OSI ディレクトリシステムにおける DIB (ディレクトリ情報ベース) のオブジェクト指向アプローチによる実現†

中川路 哲男†† 勝山 光太郎†† 宮内 直人††
玉田 純††† 水野 忠則††

ディレクトリは，異機種間相互接続のためのネットワークアーキテクチャである OSI の応用層のサービス要素の一つであり，アドレスなどのネットワークに関する各種の情報を検索・更新するためのサービスを提供する．ディレクトリの国際規格における規定の対象は，ディレクトリ情報の抽象的なモデル化とそれを交換するための手順としてのプロトコルのみである．実際にディレクトリシステムを構築するには，ディレクトリを適用する形態に応じてディレクトリ情報の内容を決定し，ディレクトリ情報を格納し，アクセスサービスを提供する DIB を開発する必要がある．我々は，汎用的に規定されているディレクトリ情報のモデルをなるだけ忠実に実現するために，データモデルの柔軟性に優れるオブジェクト指向のアプローチで DIB を実現することを試みた．それにより，複雑な構造を持つデータを容易に扱い，汎用的なディレクトリのスキーマを制約なく実現することを狙った．DIB へのアクセス言語としては，オブジェクト指向言語 superC を採用してアクセス言語とデータベース間の親和性を図ると共に，属性の継承を実現したり，名前情報管理と属性情報管理の部分を独立させるなどソフトウェアのモジュール性を高めた．また，実際に構築したディレクトリシステムの性能を評価し，本アプローチの実用性を実証した．

1. まえがき

異機種間相互接続のためのネットワークアーキテクチャである OSI は，標準化の段階から実使用の段階に移りつつある．OSI の実用化のために重要な課題の一つとしてネットワーク管理があり，そのための応用層のサービス要素の一つとしてディレクトリ[1]がある．ディレクトリは，開放型システムの名前からアドレスを検索するなど，ネットワークに関する情報を検索・更新するためのサービスを提供する．今後のネットワークの大規模化に伴って，各システムで他のシステムに関する情報を保持することが困難になり，ディレクトリの必要性が高まると予想される．

ディレクトリの国際規格においては，ディレクトリ情報の抽象的なモデル化と，その情報を交換するためのプロトコルを規定している．ディレクトリシステムを構築するには，そのプロトコルを実装すること以外に，ディレクトリを適用する形態に応じてディレクトリ情報の内容を決定すると共に，ディレクトリ情報を格納し，アクセスサービスを提供する DIB (Directory Information Base) を開発する必要がある．例えば電子メールに適用される場合は人の名前やメールアドレスがディレクトリ情報となるし，分散データベースに適用される場合はデータ項目名やシステムのアドレスがディレクトリ情報となる．プロトコルは規格で規定されたとおりに実装すれば良いが，DIB はその構築方法から新たに検討する必要がある．

我々は，ディレクトリの適用形態，すなわち格納される情報に依存しないディレクトリシステムの構築方法を見いだすことを目的として，DIB の構築方法を検討した．このことはディレクトリのデータモデルの汎用性を損なわずに DIB を構築することに相当する．

従来の DIB におけるディレクトリ情報の格納方法として，単にファイル内の平坦な情報として保持する方法[2]や関係データベース (RDB) を利用する方法[3]が報告されている．しかし，ディレクトリにおけるデータのモデルは以下のような特徴を持つため，平坦な情報や表形式のデータ関係に直接写像することができず，情報構造に制約を設けたり，ディレクトリ情報の内容ごとに格納方法を検討するなどの対策を講じる必要があった．

- 情報の構文は ASN. 1 (Abstract Syntax Notation One)[4]で定義され，再帰構造などの複雑な構造を持ちうる．またその意味で，格納される情報ごとにその構文の範囲内で異なる動的な構造をとりうる．

† Development of the DIB for the OSI Directory System by the Object-Oriented Approach by TETSUO NAKAKAWAJI, KOTARO KATSUYAMA, NAOTO MIYAUCHI (Information Systems and Electronics Development Laboratory, Mitsubishi Electric Corp.), JUN TAMADA (Computer Works, Mitsubishi Electric Corp.) and TADANORI MIZUNO (Information Systems and Electronics Development Laboratory, Mitsubishi Electric Corp.).

†† 三菱電機(株)情報電子研究所
††† 三菱電機(株)コンピュータ製作所

● 一つのデータ項目が，順不同な，複数のデータ値を持ちうる．

一方，近年新しいデータベースの概念として，オブジェクト指向データベースが注目を浴びている[5)-7)]．オブジェクト指向データベースは，データの自然なモデリングにより，扱うデータの構造が複雑な場合に非常に有効であると言われている．我々は，ディレクトリにおけるデータ構造が複雑であることから，オブジェクト指向のアプローチで DIB を構築することを試みた．これにより，ディレクトリ情報への依存度を少なくし，適用形態の変更や拡張にも柔軟に対応できることを狙った．

本論文では，オブジェクト指向のアプローチによる DIB の実現を，データベースの構築とプロトコルソフトウェアとの結合の観点から論じ，その結果としての実装方式について述べ，さらにそれを評価した結果について報告する．

以下，第2章ではディレクトリの概要を紹介する．第3章ではオブジェクト指向データベースの設計とソフトウェアの機能・構成について述べる．第4章ではその例証として，FTAM（ファイル転送，アクセスおよび管理）プロトコルへ適用した場合の DIB の実現例を述べると共に，本方式の評価を行い，第5章で本報告のまとめを行う．

2. ディレクトリの概要

ディレクトリは，ISO などで開発されている OSI（開放型システム間相互接続）の応用層におけるサービス要素の一つである．ディレクトリの目的は，ネットワークに関する情報を検索・更新するためのサービスを提供することであり，ディレクトリを使用することにより，例えば通信したいシステムの名前からアドレス情報を獲得することなどが可能となる．

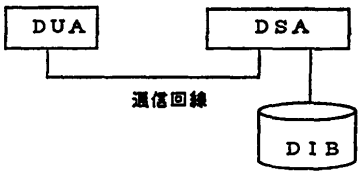

図 1 ディレクトリのモデル
Fig. 1 Model of the directory.

ディレクトリのモデルは，ネットワークに関する情報を格納する DIB，DIB にアクセスしてサービスを提供する DSA (Directory System Agent) と，そのサービスを利用する DUA (Directory User Agent) の三つの要素から構成される．DUA と DSA または，DSA 間の通信手順を規定したものがディレクトリプロトコルである．DIB は複数の DSA に分散して存在することもある．ディレクトリのモデルを**図1**に示す．

以下に DIB が格納するデータのモデルについて述べる．DIB は，情報をエントリ (Entry) という単位で管理しており，その包含関係を階層的に DIT (Directory Information Tree) なる木構造で表現している．エントリの型は，対象クラス (Object Class) と呼ばれる．エントリには，一意に識別されるための名前である DN (Distinguished Name) が付与されて

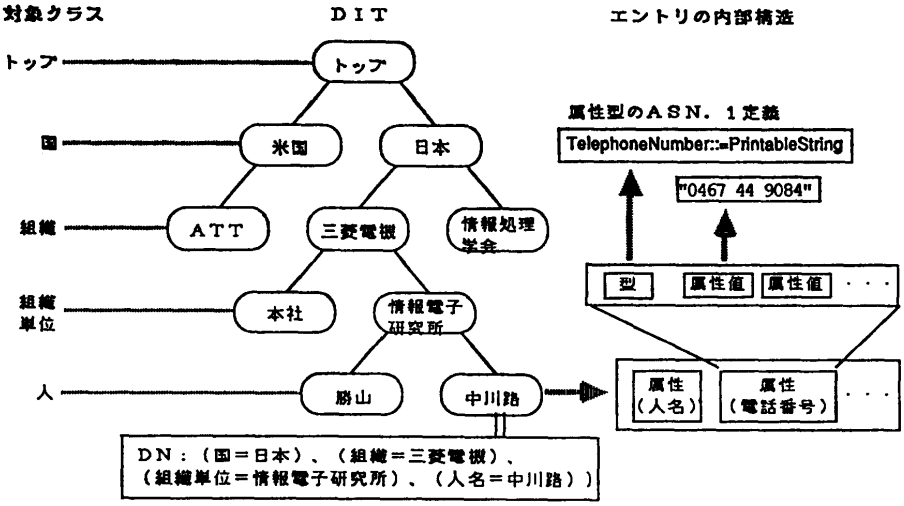

図 2 DIB の情報構造例
Fig. 2 An example of data structure in DIB.

表1 ディレクトリサービスで規定する操作
Table 1 Operations specified in the directory service.

属性情報の読み出し
属性情報の比較
属性情報の探索
属性情報の変更
RDN の変更
エントリの追加
エントリの削除
下位エントリの一覧
操作の破棄

おり，DN は木構造における相対パス名である RDN (Relative Distinguished Name) の並びである．エントリは，いくつかの属性 (Attribute) を持つ．持ちうる属性は，対象クラスごとに規定される．属性は一つ以上の値 (Value) を持ち，その構文は型 (Type) ごとに規定される．RDN も属性の一つである．また，エントリは，上位エントリの持つ属性を継承することができる．DIB 内の情報構造の例を図2に示す．

対象クラスの階層関係，対象クラスの持つ属性，属性の型と構文，属性値数や属性値長に関する制限，属性値比較時の一致規則などは，ディレクトリスキーマと呼ばれ，ASN. 1 記法[4]を用いて定義される．

ディレクトリサービスは，エントリを名前で指定し，それに対して各種のアクセスを行う操作手順を規定したものである．ディレクトリサービスで規定する操作を表1に示す．

3. オブジェクト指向データベースによる DIB

オブジェクト指向データベースは，オブジェクト指向プログラミングの概念にデータベースとしての機能を付加したものである．オブジェクト指向プログラミング言語に関する技術がかなり成熟してきたのに対して，オブジェクト指向データベースはまだ各所で研究が成されている段階にあり，汎用的な実現方法が定着していない．また，RDB における SQL のような汎用的な操作言語も出現していない．オブジェクト指向のデータモデルが概念的であるがゆえに，実現方法が困難となっている．我々は，ディレクトリへの適用を念頭に，オブジェクト指向データベースで必要と言われている以下の機能[8]を実現した．

- クラスとインスタンス
- データと手続きのカプセル化
- オブジェクトの名前付け
- オブジェクトの永続化と二次記憶管理
- オブジェクトの共用
- トランザクション機能
- 障害からの復旧
- 高速なアクセス

ディレクトリプロトコルを実装したソフトウェアに DIB のディレクトリ情報へのアクセスを実現するために，オブジェクトに対する操作をアクセスインタフェースの形で提供する必要がある．そのインタフェースとしては，オブジェクト指向データベースと適合するように，オブジェクト指向言語 superC[9] を採用することとした．また，アクセスインタフェースを提供するソフトウェアについては，superC におけるオブジェクトは同じプロセス内でのみ操作可能であることと，効率を考慮して，我々はそれをライブラリとして実現することとした．

以下，DIB アクセスライブラリにおける上記機能の実現方法について述べる．

3.1 クラスとインスタンス

オブジェクトのモデリングにおいては，対象クラスをクラス，その対象クラスのエントリをインスタンス，属性はクラスのインスタンス変数に対応づけた．この対応により，ある対象クラスのエントリは，同じ性質，すなわち同じ型の属性を持つが，それぞれの内容はインスタンスごとに異なるというオブジェクト指向の概念に写像することができた．エントリ間の属性の継承は，クラス間のインスタンス変数の継承という形でそのまま実現可能となる．例えば，トップという対象クラスの所属対象クラスという属性はすべての対象クラスに継承されるので，すべてのクラスをトップのサブクラスとして定義できる．

3.2 データと手続きのカプセル化

上記のようにクラスを設計した場合，そのデータである属性と手続きをカプセル化したものがオブジェクトとなる．一般にオブジェクト指向データベースでは，複雑なデータ構造に対するアクセスインタフェース，すなわちデータに対する手続きを汎用的に定義することが難しい．しかし，ディレクトリの場合は，通信回線を通じてディレクトリプロトコルで限定された操作でアクセスされることから，オブジェクトの提供する手続きとしては，表1のディレクトリ操作に対応するものをサポートすれば十分である．属性というデータとカプセル化する手続きとして定義した手続きと

表2 ディレクトリ対象クラスに対応するオブジェクトの提供する手続き
Table 2 Procedures provided by an object which corresponds to an object class in the directory.

手続き名	処理概要
インスタンス生成	インスタンスを生成，インスタンス変数を初期化／格納
インスタンス獲得	インスタンスを生成，インスタンス変数をロード，一時ファイルへの複写（更新が指定された時）
属性情報の獲得	指定された属性に相当するインスタンス変数の値を返却
属性情報の比較	指定された属性に相当するインスタンス変数の値と比較して結果を返却
属性情報の探索	フィルタを解析しつつ，指定された属性の情報とインスタンス変数にある属性情報との一致規則に対する整合を探索
属性情報の更新	属性追加時，インスタンス変数に属性と値を追加 属性削除時，削除可能であることを確認してインスタンス変数から属性を削除 属性値追加時，複数値が許されることと重複を検査してインスタンス変数の属性に値を追加 属性値削除時，指定された値があることを検査してインスタンス変数の属性から値を削除
RDN の変更	RDN 属性の変更，上位エントリに RDN 変更通知
下位エントリ一覧	下位エントリの RDN 情報を返却
コミット	一時ファイルの内容を属性情報ファイルに書き込む
インスタンス解放	属性情報ファイルのクローズ，インスタンスの解放
インスタンス消去	属性情報ファイルの消去，インスタンスの解放，上位エントリに削除通知

その処理内容を**表2**に示す．DSA は，DUA からディレクトリ操作要求を受信すると，DN に対応するインスタンスを生成し，それに属性情報の獲得，比較，更新のメッセージを送ることにより結果を得る．更新の場合は，後に述べるコミットメッセージを送る必要がある．また，探索操作において複数のエントリが対象となる場合には，下位エントリの一覧メッセージを使用して対象のエントリをリストアップしてから，それぞれに探索メッセージを送る．

3.3 オブジェクトの名前付け

インスタンスの名前はエントリの DN に相当するので，DN とインスタンスの対応を取る必要がある．そのために，DIT 全体の名前解析を行う名前管理オブジェクトを定義した．名前管理オブジェクトは，DN から該当するインスタンスを返すオブジェクトであり，名前管理クラスのインスタンスとして各 DSA に一つ存在する．名前情報の管理方法としては次の二つが考えられる．

1) 各エントリの情報とは独立に DIT 全体の名前情報を管理する．
2) 各エントリの RDN 属性を利用する．

指定された DN から対応するインスタンスを直接検索するには，1)の方法が性能上適していると考えられる．しかし，RDN も属性の一つであることを考慮すると，情報の一元管理という観点からは名前情報は分散されないことが望ましい．このために，我々は2)の方法を採用し，下位エントリの RDN を各エントリで保持することにして，下位エントリの RDN から該当オブジェクトを獲得する手続きを定義した．名前管理オブジェクトは，DN を RDN に分解し，各エントリに下位エントリ獲得メッセージを送ることを繰り返して，DN に該当するエントリを獲得して返却する．該当エントリが他の DIB にある場合には，その DIB 名を結果として返す．

クラスの提供する手続きには，RDN による下位エントリ獲得と属性へのアクセスの2種類がある．後者は，対象クラスの持つ属性によって異なるのでクラスごとに処理が異なる．一方，前者の処理はすべての対象クラスに共通の処理である．この点に注目して，我々は，RDN の管理を行うクラスのサブクラス（RDN 管理クラス）として，属性情報を管理する各クラスを実現した．

エントリは，対応する対象クラスの属性情報管理クラスのインスタンスとして生成されるが，上位クラスとしての RDN 管理クラスの性質を継承することになる．RDN 管理クラスは，自エントリの RDN，DIT における自エントリの上位および下位エントリの RDN 情報の管理を行い，RDN から該当インスタンスまたは他の DIB 名を返す手続きを持つ．RDN 情報は，複数の属性値を含むことがある．その場合に

は，RDN の解析においてそれぞれの属性値を含んでいることを比較する．属性情報管理クラスは，各属性に対する検索・更新操作を手続きとして提供する．

3.4 オブジェクトの永続化と二次記憶管理

オブジェクトを永続的なものにするために，生成したオブジェクトに関する情報を二次記憶に蓄積することとした．永続化に伴い，オブジェクトの獲得手続きに生成とロードの2種類を設け，生成時には新たなインスタンスを生成し，ロード時には DIB に格納されているインスタンスをメモリ上にロードしてアクセス可能にするという意味づけを行った．獲得は，いずれの場合もエントリの DN を指定して行う．逆の機能として，オブジェクトの解放手続きには，消去とクローズの2種類を用意し，消去時には永続的なインスタンス自体を消去，クローズ時には解放後再びロード可能な状態にするという意味づけとした．

オブジェクトに関する情報を二次記憶に蓄積するために，各エントリの属性情報をファイルに格納することとした．この場合，格納の単位として複数のエントリの属性情報を一つのファイルに対応させる方法と，一つのエントリの属性情報を一つのファイルに格納する方法が考えられる．複数のエントリに対する操作を同時に行うためには前者の方が性能上好ましいが，個々のエントリに対する操作の独立性という面からは後者の方が望ましい．

ディレクトリの利用形態として，複数のエントリに渡って検索を行うことより，一つのエントリを DN で指定して読み出しなどの操作を行うことが多いと予想されること，複数の DUA からの要求を非同期に処理する必要があることなどから，後者の方法を採用した．

情報格納形式としては，ディレクトリスキーマにおける属性型の ASN. 1 による定義をそのまま格納情報の定義とするために，ASN. 1 基本符号化規則[10]による転送構文形式を採用した．これにより，格納情報をそのまま通信路上に送出できる形式とすることが可能となった．また，この形式の採用で，属性値の長さや属性の値の数に制約を与えることなく，実装することが可能となった．ファイルへの格納，ファイルからの読み出し時には，変換処理として符号化・復号化が必要となる．それらの処理には，ASN. 1 ツール APRICOT[11] を採用することとした．

3.5 オブジェクトの共用

複数の DUA からの要求を非同期に行うためには，複数のプロセスにリンクされた DIB のアクセスライブラリからの，オブジェクトの共用を実現する必要がある．オブジェクトに関する情報の参照の場合は共用が可能であり，更新の場合は排他制御が必要となる．DUA から要求されたディレクトリの操作の種類によって参照か更新かが既知であるので，インスタンスの獲得時に更新の有無を指定することで，更新を行わないロードの多重化を実現した．更新を行うロードの場合は既にそのオブジェクトのロード操作が行われていれば失敗とし，更新を行うロード後の他からのロード操作も失敗とすることで，更新の排他制御を実現した．

3.6 トランザクション機能

ディレクトリにおける属性情報の更新において，あるエントリに関して複数の更新操作を行った場合，それらがすべて成功した場合以外は，すべての更新を行わないことが規定されている．これを，情報の更新に対するコミット操作を用意することで実現した．インスタンスを更新ありでロード後，各属性情報の更新を行い，そのすべての操作が成功した場合にはコミット操作を行ってから解放，一つでも失敗した場合はそのまま解放することで，この機能を実現した．

3.7 障害からの復旧

データベースの基本的な機能として，障害から一貫性を保って復旧することが要求される．一貫性の保持のために，インスタンス獲得時にあらかじめコピーした一時的なファイルに更新情報を書き込み，コミット操作時に本来のファイルに内容を複写することとした．障害からの復旧のためには，オフラインで DIB の内容を検査，バックアップ，生成するためのツールを用意した．

3.8 高速なアクセス

オブジェクト指向データベースでは，柔軟なデータモデルを扱える反面，データの内容に関する操作を統一できず，検索性能に対する工夫が要求される．我々は以下のような工夫を行った．

まず，オブジェクト指向言語におけるメッセージ束縛に関わる時間を最小にするために，superC の静的束縛機能を利用した．静的束縛機能は，送られるメッセージに対応する手続きを持つクラスを陽に指定する方法であり，プリコンパイル時に関数呼び出しに変換されるのでメッセージ束縛を高速に実行することが可能となる．

次の工夫は，格納形式を ASN. 1 基本符号化規則

```
PresentationAddress::=SEQUENCE {
    pSelector  [0] OCTET STRING OPTIONAL,
    sSelector  [1] OCTET STRING OPTIONAL,
    tSelector  [2] OCTET STRING OPTIONAL,
    nAddress   [3] SEQUENCE OF OCTET STRING}
```

(1) プレゼンテーションアドレスデータ型の構文定義
(1) Syntax definition of "Presentation Address" data type.

```
Attribute::=SEQUENCE {
    type    OBJECT IDENTIFIER,
    value   SET OF PresentationAddress}
```

(2) プレゼンテーションアドレス属性の構文定義
(2) Syntax definition of "Presentation Address" attribute.

```
Attribute::=SEQUENCE {
    type    OBJECT IDENTIFIER,
    value   SET OF ANY}
```

(3) 属性の構文を隠蔽した属性の構文定義
(3) Syntax definition of the attribute hiding the syntax of it.

```
Attribute::=SEQUENCE {
    type    OBJECT IDENTIFIER,
    value   ANY}
```

(4) 属性の数と構文を隠蔽した属性の構文定義
(4) Syntax definition of the attribute hiding the number and syntax of it.

図 3 ANY 型を利用した高速化の例
Fig. 3 An example of performance upgrade using ANY type.

による転送構文にしたことと，ANY というデータ型を利用することによるものである．ASN. 1 における ANY 型とは，任意のデータ型という意味であり，複雑な構造を持つデータ型であっても構わない．このことから，属性が複雑なデータ型の組合せである属性値の読み出しにおいて，属性を ANY 型としてファイルから読み出すことでその構造を意識する必要がなくなり，そのまま結果として返送することが可能となる．

図3の例において，プレゼンテーションアドレスという属性の型は，図3(1)のようなセレクタやアドレスの順序列として定義される．これは，属性という定義に当てはめると図3(2)のようになり，その符号化形式でファイルに格納されている．ここで，図3(3)の定義としてこの格納データを復号化して読み出すことで，プレゼンテーションアドレスの属性値が符号化された形式のまま読み出すことが可能となり，属性の型を意識することなく結果データとすることができる．

同様の工夫を，複数の属性値に対しても行った．属性の読み出しにおいて，複数の属性値がある場合にはこれらをすべて結果として返却する必要がある．これも図3(4)のように，複数の属性値全体を一つの ANY 型のデータとみなして復号化して読み出すことで，属性値の数を意識することなく結果データとすることを可能とした．

また，属性の読み出しにおいては，全属性の読み出しのほかに，読み出す属性を指定することがある．ファイルに格納されている属性情報を読み出す場合にはそれを復号化する必要があるが，読み出さない属性まで復号化するのは無駄である．この無駄を避けるために，APRICOT の要求時復号化機能を利用した．この機能は，読み出す必要のあるデータ要素に関係のある部分のみ，読み出し時に復号化を行う機能である．これにより，復号化処理の範囲を必要最小限にとどめることを狙った．

実現したアクセスライブラリのクラス構成と DIB のオブジェクト構成例を図4に示す．

4. FTAM への適用と評価

3章で述べたオブジェクト指向データベースによる DIB 実現の有用性を検証するために，適用目的を設定してディレクトリシステムを構築し，評価を行った．

4.1 FTAM への適用

我々は，適用形態として，以下の理由で，FTAM[12] によるファイル転送を利用するシステムにおいて，ファイル情報をディレクトリ情報として利用するためのディレクトリシステムを構築した．

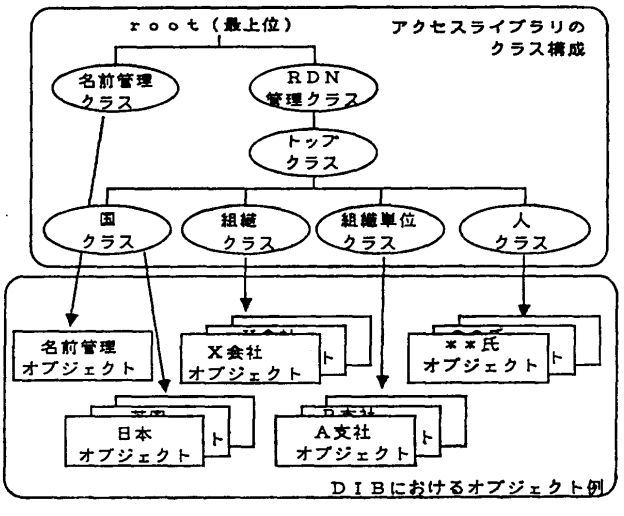

図 4 アクセスライブラリのクラス構成と DIB のオブジェクト例
Fig. 4 Class structure of the DIB access library and an example of objects in the DIB.

表 3 実現したディレクトリの構成と属性
Table 3 Summary of developed directory information.

対象クラス	保持する属性
トップ	所属対象クラス
国	国名
組織	組織名
組織単位	組織単位名
応用プロセス	応用プロセス名，説明（オプショナル）
応用エンティティ	応用エンティティ名，説明(オプショナル)，支援応用コンテキスト
仮想ファイル	仮想ファイル名，ドキュメントタイプ，許可動作，説明（オプショナル），作成日時(オプショナル)，最終変更日時（オプショナル）

- ディレクトリは電子メール[13]に適用するために開発され，従来のディレクトリシステムは電子メール用のものがほとんどであるため，他の目的にも使用できることを例証する必要がある．
- FTAM は国際規格として安定しており，我々も実装した経験を持つ[14]．
- FTAM において転送を行うには，相手システムにあるファイルの名前情報をあらかじめ知っている必要があるため，それをディレクトリに格納することにより，ディレクトリが有用なものとなる．
- FTAM における仮想ファイルは，仮想ファイル名，ドキュメントタイプ，作成日時などの様々な種類の属性情報を持つため，DIB の汎用性を検証するために適当である．

我々が想定したディレクトリの構成とそれぞれの属性を表 3 に示す．これを用いると，例えばある応用エンティティの下にある仮想ファイルのドキュメントタイプ情報を得ることなどが可能となる．また，実現した DIB アクセスライブラリのクラス構成を図 5 に示す．このクラス構成は設計したクラス構成に準じて

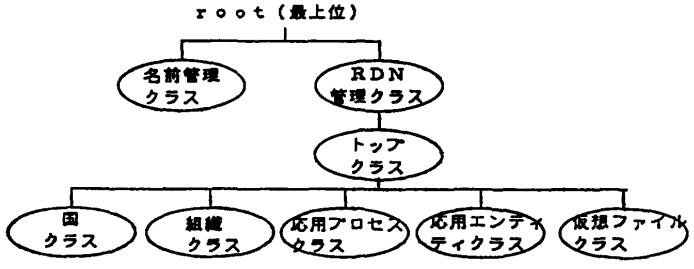

図 5 実現したアクセスライブラリのクラス構成
Fig. 5 Class structure of the DIB access library.

表 4 性能評価条件とその結果
Table 4 Condition and results of performance evaluation.

性能評価条件	
H/W	当社 EWS: ME 200 (CPU=68030)
格納エントリ数	対象クラスごとに 100 エントリ
読み出し条件	属性「説明」，「プレゼンテーションアドレス」の • 型のみの読み出し • 属性値が一つの場合の読み出し • 属性値が 10 個の場合の読み出し
下位エントリ一覧	下位の 100 エントリを一覧
探索条件	属性「説明」の最初の文字が一致する条件で探索
更新条件	属性「説明」の属性値の追加

操作	性能評価結果 (msec)	
インスタンス獲得	「国」：平均 150.0(最短 66.7, 最長 266.7)	
	「組織」：平均 233.3(最短 149.9, 最長 350.0)	
	「組織単位」：平均 305.4(最短 266.7, 最長 413.8)	
属性の読み出し	型のみ：24.7	「説明」「プレゼンテーションアドレス」共に同じ数値
	単一属性値：25.8	
	複数属性値：25.8	
下位エントリ一覧	183.0	
属性の探索	205.5	
属性の更新	233.3	

おり，RDN，継承した属性情報，固有の属性情報という 3 種類の，継承関係を持ったクラスに対応させている．名前管理クラスでは，DN から該当インスタンスを生成する手続きと獲得する手続きを記述した．RDN 管理クラスでは，RDN から該当インスタンスを獲得する手続きを記述した．他の各クラス定義では，属性情報格納ファイル名などをインスタンス変数とし，表 2 に示す各手続きを記述した．これにより，表 1 で示したすべてのディレクトリ操作が動作することを確認した．ただし，分散ディレクトリ操作，すなわち DSA 間のディレクトリ操作については，試験環境の関係で確認できていない．障害に対しても，UNIX のファイルを利用している関係上，完全な対障害性を実現できていない．

4.2 性能評価

性能評価は，エントリの獲得と，獲得したエントリに対する読み出し，下位エントリの一覧，探索，変更の操作につい

て行った．測定条件と測定結果を表4に示す．表より，エントリの獲得にはやや時間を要していることがわかる．特にDITのレベルの深いエントリにおいては，途中のエントリを獲得しながら名前解析を行うため，かなりの時間が必要となる．これは，DITの名前情報をエントリごとに属性情報として保持したことが原因と考えられる．また，エントリ獲得のための時間は，エントリ数に比例した．これも同じ原因と考えられ，名前解析の高速化のために，エントリの属性報報とは別に名前情報を保持することも検討する必要がある．

読み出し操作においては，型のみ，単一属性値，複数属性値の読み出しを，ほとんど同じ時間で実現できた．また，複雑な構造を持つ属性においても，ほとんど同様の時間である．これは，3.8節で述べた，属性の値をまとめてアクセスする方式の効果であり，属性の構造や属性値の数に依存せず，高速な読み出し操作を実現することができたことを確認した．

また，読み出しに比べて探索にはやや時間を要している．これは，探索において探索範囲内のすべてのオブジェクトや属性に対して，読み出しと比較の操作を行っているためと考えられる．これはディレクトリにおける探索操作が，汎用的なフィルタ操作の形で提供されており，それを忠実に実現しているためである．高速な探索を提供するには，特化した規定を追加する必要がある．

さらに，変更では時間を要しているが，これはディレクトリスキーマとの整合性チェック，既存属性との整合チェック，トランザクション機能による排他制御，コミット制御を行っているためである．更新操作については高速性は要求されないので，これで十分な値と考えている．

我々が過去に行ったFTAMの実装では，最初に相手システムとコネクションを確立するのに約770 ms要している[14]．しかもこれは高速なLAN環境での値であり，X. 25などの低速な回線ではそれ以上の値となる．これに対して，ディレクトリから属性情報を読み出す時間は，エントリ獲得を含めて約100〜300 ms程度である．コネクション確立時間と比較した結果，本ディレクトリシステムの性能は実用に耐えるものと判断した．

また，ソフトウェア量に関しては，名前管理クラスが約8 Kstep，RDN管理クラスが約17 Kstep，その他の対象クラスが属性数を n とすると約 $(10+2*n)$ Kstepであった．これらはsuperCによるステップ数であるが，ほぼ同じステップ数のC言語によるプログラムに変換される．各クラスにRDNの情報を保持させるために，RDN管理クラスがかなり大きなモジュールとなっており，DIBの実現において名前関係の処理が複雑であることを示している．一般の対象クラスは，属性数に比例してソフトウェア量が単純増加するが，継承する属性は含まれないので，特には問題にならないと思われる．

4.3 オブジェクト指向データベースの評価

名前管理クラスには実際の名前情報を持たせないため，ディレクトリスキーマや適用形態に依存しないクラスとすることができた．また，エントリの属性情報管理を，すべてのエントリに共通のRDN管理，上位から継承した属性情報管理，自分固有の属性情報管理という3種類に分離し，それぞれを継承関係を持ったクラスに対応させることで，一般にオブジェクト指向の利点と言われるソフトウェアのモジュール化を図ることができた．

我々は，国際規格におけるデータモデルの汎用性を損なわずにDIBを構築する方法を見いだすことを目的とした．ここでは，FTAM用のDIBを別の目的のDIBに適用する場合に必要な変更処理から，本方式の汎用性を論じる．

新たな対象クラスを追加する場合，RDN管理と継承する属性に関しては既存のクラスをそのまま再利用可能である．固有の属性情報を管理するクラスにおいても，提供する手続きの種類は同じであり，保持する属性に関わる部分を変更するだけで容易にクラス定義を作成することができる．保持する属性の構文自体はAPRICOTが処理するので，単に属性値に関する制限などを実現するだけでよい．

このように，ソフトウェア構成はディレクトリ情報に依存せず，しかもクラス構成において，ディレクトリ情報に依存するものと依存しないものが明確に区別されており，本方式は適用形態への依存性を最小・局所化していると言うことができる．

4.4 OSI ディレクトリ規格の評価

OSIディレクトリ規格で規定されたDUA-DSA間のすべての操作を実現し，それを確認することでその正当性を検証することができた．また，FTAM用のディレクトリ情報を交換することができ，ディレクトリが電子メールのみならずFTAMにも適用可能であることを実証した．このことは，ディレクトリ規格

の汎用性を示したものと言える．ただし，以下の点については，さらに規格としての改善が必要であることが明らかとなった．

- 名前と属性の関係：名前も属性の一部であるため，他の属性との違いが明確でない．例えば，名前の変更は属性変更操作でも可能となり，実現処理が複雑となる．また，どの属性が RDN に相当するのかを示す情報がない．
- ASN. 1 の IMPLICIT 記法を利用していないため，プロトコルデータ定義の中で最適でないものがある．
- 探索操作 (search) では，DIT の部分木が指定された場合，すべてのエントリのすべての属性を探索する必要がある．単純な探索のためにも，処理時間を要する可能性がある．
- プロトコル誤りや下位プロトコルである ROS（遠隔操作）における誤り発見時の処理が規定されていないため，相互接続上問題となる可能性がある．

5. あとがき

本論文では，オブジェクト指向データベースの概念を用いた，OSI ディレクトリのための DIB の実現について述べた．OSI ディレクトリ用に特化した形で，オブジェクト指向データベースの概念を利用することで，適用形態への依存性を最小・局所化し，ディレクトリスキーマの変更や拡張に柔軟に対応できる汎用的なソフトウェアの構築方法を確立することができた．さらに，性能面での実用性も実証することができ，本構築方法の有効性を示した．

今後は，探索効率の向上，分散ディレクトリ操作での実証などを行っていく予定である．

参考文献

1) ISO 9594: Information Processing Systems—OSI—The Directory (1989).
2) Robbins, C. J. et al.: The ISO Development Environment User's Manuals, Vol. 5: QUIPU, University College London (1990).
3) 小花，西山：リレーショナル型データベースを用いた OSI ディレクトリ情報ベース (DIB) の実現と評価，第 42 回情報処理学会マルチメディアと分散処理研究会，89-MDP-42-12 (1989).
4) ISO 8824: Information Processing Systems—OSI—Specification of Abstract Syntax Notation One (ASN. 1) (1988).
5) Fishman, D. H. et al.: Iris: An Object Oriented Database Management System, *ACM TOOIS*, Vol. 5, No. 1, pp. 70-95 (1987).
6) Smith, K. E. et al.: Intermedia: A Case Study of the Differences between Relational and Object-oriented Database Systems. *Proc. of ACM Conf. on OOPSLA*, pp. 452-465 (1987).
7) Andrews, T. et al.: Combining Language and Database Advances in an Object-oriented Development Environment, *Proc. of ACM Conf. on OOPSLA*, pp. 430-440 (1987).
8) Francois, B. et al.: The Object-Oriented Database System Manifesto, *Proc. of DOOD 89*, pp. 40-55 (1989).
9) 勝山，佐藤，中川路，水野：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，Vol. 30, No. 2, pp. 234-241 (1989).
10) ISO 8825: Information Processing Systems—OSI—Specification of Basic Encoding Rules for Abstract Syntax Notation One (ASN. 1) (1988).
11) 中川路，勝山，宮内，水野：OSI 抽象構文記法支援ソフトウェア APRICOT の開発と評価，電子情報通信学会論文誌，Vol. J 73-D-I, No. 2, pp. 225-234 (1990).
12) ISO 8571: Information Processing Systems—OSI—File Transfer, Access and Management (1988).
13) ISO 10021: Information Processing Systems—Message Oriented Text Interchange Systems (1988).
14) 中川路，勝山，芥川，水野：国際標準に準拠したファイル転送プロトコルの実現と評価，情報処理学会論文誌，Vol. 29, No. 11, pp. 1071-1078 (1988).

(平成 2 年 1 月 19 日受付)
(平成 2 年 12 月 18 日採録)

中川路哲男（正会員）

昭和33年生．昭和56年3月東京大学電気工学科卒業．昭和58年3月同大学大学院工学系研究科電気工学専攻修士課程修了．同年三菱電機(株)入社．現在同社情報電子研究所システム技術開発部に勤務．OSI通信ソフトウェアを中心とする分散処理システムの構築およびソフトウェア工学に関する研究・開発に従事．昭和63年9月情報処理学会全国大会において学術奨励賞授賞．電子情報通信学会会員．

勝山光太郎（正会員）

昭和29年生．昭和51年3月大阪大学基礎工学部制御工学科卒業．同年三菱電機(株)入社．現在同社情報電子研究所システム技術開発部．入社以来通信ソフトウェアの開発，分散処理システムの研究開発に従事．電子情報通信学会会員．

宮内　直人（正会員）

昭和38年生．昭和62年中央大学理工学部物理学科卒業．同年三菱電機(株)入社．現在，情報電子研究所において，OSI高位層に関する研究に従事．

玉田　純（正会員）

昭和36年生．昭和58年3月早稲田大学理工学部電気工学科卒業．同年4月三菱電機(株)入社．現在同社コンピュータ製作所コンピュータ方式統轄部に所属．OSIを中心とした通信ソフトウェア製品の開発に従事．

水野　忠則（正会員）

昭和20年生．昭和43年名古屋工業大学経営工学科卒業．同年三菱電機(株)入社．現在同社情報電子研究所システム技術開発部．工学博士．情報通信システムおよび分散処理システムに関する研究・開発に従事．著書としては『マイコンローカルネットワーク』(産報出版)，『分散処理システム・デザイン』(共訳，工学社)，『電子メールとメッセージ通信』(監訳，工学社) などがある．電子情報通信学会，オフィスオートメーション学会，日本経営工学会，IEEE 各会員．