Функционально-ориентированное проектирование ЭИС

Основными идеями функционально-ориентированной CASE-технологии являются идеи структурного анализа и проектирования информационных систем. Они заключаются в следующем:

* декомпозиция всей системы на некоторое множество иерархически подчиненных функций;
* представление всей информации в виде графической нотации (наглядность и простота понимания).

В качестве инструментальных средств структурного анализа и проектирования выступают следующие диаграммы:

* BFD (Bussiness Function Diagram) - диаграмма бизнес-функций (функциональные спецификации);
* DFD (Data Flow Diagram) - диаграмма потоков данных;
* STD (State Transition Diagram) - диаграмма переходов состояний (матрицы перекрестных ссылок);
* ERD (Entity Relationship Diagram) - ER-модель данных предметной области (информационно-логические модели «сущ­ность-связь»);
* SSD (System Structure Diagram) - диаграмма структуры программного приложения.

**Диаграммы функциональных спецификаций** позволяют представить общую структуру ИС, отражающую взаимосвязь различных задач (процедур) в процессе получения требуемых результатов. Основными объектами BFD являются:

* Функция - некоторое действие информационной системы, необходимое для решения экономической задачи;
* Декомпозиция функции - разбиение функции на множество подфункций.

При построении диаграмм BFD возможно использование следующих нотаций::

* Йодана (Yourdon);
* Гейна - Сарсона (Gane - Sarson);
* SADT (Structured Analysis and Design Technique);
* SAG (Software AG).

**Диаграммы потоков данных** (ДПД), как правило, жестко ори­ентированы на какую-либо технологию обработки данных и от­ражают передачу информации от одной функции к другой в рамках заданной технологии обработки. В узлах диаграммы пото­ков данных (прямоугольниках) отражаются процедуры, а стрелками между узлами показываются потоки данных (над стрелками задаются имена передаваемых/используемых единиц инфор­мации - документов, экранных форм, файлов).

ДПД - показывает внешние по отношению к системе источники данных и адресатов, которые принимают информацию от системы, а также идентифицируют хранилища данных (накопители данных), к которым осуществляется доступ системы. Каждая логическая функция системы (бизнес-функция) описывается своей ДПД. Причем эта ДПД может иерархически детализировать функцию на ее подфункции.

Основные объекты ДПД и их графические изображения в различных нотациях (табл. 13.2).

* Потоки данных - являются механизмами, которые показывают передачу информации от одного процесса к другому. На схемах они обычно отражаются направленной стрелкой, которая показывает направление движения информации или материален (могут отражаться материальные потоки).
* Процесс - его функция состоит в преобразовании входном информации в выходную. Имя процесса всегда должно содержать глагол в неопределенной форме с последующим дополнением (например, «нарисовать форму»).
* Хранилище информации - позволяет на определенных участках ДПД сохранить в памяти данные между процессами. Хранилище не обязательно представлено магнитным носителем (например, папка бумаг). Имя хранилища должно идентифицировать его, а также его содержимое, выражается существительным.
* Внешняя сущность (источник/приемник данных) - представляет некоторый объект вне системы, являющийся внешним объектом.
* Контекстная диаграмма - самый верхний процесс (ТОР-уровень) декомпозиции системы, который отражает общие представления о системе. В контекстной диаграмме есть 1 процесс, с которым связаны внешние сущности.

Далее контекстная диаграмма декомпозируется на основные процессы, которые происходят в системе. Каждый основной про­цесс может быть декомпозирован на более мелкие процессы. При иерархическом построении ДПД каждый процесс более низкого уровня нужно соотнести с процессом более высокого уровня. Обычно для этого используют механизм наследования узлов.

Целью построения иерархически взаимосвязанных ДПД яв­ляется необходимость сделать требования к системе ясными на каждом уровне детализации. Для этого надо пользоваться следующими рекомендациями:

* на каждом уровне представлять 3-6 процессов и не более;
* не загромождать диаграмму несущественными моментами на данном уровне детализации;
* декомпозицию процессов и потоков вести параллельно;
* выбирать ясные, отражающие суть объектов, имена для всех объектов ДПД;
* однократно определять функционально идентичные процессы (в других местах просто ссылаться на этот процесс);
* использовать ДПД для процессов, которые можно с помощью них описать.

**Диаграммы переходов состояний** (ДПС) моделируют поведение системы во времени в зависимости от происшедших событий (нажатая клавиша, дата отчетного периода и т.д.). Такие диаграммы позволяют осуществить декомпозицию управляющих процессов, происходящих в системе, и описать отношение между управляющими потоками. С помощью ДПС можно моделировать последующее функционирование системы исходя из предыдущих и текущего состояний.

Моделируемая система в текущий момент времени находится только в одном состоянии из всего множества состояний. В тече­ние времени она может изменить свое состояние и тем самым перейти в следующее состояние из заданного множества состояний. Для перехода в состояние нужно какое-либо особое условие -условие перехода. Оно может быть информационным (условие появления информации) или временным. В табл. 13.3 представлены символы ДПС в различных нотациях. Определим основные объекты ДПС:

* Состояние - рассматривается как устойчивое значение некоторого свойства в течение определенного времени. Находясь в текущем состоянии, необходимо знать о предыдущих состо­яниях, чтобы определить условие перехода в последующее состояние.
* Начальное состояние - это узел ДПС, являющийся стартовой точкой для начального системного перехода. ДПС имеет только одно начальное состояние, но может иметь множество конечных состояний.
* Переход - определяет перемещение моделируемой системы из одного состояния в другое. При этом имя перехода - это событие, которое вызвало этот переход. Переход может быть вызван каким-либо действием (например, нажатием клавиши).
* Триггер - логическое выражение, написанное на макроязыке, которое показывает условие перехода в данное состояние.
* Условие перехода - событие, вызывающее переход и идентифицируемое именем перехода.

**Диаграммы инфологических моделей** «сущность-связь» (ER-диаграммы) ориентированы на разработку базы данных, структура которой не зависит от конкретных информационных потребностей и позволяет выполнять любые запросы пользователей.

ERD-диаграмма «сущность-связь» (Чена, Беркера, SADT, SAG) представляет собой набор множества объектов и их характеристик, а также взаимосвязей между ними, нужных для выявленных данных, которые в дальнейшем используются функциями проектируемой системы:

* Сущность - представляет собой множество экземпляров реальных или абстрактных объектов, которые обладают общими свойствами (атрибутами).
* Отношение - связь между 2 и более сущностями (должны создать имя в виде глагола).
* Независимая сущность - представляет независимые данные, которые всегда присутствуют в системе.
* Зависимая сущность - представляет данные, которые зависят от других сущностей.

**Диаграмма структуры программного приложения** (SSD) задает взаимосвязь функций и программных модулей, которые их реализуют (меню, формы,отчеты и т.д.).

Структура программного приложения (SSD) представляет собой иерархическую взаимосвязь программных модулей, которые реализует ИС. SSD служит мостом для перехода от системных требований, которые отображены в предыдущих диаграммах (BFD, DFD, STD, ERD), к реализации информационной системы.

При построении ДПС рекомендуется следовать перечисленным ниже правилам:

* начинать построение ДПС на высоком уровне детализации ДПД;
* строить наиболее простые диаграммы, содержащие 4-6 состояний;
* по возможности включать детализацию в виде подчиненных шагов состояния (детализация на другом уровне);
* использовать те же приемы наименования состояний, событий и действий, что и при наименовании процессов и потоков.

Применяются 2 способа построения ДПС:

* первый способ заключается в том, что выявляются возможные состояния системы и далее выявляются переходы из одного состояния в другое;
* при втором способе сначала строится начальное состояние, затем осуществляется переход в очередное состояние и т.д. (последовательный переход).

В результате получаем предварительную ДПС. Затем она проверяется на корректность ее построения.

Этапы построения системной структурной диаграммы.

* В диаграмме бизнес-функций необходимо выделить функции, которые будут реализованы в программном виде.
* Взять диаграмму потока данных (соответствующие уровни DFD) для выделенных функций и подфункций и проанализировать ее с учетом входных и выходных потоков данных.
* Определить структуру потоков данных, задав список атрибутов сущностей из ER-диаграммы.
* На диаграмме переходов состояний определить состояния, переходы и события их вызывающие, которые реализуют бизнес-функции.
* Задать программную реализацию каждого состояния в виде библиотечного модуля CASE-системы или модуля, написанного па другом языке.
* Нарисовать эскиз системной структурной диаграммы для каждой выделенной функции.
* Объединить построенные системные структурные диаграммы в одну исходя из диаграммы бизнес-функции.
* Проконтролировать, если позволяют CASE-средства, построенную системную структурную диаграмму.
* Если во время контроля ошибок не найдено, то перейти к прототипированию (макетированию) интерфейса программного приложения на основе системной структурной диаграммы.
* Для каждого модуля необходимо выбрать шаблон интерфейса из встроенной библиотеки либо в режиме конструктора создать шаблон, либо написать программный модуль на встроенном языке программирования.

Таким образом, перед генерацией все элементы системной структурной диаграммы должны быть определены с учетом интерфейса и связи с таблицами ER-модели.

В результате процесса генерации получаем исходные тексты программ на языке выбранной среды (D9). Генерация может быть двух видов:

1. Неполная генерация - на основе диаграммы «сущность-связь» и выбранной целевой СУБД генери­руются модули описания данных DDL на языке описания данных. В результате выполнения неполной генерации на выбранном языке определения данных (SQL и т. п.) создается модуль описания данных.
2. Полная генерация включает в себя:
* генерацию DDL на языке описания данных;
* выбор среды, в которой будет приведен исходный код, полученный во время генерации;
* запуск процесса генерации.