или желаемый результат, которого стремится добиться группа, работая вместе. Организация как правило – сложная система, и большинство организаций являются многоцелевыми. Цели организации взаимосвязаны и взаимозависимы.

2. Ресурсы. В общих чертах цели всякой организации включают преобразование ресурсов для достижения результатов. **Основные ресурсы,** используемые организацией – это: люди (человеческие ресурсы), капитал, материалы, технология и информация.

3. Зависимость от внешней среды. Организация, как **открытая система** (рис. 1.1) характеризуется взаимодействием с внешней средой. Внешняя среда включает:

- *среду косвенного воздействия:* экономические условия, потребители, профсоюзы, правительственные акты, законодательство, конкурирующие организации, социокультурные факторы (система ценностей в обществе, общественные ,взгляды и т.д.), политические факторы, отношения с местным населением, технику и технологии и другие составляющие;
- *международное окружение:* международный бизнес и *факторы международной среды* (экономика и госрегулирование, политическая обстановка).

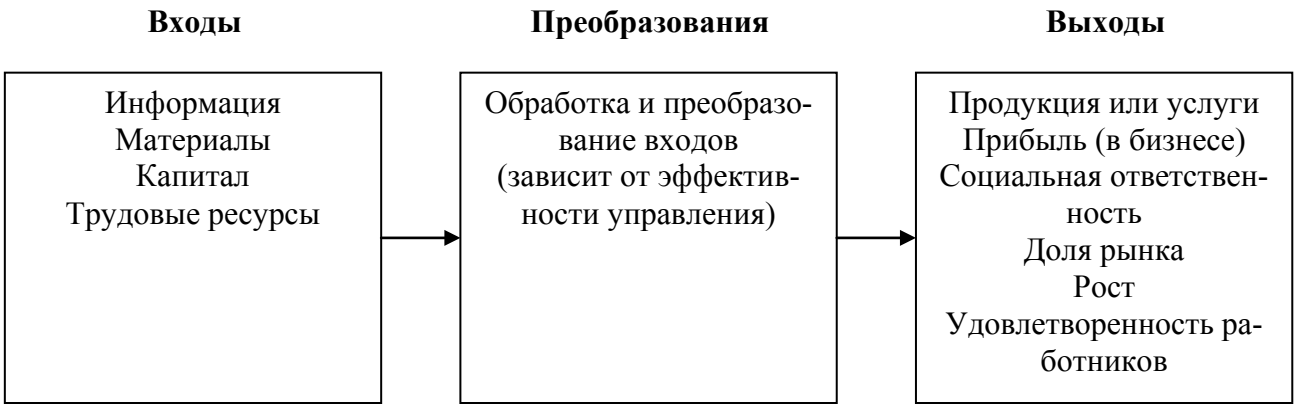

Рис. 1.1. Модель организации как открытой системы

На входе организация получает из окружающей среды *информацию*, *капитал*, *человеческие ресурсы* и *материалы*. Эти компоненты называют **входами**. В процессе преобразования организация обрабатывает эти входы, преобразуя их в *продукцию или услуги*. Эти продукция и услуги являются **выходами** организации, которые она вносит в *окружающую среду*. Если организация управления эффективна, то в ходе процесса преобразования образуется *добавочная стоимость входов*. В результате появляются многие *дополнительные выходы*, такие как *прибыль*, *увеличение доли рынка*, *увеличение объема продаж* (в бизнесе), *реализация социальной ответственности*, *удовлетворение работников*, *рост организации* и т.д.

4. Внутренние переменные – результаты управленческих решений. Основные переменные в организации, которые требуют внимания руководства, это: *цели, структура, задачи, технологии, люди*. Системный подход к управлению рассматривает организацию как совокупность взаимозависимых элементов, таких как *люди, структура, задачи и технологии*, которые ориентированы на достижение различных *целей* в условиях меняющейся внешней среды (рис. 1.2).

Цели вырабатываются руководством в ходе процесса планирования и сообщаются членам организации (процесс координирования).

Структура организации – это логические взаимоотношения уровней управления и функциональных областей, построенные в такой форме, которая позволяет наиболее эффективно достигать целей организации.

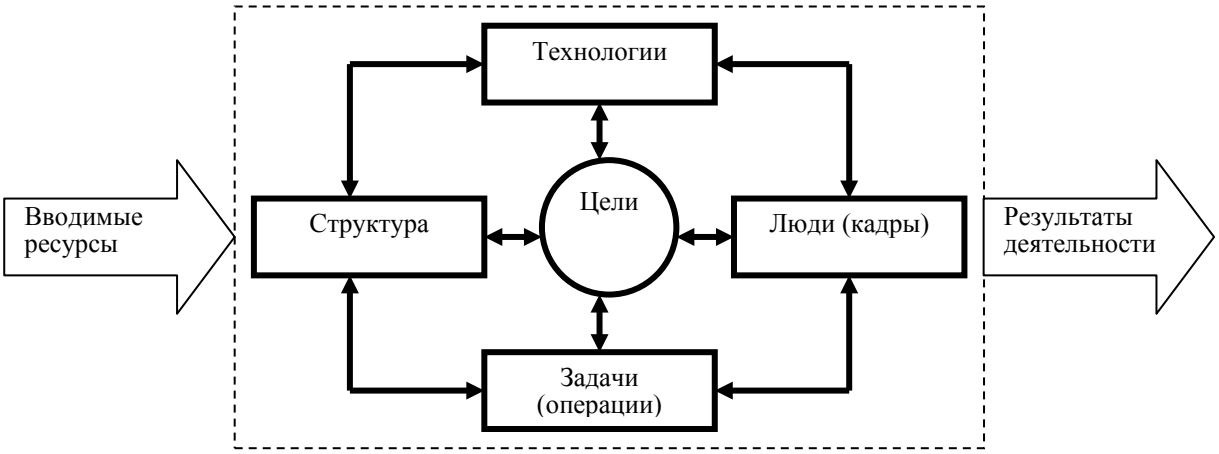

Рис. 1.2. Взаимосвязь внутренних переменных организации

Задача – это предписанная работа, серия работ или часть работы, которая должна быть выполнена заранее установленным способом в заранее оговоренные сроки. С технической точки зрения, задачи предписываются не работнику, а его должности. Задачи организации традиционно делятся на три категории: работа с людьми; работа с предметами (машинами, сырьем, инструментами); работа с информацией. Например, на обычном заводском конвейере работа людей состоит из работы с предметами. Задача мастера – это, в основном, работа с людьми. В тоже время, задачи казначея корпорации, в основном, связаны с информацией.

Технология – это средство преобразования сырья – будь то люди, информация или физические материалы – в искомые продукты и услуги. В более широком смысле *технология* – это сочетание *квалификационных навыков*, *оборудования*, *инфраструктуры*, *инструментов* и соответствующих *технических заданий*, необходимых для осуществления желаемых преобразований в *материалах, информации* или *людях*.

Задачи и технология тесно связаны между собой. Выполнение задачи включает использование конкретной технологии как средства преобразования материала, поступающего на входе, в форму, получаемую на выходе. То есть наиболее значимым компонентом технологии является *процесс*, с помощью которого исходные материалы (сырье) преобразуются в желаемый на выходе про-

дукт. По сути своей технология представляет способ, который позволяет осуществить такое преобразование.

5. Горизонтальное разделение труда – разделение большого объема работы на многочисленные небольшие специализированные задания. В очень маленьких организациях горизонтальное разделение труда может не прослеживаться достаточно четко, например, когда владелец фирмы является ее управляющим и выполняет сразу несколько функций – маркетолога, бухгалтера и т.д.

Сложные организации осуществляют горизонтальное разделение труда за счет образования подразделений, выполняющих специфические задания и добивающиеся специфических целей.

6. Организация как *сложная система* характеризуется наличием **подсистем**.

7. Вертикальное разделение труда – необходимо для *координации* работы групп, как следствие горизонтального разделения труда. Вертикальное развертывание разделения труда образует *уровни управления:* стратегический (институциональный), тактический (функциональный), технический (операционный) (рис. 1.3).

Рис. 1.3. Два способа представления уровней управления
Форма пирамиды используется для того, чтобы показать, что на каждом последующем уровне управления находится меньше людей, чем на предыдущем.

Для того чтобы организация могла добиться реализации своих целей, задачи должны быть скоординированы посредством вертикального разделения труда. Т.е. организацией необходимо управлять.

Управление организацией

Определение управления

Управление – это 1) процесс (деятельность), направленный на достижение цели; 2) перевод системы из одного состояния в заданное или удержание в заданном.

Управление социально-экономическими системами получило название менеджмент (от англ. management). В зависимости от управляемого элемента, можно выделить следующие составные области менеджмента: управление производством; управление персоналом; управление продажами; финансовый менеджмент; информационный менеджмент; управление проектами; управление качеством; маркетинг; стратегический менеджмент; инновационный менеджмент; управление инвестициями; экологический менеджмент; контроллинг и т.д.

Определение и функции менеджмента

В литературе встречаются следующие функции менеджмента: *планирование, организация, распорядительство (или командование), мотивация, руководство, контроль, коммуникация, исследование, оценка, принятие решений, подбор персонала, представительство и ведение переговоров или заключение сделок* и т.д.

Подход, основанный на объединении существенных видов управленческой деятельности в небольшое число категорий, которые можно считать применимыми ко всем организациям, выделяет **четыре первичные функции ме-**

неджмента: *планирование, организация, мотивация и контроль,* которые объединены **связующими процессами:** *коммуникации и принятия решения.*

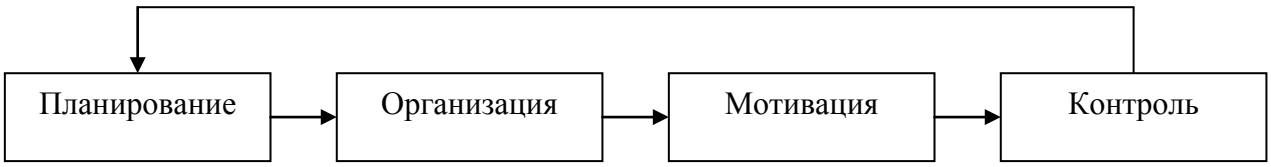

Рис. 1.4. Взаимосвязь функций управления

Большинство экспертов в области управления определяют **менеджмент** как процесс планирования, организации, мотивации и контроля, необходимый для того, чтобы сформулировать и достичь целей организации.

Планирование – определение целей организации и того, что должны делать члены организации, чтобы достичь этих целей.

Функция планирования отвечает на три следующих основных вопроса:

1. Где мы находимся в настоящее время? – оценка сильных и слабых сторон организации в различных *областях менеджмента*.
2. Куда мы хотим двигаться? – оценка возможностей и угрозы из *окружающей среды*.
3. Как мы собираемся сделать это? – решение в общих чертах и конкретно, что должны делать члены организации, чтобы достичь целей.

Стратегическое планирование, отвечаая на вопросы «что и для кого мы делаем», формирует миссию, т.е. роль, которую отводит себе организация в обществе. Миссия (по ISO-15704) – это деятельность, осуществляемая предприятием для того, чтобы выполнить функцию, для которой оно было учреждено, – предоставления заказчикам продукта или услуги. Миссия является средством позиционирования организации в рыночной среде. Правильно сформулированная миссия – это механизм, с помощью которого предприятие достигает своих целей.

Организация означает создание некой структуры. Процесс организации структурирует работу, конкретные задания и формирует подразделения, исходя из размера организации, ее целей, технологий и персонала. Известные типы организационных структур и принципы деления: функциональная – общие

навыки и производственные задачи, процессная, дивизионная – общий продукт, программа или географическое расположение, матричная – комбинация функциональной и дивизионной, командная – формирование групп для выполнения специфических задач, сетевая – подразделения независимы и обеспечивают функции для центральной компании и т.д.

Мотивация. Задача функции мотивации заключается в том, чтобы члены организации выполняли работу в соответствии с делегированными им обязанностями и сообразуясь с планом.

Контроль – это процесс обеспечения того, что организация действительно достигает своих целей. Вот почему стрелки на рисунке, исходящие от контроля, идут к планированию.

Связующие процессы управления. Четыре функции управления – планирование, организация, мотивация и контроль – имеют две общие характеристики: все они требуют принятия решений, и для всех необходима коммуникация. Из-за того, что эти две характеристики связывают все четыре управленческие функции, коммуникации и принятие решений часто называют связующими процессами.

Принятие решений – это выбор того, как и что планировать, организовывать, мотивировать и контролировать.

Основным требованием для принятия эффективного решения является наличие информации. Единственным способом получения информации является *коммуникация*. С одной стороны**, коммуникация** – это информация, передаваемая в процессе общения, с другой стороны – это процесс обмена информацией, ее смысловым значением между двумя или более людьми.

Коммуникация необходима для того чтобы: 1) договориться об общей цели, 2) принять правильные решения, 3) сделать решения понятными для исполнителей, 4) мотивировать исполнителей на выполнение решений и 4) правильно оценить, были ли достигнуты цели организации.

Доступ огромному объему информации затрудняет нахождение нужных и относящихся к делу сведений. Сокращение времени передачи информации

означает также, что у ЛПР остается все меньше времени на ее анализ. Поэтому производительность в области коммуникаций заключается не в повышении объема информации, а в улучшении ее качества. Таким образом, функционирование организации, как структурированного типа отношений между людьми, зависит от качества коммуникаций.

Два вида управления

Управление можно условно разделить на два вида:

1) *сознательное (иерархическое)* управление, означающее планомерное воздействие на объект; при иерархическом управлении цель функционирования системы задается её надсистемой;
2) *стихийное (синергетическое)* управление, когда воздействие на объект происходит в результате взаимодействия субъектов; одним из видов синергетического является адаптивное управление.

Адаптивное управление – это совокупность методов, позволяющих синтезировать системы, которые в процессе эволюции и функционирования демонстрируют способность к целенаправленному приспосабливающемуся поведению в сложных средах, то есть *адаптивные системы*.

Два типа организационных структур

В настоящее время наиболее распространены два основных подхода к описанию организаций, связанных с двумя различными методами управления.

1. **Функциональная структура управления** представляет организацию как совокупность функциональных отделов с жесткими вертикальными связями (централизованное принятие решений) и слабыми горизонтальными связями (жесткая специализация).

2. **Процессная структура управления** представляет организацию как совокупность процессов с сильными горизонтальными связями (централизованная координация) и самоорганизующимися вертикальными.

Функциональная (традиционная) структура связана с иерархическим управлением, процессная – с элементами адаптивного (синергетического) управления.

Функциональная структура управления

В конце XIX века произошла смена линейной структуры управления на функциональную. С ростом индустриализации и укрупнения предприятий стало нецелесообразно держать в штате универсальных работников. Работа усложнилась, и каждый работник, чтобы повысить производительность труда, был вынужден специализироваться на выполнении отдельных операций (горизонтальное разделение труда). Стало логично организовывать отделы, состоящие из работников родственных специальностей. Эти отделы назвали функциональными, а структуру управления – функциональной (или линейно-функциональной). Такая система давала определенные преимущества: возможность специализации способствовала повышению профессионального уровня работника, централизация функций привела к снижению затрат, стало легче формировать организационную структуру компании. Со временем рост специализации привел к обособлению функциональных подразделений и ослаблению межфункциональных связей и современная функциональная структура организации представляет собой функционально обособленные подсистемы и ряд устойчивых иерархических уровней управления для достижения целей.

Основные принципы иерархического управления: 1) централизованное принятие решений и 2) распределенная координация (разделение труда на основе жесткой специализации).

Недостатки функциональной структуры:

- разбиение технологий на отдельные, как правило, не связанные между собой фрагменты, реализуемые различными подразделениями предпри-

ятия, отсутствие цельного описания процессов и запаздывание с актуализацией соответствующей информации;

- отсутствие заинтересованности исполнителей в конечном результате. При функциональном подходе главным потребителем результатов труда работника является не клиент, а вышестоящий начальник, а видение происходящего чаще всего не выходит за рамки подразделения, в котором он работает. Работник не ориентирован на цели организации и тем более на клиента – он его просто не видит. Цель в иерархии: личная безопасность, самосохранение в системе;
- обмен информацией между различными подразделениями чрезмерно усложнен из-за ее вертикальной иерархичности, что приводит к большим накладным расходам, неоправданно длительным срокам выработки управленческих решений и, как следствие, потере клиентов;
- По подсчетам аналитиков время взаимодействия между подразделениями распределяется следующим образом: 20% – на выполнение работы и 80% – на передачу ее результатов следующему исполнителю.
- нарастание информационной энтропии. Управленческая информация передается с помощью естественного языка, обладающего информационной избыточностью, что служит причиной искажения сути сообщения.
- стандартные меры решения проблем в иерархической функциональной структуре: 1) манипуляции со структурой предприятия – «стремление на любую ситуацию отвечать реорганизацией – самым замечательным методом создания видимости прогресса, порождающим путаницу, неэффективность и деморализацию»; 2) манипуляции с численностью персонала; 3) попытки внедрения компьютерных ИС управления предприятием; 4) безуспешные попытки внедрения систем качества на базе ISO 9000.

Перечисленные недостатки устраняет процессная структура управления.

Процессная структура управления

История

Первое упоминание о процессном подходе как отдельной области исследования относится к 20 годам 19 века, когда в одной из компаний, где клерки работали с документами, был проведен анализ эффективности работы с использованием процессного подхода.

Руководитель решил проанализировать, как часто сидящие в одном большом помещении сотрудники передают друг другу документы. Была составлена схема, отражающая размещение сотрудников в помещении и все возможные взаимодействия между ними.

За небольшой промежуток времени была собрана статистика всех взаимодействий. По результатам анализа была проведена простая оптимизация: наиболее часто взаимодействующих между собой сотрудников посадили рядом друг с другом. В результате меньше времени тратилось на передачу документов. Это стало первым известным примером проведения описания и оптимизации процессов в бизнесе.

Определения понятий: процесс и процессный подход

Согласно стандарту «Основные Положения и Словарь — ISO/ОПМС 9000:2000», любая деятельность или комплекс деятельности, в которой используются ресурсы для преобразования входов в выходы, может рассматриваться как **процесс**. Чтобы эффективно функционировать, организации должны определять и управлять многочисленными взаимосвязанными и взаимодействующими процессами. Систематическая идентификация и управление применяемыми организацией процессами, и особенно взаимодействием таких процессов, могут считаться «**процессным подходом**».

В 2000 году Международная Организация по Стандартизации (ISO) при-

няла новую версию стандартов серии 9000, содержащих перечень требований к системе качества (СК) организации. Одно из принципиальных отличий новой версии стандартов – использование процессного подхода к менеджменту, а также к созданию и функционированию СК. Основную идею процессного подхода в новой версии стандартов можно свести к следующим положениям:

1) деятельность организации необходимо представить в виде сети взаимодействующих между собой процессов;
2) управление деятельностью организации должно основываться на управлении сетью процессов.

Процессный подход к описанию организации предполагает смещение акцентов от управления отдельными функциональными элементами на управление бизнес-процессами, связывающими деятельность этих функциональных элементов. Процессный подход позволяет:

- делегировать полномочия исполнителям бизнес-процессов;
- заинтересовать исполнителя в повышении качества конечного продукта;
- сосредоточить руководителей на стратегических и системных вопросах;
- рационально использовать материальные, финансовые и человеческие ресурсы;
- ускорить обмен информацией между функциональными подразделениями;
- быстро реагировать на внешние изменения.

Процессный подход к организации деятельности предприятия является предпосылкой для перехода от централизованного к **адаптивному управлению**, основными чертами которого являются: 1) распределенное принятие решений; 2) централизованная координация.

При этом чисто «процессная компания» является скорее иллюстрацией правильной организации работ. В действительности все бизнес-процессы компании протекают в рамках организационной структуры предприятия, описывающей функциональные компетентности и отношения.

Принципы процессного подхода

Одной из важных задач при проектировании ИС является получение формального описания предметной области. Для описания предметной области чаще всего используют процессный подход, и проектирование в этом случае ориентировано не на иерархическую организационную структуру, а на структуру бизнес-процессов.

Процессный подход к проектированию ИС базируется на следующих основных принципах:

1. **Восприятие бизнеса как системы** (системный подход):
    - любое предприятие следует рассматривать как систему, а её развитие – по законам сложных систем;
    - решение локальных проблем не меняет систему. Систему можно изменить только в целом;
    - система, находящаяся в устойчивом состоянии, не способна эволюционировать.
2. **Восприятие деятельности как процесса** (процессный подход):
    - любая деятельность может рассматриваться как процесс, следовательно, она может быть улучшена;
    - любая деятельность допускает разделение как по времени, так по материальным ресурсам и персоналу;
    - любая целенаправленная и спланированная деятельность, использующая ресурсы, преобразует вход в выход;
    - деятельность предприятия представляет собой сеть взаимосвязанных процессов, так как все виды деятельности и соответствующие им процессы взаимосвязаны;
    - каждый процесс имеет внешнего или внутреннего поставщика входных ресурсов и внешнего или внутреннего потребителя выходного продукта или услуги.

3. **Стандартизация и прозрачность ответственности**:
    - высшее руководство предприятия должно брать на себя полную ответственность за создание качества и управление им;
    - каждый процесс должен иметь владельца (персонификация и распределение ответственности за все виды деятельности);
    - все составляющие процессов должны быть максимально стандартизированы и понятны;
    - стандартизация должна осуществляться на базе взаимосвязанных и гармонизированных стандартов, реализующихся в виде нормативной документации и корпоративных стандартов, описывающих все виды деятельности предприятии.

Управление, направленное на успех

Организация существует для достижения определенных целей и считается добившейся успеха, если она их достигла. При этом размер предприятия и прибыльность не всегда могут считаться критериями успеха, если они не являются одними из целей. Составляющие успеха организации:

1) выживание;
2) эффективность. Эффективность определяется на основе денежной оценки ее входов;
3) производительность (результативность) – отношение количества единиц на выходе к количеству единиц на входе;
4) практическая реализация (implementation). Успешным считается такое решение, которое реализуется практически (т.е. превращается в действие) результативно и эффективно.

Повышение эффективности и производительности подразумевает: инновации в структуре организации, централизацию или интеграцию бизнес-процессов, качество и своевременность бизнес-информации.

1.3. Основные понятия информационных технологий и систем

Определения ИТ

В широком смысле, **Информационные технологии, ИТ** (Information technology, **IT**) **–** общий термин для обозначения различных технологий обработки и передачи информации. К ИТ относятся компьютерные технологии, телекоммуникация и микроэлектроника.

Обычно под ИТ понимают *компьютерные технологии*, а именно – методы применения вычислительной техники (ВТ) и компьютерного ПО при работе с информацией (*выполнении функций: сбора, хранения, защиты, обработки, передачи и использования данных).*

ИТ (по UNESCO) – это комплекс взаимосвязанных научных и инженерных дисциплин, изучающих ВТ, способы взаимодействия ее с людьми и производственным оборудованием, способы эффективной организации труда людей, а также связанные со всем этим социальные, экономические и культурные проблемы.

По определению ITAA, Американской ассоциации по ИТ (Information Technology Association of America), **ИТ** – это изучение, проектирование, разработка, внедрение, поддержка или управление **компьютерными информационными системами**, включая программные приложения и компьютерное аппаратное обеспечение.

Бурный рост отрасли ИТ приходится конец 1990-х годов, связанный с появлением и развитием **информационно-коммуникационных технологий.**

Информационно-коммуникационные технологии (Information and communication technologies, **ICT**) – термин, покрывающий все технические средства обработки и обмена информацией. Наряду с тем, что он формально включает «доцифровые» технологии, включая бумажные, часто используется для описания цифровых технологий, включая как способы коммуникации

(коммуникационные протоколы, способы обмена, коммуникационное оборудование, медиа), так и способы хранения и обработки информации (вычисления, хранение данных и т.д.).

Термин стал популярным (за рубежом) частично благодаря объединению ИТ и телекоммуникационной технологии.

Определения и типы информации

По определению БСЭ, *информация* (от лат. informatio − «научение», «сведение», «оповещение») − это сведения, передаваемые системой знаков любого рода.

Федеральный закон РФ от 27.07.06 N149-ФЗ «Об информации, информационных технологиях и о защите информации» определяет информацию, как сведения (сообщения, данные), независимые от формы их представления.

Точное общепринятое определение отсутствует ввиду отсутствия определения того, что такое «*сведения*». Поэтому понятие «информация» принято определять через ее *свойства* (подобно понятию «материя»), т. е. *информация* − это явление, которое характеризуется наличием *источника, приемника, канала связи* и т. д. При этом многие из свойств информации (например, «*смысл*») еще не вполне познаны наукой, что еще более затрудняет формулировку точного определения.

В XX веке термин «информация» был введен в ряде научных областей, получив особые для них толкования и определения.

Информация, зафиксированная на материальных носителях и хранимая в ИС (библиотеках, архивах, хранилищах, фондах, банках данных, системах знаний и т. п.), носит название *информационные ресурсы*.

Информацию можно условно разделить на группы **по сфере использования.**

Например, *экономическая информация* − это совокупность сведений о социально-экономических процессах, служащих для управления ими и коллек-

тивами людей. Характеристики экономической информации: большие объемы, многократное повторение циклов ее получения и преобразования в установленные временные периоды (месяц, квартал, год и т. д.), многообразие источников и потребителей, значительный удельный вес рутинных процедур при ее обработке.

Информация в ИС может быть *структурированной* и *неструктурированной* и условно делится на *данные* и *знания*.

Структурированные данные – это информация в виде чисел и текста, хранимая в нормализованных БД. Над такими данными можно выполнять различные операции.

К **неструктурированной информации** можно отнести обычные офисные электронные документы в формате Word или Excel, PDF, а также рисунки, чертежи, графики, сканированные изображения и вообще файлы любых форматов, сообщения электронной почты, Web-страницы, видео и другую информацию в электронном виде − в противоположность структурированным данным, обычно находящимся под управлением реляционной или многомерной СУБД. Для хранения неструктурированной информации используется файловая система ОС или объектно-ориентированные СУБД.

Данные – это отдельные факты, характеризующие объекты, процессы и явления предметной области, а также их свойства. В теории ИТС **данные** − это информация, фиксированная в определенной форме, пригодной для последующей обработки, хранения и передачи, а именно хранимая в базах данных и обрабатываемая прикладными программами.

Знание (обще определение) – это совокупность понятий, представлений о чем-либо, унаследованных, полученных, приобретенных, накопленных в результате учения, опыта, в процессе жизни и т. д. и обычно реализуемых в деятельности.

Оксфордский словарь английского языка (Oxford English Dictionary) определяет *знание* как «навыки, приобретенные человеком через опыт или об-

разование»; «теоретическое или практическое понимание предмета в специфической области или в целом»; «факты или информация, понимание или осведомленность о факте или ситуации, приобретенные опытным путем».

Философские дебаты начались с формулировки знания как justified true belief. Однако до сих пор не существует ни общепринятой формулировки понятия *знание*, ни полного обзора по понятию, несмотря на огромное количество различных теорий.

Приобретение знаний включает комплекс когнитивных процессов: восприятие, изучение, коммуникацию, ассоциацию и рассуждение. Термин *знание* также используется для обозначения уверенного понимания предмета со способностью применения его для разрешения конкретных ситуаций.

В теории *искусственного интеллекта* (ИИ) *знание* − это совокупность сведений о мире (у индивидуума, общества или у системы ИИ), включающих в себя информацию о свойствах объектов, закономерностях процессов и явлений, а также правилах использования этой информации для принятия решений.

В системах управления знаниями (СУЗ) используют понятие *корпоративных знаний*, означающее коллективный опыт и понимание процессов управления запланированными и незапланированными ситуациями в организации. Знания могут поступать из баз данных, от людей, с веб-страниц и рабочих процессов, в которых участвуют люди, и делятся на *неявные* (те, что в головах людей) и *явные* (задокументированные).

Определения информационных систем (ИС)

Термин **Информационная система** в широком понимании относится к взаимодействию между процессами и технологией, в узком – к взаимодействию между людьми, процессами, информацией и технологией (рис. 1.5).

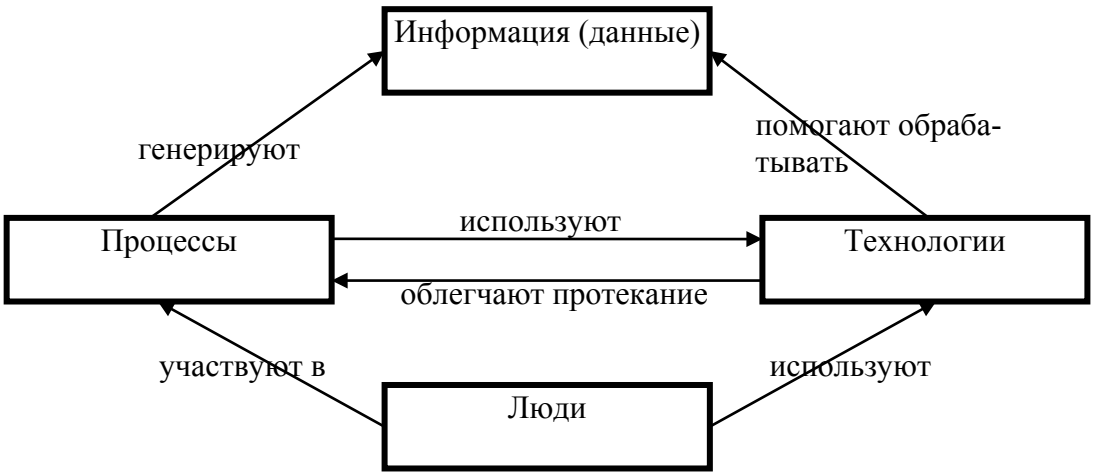

Рис. 1.5. Концепция ИС

По определению стандарта **ISO** 12207, (**информационная**) **система** – это объединение одного или нескольких процессов, аппаратных средств, программного обеспечения, оборудования и людей для обеспечения возможности удовлетворения определенных потребностей или целей.

Существует множество других определений ИС, имеющих смысл в соответствующих контекстах. Например, по федеральному закону РФ от 27.07.06 N149-ФЗ «Об информации, информационных технологиях и о защите информации», информационная система – это совокупность содержащейся в базах данных информации и обеспечивающих ее обработку ИТ и технических средств.

Функции, состав и структуры ИС
(ГОСТ 24.103-84 – Функции, состав и структура АСУ)

ИС состоит из **функциональной** и **обеспечивающей** частей. Подсистемы, входящие в функциональную часть, называются *функциональными подсистемами* ИС, а подсистемы, входящие в обеспечивающую часть – *обеспечивающими подсистемами* ИС.

Задачи функциональных подсистем – это те задачи, ради решения которых и создается ИС. Они различны для различных видов ИС. Например:

- подсистема планирования;
- подсистема управления качеством;
- подсистема управления персоналом
- и т.д.

Целью обеспечивающих подсистем является обеспечение решения задач функциональных подсистем ИС. *Обеспечивающая часть* не зависит от вида ИС и включает следующие подсистемы:

1. Информационное обеспечение (ИО). В состав ИО входят классификаторы технико-экономической информации, нормативно-справочная информация, форма представления и организация данных в системе, в том числе: формы документов, видеограмм, массивов и логические интерфейсы (протоколы обмена данными).

Информационное обеспечение ИС предоставляет:

1) за счет кодирования объектов – однозначное и экономичное представление информации в системе;
2) на основе классификации объектов – организацию процедур анализа и обработки информации с учетом характера связей между объектами;
3) на основе экранных форм ввода-вывода данных – организацию взаимодействия пользователей с системой;
4) на основе унифицированной системы документации – обеспечение эффективного использования информации в контуре управления деятельностью объекта автоматизации.

Информационное обеспечение включает в себя внемашинную и внутримашинную информационные базы (ИБ).

Внемашинная ИБ – совокупность всех документированных сведений (данных) и сообщений, используемых в ИС, включая: *классификаторы технико-экономической* информации и *нормативно-справочную информацию*.

Внутримашинная ИБ – совокупность всех данных на машинных носителях, сгруппированных по определенному признаку.

База данных (БД) – часть внутримашинной ИБ, представляющая сово-

купность массивов (файлов, сегментов и т.д.) и выделенная для реализации определенных функций ИС; это структурированный организованный набор данных, описывающих характеристики какой-либо физической или виртуальной системы.

К информационному обеспечению предъявляются следующие *общие требования* (по ГОСТ 24.205-80 «Требования к содержанию документов по информационному обеспечению»):

- информационное обеспечение должно быть достаточным для поддержания всех автоматизируемых функций объекта;
- для кодирования информации должны использоваться принятые у заказчика классификаторы;
- для кодирования входной и выходной информации, которая используется на высшем уровне управления, должны быть использованы классификаторы этого уровня;
- должна быть обеспечена совместимость с информационным обеспечением систем, взаимодействующих с разрабатываемой системой;
- формы документов должны отвечать требованиям корпоративных стандартов заказчика (или унифицированной системы документации);
- структура документов и экранных форм должна соответствовать характеристикам терминалов на рабочих местах конечных пользователей;
- графики формирования и содержание информационных сообщений, а также используемые аббревиатуры должны быть общеприняты в этой предметной области и согласованы с заказчиком;
- в ИС должны быть предусмотрены средства контроля входной и выходной информации, обновления данных в информационных массивах, контроля целостности информационной базы, защиты от несанкционированного доступа.

2. Программное обеспечение, ПО (software) – это комплекс программ, применяющихся в ИС. В состав программного обеспечения входят программы (в том числе программные средства) с программной документацией на них, не-

обходимые для реализации всех функций системы в объеме, предусмотренном в техническом задании на создание ИС. Различают *общее* и *специальное* ПО.

Общее (базовое) осуществляет управление работой технических средств и ИБ. К общему ПО относят:

- СУБД;
- операционные системы (ОС);
- сервисные средства и утилиты;
- инструментальные средства разработки ПО.

Специальное (прикладное, application software) предназначено для решения функциональных задач пользователей.

3. Техническое/аппаратное обеспечение (hardware) – это комплекс технических средств для сбора, передачи, хранения и обработки информации. В **состав** технического обеспечения входят технические средства, необходимые для реализаций функций системы. Аппаратное обеспечение включает в себя все *физические части компьютера (ЭВМ)*, но не включает информацию (данные), которые он хранит и обрабатывает, и программное обеспечение, которое им управляет. Кроме того, в аппаратное обеспечение также входят внешние компоненты – *сетевое оборудование* и *периферийные устройства ввода* (клавиатуры, манипуляторы типа «мышь», сенсорные экраны, сканеры различного назначения, средства распознавания речи, сенсорные датчики, устройства видеозахвата и др.) и *вывода* (мониторы, печатающие устройства, устройства вывода звука, модемы и др.). В общем случае аппаратное обеспечение включает средства получения, ввода, подготовки, обработки, хранения (накопления), регистрации, вывода, отображения, использования, передачи информации и средства реализации управляющих воздействий.

4. Организационное обеспечение – это документы, определяющие функции подразделений управления, действия и взаимодействие персонала ИС.

5. Метрологическое обеспечение – это метрологические средства и инструкции по их применению.

6. Правовое обеспечение – это совокупность документов, определяющих

юридические аспекты функционирования системы (нормативные документы, определяющие правовой статус системы, персонала, правила функционирования и нормативы на автоматически формируемые документы, в том числе на машинных носителях информации). Правовое обеспечение в составе функционирующей системы реализуется в виде документов организационного обеспечения.

7. Лингвистическое обеспечение – это совокупность языковых средств, используемых для машинной обработки информации и облегчающих общение человека с техническими средствами ИС. В состав лингвистического обеспечения входят *тезаурусы* и *языки описания и манипулирования данными*. Лингвистическое обеспечение функционирующей ИС может присутствовать в ней самостоятельно или в виде решений по информационному обеспечению и в документах организационного обеспечения.

8. Математическое обеспечение – это методы решения задач управления, модели и алгоритмы. В функционирующей системе математическое обеспечение реализовано в составе программного обеспечения.

При описании ИС пользуются следующими видами структур, отличающимися типами элементов и связей между ними (табл. 1.1.):

Таблица 1.1.

Структуры ИС (ГОСТ 34)

Вид структуры	**Элементы**	**Связи**
функциональная	функции, задачи, операции	информационные
техническая	устройства	линии связи
организационная	коллективы людей и отдельные исполнители	информационные, соподчинения и взаимодействия
алгоритмическая	алгоритмы	информационные
программная	программные модули	информационные и управляющие
информационная	формы существования и представления информации в системе	операции преобразования информации в системе

Необходимый состав элементов выбирают в зависимости от вида конкретной АСУ.

1.4. Основные понятия проектирования

Понятия проектирования

В общем смысле, **проектирование** – это процесс создания проекта, прототипа, прообраза предполагаемого или возможного объекта, состояния.

Проектирование (в технике) – это разработка проектной, конструкторской и другой технической документации, предназначенной для осуществления строительства, создания новых видов и образцов. В процессе проектирования выполняются технические и экономические расчёты, схемы, графики, пояснительные записки, сметы, калькуляции и описания.

Проект (в технике) – это комплект указанной документации и материалов (определённого свойства). Завершенная разработка по проектированию некоторой новой системы представляет собою носитель (например, бумажный или электронный), на котором изображены текстовые описания, чертежи, формулы, модели, алгоритмы, на основе которых создается работающая система.

Проект (в теории Управления проектами) – это уникальная (в отличие от процесса) деятельность, имеющая начало и конец во времени, направленная на достижение определённого результата (цели), создание определённого уникального продукта или услуги при заданных ограничениях по ресурсам и срокам, а также требованиям к качеству и допустимому уровню риска.

Таким образом, проект – это временное предприятие, предназначенное для создания уникальных продуктов, услуг или результатов.

Создание ИС представляет собой программный проект.

Типология проектов по созданию ИС

Проект создания ИС может быть *индивидуальным* или *типовым*.

Индивидуальный проект – подразумевает разработку ИС, как правило с помощью специалистов самой организации.

Типовое проект ИС предполагает создание системы из готовых типовых проектных решений.

Типовое проектное решение (ТПР) – это тиражируемое (пригодное к многократному использованию) проектное решение. Выделяют следующие классы ТПР:

- *элементные ТПР* – типовые решения по задаче или по отдельному виду обеспечения задачи (информационному, программному, техническому, математическому, организационному);
- *подсистемные ТПР* – в качестве элементов типизации выступают отдельные подсистемы, разработанные с учетом функциональной полноты и минимизации внешних информационных связей;
- *объектные ТПР* – типовые отраслевые решения, которые включают полный набор функциональных и обеспечивающих подсистем ИС.

Каждое типовое решение предполагает наличие, кроме собственно функциональных элементов (программных или аппаратных), документации с детальным описанием ТПР и процедур настройки в соответствии с требованиями разрабатываемой системы.

Для реализации типового проектирования используются два подхода: *параметрически-ориентированный* и *модельно-ориентированный.*

Параметрически-ориентированное проектирование включает следующие основные этапы:

1. декомпозиция проектируемой ИС на множество составляющих компонентов (подсистем, программных модулей и т.д.);
2. выбор и приобретение из имеющихся на рынке ТПР, необходимых для реализации выделенных компонентов;
3. настройка (доработка) приобретенного ТПР на особенности конкретного предприятия с помощью обслуживающей организации (либо самостоятельно с помощью штатных IT-специалистов).

Выбор и приобретение ТПР подразумевает выполнение следующих шагов:

1) определение критериев оценки компонентов ИС с точки зрения решения поставленных задач;
2) анализ и оценка доступных ТПР по сформулированным критериям;
3) выбор и закупка наиболее подходящего пакета.

Критерии оценки ТПР делятся на следующие группы:

- назначение и возможности;
- отличительные признаки и свойства;
- требования к техническим и программным средствам;
- документация;
- факторы финансового порядка;
- особенности установки;
- особенности эксплуатации;
- помощь поставщика по внедрению и поддержке;
- оценка качества решения и опыт его использования;
- перспективы развития.

Внутри каждой группы критериев выделяется некоторое подмножество частных показателей, детализирующих каждый из десяти выделенных аспектов анализа выбираемых ППП. Достаточно полный перечень показателей можно найти в литературе. Числовые значения показателей для конкретных ППП устанавливаются экспертами по выбранной шкале оценок (например, 10-балльной). На их основе формируются групповые оценки и комплексная оценка пакета (путем вычисления средневзвешенных значений). Нормированные взвешивающие коэффициенты также получаются экспертным путем.

Модельно-ориентированное проектирование заключается в адаптации состава и характеристик типовой ИС к модели объекта автоматизации. Технология проектирования в этом случае должна обеспечивать единые средства для работы как с моделью типовой ИС, так и с моделью конкретного предприятия.

Модельно-ориентированное проектирование ИС предполагает, прежде всего, построение модели объекта автоматизации с использованием специаль-

ного программного инструментария (например, SAP Business Engineering Workbench (BEW), BAAN Enterprise Modeler).

Возможно также создание ИС на базе *типовой модели* ИС из *репозитория*, который поставляется вместе с программным продуктом и содержит как базовую (эталонную) модель ИС, так и конфигурации для определенных отраслей или типов производства.

Подходы к проектированию систем

Стихийная («лоскутная») автоматизация (подход «снизу-вверх»)

Индустрия разработки ПО начала зарождаться в середине 50-х годов XIX в., однако почти до конца 60-х ей не уделялось серьезного внимания, поскольку ее доля в компьютерном бизнесе была слишком мала. Серьезный рост начался в 70-х годах XX в., начиная с принятого фирмой IBM в 1969 г. решения о развязывании цен (раздельном назначении цен на аппаратуру, ПО и услуги), и продолжился до появления персонального компьютера.

На первом этапе основным подходом к проектированию ИС был *метод «снизу-вверх»*. ИС создавалась в виде набора приложений, наиболее важных в данный момент для поддержки деятельности организации. Основной целью этих проектов было обслуживание текущих потребностей конкретного предприятия, а не создание тиражируемых продуктов. Такой подход отчасти сохраняется и сегодня.

Основной недостаток метода: возникновение проблем при объединении существующих систем.

Системное проектирование (подход «сверху-вниз»)

Противоположностью стихийной автоматизации является системное проектирование, или *автоматизация «сверху-вниз»*. Смысл системного проектиро-

вания – реорганизация управления и перепроектирование всей корпоративной информационной системы, которые наилучшим образом достигают целей управления. Этапы системного проектирования:

1) определение целей и задач управления организацией;
2) создание модели организации, главное требование к которой – системная целостность; каждое изменение элемента модели требует перепроверки и согласования как «сверху-вниз», так и «снизу-вверх»;
3) создание корпоративной ИС на основе этой модели.

Основной недостаток системного подхода к проектированию – трудоемкость поддержания целостности модели.

В настоящий момент большинство организаций уже имеет ИС, в различной степени автоматизирующие процессы в них протекающие. Поэтому типичными в настоящее время являются следующие проекты:

- по разработке новых ИС и их интеграции с существующими ИС;
- по разработке новых ИС с целью замены существующих ИС;
- по модернизации (наращиванию функциональности, развитию) существующих ИС.

1.5. Жизненный цикл проекта по созданию ИС

Понятие жизненного цикла информационной системы (ЖЦ ИС)

Каждый проект, независимо от сложности и объема работ проходит в своем развитии путь от состояния когда «проекта еще нет» до состояния когда «проекта уже нет». Совокупность ступеней развития от возникновения идеи до полного завершения проекта принято разделять на фазы, стадии и этапы.

ЖЦ ИС – это непрерывный процесс, который начинается с момента принятия решения о необходимости создания ИС и заканчивается в момент ее полного изъятия из эксплуатации.

В определении количества фаз и их содержания имеются некоторые отличия, поскольку эти характеристики во многом зависят от условий осуществления конкретного проекта и опыта основных участников. Тем не менее логика и основное содержание процесса разработки ИС почти во всех случаях являются общими. Здесь можно выделить следующие фазы развития ИС:

1) **концептуальная фаза.** Главным содержанием работ на этой фазе является определение проекта и разработка его концепции, включающая:
   - формирование идеи, постановку целей;
   - формирование ключевой команды проекта;
   - изучение мотивации и требований заказчика и других участников;
   - сбор исходных данных и анализ существующего состояния (*предпроектное обследование объекта автоматизации*);
   - определение основных требований и ограничений, требуемых материальных, финансовых и трудовых ресурсов;
   - сравнительную оценку альтернатив;
   - представление предложений, их экспертизу и утверждение;
2) **разработка технического предложения.** Главным содержанием этой фазы является разработка технического предложения и переговоры с заказчиком о заключении контракта. Общее содержание работ этой фазы:

- разработка основного содержания и *базовой структуры проекта*;
- разработка и утверждение *технического задания*;
- планирование и декомпозиция базовой структурной модели проекта;
- составление сметы и бюджета проекта, определение потребности в ресурсах;
- разработка календарных планов и укрупненных графиков работ;
- подписание контракта с заказчиком;
- ввод в действие средств коммуникации участников проекта и средств контроля за ходом работ;

3) **проектирование.** На этой фазе определяются подсистемы, их взаимосвя-

зи, выбираются наиболее эффективные способы выплнения проекта и использования ресурсов. В основе проектирования ИС лежит *моделирование предметной области*. Цель моделирования – избежать ошибок, приводящих к экономическим потерям и затратам на последующее перепроектирование системы. Другие работы, характерные для фазы проектирования:

- выполнение базовых проектных работ;
- разработка частных технических заданий;
- выполнение концептуального проектирования;
- составление технических спецификаций и инструкций;
- представление проектной разработки, экспертиза и утверждение.

4) **разработка.** На этой фазе производятся координация и оперативный контроль работ по проекту, осуществляется изготовление подсистем, их объединение и тестирование. Основное содержание фазы:

- выполнение работ по разработке программного обеспечения;
- выполнение подготовки к внедрению системы;
- контроль и регулирование основных показателей проекта;

5) **ввод системы в эксплуатацию.** На этой фазе проводятся испытания, опытная эксплуатация системы в реальных условиях, ведутся переговоры о результатах выполнения проекта и о возможных новых контрактах. Основные виды работ:

- комплексные испытания;
- подготовка кадров для эксплуатации создаваемой системы;
- подготовка рабочей документации, сдача системы заказчику и ввод ее в эксплуатацию;
- сопровождение, поддержка, сервисное обслуживание;

6) **изъятие из эксплуатации или замена:**

- оценка результатов проекта и подготовка итоговых документов;
- разрешение конфликтных ситуаций и закрытие работ по проекту;

- накопление опытных данных для последующих проектов, анализ опыта, состояния, определение направлений развития.

На обнаружение ошибок, допущенных на стадии проектирования, расходуется примерно в два раза больше времени, чем на последующих фазах, а их исправление обходится в пять раз дороже. Поэтому на начальных стадиях проекта разработку следует выполнять особенно тщательно. Наиболее частые ошибки, допускаемые на начальных этапах:

1) ошибки в определении интересов заказчика;
2) концентрация на маловажных, сторонних интересах;
3) неправильная интерпретация исходной постановки задачи;
4) неправильное или недостаточное понимание деталей;
5) неполнота функциональных спецификаций (системных требований);
6) ошибки в определении требуемых ресурсов и сроков;
7) редкая проверка на согласованность этапов и отсутствие контроля со стороны заказчика (нет привлечения заказчика).

Модели жизненного цикла ИС

Моделью жизненного цикла ИС будем называть некторую структуру, определяющую последовательность выполнения процессов, действий и задач, выполняемых на протяжении ЖЦ ИС, а также взаимосвязи между этими процессами, действиями и задачами.

К настоящему времени наибольшее распространение получили следующие две модели ЖЦ *каскадная* (70-85 гг.) и *спиральная* (86-90 гг.).

Каскадная модель ЖЦ

Основной характеристикой какскадного способа является разбиение всей разработки на этапы, причем переход с одного этапа на следующий происходит

только после того, как будет полностью завершена работа на текущем (рис. 1.6.).

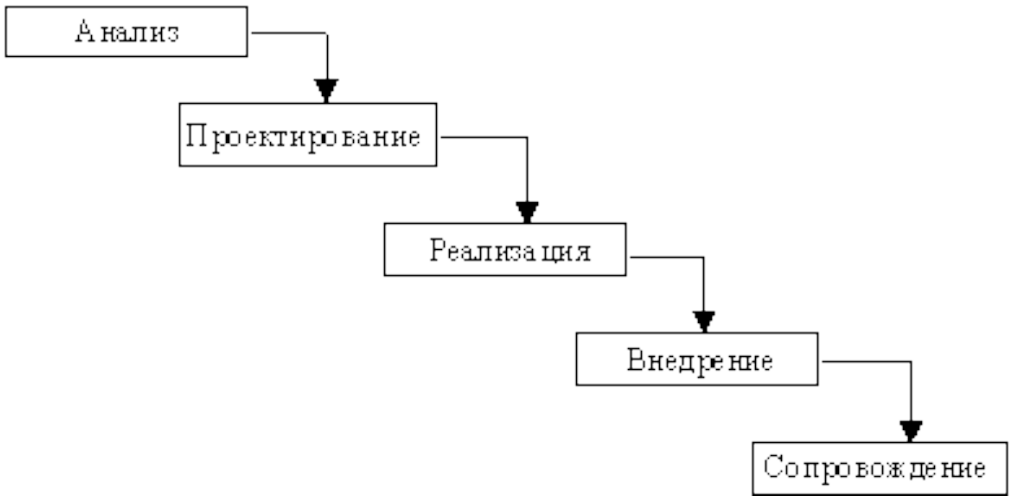

Рис. 1.6. Каскадная схема разработки ПО

Создание системы рассматривается как однократная последовательность стадий и этапов. Каждый этап завершается выпуском полного комплекта документации, достаточной для того, чтобы разработка могла быть продолжена другой командой разработчиков.

Достоинства каскадного подхода:

1) на каждом этапе формируется законченный набор проектной документации, отвечающий критериям полноты и согласованности;
2) выполняемые в логичной последовательности этапы работ позволяют планировать сроки завершения всех работ и соответствующие затраты.

Недостатки каскадного подхода выявились в процессе его использования. Дело в том, что реальный процесс создания ПО почти никогда полностью не укладывается в жесткую схему каскадной модели. В процессе создания ПО может возникнуть потребность возврата к предыдущему этапу и уточнения или пересмотра ранее принятых решений.

В результате реальный процесс создания ПО часто принимает следующий вид (рис. 1.7.):

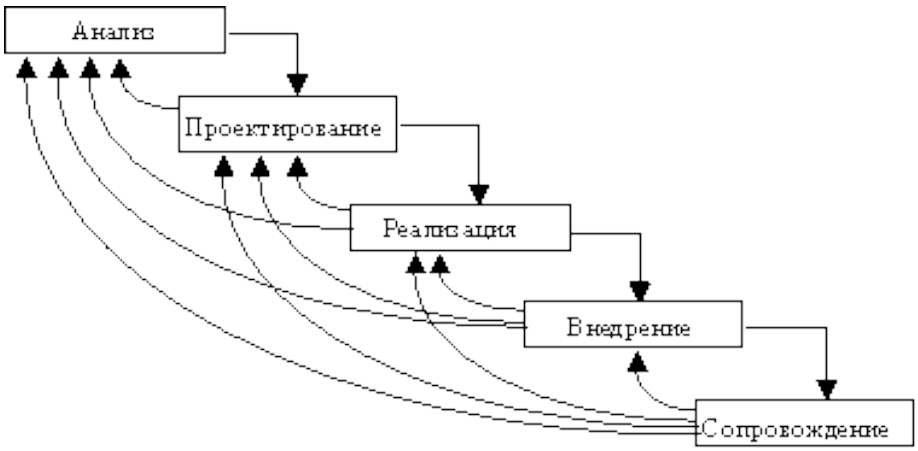

Рис. 1.7. Реальный процесс разработки ПО по каскадной схеме

Основной недостаток каскадного подхода – существенное запаздывание с получением результатов. Согласование результатов с пользователями производится только в точках, планируемых после завершения каждого этапа работ, требования к ИС «заморожены» в виде технического задания на все время ее создания.

Таким образом, пользователи могут внести свои замечания только после того, как работа над системой будет полностью завершена. В случае неточного изложения требований или их изменения в течение длительного периода создания ПО, пользователи получают систему, не удовлетворяющую их потребностям. Модели (как функциональные, так и информационные) автоматизируемого объекта могут устареть одновременно с их утверждением.

Область применения каскадного подхода (где каскадный подход хорошо зарекомендовал себя):

1) однородные ИС, где каждое приложение представляет собой единое целое;
2) ИС, для которых в самом начале разработки можно достаточно точно и полно сформулировать все требования, с тем, чтобы предоставить разработчикам свободу реализовать их как можно лучше с технической точки зрения. В эту категорию попадают сложные расчетные системы, системы реального времени и другие подобные системы.

Oracle CDM (Custom Development Method, **CDM**) – методика разработки ИС небольших масштабов на основе Oracle, ориентированная на использование *каскадной модели ЖЦ*.

Методика разработки Oracle CDM (Custom Development Method)

Компания Oracle, долгие годы специализирующаяся в области средств автоматизации процессов разработки сложных информационных систем, ориентированных на интенсивное использование баз данных, предлагает свою методику. Программный комплекс разработчика, выпускаемый Oracle, носит название Oracle Designer (раннее Oracle CASE). *Основные положения методики Oracle:*

1) проектирование имеет структурный нисходящий характер, весь процесс разработки системы представляется в виде последовательности чётко определённых этапов;
2) поддержка осуществляется на всех этапах жизненного цикла системы, начиная от общих предложений и заканчивая сопровождением готового продукта;
3) предпочтение отдаётся архитектуре клиент-сервер, включая сложные структуры распределённых баз данных;
4) во время разработки все спецификации проекта хранятся в специальной базе данных (репозитарии). Данное средство включено в состав ПО Designer и работает под управлением СУБД Oracle. Репозитарий позволяет подключаться к нему большому числу пользователей с различными уровнями прав доступа путём стандартных средств СУБД Oracle. В результате все действия разработчиков становятся строго согласованными, невозможно независимое действие каждого из них;
5) последовательный переход от одного этапа к другому автоматизирован благодаря использованию специальных утилит. С помощью них по спе-

цификациям концептуальной стадии можно получить первоначальный вариант спецификации уровня проектирования. В дальнейшем генерация дополнений значительно упрощается;

6) стандартные действия этапов проектирования и разработки автоматизированы. В любой момент может быть сгенерирован любой объём отчётов по содержимому репозитария, которые обеспечивают документирование текущей версии системы на всех этапах её разработки. Имеются специальные процедуры, позволяющие осуществить проверку спецификаций на полноту и непротиворечивость.

Методика Oracle предлагает свой вариант структуры жизненного цикла информационной системы:

1) **формирование стратегии.** Здесь происходит моделирование и анализ процессов, которые описывают деятельность организации, особенности работы. Конечная цель – создание моделей процессов, выявление возможностей их усовершенствования. Этап не обязателен, если существующие технологии и организационные структуры чётко определены, хорошо изучены и в целом понятны;
2) **анализ.** На этом этапе происходит разработка детальных концептуальных моделей, которые описывают информационные потребности организации, особенности функционирования. В результате формируются модели двух типов: информационные (они отражают структуру и общие закономерности предметной области) и функциональные (описывают особенности решаемых задач);
3) **проектирование.** Здесь начальные требования к системе преобразуются в детальные спецификации. Специальные утилиты, входящие в состав ПО Oracle, значительно упрощают эту процедуру;
4) **реализация**. На этом этапе разрабатываются и тестируются приложения, входящие в состав будущей ИС. ПО Oracle содержит специальные генераторы приложений, которые предельно автоматизируют этот этап,

обычно самый сложный и продолжительный. Сроки разработки значительно снижаются;

5) **внедрение.** На этой стадии система устанавливается, подготавливается к началу эксплуатации. Происходит подготовка персонала;
6) **эксплуатация.** Поддержка системы, планирование будущих дополнений и изменений.

Интересно отметить, что Oracle не рассматривает такую стадию существования ИС, как снятие с эксплуатации. То есть разработчик, применяющий методику и ПО Oracle, автоматически «подсаживается» на него. Другое дело, что удобство средств разработки во многом оправдывает такую зависимость.

Особенности методики Oracle CDM:

1) по степени адаптивности методика CDM предлагает три модели жизненного цикла: *классическая* – предусматривает все этапы, *быстрая разработка* – с использованием инструментов моделирования и программирования Oracle, *облегченный подход* – для малых проектов;
2) методикой не предусмотрено включение дополнительных задач, которые не оговорены в CDM, не предусмотрена тем более привязка их к остальным. Кроме того, невозможно удаление задачи и порождаемых ею документов, которое не было предусмотрено ни в одной из трёх моделей жизненного цикла ИС, невозможно изменение последовательности выполнения задач относительно предложенной;
3) модель жизненного цикла ИС в Oracle CDM является, по сути, каскадной;
4) методика не является обязательной, хотя и является фирменным стандартом, однако в случае использования ПО Oracle иной подход маловероятен;
5) методика опирается на использование ПО разработчика от Oracle, приспособление её к другим средствам затруднительно.

Спиральная модель ЖЦ

Поскольку программные проекты отличаются от других, например, строительных проектов, то и управлять ими тоже нужно по-другому. Наглядным подтверждением этого является тот факт, что к концу 1980-х гг. Министерство обороны США начало испытывать серьезные трудности с разработкой ПО в соответствии с жесткой, основанной на директивных документах и предусматривающей один проход модели, как это требовалось стандартом DoD-Std-2167A. Проведенная позже в 1999 проверка по выборке ранее утвержденных в Министерстве обороны проектов дала удручающие результаты. Из общего числа входящих в выборку проектов, на реализацию которых было выделено 37 млрд долл., 75% проектов закончились неудачей или выделенные на них средства не были использованы, и только 2% выделенных средств были использованы без значительной модификации проектов. В результате в конце 1987 г. Министерство отступило от стандартов на базе каскадной модели и допустило применение **итерационного подхода.**

Истоки концепции итерационной разработки прослеживаются в относящихся к 1930-м годам работах эксперта по проблемам качества продукции Уолтера Шеварта из компании Bell Labs, который предложил ориентированную на повышение качества методику, состоящую из серии коротких циклов шагов по планированию, реализации, изучению и действию (plan-do-study-act, PDSA). С 1940-х годов энергичным поборником PDSA стал известный авторитет в области качества Эдварде Деминг. В более поздних работах PDSA была исследована применительно к разработке ПО.

В середине 1980-х годов Барри Боэм предложил свой вариант итерационной модели под названием *«спиральная модель»* (spiral model).

Принципиальные особенности спиральной модели:

- отказ от фиксации требований и назначение приоритетов пользовательским требованиям;

- разработка последовательности прототипов, начиная с требований наивысшего приоритета;
- идентификация и анализ риска на каждой итерации;
- использование каскадной модели для реализации окончательного прототипа;
- оценка результатов по завершении каждой итерации и планирование следующей итерации.

При использовании спиральной модели прикладное ПО создается в несколько итераций (витков спирали) методом прототи-пирования. Под *прототипом* понимается действующий программный компонент, реализующий отдельные функции и внешние интерфейсы разрабатываемого ПО. Создание прототипов осуществляется в несколько итераций, или витков спирали. Каждая итерация соответствует созданию фрагмента или версии ПО, на ней уточняются цели и характеристики проекта, оценивается качество полученных результатов и планируются работы следующей итерации. На каждой итерации производится тщательная оценка риска превышения сроков и стоимости проекта, чтобы определить необходимость выполнения еще одной итерации, степень полноты и точности понимания требований к системе, а также целесообразность прекращения проекта.

Спиральная модель избавляет пользователей и разработчиков ПО от необходимости полного и точного формулирования требований к системе на начальной стадии, поскольку они уточняются на каждой итерации. Таким образом, углубляются и последовательно конкретизируются детали проекта, и в результате выбирается обоснованный вариант, который доводится до реализации.

Для преодоления перечисленных проблем была предложена **спиральная модель ЖЦ** (рис. 1.8.), делающая упор на начальные этапы ЖЦ: *анализ* и *проектирование*.

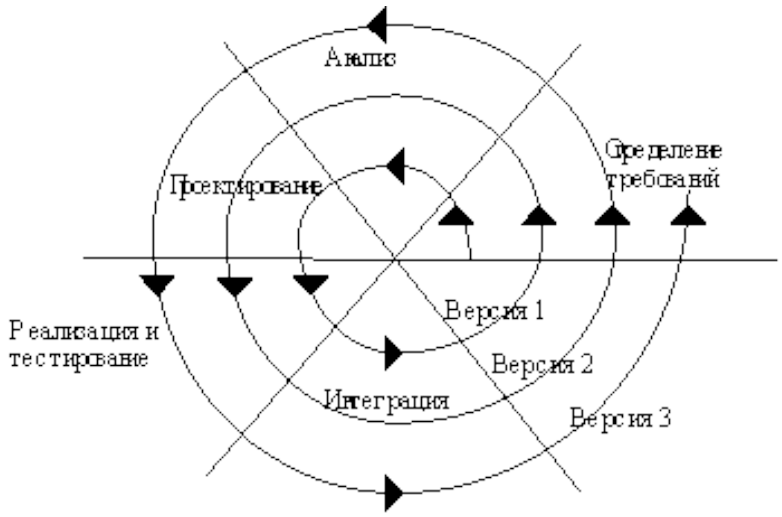

Рис. 1.8. Спиральная модель ЖЦ

На этих этапах реализуемость технических решений проверяется путем *создания прототипов*. Каждый виток спирали соответствует созданию фрагмента или версии ПО, на нем уточняются цели и характеристики проекта, определяется его качество и планируются работы следующего витка спирали. Таким образом, углубляются и последовательно конкретизируются детали проекта и в результате выбирается обоснованный вариант, который доводится до реализации.

Разработка итерациями отражает объективно существующий спиральный цикл создания системы. Неполное завершение работ на каждом этапе позволяет переходить на следующий этап, не дожидаясь полного завершения работы на текущем. При итеративном способе разработки недостающую работу можно будет выполнить на следующей итерации. Главная же задача – как можно быстрее показать пользователям системы работоспособный продукт, тем самым активизируя процесс уточнения и дополнения требований.

Основная проблема спирального цикла – определение момента перехода на следующий этап. Для ее решения необходимо ввести временные ограничения на каждый из этапов жизненного цикла. Переход осуществляется в соответствии с *планом*, даже если не вся запланированная работа закончена. *План* составляется на основе статистических данных, полученных в предыдущих проектах, и личного опыта разработчиков.

В рамках спиральной модели была разработана *методология RAD* (Rapid Application Development, 1980 – разработка подхода, James Martin в IBM, 1991 – публикация), использующая инструментальные средства быстрой разработки ПО.

Методология быстрой разработки приложений RAD

Методология быстрой разработки приложений RAD (Rapid Application Development) основана на использовании *средств быстрой разработки приложений* и ориентирована на использование *спиральной модели ЖЦ*. При проектировании использует *объектно-ориентированные методы* описания предметной области. *Специфика RAD*:

1) небольшая команда программистов (от 2 до 10 человек). При этом команда разработчиков должна представлять из себя группу профессионалов, имеющих опыт в анализе, проектировании, генерации кода и тестировании ПО с использованием CASE-средств. Члены коллектива должны также уметь трансформировать в рабочие прототипы предложения конечных пользователей;
2) короткий, но тщательно проработанный производственный график (от 2 до 6 мес.);
3) повторяющийся цикл, при котором разработчики, по мере того, как приложение начинает обретать форму, запрашивают и реализуют в продукте требования, полученные через взаимодействие с заказчиком;

Основные принципы методологии RAD:

- разработка приложений итерациями;
- необязательность полного завершения работ на каждом из этапов жизненного цикла;
- обязательное вовлечение пользователей в процесс разработки ИС;
- обязательное применение CASE-средств, обеспечивающих целостность проекта;

- применение средств управления конфигурацией, облегчающих внесение изменений в проект и сопровождение готовой системы;
- необходимое использование генераторов кода;
- использование прототипирования, позволяющее полнее выяснить и удовлетворить потребности конечного пользователя;
- тестирование и развитие проекта, осуществляемые одновременно с разработкой;
- ведение разработки немногочисленной хорошо управляемой командой профессионалов (обычно от 2 до 10 человек);
- грамотное руководство разработкой системы, четкое планирование и контроль выполнения работ.

Жизненный цикл ПО по методологии RAD состоит из четырех фаз: 1) *анализа и планирования требований*; 2) *проектирования*; 3) *построения*; 4) *внедрения*.

1. На фазе анализа и планирования требований определяются функции системы, выделяются наиболее приоритетные из них, требующие проработки в первую очередь, описываются информационные потребности. Определение требований выполняется в основном пользователями системы при помощи специалистов-разработчиков. Ограничивается масштаб проекта, определяются временные рамки для каждой из последующих фаз. Кроме того, определяется сама возможность реализации данного проекта в установленных рамках финансирования, на данных аппаратных средствах и т.п. Результат данной фазы:

1) список и приоритетность функций будущей ИС;
2) предварительные функциональные и информационные модели ИС.

2. На фазе проектирования пользователи принимает участие в техническом проектировании системы под руководством специалистов-разработчиков при помощи CASE-средств. Уточняются и дополняются требования к системе, которые не были выявлены на предыдущей фазе, а именно:

- более подробно рассматриваются процессы системы. Анализируется и, при необходимости, корректируется функциональная модель. Каждый процесс рассматривается детально. При необходимости для каждого элементарного процесса создается частичный прототип: экран, диалог, отчет, устраняющий неясности или неоднозначности;
- определяются требования разграничения доступа к данным;
- определяется набор необходимой документации;
- оценивается количество функциональных элементов разрабатываемой системы и принимается решение о разделении ИС на подсистемы, поддающиеся реализации одной командой разработчиков за приемлемое для RAD-проектов время (порядка 60 – 90 дней);
- с использованием CASE-средств проект распределяется между различными командами (делится функциональная модель).

Результат фазы проектирования:

1) общая информационная модель системы;
2) функциональные модели системы в целом и подсистем, реализуемых отдельными командами разработчиков;
3) точно определенные с помощью CASE-средства интерфейсы между автономно разрабатываемыми подсистемами;
4) построенные прототипы экранов, отчетов, диалогов.

Все модели и прототипы должны быть получены с применением тех CASE-средств, которые будут использоваться в дальнейшем при построении системы. Данное требование вызвано тем, что в традиционном подходе при передаче информации о проекте с этапа на этап может произойти фактически неконтролируемое искажение данных. Применение единой среды хранения информации о проекте позволяет избежать этой опасности.

В отличие от традиционного подхода, при котором использовались специфические средства прототипирования, не предназначенные для построения реальных приложений, а прототипы выбрасывались после того, как выполняли задачу устранения неясностей в проекте, в подходе RAD каждый прототип раз-

вивается в часть будущей системы. Таким образом, на следующую фазу передается более полная и полезная информация.

3. На фазе построения выполняется непосредственно сама быстрая разработка приложения. На данной фазе разработчики производят итеративное построение реальной системы на основе полученных в предыдущей фазе моделей, а также требований нефункционального характера. Программный код частично формируется при помощи автоматических генераторов, получающих информацию непосредственно из репозитория CASE-средств. Конечные пользователи на этой фазе оценивают получаемые результаты и вносят коррективы, если в процессе разработки система перестает удовлетворять определенным ранее требованиям. *Тестирование системы* осуществляется непосредственно в процессе разработки. После окончания работ каждой отдельной команды разработчиков производится постепенная интеграция данной части системы с остальными, формируется полный программный код, выполняется тестирование совместной работы данной части приложения с остальными, а затем тестирование системы в целом. Завершение физического проектирования системы может выглядеть следующим образом:

1) определяется необходимость распределения данных;
2) производится анализ использования данных;
3) производится физическое проектирование базы данных;
4) определяются требования к аппаратным ресурсам;
5) определяются способы увеличения производительности;
6) завершается разработка документации проекта.

Результат фазы построения – готовая система, удовлетворяющая всем согласованным требованиям.

4. На фазе внедрения производится обучение пользователей, организационные изменения и параллельно с внедрением новой системы осуществляется работа с существующей системой (до полного внедрения новой). Так как фаза построения достаточно непродолжительна, *планирование* и *подготовка к внедрению* должны начинаться заранее, как правило, на *этапе проектирования системы*.

Приведенная схема разработки ИС не является абсолютной. Возможны различные варианты, зависящие, например, от начальных условий, таких как:

- разрабатывается совершенно новая система;
- уже было проведено обследование предприятия и существует модель его деятельности;
- на предприятии уже существует некоторая ИС, которая а)может быть использована в качестве начального прототипа или б)должна быть интегрирована с разрабатываемой.

Ограничения методологии RAD. Как и любая методология, RAD, несмотря на все достоинства, не может претендовать на универсальность. Ее применение наиболее эффективно при выполнении сравнительно небольших систем, разрабатываемых для вполне определенного предприятия. Методология RAD неприменима:

1) для создания *типовых ИС*, которые не являются законченным программным продуктом, а представляют собой совокупность типовых элементов ИС, и где большое значение имеют такие показатели проекта, как управляемость и качество, что может войти в противоречие с простотой и скоростью разработки. Для типовых проектов, которые обычно централизованно сопровождаются и могут быть адаптированы к различным программно-аппаратным платформам, СУБД, коммуникационным средствам, а также интегрироваться с существующими разработками, необходим высокий уровень планирования и жесткая дисциплина проектирования, строгое следование заранее разработанным протоколам и интерфейсам, что снижает скорость разработки;
2) для построения программ, требующих написания большого объема уникального кода, например: *сложных расчетных программ, операционных систем, программ управления сложными инженерно-техническими объектами*;
3) для разработки приложений, в которых интерфейс пользователя является вторичным, т.е. отсутствует наглядное определение логики работы си-

стемы, например для создания приложений реального времени, драйверов и служб;

4) для разработки систем, от которых зависит безопасность людей, например, систем управления транспортом или АЭС. Это обусловлено тем, что итеративный подход, являющийся одной из основ RAD, предполагает, что первые версии системы не будут полностью работоспособны.

Итерационный подход

Естественное развитие каскадной и спиральной моделей привело к их сближению и появлению современного итерационного подхода, который по существу представляет собой рациональное сочетание этих моделей (рис. 1.9).

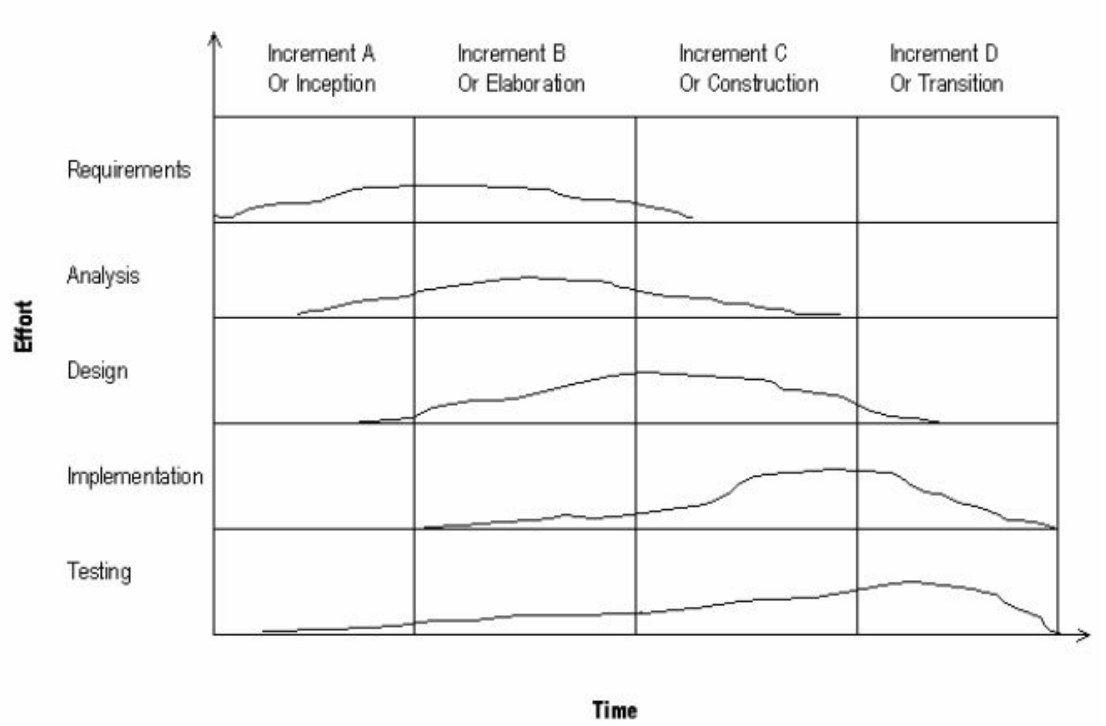

Рис. 1.9. Итеративная модель ЖЦ

Различные варианты итерационного подхода реализованы в большинстве современных методов: Rational Unified Process (RUP), Microsoft Solutions Framework (MSF), XR и др.

1. **RUP** (Rational Unified Process, компания Rational Software – подразделение IBM, 2003) – использует итеративную модель разработки, включаю-

щую четыре фазы (начало, исследование, построение, внедрение), разбитых на итерации, каждая из которых завершается получением промежуточной, но функциональной версии конечного продукта. RUP опирается на интегрированный комплекс инструментальных средств Rational Suite, в состав которого, кроме самой технологии RUP как продукта, входят такие компоненты, как:

- Rational Rose – средство визуального моделирования (анализа и проектирования), использующее язык UML;
- Rational XDE – средство анализа и проектирования, интегрируемое с платформами MS Visual Studio .NET и IBM WebSphere Studio Application Developer и др.;

2. **MSF** (Microsoft Solution Framework, 1993 – v.1.0, 2002 – v.1.0, 2005 – v.4.0) сходна с RUP, так же включает четыре фазы: анализ, проектирование, разработка, стабилизация, является итерационной и предполагает использование объектно-ориентированного моделирования. MSF в сравнении с RUP в большей степени ориентирована на разработку бизнес-приложений.
3. **Методология XP** (Extreme Programming, Экстремальное программирование, разработана в 1996, опубликована в 1999 г.). Разработка представляет собой итеративный процесс, где фазы разбиваются на «экстремально» малые шаги. В основе методологии – командная работа и эффективная коммуникация между заказчиком и исполнителем.

1.6. Парадигмы проектирования ИС

В настоящее время известны две парадигмы проектирования, использующие два различных подхода к описанию систем:

- *структурная* (процессно-ориентированная), основанная на каскадной (водопадной) модели жизненного цикла ИС (см. рис. 1.7);

- *объектно-ориентированная*, основанная на итеративной модели ЖЦ ИС (см. рис. 1.9).

Структурный анализ (Structured Analysis, SA) и **структурное проектирование** (Structured Design, SD) – результат появившегося в 1970-х структурного программирования, развивался из классического системного анализа. Методологии структурного анализа используют каскадную (водопадную) модель ЖЦ ИС; самые известные и используемые методологии структурно анализа – SADDT, SSADM. Методы структурного анализа дорабатывались и используются уже на протяжении многих лет. Структурному анализу и структурному проектированию посвящен раздел 4 данной книги.

Сравнительно позже появились и стали невероятно популярны объектно-ориентированные языки. По мере нарастания их популярности была разработана методология помощи программисту в разработке приложений с использованием объектно-ориентированных языков. Эта методология стала известна как **объектно-ориентированный анализ и проектирование** (object-oriented analysis and design, **OOAD**).

OOAD – это подход к инженерии ПО, моделирующий систему как группу взаимодействующих объектов. *Объектно-ориентированный анализ* (Object-oriented analysis, **OOA**) использует методы объектного моделирования для анализа функциональных требований к системе.

Объектно-ориентированное проектирование, ООП (Object-oriented design, OOD) разрабатывает аналитические модели для создания спецификаций реализации (например, ТЗ). Концептуальной основой ООП является *объектная модель*, которая строится с учетом принципов *абстрагирования*, *инкапсуляции*, *модульности*, *иерархии*, *типизации*, *параллелизма*, *устойчивости*.

Основными понятиями объектно-ориентированного подхода являются *объект и класс*. *Объект* – представляет собой определенную *сущность*, соответствующую значимому предмету или явлению предметной области, характеризуется *классом*, *состоянием* (state (data elements)) и *поведением*. Для этих взаимодействующих (collaborating) между собой объектов можно создать раз-

личные модели, характеризующие *статическую структуру*, *динамическое поведение* и развертывание в действии (run-time deployment). *Класс* – это множество объектов, связанных общностью структуры и *поведения*.

Следующую группу важных понятий объектного подхода составляют *полиморфизм* (способность класса принадлежать более чем одному типу) и *наследование* (построение новых классов на основе существующих с возможностью добавления или переопределения данных и методов).

На сегодняшний день существует более тридцати объектно-ориентированных методов проектирования (например, IDEF4 – Object-Oriented Design – методология ООП, позволяющая отображать структуру объектов и принципы их взаимодействия) с множеством различных нотаций представления объектных моделей.

Два самых популярных стандартных языка объектно-ориентированного моделирования – UML (The Unified Modeling Language) и SysML.

Унифицированный язык моделирования UML (Unified Modeling Language) – открытый стандарт, использующий графические обозначения для создания абстрактной модели системы (UML-модели); был создан для определения, визуализации, проектирования и документирования в основном программных приложений. Методы описания результатов анализа и проектирования семантически близки к методам программирования на современных объектно-ориентированных языках. На основании UML-модели возможна генерация программного кода. С помощью UML можно также разработать детальный план создаваемой системы, содержащий системные функции и бизнес процессы, схемы баз данных, программные компоненты многократного использования. UML позволяет рассмотреть систему со всех точек зрения, имеющих отношение к ее разработке и последующему развертыванию.

UML обеспечивает поддержку всех этапов жизненного цикла ИС и предоставляет для этих целей следующие набор диаграмм: *структурные диаграммы (Structure Diagrams); диаграммы поведения (Behavior Diagrams);диаграммы взаимодействия (Interaction Diagrams).*

UML фокусируется не трех архитектурных видениях системы:

- *функциональном* (описывает внешнее поведение системы с точки зрения пользователя с помощью *use-case диаграмм*);
- *статическом* (в терминах *атрибутов*, *методов*, *связей*, *классов*, *сообщений* с помощью *CRC cards*, *class diagrams*, *object diagrams*);
- *динамическом* (*sequence diagrams*, *collaboration diagrams*, *statecharts*)

и имеет практическое значение в описании *сервисно-ориентированных архитектур* (SOAs).

Формальная спецификация последней версии UML 2.0 опубликована в августе 2005 года. Семантика языка уточнялась и была расширена для поддержки методологии Model Driven Development, MDD. Последняя версия UML 2.4.1 опубликована в августе 2011 года. UML 2.4.1 принят в качестве международного стандарта ISO/IEC 19505-1, 19505-2.

Проблематика выбора. Появление новых и все более сложных объектно-ориентированных языков должно было, как представлялось, увеличить потребность в использовании объектного подхода для разработки бизнес приложений. Но так ли это на самом деле, дает ли популярность объектно-ориентированного программирования гарантию преимущества нового объектного подхода к проектированию перед традиционной структурной методологией, остается вопросом. Ниже перечислены преимущества и недостатки структурной и объектно-ориентированных методологий.

Основные преимущества структурных методологий:

- основаны на классическом системном анализе и процессно-ориентированы, подходят для описания любых (не только информационных) систем, идеальны для исследования предметной области, реинжиниринга бизнес процессов;
- понятное и относительно простое визуальное представление системы, легко понимаемое как пользователями, так и разработчиками;
- акцент на командной работе;

- четко определенные этапы, что облегчает управление проектом и позволяет разрабатывать системы лучшего качества;
- допускают использование средств проверки требований;
- SSAD и IDEF0 – классические, широко известные методологии в области проектирования ИС, существующие на протяжении длительного времени и достаточно «зрелые»;

Недостатки структурных методологий:

- поскольку процессно-ориентированы, то соответственно, игнорируют нефункциональные требования: не идеальные решения при использовании объектно-ориентированных языков программирования, т.к. изначально создавались для структурных языков;
- поскольку SSAD и SSADM неитеративны, то изменение требований может привести к перезапуску всего процесса разработки;
- возможны трудности при определении глубины декомпозиции – момента, когда нужно остановиться и переходить к реализации модели;

Преимущества объектно-ориентированных методологий:

- упрощение и ускорение программной реализации системы по сравнению со структурными методологиями;
- повторное использование кода в других проектах, благодаря независимости объектов и инкапсуляции, что сокращает стоимость проектирования, программирования и проверки; повторное использование кода может способствовать улучшению качества последующих проектов («Если 90% нового приложения содержит проверенные компоненты, то только 10% кода должна быть проверена с нуля» (Vivek Shah);
- отсутствие разделения между фазами анализа и разработки обеспечивает взаимодействие с пользователями до самого конца проекта;
- аналитики и программисты не связаны ограничениями внедрения системы, поэтому могут формулировать проекты, которые будут соответствовать различным средам исполнения;

- программное обеспечение устойчиво к изменениям, что обеспечивает более высокий уровень уверенности в его корректности, способствуя снижению рисков при разработке сложных систем;
- те преимущества, которые представляет объектно-ориентированное программирование по сравнению со структурным: при разработке объектов со сложным взаимодействием, аналитик думает на ином уровне детализации, чем это возможно в структурном коде, т.е. об атрибутах объекта; стандартизация объектов увеличивает степень понимания проекта.

Недостатки объектно-ориентированных методологий:

- изначальная модель слишком упрощена для того, чтобы быть адекватной;
- чрезмерная фокусировка на коде;
- не так много внимания уделяется командной работе, как в структурных методологиях;
- определение всех необходимых для системы классов и объектов – это не такая, на самом деле, простая задача;
- попытка сочетания объектного программирования с анализом различных функций системы; однако, эти функциональные методы совершенно не соответствуют OOAD;
- преувеличение значимости и универсальности объектной методологии, когда, фактически, другой подход мог бы подойти лучше для анализа и разработки системы в зависимости от конкретных обстоятельств;
- требует новый вид управления проектами, который включает различные типы анализа, отличные от традиционного функционального подхода декомпозиции. Следовательно, для команд разработчиков проекта, которые имеют долгую историю использования методологий SSAD и SSADM, переход к методологии OOAD является чрезвычайно сложным, длительным и трудоемким (Hantos, 2005);
- другой недостаток – это сама концепция повторного использования, которая заявляется как крупное преимущество и причина для перехода на OOAD. Однако, без явной процедуры повторного использования многие

из систем, разработанных с помощью данной методологии, не ведут к успешному повторному использованию в больших масштабах (Hantos, 2005);

- функциональное описание системы в UML основано на сценариях использования, которые подходят для документирования требований, не основанных на взаимодействии с системой (таких как алгоритм или математические требования) или нефункциональных требований (такие как платформа, производительность, синхронизация, безопасность); следование шаблонам не гарантирует качества сценариев, качество зависит только от навыков создателя сценария;
- объектный подход к моделированию данных при том, что большинство ИС используют реляционные модели; В конце концов, ИС чаще разрабатываются через комбинацию объектно-ориентированных языков программирования и реляционных баз данных, а не полностью объектные базы данных и языки.

В одной из опубликованных в 2001 году работ (Sircar, Nerur, and Mahapatra) было сделано интересное замечание: «недавний опрос ИТ менеджеров показал, что 39% организаций приняли ОО технологии в той или иной форме. Тем не менее, ОО методологии проектирования используются только в 5% ИТ проектов»

Заключение. При разработке конкретного приложения первая задача – решить, какая методология наиболее подходит для этой разработки. Иногда, возможно, придется адаптировать различные методологи. Существует мнение, что для решения простых задач лучше подходят структурные методы, в то время как объектно-ориентированные методы уместнее для более высоких уровней абстракции. В ситуациях, где вероятность изменения данных выше, чем изменение функциональности, объектные методы также будут более подходящими.

2. НОРМАТИВНО-МЕТОДИЧЕСКАЯ ПОДДЕРЖКА ЖЦ ИС

2.1. Нормативно-методическое обеспечение (НМО) ЖЦ ИС

Методологии создания ИС описывают поддержку определенной модели жизненного цикла (ЖЦ) ИС и реализуется через *стандарты* и *методики*, а также поддерживающие их *технологии* и инструментальные средства (*CASE-средства*).

Комплекс документов, регламентирующих деятельность разработчиков называют *нормативно-методическим обеспечением (НМО)*. Входящие в состав НМО документы – это стандарты, руководящие документы, методики, положения, инструкции, шаблоны и т.п., которые регламентируют:

- порядок разработки, внедрения и сопровождения ПО;
- общие требования к составу, связям между компонентами и качеству ПО;
- виды, состав и содержание проектной и программной *документации*.

В СССР в 1970-е годы процесс создания ПО регламентировался стандартами ЕСПД (Единой Системы Программной Документации – серия ГОСТ 19.XXX), которые были ориентированы на класс относительно простых программ небольшого объема, создаваемых отдельными программистами в период «*лоскутной автоматизации*».

Стандарты ЕСПД практически лишены содержательной составляющей и содержат формальные требования к составу, содержанию и оформлению документов, описывающих программу на разных стадиях ее жизненного цикла, а также порядку хранения и обновления документации. Многие авторы считают эти стандарты морально устаревшими (концептуально и по форме), тем не менее, ими продолжают активно пользоваться при оформлении проектной документации.

Альтернативной современной нормативной базой НМО являются следующие международные и отечественные стандарты (сгруппированы **по статусу**):

1) официальные (public) стандарты:
   - международные:
     - ISO/IES (ISO – International Organization of Standardization, международная организация по стандартизации; IEC – International Electrotechnical Commission, международная комиссия по электротехнике);
     - ANSI (American National Standards Institute);
     - стандарты международных консорциумов и комитетов по стандартизации (например, консорциума OMG);
   - стандарты Российской Федерации (ГОСТ):
     - ГОСТ 34.601-90 «Информационная технология. Комплекс стандартов на автоматизированные системы. Автоматизированные системы. Стадии создания»;
     - ГОСТ 34.602-89 «Информационная технология. Комплекс стандартов на автоматизированные системы. Техническое задание на создание автоматизированной системы»;
     - ГОСТ 34.603-92 «Информационная технология. Виды испытаний автоматизированных систем».
   - отраслевые стандарты,
   - ведомственные стандарты.
2) Стандарты «де-факто» – официально никем не утвержденные, но фактически действующие (например, стандартом «де-факто» долгое время были языки взаимодействия с реляционными базами данных SQL и язык программирования C), фирменные стандарты (например, Microsoft ODBC).

Другие способы классификации стандартов:

по объекту стандартизации:

- стандарты на продукты и услуги;
- стандарты на процессы и технологии;
- стандарты на формы коллективной деятельности, или управленческие стандарты;

по предмету стандартизации:

1) *функциональные стандарты*:
   - стандарты на языки программирования;
   - стандарты на интерфейсы и протоколы и др.;
2) *стандарты на организацию ЖЦ ИС*.

Отдельно выделяют корпоративные стандарты.

Корпоративные стандарты. Реальное применение любой технологии проектирования, разработки и сопровождения ИС в конкретной организации и конкретном проекте невозможно без выработки ряда стандартов (правил, соглашений), которые должны соблюдаться всеми участниками проекта. Для большинства сложных проектов приходится создавать свои комплекты нормативных и методических документов, регламентирующих процессы, этапы, работы и документы конкретных программных продуктов. Такие стандарты называются **корпоративными** и представляют собой соглашение об единых правилах организации технологии или управления в организации. К таким стандартам относятся:

- стандарты проектирования;
- стандарты оформления проектной документации;
- стандарты пользовательского интерфейса.

Стандарт проектирования должен устанавливать:

- набор необходимых моделей (диаграмм) на каждой стадии проектирования и степень их детализации;
- правила фиксации проектных решений на диаграммах, в том числе: правила именования объектов (включая соглашения по терминологии), набор

атрибутов для всех объектов и правила их заполнения на каждой стадии, правила оформления диаграмм, включая требования к форме и размерам объектов, и т. д.;

- требования к конфигурации рабочих мест разработчиков, включая настройки операционной системы, настройки CASE-средств, общие настройки проекта и т. д.;
- механизм обеспечения совместной работы над проектом, в том числе: правила интеграции подсистем проекта, правила поддержания проекта в одинаковом для всех разработчиков состоянии (регламент обмена проектной информацией, механизм фиксации общих объектов и т.д.), правила проверки проектных решений на непротиворечивость и т.д.

Стандарт оформления проектной документации должен устанавливать:

- комплектность, состав и структуру документации на каждой стадии проектирования;
- требования к ее оформлению (включая требования к содержанию разделов, подразделов, пунктов, таблиц и т.д.),
- правила подготовки, рассмотрения, согласования и утверждения документации с указанием предельных сроков для каждой стадии;
- требования к настройке издательской системы, используемой в качестве встроенного средства подготовки документации;
- требования к настройке CASE-средств для обеспечения подготовки документации в соответствии с установленными требованиями.

Стандарт интерфейса пользователя должен устанавливать:

- правила оформления экранов (шрифты и цветовая палитра), состав и расположение окон и элементов управления;
- правила использования клавиатуры и мыши;
- правила оформления текстов помощи;

- перечень стандартных сообщений;
- правила обработки реакции пользователя.

За основу корпоративных стандартов могут приниматься *отраслевые*, *национальные* или *международные* стандарты. Сюда могут относятся различного рода методические материалы ведущих фирм-разработчиков ПО, фирм-консультантов, научных центров, консорциумов по стандартизации.

2.2. Стандарты на процессы ЖЦ ИС

ISO/IEC 12207 (по определению) – базовый стандарт на процессы ЖЦ ИС, ориентированный на различные типы проектов ИС. В стандарте не предусмотрено каких-либо этапов ЖЦ ИС, а определен лишь ряд процессов. Стандарт позволяет реализовать любую модель ЖЦ.

ГОСТ 34.601-90 – распространяется на АИС и устанавливает стадии и этапы их создания, содержит описание содержания работ на каждом этапе. Стандарт ориентирован на использование *каскадной модели* ЖЦ.

ISO/IEC 12207:1995-08-01 и сопутствующие стандарты

Первая редакция ISO 12207 была подготовлена в 1995 г. объединенным техническим комитетом ISO/IEC JTC1 «Информационные технологии, подкомитет SC7, проектирование программного обеспечения».

Международный стандарт **ISO/IEC 12207** является основным нормативным документом, регламентирующим ЖЦ ПО. Он определяет структуру ЖЦ, содержащую процессы, действия и задачи, которые должны быть выполнены во время создания ПО. Его регламенты являются общими для любых моделей ЖЦ, методологий и технологий разработки ПО. Способы выполнения действий задач, включенных в перечисленные процессы, могут быть любыми.

В соответствии с базовым международным стандартом ISO/IEC 12207 все процессы ЖЦ ПО делятся на три группы.

I. **Основные процессы:**

1) процесс приобретения – определяет действия предприятия-покупателя;
2) процесс поставки – определяет действия предприятия-поставщика;
3) процесс разработки – определяет действия предприятия-разработчика;
4) процесс функционирования – определяет действия предприятия-оператора, которое обеспечивает обслуживание системы в целом (а не только ПО) в процессе ее функционирования в интересах пользователя;
5) процесс сопровождения – определяет действия персонала, обеспечивающего сопровождение программного продукта, т.е. управление модификациями, поддержку текущего состояния и функциональной пригодности. Сюда же относится установка программного изделия на вычислительной системе и его удаление.

II. **Вспомогательные процессы** – предназначены для поддержки выполнения основных процессов, обеспечения качества проекта, организации верификации, проверки и тестирования ПО:

1) процесс документирования;
2) процесс управления конфигурацией;
3) процесс обеспечения качества;
4) процесс верификации;
5) процесс аттестации;
6) процесс аудита;
7) процесс совместной оценки;
8) процесс решения проблем.

III. **Организационные процессы** – определяют действия и задачи, выполняемые как заказчиком, так и разработчиком проекта для управления своими процессами:

1) процесс управления;
2) процесс создания инфраструктуры проекта;
3) процесс усовершенствования;
4) процесс обучения.

Характеристики стандарта ISO/IEC 12207:

- *динамичность*: обеспечивается способом определения последовательности выполнения процессов, при котором один процесс при необходимости вызывает другой или его часть. Это позволяет реализовать любую модель ЖЦ;
- *адаптивность*: стандарт ISO 12207 предусматривает исключение процессов, видов деятельности и задач, неприменимых в конкретном проекте.

Ниже приведены ориентировочные описания основных процессов ЖЦ (табл. 2.1. – 2.3.).

Таблица 2.1.

Ориентировочное описание процесса *приобретения*. Исполнитель – Заказчик

Вход	**Действия**	**Выход**
1. Решение о начале работ по внедрению ИС. 2. Результаты обследования деятельности заказчика. 3. Результаты анализа рынка ИС/ тендера. 4. План поставки/ разработки. 5. Комплексный тест ИС.	1. Инициирование. 2. Подготовка заявочных предложений. 3. Подготовка договора. 4. Контроль деятельности поставщика. 5. Приемка ИС.	1. Технико-экономическое обоснование внедрения ИС. 2. Техническое задание на ИС. 3. Договор на поставку/ разработку. 4. Акты приемки этапов работы. 5. Акт приемно-сдаточных испытаний.

Таблица 2.2.

Ориентировочное описание процесса *поставки*. Исполнитель – Разработчик ИС

Вход	Действия	Выход
1. Техническое задание на ИС. 2. Решение руководства об участии в разработке. 3. Результаты тендера. 4. Техническое задание на ИС. 5. Разработанная ИС и документация.	1. Инициирование. 2. Ответ на заявочные предложения. 3. Подготовка договора. 4. Планирование исполнения. 5. Поставка ИС.	1. Решение об участии в разработке. 2. Коммерческие предложения/ конкурсная заявка. 3. Договор на поставку/ разработку. 4. План управления проектом. 5. Акт приемно-сдаточных испытаний.

Таблица 2.3.

Ориентировочное описание процесса *разработки*. Исполнитель – разработчик ИС

Вход	Действия	Выход
1. Техническое задание на ИС. 2. Техническое задание на ИС, модель ЖЦ. 3. Техническое задание на ИС. 4. Подсистемы ИС. 5. Спецификации требования к компонентам ПО. 6. Архитектура ПО. 7. Материалы детального проектирования ПО. 8. План интеграции ПО, тесты. 9. Архитектура ИС, ПО, документация на ИС, тесты.	1. Подготовка. 2. Анализ требований к ИС. 3. Проектирование архитектуры ИС. 4. Разработка требований к ПО. 5. Проектирование архитектуры ПО. 6. Детальное проектирование ПО. 7. Кодирование и тестирование ПО. 8. Интеграция ПО и квалификационное тестирование ПО. 9. Интеграция ИС и квалификационное тестирование ИС.	1. Используемая модель ЖЦ, стандарты разработки. 2. План работ. 3. Состав подсистем, компоненты оборудования. 4. Спецификации требования к компонентам ПО. 5. Состав компонентов ПО, интерфейсы с БД, план интеграции ПО. 6. Проект БД, спецификации интерфейсов между компонентами ПО, требования к тестам. 7. Тексты модулей ПО, акты автономного тестирования. 8. Оценка соответствия комплекса ПО требованиям ТЗ. 9. Оценка соответствия ПО, БД, технического комплекса и комплекта документации требованиям ТЗ.

Для поддержки практического применения стандарта ISO/IEC 12207 разработан ряд технологических документов:

- руководство для ISO/IEC 12207 (ISO/IEC TR 15271:1998 Information technology – Guide for ISO/IEC 12207);
- руководство по применению ISO/IEC 12207 к управлению проектами (ISO/IEC TR 16326:1999 Software engineering – Guide for the application of ISO/IEC 12207 to project management).

Позднее был разработан и в 2002 г. опубликован **стандарт на процессы ЖЦ систем ISO/IEC 15288** (System life cycle processes). При создании стандарта был учтен практический опыт создания систем в правительственных, коммерческих, военных и академических организациях. К разработке стандарта были привлечены специалисты различных областей: системной инженерии, программирования, управления качеством, человеческими ресурсами, безопасностью и пр. Стандарт применим для широкого класса систем, но его основное предназначение – поддержка создания компьютеризированных систем.

Согласно стандарту **ISO/IEC серии 15288** в структуру ЖЦ следует включать следующие группы процессов:

договорные процессы:

- приобретение (внутренние решения или решения внешнего поставщика);
- поставка (внутренние решения или решения внешнего поставщика).

процессы предприятия:

- управление окружающей средой предприятия;
- инвестиционное управление;
- управление ЖЦ ИС;
- управление ресурсами;
- управление качеством.

проектные процессы:

- планирование проекта;
- оценка проекта;
- контроль проекта;
- управление рисками;
- управление конфигурацией;
- управление информационными потоками;
- принятие решений.

технические процессы:

- определение требований;
- анализ требований;
- разработка архитектуры;
- внедрение;
- интеграция;
- верификация;
- переход;
- аттестация;
- эксплуатация;
- сопровождение;
- утилизация.

специальные процессы: определение и установка взаимосвязей исходя из задач и целей.

Стадии создания системы, предусмотренные **ISO/IEC 15288**:

1) формирование концепции – анализ потребностей, выбор концепции и проектных решений;
2) разработка – проектирование системы;
3) реализация – изготовление системы;
4) эксплуатация – ввод в эксплуатацию и использование системы;
5) поддержка – обеспечение функционирования системы;
6) снятие с эксплуатации – прекращение использования, демонтаж, архивирование системы.

ГОСТ 34.601-90 (каноническое проектирование)

Комплекс стандартов ГОСТ 34 задумывался в конце 80-х годов как всеобъемлющий комплекс взаимоувязанных межотраслевых документов. *Основной целью* комплекса было разрешение противоречий, возникающих при интеграции систем вследствие несогласованности нормативно-технической документации.

В 80-х годах действовали следующие комплексы и системы стандартов, устанавливающие требования к различным видам АС:

- единая система стандартов АСУ (24-я система) для АСУ, ОАСУ (отраслевая АСУ), АСУП, АСУТП и других организационно-экономических систем;
- комплекс стандартов системы 23501, распространявшихся на системы автоматизированного проектирования (САПР);
- четвертая группа 1-й системы стандартов, распространявшихся на автоматизированные системы технологической подготовки производства (АСТПП).

Несмотря на общие понятия, требования стандартов не были согласованы между собой, имелись различия по составу и содержанию работ, обозначениям и оформлению документов. В этих условиях было решено выработать одну обобщенную понятийную и терминологическую систему, общую схему разработки, общий набор документов и их содержания и определить их как обязательные для всех АИС.

В этом смысле комплекс стандартов ГОСТ 34 более близок к схемам конкретных методик, чем к стандартам типа ISO 12207. *Объектами стандартизации* являются автоматизированные системы различных видов и все виды их компонентов, а не только ПО и БД.

Комплекс рассчитан на взаимодействие заказчика и разработчика. При этом, аналогично ISO 12207 в нем предусмотрено, что заказчик может разрабатывать АС сам для себя, например, создав для этого специализированное подразделение. ГОСТ 34 уделяет основное внимание содержанию проектных документов, а распределение действий между сторонами обычно производятся исходя из этого содержания.

Наиболее популярными в группе **ГОСТ 34** можно считать следующие стандарты:

- **ГОСТ 34.601-90** (стадии и этапы создания автоматизированной системы);

- **ГОСТ 34.602-89** (техническое задание на создание автоматизированной системы);
- методические указания **РД 50-34.698-90** (требования к содержанию документов).

Согласно ГОСТ 34.601-90 разработка ИС разбивается на следующие **стадии:**

1) **формирование требований к ИС**:

   1.1) обследование объекта и обоснование необходимости создания ИС;

   1.2) формирование требований пользователя к ИС;

   1.3) оформление отчёта о выполненной работе и заявки на разработку ИС (тактико-технического задания);

2) **разработка концепции ИС**:

   2.1) изучение объекта;

   2.2) проведение необходимых научно-исследовательских работ;

   2.3) разработка вариантов концепции ИС, удовлетворяющего требованиям пользователя;

   2.4) оформление отчёта о выполненной работе;

3) **техническое задание –** разработка и утверждение технического задания на создание ИС;

4) **эскизный проект**:

   4.1) разработка предварительных проектных решений по системе и её частям, а именно:

   - по функционально-алгоритмической структуре системы;
   - по функциям персонала и организационной структуре;
   - по структуре технических средств;
   - по алгоритмам решения задач и применяемым языкам;
   - по организации и ведению информационной базы;
   - по системе классификации и кодирования информации;
   - по программному обеспечению;

4.2) разработка документации на ИС и её части.

5) **технический проект**:

5.1) разработка проектных решений по системе и её частям;

5.2) разработка документации на ИС и её части;

5.3) разработка и оформление документации на поставку изделий для комплектования ИС и (или) технических требований (технических заданий) на их разработку;

5.4) разработка заданий на проектирование в смежных частях проекта объекта автоматизации.

6) **рабочая документация**:

6.1) разработка рабочей документации на систему и её части;

6.2) разработка или адаптация программ.

7) **ввод в действие:**

7.1) подготовка объекта автоматизации к вводу ИС в действие;

7.2) подготовка персонала;

7.3) комплектация ИС поставляемыми изделиями (программными и техническими средствами, программно-техническими комплексами, информационными изделиями);

7.4) строительно-монтажные работы;

7.5) пусконаладочные работы;

7.6) проведение предварительных испытаний;

7.7) проведение опытной эксплуатации;

7.8) проведение приёмочных испытаний;

8) **сопровождение ИС**:

8.1) выполнение работ в соответствии с гарантийными обязательствами;

8.2) послегарантийное обслуживание.

Виды документов, разрабатываемых на стадиях «Эскизный проект», «Технический проект», «Рабочая документация»:

- «ведомость» – перечисление в систематизированном виде объектов, предметов и т.д.;

- «схема» – графическое изображение форм документов, частей, элементов системы и связей между ними в виде условных обозначений;
- «инструкция» – изложение состава действий и правил их выполнения персоналом;
- «обоснование» – изложение сведений, подтверждающих целесообразность принимаемых решений;
- «описание» – пояснение назначения системы, ее частей, принципов их действия и условий применения;
- «конструкторский документ» – по ГОСТ 2.102;
- «программный документ» – по ГОСТ 19.101.

В зависимости от сложности объекта автоматизации и набора задач, требующих решения при создании конкретной ИС, стадии и этапы работ могут иметь различную трудоемкость.

ГОСТ 34 допускает объединение последовательных этапов и даже исключение некоторых из них на любой стадии проекта. Допускается исключить стадию «Эскизный проект» и отдельные этапы работ на всех стадиях, объединять стадии «Технический проект» и «Рабочая документация» в одну стадию «Технорабочий проект».

В зависимости от специфики создаваемых АС и условий их создания допускается выполнять отдельные этапы работ до завершения предшествующих стадий, параллельное во времени выполнение этапов работ, включение новых этапов работ.

Стадии и этапы, выполняемые организациями-участниками, прописываются в договорах и технических заданиях на выполнение работ.

2.3. Документирование проекта

Назначение документации

Документация входит в состав проекта по созданию, внедрению, сопровождению, модернизации и ликвидации ИС на протяжении полного жизненного цикла этой ИС. Документация необходима:

- для обеспечения эффективных и экономичных процедур разработки, сопровождения и использования программных средств и всей ИС;
- для организации обмена информацией между управляющим персоналом, разработчиками, администратором, пользователями ИС, а также другими, не предусмотренными проектом лицами и группами (инспектирующими структурами и т.п.) на всех стадиях жизненного цикла (ЖЦ) ИС.

Документация выполняет следующие функции:

- дает описание возможностей системы, то есть позволяет пользователю определить соответствие программного продукта требованиям, предъявляемым к ИС в целом;
- обеспечивает фиксацию принятых и реализованных проектных решений, давая возможность для дальнейшей модификации и совершенствования программного обеспечения ИС;
- предоставляет технические материалы для анализа информационной системы на этапах её приобретения и разработки;
- предоставляет информацию о процедурах эксплуатации и технического обслуживания ИС;
- регламентирует средства и процедуры защиты информации, регулирует права и обязанности различных групп пользователей ИС, условия функционирования ИС, включая вопросы ее модернизации, масштабирования, переносимости и ликвидации.

Требования к документации

К документации предъявляют следующие требования:

1. документы должны быть ясными, краткими, точными и полными;
2. для повышения эффективности работы с документами должны использоваться стандарты, регламентирующие форму, содержание и, иногда, стиль документов;
3. документация должна создаваться параллельно с разработкой ПО;
4. обязанности по документированию системы лежат на ее разработчике, создающем, модернизирующем и привлекающем в проект ИС те или иные программные средства. Особенно важна внешняя документация;
5. документация должна высокий уровень абстракции при возможности четкого и однозначного толкования и достаточности информации об описываемых объектах;
6. перед составлением документации необходимо иметь ответ на следующие вопросы:
   - что и зачем должно быть документировано;
   - для кого предназначен тот или иной документ;
   - возможные способы решения тех или иных задач, стоящих перед пользователем;
   - какие ошибки может допустить пользователь, и что нужно сделать для их устранения;
   - как и в каких условиях будет использоваться документ;
   - сколько выделено средств, и каковы сроки разработки документа;
   - кто будет оценивать документ и как он соотносится к отраслевым или ведомственным требованиям на сертификацию разработки;
   - как будет обновляться и поддерживаться документация и каковы механизмы и сроки внесения изменений и пересмотра документа; кто ответственен за реализацию этих действий, а также за хранение, неиз-

менность и контроль за исполнением.

Ответы на эти вопросы должны быть получены на ранних стадиях разработки ИС (на стадии разработки технико-экономического обоснования к ТЗ) и входить в состав разрабатываемой в рамках проекта документации.

Для повышения эффективности разработки программных изделий (ПИ), а также повышения их качества необходима стандартизация и унификация документов, описывающих как процедуры работ, так и результаты выполнения работ по созданию программного продукта. С этой целью было разработано несколько десятков отечественных государственных стандартов, из которых больше половины были стандартами в рамках Совета экономической взаимопомощи (СЭВ) бывшего содружества стран социалистического лагеря (СССР и др.), остальные – международными (ISO).

Состав программных документов по фазам ЖЦ ИС

Системная документация (классификация документов, существовавшая с 1970 по 1980 годы и не отмененная в настоящее время) по стадиям ЖЦ ИС:

постановка задачи:

1) техническое задание (ТЗ), включает в свой состав: технико-экономическое описание проекта (ТЭО); календарный план проектирования; сметную калькуляцию проекта; каталожное описание разработки (КО), технические требования (ТТ) и технические условия (ТУ), относящиеся к проектируемому изделию и т.п.;

разработка:

2) проектная документация, в составе: проект системы; подготовка данных; разработка программы;

реализация испытаний:

3) пособия руководства: руководство пользователя; руководство по

обслуживанию; руководство оператора; руководство администраторов (данных, баз данных, серверного обеспечения, сетевого обеспечения, сервера защиты и т.п.)

эксплуатация:

4) реализация: программный код; информация, вызываемая системой; тесты и тестовые прогоны программы; требования, процедуры и условия сертификации продукта.

Альтернативный состав документации, предусмотренный действующими стандартами (по стадиям ЖЦ ИС):

выработка требований:

1) требования к функциональной структуре;
2) требования к информационной структуре;

проектирование:

3) системная спецификация и описание подсистем;
4) программная спецификация;
5) спецификация БД;
6) руководство системных специалистов, администраторов;
7) руководство пользователя, план испытаний;

программирование, испытание, сертификация:

8) руководство по эксплуатации;
9) руководство по сопровождению;

Единая система программной документации (ЕСПД)

ЕСПД – комплекс государственных стандартов Российской Федерации, устанавливающих взаимосвязанные правила разработки, оформления и обращения программ и программной документации. В стандартах ЕСПД устанавливают требования, регламентирующие разработку, сопровождение, изготовление и эксплуатацию программ. Сопровождение программы включает анализ функционирования, развитие и совершенствование программы, а также внесение

изменений в нее с целью устранения ошибок. Различают следующие классификационные группы стандартов ЕСПД:

1) общие положения;
2) основополагающие стандарты;
3) правила выполнения документации при разработке;
4) правила выполнения документации при изготовлении;
5) правила выполнения документации при сопровождении;
6) правила выполнения документации при эксплуатации;
7) правила обращения к программной документации;
8) резервные группы;
9) прочие стандарты.

Подразумевается, что за счет единых средств формирования документов, унификации их структуры, последовательности выполнения операций, использование указанных групп стандартов обеспечивает взаимный обмен программами, применение ранее разработанных программ в новых разработках и снижение затрат на разработку, оформление и использование программ.

Согласно ЕСПД предусмотрен следующий перечень обязательных документов, входящих в состав ИС: *спецификация*; *ведомость держателей подлинников*; *текст программы* – сведения о логической структуре и функции программ. Программа и методика испытаний в составе этого пакета документов отображает требования, подлежащие проверке, а также методы контроля.

В *техническом задании (ТЗ)* обосновываются назначение и области применения программы, технические, технико-экономические и специальные требования, необходимые стадии и сроки разработки, виды испытаний. Каждая созданная или привлеченная в проект программа сопровождается пояснительной запиской, в которой наряду с ее обобщенным описанием приводятся схема алгоритма и общее описание алгоритма и функция программы, а также обоснование принятых решений. Эта записка входит в состав расчетно-пояснительной записки к проекту ИС. *Эксплуатационные документы* содержат сведения необходимые для обеспечения функционирования и эксплуатации системы и включают в состав:

ведомость эксплуатационных документов на программу;

- формуляр (основные характеристики, комплектность, сведения об эксплуатации);
- описание применения (сведения о назначении, класс задач, область применения, используемые методы, организация, минимальная конфигурация технических средств, в том числе по вопросам организации АРМ пользователей, серверного хозяйства, маршрутизации, сетевого обеспечения, организации бесперебойного энергопитания, средств защиты, периферийных устройств и ТСО);
- руководство системного программиста (сведения для проверки, обеспечения функционирования и настройки программы);
- руководство программиста (сведения для эксплуатации программ);
- руководство оператора (сведения для осуществления действий по выполнению программой / системой требований);
- руководство по техническому обеспечению;
- журнал документов;
- руководство (инструкции) по сертификации, модернизациям, масштабированию и ликвидации ИС (АСУ) по истечении действия предусмотренного проектом полного жизненного цикла системы;
- обучающие и учебно-методические материалы по системе, ознакомительные с ней материалы (демоверсии и их описание);
- другие эксплуатационные документы (специального назначения), оговоренные в ТЗ.

Стандарты в составе ЕСПД:

- ГОСТ 19.001-77. ЕСПД. Общие положения;
- ГОСТ 19.003-80. ЕСПД. Схемы алгоритмов и программ. Обозначения условные графические. Заменен на ГОСТ 19.701-90;
- ГОСТ 19.005-85. ЕСПД. Р-схемы алгоритмов и программ. Обозначения условные графические и правила выполнения;

- ГОСТ 19.101-77. ЕСПД. Виды программ и программных документов;
- ГОСТ 19.102-77. ЕСПД. Стадии разработки;
- ГОСТ 19.103-77. ЕСПД. Обозначение программ и программных документов;
- ГОСТ 19.104-78. ЕСПД. Основные надписи;
- ГОСТ 19.105-78. ЕСПД. Общие требования к программным документам;
- ГОСТ 19.106-78. ЕСПД. Требования к программным документам, выполненным печатным способом;
- ГОСТ 19.201-78. ЕСПД. Техническое задание. Требования к содержанию и оформлению;
- ГОСТ 19.202-78. ЕСПД. Спецификация. Требования к содержанию и оформлению;
- ГОСТ 19.301-79. ЕСПД. Программа и методика испытаний. Требования к содержанию и оформлению;
- ГОСТ 19.401-78. ЕСПД. Текст программы. Требования к содержанию и оформлению;
- ГОСТ 19.402-78. ЕСПД. Описание программы;
- ГОСТ 19.403-79. ЕСПД. Ведомость держателей подлинников;
- ГОСТ 19.404-79. ЕСПД. Пояснительная записка. Требования к содержанию и оформлению;
- ГОСТ 19.501-78. ЕСПД. Формуляр. Требования к содержанию и оформлению;
- ГОСТ 19.502-78. ЕСПД. Описание применения. Требования к содержанию и оформлению;
- ГОСТ 19.503-79. ЕСПД. Руководство системного программиста. Требования к содержанию и оформлению;
- ГОСТ 19.504-79. ЕСПД. Руководство программиста. Требования к содержанию и оформлению;
- ГОСТ 19.505-79. ЕСПД. Руководство оператора. Требования к содержанию и оформлению;

- ГОСТ 19.506-79. ЕСПД. Описание языка. Требования к содержанию и оформлению;
- ГОСТ 19.507-79. ЕСПД. Ведомость эксплуатационных документов;
- ГОСТ 19.508-79. ЕСПД. Руководство по техническому обслуживанию. Требования к содержанию и оформлению;
- ГОСТ 19.601-78. ЕСПД. Общие правила дублирования, учета и хранения;
- ГОСТ 19.602-78. ЕСПД. Правила дублирования, учета и хранения программных документов, выполненных печатным способом;
- ГОСТ 19.603-78. ЕСПД. Общие правила внесения изменений;
- ГОСТ 19.604-78. ЕСПД. Правила внесения изменений в программные документы, выполненные печатным способом;
- ГОСТ 19.701-90 (ИСО 5807-85). ЕСПД. Схемы алгоритмов, программ, данных и систем. Условные обозначения и правила выполнения.

Техническое задание на создание ИС

Определение и назначение технического задания

На стадии «Техническое задание» разрабатывают Техническое задание (ТЗ) на создание автоматизированной системы в соответствии с требованиями ГОСТ 34.602-89.

Техниическое задание (ТЗ, техзадание) – исходный документ для проектирования сооружения или промышленного комплекса, конструирования технического устройства (прибора, машины, **системы управления** и т. д.), разработки информационных систем, стандартов либо проведения научно-исследовательских работ (НИР). ТЗ содержит основные технические требования, предъявляемые к сооружению или изделию и исходные данные для разработки.

ТЗ на ИС является основным документом, определяющим требования и

порядок создания (развития или модернизации – далее создания) ИС, в соответствии с которым проводится разработка ИС и ее приемка при вводе в действие.

ТЗ на ИС разрабатывают на систему в целом, предназначенную для работы самостоятельно или в составе другой системы. Дополнительно могут быть разработаны ТЗ на части ИС:

- на подсистемы ИС, комплексы задач ИС и т. п. в соответствии с требованиями настоящего стандарта;
- на комплектующие средства технического обеспечения и программно-технические комплексы в соответствии со стандартами ЕСКД и СРПП;
- на программные средства в соответствии со стандартами ЕСПД;
- на информационные изделия в соответствии с ГОСТ 19.201 и научно-технической документацией, действующей в ведомстве заказчика ИС.

В ТЗ указываются назначение объекта, область его применения, стадии разработки конструкторской (проектной, технологической, программной и т.п.) документации, её состав, сроки исполнения и т. д., а также особые требования, обусловленные спецификой самого объекта либо условиями его эксплуатации. Как правило, ТЗ составляют на основе анализа результатов предварительных исследований, расчётов и моделирования.

Как инструмент коммуникации в связке общения заказчик-исполнитель, техническое задание позволяет *обеим сторонам*:

- представить готовый продукт;
- выполнить попунктную проверку готового продукта (приёмочное тестирование – проведение *испытаний*);
- уменьшить число ошибок, связанных с изменением требований в результате их неполноты или ошибочности (на всех стадиях и этапах создания, за исключением *испытаний*);

заказчику:

- осознать, что именно ему нужно;
- требовать от исполнителя соответствия продукта всем условиям, оговорённым в ТЗ;

исполнителю:

- понять суть задачи, показать заказчику «технический облик» будущего изделия, программного изделия или автоматизированной системы;
- спланировать выполнение проекта и работать по намеченному плану;
- отказаться от выполнения работ, не указанных в ТЗ.

Разделы технического задания

ТЗ на ИС содержит нижеследующие разделы, которые могут быть разделены на подразделы.

1. Общие сведения.
   1.1. Полное наименование системы и её условные обозначения.
   1.2. Шифр темы или номер договора.
   1.3. Наименование организации разработчика и заказчика системы.
   1.4. Перечень документов, на основании которых создается система.
   1.5. Плановые сроки начала и окончания работы по созданию системы.
   Начало: __.__.____
   Окончание: __.__.____
   1.6. Порядок оформления и предъявления заказчику результатов.
2. Назначение и цели создания (развития) системы. Раздел «Назначение и цели создания (развития) системы» состоит из подразделов.
   2.1. Назначение системы – указывают вид автоматизируемой деятельности (управление, проектирование и т. п.) и перечень объектов автоматизации (объектов), на которых предполагается ее использовать.
   2.2. Цели создания системы – приводят наименования и требуемые значения технических, технологических, производственно-экономических или других показателей объекта автоматизации, которые должны быть достигнуты в результате создания АС, и указывают критерии оценки достижения целей создания системы. *Цель в технике* предусматривает положительную динамику, изменение текущего состояния

чего-либо в сторону улучшения, удовлетворения определенных потребностей или требований. Измеримость цели предполагает, что по описанию цели можно легко определить, насколько ее достижение улучшит текущее состояние. *Цель в технике* часто ошибочно идентифицируют с задачей. Например, «цель – строительство нового многоэтажного жилого дома». На самом деле, «строительство многоэтажного жилого дома» – задача, цель – прибыль, а миссия – «повышение благосостояния граждан». *Пример целей и задач создания автоматизированной системы учета*:

«Целями создания автоматизированной системы учета являются:

1) повышение точности учета…;
2) снижение затрат, связанных с…;
3) повышение эффективности…;

Задачи создания автоматизированной системы учета:

1) замена устаревших приборов учета на приборы, отвечающие современным требованиям;
2) автоматизация процесса измерения учитываемых физических величин;
3) автоматизация процесса консолидации данных об измеренных величинах».

3. Характеристика объектов автоматизации.
4. Требования к системе.

4.1. Требования к системе в целом.

4.1.1. Требования к структуре и функционированию системы – перечень подсистем, их назначение и основные характеристики, требования к числу уровней иерархии и степени централизации системы.

4.1.2. Требования к численности и квалификации персонала системы и режиму его работы:

- требования к численности персонала (пользователей) АС;

- требования к квалификации персонала, порядку его подготовки и контроля знаний и навыков;
- требуемый режим работы персонала АС.

4.1.3. Показатели назначения:

- степень приспособляемости системы к изменению процессов и методов управления, к отклонениям параметров объекта управления;
- допустимые пределы модернизации и развития системы;
- вероятностно-временные характеристики, при которых сохраняется целевое назначение системы.

4.1.4. Требования к надежности – требования к надежности технических средств и программного обеспечения.

4.1.5. Требования по безопасности – требования по обеспечению безопасности при эксплуатации (защита от воздействий электромагнитных полей, акустических шумов и т. п.), по допустимым уровням освещенности, вибрационных и шумовых нагрузок.

4.1.6. Требования к эргономике и технической эстетике – показатели АС, задающие необходимое качество взаимодействия человека с машиной и комфортность условий работы персонала.

4.1.7. Требования к эксплуатации, техническому обслуживанию, ремонту и хранению компонентов системы:

- виды и периодичность обслуживания ТС системы или допустимость работы без обслуживания;
- предварительные требования к допустимым площадям для размещения персонала и ТС системы, к параметрам сетей энергоснабжения и т.п.;
- требования по количеству, квалификации обслуживающего персонала и режимам его работы;
- требования к составу, размещению и условиям хранения комплекта запасных изделий и приборов;

4.1.8. Требования к защите информации от несанкционированного доступа.

4.1.9. Требования по сохранности информации при авариях – приводят перечень событий: аварий, отказов технических средств (в том числе – потеря питания) и т. п., при которых должна быть обеспечена сохранность информации в системе.

4.1.10. Дополнительные требования.

4.2. Требования к функциям (задачам), выполняемым системой.

4.2.1. Перечень функций и задач по каждой подсистеме.

4.2.2. Требования к форме представления выходной информации.

4.2.3. Перечень и критерии отказов для каждой функции, по которой задаются требования по надежности.

4.3. Требования к видам обеспечения.

5. Состав и содержание работ по созданию системы.
6. Порядок контроля и приемки системы.
7. Требования к составу и содержанию работ по подготовке объекта автоматизации к вводу системы в действие.
8. Требования к документированию.
9. Источники разработки.

В подразделе 4.3. «**Требования к видам обеспечения**» в зависимости от вида системы приводят требования к математическому, информационному, лингвистическому, программному, техническому, метрологическому, организационному, методическому и другие видам обеспечения системы.

Для **программного обеспечения** системы приводят перечень покупных программных средств и требования, предъявляемые к ним.

Описание **информационного обеспечения** содержит следующие разделы:

- принципы организации ИО;

- организация сбора и передачи информации;
- построение системы классификации и кодирования;
- организация внутримашинной информационной базы;
- организация внемашинной информационной базы.

Для **организационного обеспечения** приводят требования:

- к структуре и функциям подразделений, участвующих в функционировании системы или обеспечивающих эксплуатацию;
- к организации функционирования системы и порядку взаимодействия персонала АС и персонала объекта автоматизации;
- к защите от ошибочных действий персонала системы.

Для **математического обеспечения** системы приводят требования к составу, области применения (ограничения) и способам использования в системе математических методов и моделей, типовых алгоритмов и алгоритмов, подлежащих разработке.

В ТЗ на АИС могут включаться **приложения**. В зависимости от вида, назначения, специфических особенностей объекта автоматизации и условий функционирования системы допускается оформлять разделы ТЗ в виде приложений, вводить дополнительные, исключать или объединять подразделы ТЗ.

2.4. Технологии поддержки ЖЦ ИС

Определение и требования к технологии поддержки ЖЦ ИС

Технология проектирования определяется как совокупность трех составляющих:

1) пошаговой процедуры, определяющей последовательность технологических операций проектирования (рис.);
2) критериев и правил, используемых для оценки результатов выполнения технологических операций;
3) нотаций (графических и текстовых средств), используемых для описания проектируемой системы.

Технологические инструкции, составляющие основное содержание технологии, должны состоять из описания последовательности технологических операций, условий, в зависимости от которых выполняется та или иная операция, и описаний самих операций.

Технология проектирования, разработки и сопровождения ИС должна удовлетворять следующим общим требованиям:

- поддерживать полный ЖЦ ПО;
- обеспечивать гарантированное достижение целей разработки ИС с заданным качеством и в установленное время;
- обеспечивать возможность выполнения крупных проектов в виде подсистем (т.е. возможность декомпозиции проекта на составные части, разрабатываемые группами исполнителей ограниченной численности с последующей интеграцией составных частей). Опыт разработки крупных ИС показывает, что для повышения эффективности работ необходимо разбить проект на отдельные слабо связанные по данным и функциям подсистемы. Реализация подсистем должна выполняться отдельными группами специалистов. При этом необходимо обеспечить координацию ведения общего проекта и исключить дублирование результатов работ каждой

проектной группы, которое может возникнуть в силу наличия общих данных и функций;

- обеспечивать возможность ведения работ по проектированию отдельных подсистем небольшими группами (3-7 человек). Это обусловлено принципами управляемости коллектива и повышения производительности за счет минимизации числа внешних связей;
- обеспечивать минимальное время получения работоспособной ИС. Речь идет не о сроках готовности всей ИС, а о сроках реализации отдельных подсистем. Реализация ИС в целом в короткие сроки может потребовать привлечения большого числа разработчиков, при этом эффект может оказаться ниже, чем при реализации в более короткие сроки отдельных подсистем меньшим числом разработчиков. Практика показывает, что даже при наличии полностью завершенного проекта, внедрение идет последовательно по отдельным подсистемам;
- предусматривать возможность управления конфигурацией проекта, ведения версий проекта и его составляющих, возможность автоматического выпуска проектной документации и синхронизацию ее версий с версиями проекта;
- обеспечивать независимость выполняемых проектных решений от средств реализации ИС (систем управления базами данных (СУБД), операционных систем, языков и систем программирования);
- иметь поддержку со стороны согласованных CASE-средств, обеспечивающих автоматизацию процессов, выполняемых на всех стадиях ЖЦ.

Понятие CASE

Первоначально (1980-е) значение термина CASE (Computer Aided Software/System Engineering) ограничивалось вопросами автоматизации черчения диаграмм.

Современные CASE-средства охватывают обширную область поддержки различных технологий проектирования, покрывающих весь жизненный цикл ПО.

В настоящее время к CASE-средствам относят любое программное средство, автоматизирующее ту или иную совокупность процессов ЖЦ ИС, включая анализ и формулировку требований, проектирование прикладного ПО и баз данных, генерацию кода, тестирование, документирование, обеспечение качества, конфигурационное управление, управление проектом и др. и обладающее следующими основными характерными особенностями:

- графические средства для описания и документирования ИС;
- удобный интерфейс с разработчиками;
- интеграция отдельных компонент, поддерживающих ЖЦ ПО;
- использование специальным образом организованного хранилища проектных метаданных (*репозитория*).

При этом большую роль играют *методы визуального представления информации*. Это предполагает построение *структурных* или иных диаграмм в реальном масштабе времени, использование многообразной цветовой палитры, сквозную проверку синтаксических правил.

Классификация CASE-средств

Современный рынок программных средств насчитывает около 300 различных CASE-средств, включая как относительно дешевые и ограниченные по возможностям системы для ПК, так и дорогостоящие системы для неоднородных вычислительных платформ и операционных сред.

Все современные CASE-средства могут быть классифицированы в основном по *типам и категориям.*

По функциональным возможностям, отражающим функциональную ориентацию CASE-средств на те или иные процессы ЖЦ и компонентный состав, выделяют следующие типы CASE-средств:

- **средства анализа** (Upper CASE). Предназначенны для построения и анализа моделей предметной области (Design/IDEF (Meta Software), Bpwin (Logic Works));
- **средства анализа и проектирования** (Middle CASE). Поддерживают наиболее распространенные методологии проектирования и использующиеся для создания проектных спецификаций (Vantage Team Builder (Cayenne), Designer/2000 (ORACLE), Silverrun (CSA), PRO-IV (McDonnell Douglas), CASE.Аналитик (МакроПроджект));

  Выходом таких средств являются спецификации компонентов и интерфейсов системы, архитектуры системы, алгоритмов и структур данных;
- **средства проектирования баз данных** – обеспечивают моделирование данных и генерацию схем баз данных (как правило, на языке SQL) для наиболее распространенных СУБД. К ним относятся Erwin (Logic Works), S-Designor (SDP) и DataBase Designer (ORACLE). Средства проектирования баз данных имеются также в составе CASE-средств Vantage Team Builder, Designer/2000, Silverrun и PRO-IV;
- **средства (быстрой) разработки приложений.** На начальном этапе существования ИС их разработка велась на традиционных языках программирования. По мере возрастания сложности разрабатываемых систем появлялась потребность в новых средствах, обеспечивающих сокращение сроков разработки. Это послужило предпосылкой к созданию целого направления в области ПО – инструментальных средств для быстрой разрабоки приложений (RAD – Rapid Application Development). RAD получила широкое распространение и одобрение в конце XX века. Концепцию RAD также часто связывают с концепцией визуального программирования. Примеры сред разработки, частично использующи принципы RAD: C++ Builder и Delphi (Borland). Развитие этого направления привело к появлению на рынке ПО средств автоматизации практически всех этапов ЖЦ ИС и новым подходам к созданию и проектированию ИС. К CASE средствам разработки приложений относятся также: 4GL (Uniface

(Compuware), JAM (JYACC), PowerBuilder (Sybase), Developer/2000 (ORACLE), New Era (Informix), SQL Windows (Gupta) и др.) и генераторы кодов, входящие в состав Vantage Team Builder, PRO-IV и частично – в Silverrun;

- **средства реинжиниринга** – обеспечивают анализ программных кодов и схем баз данных и формирование на их основе различных моделей и проектных спецификаций. Средства анализа схем БД и формирования ERD входят в состав Vantage Team Builder, PRO-IV, Silverrun, Designer/2000, Erwin и S-Designor;
- в области анализа программных кодов наибольшее распространение получают объектно-ориентированные CASE-средства, обеспечивающие реинжиниринг программ на языке C++ (Rational Rose (Rational Software), Object Team (Cayenne));
- **вспомогательные типы** включают:
    - средства планирования и управления проектом (SE Companion, Microsoft Project и др.);
    - средства конфигурационного управления (PVCS (Intersolv));
    - средства тестирования (Quality Works (Segue Software));
    - средства документирования (SoDA (Rational Software)).

По степени интегрированности выполняемых функций выделяют следующие типы CASE-средств:

- отдельные локальные средства, решающие небольшие автономные задачи (tools);
- набор частично интегрированных средств, охватывающих большинство этапов жизненного цикла ИС (toolkit);
- полностью интегрированные средства, поддерживающие весь ЖЦ ИС и связанные общим репозиторием.

Интегрированное CASE-средство (или комплекс средств, поддерживающих полный ЖЦ ПО) содержит следующие компоненты:

- репозиторий, являющийся основой CASE-средства. Он должен обеспечивать хранение версий проекта и его отдельных компонентов, синхронизацию поступления информации от различных разработчиков при групповой разработке, контроль метаданных на полноту и непротиворечивость;
- графические средства анализа и проектирования, обеспечивающие создание и редактирование иерархически связанных диаграмм;
- средства разработки приложений, включая языки 4GL и генераторы кодов;
- средства конфигурационного управления;
- средства документирования;
- средства тестирования;
- средства управления проектом;
- средства реинжиниринга.

Другие способы классификации CASE-средств:

- по применяемым методологиям и моделям систем и БД;
- по степени интегрированности с СУБД;
- по доступным платформам.

На сегодняшний день Российский рынок программного обеспечения располагает следующими наиболее развитыми CASE-средствами: Vantage Team Builder (Westmount I-CASE); Designer/2000; Silverrun; Erwin; Bpwin; S-Designor; CASE.Аналитик. Кроме того, на рынке постоянно появляются как новые для отечественных пользователей системы (например, CASE /4/0, PRO-IV, System Architect, Visible Analyst Workbench, EasyCASE), так и новые версии и модификации перечисленных систем.

Оценка и выбор CASE-средств

Входной информацией для процесса оценки является:

- определение пользовательских потребностей;

- цели и ограничения проекта;
- данные о доступных CASE-средствах;
- список критериев, используемых в процессе оценки.

Элементы процесса оценки включают:

- формулировку цели, предположений и ограничений, которые могут уточняться в ходе процесса;
- определение потребности пользователей, отражающие количественные и качественные требования пользователей к CASE-средствам;
- определение критериев, определяющих набор параметров, в соответствии с которыми производится оценка и принятие решения о выборе;
- формализация результатов оценок одного или более средств;
- принятие решения (либо дальнейшая оценка).

Определение списка критериев основано на пользовательских требованиях и включает:

- выбор критериев оценки;
- определение области использования каждого критерия;
- определение одной или более метрик для каждого критерия;
- назначение веса каждому критерию.

Современные методологии и реализующие их технологии поставляются в электронном виде вместе с CASE-средствами и включают библиотеки процессов, шаблонов, методов, моделей и других компонент, предназначенных для построения ПО того класса систем, на который ориентирована методология.

Электронные методологии включают также средства, которые должны обеспечивать их адаптацию для конкретных пользователей и возможность развития по результатам выполнения конкретных проектов. Процесс адаптации методологии заключается в следующем:

- удаление ненужных процессов и других компонентов методологии;
- изменение неподходящих или добавление собственных процессов, также методов, моделей, стандартов и руководств.

Настройка методологии может осуществляться по следующим аспектам:

- этапы и операции ЖЦ;
- участники проекта;
- используемые модели ЖЦ;
- поддерживаемые концепции и др.

Электронные методологии, технологии и поддерживающие их CASE-средства составляют комплекс согласованных инструментальных средств разработки ИС.

2.5. Рекомендации по управлению программным проектом

Особенности программных проектов

Программные проекты относятся к проектам, связанным с созданием программных средств, услуг или выдачей соответствующих результатов.

Основные особенности программных проектов заключаются в том, что программные средства: являются наиболее сложными; являются элементами общей системы, увеличивающими ее сложность и создающими предпосылки для последующих ее изменений; наиболее доступны для пользователей и поэтому являются основным объектом их претензий.

Программные средства по своей природе отличаются от непрограммных продуктов, услуг и результатов, поэтому управление программными проектами имеет характерные особенности.

Процесс управления программным проектом

Процесс управления программным проектом зависит от многих факторов, например персонала, организационных и договорных требований и сложности

проекта. Администраторы программного проекта определяют методологию и методы (технологию) реализации проекта, необходимые для выполнения задач прогнозирования и соответствующего предотвращения или минимизации рисков, решения возникающих проблем.

Процесс управления (по ГОСТ Р ИСО/МЭК 12207) состоит из общих работ и задач, которые могут быть использованы любой стороной, управляющей соответствующим процессом (ами). Администратор отвечает за управление продуктом, проектом, работами и задачами соответствующего процесса (ов), таких как заказ, поставка, разработка, эксплуатация, сопровождение или вспомогательные процессы.

Ниже приведено описание процесса управления программным проектом по ГОСТ Р ИСО/МЭК 12207.

1. Подготовка и определение области управления. Данная работа состоит из следующих задач:

1) установление требований к реализуемому процессу;
2) определение возможностей реализации процесса: проверка наличия, соответствия и применимости ресурсов, выделенных для выполнения и управления процессом (персонала, материалов, технологии и условий), а также реальности сроков реализации процесса;
3) при необходимости и по согласованию со всеми заинтересованными сторонами изменение требований к процессу с точки зрения удовлетворения критериям завершения процесса.

2. Планирование. Данная работа состоит в подготовке планы для выполнения процесса. Планы должны содержать описания соответствующих работ и задач, обозначения создаваемых программных продуктов; охватывать следующие вопросы:

- установление графиков своевременного решения задач;
- оценка необходимых трудозатрат;
- определение ресурсов, необходимых для выполнения задач;
- распределение задач по исполнителям;
- определение обязанностей исполнителей;
- определение критических ситуаций, связанных с задачами или самим процессом;
- установление используемых в процессе критериев управления качеством;
- определение затрат, связанных с реализацией процесса;
- обеспечение условий и определение инфраструктуры выполнения процесса.

Планы должны также устанавливать модель ЖЦ программного средства, задачи, распределение задач и соответствующие ресурсы.

Оценки проекта, используемые при планировании, должны охватывать:

- стоимость реализации соответствующих процессов;
- инфраструктуру;
- потребности в ресурсах, включая соответствующее управление и контроль; оценку и контроль качества;
- управление риском;
- задания, выполняемые в каждом процессе и (или) работе.

Администраторы каждого программного проекта должны стремиться по возможности использовать существующую организационную инфраструктуру. Если существующая инфраструктура не удовлетворяет потребностям проекта, тогда она должна быть соответствующим образом адаптирована или дополнена.

Архивные данные следует постоянно использовать для усовершенствования процессов жизненного цикла этой организации.

3. Выполнение и контроль. Данная работа состоит из следующих задач:

- начать реализацию плана, чтобы удовлетворить поставленным целям и критериям проекта, выполняя управление процессом;

- осуществлять текущий надзор за выполнением процесса, подготавливая как внутренние отчеты о развитии процесса, так и внешние отчеты для заказчика в соответствии с условиями договора;
- исследовать, анализировать и решать проблемы, обнаруженные при выполнении процесса; Все обнаруженные проблемы и их решения должны быть документально оформлены;
- в установленные сроки отчитаться о реализации процесса.

Измерения программного средства могут быть использованы для проверки соответствия между ожидавшимися функциями программного продукта и его функциями при эксплуатации.

4. Проверка и оценка. Данная работа состоит из следующих задач:

- обеспечить оценку программных продуктов и планов на соответствие установленным требованиям;
- проверить результаты оценок программных продуктов, работ и задач, реализуемых в ходе процесса, на соответствие поставленным целям и на выполнение утвержденных планов.

Вспомогательные процессы по ГОСТ Р ИСО/МЭК 12207 (например, процессы совместного анализа и аудита) дополняют работу по проверке и оценке.

5. Завершение. Данная работа состоит из следующих задач:

1) определить степень соответствия конечных результатов критериям, установленным в договоре или организационной процедуре;
2) проконтролировать результаты и полноту документации созданных программных продуктов и выполненных работ.

Все представленные окончательные результаты и соответствующая документация должны быть сохранены в архиве в соответствии с условиями договора.

Стандарты в области управления программным проектом:

- Руководство PMBOK™ содержит рекомендации по управлению проектами в целом;
- ГОСТ Р ИСО/МЭК 12207 содержит информацию о программных проектах, вспомогательных процессах (5.2 ГОСТ Р ИСО/МЭК 12207) и описание большинства подлежащих реализации работ (видов деятельности) и задач (заданий).
- ИСО 10006 содержит информацию, относящуюся к повышению качества управления проектом.

3. ПРЕДПРОЕКТНОЕ ОБСЛЕДОВАНИЕ ОБЪЕКТА

3.1. Задачи и этапы предпроектного обследования

Обследование предприятия является важным и определяющим этапом проектирования ИС и при правильном подходе позволяет сократить эксплуатационные расходы и время на исправление ошибок, обнаруживаемых после сдачи системы.

Предпроектное обследование обычно состоит из трех этапов:

1) *предварительное обследование (сбор сведений об объекте);*
2) *анализ сведений (описание и моделирование предметной области);*
3) *оценка эффективности и целесообразности ИТ-проекта.*

На каждом этапе применяются различные *методы исследования систем управления (ИСУ)*.

Цели ИСУ: улучшение, изменение (реинжиниринг) или автоматизация систем управления.

Предмет исследования, то есть то, что изучает исследователь с целью решения проблем на объекте, это: система знаний, умений и навыков; методы и способы; факторы внешней и внутренней среды; процессы, происходящие в организации.

Объектом исследования является организация и ее подсистемы – реальные физические объекты, измеряемые качественными и количественными показателями. Научное исследование, как правило, проводится в предметных рамках определенного научного подхода с использованием группы научных методов. Цели исследования системы управления: улучшение, изменение (реинжиниринг) или автоматизация систем управления.

Метод исследования – это способ получения нового знания, а также инструментарий, с помощью которого проводится исследование; совокупность приемов обработки информации, позволяющих достичь целей исследования.

Выбор методов исследования, интеграция различных методов при проведении предпроектного обследования определяется знанием, опытом и интуицией специалистов, проводящих исследование.

Методы исследования предметной области проектирования ИС можно разбить на следующие группы:

- основанные на использовании знаний и интуиции специалистов (*экспертные методы*);
- формализованного представления систем, основанные на использовании математических, экономико-математических методов и методов моделирования, среди которых при проектировании информационных систем особое место занимают *графические*, включающие различные методы графического представления информации.
- комплексированные – сформировались путем интеграции экспертных и формализованных методов (комбинаторика, ситуационное моделирование, топология, графосемиотика и др).

3.2. Сбор сведений об объекте

На предварительном этапе обследования решаются следующие задачи:

- предварительное выявление требований к будущей системе;
- определение структуры организации;
- определение перечня целевых функций организации;
- анализ распределения функций по подразделениям и сотрудникам;
- выявление функциональных взаимодействий между подразделениями, информационных потоков внутри подразделений и между ними, внешних информационных воздействий;
- анализ существующих средств автоматизации организации и др.

Длительность предварительного обследования обычно составляет 1-2 недели. В течение этого времени системный аналитик должен обследовать не бо-

лее 2-3 видов деятельности (учет кадров, бухгалтерия, перевозки, маркетинг и др.).

Сбор информации для построения *полной бизнес-модели организации* часто сводится к изучению документированных информационных потоков и функций подразделений, а также производится путем интервьюирования и анкетирования.

К началу работ по обследованию организация обычно предоставляет комплект документов, в состав которого обычно входят:

1) сводная информация о деятельности предприятия (оргструктура, информация об управленческой, финансово-экономической, производственной деятельности предприятия, сведения об учетной политике и отчетности);
2) регулярный документооборот предприятия (реестры входящей, исходящей и внутренней информации (табл. 3.1.);
3) сведения об инфраструктуре предприятия;
4) сведения об ответственных лицах и исполнителях.

Таблица 3.1.

Реестры информации

Реестр входящей инфрмации						
(Наименование предприятия)		(Наименование подразделения)		Характеристики обработки документов		
№	Наименование и назначение документа	Кто обрабатывает	Откуда поступает	Трудоемкость	Периодичность, регламент	Способ получения
Реестр внутренней информации						
(Наименование предприятия)		(Наименование подразделения)		Характеристики обработки документов		
№	Наименование и назначение документа	Кто обрабатывает	Кому передает	Трудоемкость	Периодичность, регламент	Способ получения
Реестр исходящей информации						
(Наименование предприятия)		(Наименование подразделения)		Характеристики обработки документов		
№	Наименование и назначение документа	Кто обрабатывает	Куда поступает	Трудоемкость	Периодичность, регламент	Способ получения

Один из методов исследования, который может быть применен на этапе сбора сведений – *интервьюирование*.

Списки вопросов для интервьюирования и анкетирования составляются по каждому обследуемому подразделению и утверждаются руководителем компании. Это делается с целью:

- предотвращения доступа к конфиденциальной информации;
- усиления целевой направленности обследования;
- минимизации отвлечения сотрудников предприятий от выполнения должностных обязанностей.

Общий перечень вопросов (с их последующей детализацией) включает следующие пункты:

- основные задачи подразделений;
- собираемая и регистрируемая информация;
- отчетность;
- взаимодействие с другими подразделениями.

Анкеты для руководителей и специалистов могут содержать следующие вопросы:

- каковы (с позиций вашего подразделения) должны быть цели создания интегрированной системы управления предприятием;
- организационная структура подразделения;
- задачи подразделения;
- последовательность действий при выполнении задач;
- с какими типами внешних организаций (банк, заказчик, поставщик и т.п.) взаимодействует подразделение и какой информацией обменивается;
- каким справочным материалом вы пользуетесь;
- сколько времени (в минутах) вы тратите на исполнение основных операций; на какие даты приходятся «пиковые нагрузки» (периодичность в месяц, квартал, год и т.д.); техническое оснащение подразделения (компью-

теры, сеть, модем и т.п.); используемые программные продукты для автоматизации бизнес-процессов;

- какие отчеты и как часто вы готовите для руководства; ключевые специалисты подразделения, способные ответить на любые вопросы по бизнес-процессам, применяемым в подразделении;
- характеристики удаленных объектов управления;
- документооборот на рабочем месте.

Собранные таким образом данные, как правило, не охватывают всех существенных сторон организационной деятельности и обладают высокой степенью субъективности. И самое главное, что такого рода обследования не выявляют устойчивых факторов, связанных со специфическими особенностями организации, воздействовать на которые можно исключительно методами функциональной настройки организационной системы.

Анализ опросов руководителей обследуемых организаций и предприятий показывает, что их представления о структуре организации, общих и локальных целях функционирования, задачах и функциях подразделений, а также подчиненности работников иногда имеют противоречивый характер. Кроме того, эти представления подчас расходятся с официально декларируемыми целями и правилами или противоречат фактической деятельности.

Если структуру информационных потоков можно выявить по образцам документов и конфигурациям компьютерных сетей и баз данных, то структура реальных микропроцессов, осуществляемых персоналом в информационных контактах (в значительной мере недокументированных) остается неизвестной.

Ответы на эти вопросы может дать *структурно-функциональная диагностика*, основанная на методах сплошной (или выборочной) фотографии рабочего времени персонала. Цель диагностики – получение достоверного знания об организации и организационных отношениях ее функциональных элементов.

В связи с этим к важнейшим задачам функциональной диагностики организационных структур относятся:

- классификация субъектов функционирования (категорий и групп работников);
- классификация элементов процесса функционирования (действий, процедур);
- классификация направлений (решаемых проблем), целей функционирования;
- классификация элементов информационных потоков;
- проведение обследования деятельности персонала организации;
- исследование распределения (по времени и частоте) организационных характеристик: процедур, контактов персонала, направлений деятельности, элементов информационных потоков – по отдельности и в комбинациях друг с другом по категориям работников, видам процедур и их направлениям (согласно результатам и логике исследований);
- выявление реальной структуры функциональных, информационных, иерархических, временных, проблемных отношений между руководителями, сотрудниками и подразделениями;
- установление структуры распределения рабочего времени руководителей и персонала относительно функций, проблем и целей организации;
- выявление основных технологий функционирования организации (информационных процессов, включая и недокументированные), их целеполагания в сравнении с декларируемыми целями организации;
- выявление однородных по специфике деятельности, целевой ориентации и реальной подчиненности групп работников, формирование реальной модели организационной структуры и сравнение ее с декларируемой;
- определение причин рассогласования декларируемой и реальной структуры организационных отношений.

Сплошной «фотографией» рабочего времени называется непрерывное наблюдение и регистрация характеристик работников в процессе функционирования в течение всего рабочего дня. Информация последовательно вносятся в

заранее заготовленную рабочую таблицу, которая по окончании процедуры обследования пополняется дополнительными характеристиками: технологическая ветвь, системная функция, предмет, аспект, эмоциональный фон и др. Часть показателей, те, что помечены звездочкой, заполняются в процессе обследования, остальные – после. Содержание рабочей таблицы может быть следующим:

- номер (по порядку);
- агент (должность обследуемого работника);
- время, в течение которого выполнялась процедура;
- процедура (наименование содержания совокупности элементарных действий, объединенных общностью решаемой частной задачи);
- содержание (суть процедуры, которая должна быть классифицирована);
- информация (направление движения информации между агентом и контрагентом);
- инициатива (инициатор начала выполнения данной процедуры);
- контрагент (должность работника, который находится с обследуемым в контакте);
- отношение (отражающая субординацию агента и контрагента форма взаимодействия в данной процедуре);
- проблема (словесная характеристика решаемой проблемы).

Результатом предпроектного обследования должен явиться «*Отчет об экспресс-обследовании предприятия»*, который включает следующее:

- краткое схематичное описание бизнес-процессов (например: управление закупками и запасами, управление производством, управление продажами, управление финансовыми ресурсами);
- основные требования и приоритеты автоматизации;
- оценка необходимых для обеспечения проекта ресурсов заказчика;
- оценка возможности автоматизации, предложения по созданию автоматизированной системы с оценкой примерных сроков и стоимости.

Документы, входящие в отчет об обследовании, могут быть представлены в виде текстового описания или таблиц с описанием операций, исполнителей и документов бизнес процессов.

Информация, полученная в результате предпроектного обследования, анализируется с помощью *методов структурного и/или объектного анализа* и используется для построения *моделей деятельности организации*. Модель организации предполагает построение двух видов моделей:

- модели «как есть», отражающей существующее на момент обследования положение дел в организации и позволяющей понять, каким образом функционирует данная организация, а также выявить узкие места и сформулировать предложения по улучшению;
- модели «как должно быть», отражающей представление о новых технологиях работы организации.

Каждая из моделей включает в себя полную функциональную и информационную модель деятельности организации, а также модель, описывающую динамику поведения организации (в случае необходимости).

В качестве основного каркаса, объединяющего и систематизирующего все знания по бизнес-модели, можно использовать различные эталонные (референтные) модели.

3.3. Описание сведений

Процедура СА для описания предметной области

Процедура системного анализа для *описания предметной области* включает нижеследующие этапы:

1. **Определение аспекта** рассмотрения системы, т.е. определение точки зрения, с которой рассматривается объект.
2. **Определение объекта.** В качестве системы может изучаться целое предприятие или его подсистема, например, отдельное подразделение. Здесь

определяется основная деятельность, например выпуск определенных изделий и главная цель функционирования, например, получение прибыли, если объект функционирует за счет собственных средств.

3. **Выделение элементов.** Элементы должны быть выделены с точки зрения поставленной задачи на объекте. Элементами в системе могут являться подразделения и должности. Например:
   - руководитель предприятия (он же является Главным директором) – главная функция которого принимать главные управленческие решения по различным вопросам: о найме и увольнении персонала, и заключать договоры с потенциальными клиентами;
   - главный бухгалтер предприятия. Основной функцией является ведение бухгалтерского учета на предприятии, распределение средств, выплата заработной платы;
   - бухгалтер (работники бухгалтерии) – осуществляют бумажную и учетную работу по ведению бухгалтерии;
   - заведующий хозяйственной частью предприятия – распределяет материальные ресурсы по цехам и отделениям;
   - литейщики – осуществляют выплавку и создание ТМЦ;
   - слесари – осуществляют мелкий ремонт и обслуживание производственного оборудования и другой техники;
   - работники отдела компьютерного обеспечения – обеспечивают работу офисной техники на предприятии;
   - охранники – обеспечивают безопасность на предприятии;
   - инженеры – ведут расчеты, связанные с улучшением старой продукции и введения новой.
4. **Определение подсистем.** Подсистемами могут являться отделы, работающие самостоятельно, но для достижения основной деятельности – выпуска продукции:
   - руководство предприятием;
   - отдел инженеров;

- отдел сбыта и продажи продукции;
- отдел маркетинга;
- бухгалтерия;
- финансовый отдел;
- отдел компьютерного обеспечения;
- хозяйственный отдел;
- отдел кадров;
- отдел курьерской доставки.

5. **Определение внешней среды.** Определяются субъекты внешней среды, оказывающие существенное влияние на предприятие, например, конкуренты, поставщики, потенциальные потребители и др.
6. **Определение функциональной структуры предприятия** (функциональных областей и их деятельности).
7. **Выделение и описание бизнес процессов** (процессное описание предметной области).

Процессное описание предметной области

Основные понятия процесса

Ниже приведен список основных понятий процесса:

- *процесс* в терминах ISO 9000:2000 – это совокупность взаимосвязанных или взаимодействующих видов деятельности, преобразующих «входы» в «выходы».
- *владелец процесса* – должностное лицо, несущее ответственность за ход и результаты *процесса*;
- *ресурсы* – ресурсы, выделенные в распоряжение *владельца процесса* для его проведения; могут включать – оборудование (производственное, контрольно-измерительное, офисное и др.), персонал, помещения, среду,

транспорт, связь, материалы (вспомогательные), финансы, документация и т. д.;

- *параметры процесса* – характеристики (информация) по которым Владелец *процесса* и высший руководитель могут судить о том, насколько эффективно выполняется *процесс* и достигаются ли запланированные результаты;
- *поставщик* – субъект, предоставляющий ресурсы.
- *потребитель* – потребитель результатов *процесса*, степень удовлетворенности которого также предназначена для оценки эффективности Процесса;
- *входы процесса* – входные объекты (сырье, продукция, комплектация, информация или услуга), которые преобразуются в *выходы* процесса в ходе выполнения *процесса*. Часто *входы* одного *процесса* являются выходами другого;
- *выходы процесса* – продукция, информация или услуга ради которой существует *процесс*.
- *сеть процессов организации* – объединение взаимосвязанных и взаимосогласованных *процессов* организации в единую систему.

Правила процессного описания предметной области

1) все процессы делятся на *основные* и вспомогательные (см. «Классификация бизнес процессов»). Через *основные процессы* проходят производимая продукция и/или услуги и их компоненты (маркетинг, проект, входящие материалы и др.). Вспомогательные процессы предназначены для нормального функционирования основных процессов (работа офиса, хозяйственная деятельность, обучение персонала, подготовка документации);
2) основных процессов должно быть не более чем 7±2;
3) каждый Процесс должен иметь только одного владельца;

4) владельцу должны быть выделены все необходимые ресурсы и полномочия и установлены *показатели эффективности процесса*, основные из которых: *затраты на осуществление процесса*; *расчет времени на осуществление процесса*; *показатели качества процесса*;
5) приоритет в установлении требований к *выходам процесса* имеет *потребитель* результатов Процесса. «Клиент всегда прав». При этом *потребитель* может быть как внешним, так и внутренним, то есть выход одного процесса может являться входом другого в пределах одной организации;
6) владелец процесса и только он один несет ответственность:
   - за выявление и выполнение требований Потребителя, в том числе внутреннего;
   - за эффективность процесса;
   - за результат процесса;
   - за его своевременную доставку Потребителю.
7) все отклонения от нормального течения процесса должны фиксироваться и рассматриваться как база для улучшений с учетом экономической целесообразности корректировки процесса.

Классификация бизнес процессов

Реализация процессного подхода требует четкой структуризации бизнес процессов.

Первой попыткой классифицировать бизнес-процессы была разработка модели цепочки создания добавленной стоимости (Value Chain), предложенная Майклом Портером (табл.) в 1980 году. БП было предложено делить на *основные* и *вспомогательные*. Классификация М. Портера получила развитие в результате реализации норвежского проекта TOPP (The Productivity Program of the Technology Industry – программа повышения производительности промышленности) по сравнительному бенчмаркингу. Так же посредством бенчмаркинга Американский центр производительности и качества APQC (American

Productivity & Quality Center) разработал кросс-отраслевую структуру классификации процессов PCF (Process Classification Framework). В настоящее время многие предприятия используют классификацию, предложенную в стандарте ГОСТ Р ИСО 9001-2008.

Таблица 3.2.

Подходы к классификации БП

М. Портер	TOPP	APQC	ГОСТ Р ИСО 9001-2008
• основные; • вспомогательные.	• первичные; • поддерживающие; • развития.	• операционные; • управления и поддержки.	• процессы высшего руководства; • процессы менеджмента ресурсов; • процессы измерения, анализа и улучшения; • процессы жизненного цикла продукции.

По APQC делятся на 12 категорий (каждая категория разбивается на группы), из которых 5 категорий – операционные процессы, 7 – процессы управления и поддержки:

категории операционных процессов (operating processes):

- разработка стратегии (Develop Vision and Strategy);
- разработка и управление продуктами и услугами (Develop and Manage Products and Services);
- рынок и продажа продуктов и услуг (Market and Sell Products and Services);
- доставка продуктов и услуг (Deliver Products and Services);
- управление обслуживанием клиентов (Manage Customer Service);

категории процессов управления и поддержки (management and support services):

- разработка и поддержка человеческого капитала (Develop and Manage Human Capital);
- управление информационными технологиями (Manage Information Technology);

- управление финансовыми ресурсами (Manage Financial Resources);
- приобретение, строительство и управление собственностью (Acquire, Construct, and Manage Property);
- управление охраной окружающей среды (Manage Environmental Health and Safety (EHS));
- управление внешними связями (Manage External Relationships);
- управление знаниями, улучшениями и изменениями (Manage Knowledge, Improvement, and Change).

Существуют другие способы классификации БП: по степени сложности, детализации, по месту в оргструктуре, иерархии целей и т.д.

Наиболее распространена следующая классификация процессов в зависимости от их назначения: *основные*, *управления* и *обеспечения*, в соответствии с которой модель деятельности организации можно представить следующим образом (рис. 3.1.)

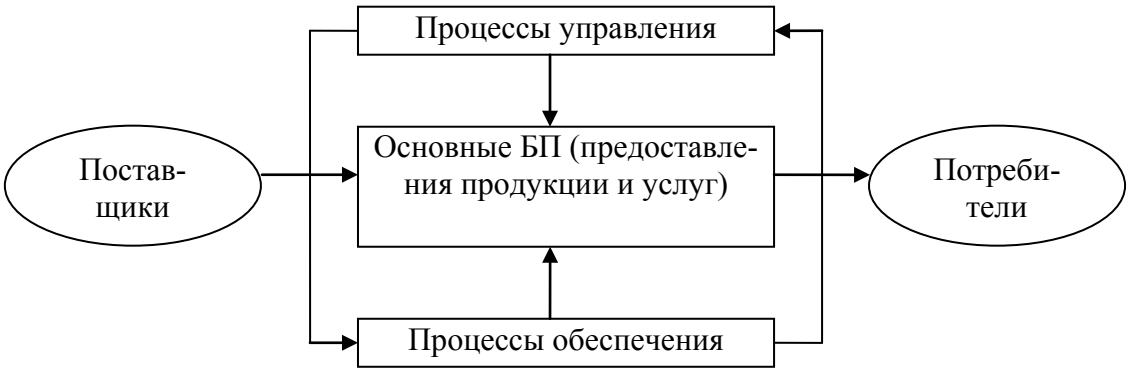

Рис. 3.1. Упрощенная модель деятельности компании

Основные процессы – это процессы ЖЦ продукции компании. Это «горизонтальные» процессы, направленные на создание и реализацию товара или услуги, представляющих ценность для клиента и обеспечивающие доход компании.

Основные процессы принято описывать по следующей производственно-коммерческой цепочке:

1) первичное взаимодействие с клиентом и определение его потребностей;
2) реализация запроса (заявки, заказа, контракта и т.п.) клиента;
3) послепродажное сопровождение;
4) мониторинг удовлетворения потребностей.

Процесс «*реализация (запроса клиента*)» может быть декомпозирован на следующие подпроцессы (процессы более низкого уровня):

- разработка (проектирование) продукции;
- закупка (товаров, материалов, комплектующих изделий), в том числе:
  - транспортировка (закупленного);
  - разгрузка, приемка на склад и хранение (закупленного);
- производство (со своим технологическим циклом и внутренней логистикой);
- приемка на склад и хранение (готовой продукции);
- отгрузка, в том числе, консервация и упаковка, погрузка, доставка);
- пуско-наладка;
- оказание услуг (предусмотренных контрактом на поставку или имеющих самостоятельное значение) и т.п.

Эти этапы цепочки также стандартны, например, в стандарте ISO редакции 1994 г. приведены многие из этих процессов в качестве обязательных и подлежащих сертификации). Проверить, какие бизнес-цепочки существуют на предприятии, можно с помощью проекции каждого из выделенных «бизнесов, продукции и услуг» на вышеуказанный (стандартный) библиотечный классификатор жизненного или уже производственного цикла.

Для оценки этапов работы с любым документом можно использовать также анализ «жизненного цикла документа», который может выглядеть следующим образом:

- предоставляет исходные данные;
- подготавливает, разрабатывает;
- заполняет;
- корректирует;

- оформляет;
- подписывает;
- контролирует соответствие установленным требованиям;
- визирует;
- согласует;
- утверждает;
- акцентирует (принимает к сведению, использует);
- хранит;
- снимает копию.

Процессы управления – это процессы планирования, организации, мотивации и контроля, направленные на выработку и реализацию управленческих решений.

Управленческие решения могут приниматься относительно организации в целом, отдельной функциональной области или отдельных бизнес процессов, например:

- стратегическое управление;
- организационное проектирование (структуризация);
- маркетинг;
- финансово-экономическое управление;
- логистика и организация процессов;
- менеджмент качества;
- управление персоналом.

При процессном описании управленческую деятельность можно «развертывать» по так называемому «управленческому циклу» (принятия решения), который включает следующие процессы:

1) сбор и анализ информации (проверка на достоверность);
2) подготовка альтернатив;
3) выбор альтернативы (принятие решения);

4) организация реализации решения;
5) контроль исполнения;
6) анализ;
7) коррекция (регулирование).

Наиболее часто встречающиеся варианты детализации для вышеперечисленных процессов:

сбор информации:

- определение состава собираемой информации;
- определение форм отчетности.

выработка решения:

- анализ альтернатив;
- подготовка вариантов решения;
- принятие решения;
- выработка критериев оценки;

организация реализации решения:

- планирование;
- организация;
- мотивация;

контроль исполнения

- учет результатов;
- сравнение по принятым критериям;

анализ:

- анализ дополнительной информации;
- диагностика возможных причин отклонений;

регулирование:

- регулирование на уровне реализации;
- регулирование на уровне выработки решения;

Каждый из этих этапов имеет своих характерных для него исполнителей – управленцев, которых можно отнести к трем основным категориям:

- руководитель (ответственный за принятие и организацию выполнения решений);
- специалист-аналитик (ответственный за подготовку решения и анализ отклонений);
- технические исполнители (сбор информации, учет, коммуникации).

Некоторые подходы выделяют типы управленческих процессов в зависимости от методов управления в организации.

В основе цикла *управления ресурсами* лежит расчет (или имитационное моделирование) и контроль результатов:

1) выбор (или получение от системы верхнего уровня) целевого критерия оценки качества решения;
2) сбор информации о ресурсах предприятия или возможностях внешней среды;
3) просчет вариантов (с различными предположениями о возможных значениях параметров);
4) выбор оптимального варианта (решения, ресурсного плана);
5) учет результатов (и отчетность);
6) сравнение с принятым критерием оценки (контроль результатов);
7) анализ причин отклонений и регулирование (возврат к 1, 2 или 3).

В основе цикла *организационного менеджмента* лежит *структурное или процессное моделирование и процедурный контроль*:

1) определение состава задач (обособленных функций, операций);
2) выбор исполнителей (распределение зон и степени ответственности);
3) проектирование процедур (последовательности и порядка исполнения);
4) согласование и утверждение регламента исполнения (- процесса, плана мероприятий);
5) отчетность об исполнении;
6) контроль исполнения (процедурный контроль);
7) анализ причин отклонений и регулирование (возврат к 1, 2 или 3).

Таким образом, на определенных шагах декомпозиции необходимо определить, какие стадии управленческого цикла реализуются по каждой из ранее выделенных задач управления.

Процессы обеспечения – это процессы, предназначенные для жизнеобеспечения основных и управленческих процессов и ориентированные на поддержку их универсальных средств (например, ремонт оборудования, с помощью которого осуществляется процесс производства). Это «горизонтальные» бизнес-процессы, не имеющие непосредственного отношения к производимым товарам и услугам, однако, без них невозможно выполнение операций по созданию добавленной стоимости.

Например, процессы *финансового обеспечения*, *обеспечения кадрами*, *юридического обеспечения* являются вторичными. Они создают и поддерживают необходимые условия для выполнения основных функций и функций менеджмента. Клиенты обеспечивающих процессов находятся внутри компании.

На верхнем уровне детализации можно выделить примерно следующие стандартные процессы обеспечения:

- техобслуживание и ремонт оборудования;
- обслуживание и ремонт зданий и сооружений;
- метрологическое обеспечение;
- юридическое обеспечение;
- обеспечение безопасности;
- хозяйственное обеспечение;
- обеспечение коммунальными услугами;
- транспортное обслуживание и т.п.

Для каждого из выделенных выше подпроцессов следует определить, какой *основной* или *управленческий* процесс является потребителем этих «внутренних» услуг. Процессы обеспечения условно можно делить на *сопутствующие*, *вспомогательные* и процессы *развития*.

Текстовое описание БП

Текстовое описание БП является результатом исследования системы управления с применением определенного метода (интервьюирование, фотография рабочего времени и т.д.) (см. «Предпроектное обследование объекта»). Текстовому описанию БП обычно предшествует определение основной деятельности организации и иерархической функциональной организационной структуры с выделением функциональных областей – подразделений и должностей.

Если применяется метод интервьюирования, то текстовое описание БП может представлять собой ответы компетентных лиц (руководителей подразделений и основных исполнителей) на следующие вопросы:

- что поступает в организацию (подразделение) на «входе»;
- какие функции и в какой последовательности выполняются в рамках должности (подразделения);
- кто является ответственным за выполнение каждой из функций;
- чем руководствуется исполнитель при выполнении каждой из функций;
- что является результатом работы организации (подразделения) на выходе?

При формулировке ответов рекомендуется каждой работе присвоить номер или идентификатор, а также использовать правила, представленные ниже-следующими формулами.

Исполнитель + «:» + Номер работы + Названия работы + Основание (правило) выполнения

Название работы = Действие + Объект, над которым действие осуществляется

При этом необходимо конкретизировать объект, над которым осуществляется действие. Например, основная деятельность компании – сборка и про-

дажа ПК, и работа связана с действием по «производству продукции». В этом случае работу нужно назвать не «производство продукции», а «сборка ПК» и еще лучше – с указанием правила, например – «сборка ПК на основании производственного плана». В данном случае «сборка» – это действие, «ПК» – объект над которым действие осуществляется, «производственный план» – основание (правило) выполнения.

При формулировании названия работы нужно стараться использовать краткую и лаконичную формулировку. Идеальный вариант – формулирование названия работы при помощи 2-3 слов, в крайнем случае – использование в названии не более 50 символов. В сложных случаях рекомендуется для каждого краткого названия работы сделать ее подробное описание, которое поместить в глоссарий.

Графическое описание БП

Этапы графического описания БП

Графическое описание процессов организации обычно состоит из двух этапов:

1) разработка схемы бизнес-процессов в виде дерева;
2) построение сетевой схемы (сети) бизнес-процессов на основе дерева.

Для графического описания БП используют различные методики, стандарты, CASE-средства. При проектировании ИС для моделирования и анализа предметной области используют стандарты (формальные языки графического описания) IDEF0, DFD, IDEF3. При создании черновика модели удобно использовать менее формализованные способы описания, одно из которых описано ниже.

Построение дерева БП

Основанием для построения иерархии процессов целесообразно использовать классификационные стандарты (см. «Классификация бизнес-процессов»). Пример дерева БП приведен на рис. 3.2. «Ветками» первого уровня будут процессы основные, обеспечивающие и управления. Названия «веток» следующих уровней должны браться из текстового описания процессов.

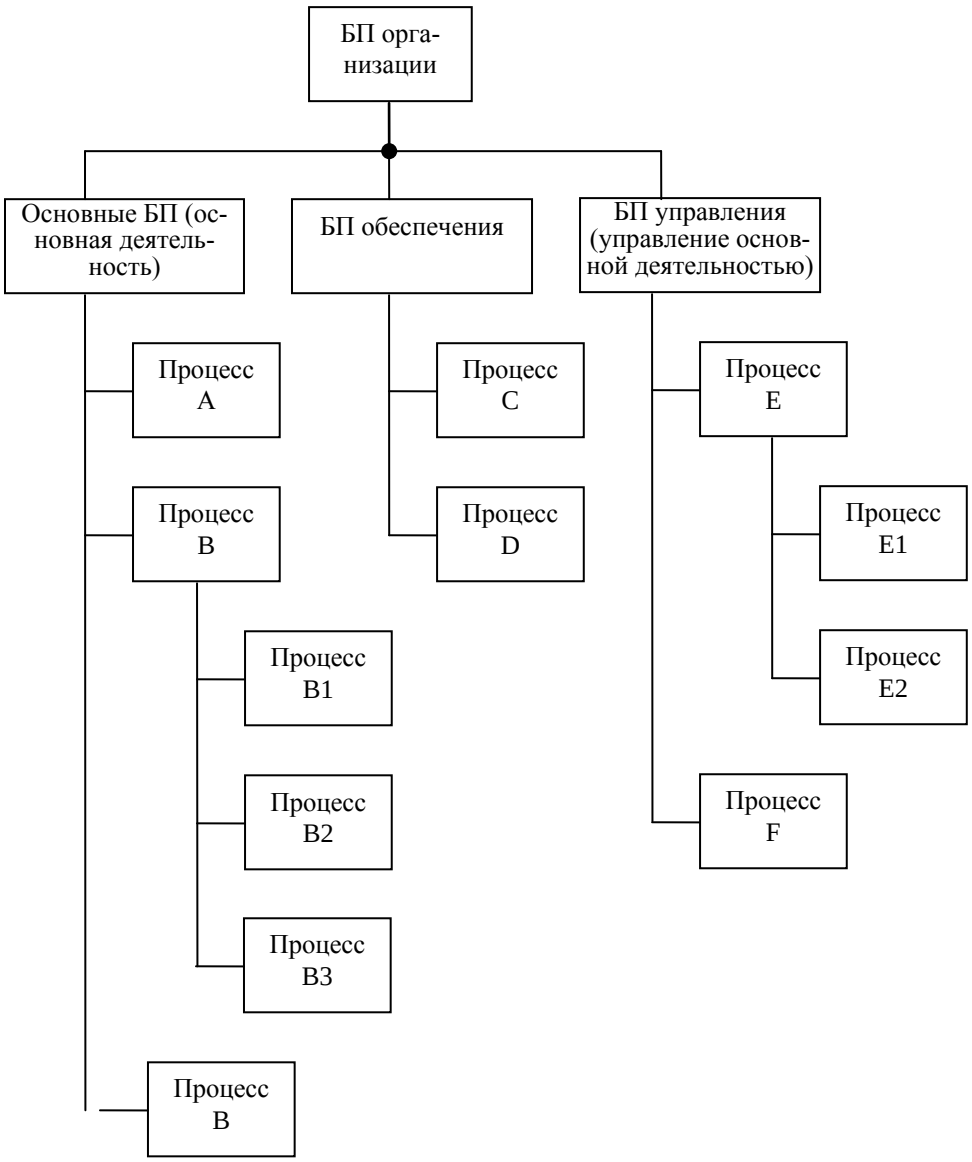

Рис. 3.2. Дерево бизнес процессов

Таким образом, работы, перечисленные на этапе текстового описания, «распределяются» по «веткам» классификационного дерева.

Примеры БП управления (второй уровень дерева на ветке «БП управления»): *планирование бюджета; составление штатного расписания; планирование закупок; планирование продаж и т.д*. Для крупных предприятий БП можно объединять в подкатегории и подгруппы. Например, для БП управления это могут быть следующие категории: *управление закупками и запасами; управление производством; управление продажами; управление финансами и т.д*.

Следующим шагом при проектировании ИС является выбор объекта автоматизации – группы процессов, которые будут осуществляться при помощи средств вычислительной техники. Это могут быть все основные процессы (для разработки автоматизированной системы учета основной деятельности), часть основных, например, для автоматизации продаж, для автоматизации производства, могут включать процессы управления или часть процессов управления при проектировании АСУ. В результате происходит «сужение» предметной области, и процессы, оставшиеся за ее пределами, попадают в область «внешней среды».

В IDEF0 инструментом построения иерархии процессов является «дерево узлов» (Node Tree) (рис. 3.3.).

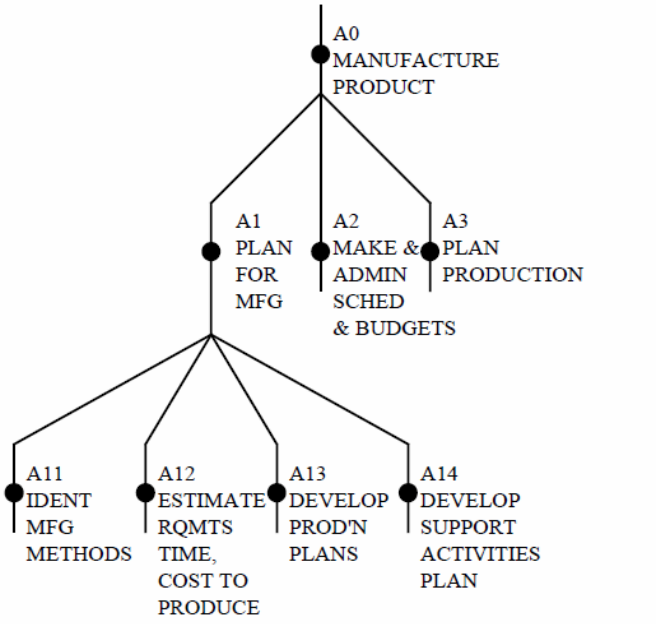

Рис. 3.3. Пример стандартного дерева узлов

При «сужении» предметной области старший (контекстный) процесс выделенной предметной области именуется A0, а процессы «окружения» – A-n (A минус n), где n – не менее 1 (см. IDEF0 Standards Publication 183, 3.4.3 High-Level Context Diagrams) (рис. 3.4.).

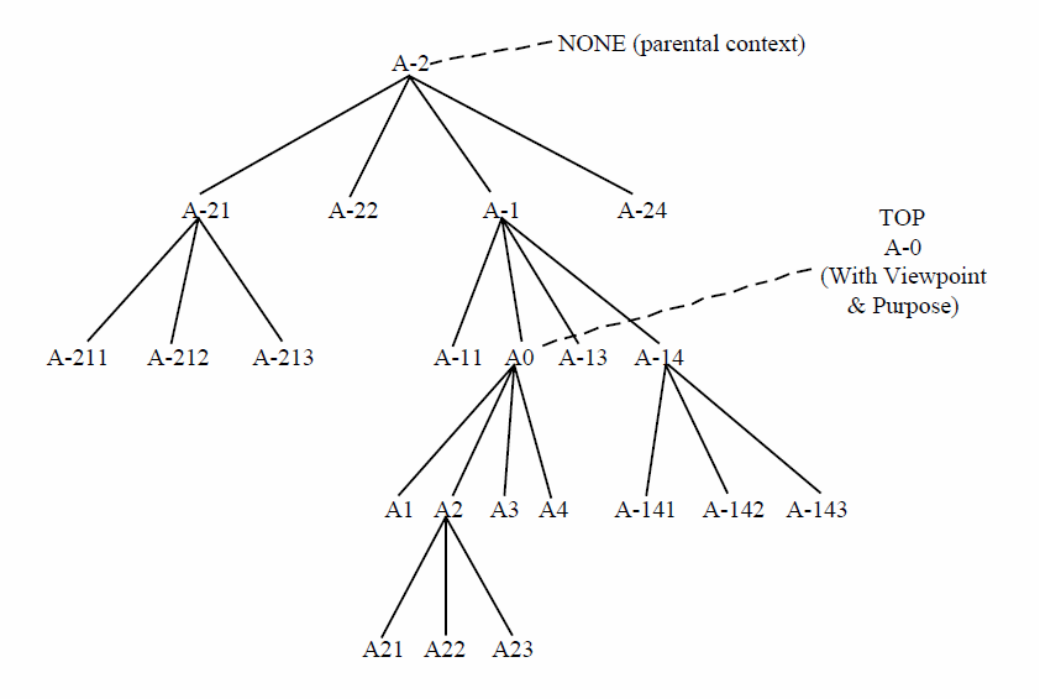

Рис. 3.4. Negative Node-Numbered Context

Построение сети БП

На практике сеть процессов часто называют схемой взаимодействия бизнес-процессов. Отличие сети процессов от классической древовидной схемы состоит в том, что на сети показывают внешние субъекты, с которым взаимодействуют бизнес-процессы компании – клиенты, поставщики, банки и др. В результате сетевая схема дает более наглядное представление бизнес-процессов компании, поскольку описывает их взаимодействие друг с другом и внешней средой. При этом *внешним* субъектом на сети БП может быть как тот, с кем взаимодействует организация, так и исполнитель внутреннего организационного, но внешнего по отношению к выбранной предметной области процесса.

Первый шаг построения сети БП – формулирование **целей** моделирования и выбор **точки зрения** (см. гл. 4.2). Входы (то, что расходует процесс) и выходы (результат выполнения процесса) зависят от выбранной точки зрения и поставленных целей. При проектировании ИС чаще всего выбирают точку зрения заказчика (владельца или директора организации). В этом случае входами будет то, что «имеет» («видит») заказчик на начало выполнения, а выходами – то, что он «видит» («имеет») по завершении процесса.

Вход и выход каждого бизнес-процесса являются соответственно выходом и входом для другого бизнес-процесса или *внешнего* субъекта (рис.).

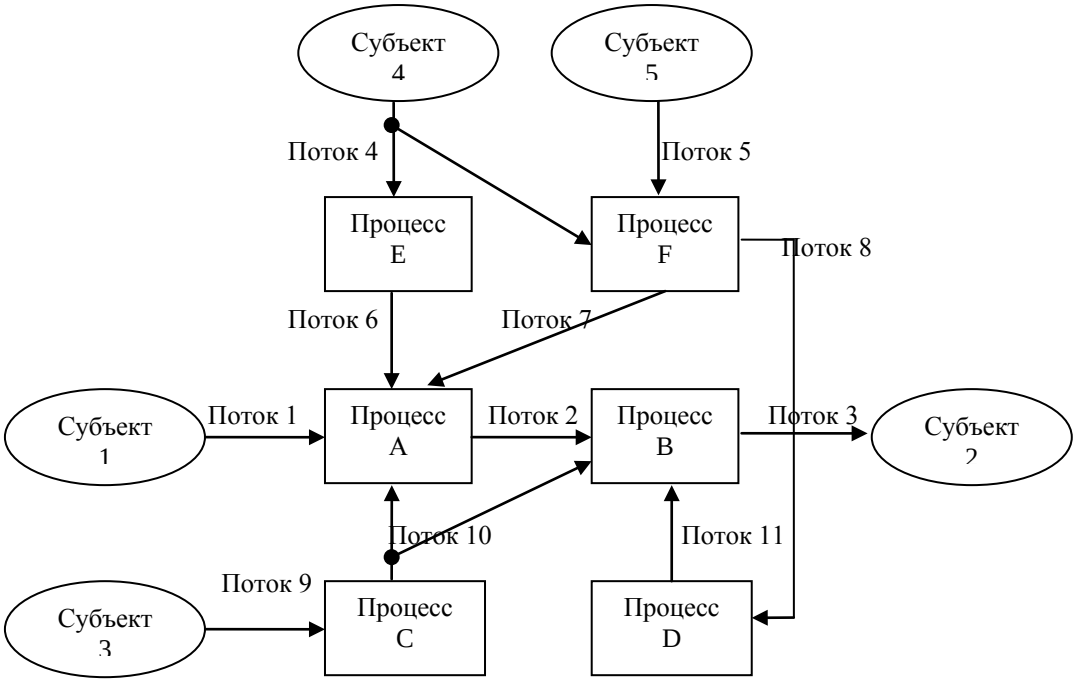

Рис. 3.5. Построение сети бизнес-процессов

При формулировании названий материальных и информационных потоков нужно использовать правило, которое можно сформулировать следующей формулой:

Название потока = Объект, представляющий поток + Статус объекта

Например, если речь идет о продукции, которую отгрузили клиенту, то данный поток нужно сформулировать следующим образом: «Продукция, отгруженная» или «Продукция, отгруженная клиенту». В данном случае «Про-

дукция» – это объект, представляющий поток, а «отгруженная клиенту» – статус объекта.

При построении сетевой схемы бизнес процесса нужно помнить, что такая схема показывает потоки материальных и информационных объектов и не говорит о временной последовательности работ. При этом временная последовательность выполнения процессов может совпадать с направлением движения объектов.

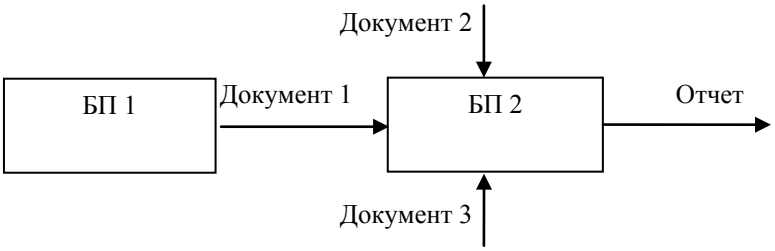

Рис. 3.6. Фрагмент схемы получения отчета

В примере, приведенном на рис. 3.6 БП 2 может начаться раньше первого, но Документ 1 движется от первого БП ко второму.

Рассмотрим пример бизнес-процесса, схема которого приведена на рис. 3.7.

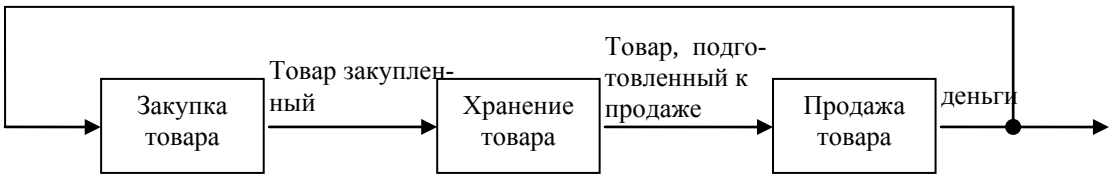

Рис. 3.7.Фрагмент схемы БП верхнего уровня

На вопрос, что происходит раньше – закупка продукции или ее продажа, могут быть даны два различных ответа в зависимости от двух различных ситуаций. Если конкретный продукт имеется на складе, то его закупка по времени происходит раньше, чем продажа. Если при обращении клиента продукции на складе нет, и клиент готов подождать пока будет произведена закупка, то процесс продажи начнется раньше, чем закупка, а закончится позже.

Декомпозиция БП. При декомпозиции каждый процесс описывают в виде отдельной схемы следующего уровня. При этом каждый процесс должен

включать от трех до шести «вложенных» процессов (или подпроцессов), что связано с особенностями восприятия. В случае необходимости работы на схеме процесса второго уровня могут быть декомпозированы на схемы бизнес-процессов третьего уровня и т.д. Декомпозиция бизнес-процесса должна продолжаться до тех пор, пока не будут достигнуты цели его описания. Использование слишком глубокой детализации и большого количества работ приведет к сильному усложнению схемы и снижению возможности проведения качественного анализа бизнес-процесса. Использование небольшой детализации и меньшего количества работ на схеме бизнес-процесса приведет к тому, что работы будут достаточно укрупненными, что также уменьшит возможность проведения их качественного анализа и оптимизации.

В отличие от дерева бизнес-процессов, сеть процессов дает более полное системное представление о предметной области, так как позволяет показать не только элементы деятельности, но и взаимодействия между ними.

Кроме того, сеть процессов обеспечивает *проверку* разработанной модели деятельности организации на целостность, правильность выделения бизнес-процессов и описания их окружения. Если выход одного из бизнес-процессов, например, документ, нигде далее не используется, то есть не является входом для другого бизнес-процесса или внешнего субъекта, это означает, что описанный выход бизнес-процесса является либо ошибочным, либо лишним. В противном случае нужно найти бизнес-процесс, для которого данный выход является входом, и доработать схему окружения этого бизнес процесса.

Стандарты графического описания бизнес процессов

Процессы могут быть описаны посредством IDEF0-модели, диаграммы потоков данных DFD или диаграммы потоков работы WFD (см. гл. 3.2 – 3.3).

Стандарты IDEF0 и DFD удобны для описания процессов верхнего уровня, когда невозможно указать временную последовательность работ, так как все

работы могут выполняться одновременно, или существует несколько вариантов различных последовательностей, которые, к тому же, могут зависеть от точки зрения. Для представления простых операций нижнего уровня, когда необходимо описать последовательность и условия их выполнения, можно использовать WDF (IDEF3) или другие графические языки описания алгоритмов.

Для описания предметной области чаще применяют методологию функционального моделирования IDEF0, которая является универсальным стандартом описания любых систем (как социо-организационных, так и информационных). Диаграммы потоков данных DFD чаще используют для описания информационных систем – существующих и проектируемых.

В итоге описание деятельности объекта представляет собой иерархически упорядоченный набор IDEF0, DFD и WFD связанных между собой диаграмм, в котором диаграммы верхнего уровня ссылаются на схемы нижнего уровня.

Язык UML, изначально разработанный в основном для проектирования программного обеспечения, может быть использован и для моделирования бизнес-процессов, а также отображения организационных структур.

3.4. Моделирование предметной области

Основные понятия теории моделирования

Моделирование – это замещение одного объекта другим с целью получения информации о важнейших свойствах объекта-оригинала с помощью объекта-модели путем проведения эксперимента с моделью; метод прогнозирования возможных состояний объекта в будущем и способов достижения заданных параметров с применением моделей (предметных, знаковых, математических, имитационных, аналитических и т.д.).

В результате деятельности математиков, логиков и философов была создана *теория моделей*, согласно которой *модель* (от лат. *modulus* – мера) – это

результат отображения одной абстрактной материальной структуры на другую, также абстрактную, либо результат интерпретации первой структуры в терминах и образах второй. Таким образом, *модель* – некий объект-заместитель, который в определенных условиях может заменять объект-оригинал, воспроизводя интересующие нас свойства и характеристики оригинала, причем имеет существенные преимущества удобства и обеспечивает возможность изучения некоторых свойств оригинала. *Модель* можно также определить как способ существования знаний.

Модель считается *адекватной,* если с приемлемой точностью выходные параметры модели (свойства, характеристики) совпадают с истинными их значениями объекта. Адекватность зависит от *цели* моделирования и принятых *критериев*.

Модели могут быть качественно различными, они образуют иерархию, в которой модель более высокого уровня (например, *теория*) содержит модели нижних уровней (например, *гипотезы*) как свои части, элементы.

Моделирование лежит в основе любой целесообразной деятельности, и сама по себе цель уже есть модель желаемого состояния. Алгоритм деятельности – это модель этой деятельности, которую предстоит реализовать.

Общее представление системы наиболее удобно использовать в форме математической модели, например, в виде контуров обслуживания или агрегата (рис. 3.8.), который является центральным звеном функционирования сложной системы. В каждый момент времени t агрегат находится в одном из возможных состояний Z(t). Состояние агрегата в фиксированный момент времени определяется управляющим воздействием g(t) в соответствии с оператором перехода H с использованием зависимости: $Z(t) = H\{Z(t°), g(t)\}$. Агрегат имеет входные контакты. На них поступают входные сигналы X(t), которые в соответствии с оператором G преобразуются в выходные сигналы Y(t). Данная схема допускает варьирование большим числом параметров, используемых для характеристики состояния системы, в то же время требует упрощения набора этих параметров до предельно абстрактной модели, наиболее полно отражающей основные

из них, и позволяющая прогнозировать будущие тенденции развития. Наглядность и абстрактность является преимуществом рассматриваемой векторной модели.

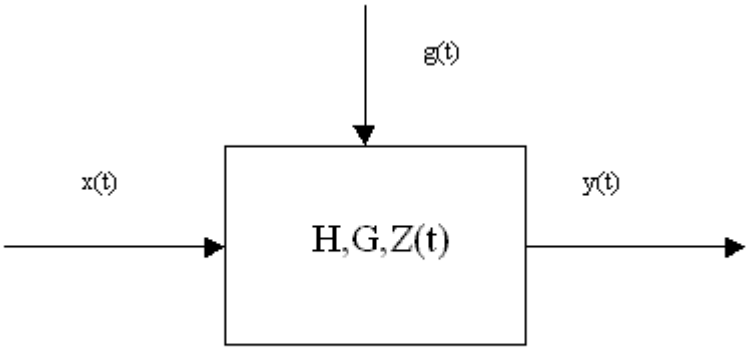

Рис. 3.8. Представление системы в виде агрегата

Модели предметной области

В основе моделирования лежит *теория подобия*, которая утверждает, что абсолютное подобие может иметь место лишь при замене одного объекта другим, точно таким же. При этом при моделировании абсолютное подобие не имеет места, необходимо лишь, чтобы модель адекватно отображала исследуемую сторону функционирования объекта.

Модель при исследовании системы управления – это упрощенное представление объекта, которое должно отвечать требованиям полноты, адаптивности и обеспечивать возможность включения достаточно широких изменений.

При моделировании предметной области выделяют (с точки зрения учета временного фактора) *статические* и *динамические* модели.

Статические модели описывают содержательную сторону системы. Они могут быть *функциональными*, т.е. описывать принципы функционирования системы и *информационными*, т.е. описывать состояние информации, на основе которой функционирует система.

Динамические модели отражают поведение системы во времени, вплоть до учета факторов ее развития.

Существует и другая классификация моделей предметных областей: *морфогенеза (структуры)*, *поведения*, *управления*, *развития и др*.

К моделям предметных областей предъявляются следующие *требования*:

- формализация, обеспечивающая однозначное описание структуры предметной области;
- понятность для заказчиков и разработчиков на основе применения графических средств отображения модели;
- реализуемость, подразумевающая наличие средств физической реализации модели предметной области в ИС;
- обеспечение оценки эффективности реализации модели предметной области на основе определенных методов и вычисляемых показателей.

Для реализации перечисленных требований, как правило, строится *система моделей*, которая отражает *структурный и оценочный аспекты* функционирования предметной области.

Структурный аспект предполагает построение:

1) *организационной структуры*, отражающей взаимодействие организационных единиц предприятия и персонала в процессах;
2) *функциональной структуры*, отражающей взаимосвязь функций (действий) по преобразованию объектов в процессах;
3) *информационной структуры*, отражающей состав взаимодействующих в процессах материальных и информационных объектов предметной области;
4) *технической структуры*, описывающей топологию расположения и способы коммуникации комплекса технических средств.
5) структуры управления, отражающей события и бизнес-правила, которые воздействуют на выполнение процессов.

Для отображения структурного аспекта моделей предметных областей в основном используются *графические методы*, как наиболее емкий способ представления информации о системе. Главное требование к графическим методам документирования – простота.

С моделированием связана *проблема выбора языка моделирования*. *Язык моделирования* – это нотация, в основном графическая, которая используется для описания проектов. Нотация представляет собой совокупность графических объектов, используемых в модели и является синтаксисом языка моделирования. Язык моделирования, с одной стороны, должен делать решения проектировщиков понятными пользователю, с другой стороны, предоставлять проектировщикам средства формализованного и однозначного представления проекта, подлежащего дальнейшей программной реализации.

Выбор языка моделирования является одной из первых задач процесса проектирования системы (см. гл. 1.6).

Оценочные аспекты моделирования предметной области связаны с разрабатываемыми показателями эффективности автоматизируемых процессов, которые включают также косвенные показатели эффективности, такие, как объемы производства, производительность труда, оборачиваемость капитала, рентабельность и т.д.

Для расчета показателей эффективности, как правило, используются *статические* методы *функционально-стоимостного анализа* (ABC) и *динамические* методы *имитационного моделирования*.

Моделирование процессов, как правило, выполняется с помощью CASE-средств. К таким средствам относятся *BPwin* (PLATINUM technology), Silverrun (Silverrun technology), Oracle Designer (Oracle), Rational Rose (Rational Software) и др. *BPwin* поддерживает три методологии моделирования: функциональное моделирование (IDEF0); моделирование рабочих потоков (IDEF3); диаграммы потоков данных (DFD).

3.5. Оценка эффективности и целесообразности ИТ-проекта

Подходы к оценке эффективности ИТ -проекта

Первичные цели автоматизации – ускорение выполнения процессов, разгрузка персонала, удешевление производства, улучшения качества информация для принятия решения. Тенденция в развитии ИТ на фазе роста – непрерывное усложнение технологий, а также методик проектирования и требования к системам (достаточно взглянуть на группы ГОСТов Союза ССР групп 24 и 34). В результате (ради ускорения бизнес-процесса) тратится огромное количество времени на проектирование и разработку ПО (правда один раз). Разгрузка персонала (пользователей системы) при выполнении работ становится несравнима с загрузкой проектировщиков и программистов. Мы имеем удешевление производства, с одной стороны, и колоссальные затраты на стадии проектирования, разработки, внедрения и поддержки системы, с другой. Уменьшается штат производственного персонала, при этом неимоверно разрастаются отделы АСУ, затраты на труд ИТ-специалистов по непрерывной поддержке и модернизации системы часто несравнимы с эффектом от использования ИС. Иногда отказ от технологии является выходом из ситуации. Оценка целесообразности внедрения ИТС до начала проектных работ помогает избежать потерь.

В современной практике существуют два основных метода оценки эффективности проекта, реализуемых на действующих предприятиях: *метод расчета по предприятию в целом* и *приростной метод*.

Расчет по предприятию в целом рекомендуется производить, сопоставляя варианты проекта развития предприятия в целом «с проектом» и «без проекта» (далее соответственно «основной» и «нулевой» варианты). Формирование основного варианта производится путем внесения соответствующих корректировок в показатели нулевого варианта. Если внедрение ИТС оказывает влияние на различные стороны деятельности предприятия, могут измениться технико-

экономические и финансовые показатели работы предприятия в целом. Такая ситуация может возникнуть при внедрении интегрированных корпоративных систем управления. При этом необходимо сопоставить перспективные показатели работы предприятия при условии реализации ИТС-проекта, связанного с внедрением интегрированных корпоративных систем управления и при условии отказа от него. Такое сопоставление выполняется на основе метода расчета по предприятию в целом.

Для оценки эффективности инвестиционных проектов внедрения ИТС, имеющих локальный характер на предприятии, используется **приростной метод**, основная идея которого состоит в определении изменений притоков и оттоков денежных средств, обусловленных реализацией проекта. Главная проблема, которая возникает при применении **приростного метода**, − точное выявление факторов, определяющих эффективность проекта (например, уменьшение трудоемкости выполнения операции, сокращение количества работающих и т. д.) и правильная количественная оценка изменений финансовых затрат и результатов с учетом указанных факторов.

Фактическим стандартом при **расчете стоимости** ИС стал **метод ТСО** (совокупная стоимость владения), предложенный компанией Gartner. Позже многие крупные поставщики ИТ представили свои версии данного метода. Но ТСО оценивает только затратную часть, не учитывая преимуществ от внедрения. Поэтому эта методика в чистом виде применима только для оценки решений, обеспечивающих сходную функциональность.

Метод ROI (отдача от инвестиций) дает возможность определить, какой финансовый результат обеспечивает каждый рубль, инвестированный в проект, но не дает четкого и ясного способа определить абсолютную величину этого результата. Определение качественных и финансовых эффектов наилучшим образом обеспечивается системой сбалансированных показателей (Balanced ScoreCard, BSC), но для ее внедрения необходима длительная подготовительная работа, зачастую требующая изменения существующих подходов к управлению компанией.

Более простая и доступная для использования методика, но в то же время дающая четкие и обоснованные результаты, была разработана компанией Microsoft − *методика «быстрого экономического обоснования (Rapid Economic Justification, REJ)*.

Методика быстрого экономического обоснования

Методика «быстрого экономического обоснования» (Rapid Economic Justification, REJ, Microsoft) позволяет преодолеть языковый барьер между ИТ-специалистами и исполнителями бизнеса и продемонстрировать выгоду от инвестиции в ИТ.

Заинтересованные лица (Stakeholders):

- Генеральный директор − CEO (Chief Executive Officer).
- Финансовый директор − CFO (Chief Financial Officer).
- Вице-президент – VP.
- Директор по информационным технологиям − CIO (Chief Information Officer) − сотрудник корпорации, исполнитель высшего ранга; отвечающий за приобретение и внедрение новых технологий, управление информационными ресурсами.
- ИТ-персонал − IT stuff.
- Торговые посредники – Resellers-фирмы, специализирующиеся на оптовых поставках и торговом посредничестве.
- Поставщики – Suppliers.

Шаг 1. Обследование объекта (Understand the Business). Исследование начинается с определения ключевых проблем предприятия и поиск ИТ-решения, которое будет способствовать устранению этих проблем и позволит предприятию добиться успеха. На этой стадии определяются критические факторы успеха предприятия и их показатели, а также составляется план их дости-

жения. Для этого исследуют стратегический план развития компании, бизнес-план, проводят консультации с руководством компании, руководителями функциональных подразделений и ключевыми специалистами.

Шаг 2. Выбор технических решений (Understand the Solutions). На этой стадии проектная команда работает с владельцами ключевых бизнес-процессов, используя схемы процессов в организации, функционально-структурные схемы и диаграммы причинно-следственных связей с целью определения способов применения ИТ-решений для повышения их соответствия критическим факторам успеха организации. Для каждой работы, определенной на предыдущем шаге, необходимо найти, с использованием каких ИТ можно улучшить ее эффективность.

Шаг 3. Вычисление разницы между доходами и расходами (Understand the Benefit/Cost Equation). После того как возможные технические решения выбраны, команда аналитиков вычисляет потенциальную прибыль от их внедрения и необходимый объем капиталовложений для каждого проекта.

Шаг 4. Выявление рисков (Understand the Risks). Многие сначала экономически оправданные ИТ-проекты после внедрения не соответствуют ожиданиям руководства и других заинтересованных лиц. Анализ потенциальных рисков инвестиций в ИТ может помочь избежать провала путем выявления различных форм риска, разработки решения о их смягчении и подгонки прибыли к затратам. Существуют различные **категории рисков**:

- *риск соответствия.* Чем жестче соответствие ИТ-проекта целям предприятия, тем меньше риск. Необходимо заметить, что для некоторых проектов установление четкого соответствия технологий стратегическим целям бизнеса — задача сложно выполнимая (например, усовершенствование инфраструктуры информационной системы), однако инвестиции в них являются необходимыми для дальнейшего развития информационных технологий;

- *реализационный риск.* Учитывает возможность того, что действительная стоимость реализации проекта будет отличаться от расчетной;
- *операционный риск.* Учитывает возможность того, что стоимость функционирования системы будет отличаться от предполагаемой;
- *технологический риск*. Чем больше известно о выбранном решении и чем лучше проработаны выбранные технологии, тем меньше этот риск. Однако проекты с малым значением технологического риска не всегда обеспечивают достаточно высокие потенциальные преимущества и, наоборот, чем выше риск, тем выше возможная прибыль;
- *риск денежных потоков*. Учитывает возможность недостоверного определения выгод от проекта и неточного расчета положительных денежных потоков, а также возможность появления других непредвиденных финансовых проблем. Например, будет принято решение увеличить капитализацию бизнеса или другие, более важные с точки зрения руководства, проблемы потребуют отвлечения средств от рассматриваемого проекта, в результате чего не удастся достичь предполагаемых выгод в полном объеме.

  Риски могут быть описаны как количественно, так и качественно.

Шаг 5. Расчет финансовых показателей (Understand the Financial Metrics). На основе полученных дисконтированных денежных потоков, скорректированных с учетом рисков, рассчитываются финансовые показатели, принятые на данном предприятии. Такими показателями могут быть чистый приведенный доход (Net Present Value, NPV), внутренняя норма доходности (Internal Rate of Return, IRR), добавленная стоимость (Economic Value Added, EVA), срок окупаемости, возврат от инвестиций и другие.

4. СТРУКТУРНЫЙ АНАЛИЗ И СТРУКТУРНОЕ ПРОЕКТИРОВАНИЕ

4.1. Основные понятия структурного анализа и структурного проектирования

Определения понятий

Структурным анализом принято называть метод исследования системы, которое начинается с ее общего обзора и затем детализируется, приобретая иерархическую структуру с все большим числом уровней. Решение трудных проблем путем их разбиения на множество меньших независимых задач (так называемых «черных ящиков») и организация этих задач в древовидные иерархические структуры значительно повышают понимание сложных систем.

В инженерии ПО (software engineering), **Структурный анализ** (Structured Analysis, **SA**) и одноименное с ним **Структурное проектирование** (Structured Design, **SD**) – это методы для анализа и преобразования бизнес-требований в спецификации и, в конечном счете, в компьютерные программы, конфигурации аппаратного обеспечения и связанные с ними ручные процедуры.

Структурный анализ, **СА** (Structured Analysis, **SA**) и Структурное проектирование, **СП** (Structured Design, **SD**) являются фундаментальными инструментами **системного анализа** и развивались из классического системного анализа 1960-70-х годов (см. гл. 1.1).

Структурный подход заключается в поэтапной декомпозиции системы при сохранении целостного о ней представления Основные принципы структурного подхода (первые два являются основными):

1) *принцип «разделяй и властвуй»* – принцип решения сложных проблем путем их разбиения на множество меньших независимых задач, легких для понимания и решения;

2) *принцип иерархического упорядочивания* – принцип организации составных частей проблемы в иерархические древовидные структуры с добавлением новых деталей на каждом уровне.
3) *принцип абстрагирования* – заключается в выделении существенных аспектов системы и отвлечения от несущественных;
4) *принцип формализации* – заключается в необходимости строгого методического подхода к решению проблемы;
5) *принцип непротиворечивости* – заключается в обоснованности и согласованности элементов.

История структурного анализа и проектирования

Структурный анализ – это часть серии структурных методов, представляющих набор методологий анализа, проектирования и программирования, которые были разработанны в ответ на проблемы, с которыми столкнулся мир ПО в период с 1960 по 1980 гг. В этот период большинство программ было создано на Cobol и Fortran, потом на C и BASIC.

Это было новым положительным сдвигом в направлении проектирования и программирования, но при этом не было стандартных методологий для документирования требований и самих проектов. По мере укрупнения и усложнения систем, все более затрудниельным становился процесс их разработки.

Когда большинство специалистов билось над созданием программного обеспечения, немногие старались разрешить более сложную задачу создания крупномасштабных систем, включающих как людей и машины, так и программное обеспечение, аналогичных системам, применяемым в телефонной связи, промышленности, управлении и контроле за вооружением. В то время специалисты, традиционно занимавшиеся созданием крупномасштабных систем, стали осознавать необходимость большей упорядоченности. Таким образом, разработчики начали формализовать процесс создания системы, разбивая его на следующие фазы:

1) анализ – определение того, что система будет делать;
2) проектирование – определение подсистем и их взаимодействие;
3) реализация – разработка подсистем по отдельности;
4) объединение – соединение подсистем в единое целое;
5) тестирование – проверка работы системы;
6) установка – введение системы в действие;
7) функционирование – использование системы.

Эта последовательность всегда выполнялась итерационно, потому что система полностью никогда не удовлетворяла требованиям пользователей, поскольку их требования часто менялись. И, тем не менее, с этой моделью создания системы, ориентированной на управление, постоянно возникали сложности. Эксплуатационные расходы, возникавшие после сдачи системы, стали существенно превышать расходы на ее создание и продолжали расти с огромной скоростью из-за низкого качества исходно созданной системы. Некоторые считали, что рост эксплуатационных расходов обусловлен характером ошибок, допущенных в процессе создания системы. Исследования показали, что большой процент ошибок в системе возник в процессе анализа и проектирования, гораздо меньше их было допущено при реализации и тестировании, а цена (временная и денежная) обнаружения и исправления ошибок становилась выше на более поздних стадиях проекта. Например, исправление ошибки на стадии проектирования стоит в 2 раза, на стадии тестирования – в 10 раз, а на стадии эксплуатации системы – в 100 раз дороже, чем на стадии анализа. На обнаружение ошибок, допущенных на этапе анализа и проектирования, расходуется примерно в 2 раза больше времени, а на их исправление – примерно в 5 раз, чем на ошибки, допущенные на более поздних стадиях. Кроме того, ошибки анализа и проектирования обнаруживались часто самими пользователями, что вызывало их недовольство.

Традиционные подходы к созданию систем приводили к возникновению многих проблем:

1) не было единого подхода;

2) привлечение пользователя к процессу разработки не контролировалось;
3) проверка на согласованность проводилась нерегулярно или вообще отсутствовала, результаты одного этапа не согласовывались с результатами других;
4) процесс с трудом поддавался оценкам, как качественным, так и количественным.

Утверждалось, что когда создатели систем пользуются методологиями типа структурного программирования и проектирования сверху вниз, они решают либо не поставленные задачи, либо плохо поставленные, либо хорошо поставленные, но неправильно понятые задачи. Кроме того, ошибки в создании систем становились все менее доступны выявлению с помощью аппаратных средств или программного обеспечения, а наиболее катастрофические ошибки допускались на ранних этапах создания системы. Часто эти ошибки были следствием неполноты функциональных спецификаций или несогласованности между спецификациями и результатами проектирования. Проектировщики знали, что сложность систем возрастает и что определены они часто весьма слабо. Рост объема и сложности систем является жизненной реалией. Эту предпосылку нужно было принять как неизбежную. Но ошибочное определение системы не является неизбежным: оно – результат неадекватности методов создания систем.

Вскоре был выдвинут тезис: совершенствование методов анализа есть ключ к созданию систем, эффективных по стоимости, производительности и надежности. Для решения ключевых проблем традиционного создания систем широкого профиля требовались новые методы, специально предназначенные для использования на ранних стадиях процесса.

Для помощи в управлении большим и сложным ПО с конца 1960 годов появляется множество разнообразных структурных методов программирования, проектирования и анализа. Эти методы на начальных этапах создания системы позволяют гораздо лучше понять рассматриваемую проблему. А это сокращает затраты как на создание, так и на эксплуатацию системы, а кроме того, повышает ее надежность.

В 1960-70 появляются следующие концепции:

- примерно **1967** – Структурное программирование (Structured programming) – Edsger Dijkstra,
- примерно 1975 – Структурное программирование Джексона (Jackson Structured Programming) – Michael A. Jackson.

Структурное программирование приводит к Структурному проектированию, что в свою очередь приводит к Структурному системному анализу:

- примерно **1975** – появление на рынке Метода структурного анализа и проектирования **SADT** (Structured Analysis and Design Technique) – Douglas T. Ross;
- примерно **1975 –** Структурное проектирование (Structured Design) – Larry Constantine, Ed. Yourdon и Wayne Stevens;
- примерно **1978** – Структурный анализ (Structured Analysis) – Tom DeMarco, Yourdon, Gane & Sarson, McMenamin & Palmer;
- в 1979 опубликован Структурный анализ и системная спецификация (Structured Analysis and System Specification) – Tom DeMarco.

В течение **1980x** начинают появляться инструменты для автоматизации черчения диаграмм:

- **1981 –** опубликована (и в 85-93 получает развитие) Методология IDEF0, основанная на SADT и инструментальных средствах создания диаграмм (разработана Дугласом Т. Россом, Douglas T. Ross).
- в **1983** впервые представлен Метод структурного системного анализа и проектирования **SSADM** (Structured Systems Analysis and Design Method), разработанный в UK Office of Government Commerce.

По аналогии с Computer-Aided design and Computer-Aided Manufacturing (CAD/CAM), использование этих инструментов было названо **Computer-Aided Software Engineering** (**CASE**).

Методы СА и СП (в частности, **SSADM**) сопровождаются нотациями (диаграммами), облегчающими взаимодействие между пользователями и разработчиками. Это были:

- структурные схемы (Structure Charts) – для структурного проектирования;
- диаграммы потоков данных (Data Flow Diagrams, DFD) – для структурного анализа;
- модели данных (Data Model Diagrams).

Эти диаграммы использовались структурными методами в различных комбинациях. Среди них встречалось множество вариаций.

Примерно в **1990** появляется термин «инженерия разработки ПО» (Information Engineering, IE, James Martin), являющаяся логическим расширением структурных методов, появившихся в течение 1970х.

4.2. Метод структурного анализа и проектирования SADT

Определение SADT

SADT (Structured Analysis and Design Technique) – это методология инженерии разработки ПО (software engineering) для описания систем в виде иерархии функций (функциональной структуры).

Предпосылки и история SADT

Структурный анализ возник в конце 60-х годов в ходе революции, вызванной структурным программированием. Метод SADT был предложен Дугласом Т. Россом как способ уменьшить количество дорогостоящих ошибок за счет структуризации на ранних этапах создания системы, улучшения контактов между пользователями и разработчиками и сглаживания перехода от анализа к проектированию. Дуглас Т. Росс часть своих PLEX-теорий относящихся к методологии и языку описания систем, назвал «Методология структурного анализа и проектирования» (SADT). Исходная работа над SADT началась в 1969 г.

Первое ее крупное приложение было реализовано в 1973 г. при разработ-

ке большого аэрокосмического проекта, когда она была несколько пересмотрена сотрудниками SofTech, Inc. В 1974 г. SADT была еще улучшена и передана одной из крупнейших европейских телефонных компаний.

Таким образом, к началу 70-х SADT представляет собой четкую формальную процедуру.

Появление SADT на рынке произошло в 1975 г. после годичного оформления в виде продукта.

К 1981 г. SADT уже использовали более чем в 50 компаниях при работе более чем над 200 проектами, включавшими более 2000 людей и охватывавшими дюжину проблемных областей, в том числе телефонные сети, аэрокосмическое производство, управление и контроль, учет материально-технических ресурсов и обработку данных.

Ее широкое распространение в настоящее время в европейской, дальневосточной и американской аэрокосмической промышленности (под названием IDEF0) позволяет эти цифры существенно увеличить. Таким образом, SADT выделяется среди современных методологий описания систем благодаря своему широкому применению. Почему SADT имеет такое широкое применение?

Во-первых, SADT является единственной методологией, легко отражающей такие системные характеристики, как управление, обратная связь и исполнители. Это объясняется тем, что SADT изначально возникла на базе проектирования систем более общего вида в отличие от других структурных методов, «выросших» из проектирования программного обеспечения.

Во-вторых, SADT в дополнение к существовавшим в то время концепциям и стандартам для создания систем имела развитые процедуры поддержки коллективной работы и обладала преимуществом, связанным с ее применением на ранних стадиях создания системы.

Кроме того, широкое использование SADT показало, что ее можно сочетать с другими структурными методами. Это достигается использованием графических SADT-описаний в качестве схем, связывающих воедино раз-

личные методы, примененные для описания определенных частей системы с различным уровнем детализации.

Таким образом, неадекватные спецификации систем того времени вызвали создание графического языка SADT, а его усиленное использование преобразовало SADT в законченную методологию, способную повысить качество продуктов, создаваемых на ранних стадиях развития проекта.

Итак, SADT началась как язык описания функционирования систем общего вида, а по мере применения ее процедуры описания систем были улучшены и дополнены.

Основы SADT

SADT использует **два типа диаграмм**: 1) модели деятельности (activity models); 2) модели данных (data models).

SADT использует **стрелки** для построения этих диаграмм и имеет следующее графическое представление:

- главный блок (box), где определено названиее процесса или действия;
- с левой стороны блока – входящие стрелки: входы действия;
- сверху – входящие стрелки: данные, необходимые для действия;
- внизу – входящие стрелки: средства, используемые для действия;
- справа – исходящие стрелки: выход действия.

SADT использует **декомпозицию** на основе подхода «сверху вниз». Каждый уровень декомпозиции содержит до 6 блоков.

SADT **начинается с уровня** (level) **0**, затем может быть детализирован на более низкие уровни (1, 2, 3, ...). Например, на уровне 1, блок уровня 0 будет детализирован на несколько элементарных блоков и так далее …

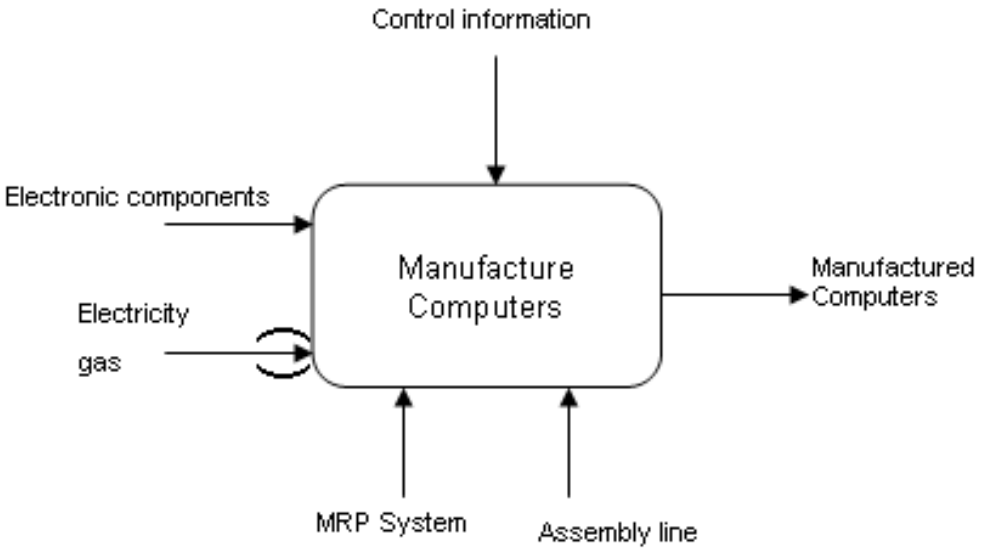

Рис. 4.1. Пример уровня 0

На уровне 1 действие «Manufacture computers», может быть разбито (declined), например на 4 блока:

1) получить электронные компоненты («receive electronic components»);
2) сохранить электронные компоненты («store electronic components»);
3) доставить электронные компоненты на сборочную линию («bring electronic components to the assembly line»);
4) собрать компьютеры («Assemble computers»).

Семантика стрелок для действий (activities):

- входы (Inputs) входят слева и представляют данные или предметы потребления (consumables), нужные действию (that are needed by the activity);
- выходы (Outputs) выходят справа и представляют данные или продукты, производимые действием (activity);
- управления (Controls) входят сверху и представляют команды, которые влияют на исполнение действия, но не потребляются. В последней редакции IDEF0 – условия, требуемые для получения корректного выхода. Данные или объекты, моделируемые как управления, могут быть трансформированы функцией, создающей выход;

- механизмы означают средства, компоненты или инструменты, используемые для выполнения действия; представляют размещение (allocation) действий.

Семантика стрелок для данных (data):

- входы (Inputs) – это действия, которые генерируют эти данные (are activities that produce the data);
- выходы (Outputs) потребляют эти данные (consume the data);
- управления (Controls) влияют на внутреннее состояние этих данных (influence the internal state of the data).

Роли SADT-процесса:

- авторы (Authors) – разработчики SADT модели;
- комментаторы (Commenters) – рецензируют (review) работу авторов;
- читатели (Readers) – возможные (the eventual) пользователи SADT диаграмм;
- эксперты (Experts) – те, от кого авторы получают специальную информацию о требованиях и ограничениях;
- технический комитет (Technical committee) – технический персонал, ответственный за рецензирование (reviewing) SADT модели на каждом уровне;
- библиотекарь проекта (Project librarian) – ответственный за все документы проекта;
- менеджер проекта (Project manager) – имеет полную техническую ответственность за системный анализ и проектирование (has overall technical responsibility the system analysis and design);
- аналитик (Monitor) (Chief analyst) – эксперт в области SADT, помогающий и консультирующий персонал проекта по использованию SADT;
- инструктор (Instructor) – обучает авторов и комментаторов SADT.

Этапы моделирования. Разработка SADT модели представляет собой итеративный процесс и состоит из нижеследующих условных этапов.

1. **Создание модели** группой специалистов, относящихся к различным сферам деятельности предприятия. На этом этапе авторы опрашивают компетентных лиц, получая ответы на следующие вопросы:
   - что поступает в предметную область на «входе»;
   - какие функции и в какой последовательности выполняются в рамках предметной области;
   - кто является ответственным за выполнение каждой из функций;
   - чем руководствуется исполнитель при выполнении каждой из функций;
   - что является результатом работы объекта (на выходе)?

   На основе полученных результатов опросов создается *черновик модели* (Model Draft).
2. **Распространение черновика** для рассмотрения, получения комментариев и согласования модели с *читателями*. При этом каждая из диаграмм черновика письменно критикуется и комментируется, а затем передается автору. Автор, в свою очередь, также письменно соглашается с критикой или отвергает ее с изложением логики принятия решения и вновь возвращает откорректированный черновик для дальнейшего рассмотрения. Этот цикл продолжается до тех пор, пока авторы и читатели не придут к единому мнению.
3. **Официальное утверждение модели.** Утверждение согласованной модели происходит руководителем рабочей группы в том случае, если у *авторов* модели и *читателей* отсутствуют разногласия по поводу ее адекватности. *Окончательная модель* представляет собой согласованное представление о системе с заданной точки зрения и для заданной цели.

Метод SADT получил дальнейшее развитие. На его основе в 1981 году разработана известная методология функционального моделирования IDEF0.

Методология функционального моделирования IDEF0

Основные понятия IDEF0

IDEF0 (Integration Definition for Function Modeling) – методология функционального моделирования для описания функций предприятия, предлагающая язык функционального моделирования для анализа, разработки, реинжиниринга и интеграции информационных систем бизнес процессов; или анализа инженерии разработки ПО (or software engineering analysis).

Модель IDEF0 – это графическое описание (информационной) системы или предметной области (subject), которое разрабатывается **с определенной целью с выбранной точки зрения**. Модель IDEF0 представляет собой набор из одной или более (иерархически связанных) **IDEF0-диаграмм**, которые описывают **функции системы** или предметной области (subject area) с помощью **графики**, **текста** и **глоссария**.

История IDEF0

Методология IDEF0 является развитием метода структурного анализа и проектирования SADT (Structured Analysis and Design Technique).

IDEF0 как стандарт был разработан в 1981 году в рамках программы ICAM (Integrated Computer Aided Manufacturing – интегрированная компьютерная поддержка производства).

IDEF0 – Integration DEFinition language 0 – основан на SADT и в своей исходной форме включает одновременно:

1) определение языка графического моделирования (синтаксис и семантику);
2) описание полной (comprehensive) методологии разработки моделей.

Последняя редакция IDEF0 была выпущена в декабре 1993 года Национальным Институтом по Стандартам и Технологиям США (NIST).

В 1993 году IDEF0 была принята в качестве федерального стандарта в США, а в 2000 году – в качестве стандарта в Российской Федерации.

Применение IDEF0

IDEF0 используется для создания *функциональной модели*, то есть результатом применения методологии IDEF0 к системе есть функциональная модель IDEF0. *Функциональная модель* – это структурное представление функций, деятельности или процессов в пределах моделируемой системы или предметной области.

Методология IDEF0 может быть использована для моделирования широкого спектра как автоматизированных, так и неавтоматизированных систем.

Для проектируемых систем IDEF0 может быть использована сначала для определения требований и функций, и затем для реализации, удовлетворяющей этим требованиям и исполняющей эти функции.

Для существующих систем IDEF0 может быть использована для анализа функций, выполняемых системой, а также для учета механизмов, с помощью которых эти функции выполняются.

Цели стандарта IDEF0

Основные цели (objectives) стандарта:

1) задокументировать и разъяснить технику моделирования IDEF0 и правила ее использования;
2) обеспечить средства для полного и единообразного (consistently) моделирования функций системы или предметной области, а также данных и объектов, которые связывают эти функции;
3) обеспечить язык моделирования, который независим от CASE методов или средств, но может быть использован при помощи этих методов и средств;

4) обеспечить язык моделирования, который имеет следующие характеристики:
   - *общий* (generic) – для анализа как (информационных) систем, так и предметных областей;
   - *строгий и точный* (rigorous and precise) – для создания корректных, пригодных к использованию моделей;
   - *краткий* (concise) – для облегчения понимания, коммуникации, согласия между заинтересованными лицами и проверки. (to facilitate understanding, communication, consensus and validation);
   - *абстрактный* (conceptual) – для представления функциональных требований, независимых от физических или организационных реализаций;
   - *гибкий* – для поддержки различных фаз жизненного цикла проекта.

Строгость и точность (Rigor and Precision). Правила IDEF0 требуют достаточной *строгости и точности* для удовлетворения нужд аналитика без чрезмерных ограничений (to satisfy needs without overly constraining the analyst). IDEF0 правила включают следующее:

- управление детализацией (control of the details communicated at each level) – **от трех до шести** функциональных блоков на каждом уровне декомпозиции;
- связанный контекст (Bounded Context) – не должно быть недостающийх или лишних, выходящих за установленные рамки деталей;
- связанность интерфейса диаграмм (Diagram Interface Connectivity) – наличие номеров узлов, функциональных блоков, C-номеров (C-numbers) и подробных ссылочных выражений (Detail Reference Expression);
- связанность структуры данных. (Data Structure Connectivity) – коды ICOM и использование круглых скобок (ICOM codes and the use of parentheses);

- уникальные метки и заголовки (Unique Labels and Titles) – отсутствие повторяющихся названий в метках и заголовках;
- синтаксические правила для графики (Syntax Rules for Graphics) – функциональные блоки и стрелки;
- ограничения на разветвления стрелок данных (Data Arrow Branch Constraint) – метки для ограничений потоков данных на разветвлениях;
- разделение данных на Вход и Управление (Input versus Control Separation) – правило для определения роли данных);
- маркировка стрелок данных. Data Arrow Label Requirements (minimum labeling rules);
- наличие Управления (Minimum Control of Function) – все функции должны иметь минимум одно Управление;
- цель и точка зрения (Purpose and Viewpoint) – все модели имеют формулировку цели и точки зрения.

Ограничения сложности. Обычно IDEF0-модели несут в себе сложную и концентрированную информацию, и для того, чтобы ограничить их перегруженность и сделать удобочитаемыми, в стандарте приняты соответствующие ограничения сложности, которые носят рекомендательный характер. При том, что, что на диаграмме рекомендуется представлять **от трех до шести** функциональных блоков, количество подходящих к одному функциональному блоку (выходящих из одного функционального блока) интерфейсных дуг предполагается **не более четырех**.

Основные понятия IDEF0

В основе методологии лежат четыре основных понятия:

- функциональный блок;
- интерфейсная дуга;

- декомпозиция;
- глоссарий.

Функциональный блок (Activity Box) представляет собой некоторую конкретную функцию в рамках рассматриваемой системы.

По требованиям стандарта название каждого функционального блока должно быть сформулировано **в глагольном наклонении** (например, «производить услуги»).

На диаграмме функциональный блок изображается *прямоугольником* (рис.). Каждая из четырех сторон функционального блока имеет свое определенное значение (роль), при этом:

- верхняя сторона имеет значение «Управление» (Control);
- левая сторона имеет значение «Вход» (Input);
- правая сторона имеет значение «Выход» (Output);
- нижняя сторона имеет значение «Механизм» (Mechanism).

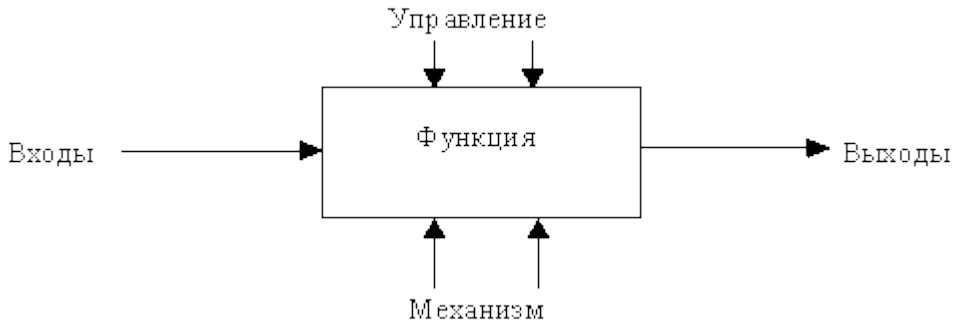

Рис. 4.2. Функциональный блок

Интерфейсная дуга/стрелка (Arrow) отображает элемент системы, который обрабатывается функциональным блоком или оказывает иное влияние на функцию, представленную данным функциональным блоком. Интерфейсные дуги часто называют потоками или стрелками.

С помощью интерфейсных дуг отображают различные объекты, в той или иной степени определяющие процессы, происходящие в системе. Такими объ-

ектами могут быть элементы реального мира (детали, вагоны, сотрудники и т.д.) или потоки данных и информации (документы, данные, инструкции и т.д.).

В зависимости от того, к какой из сторон функционального блока подходит данная интерфейсная дуга, она носит название «**входящей**», «**исходящей**» или «**управляющей**».

Необходимо отметить, что любой функциональный блок по требованиям стандарта должен иметь, по крайней мере, **одну управляющую интерфейсную дугу и одну исходящую**. Это и понятно – каждый процесс должен происходить по каким-то правилам (отображаемым управляющей дугой) и должен выдавать некоторый результат (выходящая дуга), иначе его рассмотрение не имеет никакого смысла.

Обязательное наличие *управляющих интерфейсных дуг* является одним из главных отличий стандарта IDEF0 от других методологий классов DFD (Data Flow Diagram) и WFD (Work Flow Diagram).

Механизмы показывают средства, с помощью которых осуществляется выполнение функций. Механизм может быть человеком, компьютером или любым другим устройством, которое помогает выполнять данную функцию (рис.).

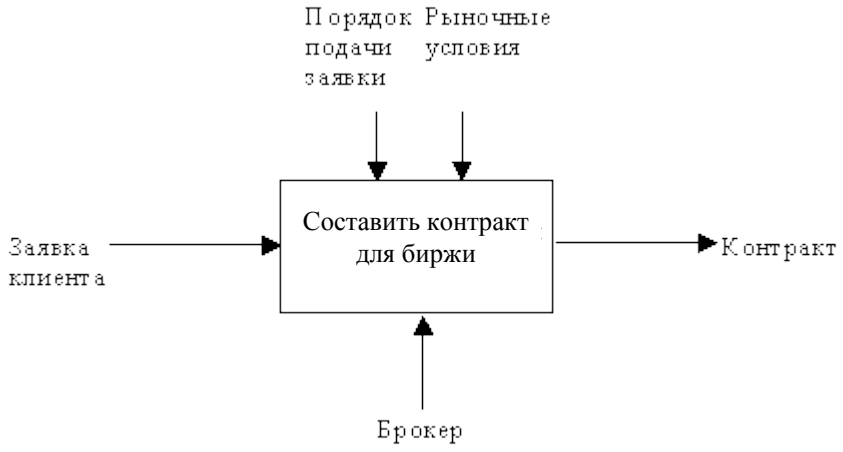

Рис. 4.3. Пример механизма

По типу связываемых элементов интерфейсные дуги (стрелки) подразделяют на *граничные*, *стрелки вызова*, *внутренние стрелки*.

Граничные (border) **стрелки** на *контекстной диаграмме* применяются для описания взаимодействия системы с окружающим миром. Они могут начинаться у границы диаграммы и заканчиваться у функции, или наоборот. На обычной (не контекстной) диаграмме граничные стрелки представляют входы, управления, выходы или механизмы родительского блока диаграммы. Внесенные граничные стрелки как бы указывают на выход за пределы страницы на диаграмме декомпозиции нижнего уровня, и для указания их отличия от других стрелок диаграммы, стрелка таких стрелок изображается в квадратных скобках. Граничные стрелки обеспечивают правильное соединение диаграмм для получения согласованной модели состоящей из отдельных диаграмм. Т.о. эти стрелки являются интерфейсом между диаграммой и остальной частью модели, поэтому необходимо, чтобы все внешние стрелки диаграммы были согласованы со стрелками, образующими границу этой диаграммы. Такое согласование стрелок обеспечивает совмещение данной диаграммы со своей родительской диаграммой, т.е. это означает, что внешние стрелки согласованы по числу и наименованию (но не обязательно по расположению) со стрелками, касающимися декомпозированного блока родительской диаграммы. Источник или приемник граничных стрелок можно обнаружить, только изучая родительскую диаграмму. Все граничные стрелки на дочерней диаграмме (за исключением стрелок, помещенных в туннель) должны соответствовать стрелкам родительского блока.

Стрелки вызова (Call) –- специальная стрелка, указывающая на другую модель работы. **Стрелка вызова** используется для указания того, что некоторая функция выполняется за пределами моделируемой системы, т.е. обозначают обращение из данной модели или из данной части модели к блоку, входящему в состав другой модели или другой части модели, обеспечивая их связь. **Наличие стрелок вызова** указывает на то, что разные модели или разные части одной и той же модели могут совместно использовать один и тот же элемент (блок). Стрелки вызова используются при слиянии и разделении моделей.

Внутренние стрелки – это стрелки IDEF0-диаграммы, характеризующие

четыре основных отношения, концы которой связывают источник и потребителя, являющиеся блоками одной диаграммы. Внутренние стрелки не касаются границы диаграммы и не выходят за ее пределы, а начинаются у одного и кончаются у другого блока.

В IDEFO различают пять типов связей для внутренних стрелок по их **направленности:**

- **связь «выход-вход»** (output-input) является простейшей связью, поскольку она отражает прямые воздействия, которые интуитивно понятны и очень просты. Связь возникает тогда, когда выход одного блока становится входом для другого. Стрелка выхода вышестоящей работы (далее – просто выход) направляется на вход нижестоящей (блока с меньшим доминированием). Связь по «выход-вход» показывает доминирование вышестоящей работы, т.е. выход блока становится входом для блока с меньшим доминированием. Данные или объекты выхода вышестоящей работы не меняются в нижестоящей;
- **связь «выход-управление»** (output-control), является простейшей связью, поскольку она отражает прямые воздействия, которые интуитивно понятны и очень просты. При такой связи выход вышестоящей работы направляется непосредственно на управление нижестоящей, таким образом показывая доминирование вышестоящей функции. Данные или объекты выхода вышестоящей функции не меняются в нижестоящей;
- обратная связь «выход-вход» (output-input feedback) является более сложной, поскольку представляют итерацию или рекурсию (выходы из одной функции влияют на будущее выполнение других функций, что впоследствии влияет на исходную функцию). Такая связь, как правило, используется для описания циклов и часто называется связью по потоку данных. В такой связи выход нижестоящей работы направляется на вход вышестоящей (блока с большим доминированием). Обратные связи могут выступать в виде комментариев, замечаний, исправлений и т.д. (рис.).

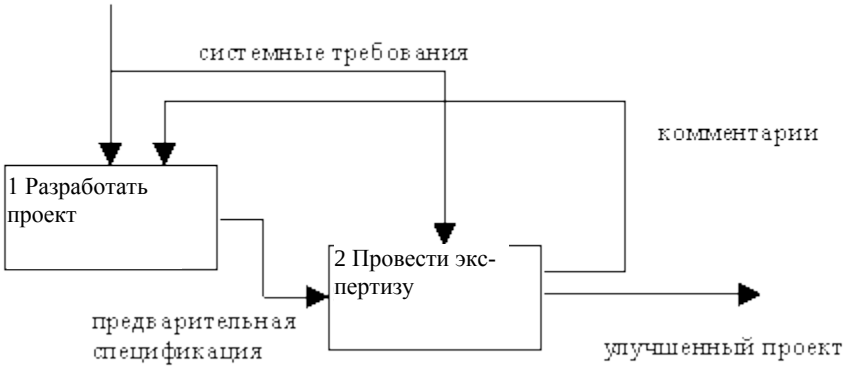

Рис. 4.4. Пример обратной связи

- **обратная связь «выход-управление»** (output-control feedback) является более сложной, поскольку представляют итерацию или рекурсию (выходы из одной функции влияют на будущее выполнение других функций, что впоследствии влияет на исходную функцию). Обратная связь возникает, когда выход некоторой функции А воздействует на выход функции В, а выход функции В воздействует на другую активацию функции А. Обратная связь по управлению возникает тогда, когда выход некоторого блока (как результат деятельности некоторой функции) влияет на блок с большим доминированием, т.е. выход нижестоящей работы направляется на управление вышестоящей. Обратная связь по управлению часто свидетельствует об эффективности процесса;
- **связь «выход-механизм»** (output-mechanism), когда выход одной работы направляется на механизм другой. Эта взаимосвязь встречаются нечасто, и используется реже остальных. Она показывает, что одна работа подготавливает ресурсы, необходимые для проведения другой работы – связь отражает ситуацию, при которой выход одной функции становится средством достижения цели (подготавливает ресурсы), необходимые для проведения другой функции для другой. Обычно связи «выход-механизм» характерны при распределении источников ресурсов (например, требуемые инструменты, обученный персонал, физическое пространство, оборудование, финансирование, материалы).

Одним из важных моментов при проектировании ИС с помощью методологии IDEF0 является точная согласованность типов внутренних связей по их **характеру** (табл. 4.1.).

Таблица 4.1.

Типы и связей и их значимость

Тип связи	**Относительная значимость**
Случайная	0
Логическая	1
Временная	2
Процедурная	3
Коммуникационная	4
Последовательная	5
Функциональная	6

Ниже даны краткие описания каждого из типов связей.

(0) Тип случайной связности: наименее желательный. Случайная связность возникает, когда конкретная связь между функциями мала или полностью отсутствует. Это относится к ситуации, когда имена данных на IDEF0-дугах в одной диаграмме имеют малую связь друг с другом. Крайний вариант этого случая показан на рис. 4.5.

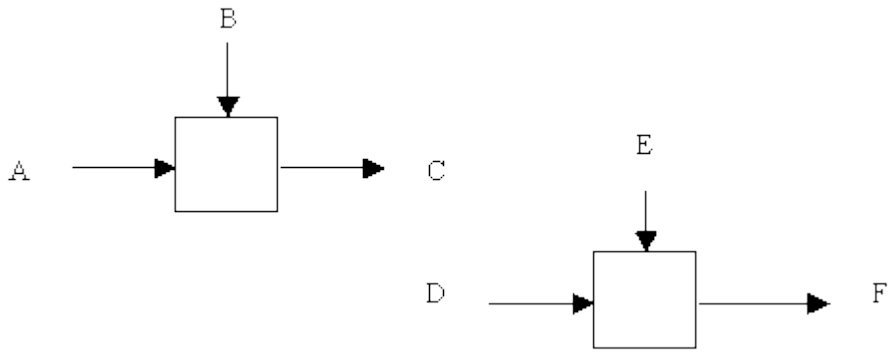

Рис. 4.5. Случайная связность

(1) Тип логической связности. Логическое связывание происходит тогда, когда данные и функции собираются вместе вследствие того, что они попа-

дают в общий класс или набор элементов, но необходимых функциональных отношений между ними не обнаруживается.

(2) Тип временной связности. Связанные по времени элементы возникают вследствие того, что они представляют функции, связанные во времени, когда данные используются одновременно или функции включаются параллельно, а не последовательно.

(3) Тип процедурной связности. Процедурно-связанные элементы появляются сгруппированными вместе вследствие того, что они выполняются в течение одной и той же части цикла или процесса. Пример процедурно-связанной диаграммы приведен на рис. 4.6.

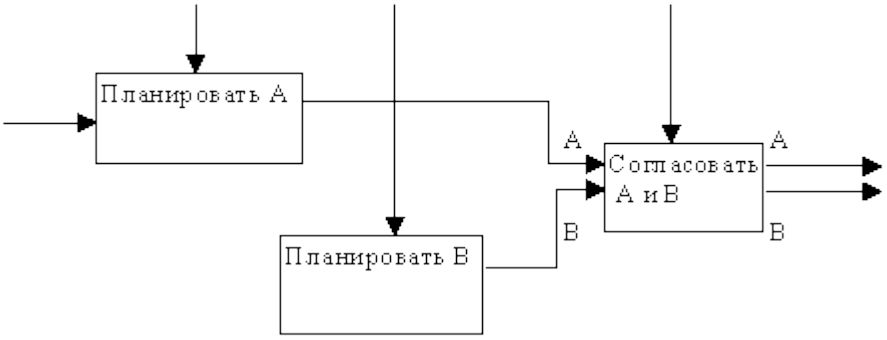

Рис. 4.6. Процедурная связность

(4) Тип коммуникационной связности. Диаграммы демонстрируют коммуникационные связи, когда блоки группируются вследствие того, что они используют одни и те же входные данные и/или производят одни и те же выходные данные (рис. 4.7.).

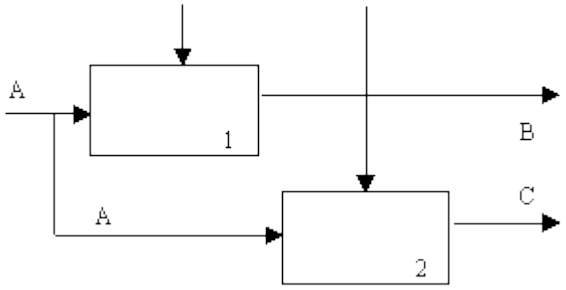

Рис. 4.7. Коммуникационная связность

(5) Тип последовательной связности. На диаграммах, имеющих после-

довательные связи, выход одной функции служит входными данными для следующей функции. Связь между элементами на диаграмме является более тесной, чем на рассмотренных выше уровнях связок, поскольку моделируются причинно-следственные зависимости (рис. 4.8.).

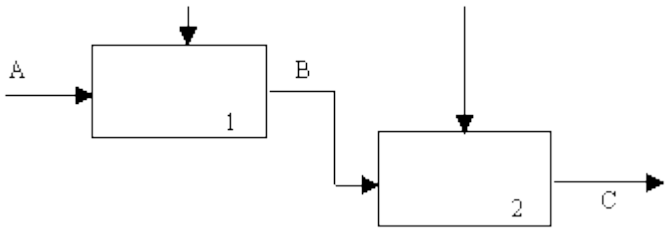

Рис. 4.8. Последовательная связность

(6) Тип функциональной связности. Диаграмма отражает полную функциональную связность, при наличии полной зависимости одной функции от другой. Диаграмма, которая является чисто функциональной, не содержит чужеродных элементов, относящихся к последовательному или более слабому типу связности. Одним из способов определения функционально-связанных диаграмм является рассмотрение двух блоков, связанных через управляющие дуги, как показано на рис. 4.9.

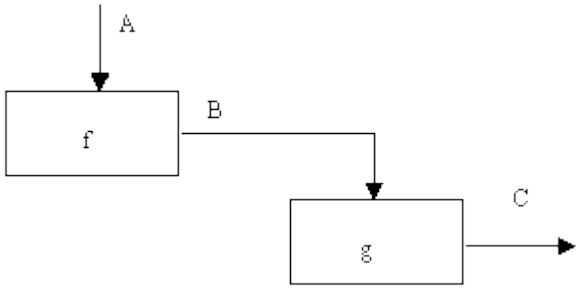

Рис. 4.9. Функциональная связность

В математических терминах необходимое условие для простейшего типа функциональной связности, показанной на рис., имеет следующий вид: C = g(B) = g (f(A))

В таблице представлены все типы связей, рассмотренные выше. Важно отметить, что **уровни 4 – 6** устанавливают типы связностей, которые разработ-

чики считают важнейшими для получения диаграмм хорошего качества.

Таблица 4.2.

Типы связностей

Значимость	Тип связности	Для функций	Для данных
0	Случайная	Случайная	Случайная
1	Логическая	Функции одного и того же множества или типа (например, «редактировать все входы»)	Данные одного и того же множества или типа
2	Временная	Функции одного и того же периода времени (например, «операции инициализации»)	Данные, используемые в каком-либо временном интервале
3	Процедурная	Функции, работающие в одной и той же фазе или итерации (например, «первый проход компилятора»)	Данные, используемые во время одной и той же фазы или итерации
4	Коммуникационнная	Функции, использующие одни и те же данные	Данные, на которые воздействует одна и та же деятельность
5	Последовательная	Функции, выполняющие последовательные преобразования одних и тех же данных	Данные, преобразуемые последовательными функциями
6	Функциональная	Функции, объединяемые для выполнения одной функции	Данные, связанные с одной функцией

Ветвление и слияние стрелок. IDEF является мощным языком описания систем во многом благодаря возможности декомпозиции стрелок. Стрелка в IDEF0 обычно изображают набор объектов (данных), поэтому они могут иметь множество начальных точек (источников) и конечных точек (назначений), поэтому стрелки могут разветвляться и соединяться разными сложными способами. Вся стрелка или ее часть может выходить из одного или нескольких блоков и заканчиваться в одном или нескольких блоках. Таким образом, одни и те же данные, порожденные одной функцией, могут использоваться или в одной, или в нескольких других функциях одновременно. С другой стороны данные, порожденные в различных функциях, могут представлять собой одинаковые или однородные данные/объекты, которые в дальнейшем используются или обрабатываются в одном месте.

Для моделирования таких ситуаций в IDEF0 используются разветвляющиеся и соединяющиеся стрелки, которые отражают иерархию объектов, представленных этими стрелками. Разветвления стрелок и их слияние, тот самый синтаксис, который позволяет описывать декомпозицию содержимого стрелок и уменьшает загруженность диаграмм графическими элементами (линиями). Однако стрелки описывают только ту иерархию, которая связывает отдельные функции системы, представленные на диаграммах.

На самом деле, из отдельной диаграммы редко можно понять полную иерархию стрелки. Обычно это требует чтения большей части модели, а иногда из-за выбранной точки зрения подробности отдельных стрелок не раскрываются совсем. Вот почему IDEF предусматривает дополнительное описание полной иерархии объектов системы посредством формирования глоссария для каждой диаграммы модели и объединения этих глоссариев в словарь стрелок.

Таким образом, словарь стрелок, важное дополнение модели, становится основным хранилищем полной иерархии объектов системы.

Чтобы стрелки и их сегменты правильно описывали связи между блоками-источниками и блоками-приемниками, используются *метки* для каждой ветви стрелок.

Для описания представления разветвлений и соединений стрелок разработаны специальные *соглашения* относительно представления, описания и правила маркирования разветвлений и соединений таких стрелок.

Ветвление стрелки – это разделение ее на два или большее число сегментов, изображенных в виде расходящихся линий, и означающее, что все содержимое стрелок или его часть может появиться в каждом ответвлении стрелки. Стрелка всегда помечается до разветвления, чтобы дать название всему набору. Кроме того, каждая ветвь стрелки может быть помечена или не помечена в соответствии со следующими *правилами*:

- если стрелка именована до разветвления, а после разветвления ни одна из ветвей не именована, то подразумевается, что каждая ветвь моделирует те

же данные или объекты, что и ветвь до разветвления, т.е. все объекты, указанные в метке стрелки перед ветвлением принадлежат этим ветвям (каждому из сегментов);

- если стрелка именована до разветвления, а после разветвления только некоторые из ветвей именованы, это означает, что маркированные ветви соответствуют своему именованию (каждая метка ветви уточняет, что именно содержит ветвь), а неименованные ветви моделируют те же данные или объекты, что и ветвь до разветвления;
- недопустима ситуация, когда стрелка до разветвления не именована, а после разветвления не именована какая-либо из ветвей.

Иногда функция разделяет стрелку на ее **компоненты** в этом случае для получения дополнительных сведений о содержании компонент и взаимосвязях между ними важно изучить, что выполняет эта функция.

Слияние стрелок – это объединение двух или большего числа стрелок, изображенных в виде сходящихся вместе линий и означающее, что содержимое каждой ветви идет на формирование метки для стрелки, являющейся результатом слияния исходных.

Правила именования сливающихся стрелок полностью аналогичны правилам ветвления.

Соглашения по слиянию и разъединению стрелок:

- объединение стрелок возможно только в случае, если их источники не указаны на диаграмме или они представляют одни и те же данные. Такое вычерчивание позволяет абсолютно точно указать единый источник сходных данных;
- объединять стрелки с общим источником или с общим приемником можно только в том случае, если они представляют связанные данные. В этом случае общее название лучше описывает суть данных;

- стрелки связываются (сливаются), если они представляют сходные данные и их источник не указан на диаграмме.

После слияния *результирующая стрелка* всегда помечается для указания нового набора объектов, возникшего после объединения. Кроме того, *каждая ветвь* перед слиянием может помечаться или не помечаться в соответствии со следующими правилами:

- непомеченные ветви содержат все объекты, указанные в общей метке стрелки после слияния;
- если результирующая ветка помечена, то это означает, что ветви, помеченные перед слиянием, содержат все или некоторые объекты, перечисленные в общей метке после слияния Если у результирующей ветки метка отсутствует, это означает, что общая ветка передает все объекты, принадлежащие сливаемым веткам;
- ошибкой является наличие на диаграмме стрелки, которая после слияния не именована, а до слияния не именована какая-либо из ее ветвей.

В методологии IDEF0 существует **соглашения по размещению стрелок**:

- вычерчивание стрелок, независимо от их назначения, выполняется только **по вертикали** и горизонтали, что позволяет проследить за направлением стрелок и определить блоки как точки сбора стрелок, которыми блоки и являются;
- следует обеспечить **максимальное расстояние** между блоками и поворотами стрелок, а также между блоками и пересечениями стрелок для облегчения чтения диаграммы. Одновременно уменьшается вероятность перепутать две разные стрелки;
- следует максимально увеличивать расстояние между входящими или выходящими стрелками на одной грани блока;
- следует максимально увеличить расстояние между поворотами и пересечениями стрелок;

- **наличие входных стрелок** у блока не является его обязательным атрибутом;
- если две стрелки проходят параллельно (начинаются из одной и той же грани одной функции и заканчиваются на одной и той же грани другой функции), то по возможности следует их объединить и назвать единым термином;
- **любой блок** обязательно должен иметь стрелки управления, наличие которых гарантирует работоспособность функции и обеспечивает наложение ограничений и включение/выключение функций системы;
- если данные служат и для управления, и для входа, то вычерчиваются только стрелки управления, что уменьшает сложность диаграммы и делает очевидным управляющий характер данных;
- расстояние между **параллельными стрелками** должно быть максимально возможным (с учетом габаритов диаграммы), что позволяет оставить больше места для меток и помогает зрительно определять количество стрелок и прослеживать их пути;
- расстояние между блоками и поворотами стрелок должно быть максимально возможным (с учетом габаритов диаграммы), что позволяет облегчить процесс чтения и уменьшить вероятность перепутать две разные стрелки;
- **циклические обратные связи** для одного и того же блока выполняются только для того, чтобы более четко подчеркнуть значение повторно используемого объекта. Обычно обратную связь изображают на диаграмме, декомпозирующей блок. Однако иногда требуется выделить повторно используемые объекты;
- **обратные связи «выход-управление»** вычерчиваются «вверх и над» («верхняя» петля), что позволяет указать ограничивающие обратные связи при минимальном числе линий и пересечений, а также собрать все стрелки управления в правой верхней части диаграммы;

- **обратные связи «выход-вход»** вычерчиваются «вниз и под» («нижняя» петля), что позволяет указать обратные потоки данных при минимальном числе линий и пересечений, а также собрать все входные стрелки в левой нижней части диаграммы;
- **обратные связи «выход-механизм»** должны быть показаны как «вниз и под»;
- **количество необязательных пересечений** стрелок при соединении большого числа блоков, число петель и поворотов каждой стрелки, должны быть минимальным, что позволяет значительно уменьшить сложность диаграммы. Это ручная и, в случае насыщенных диаграмм, творческая функция;
- иногда разрешается **выделять буферы** и повторно используемые объекты;
- **слияние стрелок** производится, если они имеют общий источник или приемник, или если они представляют связанные данные. В таком случае общее название лучше описывает суть данных;
- следует **минимизировать** число стрелок, касающихся каждой стороны блока, если, конечно, природа данных не слишком разнородна;
- если возможно, стрелки присоединяются к блокам в одной и той же ICOM-позиции. Тогда соединение стрелок конкретного типа с блоками будет согласованным, и чтение диаграммы упростится;
- при соединении большого числа блоков необходимо избегать необязательных пересечений стрелок. Следует минимизировать число петель и поворотов каждой стрелки;
- **блоки** (функции) являются сопряженными через среду, если они имеют связи с источником, генерирующим данные, без конкретного определения отношения отдельной части данных к какому-либо блоку;
- две или более функций являются сопряженными через запись, если они связаны с набором данных и не обязательно зависят от того, представлены ли все возможные интерфейсы как сопряжение через среду. Тип ин-

терфейса, показанный на последнем рисунке, предпочтителен, поскольку определяются отношения конкретных элементов данных к каждому блоку;

- необходимо использовать (где это целесообразно) выразительные возможности **ветвящихся** стрелок;
- при наличии стрелок со сложной топологией целесообразно повторить метку для удобства ее идентификации.

Туннелирование. При описании сложных систем для их корректного и подробного представления используется большое число стрелок.

Стрелки, входящие в блок и выходящие из него на диаграмме верхнего уровня, являются точно теми же самыми, что и дуги, входящие в диаграмму нижнего уровня и выходящие из нее, потому что блок и диаграмма представляют одну и ту же часть системы.

Часто стрелки могут быть объединены (слиты), но встречаются случаи, когда при описании новых деталей появляются достаточно важные объекты системы, не показанные ранее на более высоких уровнях иерархии модели. Однако, эти детали не столь важны, чтобы их показывать на более высоких уровнях модели. Например, стрелку, изображающую «деталь» на входе в функциональный блок «Обработать на токарном станке» не имеет смысла отражать на диаграммах более высоких уровней – это будет только перегружать диаграммы и делать их сложными для восприятия.

Достаточно часто встречается другой случай, когда отдельные интерфейсные стрелки не стоит рассматривать в дочерних диаграммах ниже определенного уровня – это будет только перегружать их и делать сложными для восприятия. Возникает необходимость избавиться от отдельных «концептуальных» интерфейсных стрелок и не детализировать их глубже некоторого уровня.

Для решения подобных задач в стандарте IDEF0 предусмотрен механизм *туннелирования*.

В большинстве случаев туннелирование применяется для изображения

малозначимых стрелок. В принципе любая стрелка на диаграмме, независимо от своего содержания, может быть «помещена в туннель». Туннельная стрелка – стрелка (со специальной нотацией), не удовлетворяющая обычному требованию, согласно которому каждая стрелка на дочерней диаграмме должна соответствовать стрелкам на родительской диаграмме.

Символ «**туннеля**» (Arrow Tunnel) в виде двух круглых скобок **вокруг начала интерфейсной дуги** обозначает, что эта дуга не была унаследована от функционального родительского блока и появилась (**из «туннеля»**) только на этой диаграмме.

В свою очередь, такое же обозначение **вокруг конца (стрелки) интерфейсной дуги** в непосредственной близи от блока–приемника означает тот факт, что *в дочерней по отношению к этому блоку диаграмме эта дуга отображаться и рассматриваться не будет*.

Чаще всего бывает, что отдельные объекты и соответствующие им интерфейсные дуги *не рассматриваются на некоторых промежуточных уровнях иерархии*, – в таком случае они сначала **«погружаются в туннель»**, а затем при необходимости **«возвращаются из туннеля»**. Стрелки с заключенными в скобки концами выполняют эти задачи, поскольку они не рассматриваются как часть границы при касании ими блока и, следовательно, не переносятся на диаграмму, декомпозирующую этот блок.

Туннелирование рекомендуется применять только опытным аналитикам в случаях, когда диаграммы становятся слишком сложными для чтения и понимания. Помещение стрелок в туннель осуществляется не просто для удобства – это важный способ точной регулировки модели не столько для описания системы, как для существенного упрощает ее описания. Задание стрелке статуса «*туннельная*» позволяет аналитику избежать хаотического заполнения нежелательными подробностями диаграмм высокого уровня, а наличие у туннельной стрелки идентифицирующего ее обозначения дает возможность управлять появлением необходимых деталей, не запутывая общие описания родительских диаграмм. Кроме того, «туннельные» обозначения помогают скрывать сведе-

ния, необходимые только для верхних уровней модели, что минимизирует вероятность загромождения диаграмм-декомпозиций необязательной информацией.

Виды туннельных стрелок:

- **со скрытыми приемниками**. Стрелка имеет скрытый приемник, если она касается блока, но не появляется на диаграмме, которая его декомпозирует (дочерняя). Если стрелка мигрирует с верхнего уровня на нижний, причем на нижнем уровне данные используются одинаково во всех функциях без исключения, т.е. предполагается, что не нужно детализировать стрелку, т.к. стрелка на дочерней функции именована до разветвления, а после разветвления ветви не имеют собственного имени. В этом случае стрелка на нижнем уровне может быть удалена, после чего на родительской диаграмме она может быть туннелирована, а в комментарии к стрелке или в словаре можно указать, что данные будут использоваться во всех функциях дочерних диаграммах. Такое туннелирование называется «*не-в-дочерней-функции*». Туннельные стрелки, имеющие скрытый приемник, кончаются скобками, чтобы отразить тот факт, что такая стрелка идет к какой-то другой части модели, либо непосредственно выходит из модели, или что она не будет более в этой модели рассматриваться;
- **со скрытыми источниками**. Стрелка имеет скрытый источник, если она является внешней стрелкой, отсутствующей на родительской диаграмме. Данный тип туннельной стрелки применяется, если на какой-либо диаграмме нижнего уровня возникает необходимость изо-бразить малозначимые данные или объекты, которые не обрабатываются/используются функцией на текущем уровне, то они направляются на вышестоящий уровень (на родительскую диаграмму). В свою очередь, если эти данные на используются на родительской диаграмме, то они направляются еще выше, и т. д. В результате малозначимая стрелка будет присутствовать на всех уровнях, а это затруднит чтение всех диаграмм, на которых она содержится. В этом случае организуется туннелирование стрелки на самом

нижнем уровне, которое получило название «не-в-родительской-диаграмме». Туннельные стрелки от скрытого источника имеют скобки вокруг конца интерфейсной стрелки в непосредственной близи от блока-приемника. Такая нотация означает, что на родительской диаграмме стрелка не отображается и рассматриваться не будет, а данные не определяются/описываются в исходной диаграмме и отсутствуют на родительской диаграмме. Заключение в скобки начала проходящих через туннель стрелок, появляющихся из неизвестного источника, указывает на то, что эти стрелки появляются из некоторой другой части модели или непосредственно извне.

Опыт свидетельствует, что ситуации требующие создания туннельных стрелок встречаются редко, и если это все же происходит, то следует применять туннелирование с большой осторожностью. При неправильном использовании туннельные стрелки быстро становятся прикрытием плохого моделирования. Поэтому рекомендуется сначала проводить стрелки сквозь границы блоков, а затем определять, в каких случаях это снижает качество описания. Только после этого можно помещать стрелки в туннели для улучшения читабельности модели.

ICOM. В IDEF0 принята система маркирования, позволяющая аналитику точно идентифицировать и проверять связи по стрелкам между диаграммами. Эта схема кодирования стрелок получила название **ICOM –** название по первым буквам английских эквивалентов слов вход (Input), управление (Control), выход (Output), механизм (Mechanism).

ICOM – это коды, предназначенные для идентификации граничных стрелок, содержат префикс, соответствующий типу стрелки (I, C, O или M), и порядковый номер. Буква следует перед числом, определяющим относительное положение точки подключения стрелки к родительскому блоку; это положение определяется слева направо или сверху вниз.

Основной ICOM-код – это узловой номер, который появляется там, где выполняется декомпозиция функционального блока и создается его подробное описание на дочерней диаграмме. Все остальные ссылочные коды базируются на узловых номерах. Например, код «С3», написанный возле граничной стрелки на дочерней диаграмме, указывает, что эта стрелка соответствует третьей (слева) управляющей стрелке родительского блока.

ICOM-коды представляют собой ссылочные выражения, которые присваиваются всем элементам модели: диаграммам, блокам, стрелкам и примечаниям. Коды используются в различных контекстах для точного указания на нужный элемент модели.

Например, **кодирование** связывает каждую дочернюю диаграмму со своим родительским блоком. Если блоки на дочерней диаграмме подвергаются дальнейшей декомпозиции и подробно описываются на дочерних диаграммах следующего уровня, то на каждую новую диаграмму назначаются новые ICOM-коды, связывающие граничные стрелки этих диаграмм со стрелками их родительских блоков.

Иногда **буквенные ICOM-коды**, определяющие роли граничных стрелок (вход, управление, механизм), могут меняться при переходе от родительского блока к дочерней диаграмме. Например, управляющая стрелка в родительском блоке может быть входом на дочерней диаграмме. Аналогично, вход родительского блока может быть управлением для одного или более дочерних блоков. Коды ICOM позволяют:

- быстро проверять согласованность внешних стрелок диаграммы с граничными стрелками соответствующего блока родительской диаграммы;
- связывать граничные стрелки на дочерней диаграмме со стрелками родительского блока;
- обеспечивать согласованность декомпозиции, поскольку все стрелки, входящие в диаграмму и выходящие из нее, должны быть учтены;
- обеспечивать требуемую строгость;

- создавать совокупность неявных связующих звеньев между страницами, которые можно быстро изменить при изменении границ;
- облегчать процесс чтения и рецензирования IDEF0-диаграмм;
- облегчать проверку согласованности декомпозиции;
- упрощать внесение локальных изменений в диаграмму;
- объединять различные варианты диаграмм в модели.

В моделях сбои часто происходят в *точках интерфейса*. Для IDEF-диаграмм **интерфейсными** являются места соединения диаграмм со своими родителями, именно поэтому каждую декомпозицию необходимо аккуратно соединять со своим родителем, используя ICOM-метки.

При построении диаграммы с меньшей доминантностью стрелки, касающиеся декомпозируемого блока, используются в качестве источников и приемников для стрелок, которые создаются на новой диаграмме.

После завершения создания диаграммы-потомка для обеспечения согласованности ее внешние стрелки стыкуются с родительской диаграммой. Одним из способов такой стыковки служит присваивание кодов ICOM внешним стрелкам новой диаграммы согласно следующему *алгоритму*:

1) прежде чем составлять список данных, записать имена и коды для всех стрелок, образующих границу. Это поможет при декомпозиции уменьшить вероятность пропуска части граничных стрелок;
2) аналитик должен представить себе рисунок новой диаграммы как бы внутри декомпозируемого блока;
3) зрительно продлить внешние стрелки почти до края диаграммы и зрительно соединить каждую внешнюю стрелку диаграммы с соответствующей граничной стрелкой декомпозируемого блока;
4) присвоить код каждой зрительной связи (I -для входных стрелок, C – для связей между стрелками управления, O – для связей между выходными стрелками, M – для связей между стрелками механизма;

5) добавить после каждой буквы цифру, соответствующую положению данной стрелки среди других стрелок того же типа, касающихся родительского блока. Причем входные и выходные стрелки пересчитываются сверху вниз, а стрелки управлений и механизмов пересчитываются слева направо;
6) записать каждый код около окончания каждой внешней стрелки;
7) для выделения связи внешних стрелок с соответствующими граничными стрелками, границу субъекта изобразить жирной линией;
8) выполнив декомпозицию, вернуться назад к исходному блоку родительской диаграммы и соединить каждую внешнюю стрелку новой диаграммы с соответствующей стрелкой, касающейся этого блока. Это позволит избежать пропуска необходимого соединения.

Декомпозиция (Decomposition) является основным понятием стандарта IDEF0. Принцип декомпозиции применяется при разбиении сложного процесса на составляющие его функции. При этом **уровень детализации** процесса определяется непосредственно разработчиком модели.

Декомпозиция позволяет постепенно и структурированно представлять модель системы в виде иерархической структуры отдельных **диаграмм**, что делает ее менее перегруженной и легко усваиваемой.

Диаграмма – это основная единица описания системы, расположенная на отдельном листе, представляющая функции и интерфейсы в виде блоков и дуг. Место соединения дуги с блоком определяет *тип интерфейса*. Диаграммы являются графическим представлением знаний экспертов о моделируемой системе и предметом обсуждения со специалистами конкретной предметной области. Аналитик представляет знания экспертов, изложенные с помощью естественного языка, используя графическую нотацию IDEF0, давая более наглядное, лаконичное и однозначное описание системы, не жертвуя при этом каче-

ством представления. Таким образом, в ходе разработки диаграмм знания экспертов структурируются, и устраняется неоднозначность описаний за счет:

- стандартизации интерпретации графических обозначений. Так, например, содержание контекстной диаграммы интерпретируется следующим образом: «Процесс преобразует Вход в Выход при выполнении условий, заданных в Управление с помощью Механизма»;
- декомпозиции блоков диаграмм более высокого уровня, когда за счет более точных описаний на диаграммах-потомках устраняется неоднозначность возможных интерпретаций.

Моделирование IDEF0 всегда начинается с представления системы как единого целого – одного функционального блока с интерфейсными дугами, простирающимися за пределы рассматриваемой области. Такая диаграмма с одним функциональным блоком называется **контекстной диаграммой.**

Контекстная диаграмма – это IDEF0-диаграмма, расположенная на вершине иерархии диаграмм, представляющая собой самое общее описание системы, состоит из **одного блока**, описывающего функцию верхнего уровня, **стрелок** и **пояснительного текста**, определяющего **точку зрения** и **цель** моделирования. Эти утверждения помогают руководить разработкой модели и ввести этот процесс в определенные рамки.

Контекстная диаграмма имеет **узловой номер A-n (n = 0)**, которая представляет контекст модели. Диаграмма верхнего уровня обозначается идентификатором «A-0» (произносится «A минус ноль»), на которой объект моделирования представлен единственным **блоком** с **граничными стрелками**, отображающими связь системы с окружающей средой, устанавливает область моделирования, определяет **границы модели** и является обязательной диаграммой **IDEF0-модели.**

Поскольку единственный **блок** представляет весь объект, то его имя – общее для всего проекта. Это же справедливо и для всех стрелок диаграммы,

поскольку они представляют полный комплект внешних интерфейсов объекта.

В пояснительном тексте к контекстной диаграмме должна быть указана **цель** (Purpose) построения диаграммы в виде краткого описания и зафиксирована **точка зрения** (Viewpoint).

Определение и формализация цели разработки IDEF0-модели является крайне важным моментом. Фактически цель определяет соответствующие области в исследуемой системе, на которых необходимо фокусироваться в первую очередь.

Точка зрения. С определением модели тесно связана позиция, с которой наблюдается система и создается ее модель. Поскольку качество описания системы резко снижается, если оно ни на чем не сфокусировано, SADT требует, чтобы модель рассматривалась все время с одной и той же позиции. Эта позиция называется «**точкой зрения**» данной модели.

Точка зрения определяет основное направление развития модели, уровень необходимой детализации, что и в каком разрезе можно увидеть в пределах модели. Четкое фиксирование точки зрения позволяет разгрузить модель, отказавшись от детализации и исследования отдельных элементов, не являющихся необходимыми, исходя из выбранной точки зрения на систему.

Изменение точки зрения приводит к рассмотрению других аспектов объекта. Аспекты, важные с одной точки зрения, могут не появиться в модели, разрабатываемой с другой точки зрения на тот же самый объект.

У модели может быть только одна точка зрения.

Правильный выбор точки зрения существенно сокращает временные затраты на построение конечной модели.

Точка зрения определяет субъект моделирования и что, соответственно, будет в дальнейшем рассматриваться как элементы/компоненты системы, а что – как внешняя среда/воздействие.

Имя функции, записываемое в блоке *контекстной диаграммы*, является общей функцией системы с выбранной **точки зрения** и конкретной **целью** построения модели.

Построение IDEF0-модели начинается с *контекстной диаграммы*, т.е. представления всей системы в виде простейшей компоненты – одного блока и дуг, изображающих интерфейсы с функциями вне системы. Затем блок, который представляет систему в качестве единого модуля, детализируется на другой диаграмме с помощью нескольких блоков, соединенных интерфейсными дугами.

Эти блоки представляют основные подфункции исходной функции. Каждая из этих подфункций может быть декомпозирована подобным образом для более детального представления.

При этом каждая подфункция может содержать только те элементы, которые входят в исходную функцию.

Таким образом, в процессе *декомпозиции* функциональные блоки диаграммы подвергаются детализации на другой диаграмме, которая называется **дочерней** (Child Diagram) по отношению к детализируемому блоку.

Каждый из функциональных блоков, принадлежащих дочерней диаграмме, называется **дочерним блоком** (Child Box).

В свою очередь, функциональный блок-предок называется **родительским блоком** по отношению к дочерней диаграмме (Parent Box), а диаграмма, к которой он принадлежит – **родительской диаграммой** (Parent Diagram).

Каждый блок дочерней диаграммы может быть далее детализирован аналогичным образом, став родительским по отношению с соответствующей диаграмме его детализации (рис. 4.10).

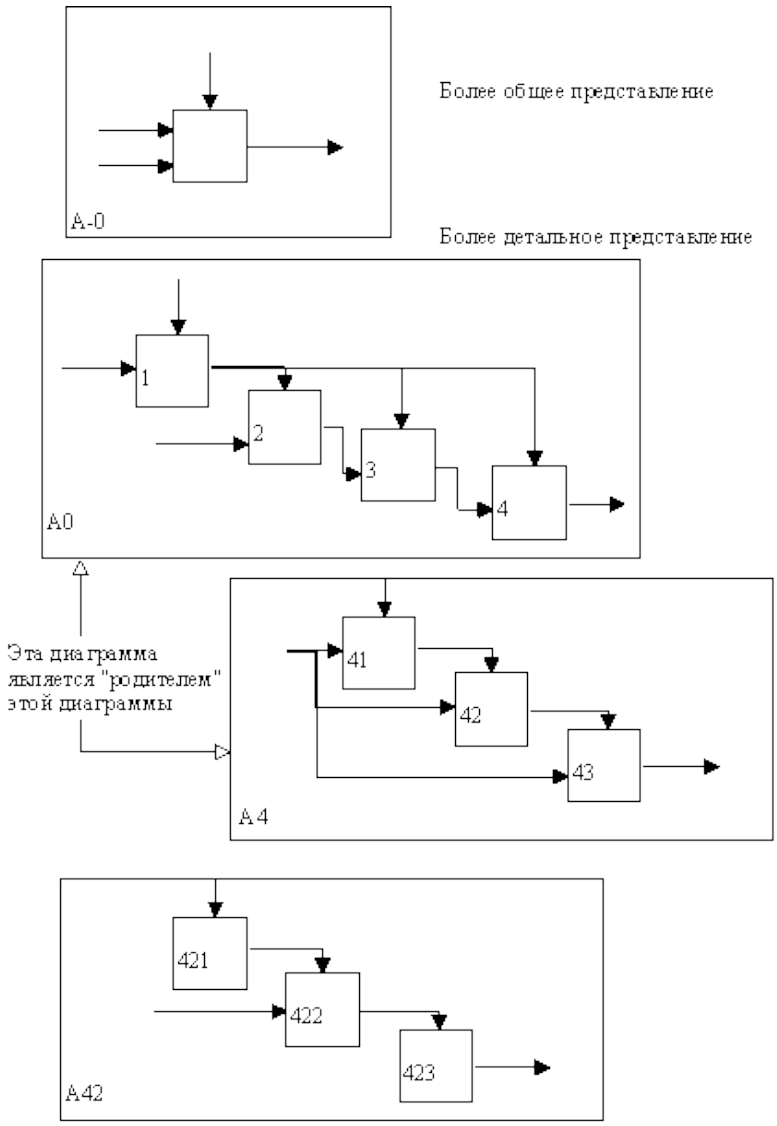

Рис. 4.10 Структура IDEF0-модели. Декомпозиция диаграмм

Согласно стандарту IDEF0 на каждом уровне **декомпозиции** должен использоваться *принцип ограничения сложности*, поэтому в соответствии с этим принципом считается, что единственный блок и несколько стрелок на самом верхнем (контекстном) уровне используются для определения границы всей системы. Соответственно, стрелки, касающиеся этого блока, описывают главные управления, входы, выходы и механизмы этой системы. В дальнейшем, текстовое описание, содержащее основные типы объектов и функции и комментарии экспертов, используется для предварительного создания диаграммы A0.

Дерево диаграмм. Каждый блок на диаграмме имеет свой **номер**. Блок любой диаграммы может быть далее описан диаграммой нижнего уровня, кото-

рая, в свою очередь, может быть далее детализирована с помощью необходимого числа диаграмм. Таким образом, формируется **иерархия диаграмм**.

Для того, чтобы указать положение любой диаграммы или блока в иерархии, используются **номера диаграмм**. Например, A21 является диаграммой, которая детализирует блок 1 на диаграмме A2. Аналогично, A2 детализирует блок 2 на диаграмме A0, которая является самой верхней диаграммой модели. На рис. 4.11 показано типичное дерево диаграмм.

Рис. 4.11. Иерархия диаграмм

Детализируя рассматриваемую систему на этапе сбора и анализа предварительной информации, необходимо обращать внимание на входные и выходные объекты самой системы и составляющих ее подсистем.

Моделирование необходимо начинать с составления описания основных типов объектов и основных функций системы. При этом необходимо учесть нормальные и аномальные ситуации, имеющиеся в системе обратные связи, и возможные случаи потенциальных ошибок.

Глоссарий. Последним из понятий IDEF0 является *глоссарий* (Glossary). Для каждого из элементов IDEF0 – диаграмм, функциональных блоков, интерфейсных дуг – существующий стандарт подразумевает создание и поддержание набора соответствующих определений, ключевых слов, повествовательных изложений и т.д., которые характеризуют объект, отображенный данным элемен-

том.

Этот набор называется **глоссарием** и является описанием сущности данного элемента. Глоссарий гармонично дополняет наглядный графический язык, снабжая диаграммы необходимой дополнительной информацией.

Пример SADT-модели

В примере модель объединяет и организует диаграммы в иерархические структуры, в которых диаграммы наверху модели менее детализированы, чем диаграммы нижних уровней. Другими словами, модель SADT можно представить в виде древовидной структуры диаграмм, где верхняя диаграмма является наиболее общей, а самые нижние наиболее детализированы.

На рис. 4.12 показано определение цели и точки зрения моделирования.

На рис.4.13 – 4.14 представлены две диаграммы из модели экспериментального механического цеха.

Верхняя диаграмма (на вершине модели) описывает механический цех как функцию, в основе которой лежит преобразование входящих рабочих комплектов (заготовок, сырья, документации) в детали при определенном контроле качества.

Нижняя диаграмма детализирует верхнюю, указывая на три главные функции механического цеха: управление выполнением заданий, выполнение задания и контроль качества выполнения.

Таким образом, общая функция, указанная на верхней диаграмме, детализируется с помощью трех функций на нижней диаграмме. Это пример того, как SADT организует описание системы, создавая иерархию добавляющихся на каждом уровне деталей.

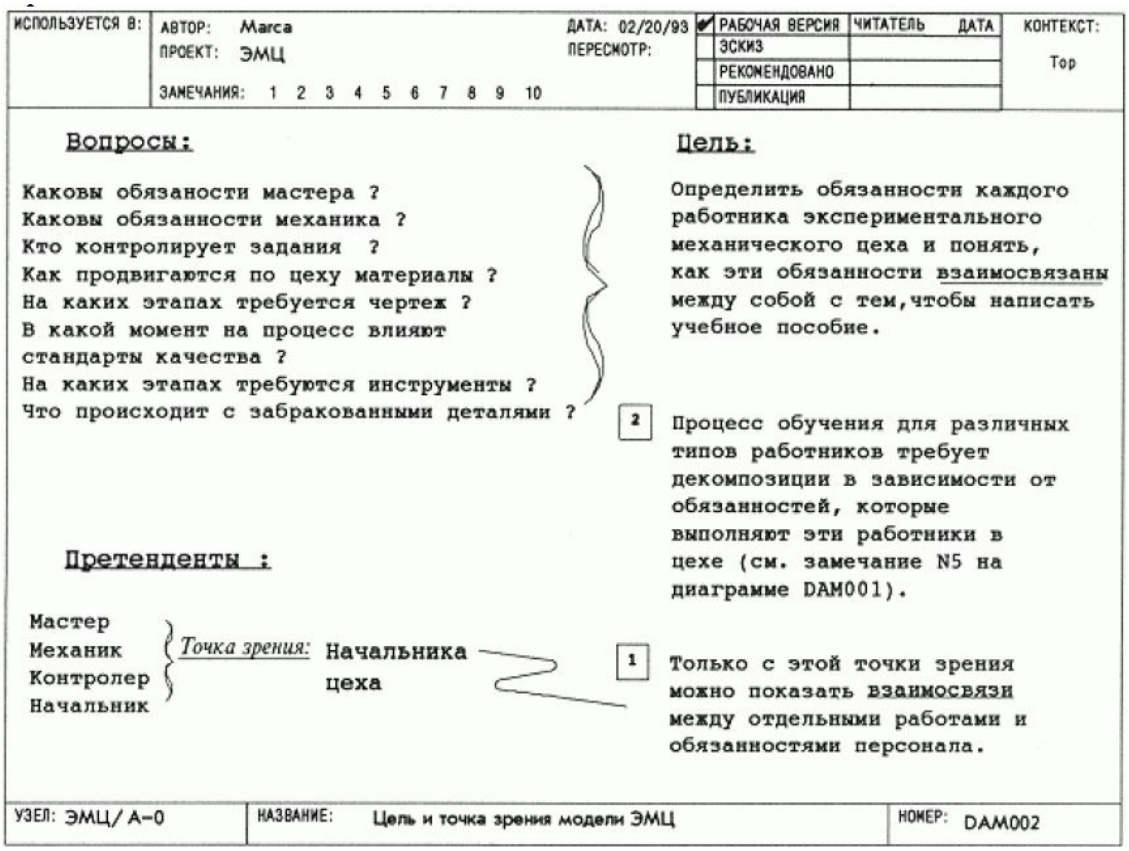

Рис. 4.12. Определение цели и точки зрения

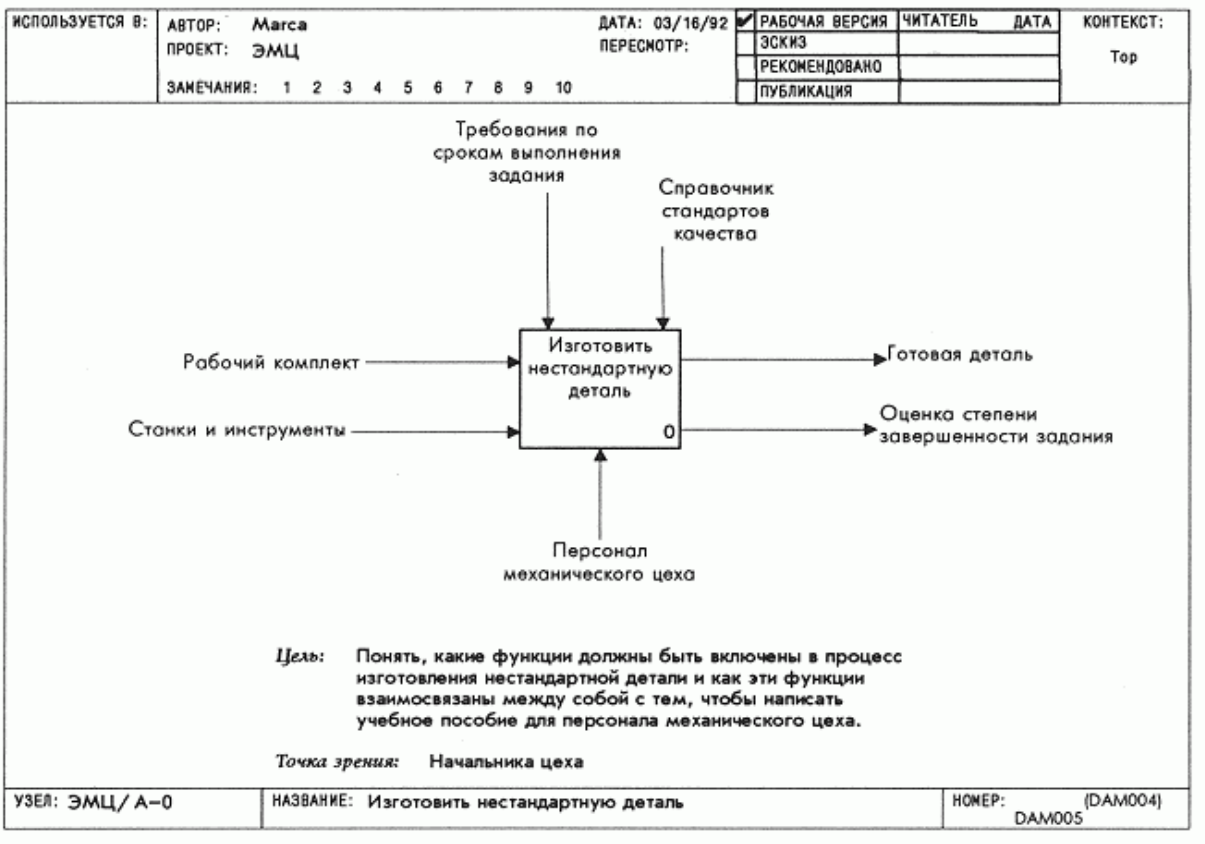

Рис. 4.13. Контекстная диаграмма модели экспериментального механического цеха (ЭМЦ)

На рис. 4.14 показано также взаимное влияние трех функций нижней диаграммы, обозначенное дугами, которые символизируют объекты механического цеха.

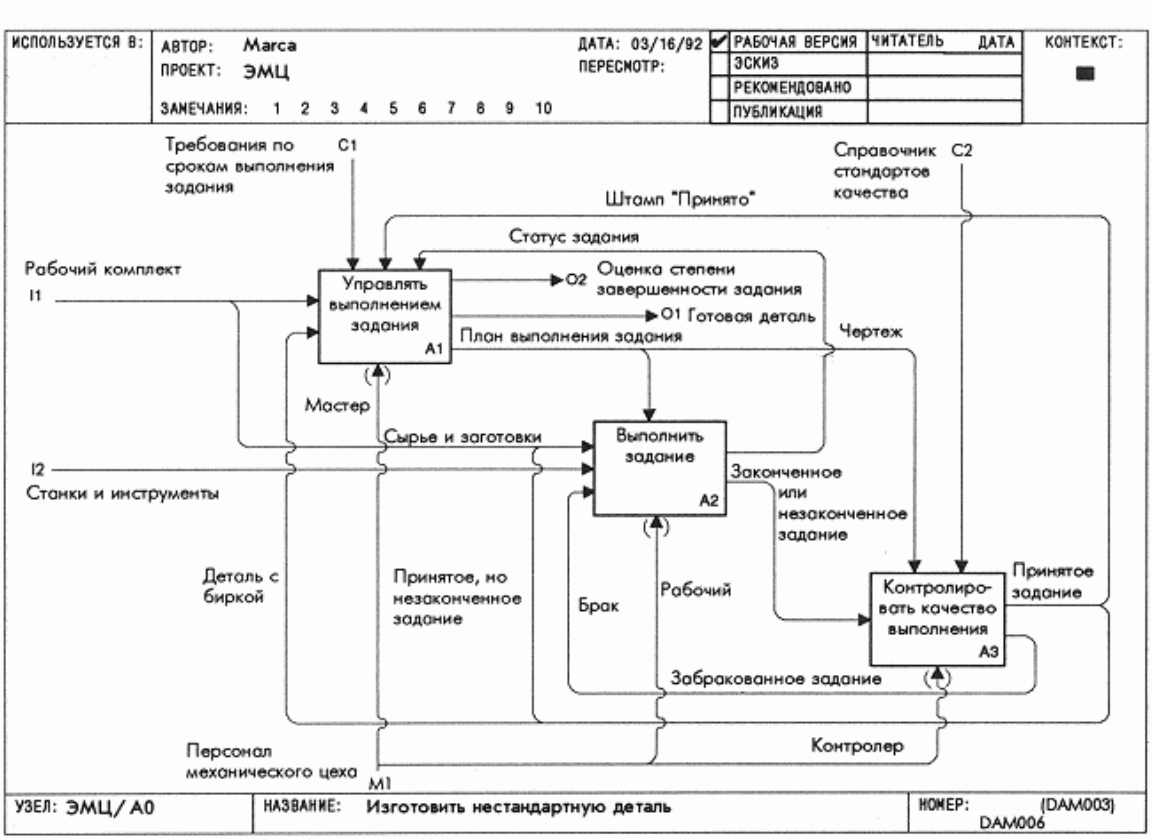

Рис. 4.14. Диаграмма первого уровня модели экспериментального механического цеха (ЭМЦ)

Если внимательно посмотреть на диаграмму, то можно заметить, что некоторые дуги доходят до ее границы, а имена этих дуг совпадают с теми, что указаны на дугах верхней диаграммы. Это пример того, как SADT соединяет диаграммы в модели через объекты системы.

Такая схема соединения требует согласованного наименования и учета объектов системы с тем, чтобы две диаграммы можно было рассматривать как связанные между собой. Например, функциональный блок на верхней диаграмме имеет семь дуг, и каждая из них может быть найдена среди дуг, идущих к границе или от границы диаграммы на следующем уровне.

Блоки SADT никогда не размещаются на диаграмме случайным образом. Они размещаются по степени важности, как ее понимает автор диаграммы. В SADT этот относительный порядок называется **доминированием**. Доминирование понимается как влияние, которое один блок оказывает на другие блоки диаграммы.

Например, самым доминирующим блоком диаграммы может быть либо первый из требуемой последовательности функций, либо планирующая или контролирующая функция, влияющая на все другие функции (такая, как *определить степень выполнения задания* на рис. 4.15).

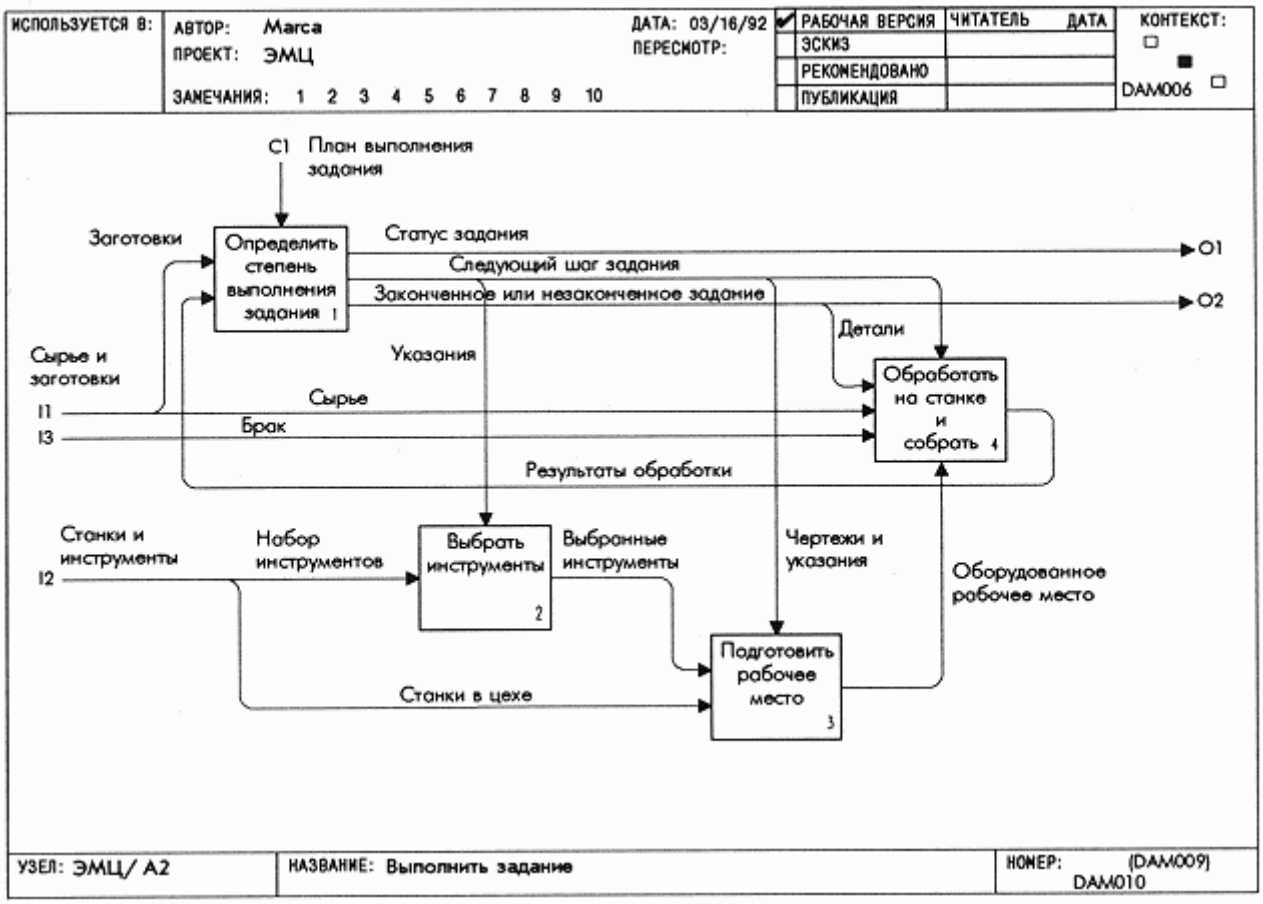

Рис. 4.15. Диаграмма второго уровня модели экспериментального механического цеха (ЭМЦ)

Наиболее доминирующий блок обычно размещается в верхнем левом углу диаграммы, а наименее доминирующий – в правом нижнем углу. В результате получается «ступенчатая» схема, как на рис. 4.15 для блоков 1, 2, 3, где

расположение блоков на странице отражает авторское определение доминирования.

Таким образом, топология диаграммы показывает, какие функции оказывают большее влияние на остальные. Чтобы подчеркнуть это, SADT-аналитик может перенумеровать блоки в соответствии с порядком их доминирования.

Порядок доминирования может обозначаться цифрой, размещенной в правом нижнем углу каждого прямоугольника: 1 будет указывать на наибольшее доминирование, 2 – на следующее после наибольшего и т.д. На рис. 4.15 показано, что блок *определить степень выполнения задания* влияет на все остальные шаги по обработке детали через *следующий шаг задания* и поэтому этот блок пронумерован единицей. Поскольку блок *подготовить рабочее место* должен быть перед блоком *обработать на станке и собрать,* этим блокам присвоены номера 3 и 4.

Документирование полученных знаний

Создание модели (блок 2 на рис. 4-1) – это второй важный этап в процессе моделирования, на котором аналитик документирует полученные им знания о данной проблемной области, представляя их в виде одной или нескольких SADT-диаграмм.

Процесс создания модели осуществляется с помощью специального метода детализации ограниченного субъекта, т.е. автор SADT-модели вначале анализирует объекты, входящие в систему, а затем ис-ользует полученные знания для анализа функций системы.

На основе этого анализа создается диаграмма, в которой объединяются сходные объекты и функции. Этот конкретный путь проведения анализа системы и документирования его результатов является уникальной особенностью методологии SADT.

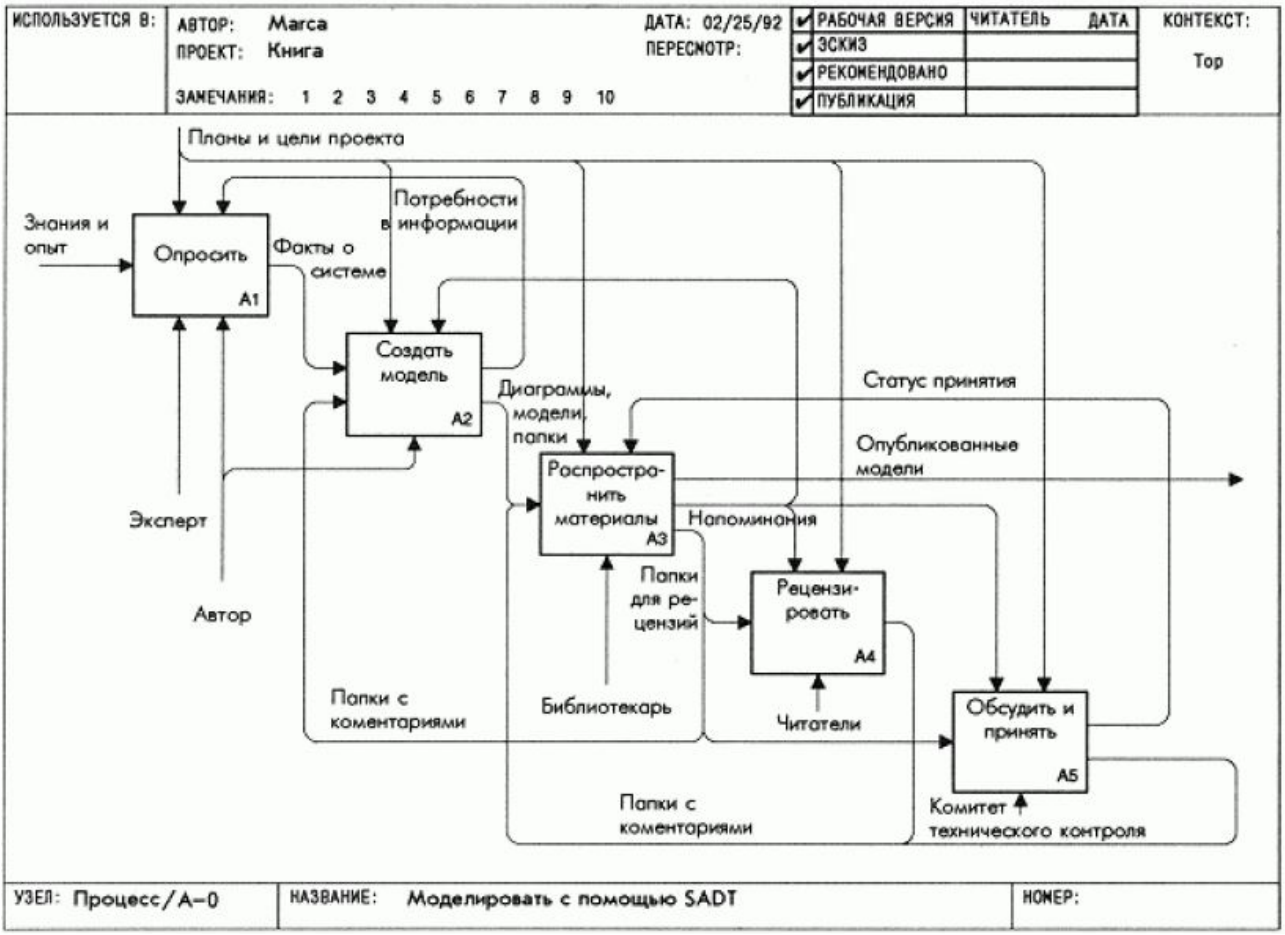

Рис. 4.16. Процесс создания SADT-модели

4.3. Метод структурного системного анализа и проектирования SSADM

Основные понятия SSADM

SSADM (Structured Systems Analysis and Design Method) – системный подход к анализу и проектированию ИС.

SSADM как комплект стандартов для системного анализа и разработки приложений был разработан в начале 1980-х для Центрального агентства по компьютерам и телекоммуникациям (Central Computer and Telecommunications Agency, сейчас это Office of Government Commerce) – государственного учреждения UK, заинтересованного в использовании технологии в управлении.

Позже SSADM широко использовался для государственных ИТ-проектов

в UK, затем нашел широкое применение во всем мире для проектирования ИС.

SSADM использует комбинацию текста и диаграмм для проектирования системы на всем ее жизненном цикле, от идеи до реального физического проекта приложения

SSADM использует комбинацию из трех методологий моделирования.

1. **Логическое моделирование данных (Logical Data Modeling, LDM)** – процесс идентификации, моделирования и документирования требований к разрабатываемой системе. Элементы логической модели данных:
    - *сущности* (entities) – то, о чем фирме нужно записать информацию;
    - *связи* (relationships) – ассоциации между сущностями.
2. **Моделирование потоков данных (Data Flow Modeling)** – процесс идентификации, моделирования и документирования движения данных в ИС. Моделирование потоков данных исследует:
    - *процессы* (processes) – деятельность по преобразованию данных из одной формы в другую;
    - *накопители данных* (data stores) – области (промежуточного) хранения данных (the holding areas for data);
    - *внешние сущности* (external entities) – сущности, которые посылают данные в систему или получают данные из системы;
    - *потоки данных* – маршруты, по которым данные могут двигаться.
3. **Моделирование поведения сущностей (Entity Behavior Modeling)** – процесс идентификации, моделирования и документирования событий, которые влияют на каждую сущность и последовательности, в которой эти события происходят.

Каждая их этих трех моделей системы обеспечивает различные точки зрения на одну и ту же систему, и каждая из точек зрения необходима для формирования полной модели проектируемой системы.

Все три методологии во взаимосвязи друг с другом (are cross-referenced against each other) дают гарантию полноты и точности всего приложения.

Проект разработки SSADM приложения делится на пять модулей, которые в дальнейшем разбиваются на иерархию из стадий, этапов и задач. Модули проекта приведены ниже.

1. **Анализ осуществимости проектного решения** (Feasibility Study) – анализ предметной области для определения сможет ли проектируемая система удовлетворить бизнес-требованиям.
2. **Анализ требований** (Requirements Analysis). На этом этапе определяются подлежащие разработке системные требования и моделируется текущая среда предприятия в терминах процессов с включением структур данных.
3. **Спецификация требований** (Requirements Specification). На этом этапе определяются детальные функциональные и нефункциональные требования и вводятся новые методики для определения необходимых процессов и структур данных.
4. **Логическая системная спецификация** (Logical System Specification). На этом этапе вырабатываются опции технической системы, логический проект обновлений, обработка запросов и системные диалоги.
5. **Физический проект (Physical Design).** На этапе физического проектирования создается физический проект базы данных и комплект программных спецификаций с использованием логической и технической системных спецификаций.

В отличие от RAD (см. гл. 1.5), который подразумевает параллельное выполнение этапов, SSADM строит каждый этап на основе работы, которая была предписана на предыдущем этапе без отклонений от модели.

По причине жесткой структуры методологии, SSADM хороша с точки зрения контроля проектов и способности разрабатывать системы лучшего качества.

Моделирование данных

Уровни представления модели данных

Существует два уровня представления модели данных – *логический* и *физический*.

Логическая модель – это абстрактное представление данных. В логической модели данные представляются и могут называться так, как выглядят и называются в реальном мире, например «Постоянный клиент», «Отдел» или «Фамилия сотрудника». Объекты модели, представляемые на логическом уровне, называются *сущностями* и *атрибутами*. Специфика логической модели:

1) логическая модель данных может быть построена на основе другой логической модели, например на основе модели процессов;
2) логическая модель данных является универсальной и никак не связана с конкретной реализацией СУБД.

Физическая модель данных. В физической модели содержится информация обо всех объектах БД. Поскольку стандартов на объекты БД не существует (например, нет стандарта на типы данных), физическая модель зависит от конкретной реализации СУБД, фактически являясь отображением ее системного каталога. Следовательно, одной и той же логической модели могут соответствовать несколько разных физических моделей. Если в *логической модели* не имеет значения, какой конкретно тип данных имеет атрибут, то в *физической модели* важно описать всю информацию о конкретных физических объектах – таблицах, колонках, индексах, процедурах и т.д. Особенности документирования физической модели заключаются в следующем:

1) многие СУБД имеют ограничение на именование объектов (например, ограничение на длину имени таблицы или запрет использования специальных символов – пробела и т. п.);

2) в нелокализованных версиях СУБД объекты БД могут называться короткими словами, только латинскими символами и без использования специальных символов (т.е. нельзя назвать таблицу, используя предложение – ее можно назвать только одним словом);
3) кроме того, проектировщики БД нередко злоупотребляют «техническими» наименованиями, в результате таблица и колонки получают наименования типа RTD_324 или CUST_A12 и т.д. Полученную в результате структуру могут понять только специалисты (а чаще всего – только авторы модели), ее невозможно обсуждать с экспертами предметной области.

Разделение модели на логическую и физическую позволяет решить эту проблему. На *физическом* уровне объекты БД могут называться так, как того требуют ограничения СУБД. На *логическом* уровне можно этим объектам дать синонимы – имена более понятные неспециалистам, в том числе на кириллице и с использованием специальных символов. Например, таблице CCUST_A12 может соответствовать сущность «Постоянный клиент». Такое соответствие позволяет лучше документировать модель и дает возможность обсуждать структуру данных с экспертами предметной области.

Организация структуры БД формируется исходя из следующих соображений:

1) удобство использования для ведения учёта и анализа данных – на уровне *физической модели*;
2) адекватность описываемому объекту/системе – на уровне *концептуальной и логической модели*; Виды концептуальных и логических моделей БД: *сетевая*; иерархическая; реляционная (ER-модель); многомерная; объектно-ориентированная.

Проектирование модели данных

Существует два типа проектирования модели данных: *прямое* и *обратное*.

Прямое проектирование (Forward Engineering) – процесс генерации физической схемы БД из логической модели, состоящий из следующих этапов:

1) разработка логической модели;
2) выбор СУБД и автоматическое создание физической модели на основе логической;
3) генерация системного каталога СУБД или соответствующего SQL-скрипта на основе физической модели.

ERwin при генерации физической схемы включает триггеры ссылочной целостности, хранимые процедуры, индексы, ограничения и другие возможности, доступные при определении таблиц в выбранной СУБД.

Прямое проектирование обеспечивает *масштабируемость* – создав одну логическую модель данных, можно сгенерировать физические модели под любые СУБД.

Обратное проектирование (Reverse Engineering) – процесс генерации логической модели из физической БД. Обратное проектирование решает задачу позволяет конвертировать БД из одной СУБД в другую. После создания логической модели БД путем обратного проектирования можно переключиться на другой сервер и произвести прямое проектирование.

1) по содержимому системного каталога или SQL-скрипту воссоздается логическая модель данных;
2) на основе полученной логической модели данных генерируется физическая модель для другой СУБД;
3) создается системный каталог этой СУБД.

Обратное проектирование решает задачу по переносу структуры данных с одного сервера на другой.

Создание логической модели данных

Информационное (концептуальное) моделирование

Цель информационного моделирования – обеспечение разработчиков ИС концептуальной схемой базы данных в форме одной модели или нескольких локальных моделей данных, которые относительно легко могут быть отображены в любую систему баз данных.

Концептуальная схема представляет собой карту понятий и отношений между ними, т.е. семантику организации (а не дизайн базы данных) и может существовать на различных уровнях абстракции. Если функциональная схема представляет собой определенную точку зрения на мир и не является гибкой (меняется мир – должна поменяться схема), то концептуальная схема объединяет в себе все точки зрения и дает более абстрактное представление об объекте, описывая фундаментальные понятия, лишь с экземплярами которых человек имеет дело.

В инженерии ПО, моделирование «сущность-связь» (Entity-Relationship Model, ERM) – это пример абстрактного и концептуального представление данных; это метод моделирования БД, используемый для создания концептуальной схемы или семантической модели данных системы (часто реляционной БД), а также требований к системе в форме «сверху-вниз». Диаграммы, созданные таким способом, называются диаграммами «сущность-связь» (entity-relationship diagrams) или ER-диаграммами, или кратко ERD.

ERD является наиболее распространенным средством документирования данных. С их помощью определяются важные для предметной области объекты (*сущности*), их свойства (*атрибуты*) и отношения друг с другом (*связи*). ERD непосредственно используются для проектирования реляционных баз данных. На уровне **физической модели** *сущности* соответствует *таблица*; *экземпляру* сущности – *строка* в таблице; *атрибуту* – *колонка* таблицы.

Различают **три уровня логической модели**, отличающихся по глубине представления информации о данных: *диаграмма сущность-связь*, *модель данных, основанная на ключах* и *полная атрибутивная модель*.

1. **Диаграмма сущность-связь** (Entity Relationship Diagram, ERD) представляет собой модель данных верхнего уровня. Она включает сущности и взаимосвязи, отражающие основные правила предметной области. Такая диаграмма не слишком детализирована, в нее включаются основные сущности и связи между ними, которые удовлетворяют основным требованиям, предъявляемым к ИС. Диаграмма сущность-связь может включать связи «многие-ко-многим» и не включать описание ключей. Как правило, ERD используется для презентаций и обсуждения структуры данных с экспертами предметной области.
2. **Модель данных, основанная на ключах** (Key Based model, KB) – более подробное представление данных. Она включает описание всех сущностей и первичных ключей и предназначена для представления структуры данных и ключей, которые соответствуют предметной области.
3. **Полная атрибутивная модель** (Fully Attributed model, FA) – наиболее детальное представление структуры данных: представляет данные в третьей нормальной форме и включает все сущности, атрибуты и связи.

Существует несколько соглашений для ERD. Классическая нотация связана, главным образом, с концептуальным моделированием. При этом существует ряд нотаций, применяемых для логического и физического проектирования БД, например IDEF1X. Диаграмма структуры данных DSD (data structure diagram) – это модель данных для описания концептуальных моделей данных с помощью графических нотаций, которые документируют сущности и связи между ними, а также условия по ограничению связей (и the constraints that binds them). Диаграммы структуры данных DSD являются расширением E-R-модели (entity-relationship model, E-R model).

Case-метод Баркера

Нотация ERD была впервые введена П. Ченом (Chen) и получила дальнейшее развитие в работах Баркера.

Первый этап моделирования – *выделение сущностей*. **Сущность (Entity)** – реальный либо воображаемый объект, имеющий существенное значение для рассматриваемой предметной области, информация о котором подлежит хранению. Графическое изображение сущности показано на рис. 4.17.

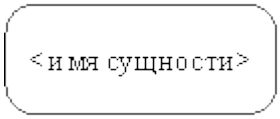

Рис. 4.17. Графическое изображение сущности

Следующим шагом моделирования является *идентификация связей*. Изображение **связей**, их степени и обязательности показано на рис. 4.18.

Рис. 4.18. Графическое изображение связей

Последним шагом моделирования является *идентификация атрибутов*. **Атрибут** – любая характеристика сущности, значимая для рассматриваемой предметной области и предназначенная для квалификации, идентификации, классификации, количественной характеристики или выражения состояния сущности. **Экземпляр атрибута** – это определенная характеристика отдельного элемента множества. Экземпляр атрибута определяется типом характеристики и ее значением, называемым значением атрибута. В ER-модели атрибуты ассоциируются с конкретными сущностями. Таким образом, экземпляр сущности должен обладать единственным определенным значением для ассоциированного атрибута.

Атрибут может быть либо *обязательным*, либо *необязательным* (рис. 4.19). *Обязательность* означает, что атрибут не может принимать неопределенных значений (null values).

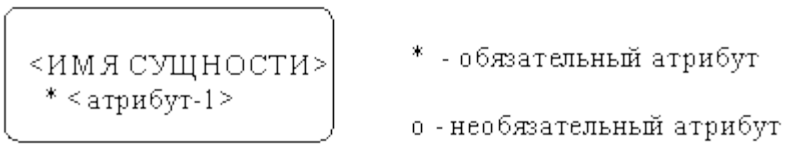

Рис. 4.19. Обязательные и необязательные атрибуты

Атрибут может быть либо *описательным* (т.е. обычным дескриптором сущности), либо входить в состав *уникального идентификатора* (первичного ключа). **Уникальный идентификатор** – это атрибут или совокупность атрибутов и/или связей, предназначенная для уникальной идентификации каждого экземпляра данного типа сущности. В случае полной идентификации каждый экземпляр данного типа сущности полностью идентифицируется своими собственными ключевыми атрибутами, в противном случае в его идентификации участвуют также атрибуты другой сущности-родителя (рис. 4.20).

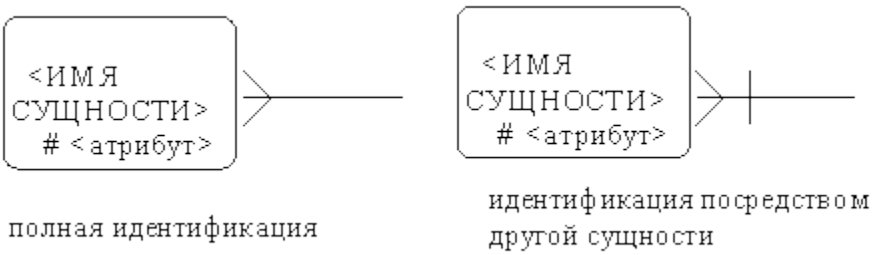

Рис. 4.20. Идентификация атрибутов

Каждый атрибут идентифицируется уникальным именем, выражаемым грамматическим оборотом существительного, описывающим представляемую атрибутом характеристику. Атрибуты изображаются в виде списка имен внутри блока ассоциированной сущности, причем каждый атрибут занимает отдельную строку. Атрибуты, определяющие первичный ключ, размещаются наверху списка и выделяются знаком «#».

Каждая сущность должна обладать хотя бы одним *возможным ключом*. **Возможный ключ сущности** – это один или несколько атрибутов, чьи значения однозначно определяют каждый экземпляр сущности. При существовании нескольких возможных ключей один из них обозначается в качестве первичного ключа, а остальные – как альтернативные ключи.

Помимо перечисленных основных конструкций модель данных может содержать ряд дополнительных.

Подтипы и супертипы: одна сущность является обобщающим понятием для группы подобных сущностей (рис. 4.21).

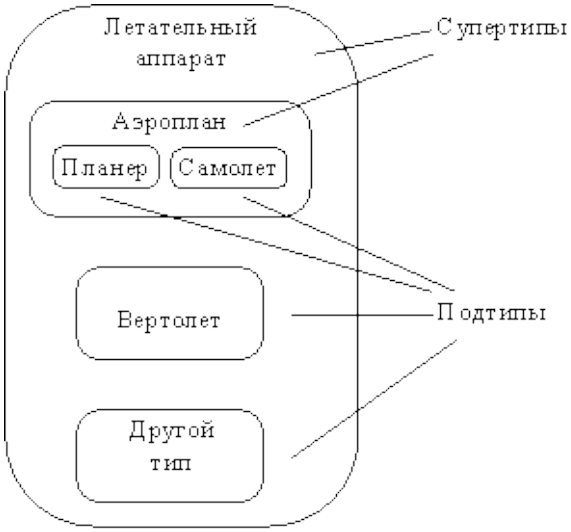

Рис. 4.21. Подтипы и супертипы

Взаимно исключающие связи: каждый экземпляр сущности участвует только в одной связи из группы взаимно исключающих связей (рис. 4.22).

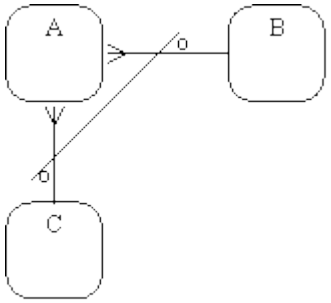

Рис. 4.22. Взаимно исключающие связи

Рекурсивная связь: сущность может быть связана сама с собой (рис. 4.23).

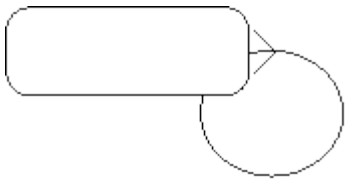

Рис. 4.23. Рекурсивная связь

Неперемещаемые (non-transferrable) связи: экземпляр сущности не может быть перенесен из одного экземпляра связи в другой (рис. 4.24).

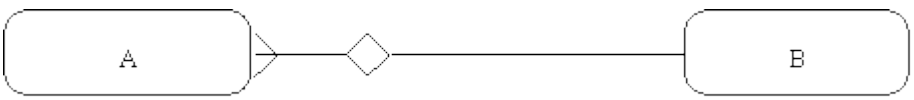

Рис. 4.24. Неперемещаемая связь

Методология IDEF1X

Метод IDEF1, разработанный Т.Рэмей (T.Ramey), также основан на подходе П.Чена и позволяет построить модель данных, эквивалентную реляционной модели в третьей нормальной форме. В настоящее время на основе совершенствования методологии IDEF1 создана ее новая версия – методология IDEF1X. IDEF1X разработана с учетом таких требований, как простота изучения и возможность автоматизации. IDEF1X-диаграммы используются рядом распространенных CASE-средств (в частности, ERwin, Design/IDEF).

Сущность в методологии IDEF1X является *независимой от идентификаторов* или просто независимой, если каждый экземпляр сущности может быть однозначно идентифицирован без определения его отношений с другими сущностями. Сущность называется *зависимой от идентификаторов* или просто зависимой, если однозначная идентификация экземпляра сущности зависит от его отношения к другой сущности (рис. 4.25).

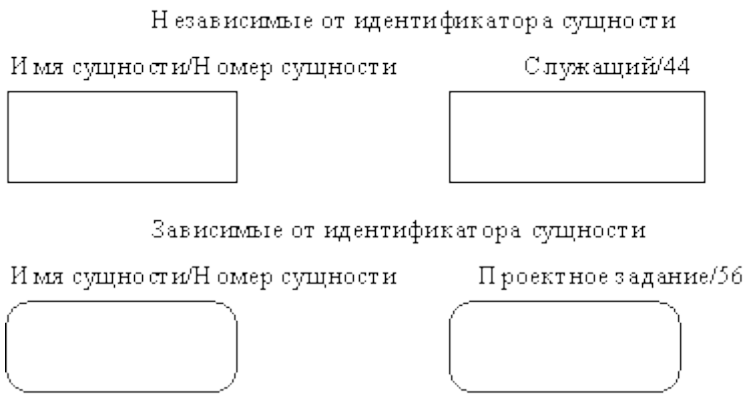

Рис. 4.25. Независимые и зависимые сущности

Каждой сущности присваивается уникальное имя и номер, разделяемые косой чертой "/" и помещаемые над блоком.

Связь является логическим соотношением между сущностями. Каждая связь должна именоваться глаголом или глагольной фразой (рис. 4.26.).

Рис. 4.26. Связь

Связь может дополнительно определяться с помощью указания степени или мощности (количества экземпляров сущности-потомка, которое может существовать для каждого экземпляра сущности-родителя). В IDEF1X могут быть выражены следующие мощности связей:

- каждый экземпляр сущности-родителя может иметь ноль, один или более связанных с ним экземпляров сущности-потомка;
- каждый экземпляр сущности-родителя должен иметь не менее одного связанного с ним экземпляра сущности-потомка;
- каждый экземпляр сущности-родителя должен иметь не более одного связанного с ним экземпляра сущности-потомка;
- каждый экземпляр сущности-родителя связан с некоторым фиксированным числом экземпляров сущности-потомка.

Если экземпляр сущности-потомка однозначно определяется своей связью с сущностью-родителем, то связь называется идентифицирующей, в противном случае – неидентифицирующей.

Связь изображается линией, проводимой между сущностью-родителем и сущностью-потомком с точкой на конце линии у сущности-потомка. Мощность связи обозначается как показано на рис. 4.27 (мощность по умолчанию – N).

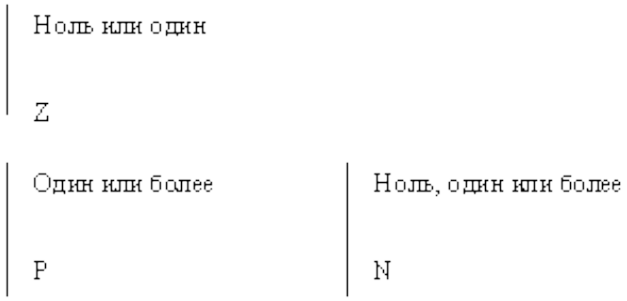

Рис. 4.27. Мощность связи

Идентифицирующая связь между сущностью-родителем и сущностью-потомком изображается сплошной линией (рис. 4.28.).

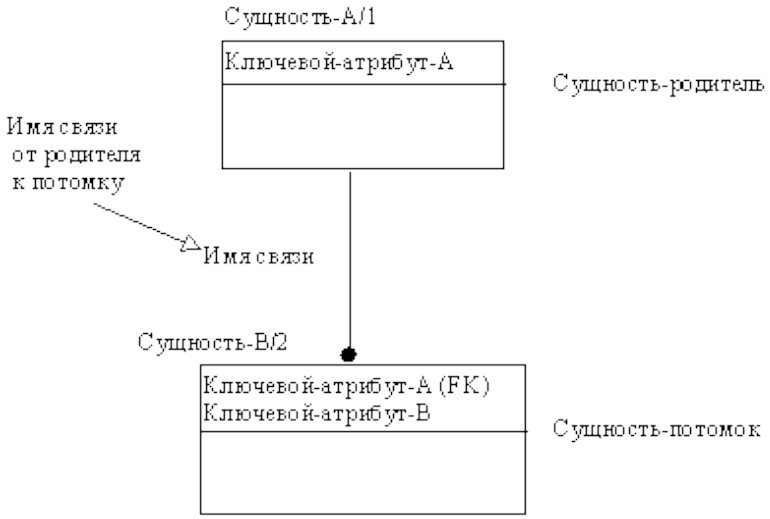

Рис. 4.28. Идентифицирующая связь

Сущность-потомок в идентифицирующей связи является зависимой от идентификатора сущностью. Сущность-родитель в идентифицирующей связи может быть как независимой, так и зависимой от идентификатора сущностью (это определяется ее связями с другими сущностями).

Пунктирная линия изображает *неидентифицирующую связь* (рис. 4.29). Сущность-потомок в неидентифицирующей связи будет независимой от идентификатора, если она не является также сущностью-потомком в какой-либо идентифицирующей связи.

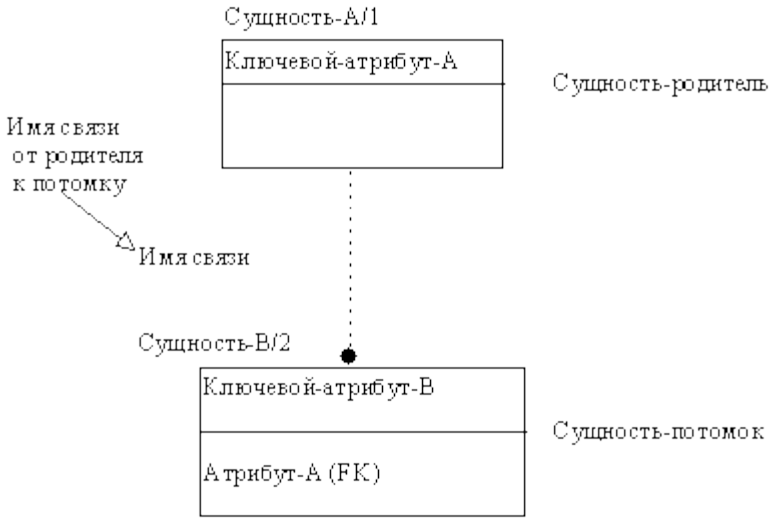

Рис. 4.29. Неидентифицирующая связь

Атрибуты изображаются в виде списка имен внутри блока сущности. Атрибуты, определяющие *первичный ключ*, размещаются наверху списка и отделяются от других атрибутов горизонтальной чертой (рис. 4.30).

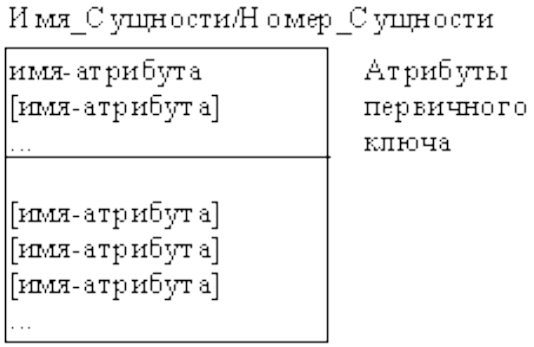

Рис. 4.30. Атрибуты

Сущности могут иметь также *внешние ключи* (Foreign Key), которые могут использоваться в качестве части или целого первичного ключа или неключевого атрибута. *Внешний ключ* изображается с помощью помещения внутрь блока сущности имен атрибутов, после которых следуют буквы FK в скобках (рис. 4.31).

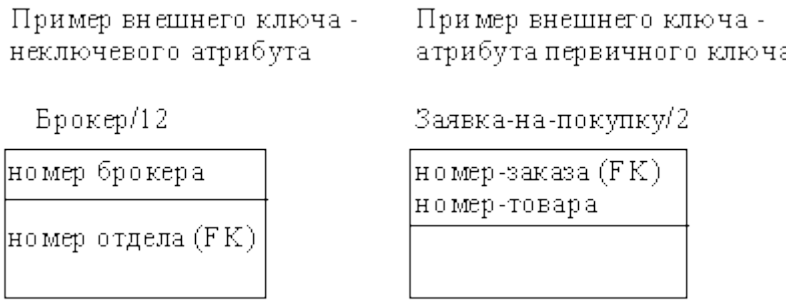

Рис. 4.31. Примеры внешних ключей

Когда рисуется идентифицирующая связь, ERwin автоматически преобразует дочернюю сущность в зависимую. Зависимая сущность изображается прямоугольником со скругленными углами (рис. 4.32).

Рис. 4.32. Изображение зависимой сущности

Экземпляр зависимой сущности определяется только через отношение к родительской сущности. При установлении идентифицирующей связи атрибуты первичного ключа родительской сущности автоматически переносятся в состав первичного ключа дочерней сущности. Эта операция дополнения атрибутов дочерней сущности при создании связи называется миграцией атрибутов. В дочерней сущности новые атрибуты помечаются как внешний ключ – (FK).

В дальнейшем, при генерации схемы БД, атрибуты первичного ключа получат признак NOT NULL, что означает невозможность внесения записи в таблицу заказов без информации о номере клиента.

При установлении неидентифицирующей связи дочерняя сущность остается независимой, а атрибуты первичного ключа родительской сущности мигрируют в состав неключевых компонентов родительской сущности. Неидентифицирующая связь (рис. 4.33) служит для связывания независимых сущностей.

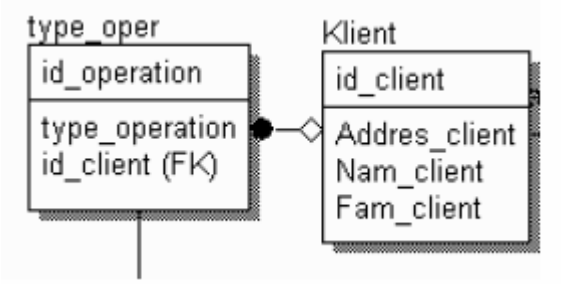

Рис. 4.33 Неидентифицирующая связь

Связь *многие-ко-многим* возможна только на уровне логической модели данных. Такая связь обозначается сплошной линией с двумя точками на концах (рис. 4.34).

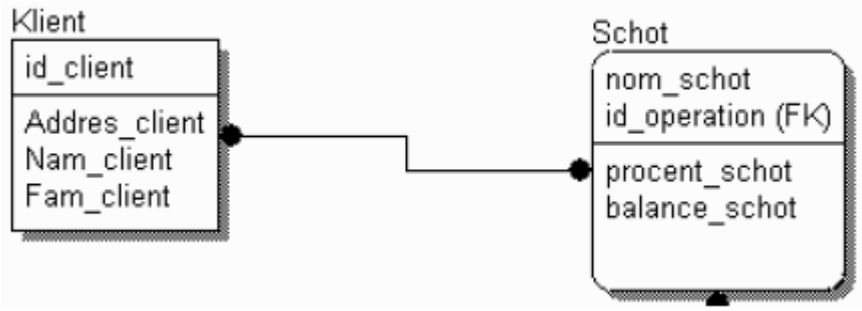

Рис. 4.34. Связь «многие-ко-многим»

Связь многие-ко-многим должна именоваться двумя фразами – в обе стороны. Это облегчает чтение диаграммы.

При переходе к физическому уровню ERwin автоматически преобразует связь многие-ко-многим, добавляя новую таблицу и устанавливая две новые связи один-ко-многим от старых к новой таблице. При этом имя новой таблице автоматически присваивается как «Имя1_Имя2».

Типы сущностей и иерархия наследования

Как уже было сказано, связи определяют, является ли сущность независимой или зависимой. Различают несколько типов зависимых сущностей:

- *характеристическая* – зависимая дочерняя сущность, которая связана только с одной родительской и по смыслу хранит информацию о характеристиках родительской сущности.
- *ассоциативная* – сущность, связанная с несколькими родительскими сущностями. Такая сущность содержит информацию о связях сущностей.
- *именующая* – частный случай ассоциативной сущности, не имеющей собственных атрибутов (только атрибуты родительских сущностей, мигрировавших в качестве внешнего ключа).
- *категориальная* – дочерняя сущность в иерархии наследования.

Иерархия наследования (или иерархия категорий) представляет собой особый тип объединения сущностей, которые разделяют общие характеристики. Обычно иерархию наследования создают, когда несколько сущностей имеют общие по смыслу атрибуты, либо когда сущности имеют общие по смыслу связи, либо когда это диктуется бизнес-правилами.

Для каждой категории можно указать **дискриминатор** – атрибут родового предка, который показывает, как отличить одну категориальную сущность от другой.

Иерархии категорий делятся на два типа – *полные* и *неполные*. В *полной категории* одному экземпляру родового предка обязательно соответствует эк-

земпляр в каком-либо потомке. Если категория еще не выстроена полностью и в родовом предке могут существовать экземпляры, которые не имеют соответствующих экземпляров в потомках, то такая категория будет *неполной*. *Полная категория* помечается кружком с двумя горизонтальными чертами, *неполная* – кружком с одной чертой. Возможна комбинация полной и неполной категорий (рис. 4.35).

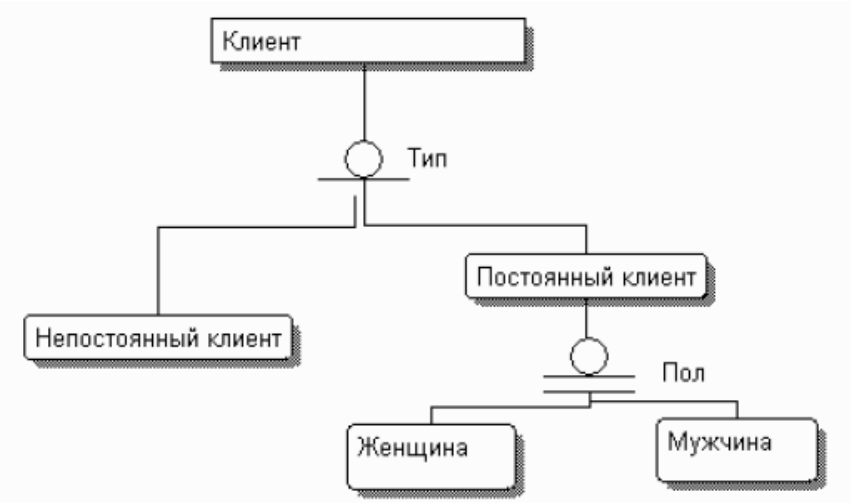

Рис. 4.35. Комбинация полной и неполной категорий

Моделирование потоков данных

Методологию моделирования потоков данных (Data Flow Modeling) используют при анализе ИС – проектируемых или реально существующих. Получаемая в результате модель потоков данных DFDs (Data-flow diagrams) является графическим представлением «потоков данных» через ИС, а также одним из трех основных ракурсов методологии **SSADM**. В соответствии с методологией DFD **модель системы** представляется в виде иерархии диаграмм потоков данных (ДПД, DFD), описывающих асинхронный процесс преобразования информации от ее ввода в систему до выдачи пользователю. **Основные объекты нотации:**

- *работы* (*Activities*) – отображают процессы обработки и изменения информации;
- *стрелки* (*Arrows*) – отображают информационные потоки;

- *хранилища (накопители) данных* (*Data Store*) – отображают данные, к которым осуществляется доступ, эти данные используются, создаются или изменяются работами;
- *внешние сущности* (*External References*) – отображают объекты, с которыми происходит взаимодействие.

Работы (процессы) представляют собой функции системы по преобразованию входных потоков данных в выходные в соответствии с определенным алгоритмом.

Работы DFD соответствуют процессам IDEF0. Так же как IDEF0, DFD отображает входы и выходы процессов, но не показывают стрелки управления и механизма.

Физически процесс может быть реализован различными способами: это может быть подразделение организации (отдел), выполняющее обработку входных документов и выпуск отчетов, программа, аппаратно реализованное логическое устройство и т.д.

Процесс на диаграмме потоков данных изображается прямоугольниками со скругленными углами, как показано на рис. 4.36.

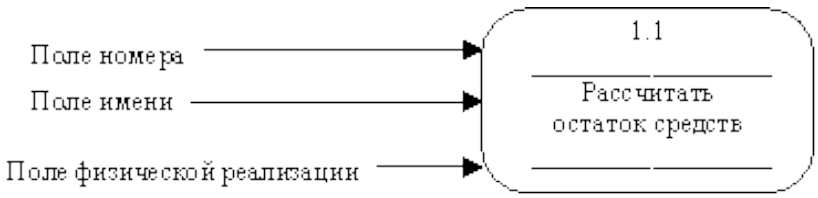

Рис. 4.36. Процесс

Номер процесса служит для его идентификации. Каждый процесс имеет уникальный номер для ссылок на него внутри диаграммы, который может использоваться совместно с номером диаграммы для получения уникального индекса процесса во всей модели.

В поле *имени* вводится наименование процесса в виде предложения с активным недвусмысленным глаголом в неопределенной форме (вычислить, рассчитать, проверить, определить, создать, получить), за которым следуют существительные в винительном падеже, обозначающие объект преобразования, например:

- «ввести сведения о клиентах»;
- «выдать информацию о текущих расходах»;
- «проверить кредитоспособность клиента»;
- «получить документы по отгрузке продукции».

Назначение *процесса (работы)* состоит в продуцировании выходных потоков из входных в соответствии с действием, задаваемым *именем* процесса.

Информация в *поле физической реализации* показывает, какое подразделение организации, программа или аппаратное устройство выполняет данный процесс.

Внешняя сущность представляет собой субъект окружающей среды (юридическое или физическое лицо, иногда материальный предмет), представляющее собой источник или приемник информации, например, заказчики, поставщики, клиенты, склад.

Определение некоторого объекта или системы в качестве внешней сущности указывает на то, что она находится за пределами границ анализируемой ИС. В процессе анализа некоторые внешние сущности могут быть перенесены внутрь диаграммы анализируемой ИС, если это необходимо, или, наоборот, часть процессов ИС может быть вынесена за пределы диаграммы и их исполнители будут представлены как внешние сущности.

Внешняя сущность обозначается прямоугольником (рис. 4.37), расположенным как бы «над» диаграммой и бросающим на нее тень, для того, чтобы можно было выделить этот символ среди других обозначений.

Рис. 4.37. Внешняя сущность

Внешние сущности обычно располагаются по краям диаграммы. Одна внешняя сущность может быть использована многократно на одной или нескольких диаграммах. Обычно такой прием используют, чтобы не рисовать слишком длинных и запутанных стрелок

Потоки данных описывают движение информации из одной части системы в другую. Поток данных на диаграмме изображается линией, оканчивающейся стрелкой, которая показывает направление потока (рис. 4.38). Каждый поток данных имеет имя, отражающее его содержание.

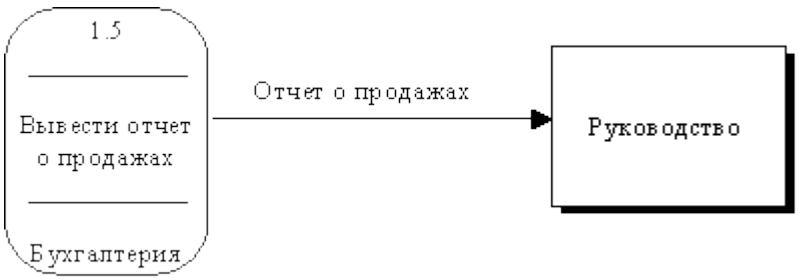

Рис. 4.38. Поток данных

Поскольку в DFD каждая сторона работы не имеет четкого назначения, как в IDEF0, стрелки могут подходить и выходить из любой грани прямоугольника работы. В DFD могут также применяться двунаправленные стрелки для описания диалогов типа «команда-ответ» между работами, между работой и внешней сущностью и между внешними сущностями.

Понятие «потоки данных» является абстракцией, использующейся для моделирования передачи информации (или физических компонент) из одной части системы в другую. Информация передается через некоторое соединение от источника к приемнику. Реальный поток данных может быть информацией, передаваемой по кабелю между двумя устройствами, пересылаемыми по почте письмами, магнитными лентами или дискетами, переносимыми с одного компьютера на другой и т.д.

Источники информации (*внешние сущности*) порождают информационные потоки (*потоки данных*), переносящие информацию к *подсистемам* или *процессам*. Те в свою очередь преобразуют информацию и порождают новые потоки, которые переносят информацию к другим *процессам* или *подсистемам*, *накопителям данных* или *внешним сущностям* – потребителям информации.

Хранилище (накопитель данных) позволяет на указанных участках определять данные, которые будут сохраняться в памяти между процессами. Фактически хранилище представляет «срезы» потоков данных во времени. Ин-

формация, которую оно содержит, может использоваться в любое время после ее получения, при этом данные могут выбираться в любом порядке. Имя хранилища должно определять его содержимое и быть существительным.

Накопитель данных на диаграмме потоков данных изображается, как показано на рис. 4.39.

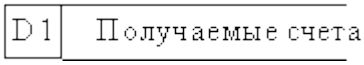

Рис. 4.39. Накопитель данных

Накопитель данных идентифицируется буквой «D» и произвольным числом. *Имя накопителя* выбирается из соображения наибольшей информативности для проектировщика.

Накопитель данных в большинстве случаев является прообразом существующей или будущей базы данных (или таблицы БД) и описание хранящихся в нем данных должно быть увязано с информационной моделью. В широком смысле накопитель данных представляет собой абстрактное устройство для хранения информации, которую можно в любой момент поместить в накопитель и через некоторое время извлечь. Причем способы помещения и извлечения могут быть любыми, а физически накопитель данных может быть реализован в виде микрофиши, ящика в картотеке, таблицы в оперативной памяти, файла на магнитном носителе и т.д.

Словари данных являются каталогами всех элементов данных, присутствующих в DFD, включая групповые и индивидуальные потоки данных, хранилища и процессы, а также все их атрибуты.

Миниспецификации обработки описывают DFD-процессы нижнего уровня. Фактически миниспецификации представляют собой алгоритмы описания задач, выполняемых процессами. Множество всех миниспецификаций является полной спецификацией системы.

Таким образом, диаграммы верхних уровней иерархии (контекстные диаграммы) определяют основные процессы (подсистемы) с внешними входами и выходами. Они детализируются при помощи диаграмм нижнего уровня. *Деком-*

позиция продолжается до тех пор, пока не будет достигнут такой уровень декомпозиции, на котором процессы становятся элементарными и детализировать их далее невозможно.

Стрелки DFD показывают, как данные двигаются от одной работы к другой. *Диаграммы* DFD содержат элементы для обозначения *внешних сущностей* (*источников* и *приемников данных),* а также *хранилищ данных. Процессы* и *потоки информации* совместно с *хранилищами* и *внешними сущностями* на DFD представляют *функциональную часть ИС*, *информационное обеспечение*, а также *границы предметной области* моделирования.

Для описания последовательности и условий выполнения процессов используют **методологию IDEF3** моделирования потоков работ.

IDEF3 захватывает также поведенческие аспекты существующих или предполагаемых систем, структурируя процессы в контексте определенных сценариев.

Моделирование потоков работ

Описание стандарта

Метод моделирования потоков работ IDEF3 (workflow diagramming, WFD), являющийся частью семейства стандартов IDEF, был разработан в конце 1980-х годов для закрытого проекта ВВС США и был предназначен для моделирования таких процессов, в которых необходимо понять последовательность выполнения операций и взаимозависимости между ними.

Хотя IDEF3 и не достиг статуса федерального стандарта США, он приобрел широкое распространение среди системных аналитиков как дополнение к методу функционального моделирования IDEF0 и методу DFD.

Модели IDEF3 могут использоваться для описания сценариев или правил выполнения процессов, представленных функциональными блоками на диаграммах IDEF0 и DFD.

IDEF3 позволяет описать следующее:

- последовательность выполнения процессов;
- условия перехода от одной операции к другой;
- объекты, участвующие совместно в одном *процессе*.

Схемы IDEF3 содержат следующие **элементы**:

- *логические операторы* (блоки принятия решений);
- *события начала и окончания процесса*;
- *элементы, показывающие временные задержки*.

С помощью **логических операторов** показывается, в каких случаях процесс протекает по одной технологии, а в каких – по другой. Например, с помощью данных элементов можно описать ситуацию, когда договор, стоимость которого меньше определенной суммы, согласуется одной группой сотрудников, а договор с большей стоимостью согласуется по более сложной технологии или цепочке, в которой участвует большее количество сотрудников.

С помощью **событий начала и окончания процесса** показывается, когда процесс начинается и когда заканчивается. Для жестко формализованных бизнес-процессов, например, таких, как бюджетирование, в качестве событий может выступать время.

В случаях, когда описание бизнес-процесса проводится с целью его дальнейшей временной оптимизации, используют элементы **задержки времени**, которые показывают места, в которых между последовательно выполняемыми работами имеется временной разрыв. Например, можно показать, что последующая работа начинается только через некоторое время после завершения предшествующей.

IDEF3 является стандартом документирования информационных, технологических и иных процессов, происходящих на предприятии, и предоставляет инструментарий для наглядного исследования и моделирования их сценариев**.**

Сценарием (Scenario) – описание последовательности изменений свойств объекта, в рамках рассматриваемого процесса (например, описание по-

следовательности этапов обработки детали в цеху и изменение её свойств после прохождения каждого этапа).

Исполнение каждого сценария сопровождается соответствующими потоками информации, например, в виде документов. На промышленном предприятии документооборот производственных процессов состоит из двух основных потоков: документов, определяющих структуру и последовательность процесса (технологических указаний, описаний стандартов и т.д.), и документов, отображающих ход его выполнения (результатов тестов и экспертиз, отчетов о браке, и т.д.).

Для эффективного управления любым процессом, необходимо иметь детальное представление об его сценарии и структуре сопутствующего документооборота. Средства документирования и моделирования IDEF3 позволяют выполнять следующие задачи:

- документировать имеющиеся данные о технологии процесса, выявленные, в процессе предпроектного обследования путем опроса компетентных сотрудников, ответственных за организацию рассматриваемого процесса;
- определять и анализировать точки слияния и разделения потоков информации;
- определять ситуации, в которых требуется принятие решения, влияющего на жизненный цикл процесса;
- содействовать принятию оптимальных решений при реорганизации процессов;
- разрабатывать модели процессов, по принципу "КАК БУДЕТ, ЕСЛИ..."

IDEF3 имеет прямую взаимосвязь с методологией IDEF0 – каждая функция (функциональный блок IDEF0) может быть представлена в виде отдельного процесса средствами IDEF3.

Два типа диаграмм в IDEF3

Существуют два типа диаграмм в стандарте IDEF3, представляющие описание одного и того же сценария технологического процесса в разных ракурсах.

Диаграммы относящиеся к первому типу называются **диаграммами Описания Последовательности Этапов Процесса** (Process Flow Description Diagrams, **PFDD**). Иное встречающееся название для **PFDD** – диаграмма работ WFD (Work Flow Diagram).

Ко второму типу относятся **диаграммы Состояния Объекта и его Трансформаций в Процессе** (Object State Transition Network, **OSTN**).

Пример. Требуется задокументировать процесс окраски детали в производственном цеху на предприятии. С помощью диаграмм PFDD документируется последовательность и описание стадий обработки детали в рамках исследуемого технологического процесса. Диаграммы OSTN используются для иллюстрации трансформаций детали, которые происходят на каждой стадии обработки.

Процесс состоит непосредственно из самой окраски, производимой на специальном оборудовании и этапа контроля ее качества, который определяет, нужно ли деталь окрасить заново (в случае несоответствия стандартам и выявления брака) или отправить ее в дальнейшую обработку. На рис. 4.40 изображена диаграмма PFDD, являющаяся графическим отображение сценария обработки детали.

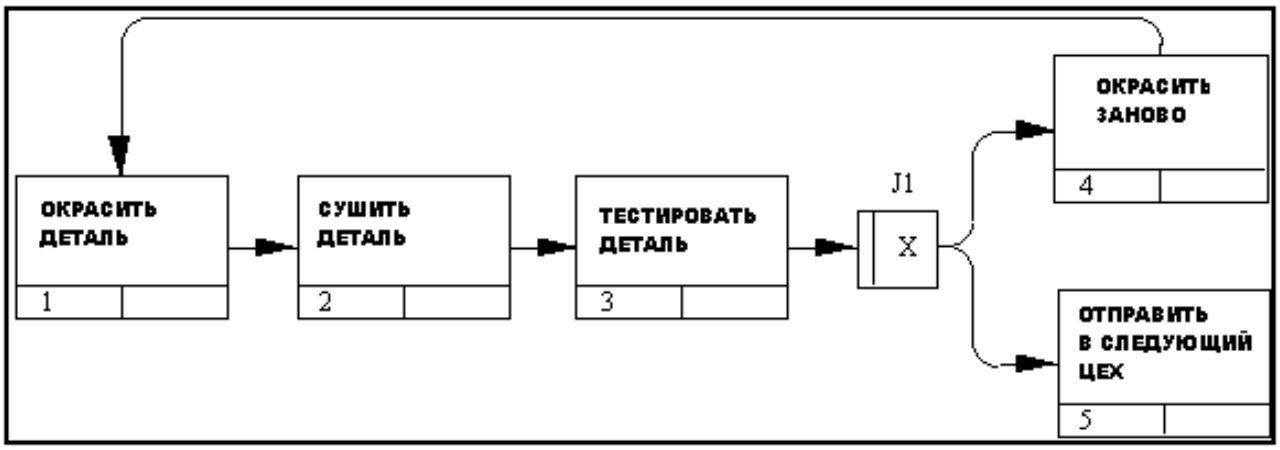

Рис. 4.40. Пример PFDD диаграммы окраски детали

Прямоугольники на диаграмме PFDD называются **функциональными элементами** или **элементами поведения** (Unit of Behavior, UOB) и обозначают *событие***,** *стадию процесса* или *принятие решения*.

Каждый UOB имеет свое *имя*, отображаемое в глагольном наклонении и *уникальный номер*.

Стрелки или *линии* являются отображением перемещения детали между UOB-блоками в ходе процесса.

Объект, обозначенный J1 – называется *перекрестком* (Junction).

Перекрестки используются для отображения логики взаимодействия стрелок (потоков) при слиянии и разветвлении или для отображения множества событий, которые могут или должны быть завершены перед началом следующей работы.

Различают перекрестки для слияния (Fan-in Junction) и разветвления (Fan-out Junction) стрелок. Перекресток не может использоваться одновременно для слияния и для разветвления. При внесении перекрестка в диаграмму необходимо указать тип перекрестка.

Классификация возможных типов перекрестков приведена в таблице 4.3.

Все перекрестки в PFDD диаграмме нумеруются, каждый номер имеет префикс "J".

Сценарий, отображаемый на диаграмме, можно описать в следующем виде. Деталь поступает в окрасочный цех подготовленной к окраске. В процессе окраски наносится один слой эмали при высокой температуре. После этого, производится сушка детали, после которой начинается этап проверки качества нанесенного слоя. Если тест подтверждает недостаточное качество нанесенного слоя (недостаточную толщину, неоднородность и т.д.), то деталь заново пропускается через цех окраски. Если деталь успешно проходит контроль качества, то она отправляется в следующий цех для дальнейшей обработки.

Таблица 4.3.

Классификация перекрестков

Обозначение	Наименование	Смысл в случае слияния стрелок (Fan-in Junction)	Смысл в случае разветвления стрелок (Fan-out Junction)
	Asynchronous AND	Все предшествующие процессы должны быть завершены	Все следующие процессы должны быть запущены
	Synchronous AND	Все предшествующие процессы завершены одновременно	Все следующие процессы запускаются одновременно
	Asynchronous OR	Один или несколько предшествующих процессов должны быть завершены	Один или несколько следующих процессов должны быть запущены
	Synchronous OR	Один или несколько предшествующих процессов завершаются одновременно	Один или несколько следующих процессов запускаются одновременно
	XOR (Exclusive OR)	Только один предшествующий процесс завершен	Только один следующий процесс запускается

Каждый функциональный блок UOB может иметь последовательность **декомпозиций**, и, следовательно, может быть детализирован с любой необходимой точностью. Под **декомпозицией** понимается представление каждого UOB с помощью отдельной IDEF3 диаграммы.

Пример. Можно декомпозировать UOB «Окрасить Деталь», представив его отдельным процессом и построив для него свою PFDD диаграмму. При этом эта диаграмма будет называться *дочерней* по отношению к изображенной на рис., а та, соответственно, *родительской*.

Номера UOB дочерних диаграмм имеют иерархическую нумерацию, т.е., если родительский UOB имеет номер «1», то блоки UOB на его декомпозиции будут соответственно иметь номера «1.1», «1.2» и т.д.

Применение принципа декомпозиции в IDEF3 позволяет структурировано описывать процессы с любым требуемым уровнем детализации.

Диаграммы состояния объекта и его трансформаций в процессе
(Object State Transition Network, **OSTN**)

Если диаграммы PFDD рассматривают технологический процесс «с точки зрения наблюдателя», то другой класс диаграмм IDEF3 OSTN позволяет рассматривать тот же самый процесс *с точки зрения объекта*. На рис. 4.41 представлена OSTN диаграмма процесса окраски с точки зрения детали.

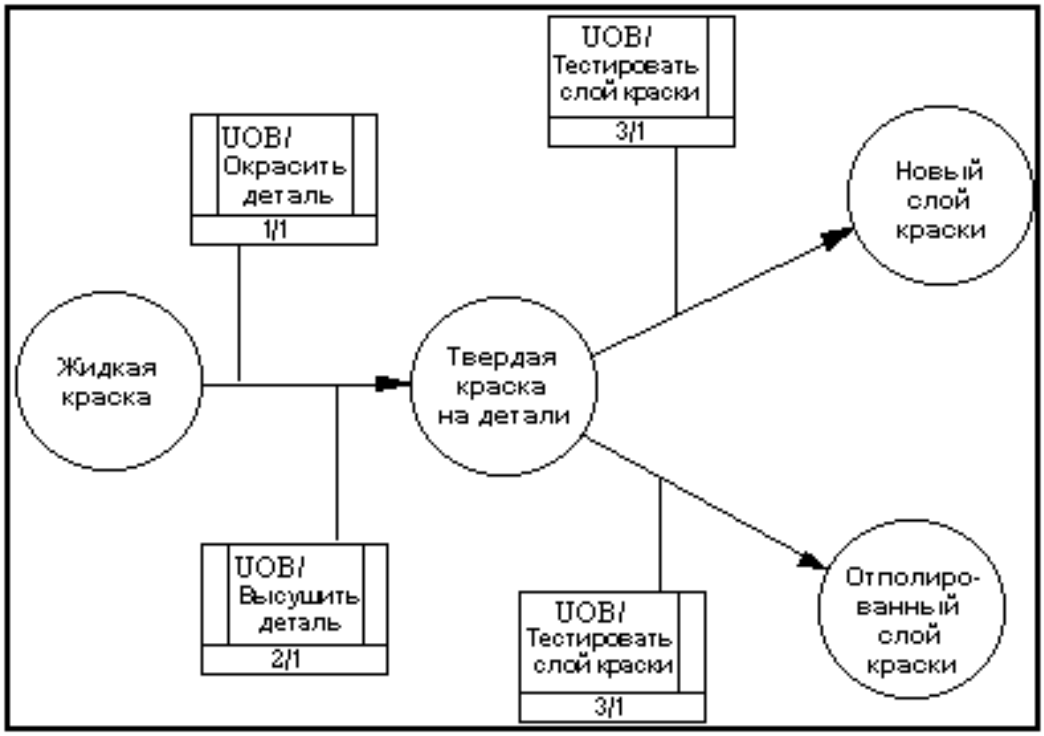

Рис. 4.41. Пример OSTN диаграммы

Состояния объекта (в нашем случае детали) и *изменение состояния* являются ключевыми понятиями OSTN диаграммы. Состояния объекта отображаются *окружностями*, а их *изменения – направленными линиями*. Каждая линия имеет ссылку на соответствующий функциональный блок UOB, в результате которого произошло отображаемое ей изменение состояния объекта.

На рис. 4.42 приведен более сложный пример IDEF3 диаграммы.

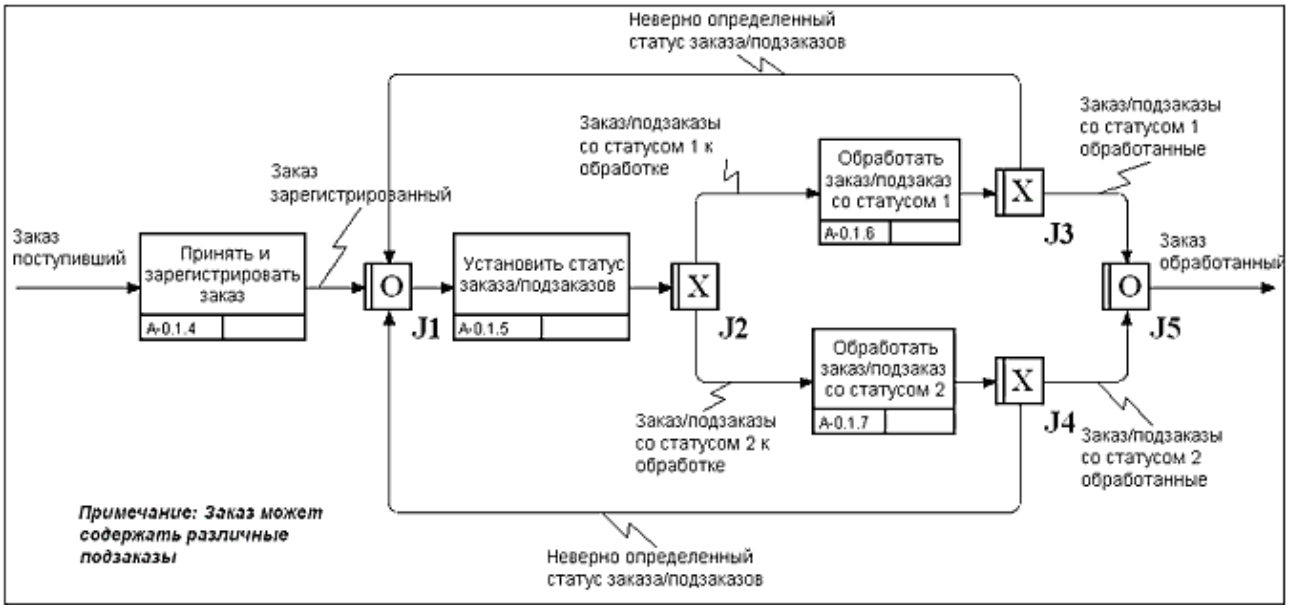

Рис. 4.42. Фрагмент PFDD диаграммы

Отличительной особенностью IDEF3-диаграммы является то, что стрелки между операциями бизнес-процесса обозначают не потоки объектов (информационные и материальные), а временную последовательность выполнения работ.

IDEF3 используют в основном для описания бизнес-процессов нижнего уровня детализации (детальные работы, по названию которых понятно, что является входом и что – выходом) и могут быть использованы для анализа завершенности процедур обработки информации.

IDEF3 дополняет IDEF0 и DFD и содержит все необходимое для построения моделей, которые в дальнейшем могут быть использованы для **имитационного анализа**.

Имитационное моделирование

Имитационное моделирование может служить для получения *оценочных аспектов* моделирования предметной области, которые связаны с разрабатываемыми показателями эффективности автоматизируемых *процессов*.

Имитационное моделирование заключается в построении и исследовании имитационной модели. **Имитационные модели –** это динамические модели,

учитывающие время выполнения операций, и позволяющие анализировать динамику бизнес-процессов.

Имитационные модели описывают не только сущности, потоки информации и управление, но и различные метрики. Полученную модель можно «проиграть» во времени и получить статистику происходящих процессов так, как это было бы в реальности. В имитационной модели изменения процессов и данных ассоциируются с *событиями*. «Проигрывание» модели заключается в последовательном переходе от одного события к другому.

Имитационная модель включает следующие *основные элементы*:

1) *источники и стоки* (Create и Dispose). **Источники** – это элементы, от которых в модель поступает информация или материальные объекты. По смыслу они близки к понятиям «внешняя сущность» на DFD или «объект ссылки» на диаграммах IDEF3. Скорость поступления данных или объектов от источника обычно задается статистической функцией. **Сток** – это устройство для приема информации или объектов;
2) *очереди (Queues)*. Понятие очереди близко к понятию «хранилища данных» на DFD-диаграммах – это место, где объекты ожидают обработки. Время обработки объектов в разных работах может быть разным. В результате перед некоторыми работами могут накапливаться объекты, ожидающие своей очереди. Часто целью имитационного моделирования является минимизация количества объектов в очередях;
3) *процессы (Process)* – это аналог работ в функциональной модели. В имитационной модели может быть задана производительность процессов.

Функциональные и имитационные модели взаимосвязаны и дополняют друг друга. Имитационная модель дает больше информации для анализа системы, в свою очередь результаты такого анализа могут быть причиной модификации модели процессов.

Существует также возможность преобразования функциональной модели

в имитационную модель. Целесообразно сначала строить функциональную модель, а на ее основе – имитационную.

Построение модели производится с помощью специальных CASE средств путем переноса из панели инструментов в рабочее пространство модулей Create, Dispose и Process.

Связи между модулями устанавливаются автоматически, но могут быть переопределены вручную. Далее модулям назначаются свойства. Для контроля проигрывания модели необходимо в модель добавить модуль Simulate и задать для него параметры. Результаты проигрывания модели отображаются в автоматически генерируемых отчетах.

BPwin не имеет собственных инструментов, позволяющих создавать имитационные модели, однако дает возможность экспортировать модель IDEF3 в специализированное средство создания таких моделей.

Для экспорта модели необходимо настроить свойства, определяемые пользователем UDP, специально включенные в BPwin для целей экспорта.

ЗАКЛЮЧЕНИЕ

Материал книги рекомендуется студентам для изучения дисциплины «Методы и средства проектирования информационных систем и технологий» направления подготовки высшего профессионального образования для бакалавров 230400 «Информационные системы и технологии»; преподавателями для обучения студентов решению следующих профессиональных задач проектной деятельности в соответствии с ФГОС ВПО направления: предпроектного обследования объекта автоматизации, системного анализа предметной области, моделирования систем, технического и рабочего проектирования, расчета экономической эффективности разрабатываемой системы, подготовке проектной документации. В рамках дисциплины предполагается усвоение теоретического материала на практике.

Учебное пособие содержит четыре раздела с соответствующими главами. Раздел 1 включает основные понятия: общей теории систем, предметной области проектирования, информационных систем и технологий, проектирования информационных систем, жизненного цикла ИС. Раздел содержит описание системного анализа, как основополагающего метода при исследовании систем управления и лежащего в основе современных методологий проектирования ИС, а также принципов, правил и процедур системного подхода при исследовании объекта и создании ИС.

Раздел 2 – нормативно-методичечская поддержка ЖЦ ИС – обзор стандартов, методов и инструментальных средств в области создания автоматизированных информационных систем, рекомендации по управлению программным проектом.

Материалы разделов 3 (предпроектное обследование объекта) и 4 (структурный анализ и структурный подход) могут быть использованы при выполнении практических занятий по дисциплине «Методы и средства проектирования информационных систем и технологий».

Библиографический список

1. Инюшкина О.Г., Кормышев В.М. Исследование систем управления при проектировании информационных систем: учебное пособие. / О.Г. Инюшкина, В.М. Кормышев. Екатеринбург: «Форт-Диалог Исеть», 2013. 370 с.
2. Гольдштейн С.Л., Инюшкина О.Г. Практика использования информационных технологий и систем (на примерах управления организацией): учебное пособие / С.Л. Гольдштейн, О.Г. Инюшкина. Екатеринбург: УрФУ, 2010. 185 с.
3. Инюшкина О.Г., Кормышев В.М. Управление знаниями в информационных системах (монография). / О.Г. Инюшкина, В.М. Кормышев, Екатеринбург: УрФУ, 2012. 212 с.
4. Rob M.A. Issues of Structured vs. Object-Oriented methodology of systems analysis and design. [Электронный ресурс] Режим доступа: PDF.
5. IDEF4 Object-Oriented Design Method. [Электронный ресурс] Режим доступа: http://www.idef.com/IDEF4.htm.
6. Wikipedia: UML [Электронный ресурс] Режим доступа: https://ru.wikipedia.org/wiki/UML.
7. Буч Г., Рамбо Д., Джекобсон А. Язык UML Руководство пользователя. [Электронный ресурс] Режим доступа: PDF.
8. Övergaard G., Selic B., Bock C., Björkander M. Behavioral Modeling. UML Revision Task Force Object Modeling with OMG UML Tutorial Series. [Электронный ресурс] Режим доступа: 01-2_Bock_Behavioral_ModelingTutorial.pdf (http://www.omg.org/).
9. Pefkaros K. Using object-oriented analysis and design over traditional structured analysis and design // International Journal of Business Research Publisher. 03/01/2008 [Электронный ресурс] Режим доступа: http://www.freepatentsonline.com/article/International-Journal-Business-Research/190463129.html.
10. Маклаков С.В. «ERwin и BPwin. CASE-средства разработки информацион-

ных систем» / С.В. Маклаков, 2-е изд., испр. и доп., М. : Диалог-Мифи, 2001. – 304 с.М: «Диалг-МИФИ», 2001.

11. Вендров А.М. CASE-технологии. Современные методы и средства проектирования информационных систем / А.М. Вендров. – М. : Финансы и статистика, 1998. – 176 с. Программное обеспечение. [Электронный ресурс] Режим доступа: www.interface.ru.
12. Руководство по применению ГОСТ Р ИСО/МЭК 12207 при управлении проектом. PDF, www.complexdoc.ru.
13. Дэвид А. Марка, Клемент МакГоуэн. Предисловие Дугласа Т. Росса. Методология структурного анализа и проектирования SADT Structured Analysis & Design Technique. [Электронный ресурс] Режим доступа: www.pqm-online.com/assets/files/lib/mar

Учебное издание

Инюшкина Ольга Георгиевна

ПРОЕКТИРОВАНИЕ ИНФОРМАЦИОННЫХ СИСТЕМ
(на примере методов структурного системного анализа)

Научный редактор *д.п.н., проф. Матвеева Т.А.*

Компьютерный набор *авторский*

Издательство ООО «Форт-Диалог Исеть»
Подписано в печать 28.11.2014. Бумага писчая Формат 60х84 1/16.
Печать на ризографе. Усл. п.л. 13,95
Тираж 50 экз. Заказ 1470282

Отпечатано в типографии ООО «Форт-Диалог Исеть»
620085, г. Екатеринбург, ул. Монтерская, д. 3 тел.: (343) 228-02-32