С. И. Жуков
(ВНИИгеосистем, Москва)

ИНСТРУМЕНТАЛЬНЫЕ СРЕДСТВА УПРАВЛЕНИЯ ПРЕДСТАВЛЕНИЕМ

The problem of data presentation management in networks becoms more actual as networks, running distributed software, emerege and evoive. One of the approaches to the problem, followed by ISO (based on ISO standartised language ASN.1) is discussed.

В настоящее время одним из актуальных направлений являются создание и разработка неоднородных распределенных компьютерных систем обработки данных (РСОД). Неоднородность естественным образом возникает при объединении в составе РСОД систем, изготовляемых различными производителями и выполняющих различные функции.

Для решения проблемы неоднородности примерно с середины 1970-х гг. различными организациями, в том числе и международными, проводились работы по стандартизации взаимодействия систем в рамках РСОД. Результатом указанных работ стали серии стандартов и рекомендаций, касающихся различных аспектов взаимодействия систем в составе РСОД. Их концептуальной основой стала Базовая Модель Взаимодействия Открытых Систем (OSI), разработанная ISO.

С точки зрения преодоления неоднородности особое значение имеет шестой уровень Базовой Модели — уровень представления. В его задачи входит преобразование представления данных, передаваемых между открытыми системами таким образом, чтобы разнородные открытые системы могли взаимодействовать между собой. С этой целью уровень представления поддерживает один или несколько стандартных форматов представления данных в процессе передачи (синтаксисов передачи), а абстрактный синтаксис (т. е. логическая структура) данных прикладного уровня описывается с использованием специального языка — нотации ASN.1, также стандартизованной ISO. С каждым сервисным элементом прикладного уровня связывается абстрактный синтаксис

протокольной управляющей информации. Описание этого абстрактного синтаксиса в виде модуля ASN.1 составляет неотъемлемую часть соответствующего международного стандарта [1—4].

С точки зрения логической структуры уровень представления можно представить в виде совокупности двух компонент [5]:

1. Протокольная компонента уровня представления. Эта компонента включает функции интерпретации протокола: установления и освобождения представительного соединения; аварийного разъединения; ресинхронизации соединения; управления множеством определенных контекстов (DCS) — добавление и удаления контекстов представления; непосредственной передачи прочих интерфейсных примитивов между прикладным и сеансовым уровнями.

2. Компонента преобразования данных. Эта компонента осуществляет преобразование параметра "данные пользователя" интерфейсных примитивов из локального представления в синтаксис передачи и обратно. В задачи данной компоненты входит также согласование синтаксиса передачи при установлении соединения и добавлении нового контекста в DCS.

Программно уровень представления реализуется, вообще говоря, в виде протокольного объекта (рис. 1). Программная структура протокольного объекта возникает как результат некоторого отображения логической структуры уровня представления. Протокольная компонента логической структуры отображается на совокупность процедур-обработчиков интерфейсных примитивов и протокольных блоков данных аналогично протокольным компонентам других уровней. При этом выбор и активизация нужного обработчика осуществляются специальной программой-монитором в соответствии с таблицей обработчиков по событиям. Таблица обработчиков и сами обработчики строятся по описанию протокола в виде расширенного конечного автомата.

Программа-монитор способна обрабатывать следующие типы событий: порождение интерфейсного примитива, адресованного данному уровню; завершение обработки интерфейсного примитива, порожденного данным уровнем; таймерное

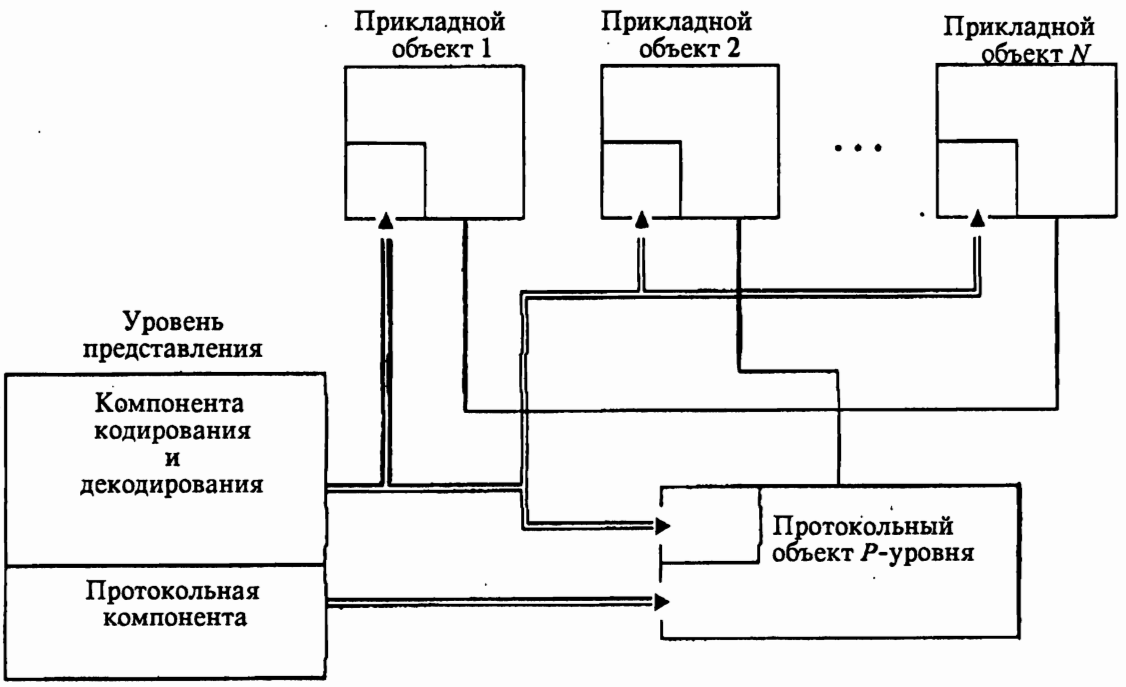

Рис. 1. Общая схема организации уровня представления.

событие. Каждое событие параметризуется идентификатором соединения. Вся информация о соединении содержится в специальной структуре данных — дескрипторе соединения, который передается процедуре-обработчику в качестве параметра. В каждый момент времени существует взаимно однозначное соответствие между идентификаторами соединения и дескрипторами.

Задача отображения компоненты преобразования данных на некоторую совокупность программных компонент является более сложной. Реализация функций преобразования данных непосредственно в составе протокольного объекта ведет либо к значительной потере эффективности, либо к невозможности включения в систему новых прикладных служб, т. е. к потере открытости. Исходя из этого предлагается следующая схема программной реализации компоненты преобразования данных: преобразование данных пользователя выносится из протокольного объекта в прикладные процессы и сервисные элементы, создающие и потребляющие соответствующие данные. Протокольный объект осуществляет контроль передаваемых данных и управление контекстами представления. Однако компонента преобразования данных присутствует и в протокольном объекте, выполняя функции кодирования и декодирования собственной протокольной информации уровня представления.

Компонента преобразования данных программно реализуется в виде библиотеки подпрограмм. Они компонуются вместе с прикладным процессом и осуществляют формирование и разбор блоков данных в синтаксисе передачи по запросу прикладного процесса. Для каждого примитивного типа данных, определенного нотацией ASN.1, задается одна функция кодирования, формирующая представление объекта этого типа в синтаксисе передачи, и одна функция декодирования, осуществляющая обратное действие.

Для структур данных (SEQUENCE, SET, SEQUENCE OF, SET OF) определяются функции кодирования—декодирования начала и окончания соответствующего составного элемента данных. Таким образом, для кодирования составного элемента данных сначала кодируется признак начала составного элемента данных, затем последовательно кодируются компоненты, а в конце вызывается функция, кодирующая признак окончания составного элемента данных. При декодировании элемента данных в ряде случаев для определения его типа (и, следовательно, требуемой декодирующей функции) необходима информация о теге. Для этого используется функция получения тега очередного декодируемого элемента данных. Как правило, декодирование осуществляется путем обращения прикладного процесса к этой функции, после чего на основе полученной информации производится выбор нужной декодирующей функции и обращение к ней.

Необходимость для прикладного процесса самому управлять кодированием—декодированием данных является серьезным недостатком принятого метода реализации. В качестве средства преодоления этого недостатка предлагаются инструментальные средства разработчика прикладных сервисных элементов (ИС РПСЭ) [6, 7]. Эти средства включают, в дополнение к функциям базового кодирования—декодирования, компилятор языка ASN.1 и компилятор языка манипулирования представлением данных (ЯМПД). В функции компилятора ASN.1 входит: синтаксический и семантический анализ модуля описания абстрактного синтаксиса; его трансляция в специальное внутреннее представление и запись результата в библиотеку абстрактных синтаксисов.

Язык ЯМПД построен на основе нотации описания значений языка ASN.1. Предложения ЯМПД встраиваются с текст программы на базовом языке программирования, а компилятор ЯМПД работает как препроцессор базового языка. Результатом трансляции текста на ЯМПД является программа на базовом языке, которая выполняет формирование или разбор составного элемента данных, обращаясь к функциям базового кодирования—декодирования.

В качестве входных данных для трансляции используется также оттранслированный ASN.1-модуль описания абстрактного синтаксиса передаваемых данных. Имена используемых модулей перечисляются в ЯМПД-предложении USE. Поскольку имя модуля может иметь произвольную длину, оттранслированные модули описаний хранятся в специальной библиотеке, для которой поддерживается справочник имен модулей. Компилятор ЯМПД динамически разрешает ссылки на объекты, определенные в других модулях, если имена этих модулей также указаны в предложении USE.

На основе указанных инструментальных средств разработана технология создания программных компонент прикладных служб, осуществляющих формирование и разбор протокольной управляющей информации. Эта технология включает следующие этапы (рис. 2):

1. Разрабатывается (или копируется из стандарта) модуль описания абстрактного синтаксиса на языке ASN.1;

2. Модуль описаний обрабатывается компилятором ASN.1 и помещается в библиотеку абстрактных синтаксисов;

3. Разрабатывается программа на базовом языке с вставленными в нее предложениями ЯМПД, которые осуществляют формирование/разбор значений протокольной информации в синтаксисе передачи;

4. Текст программы обрабатывается ЯМПД-компилятором, в результате чего получается текст программы на базовом языке;

5. Полученный на этапе 4 текст транслируется компилятором базового языка и компонуется вместе с другими модулями и библиотекой базового кодирования—декодирования, результатом чего является исполняемая программа.

Язык ЯМПД обеспечивает следующие возможности: описание структуры формируемых и разбираемых составных элементов данных; задание значений простых элементов данных при кодировке или переменных, которым должны быть присвоены значения простых элементов данных при разборе, в терминах базового языка программирования; указание логических переменных, значение которых определяет наличие необязательных компонент (OPTIONAL) при формировании и разборе; обработку ошибок кодирования при разборе; вставку в программы формирования и разбора специальных действий, описанных на базовом языке. В качестве базового языка программирования в настоящее время используется язык Си.

Указанные возможности обеспечиваются следующими типами предложений языка ЯМПД:

— предложение USE: в этом предложении перечисляются имена ASN.1-модулей, определяющих используемые при кодировании абстрактные синтаксисы;

— предложение WITH: создает область видимости для целочисленных идентификаторов, которые определены внутри описаний типов ASN.1, а также для вложенных значений;

— предложение VARIABLE: предназначено для описания ЯМПД-переменных — специальных переменных, хранящих информацию о данных в синтаксисе передачи и о текущем состоянии кодирования—декодирования. ЯМПД-переменные содержат именованные компоненты, называемые атрибутами кодирования. Определены следующие атрибуты: start — адрес начала данных в синтаксисе передачи; size — текущая длина данных в синтаксисе передачи; ptr — адрес текущей позиции при кодировании и декодировании; tag — последний записанный или прочитанный тег; error — номер последней обнаруженной ошибки кодирования—декодирования;

— предложение PARAMETER: предназначено для описания формальных параметров, которым в качестве фактических аргументов соответствуют ЯМПД-переменные;

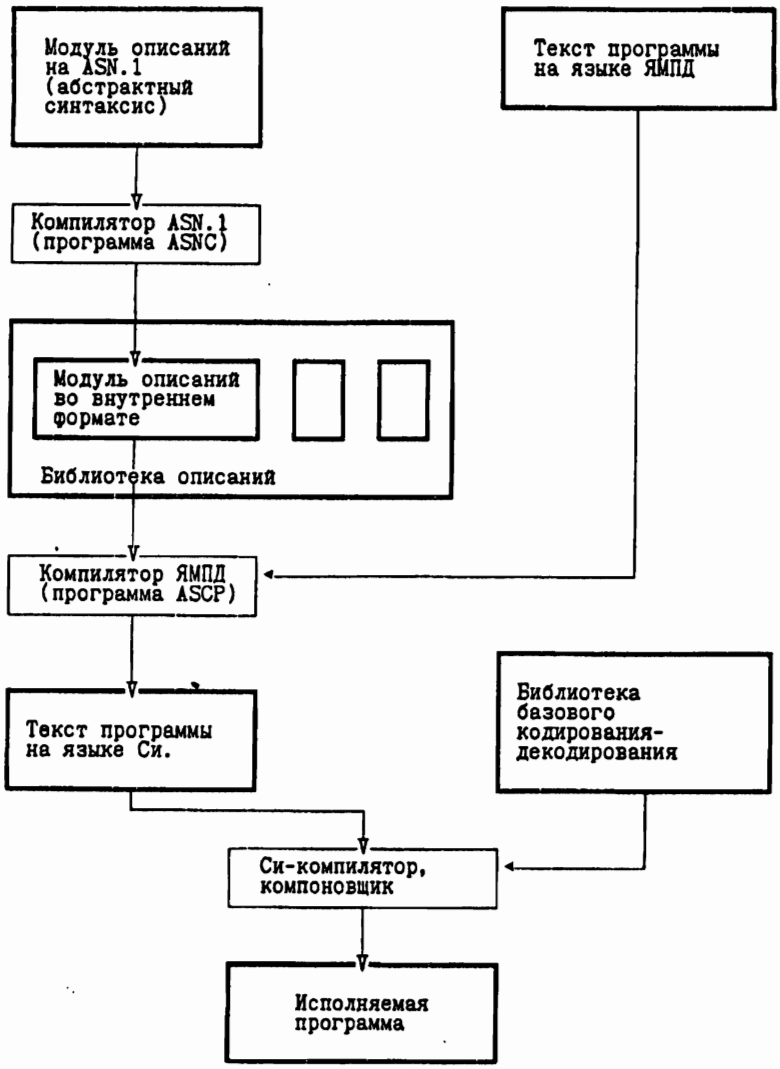

Рис. 2. Схема технологии разработки прикладных служб.

— предложение доступа к значению: предназначено для использования в выражениях базового языка именованных и вложенных ASN.1-значений целого типа; целых констант, обозначаемых идентификаторами внутри описаний типов, а также ЯМПД-переменных и ЯМПД-параметров как единого целого;

— предложение доступа к атрибутам: предназначено для доступа к атрибутам кодирования средствами базового языка;

— предложение ENCODE: описывает процесс кодирования элемента данных — формирования данных в синтаксисе передачи;

— предложение DECODE: описывает процесс декодирования элемента данных — разбора данных в синтаксисе передачи.

Синтаксис двух последних предложений является расширением нотации описания значений ASN.1. Стандартная нотация описания значений может быть использована для кодирования константных значений.

Управление процессом формирования и разбора данных в синтаксисе передачи осуществляется с помощью следующих конструкторов языка ЯМПД:

1. Базовые выражения — размещаются там, где в стандартной нотации могут находиться описания значений. Указание базового выражения вместо описания значения при кодировании означает, что соответствующее значение

вычисляется динамически средствами базового языка. При декодировании базовое выражение является адресом переменной базового языка, которой будет присвоено соответствующее значение.

2. Базовые вставки — представляют собой текст на базовом языке, который неизменным помещается в выходной файл, и используются для дополнительного управления процессом кодирования—декодирования средствами базового языка.

3. Префиксы условия — представляют собой базовые выражения, значение которых ("истина" или "ложь") определяет, следует ли кодировать компоненты составных элементов данных, объявленные как OPTIONAL или DEFAULT. Используются также в качестве предисловий цикла при кодировании значений типа SEQUENCE OF, SET OF.

4. Префиксы наличия — представляют собой адреса переменных базового языка, которым при декодировании присваивается значение "истина" или "ложь" в зависимости от того, присутствовали ли в составном элементе данных соответствующие компоненты, объявленные как OPTIONAL. Используются также при декодировании значений типов CHOICE, SEQUENCE OF, SET OF для указания, производилось ли декодирование согласно соответствующему описателю значения.

Приведем пример использования ЯМПД для обработки протокольной управляющей информации прикладного уровня. Пусть логическая структура информации описана следующим образом:

```
EXAMPLE DEFINITIONS ::= BEGIN
            InfoType ::= SEQUENCE {
                    info1 INTEGER
                    info2 INTEGER OPTIONAL,
            }
END
```

То есть информация включает два целых значения, второе из которых может быть опущено. Тогда модуль, осуществляющий кодирование и декодирование информации, может выглядеть следующим образом:

```
#include <basic.h>

@USE EXAMPLE;

int encode( info1, info2, @PARAMETER lv )
int info1, info2;
{
                @ENCODE lv InfoTupe ::= {
                        info1 (info1),
                        info2 (info2 != 0) ? (info2)
                }
                return @lv.size;
}
void decode( info1, info2, info2_present, @PARAMETER lv )
int *info1, *info2;
bool *info2_present;
{
                @DECODE lv InfoTupe ::= {
                        info1 (info1),
                        info2 (*info2_present) : (info2)
                }
```

```
}
```

Примечания.

1. Параметр lv представляет собой указатель на специальную структуру данных — описатель местоположения закодированных данных.

2. Предполагается, что при кодировании элемент info2 опущен, если его значение равно 0. При возврате из функции декодирования логический флаг info2_present указывает на наличие элемента info2.

ЛИТЕРАТУРА

1. *ISO 8822.* Information Processing Systems — Open System Interconnection: Connection Oriented Presentation Service Definition. 1988. 30 p.

2. *ISO 8823.* Information Processing Systems — Open System Interconnection: Connection Oriented Presentation Protokol Specification. 1988. 54 p.

3. *ISO 8824.* Information Processing Systems — Open System Interconnection: Specification of Abstract Syntax Notation One (ASN. 1). 1987. 50 p.

4. *ISO 8825.* Information Processing Systems — Open System Interconnection: Specification of Basic Encoding Rules for Abstract Syntax Notation One. 1987. 16 p.

5. *Сутормина Т. М.* // Пятнадцатая всесоюзная школа-семинар по вычислительным сетям. М.; Л. 1990. Т. 2. С. 65—66.

6. *Жуков С. И.* // Прикладное математическое и программное обеспечение, моделирование и современные технические средства автоматизации машиностроения. М., 1990. С. 99—103.

7. *Сутормина Т. М., Жуков С. И., Лехтуз Д. Е.* // Труды I-III Всес. школ по супер-ЭВМ. Одесса., 1991. С. 109—112.

Рукопись поступила 25.10.93