マルチメディア通信と分散処理 40 － 3
（１９８９．１．１９）

ＯＳＩディレクトリの実現

宮内直人、中川路哲男、勝山光太郎、水野忠則
三菱電機(株) 情報電子研究所

ＯＳＩ（開放型システム間相互接続）ネットワーク管理サービスの一つに、ディレクトリサービスが標準化されている。今回、我々は集中型のディレクトリを設計し、ワークステーション上に実装した。ディレクトリを実現するソフトウェアは、ユーザにディレクトリへのアクセス手段を提供するＤＵＡライブラリ、ディレクトリ情報へのアクセス手段を提供するＤＩＢライブラリ、及びＤＵＡの要求を処理するＤＳＡの３つの部分から構成される。また、ＤＩＢ本体をオブジェクト指向データベースとして実現した。ディレクトリを実装する際に、 ASN.1ツールAPRICOTやオブジェクト指向言語superC等の既存のソフトウェアを利用して、開発効率を上げると共に、移植性や拡張性を重視したソフトウェア構成とした。

An Implementation of OSI Directory System

Naoto MIYAUCHI, Tetsuo NAKAKAWAJI,
Kotaro KATSUYAMA and Tadanori MIZUNO
Information Systems and Electronics Development Laboratory
MITSUBISHI ELECTRIC CORPORATION
5-1-1 OFUNA KAMAKURA-CITY KANAGAWA 247 JAPAN

ISO and CCITT standardize the Directory service as one of the OSI application service elements for management. We designed the Directory software to realize DAP(Directory Access Protocol), and implemented it on the Engineering Workstation. The software of Directory consists of DUA library that provides the means of access to the Directory, DIB library that provides the means of access of Directory information , and DSA that processes the request from DUA. And we designed the DIB as an Object Oriented Data Base. On Implementing the Directory software, we could develop it efficiently by using the developed software, such as the ASN.1 tool "APRICOT", the object oriented language "superC", and achieved to make software architecture portable and expansible.

1. はじめに

ＯＳＩ(Open Systems Interconnection)の普及、発展に伴い、各種の応用層プロトコルの標準化も順調に進み、各地で実装が試みられている。実装に伴い、計算機やプリンタ、データベース等のネットワーク上の資源の管理方法を検討する必要が生じてきている。ＩＳＯ(Organization for International Standardization)及びＣＣＩＴＴ(Consultative Committee for International Telegraph & Telephone)では、ＯＳＩの応用層サービスの中にネットワーク管理サービスの一つとして、ディレクトリサービス[1]を標準化している。ディレクトリサービスとは、ネットワークの構成要素の物理的な位置等の情報を管理するサービスであり、名前とアドレスの変換等を行なう機能を持っている。

今回、我々は、ディレクトリを設計したので、これを報告する。第２章では、ＯＳＩディレクトリの概要について、第３章ではディレクトリの設計方針を、第４章ではディレクトリのシステム構成を主にソフトウェア構成の面から報告する。さらに、第５章で今回の設計に関する考察を述べる。

2. ＯＳＩディレクトリの概要

本章では、ＯＳＩディレクトリについて、モデル、機能、プロトコル、動作と情報の側面から概説する。

2.1. ＯＳＩディレクトリのモデル構成

ディレクトリのモデル構成を図１に示す。

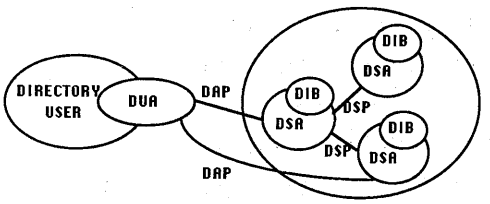


図１：ディレクトリモデル

ディレクトリは、次の３つの部分から構成される。

(1)ＤＵＡ(Directory User Agent):
ＤＵＡは、ディレクトリの情報にアクセスする応用プロセスである。ユーザがディレクトリを利用するための窓口としての役割を果たす。

(2)ＤＳＡ(Directory System Agent):
ＤＳＡは、ディレクトリサービスを提供する応用プロセスである。ＤＵＡから要求されたオペレーションを、ＤＩＢにアクセスすることによって処理する。

(3)ＤＩＢ(Directory Information Base):
ディレクトリサービスを提供するために、必要な情報を持つ各エンティティのアドレス情報等を格納するデータベースである。ＤＩＢに格納される情報は、ＤＩＴ(Directory Information Tree)と呼ばれる木構造を構成している。

尚、図の中のDAPはDUA-DSA間のプロトコルを、DSPはDSA-DSA間のプロトコルを示す。DAP,DSPの詳細な説明は、2.4節で述べる。

2.2. ＯＳＩディレクトリの機能

ディレクトリは、ＯＳＩに準拠したシステムのネットワーク管理機能の１つであり、電話番号の問い合わせ及び更新に相当するサービスを提供する。すなわち、ネットワークの構成機器の物理的、あるいは論理的な名前から、その物理的、論理的なアドレスを照会する等の機能を持っている。

具体的な機能としては、ディレクトリ情報の(1)照会、(2)変更、(3)追加、(4)削除の４つが挙げられる。ここで、ディレクトリ情報とは、複数のオープンシステムがネットワークで結ばれるための物理的なシステムの情報のことである。ディレクトリの持つ情報については、2.3節で詳しく述べる。

2.3. ＯＳＩディレクトリのもつ情報形態

ディレクトリでは、すべての情報がＤＩＢに格納されている。ＤＩＢは、複数のエントリの集合から構成される。

本節では、エントリとエントリの名前、及びＤＩＢの情報構造について説明する。

(1)エントリ
エントリとは、あるひとつの対象（オブジェクト）についての情報を含んでいる。エントリは、一つ以上の属性から構成される。図２に、ＯＳＩで規定されているエントリの構造を示す。エントリの属性は、オブジェクトに関する情報を表しており、１つの属性は、属性値と属性型から構成されている。属性型は、情報の種類を表し、属性値は、属性型で示される情報の実体を表している。

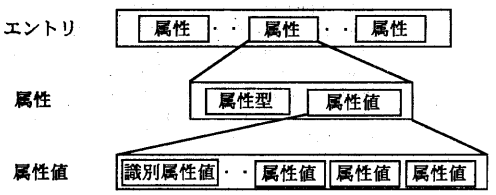


図２：エントリの構造

(2)名前

名前は、あるオブジェクトを識別するためのものである。ディレクトリ内で定義される名前は、次の３種類である。

(a)Distinguished Name（固有名）

オブジェクトの名称の一つ。固有名は、ディレクトリ内で一意でなければならない。エントリとその上位のエントリの相対固有名の並びで構成される。

(b)Relative Distinguished Name(RDN:相対固有名)あるクラスの中で識別可能な名前。一組の属性型と属性値によって定義される。

(c)別名

ディレクトリの利用者が、あるエントリにアクセスするときに、固有名以外の名前でアクセスしたいことがある。これを可能にするために、ディレクトリでは、一つのオブジェクトに複数の別名を設けることを認めている。別名によって、ディレクトリにアクセスした場合、システム内で別名を固有名に変換してから、要求を処理する。エントリとして別名を持つノードは、リーフノードでなければならない。

(3)情報構造

ＤＩＢで保持されているディレクトリの情報の構造は、ＤＩＴと呼ばれる階層的な木構造を成している。図３に、ＯＳＩで規定されているＤＩＴの構造例を示す。ディレクトリの情報は、ルートエントリからリーフエントリに到るパスの名前(固有名）によって一意に識別できる。

2.4. ＯＳＩディレクトリのプロトコル

ディレクトリのプロトコルは、次の２つに分けられる。

(1)ＤＡＰ(Directroy Access Protocol):

DUAとDSAの間の要求と結果の交換を定義するプロトコル。分散ディレクトリをサポートしないもの。

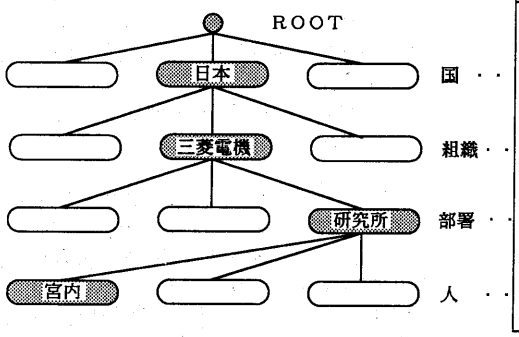
(2)ＤＳＰ(Directory System Protocol):

２つのDSAの間の要求と結果の交換を定義するプロトコル。

ディレクトリモデルにおける上記の二つのプロトコルの適用範囲を図１に示す。ＤＡＰをサポートするディレクトリのオペレーションを表１に示す。尚、表中の種類の項では、以下の記号を使用している。

表１：ＤＡＰをサポートするディレクトリのオペレーション一覧

オペレーション 名称	種類	機能
bind	A	DUA-DSA間にアソシエーションを設定する。
unbind	A	DUA-DSA間のアソシエーションを解放する。
read	Q	DUAがオブジェクト名を入力すると、 DSAがオブジェクトの属性を出力する。
compare	Q	DUAの入力した内容と DIBの内容を比較する。
abandon	X	要求したオペレーションの破棄を行なう。
list	Q	指定したオブジェクトエントリの下位のエントリの固有名を得る。
search	Q	指定したオブジェクトエントリの下位のエントリを検索する。
addEntry	M	DITにリーフエントリを追加する。
removeEntry	M	DITからリーフエントリを削除する。
modifyEntry	M	エントリの属性情報の更新を行なう。
modifyRDN	M	エントリの相対固有名を変更する。



相対固有名	固有名
{ }	{ }
国＝日本	国＝日本
組織＝三菱電機	{国＝日本 ,組織＝三菱電機}
部署＝研究所	{国＝日本 ,組織＝三菱電機 部署＝研究所}
人＝宮内	{国＝日本 ,組織＝三菱電機 部署＝研究所 人＝宮内}

図３：ＤＩＴの構造

A：アソシエーションに関するオペレーション

Q：問い合わせに関するオペレーション

X：以前に発行したオペレーションを破棄するオペレーション

M：更新に関するオペレーション

2.5. OSIディレクトリの動作

DAPをサポートするディレクトリの動作を図4に示す。

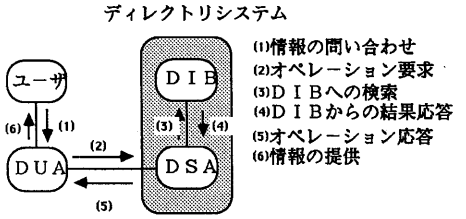


図4：ディレクトリの動作

ディレクトリの処理動作の手順を以下に示す。

(1) DUAが、ユーザから情報の問い合わせや更新の要求を受け付ける。（図中の(1)）

(2) DUAは、DSAとアソシエーションを設定する。（図中の(2)）

(3) DUAは、DSAにオペレーションの要求を発行する。（図中の(2)）

(4) DSAは、DUAからのオペレーションを処理するため、DIBにアクセスし、必要な情報を得る。（図中の(3)(4)）

(5) DSAは、DUAにオペレーションの結果を返信する。（図中の(5)）

(6) DUAは、DSAから結果を受け取る。（図中の(5)）

(7) DUAは、ユーザに問い合わせや更新の結果を報告する。（図中の(6)）

3. 設計方針

ディレクトリの設計に当たり、次のような方針を立てた。

(1)開発環境

ディレクトリのプログラムは、現在までに開発したOSIプログラム[2][3][4][5]（ACSE，ROSE，プレゼンテーション層、セション層のプロトコルを実現している）上に作成し、OSIに準拠したソフトウェア構成とした。以下に、使用したプロトコルとサービスを述べる。

(1)ACSE

ACSEの全サービスプリミティブを使用した。

(2)ROSE

ROSEのサービスプリミティブを使用した。 ROSEの機能には、アソシエーションクラスとオペレーションクラスの概念がある。

アソシエーションクラスは、オペレーションの要求を発行できる ROSEユーザを指定することによって、アソシエーションを分類する概念である。本ディレクトリでは、クラス1を使用した。すなわち、オペレーションを起動できるのは、アソシエーション設定を要求した ROSEユーザである。

オペレーションクラスは、同期モードと応答のパターンによって、オペレーションを分類する概念である。本ディレクトリでは、クラス2を使用した。すなわち、非同期モードで、応答のパターンは結果及びエラーである。

(3)プレゼンテーション層

プレゼンテーションプロトコルとして、カーネル機能単位を使用した。

(4)セション層

セションプロトコルとして、カーネルと全二重の機能単位を使用した。

(5)トランスポート層

トランスポートプロトコルとして、クラス0を使用した。

(6)ネットワーク層以下

ネットワーク層以下のプロトコルとしては、回線としてX.25を、LANとしてINP+LLCタイプ2+MACを使用した。

(2)ASN.1処理

応用層のプロトコルデータは、ASN.1(Abstract Syntax Notation One：抽象構文記法1）[6]で定義されている。ディレクトリのプロトコルデータを処理するために、当社で開発したASN.1ツールAPRICOT[7]を使用する。

APRICOTは、ASN.1抽象構文で記述されるデータの編集と処理を行なうためのツール群である。

APRICOTは、（1）ASN.1定義から階層構造を成す型定義テーブルを出力するプリコンパイラ、（2）階層構造データの符号化と復号化を行なうためのコーデックライブラリ、（3）プログラムの試験を行なう時に、ASN.1で記述されたデータの階層関係を解析し表示

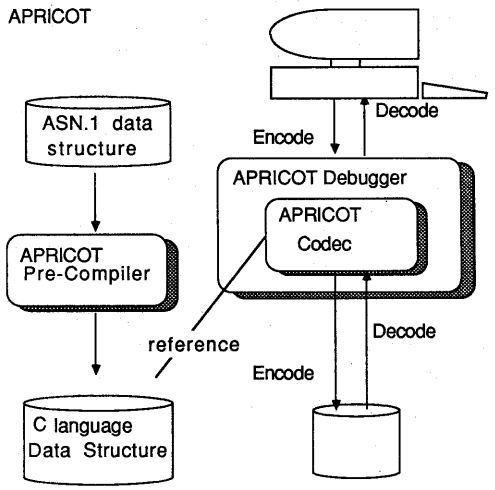


図5：APRICOTの構成

するデバッガ、の3つの部分から構成される。APRICOTの構成を図5に示す。

(3)言語

開発に当たっては、移植性を考慮して、当社で開発したオブジェクト指向言語 superC [8]を使用した。但し、DUAの部分は、C言語を使用した。

superCの処理系は、プリプロセッサとして実現されており、その生成コードがC言語であるので、移植性に優れており、さらにC言語との混合記述や静的束縛（コンパイル時にメソッドの探索を実行する）の機能を提供しているので、通常のオブジェクト指向言語で懸念される性能への影響も最小限に押さえることができる。

(4)実装範囲

国際標準にはDAPとDSPの2つのプロトコルが規定されているが、第一段階としてはDAPをサポートし、第二段階としてDSPをサポートすることとした。今回の実装ではDAPのみをサポートする。

また、実装に当たり、MAP3.0[9]を満足する仕様とした。表2に、DIS(Draft International Standard)の仕様と、今回のサポートの範囲を示す。表中のOはサポート、Xは未サポートを表す。

(5)セキュリティ

セキュリティについては、ISO/DIS9594-8において種々の規定がなされているが、今回の実装では、単純認証（パスワード）のみとする。

表2：DISの仕様と今回の実装範囲

プロトコル	DIS	実装範囲
DAP & DSP	Bind	O
	Unbind	O
DAP	Read	O
	Compare	O
	Abandon	O
	List	O
	Search	O
	AddEntry	O
	RemoveEntry	O
	ModifyEntry	O
	ModifyRDN	O
DSP	DistributedRead	X
	DistributedCompare	X
	DistributedAbandon	X
	DistributedList	X
	DistributedSearch	X
	DistributedAddEntry	X
	DistributedRemoveEntry	X
	DistributedModifyEntry	X
	DistributedModifyRDN	X

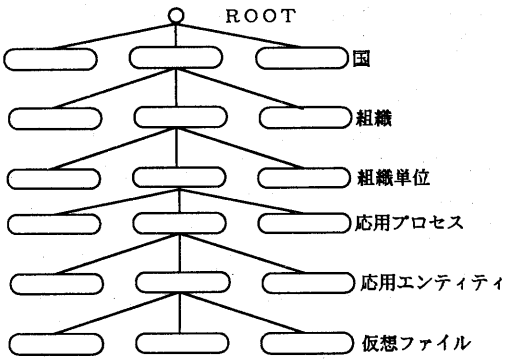


図6：FTAM用のDIB構成

(6)本ディレクトリの適用範囲

基本標準では、ディレクトリの構成は、使用目的に依存しないが、現時点では、具体的な適用規準が規定されていないので、以下の項目を適用範囲とする。

(1) FTAMにおける応用プロセス、応用エンティティーの名称とアドレスの関係。

(2) FTAMにおける仮想ファイル情報の読みだし、検索、変更。

従って、DIBに格納される情報は図6のような構成を持つ。

(7)アソシエーション数

DSAは、複数のアソシエーションの設定を可能とする。すなわち、複数のDUAからのオペレーション要求の受付が可能な仕様とする。

4. システム構成

システム構成を以下に述べる。

今回ディレクトリを実装した計算機は、当社のEWS（エンジニアリングワークステーション）であり、LAN及び回線によって接続されている。OSはUNIXである。今回の実装では、DUA，DSA，及び下位層を各々1プロセスとした。DUAと下位層、及びDSAと下位層の間では、プロセス間通信を行なう。

4.1. ソフトウェア構成

本ディレクトリを実現するソフトウェアは、以下の5つの部分から構成される。

(1)DUA
(2)DSA
(3)DIB
(4)下位層ソフトウェア
(5)システム管理ツール

ソフトウェア構成を図7に示す。なお、図の中のVOSは下位層を実現するOSIプログラムに対する仮想OSを示し、 MVOSは仮想OSへのアクセスモジュールを示す。

4.1.1. DUA

DUAの実現方式として、プロセスとするか、ライブラリとするかを検討した。検討の結果、次の理由から、DUAをライブラリとして実現した（以下、DUAライブラリと呼ぶ）。

(1) DUAが単独で動作することは余りなく、他のプロトコルから利用されることが多い。
(2) DUA自体のプログラムサイズが小さいと予想される。

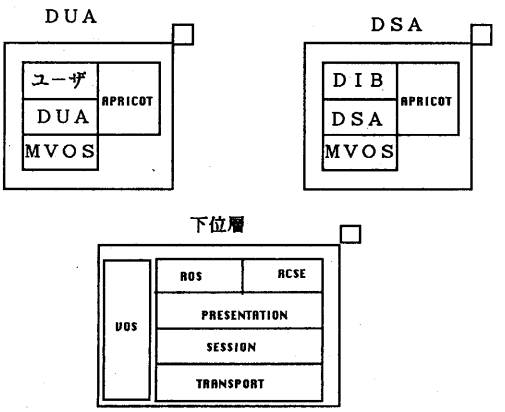


図7：ディレクトリのソフトウェア構成

DUAライブラリは、UNIX上のライブラリとして存在し、利用者プログラムに対して、ディレクトリへのアクセス機能をC言語の関数として提供する。

DUAライブラリは、利用者プログラムとAPRICOT、及びOSIプログラム(MVOS)とリンクされて、1つのロードモジュールを構成する。

DUAライブラリの下位層とのインタフェースは、 MVOSの提供するアクセス関数を使用して行なう。

DUAは、オペレーションの処理結果を受け取る前に、次のオペレーション要求を発行することができる。また、利用者プログラムの複数利用者への対応を想定して、複数のアソシエーションをサポートする。

4.1.2. DSA

DSAは、ネットワーク上のDUAからDIBへのアクセスを制御する。

DSAのプロセス構成として、図8に示す3種類の実現案を検討した。各モデルの特徴、及び長所と短所を表3に示す。

検討の結果、最終的には(3)に示すプロセス構成を目指すが、当面は設計の容易さを優先して、(1)に示すプロセス構成とした。ただし、(3)への移行を行ないやすいようなソフトウェア構成とした。DSAは、(1)のモデルの下で次のような機能を実現した。

(1) DSAは、1つのPSAP (Presentation Service Access Point)上に存在するが、複数のDUAからのオペレーションを処理する。
(2) アソシエーション数に関わらず、同時に処理できるオペレーションは1つである。オペレーションの処理中にDSAに到着したオペレーション要求は、キューイングされ、処理中のオペレーションの終了後にキューから取り出され、処理を実行する。

DSAは、UNIX上のロードモジュールとして存在し、DIBとAPRICOT、及びOSIプログラムをリンクして、1プロセスとして動作する。

DSAは、DIB及び下位層とのインタフェースを持つ。DIBとのインタフェースは、DIBライブラリの提供するアクセス関数を使用する。下位層とのインタフェースは、OSIプログラムの提供するアクセス関数を使用して行なう。

4.1.3. ＤＩＢ

ＤＩＢは、ディレクトリ情報を格納するためのデータベースを構築する必要がある。データベースの構築方法として、次の２つの方法がある。

案１：汎用データベースを利用する。
現在利用可能なデータベースは、リレーショナルデータベースである。商用のものを利用すれば、データベース自体の機能や安全性は保証される。

案２：専用データベースを自作する。
ディレクトリ専用のデータベースを構築すれば、ディレクトリ情報の汎用性が保証され、性能面でのオーバーヘッドも少ないと思われる。

また、ディレクトリの特徴として、以下のものが挙げられる。

特徴１：
更新が問い合わせに比べて非常に少ないこと

特徴２：
情報構造が木構造で、属性がそれぞれ異なる型を持つため、リレーショナルな表に写像することが困難であること

特徴３：
情報が様々な型からなる複雑な構造を取る。

以上のことから、案１を採用し、専用データベースを自作することとした。また、特徴２、３を考慮して、オブジェクト指向データベースをディレクトリ用に構築することとした。

オブジェクト指向データベースを自作するに当たり、記述言語として、 superCを採用した。

ディレクトリ情報のObject Class Definitionによって定義されるオブジェクトを、 superCの１つのクラスに写像した。各エントリは、同じObject Class Definitionを持っているが、固有の属性値を持っているので、 Object Classの構造を持つエントリをsuperCの１つのインスタンスに対応させた。

ディレクトリ情報の各属性は、superCのインスタンス変数に写像した。属性型がインスタンス変数の型に対応し、属性値がインスタンス変数の値に対応する。

属性は、ASN.1で記述されるため、APRICOTで記述されたデータ構造として表現した。 APRICOTによるデータ値は、階層構造を持っているので一様ではないため、 ASN.1符号化規則によって符号化された転送構文を実際のディスク格納データとする。

ＤＩＢは、ＵＮＩＸファイルシステム上に構築され、ＤＳＡがＤＩＢの情報にアクセスする手段は、ライブラリの形式で提供する。

表３：ＤＳＡのプロセスモデルの比較

番号	特徴	長所	短所
1	構造が簡単	構造が簡単	１つのオペレーションの処理中には、ＤＵＡからのオペレーション要求を検知できない
2	親ＤＳＡプロセスは、ＤＵＡからのbind要求を受けるのみで、以降のオペレーションの処理は子ＤＳＡプロセスが行なう	異なるＤＵＡから同時にオペレーション要求が到着してもオペレーションの処理時間には影響しない	１つのオペレーションの処理中には、他のＤＵＡからのオペレーション要求を検知できない
3	親ＤＳＡは、常にＤＵＡからのオペレーション要求を待ち続けてアソシエーションの管理のみを行ない、オペレーションの処理は、子ＤＳＡが行なう	オペレーションの処理中でも、ＤＵＡからのオペレーション要求を検知できる	ＤＳＡ内で、プロセス間通信のオーバーヘッドが常に存在する

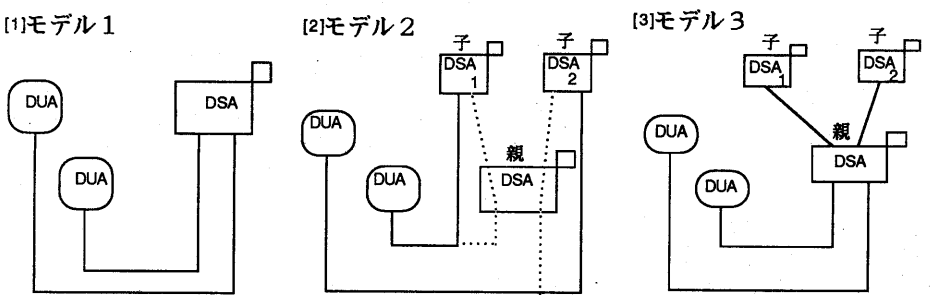


図８：ＤＳＡのプロセス構成モデル図

情報の提供形式は、superCによるメッセージ形式である。

ＤＩＢライブラリは、ＤＳＡとリンクされて、ロードモジュールを構成する。

ＤＩＢライブラリは、ＤＳＡが複数のプロセス構成に拡張された場合にも対応できるように、設計した。すなわち、複数のＤＳＡプロセスがＤＩＢを共有したときに、ＤＩＢへのアクセスの競合を防ぐように設計した。

4.1.4. 下位層ソフトウェア

下位層ソフトウェアは、当社で開発したＯＳＩプログラムを使用した。ＯＳＩプログラムは、基本参照モデルに基づいて、トランスポート層、セション層、プレゼンテーション層、ＡＣＳＥ，ＲＯＳＥの各サービスを提供する。ＯＳＩプログラムとのインタフェースには、「仮想OS(VOS)」を使用している。

4.1.5. システム管理ツール

ディレクトリを管理、運用するためのツールである。システム管理者が、ディレクトリをオフラインで使用するためのツールである。

システム管理ツールは、ＵＮＩＸ上のロードモジュールとして存在し、通信機能をコマンドの形式で提供する。

5. 考察

今回ディレクトリを実装するに当たり考慮した点は次のようなものである。

(1) ＭＡＰ3.0のディレクトリクラス0を満たすディレクトリを実現する。

(2) Ｃ言語を使用することによって、移植性の高いソフトウェアとする。

(3) ＤＳＡは、複数のＤＵＡからのオペレーション要求をサポートできる。
すなわち、ＤＳＡは、複数のアソシエーションの設定と制御を可能とする。

(4) ＤＩＢへの情報の格納方法は、APRICOTを利用して、転送構文の形で格納する。
従って、ＤＳＡは、ＤＩＢから受け取った情報を転送構文に変換することなく、下位層のユーザデータに格納することができる。

今回の設計では、ＤＡＰのみをサポートしたが、機能標準等が整い次第、ＤＳＰについてもサポートする予定である。ＤＳＰをサポートするためには、さらに次のような点を考慮する必要がある。

(1) ＤＩＢのディレクトリ情報及びＤＳＡの持つ知識情報 (Knowledge Tree)として、他のディレクトリシステム内のエントリ情報の位置を保持する必要がある。

(2) 複数のＤＳＡ間でオペレーションを処理する場合、ＤＳＡの保持する知識情報に矛盾があると無限ループに陥る可能性や、ＤＩＢの保持するディレクトリ情報に矛盾が生じる可能性があるので、このための対処方法を検討する必要がある。

(3) searchやlistオペレーションを処理する場合、複数のＤＩＢにエントリが分散されているので、ＤＳＡ間でのオペレーションの処理分担を検討する必要がある。例えば、複数のＤＩＢから得た情報のまとめ方などを決める必要がある。

6. おわりに

ＯＳＩで規定されているディレクトリを当社のＥＷＳに設計した。今後の課題としては、実装の評価、ＤＳＰをサポートする分散ディレクトリの開発、ＥＷＳ以外の機種への実装等を検討していきたいと考えている。

7. 参考文献

[1] ISO:DIS 9594 part1-8(1988).

[2] 新沢他:MNAにおけるOSIの実装方式(1), 昭和６２年前期情処大会5Z-1,(1987).

[3] 瀬戸他:MNAにおけるOSIの実装方式(2), 昭和６２年前期情処大会5Z-2,(1987).

[4] 安藤他:MNAにおけるOSIの実装方式(3), 昭和６２年前期情処大会5Z-3,(1987).

[5] 臼坂他:MNAにおけるOSIの実装方式(4), 昭和６２年前期情処大会5Z-4,(1987).

[6] ISO:IS 8824 (1987).

[7] 中川路他:ASN.1ツールAPRICOTの設計, 昭和６２年後期情処大会 5U-9,(1987).

[8] 勝山他：通信ソフトウェア向けオブジェクト指向言語superC,情報処理学会論文誌 ,Vol30,No.2(1989).

[9] General Motors Corporation :MANUFACTURING AUTOMATION PROTOCOL SPECIFICATION version3.0, C10A1(1987).