Лекция 5. Методологии моделирования предметной области

1. Структурная модель предметной области

2. Функционально-ориентированные и объектно-ориентированные методологии описания предметной области

1. Структурная модель предметной области

В основе *проектирования ИС* лежит моделирование предметной области. Для того чтобы получить адекватный предметной области проект ИС в виде системы правильно работающих программ, необходимо иметь целостное, системное представление модели, которое отражает все аспекты функционирования будущей информационной системы. При этом под ***моделью предметной области*** понимается некоторая система, имитирующая структуру или функционирование исследуемой предметной области и отвечающая основному требованию – быть адекватной этой области.

Предварительное *моделирование предметной области* позволяет сократить время и сроки проведения проектировочных *работ* и получить более эффективный и качественный проект. Без проведения моделирования *предметной области* велика *вероятность* допущения большого количества ошибок в решении стратегических вопросов, приводящих к экономическим потерям и высоким затратам на последующее перепроектирование системы. Вследствие этого все современные технологии *проектирования ИС* основываются на использовании методологии моделирования предметной области.

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:

- объектной структуры, отражающей состав взаимодействующих в процессах материальных и информационных объектов предметной области;
- функциональной структуры, отражающей взаимосвязь *функций* (действий) по преобразованию объектов в процессах;
- структуры управления, отражающей события и бизнес-правила, которые воздействуют на выполнение процессов;

- организационной структуры, отражающей взаимодействие *организационных единиц* предприятия и персонала в процессах;
- технической структуры, описывающей топологию расположения и способы коммуникации комплекса технических средств.

Для отображения структурного аспекта *моделей предметных областей* в основном используются графические методы, которые должны гарантировать *представление* информации о компонентах системы. Главное требование к графическим методам документирования — простота. Графические методы должны обеспечивать возможность структурной декомпозиции спецификаций системы с *максимальной степенью* детализации и согласований описаний на смежных уровнях декомпозиции.

С моделированием непосредственно связана проблема **выбора языка** представления проектных решений, позволяющего как можно больше привлекать будущих пользователей системы к ее разработке. ***Язык моделирования*** – это *нотация*, в основном графическая, которая используется для описания проектов. ***Нотация*** представляет собой совокупность графических объектов, используемых в модели. *Нотация* является синтаксисом *языка моделирования*. *Язык моделирования*, с одной стороны, должен делать решения проектировщиков понятными пользователю, с другой стороны, предоставлять проектировщикам средства достаточно формализованного и однозначного определения проектных решений, подлежащих реализации в виде программных комплексов, образующих целостную систему программного обеспечения.

Графическое изображение нередко оказывается наиболее емкой формой представления информации. При этом проектировщики должны учитывать, что графические методы документирования не могут полностью обеспечить декомпозицию проектных решений от постановки задачи проектирования до реализации программ ЭВМ. Трудности возникают при переходе от этапа анализа системы к этапу проектирования и в особенности к программированию.

Главный **критерий адекватности структурной модели** *предметной области* заключается в *функциональной полноте* разрабатываемой ИС.

Оценочные аспекты моделирования *предметной области* связаны с разрабатываемыми показателями эффективности автоматизируемых процессов, к которым относятся:

- время решения задач;
- стоимостные затраты на обработку данных;
- надежность процессов;
- 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* (*АВС*) и *динамические методы имитационного моделирования*.

В основе различных методологий моделирования *предметной области* ИС лежат принципы последовательной детализации абстрактных категорий. Обычно модели строятся на трех уровнях: на внешнем уровне (**определении требований**), на концептуальном уровне (**спецификации требований**) и внутреннем уровне (**реализации требований**). Так, на внешнем уровне модель отвечает на вопрос, что должна делать система, то есть определяется состав основных компонентов системы: объектов, *функций,* событий, *организационных единиц*, технических средств. **На концептуальном уровне** модель отвечает на вопрос, как должна функционировать система? Иначе говоря, определяется характер взаимодействия компонентов системы одного и разных типов. На внутреннем уровне модель отвечает на вопрос: с помощью каких программно-технических средств реализуются требования к системе? С позиции жизненного *цикла* ИС описанные уровни моделей соответственно строятся на этапах *анализа требований*, логического (технического) и физического (рабочего) проектирования. Рассмотрим особенности построения *моделей предметной области* на трех уровнях детализации.

Объектная структура

Объект — это сущность, которая используется при выполнении некоторой *функции* или *операции* (преобразования, обработки, формирования и т.д.). Объекты могут иметь динамическую или статическую природу: *динамические объекты* используются в одном цикле воспроизводства, например заказы на продукцию, счета на оплату, платежи; статические объекты используются во многих циклах воспроизводства, например, оборудование, персонал, запасы материалов.

На внешнем уровне детализации модели выделяются основные виды материальных объектов (например, сырье и материалы, полуфабрикаты, готовые изделия, услуги) и основные виды информационных объектов или документов (например, заказы, накладные, счета и т.д.).

На концептуальном уровне построения *модели предметной области* уточняется состав классов объектов, определяются их атрибуты и взаимосвязи. Таким образом строится обобщенное представление структуры предметной области.

Далее *концептуальная модель* на внутреннем уровне отображается в виде файлов базы данных, входных и выходных документов ЭИС. Причем *динамические объекты* представляются единицами переменной информации или документами, а *статические объекты* — единицами условно-постоянной информации в виде списков, номенклатур, ценников, справочников, *классификаторов*. Модель базы данных как постоянно поддерживаемого информационного ресурса отображает хранение условно-постоянной и накапливаемой переменной информации, используемой в повторяющихся информационных процессах.

Функциональная структура

Функция (*операция*) представляет собой некоторый преобразователь входных объектов в выходные. Последовательность взаимосвязанных по

входам и выходам *ункций* составляет *бизнес-процесс*. *Функция бизнес-процесса* может порождать объекты любой природы (материальные, денежные, информационные). Причем *бизнес-процессы* и информационные процессы, как правило, неразрывны, то есть *функции* материального процесса не могут осуществляться без информационной поддержки. Например, отгрузка готовой продукции осуществляется на основе документа "Заказ", который, в свою очередь, порождает документ "Накладная", сопровождающий партию отгруженного товара.

Функция может быть представлена одним действием или некоторой совокупностью действий. В последнем случае каждой *функции* может соответствовать некоторый процесс, в котором могут существовать свои *подпроцессы*, и т.д., пока каждая из подфункций не будет представлять некоторую недекомпозируемую последовательность действий.

На внешнем уровне моделирования определяется список основных бизнес-функций или видов *бизнес-процессов*. Обычно таких *функций* насчитывается 15–20.

На концептуальном уровне выделенные *функции* декомпозируются и строятся иерархии взаимосвязанных *функций*.

На внутреннем уровне отображается структура информационного процесса в компьютере: определяются иерархические структуры программных модулей, реализующих автоматизируемые *функции*.

Структура управления

В совокупности *функций бизнес-процесса* возможны альтернативные или циклические последовательности в зависимости от различных условий протекания процесса. Эти условия связаны с происходящими событиями во внешней среде или в самих процессах и с образованием определенных состояний объектов (например, заказ принят, отвергнут, отправлен на корректировку). **События** вызывают выполнение *функций*, которые, в свою очередь, изменяют состояния объектов и формируют новые события, и т.д., пока не будет завершен некоторый *бизнес-процесс*. Тогда последовательность событий составляет конкретную реализацию *бизнес-процесса*.

Каждое событие описывается с двух точек зрения: **информационной** и **процедурной**. Информационно событие отражается в виде некоторого сообщения, фиксирующего факт выполнения некоторой *функции*изменения состояния или появления нового. Процедурно событие вызывает выполнение новой *функции*, и поэтому для каждого состояния объекта должны быть заданы описания этих вызовов. Таким образом, события выступают в связующей роли для выполнения *функций бизнес-процессов*.

На внешнем уровне определяются список внешних событий, вызываемых взаимодействием предприятия с внешней средой (платежи налогов, процентов по кредитам, поставки по контрактам и т.д.), и список целевых установок, которым должны соответствовать *бизнес-*

процессы (регламент выполнения процессов, поддержка уровня материальных запасов, уровень качества продукции и т.д.).

На концептуальном уровне устанавливаются бизнес-правила, определяющие условия вызова *функций* при возникновении событий и достижении состояний объектов.

На внутреннем уровне выполняется формализация бизнес-правил в виде триггеров или вызовов программных модулей.

Организационная структура

Организационная структура представляет собой совокупность *организационных единиц*, как правило, связанных иерархическими и процессными отношениями. Организационная единица — это подразделение, представляющее собой объединение людей (персонала) для выполнения совокупности общих *функций* или *бизнес-процессов*. В функционально-ориентированной организационной структуре организационная единица выполняет набор *функций*, относящихся к одной *функции* управления и входящих в различные процессы. В процессно-ориентированной структуре организационная единица выполняет набор *функций*, входящих в один тип процесса и относящихся к разным *функциям* управления.

На внешнем уровне строится *структурная модель* предприятия в виде иерархии подчинения *организационных единиц* или списков взаимодействующих подразделений.

На концептуальном уровне для каждого подразделения задается организационно-штатная структура должностей (ролей персонала).

На внутреннем уровне определяются требования к правам доступа персонала к автоматизируемым *функциям* информационной системы.

Техническая структура

Топология определяет территориальное размещение технических средств по структурным подразделениям предприятия, а коммуникация — технический способ реализации взаимодействия структурных подразделений.

На внешнем уровне модели определяются типы технических средств обработки данных и их размещение по структурным подразделениям.

На концептуальном уровне определяются способы коммуникаций между техническими комплексами структурных подразделений: физическое перемещение документов, машинных носителей, обмен информацией по каналам связи и т.д.

На внутреннем уровне строится модель "клиент-серверной" архитектуры вычислительной сети.

Описанные *модели предметной области* нацелены на проектирование отдельных компонентов ИС: данных, функциональных программных модулей, управляющих программных модулей, программных модулей интерфейсов пользователей, структуры технического комплекса. Для более качественного проектирования указанных компонентов требуется построение моделей, увязывающих различные компоненты ИС между собой. В простейшем случае в качестве таких моделей взаимодействия могут

использоваться матрицы перекрестных ссылок: "объекты-функции", "функции-события", "*организационные единицы — функции* ", "*организационные единицы* — объекты", "*организационные единицы* — технические средства" и т д. Такие матрицы не наглядны и не отражают особенности реализации взаимодействий.

Для правильного отображения взаимодействий компонентов ИС важно осуществлять совместное моделирование таких компонентов, особенно с содержательной точки зрения объектов и *функций*. Методология структурного *системного анализа* существенно помогает в решении таких задач.

Структурным анализом принято называть метод исследования системы, который начинается с ее общего обзора, а затем детализируется, приобретая иерархическую структуру с все большим числом уровней. Для таких методов характерно: разбиение на уровни абстракции с ограниченным числом элементов (от 3 до 7); ограниченный контекст, включающий только существенные детали каждого уровня; использование строгих формальных правил записи; последовательное приближение к результату. *Структурный анализ* основан на двух базовых принципах – "разделяй и властвуй" и принципе иерархической упорядоченности. 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. Определим ключевые понятия *структурного анализа*.

Операция – элементарное (неделимое) действие, выполняемое на одном рабочем месте.

Функция – совокупность *операций*, сгруппированных по определенному признаку.

Бизнес-процесс — связанная совокупность *функций*, в ходе выполнения которой потребляются определенные ресурсы и создается продукт (предмет, услуга, научное открытие, идея), представляющая ценность для потребителя.

Подпроцесс – это *бизнес-процесс*, являющийся структурным элементом некоторого *бизнес-процесса* и представляющий ценность для потребителя.

Бизнес-модель – структурированное графическое описание сети процессов и *операций*, связанных с данными, документами, организационными единицами и прочими объектами, отражающими существующую или предполагаемую деятельность предприятия.

Существуют различные методологии структурного моделирования предметной области, среди которых следует выделить **функционально-ориентированные и объектно-ориентированные методологии.**

2. Функционально-ориентированные и объектно-ориентированные методологии описания предметной области

Процесс бизнес-моделирования может быть реализован в рамках различных методик, отличающихся прежде всего своим подходом к тому, что

представляет собой моделируемая организация. В соответствии с различными представлениями об организации методики принято делить на объектные и функциональные (структурные).

Объектные методики рассматривают моделируемую организацию как набор взаимодействующих объектов – производственных единиц. *Объект* определяется как осязаемая реальность – предмет или явление, имеющие четко определяемое поведение. Целью применения данной методики является выделение объектов, составляющих организацию, и распределение между ними ответственностей за выполняемые действия.

Функциональные методики, наиболее известной из которых является методика *IDEF*, рассматривают организацию как набор *функций*, преобразующий поступающий поток информации в выходной поток. Процесс преобразования информации потребляет определенные ресурсы. Основное отличие от *объектной методики* заключается в четком отделении *функций* (методов обработки данных) от самих данных.

С точки зрения бизнес-моделирования каждый из представленных подходов обладает своими преимуществами. Объектный подход позволяет построить более устойчивую к изменениям систему, лучше соответствует существующим *структурам организации*. *Функциональное моделирование* хорошо показывает себя в тех случаях, когда организационная структура находится в процессе изменения или вообще слабо оформлена. Подход от выполняемых *функций* интуитивно лучше понимается исполнителями при получении от них информации об их текущей работе.

Функциональная методика IDEF0

Методологию *IDEF0* можно считать следующим этапом развития хорошо известного графического языка описания функциональных систем *SADT* (*Structured Analysis* and *Design Technique*). Исторически *IDEF0* как стандарт был разработан в 1981 году в рамках обширной программы автоматизации промышленных предприятий, которая носила обозначение *ICAM* (Integrated Computer Aided *Manufacturing*). Семейство стандартов*IDEF* унаследовало свое обозначение от названия этой программы (*IDEF=Icam* DEFinition), и последняя его редакция была выпущена в декабре 1993 года Национальным Институтом по Стандартам и Технологиям США (*NIST*).

Целью методики является построение *функциональной схемы* исследуемой системы, описывающей все необходимые процессы с точностью, достаточной для однозначного моделирования деятельности системы.

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. 6.1). 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*).

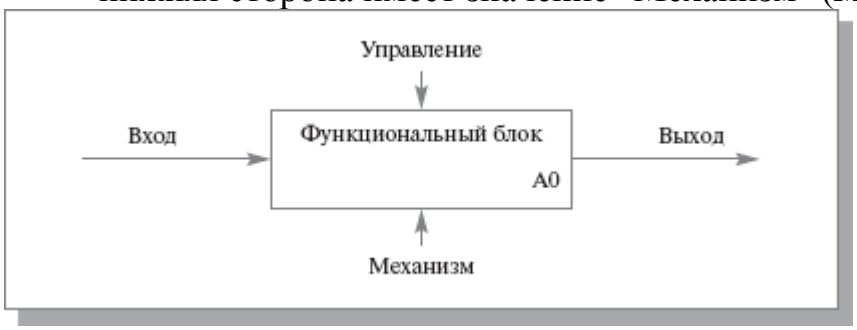

Рис. 6.1. Функциональный блок

Интерфейсная дуга (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 объектами 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).

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 структурировано представлять модель системы в виде иерархической структуры отдельных диаграмм, что делает ее менее перегруженной и легко усваиваемой.

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 элементом. Этот набор называется глоссарием и является описанием сущности данного элемента. Глоссарий гармонично дополняет наглядный графический язык, снабжая диаграммы необходимой дополнительной информацией.

Модель *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.**

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 определяет основное направление развития модели и уровень необходимой детализации. Четкое фиксирование точки зрения позволяет разгрузить модель, отказавшись от детализации и исследования отдельных элементов, не являющихся необходимыми, исходя из выбранной точки зрения на систему. Правильный выбор точки зрения существенно сокращает временные затраты на построение конечной модели.

Выделение *подпроцессов*. В процессе декомпозиции функциональный блок, который в *контекстной диаграмме* отображает систему как единое целое, подвергается детализации на другой диаграмме. Получившаяся диаграмма второго уровня содержит функциональные блоки, отображающие главные подфункции функционального блока *контекстной диаграммы*, и называется дочерней (Child Diagram) по отношению к нему (каждый из функциональных блоков, принадлежащих дочерней диаграмме, соответственно 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). Каждая из подфункций дочерней диаграммы может быть далее детализирована путем аналогичной декомпозиции соответствующего ей функционального блока. В каждом случае декомпозиции функционального блока все интерфейсные дуги, входящие в данный блок или исходящие из него, фиксируются на

дочерней диаграмме. Этим достигается структурная целостность *IDEF0*–модели.

Иногда отдельные интерфейсные дуги высшего уровня не имеет смысла продолжать рассматривать на диаграммах нижнего уровня, или наоборот — отдельные дуги нижнего отражать на диаграммах более высоких уровней – это будет только перегружать диаграммы и делать их сложными для восприятия. Для решения подобных задач в стандарте *IDEF0* предусмотрено понятие *туннелирования*. Обозначение "*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*".

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.

Рекомендуется представлять на диаграмме от трех до шести функциональных блоков, при этом количество подходящих к одному функциональному блоку (выходящих из одного функционального блока) интерфейсных дуг предполагается не более четырех.

Стандарт *IDEF0* содержит набор процедур, позволяющих разрабатывать и согласовывать модель большой группой людей, принадлежащих к разным областям деятельности моделируемой системы. Обычно процесс разработки является итеративным и состоит из следующих условных этапов:

- Создание модели группой специалистов, относящихся к различным сферам деятельности предприятия. Эта группа в терминах *IDEF0* называется авторами (Authors). Построение первоначальной модели является динамическим процессом, в течение которого авторы опрашивают компетентных лиц о структуре различных процессов, создавая модели деятельности подразделений. При этом их интересуют ответы на следующие вопросы:

Что поступает в подразделение "на входе"?

- Какие *функции* и в какой последовательности выполняются в рамках подразделения?
- Кто является ответственным за выполнение каждой из *функций*?
- Чем руководствуется исполнитель при выполнении каждой из *функций*?

- Что является результатом работы подразделения (на выходе)?

На основе имеющихся положений, документов и результатов опросов создается черновик (Model Draft) модели.

- Распространение черновика для рассмотрения, согласований и комментариев. На этой стадии происходит обсуждение черновика модели с широким кругом компетентных лиц (в терминах *IDEF0* — читателей) на предприятии. При этом каждая из диаграмм черновой модели 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.
- 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 предприятии (системе) с заданной точки зрения и для заданной цели.

Наглядность графического языка *IDEF0* делает модель вполне читаемой и для лиц, которые не принимали участия в проекте ее создания, а также эффективной для проведения показов и презентаций. В дальнейшем на базе построенной модели могут быть организованы новые проекты, нацеленные на производство изменений в модели.

Функциональная методика потоков данных

Целью методики является построение модели рассматриваемой системы в виде *диаграммы потоков данных* (*Data Flow Diagram — DFD*), обеспечивающей правильное описание выходов (отклика системы в виде данных) при заданном воздействии на вход системы (подаче сигналов через внешние интерфейсы). *Диаграммы потоков данных* являются основным средством моделирования *функциональных требований* к проектируемой системе.

При создании *диаграммы потоков данных* используются четыре основных понятия: **потоки данных, процессы (работы) преобразования входных потоков данных в выходные, внешние сущности, накопители данных (хранилища).**

Потоки данных являются абстракциями, использующимися для моделирования передачи информации (или физических компонент) из одной части системы в другую. Потоки на диаграммах изображаются именованными стрелками, ориентация которых указывает направление движения информации.

Назначение **процесса** (работы) состоит в продуцировании выходных потоков из входных в соответствии с действием, задаваемым именем процесса. Имя процесса должно содержать глагол в неопределенной форме с последующим дополнением (например, "получить документы по отгрузке

продукции"). 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.

Хранилище (накопитель) данных позволяет на указанных участках определять данные, которые будут сохраняться в памяти между процессами. Фактически хранилище представляет "срезы" потоков данных во времени. Информация, 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.

Внешняя сущность представляет собой материальный объект вне контекста системы, являющейся источником или приемником системных данных. Ее имя должно содержать существительное, например, "склад товаров". Предполагается, что объекты, представленные как *внешние сущности*, не должны участвовать ни в какой обработке.

Кроме основных элементов, в состав *DFD* входят словари данных и миниспецификации.

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.

Процесс построения *DFD* начинается с создания так называемой основной диаграммы типа "звезда", на которой представлен моделируемый процесс и все *внешние сущности*, с которыми он взаимодействует. В случае сложного основного процесса он сразу представляется в виде декомпозиции на ряд взаимодействующих процессов. Критериями сложности в данном случае являются: наличие большого числа *внешних сущностей*, многофункциональность системы, ее распределенный характер. Внешние сущности выделяются по отношению к основному процессу. Для их определения необходимо выделить поставщиков и потребителей основного процесса, т.е. все объекты, которые взаимодействуют с основным процессом. На этом этапе описание взаимодействия заключается в выборе глагола, дающего представление о том, как внешняя сущность использует основной процесс или используется им. Например, основной процесс – "учет обращений граждан", внешняя сущность – "граждане", описание взаимодействия – "подает заявления и получает ответы". Этот этап является принципиально важным, поскольку именно он определяет границы моделируемой системы.

Для всех *внешних сущностей* строится *таблица событий*, описывающая их взаимодействие с основным потоком. *Таблица событий* включает в себя наименование внешней сущности, событие, его тип

(типичный для системы или исключительный, реализующийся при определенных условиях) и реакцию системы.

На следующем шаге происходит декомпозиция основного процесса на набор взаимосвязанных процессов, обменивающихся потоками данных. Сами потоки не конкретизируются, определяется лишь характер взаимодействия. Декомпозиция завершается, когда процесс становится простым, т.е.:

1. процесс имеет два-три входных и выходных потока;
2. процесс может быть описан в виде преобразования входных данных в выходные;
3. процесс может быть описан в виде *последовательного алгоритма*.

Для простых процессов строится миниспецификация – формальное описание алгоритма преобразования входных данных в выходные.

Миниспецификация удовлетворяет следующим требованиям: для каждого процесса строится одна спецификация; спецификация однозначно определяет входные и выходные потоки для данного процесса; спецификация не определяет способ преобразования входных потоков в выходные; спецификация ссылается на имеющиеся элементы, не вводя новые; спецификация по возможности использует стандартные подходы и*операции*.

После декомпозиции основного процесса для каждого *подпроцесса* строится аналогичная таблица *внутренних событий*.

Следующим шагом после определения полной *таблицы событий* выделяются **потоки данных**, которыми обмениваются процессы и *внешние сущности*. Простейший способ их выделения заключается в анализе таблиц событий. События преобразуются в потоки данных от инициатора события к запрашиваемому процессу, а реакции – в обратный поток событий. После построения входных и выходных потоков аналогичным образом строятся внутренние потоки. Для их выделения для каждого из внутренних процессов выделяются поставщики и потребители информации. Если поставщик или потребитель информации представляет процесс сохранения или запроса информации, то вводится хранилище данных, для которого данный процесс является интерфейсом.

После построения потоков данных диаграмма должна быть проверена на полноту и непротиворечивость. Полнота диаграммы обеспечивается, если в системе нет "повисших" процессов, не используемых в процессе преобразования входных потоков в выходные. Непротиворечивость системы обеспечивается выполнением наборов формальных правил о возможных типах процессов: на диаграмме не может быть потока, связывающего две *внешние сущности* – это взаимодействие удаляется из рассмотрения; ни одна сущность не может непосредственно получать или отдавать информацию в хранилище данных – хранилище данных является пассивным элементом, управляемым с помощью интерфейсного процесса; два хранилища данных не могут непосредственно обмениваться информацией – эти хранилища должны быть объединены.

К преимуществам методики *DFD* относятся:

- возможность однозначно определить *внешние сущности*, анализируя потоки информации внутри и вне системы;
- возможность проектирования сверху вниз, что облегчает построение модели "как должно быть";
- наличие спецификаций процессов нижнего уровня, что позволяет преодолеть логическую незавершенность *функциональной модели* и построить полную *функциональную спецификацию* разрабатываемой системы.

К недостаткам модели отнесем: необходимость искусственного ввода *управляющих процессов*, поскольку управляющие воздействия (потоки) и *управляющие процессы* с точки зрения *DFD* ничем не отличаются от обычных; отсутствие понятия времени, т.е. отсутствие анализа временных промежутков при преобразовании данных (все ограничения по времени должны быть введены в спецификациях процессов).

Объектно-ориентированная методика

Принципиальное отличие между функциональным и объектным подходом заключается в способе декомпозиции системы. Объектно-ориентированный подход использует объектную декомпозицию, при этом статическая структура описывается в терминах **объектов и связей** между ними, а поведение системы описывается в терминах **обмена сообщениями** между объектами. Целью методики является построение *бизнес-модели*организации, позволяющей перейти от модели *сценариев использования* к модели, определяющей отдельные объекты, участвующие в реализации бизнес-функций.

Концептуальной основой *объектно-ориентированного подхода* является объектная модель, которая строится с учетом следующих принципов:

- абстрагирование;
- инкапсуляция;
- модульность;
- иерархия;
- типизация;
- параллелизм;
- устойчивость.

Основными понятиями *объектно-ориентированного подхода* являются объект и класс.

Объект — предмет или явление, имеющее четко определенное поведение и обладающие состоянием, поведением и индивидуальностью. Структура и поведение схожих объектов определяют общий для них класс. **Класс – это множество объектов, связанных общностью структуры и поведения**. Следующую группу важных понятий объектного подхода составляют наследование и полиморфизм. Понятие**полиморфизм** может быть интерпретировано как способность класса принадлежать более чем одному типу. **Наследование** означает построение новых классов на основе

существующих с возможностью добавления или переопределения данных и методов.

Важным качеством объектного подхода является согласованность моделей деятельности организации и моделей проектируемой информационной системы от стадии формирования требований до стадии реализации. По объектным моделям может быть прослежено отображение реальных сущностей моделируемой предметной области (организации) в объекты и классы информационной системы.

Большинство существующих методов *объектно-ориентированного подхода* включают *язык моделирования* и описание процесса моделирования. **Процесс** – это описание шагов, которые необходимо выполнить при разработке проекта. В качестве *языка моделирования* объектного подхода используется унифицированный *язык моделирования* UML, который содержит стандартный набор диаграмм для моделирования.

Диаграмма (Diagram) — это графическое представление множества элементов. Чаще всего она изображается в виде связного графа с вершинами (сущностями) и ребрами (отношениями) и представляет собой некоторую проекцию системы.

Объектно-ориентированный подход обладает следующими преимуществами:

- Объектная декомпозиция дает возможность создавать модели меньшего размера путем использования общих механизмов, обеспечивающих необходимую экономию выразительных средств. Использование объектного подхода существенно повышает уровень унификации разработки и пригодность для повторного использования, что ведет к созданию среды разработки и переходу к сборочному созданию моделей.
- Объектная декомпозиция позволяет избежать создания сложных моделей, так как она предполагает эволюционный путь развития модели на базе относительно небольших подсистем.
- Объектная модель естественна, поскольку ориентирована на человеческое восприятие мира.

К недостаткам *объектно-ориентированного подхода* относятся высокие начальные затраты. Этот подход не дает немедленной отдачи. Эффект от его применения сказывается после разработки двух–трех проектов и накопления повторно используемых компонентов. Диаграммы, отражающие специфику объектного подхода, менее наглядны.

Сравнение существующих методик

В **функциональных моделях** (*DFD-диаграммах потоков данных*, *SADT*-диаграммах) главными структурными компонентами являются *функции* (*операции*, действия, работы), которые на диаграммах связываются между собой *потоками объектов*.

Несомненным достоинством *функциональных моделей* является реализация *структурного подхода* к проектированию ИС по принципу

"сверху-вниз", когда каждый функциональный блок может быть декомпозирован на множество подфункций и т.д., выполняя, таким образом, модульное *проектирование ИС*. Для *функциональных моделей* характерны процедурная строгость декомпозиции ИС и наглядность представления.

При *функциональном подходе* объектные модели данных в виде *ER-диаграмм* "объект — свойство — связь" разрабатываются отдельно. Для проверки корректности моделирования предметной области между функциональными и объектными моделями устанавливаются взаимно однозначные связи.

Главный недостаток *функциональных моделей* заключается в том, что процессы и данные существуют отдельно друг от друга — помимо функциональной декомпозиции существует структура данных, находящаяся на втором плане. Кроме того, не ясны условия выполнения процессов обработки информации, которые динамически могут изменяться.

Перечисленные недостатки *функциональных моделей* снимаются в **объектно-ориентированных моделях**, в которых главным структурообразующим компонентом выступает класс объектов с набором *функций*, которые могут обращаться к атрибутам этого класса.

Для классов объектов характерна иерархия обобщения, позволяющая осуществлять **наследование** не только атрибутов (свойств) объектов от вышестоящего класса объектов к нижестоящему классу, но и *функций*(методов).

В случае наследования *функций* можно абстрагироваться от конкретной реализации процедур (**абстрактные типы данных**), которые отличаются для определенных подклассов ситуаций. Это дает возможность обращаться к подобным программным модулям по общим именам (**полиморфизм**) и осуществлять повторное использование программного кода при модификации программного обеспечения. Таким образом, адаптивность объектно-ориентированных систем к изменению предметной области по сравнению с *функциональным подходом* значительно выше.

При объектно-ориентированном подходе изменяется и принцип *проектирования ИС*. Сначала выделяются классы объектов, а далее в зависимости от возможных состояний объектов (жизненного цикла объектов) определяются методы обработки (функциональные процедуры), что обеспечивает наилучшую реализацию динамического поведения информационной системы.

Для *объектно-ориентированного подхода* разработаны графические методы моделирования предметной области, обобщенные в языке унифицированного моделирования UML. Однако по наглядности представления модели пользователю-заказчику объектно-ориентированные модели явно уступают функциональным моделям.

При выборе методики моделирования предметной области обычно в качестве критерия выступает степень ее динамичности. Для более регламентированных задач больше подходят функциональные модели, для более адаптивных *бизнес-процессов* (управления рабочими потоками,

реализации динамических запросов к информационным хранилищам) — объектно-ориентированные модели. Однако в рамках одной и той же ИС для различных классов задач могут требоваться различные виды моделей, описывающих одну и ту же проблемную область. В таком случае должны использоваться комбинированные *модели предметной области*.

Синтетическая методика

Как можно видеть из представленного обзора, каждая из рассмотренных методик позволяет решить задачу построения формального описания рабочих процедур исследуемой системы. Все методики позволяют построить модель "как есть" и "как должно быть". С другой стороны, каждая из этих методик обладает существенными недостатками. Их можно суммировать следующим образом: недостатки применения отдельной методики лежат не в области описания реальных процессов, а в неполноте методического подхода.

Функциональные методики в целом лучше дают *представление* о существующих *функциях* в организации, о методах их реализации, причем чем выше степень детализации исследуемого процесса, тем лучше они позволяют описать систему. Под лучшим описанием в данном случае понимается наименьшая ошибка при попытке по полученной модели предсказать поведение реальной системы. На уровне отдельных рабочих процедур их описание практически однозначно совпадает с фактической реализацией в *потоке работ*.

На уровне общего описания системы *функциональные методики* допускают значительную степень произвола в выборе общих интерфейсов системы, ее механизмов и т.д., то есть в определении границ системы. Хорошо описать систему на этом уровне позволяет объектный подход, основанный на понятии сценария использования. Ключевым является понятие о сценарии использования как о сеансе взаимодействия действующего лица с системой, в результате которого действующее лицо получает нечто, имеющее для него ценность. Использование критерия ценности для пользователя дает возможность отбросить не имеющие значения детали потоков*работ* и сосредоточиться на тех *функциях* системы, которые оправдывают ее существование. Однако и в этом случае задача определения границ системы, выделения внешних пользователей является сложной.

Технология потоков данных, исторически возникшая первой, легко решает проблему границ системы, поскольку позволяет за счет анализа информационных потоков выделить *внешние сущности* и определить основной внутренний процесс. Однако отсутствие выделенных управляющих процессов, потоков и событийной ориентированности не позволяет предложить эту методику в качестве единственной.

Наилучшим способом преодоления недостатков рассмотренных методик является формирование **синтетической методики**, объединяющей различные этапы отдельных методик. При этом из каждой методики необходимо взять часть методологии, наиболее полно и формально

изложенную, и обеспечить возможность обмена результатами на различных этапах применения синергетической методики. В бизнес-моделировании неявным образом идет формирование подобной синергетической методики.

Идея **синтетической методики** заключается в последовательном применении функционального и объектного подхода с учетом возможности реинжиниринга существующей ситуации.

Рассмотрим применение синтетической методики на примере разработки административного регламента.

При построении административных регламентов выделяются следующие стадии:

1. Определение границ системы. На этой стадии при помощи **анализа потоков данных выделяют внешние сущности** и собственно моделируемую систему.
2. Выделение *сценариев использования* системы. На этой стадии **при помощи критерия** полезности **строят** для каждой внешней сущности **набор сценариев использования системы.**
3. Добавление системных *сценариев использования*. На этой стадии **определяют сценарии, необходимые для реализации целей системы**, отличных от целей пользователей.
4. Построение диаграммы активностей по сценариям использования. На этой стадии строят **набор действий системы,** приводящих к реализации *сценариев использования*;
5. Функциональная **декомпозиция диаграмм активностей** как контекстных диаграмм методики *IDEF0.*
6. Формальное описание отдельных функциональных активностей в виде административного регламента (с применением различных *нотаций*).