特 別 研 究 報 告 書

題 目

IMSにおけるXDMSを用いた文書管理システム

指導教員

報 告 者

須賀院 吉伸

岡山大学工学部 情報工学科

平成22年2月5日 提出

要約

現在，Extensible Markup Language(XML) は，アプリケーション間のデータ交換フォーマットとして，広く利用されている．異なる環境間におけるドキュメント共有やアプリケーション間の互換性の問題を解決するため，オープンなファイルフォーマットを策定する動きがあり，様々な XML ベースのファイルフォーマットが登場している．また，次世代ネットワーク (NGN) における IP Multimedia Subsystem(IMS) は，XML Document Management Server(XDMS) を提供している．XDMS は携帯電話の通信記録や IM(Instant Messaging) のコンタクトリストなどを保存，管理する目的で使用される．このように，XDMS は，小さなデータを保存，管理することが想定されており，一般のオフィス文書を XDMS に保存，管理することは，考慮されていない．

本論文では，XDMS と IMS の概要を述べ，XDMS の操作インタフェースである XCAP について述べた．XDMS を DBMS として利用することを検討し，中小規模システム内では，XDMS を DBMS として利用可能であることを示した．また，文書管理システムで XDMS を利用することの利点と問題点を明らかにした．

XDMS に文書管理用の XML 文書を導入することより，XDMS 上で仮想的なディレクトリ操作および文書ファイルのディレクトリ間移動操作が実現できることを示した．また，この XML 文書を参照することで，文書ファイルの一覧を取得できることを示した．文書ファイルの排他制御機構について検討した．排他制御機構には，テストアンドセット操作が必要であることを示し，PUT 方式と DELETE 方式のテストアンドセット操作を提案した．また，この 2 つのテストアンドセット操作を比較し，DELETE 方式より PUT 方式が優れていることを示した．また，XDMS によっては，PUT 方式が使用できず，DELETE 方式を使用する必要があることを示した．実装した文書管理システムのシステム構成を示し，動作概要を述べた．

目次

図 目 次

表 目 次

第 1 章

はじめに

Extensible Markup Language(XML) は，アプリケーション間のデータ交換フォーマットとして，広く利用されている．近年では，異なる環境間におけるドキュメント共有やアプリケーション間の互換性の問題を解決するため，オープンなファイルフォーマットを策定する動きがあり，様々な XML ベースのファイルフォーマットが登場している．例として，Office Open XML(OOXML)，Open Document Format(ODF)，Scalable Vector Graphics(SVG) などがある．また，一般的なオフィススイートにおいても，標準ファイルフォーマットとして XML ベースのファイルフォーマットを採用する例が増えている．例として，Microsoft Office 2007，OpenOffice.org などがある．

XML は，データベース分野でも利用されてきた．XML によるデータベースは，拡張性が高く，柔軟なデータ構造を表現できる．しかし，Relational Database(RDB) と比較すると，データベースのサイズや検索速度など，性能面で劣る点が多い．また，SQL のような統一規格が存在せず，共通的なインタフェースによる検索が行えなかった．これらの点が，XML Database(XMLDB) の普及を妨げていた．しかし，近年では，ハードウェア性能の向上，XML Path Language(XPath) や XQuery など問い合わせ言語の策定など，技術の進歩により，XMLDB の問題点は解決されつつある．このような理由から，XMLDB 機能を持つ Database Management System(DBMS) が増えてきている．例として，IBM DB2，Oracle Database，Mircrosoft SQL Server がある．

次世代ネットワーク (NGN) における IP Multimedia Subsystem(IMS) は，XML Document Management Server(XDMS) を提供している．IMS において，XDMS は携帯電話の通信記録や IM(Instant Messaging) のコンタクトリストなどを保存，管理する目的で使用される．しかしながら，XDMS は小さなデータを保存，管理することが想定されており，一般のオフィ

ス文書を XMDS に保存，管理することは，考慮されていない．そこで，一般のオフィス文書をはじめとする XML 文書を XDMS で管理する際の利点と問題点を明らかにし，XDMS を利用した文書管理システムを検討する．

第 2 章

XML Document Management Server

2.1 概要

次世代ネットワーク (NGN) のコア技術である IP Multimedia Subsystem(IMS) では，IM[1] や Push-to-Talk over Cellular(PoC)[2] など様々なサービスがアプリケーションサーバ (AS) によって提供される．IMS で提供されるサービスには，複数の AS が連携するものがある．このようなサービスでは，各 AS が個別にユーザ情報や設定情報を管理すると，複数 AS 間での動作連携が困難となる．IMS では，このような共有情報を格納するため，XDMS を提供している．XDMS は，”OMA XML Document Management”[3] により定義されている．また，NGN では，IMS の外にも AS を設置できる．このため，AS による様々な付加サービスが期待できる．また，AS は，XML Configuration Access Protocol(XCAP) を用いて XDMS 上の XML 文書を操作する．XCAP は RFC4825[4] で定義されている．XDMS と AS の接続関係を図 2.1 に示す.

2.2 IP Multimedia Subsystem

IMS は，交換電話網，携帯電話網およびパケット通信網といった通信方式の異なる公衆通信サービスを Internet Protocol(IP) 技術と Session Initiation Protocol(SIP)[5] で統合し，マルチメディアサービスを実現する通信方式である.“ OMA IP Multimedia Subsystem V1.0 ”[6] によって標準化されている.

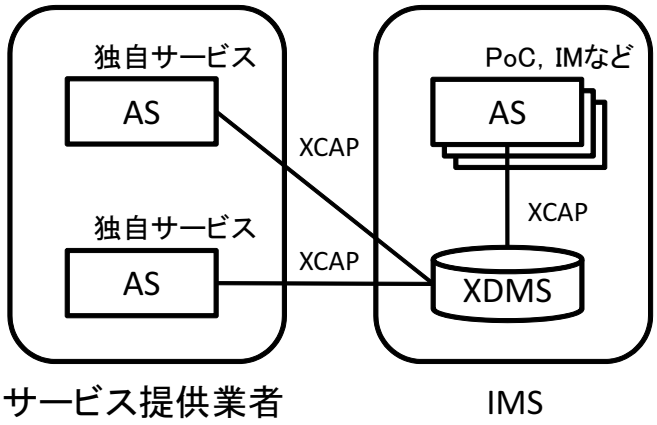

図 2.1 XDMS と AS の接続関係

2.3 XML Configuration Protocol

2.3.1 XCAP 概要

XCAP とは，XDMS 上の XML 文書を操作するプロトコルであり，HTTP をベースとしている．XDMS で管理される XML 文書を XCAP ドキュメント，XCAP ドキュメント内の XML 要素および XML 属性を XML ノードと呼ぶ．また，XCAP の操作対象を XCAP リソースと呼ぶ．XCAP リソースは，XCAP ドキュメント，XML ノードである．クライアントは，XDMS 上の XCAP リソースに GET，PUT，DELETE 要求を送信することで，XML リソースの取得，更新，削除を行う．また，XCAP では，AS が操作する XML 文書の XML スキーマや XDMS へのアクセス制限，URI の解釈などを Application usage として定義する．通常，XDMS は複数の Application usage に対応している．このため，各 Application usage を Application Unique ID(AUID) によって識別する．

2.3.2 URI 構成

XCAP リソースを特定するには，URI を用いる．XCAP における URI は以下の要素から構成される．

(1) XCAP ルート

XCAP ドキュメントが保管されているツリーのルートを表す．全ての XCAP ドキュメントは XCAP ルート以下に保存される．XCAP ルートは HTTP の URI として解釈される．

(2) ドキュメントセレクタ

特定の XCAP ドキュメントを表す．ドキュメントセレクタは，以下の形式で指定する．

```
[AUID]/users/[XUI]/[XCAP ドキュメント名]
```

AUID とは，XCAP ドキュメントが関連付けられているアプリケーションを識別するための文字列である．詳細は，2.3.7 項で述べる．XUI とは，XCAP ドキュメントの所有ユーザを識別するための文字列である．ドキュメントセレクタの例を以下に示す．

```
my-service/users/sugain/friends
```

上記の例では，アプリケーション“ my-service ”と関連付けられたユーザ“ sugain ”の XCAP ドキュメント“ friends ”を表す．

(3) ノードセレクタセパレータ

ドキュメントセレクタとノードセレクタを区切るための記号である．2 つのチルダ記号 (~~) である．

(4) ノードセレクタ

XCAP ドキュメント内の XML 要素または XML 属性を指定するためのクエリー式である．XCAP ドキュメントのルート要素から指定 XML 要素までのパスを指定する．また，同名の要素が複数存在する場合，属性値を用いて指定する．例として，`<a att="1" />` を指定する場合は，〜`/a[@att="1"]` と表現する．また，XML 属性も指定できる．例として，`<a att="1" att2="2" />`の属性“ att2 ”の値を指定する場合，〜`/a[@att="1"]/@att2` と表現する．

XCAP ドキュメントを表す URI は (1) と (2)，XCAP ドキュメント内の XML 要素または XML 属性を表す URI は (1)〜(4) で構成される．URI の例を以下に示す．

```
https://example.com/xcap/my-service/users/sugain/friends/~~/header
```

2.3.3 リソース操作

XCAP では，作成，更新および削除操作に対応した HTTP 要求を送信することで，XDMS 上の XCAP リソースを操作できる．各 HTTP メソッドと役割を表 2.1 に示す．

表 2.1 XCAP における HTTP メソッドの役割

HTTP 要求	XCAP における操作
GET	XCAP リソースの参照
PUT	XCAP リソースの作成，修正
DELETE	XCAP リソースの削除

2.3.4 XCAP リソースの参照

XDMS 上の URI に GET 要求を送信することで，XCAP リソースを参照できる．URI によって指定された XCAP リソースの種類によって，取得できるデータの種類が異なる．XCAP リソースの種類とレスポンスの内容を表 2.2 に示す．

表 2.2 GET 要求における各リソースとレスポンスの内容

リソースの種類	レスポンスの内容	レスポンスの MIME type
XCAP ドキュメント	XML ドキュメント	Application usage に依存
XML 要素	XML 要素	application/xcap-el+xml
XML 属性	XML 属性の内容 (テキスト形式)	application/xcap-att+xml

URI として XML 要素を指定し GET 要求を送信した場合，指定 XML 要素のタグを含む部分的な XML データが XDMS から返される．例を以下に述べる．

```
<?xml version="1.0" encoding="UTF-8" ?>
<index xmlns="http://swlab.cs.okayama-u.ac.jp">
  <head>
    <link href="http://example.com/a"/>
    <link href="http://example.com/b"/>
  </head>
  <directory/>
</index>
```

上記の XCAP ドキュメント内の要素“ head ”を参照した場合の取得結果を以下に示す．

```
<head>
  <link href="http://example.com/a"/>
  <link href="http://example.com/b"/>
</head>
```

2.3.5 XCAP リソースの作成，置換

XDMS 上の URI に PUT 要求を送信することで，XCAP リソースを作成または置換できる．XDMS は，PUT 要求を受け取ると以下の処理を行う．

(1) 指定 XCAP リソースが存在しない場合，メッセージボディの内容を持つ新たな XCAP リソースを作成する．

(2) 指定 XCAP リソースがすでに存在する場合，指定 XCAP リソースの内容をメッセージボディの内容へ置き換える．

また，指定された XCAP リソースの種類によって，メッセージボディの内容が異なる．XCAP リソースの種類とメッセージボディの内容の対応は，表 2.2 と同様である．

2.3.6 XCAP リソースの削除

XDMS 上の URI に DELETE 要求を送信することで，XCAP リソースを削除できる．

2.3.7 Application usage

Application usage とは，XCAP リソースに対して定義されたデータ構造や制約の集合である．RFC4825 では，Application usage の内容と文書化に関する規約が定義されている．Application usage を文書化する際に記述しなければならない項目を以下に示す．

(1) Application Unique ID

AUID(Application Unique ID) とは，各 application usage を識別するためのユニークな識別子である．AUID には 2 つの名前空間が用意されている．2 つの名前空間を以下に示す．

(A) IETF namespace

Internet Assigned Numbers Authority(IANA)[7] に登録されている AUID を含む名前空間である．各 AUID は，RFC 文書と関連付けられている．例として，“ xcap-caps ”や“ pres-rules ”などがある．なお，“ xcap-caps ”は，RFC4825，“ pres-rules ”は RFC5025[8] で定義されている．

(B) vendor-proprietary namespace

IANA に登録されていない AUID を含む名前空間である．各 AUID はドメインを逆順にした文字列を接頭辞として持つ．例えば，“ example.com ”内の AUID は“ com.example.* ”となる．

(2) XML Schema

XML スキーマは XML 文書の構造定義である．XCAP サーバは，Application usage で定義された XML スキーマに従い XCAP リソースの妥当性を検証する．XML スキーマの定義には XML Schema[9] を使用する．

(3) Default Document Namespace

XCAP リソースのデフォルトネームスペースを定義する．

(4) MIME Type

スキーマに対応する MIME Type を指定する．

(5) Validation Constraints

XML Schema では定義できない制約を英語の文章として記述する．例として，“ xcap-caps ”のドキュメントの内容を以下に示す．

```
There are no additional validation constraints associated with this
application usage.
```

(6) Data Semantics

XML ドキュメント内のデータが持つ意味を英語の文章として記述する．この内容に従い，クライアントは XCAP リソースを操作する．

(7) Naming Conventions

XCAP サーバ上におけるドキュメント名の制約を定義する．“ xcap-caps ”のドキュメントの内容を以下に示す．

```
A server MUST maintain a single instance of the document in the
global tree, using the filename "index". There MUST NOT be an
instance of this document in the user's tree.
```

(8) XCAP リソースの依存関係

XCAP リソースの依存関係を英語の文章として記述する．例として,“ xcap-caps ”のドキュメントの内容を以下に示す.

```
There are no resource interdependencies in this application usage
beyond those defined by the schema.
```

(9) Authorization Policies

各XCAP リソースへのアクセス制御の規則を英語の文章として記述する．XCAP には,デフォルトのアクセス制御が定義されている．デフォルトのアクセス制御を以下に述べる.

(A) 各ユーザは自身のホームディレクトリ内の全ドキュメントを参照と更新および削除できる.
(B) グローバルディレクトリ内の全ドキュメントは全ユーザが参照できる.
(C) 特定の信頼されたユーザのみグローバルディレクトリ内のドキュメントを編集できる.

デフォルトのアクセス制御を利用する場合，Authorization Policies にデフォルトのアクセス制御を利用することを明記する．例として,“ xcap-caps ”のドキュメントの内容を以下に示す.

```
This application usage does not change the default authorization
policy defined by XCAP.
```

第 3 章

データベースとしてのXDMS

3.1 DBMSとしてのXDMS

本節では，XDMS を Database Management System(DBMS) として使用した場合の XDMS の性能について述べる．一般的な DBMS の要求に対して，XDMS の機能を用いて対処する場合，以下のような結果になる．

(1) 高速な処理

XDMS の内部的な構成は，定義されていない．通常，XDMS 上のデータは，DBMS によって管理される．このため，内部的には，大量のデータを高速に保存，検索，削除できる．しかし，データの通信に XCAP を使用するため，通信処理がオーバヘッドとなり，処理速度は低下する．

(2) 複数データベースの管理

XDMS では，各 AUID ごとに独立してデータを管理出来る．この AUID をデータベースと見なすことで，複数データベースの管理が実現できる．

(3) 検索機能

XCAP では，属性値をキーとして，XML ノードを検索できる．しかし，SQL，XQuery と比較し，単純な条件しか表現できない．また，複数の XCAP ドキュメントを同時に検索することが出来ない．なお，現在策定中である“ OMA XML Document Management V2.1 ”では，XQuery による高度な検索機能が追加される予定である．

(4) データの妥当性検証

XML Schema や各 XCAP リソースへの操作制限などを定義できる．これらの定義は，Application usage として定義する．

XDMS を DBMS として使用する場合，処理速度および検索機能に制限がある．このため，大規模なシステムでは，XDMS を DBMS として使用できない．しかし，処理速度があまり求められない中小規模のシステムの場合，XDMS を DBMS として使用できる．

3.2 文書管理における XDMS の利点と問題点

3.1 節では，中小規模のシステム内においては，XDMS を DBMS として使用できることを示した．本節では，中小規模の文書管理システム内において，XDMS を DBMS として使用した場合の利点および問題点について述べる．まず，利点を以下に示す．

(1) データストレージを AS 自身で保持しなくて良い．このため，低コストで AS を構築および運用できる．

(2) XCAP は，HTTP ベースのオープンなプロトコルであり，動作も単純である．このため，特殊なソフトウェアに依存することなくクライアント機能を実装できる．

(3) XDMS では，認証機能と各リソースへのアクセス権限管理機能を標準で提供している．また，これらの機能は，Application usage を記述することにより動作を制御できる．この機能を利用することにより，AS にデータ共有インタフェースを実装することなく，データを共有できる．通常，AS のデータを他の AS と共有する場合，データ共有インタフェースを実装する必要がある．このインタフェースは，複雑な機能を持ち，実装が難しい．

次に，問題点を以下に示す．

(1) XDMS では，ディレクトリ操作にあたる操作がない．このため，大量の XML 文書を階層構造のあるファイルシステム内の要素として管理しにくい．

(2) XCAP には，XCAP ドキュメントの一覧を取得する機能がない．このため，AS は URI を知らない XCAP ドキュメントへアクセスできない．

(3) XCAP には，XCAP リソースの排他制御機構がない．XDMS 上の XCAP リソースは，複数の AS から参照可能だが，XCAP リソースを更新する AS は，慣例的に 1 つである．このため，複数の AS から同時に更新されることが想定されていない．同一 XCAP リソースを更新する場合は，AS 間の排他制御機構が必要である．

3.3 問題点への対処

3.2 節で述べた問題を解決するため，XDMS 上にある文書ファイルの一覧を保持する XCAP ドキュメントを作成する．この XCAP ドキュメントをインデックスと呼ぶ．本文書管理システムの XDMS 上では，インデックスを除くすべての XCAP ドキュメントをユニークな ID で管理する．インデックスは，この ID と文書ファイル名の対応情報を格納する．また，インデックス内において各 XCAP ドキュメントは仮想的なディレクトリ構造内の要素として管理される．このディレクトリ構造は，XML ノードのツリー構造を用いて表現する．ディレクトリ構造の概要を図 3.1 に示す．

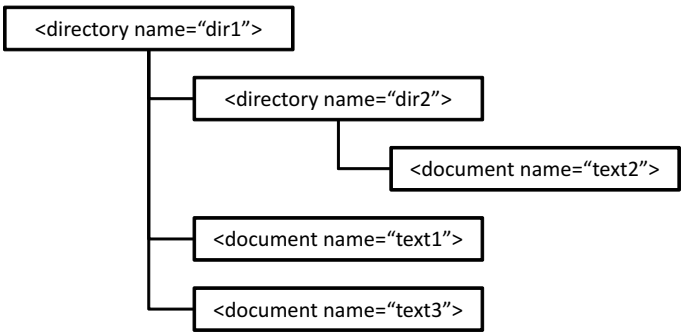

図 3.1 インデックス内のディレクトリ構造

インデックスの導入により，3.2 節で述べた問題点は以下のように解決できる．

(1) インデックス内で文書ファイルは，仮想的なディレクトリ構造の要素として管理される．このため，インデックス内の階層構造を編集することで，文書ファイルそのものに相当する XCAP ドキュメントに変更を加えることなく，ディレクトリ操作が実現できる．

(2) AS は，インデックスを参照することで，文書ファイル名の一覧及び各文書ファイル名に対応する ID を取得できる．

(3) インデックス内に排他制御を表す要素を用意して排他制御を行う．また，HTTP のステータスコードを利用し，テストアンドセット操作を実現する．

第 4 章

システム設計

4.1 文書の管理方法

本文書管理システムでは，管理対象を“ 文書ファイル ”と表現する.“ 文書ファイル ”は以下の情報を持つ.

(1) 本文 (プレーンテキストまたは XML データ)

(2) 名前

(3) ファイルの種類

(4) 作成者

(5) 作成日時

(6) 更新日時

(7) アプリケーション付加情報

また，本文書管理システムでは，文書ファイルをディレクトリ構造内の要素として管理する．ディレクトリ構造のないファイルシステム内で，文書ファイルを管理する際の問題点を以下に述べる.

(A) 名前の衝突

ディレクトリ構造がないファイルシステムにおいては，全ての文書ファイルが同一名前空間に配置される．このため，名前の衝突が頻繁に発生する.

(B) ファイルの集合の表現

ファイルの集合を表現することができない．このため，文書管理システムにおいて，ファイルの集合に対する操作が表現できない．

上記の問題点を回避するため，本文書管理システムでは，文書ファイルをディレクトリ構造内の要素として管理する．ディレクトリ構造内の文書ファイルは，文書ファイルパスで指定する．文書ファイルパスの例を以下に示す．

```
dir1/sub/sample.xml
```

4.2 インデックス

インデックスは，XDMS 上にある各文書ファイルの一覧を保持する．インデックスの例を図 4.1 に示す．各 XML 要素の内容を以下に述べる．

(1) directory 要素

仮想的なディレクトリを表す．

(2) document 要素

文書ファイルを表す．name 属性と id 属を持つ．これは，それぞれ文書ファイル名と対応する XCAP ドキュメントの ID を表す．

(3) lock 要素

その親要素に対応するリソースがロックされていることを表す．ただし，リソースをロックするためには，テストアンドセット操作を XCAP で実現する必要がある．テストアンドセット操作については，4.3 節で述べる．

(4) used-ids 要素

使用中の ID を子要素として保持する．

(5) used-names 要素

自身の親にあたるディレクトリ内で使用中の名前を子要素として保持する．

```
<?xml version="1.0" encoding="UTF-8" ?>
<index xmlns="http://swlab.cs.okayama-u.ac.jp/xdms">
  <head>
    <used-ids>
      <id id="5e3c"/>
      <id id="d2ab"/>
    </used-ids>
  </head>
  <directory>
    <used-names>
      <name name="sample.txt"/>
      <name name="sample_dir"/>
    </used-names>
    <document name="sample.txt" id="5e3c" />
    <directory name="sample_dir">
      <used-names>
        <name name="sample2.eml"/>
      </used-names>
      <document name="sample2.eml" id="d2ab"/>
    </directory>
  </directory>
</index>
```

図 4.1 インデックスの例

4.3 テストアンドセット操作

4.3.1 テストアンドセット操作の必要性

本文書管理システム上で，文書ファイルのロックを行う場合，以下の 2 つの操作を行う必要がある．

(1) 対象文書ファイルが，すでにロックされていないか確認する

(2) ロックされていない場合のみ，インデックスにロック情報を書き込む

上記 2 つの操作は，必ず連続して実行されなければならない．(1) の操作と (2) の操作の間に他のクライアントがロック操作を行うと，正常に排他制御できない可能性がある．このため，確認と書き込みを 1 つの操作で行う“ テストアンドセット操作 ”を XCAP を利用して実現する必要がある．本論文では，テストアンドセット操作の実現方法として以下の 2 つの方法を検討する．

(1) PUT 方式

(2) DELETE 方式

上記方法の詳細について，以降の節で述べる．

4.3.2 PUT 要求を用いたテストアンドセット操作

本方式で実現できるテストアンドセット操作を以下に述べる．

(1) 指定した XML ノードが存在するかどうかテストする．

(2) 指定した XML ノードが存在しない場合，指定 XML ノードを新規作成し，操作は成功する．

(3) 指定した XML ノードがすでに存在する場合，操作は失敗する．

XCAP では，XCAP リソースの参照と更新を一つの通信で行うことができない．このため，PUT 要求に対するレスポンスのステータスコードを用いて，テストアンドセット操作を実現する．XCAP では，新規作成および置換操作のために PUT 要求を送信する．PUT 要求を受け取った XDMS は，操作結果を HTTP レスポンスのステータスコードとしてクライアントへ返す．具体的には，指定 XCAP リソースが存在せず，新規作成した場合，ステータス

コードは“ 201 Created ”である．一方，指定 XCAP リソースがすでに存在し，置換した場合，ステータスコードは“ 200 OK ”である．このようにステータスコードの違いを利用して，指定した XML ノードが存在するかどうかを確認する．テストアンドセット操作を PUT 要求で実現する際の XCAP による操作手順を以下に述べる．

(1) クライアントは，任意の XML ノードを URI として指定し，PUT 要求を送信する．

(2) レスポンスのステータスコードが“ 201 Created ”の場合，テストアンドセット操作は成功である．

(3) レスポンスのステータスコードが“ 200 OK ”の場合，テストアンドセット操作は失敗である．

4.3.3 DELETE 要求を用いたテストアンドセット操作

本方式で実現できるテストアンドセット操作を以下に述べる．

(1) 指定した XML ノードが存在するかどうかテストする．

(2) 指定した XML ノードが存在する場合，指定 XML ノードを削除し，操作は成功する．

(3) 指定した XML ノードが存在しない場合，操作は失敗する．

上記のテストアンドセット操作の実現方法は，基本的に 4.3.2 項で述べた方法を同様である．まず，XCAP リソースの削除操作のために DELETE 要求を送信する．DELETE 要求を受け取った XDMS は，XCAP リソースが存在するかどうか確認する．XCAP リソースが存在する場合，XDMS は該当 XCAP リソースを削除し，レスポンスのステータスコードとして“ 200 OK ”を返す．XCAP リソースが存在しない場合，XCAP リソースの削除操作は失敗し，レスポンスのステータスコードとして“ 404 Not Found ”を返す．このように，レスポンスのステータスコードを利用して XCAP ノードの存在確認と削除操作を 1 つの通信で実現する．テストアンドセット操作を DELETE 要求で実現する際の XCAP による操作手順を以下に述べる．

(1) クライアントは，任意の XML ノードを URI として指定し，DELETE 要求を送信する．

(2) レスポンスのステータスコードが“ 200 OK ”の場合，テストアンドセット操作は成功である．

(3) レスポンスのステータスコードが“ 404 Not Found ”の場合，テストアンドセット操作は失敗である．

排他制御を DELETE 方式のテストアンドセット操作で実現する場合，図 4.1 の例とは異なり,“ lock ”要素ではなく“ unlock ”要素を用いる.“ unlock ”要素は，非ロック状態を表す要素である．この要素に DELETE 方式のテストアンドセット操作を行うことで，排他制御が実現できる．

4.3.4 PUT 方式と DELETE 方式の比較

PUT 方式は，任意の XML 要素に対してテストアンドセット操作できる．しかし，DELETE 方式の場合，あらかじめ用意しておいた XML 要素以外には，テストアンドセット操作できない．例として，使用中 ID リストを管理する場合を以下に述べる．

PUT 方式で，使用中 ID リストを管理する場合の使用中 ID リスト例を以下に示す．

```
<used-ids>
  <id id="5e3c"/>
  <id id="d2ab"/>
</used-ids>
```

上記の例は，ID“ 5e3c ”と ID“ d2ab ”が使用中であることを表す．新たに ID を確保する場合,“ used-ids ”要素の子要素となるように，テストアンドセット操作によって“ id ”要素を追加する．続いて，DELETE 方式で使用中 ID リストを管理する場合の使用中 ID リスト例を以下に示す．

```
<unused-ids>
  <id id="0000"/>
        .
        .
        .
  <id id="ffff"/>
</unused-ids>
```

DELETE 方式の場合，上記のように使用していない ID のリストを保持する.“ unused-ids ”要素の子要素として存在しない ID が使用中の ID である．このように DELETE 方式の場合，ID リストの初期化時に全使用候補の ID 一覧を“ unused-ids ”要素以下に持たなければなら

ない．このため，使用候補の ID が多い場合は，リストが大きくなる．また，ID が文字列の場合など，使用候補がほぼ無限にある場合，DELETE 方式は使用できない．DELETE 方式で使用中 ID リストを管理する場合,“ used-ids ”要素の排他制御を行い，以下に述べる手順で，使用中 ID を追加する．

(1)“ used-ids ”要素のロック

(2)“ id ”要素の追加

(3)“ used-ids ”要素のアンロック

ただし，この方法は，1 回の ID 追加処理に 3 回の通信が必要となり，非効率的である．

これらの問題があるため，PUT 方式の方が DELETE 方式に比べ優れている．しかし，XDMS の実装によっては，PUT 方式が利用できない．例として,“ Fraunhofer FOKUS XDMS ” [10] がある．“ Fraunhofer FOKUS XDMS ”は，XML 要素に対する PUT 要求が成功した場合，全て“ 200 OK ”を返す．このため，PUT 方式のテストアンドセット操作が利用できない．このような XDMS には，DELETE 方式を使用する必要がある．

第 5 章

実装

5.1 システム構成

実装した文書管理システムの構成図を図 5.1 に示す.

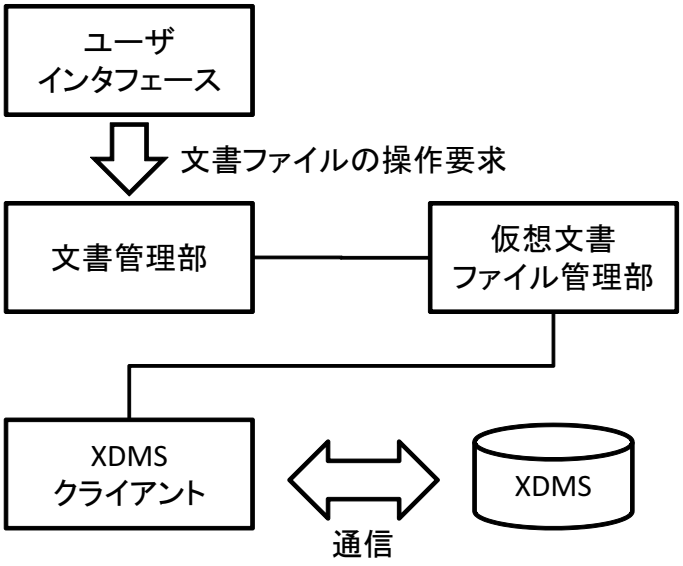

図 5.1 システム構成

以下に，システムの各構成要素について詳細を述べる.

(1) ユーザインタフェース

ユーザインタフェースは，文書ファイルおよびディレクトリの追加，削除および参照を行うためのインタフェースを提供する.

(2) 文書管理部

仮想文書ファイル管理部を通して，XDMS 上の文書ファイルを管理する．また，各文書ファイルを必要に応じて，XML 形式に変換する．

(3) 仮想文書ファイル管理部

仮想文書ファイル管理部は，文書管理部の要求に応じて，XCAP ドキュメントの参照および更新を行う．仮想文書ファイル管理部は，XDMS 上に存在する XCAP ドキュメントを一般的なファイルシステムのように扱うためのインタフェースを提供する．このインタフェースを利用することで，文書管理部は XDMS を意識することなく文書を管理できる．

(4) XDMS クライアント

XDMS クライアントは，仮想文書ファイル管理部からの要求に応じて，XDMS と通信し，XCAP ドキュメントの参照，更新および削除を行う．

(5) XDMS

インデックスおよび各文書ファイルを，XCAP ドキュメントとして保存，管理する．

5.2 実装環境

実装を行った環境を表 5.1 に示す．

表 5.1 実装環境

OS	Debian 5.0.3
使用言語	Ruby 1.8.7
Web アプリケーションフレームワーク	Ruby on Rails 2.3.4
XDMS クライアントライブラリ	xcapclient 1.2.2[11]
XDMS	Fraunhofer FOKUS XDMS

5.3 動作概要

5.3.1 ディレクトリ操作

ディレクトリを作成する際の動作概要を (A-1) から (A-2) に述べる．

(A-1) ディレクトリを作成する前に，すでに同名のディレクトリまたは文書ファイルが存在しないか確認する．存在する場合，ディレクトリの作成は失敗する．存在しない場合，新たに作成するディレクトリ名をインデックスの“ used-names ”要素に追加する．

(A-2) “ directory ”要素をインデックスに追加する．

また，ディレクトリを削除する際の動作概要を (B-1) から (B-3) に述べる

(B-1) 削除するディレクトリが空であるかどうか確認する．ディレクトリが空でない場合，ディレクトリ内に存在する全てのディレクトリおよび文書ファイルに対して，削除操作を行う．

(B-2) “ directory ”要素を削除する．

(B-3) 削除したディレクトリ名を，インデックスの“ used-names ”要素から削除する．

5.3.2 文書ファイル操作

文書ファイルを新たに追加する際の動作概要を (C-1) から (C-4) に述べる．

(C-1) 文書ファイルを追加する前に，すでに同名のディレクトリまたは文書ファイルが存在しないか確認する．存在する場合，文書ファイルの追加は失敗する．存在しない場合，新たに追加する文書ファイル名をインデックスの“ used-names ”要素に追加する．

(C-2) 作成する文書ファイルパスから ID(MD5 ハッシュ値) を算出し，この ID がすでに使用されていないか確認する．使用されている場合，文書ファイルパスの末尾に“ 0 ”を追加し，ID を再算出する．以降は，未使用の ID が得られるまで，ID の再算出を繰り返す．未使用の ID を取得後，インデックスの“ used-ids ”要素に ID を追加する．

(C-3) “ document ”要素を作成する．

(C-4) ID を名前として，XCAP ドキュメントを新規作成する．

文書ファイルを削除する際の動作概要を (D-1) から (D-5) に述べる．

(D-1) インデックスを参照し，文書ファイルパスから ID を特定する．

(D-2) ID を名前に持つ XCAP ドキュメントを削除する．

(D-3)“ document ”要素を削除する．

(D-4) 文書ファイルの ID をインデックスの“ used-ids ”要素から削除する．

(D-5) 文書ファイル名を“ used-names ”要素から削除する．

第 6 章

おわりに

本論文では，IMS における XDMS を用いた文書管理システムについて述べた．まず，XDMS と IMS の概要を述べた．また，XDMS の操作インタフェースである XCAP について述べた．次に，XDMS を DBMS として利用することについて検討し，中小規模システム内では，XDMS を DBMS として利用可能であることを示した．また，文書管理システムで XDMS を利用することの利点と問題点を明らかにした．

XDMS に文書管理用の XML 文書を導入することより，XDMS 上で仮想的なディレクトリ操作および文書ファイルのディレクトリ間移動操作が実現できることを示した．また，この XML 文書を参照することで，文書ファイルの一覧を取得できることを示した．そして，文書ファイルの排他制御機構について検討した．排他制御機構には，テストアンドセット操作が必要であることを示し，PUT 方式と DELETE 方式のテストアンドセット操作を提案した．また，この 2 つのテストアンドセット操作を比較し，DELETE 方式より PUT 方式が優れていることを示した．また，XDMS によっては，PUT 方式が使用できず，DELETE 方式を使用する必要があることを示した．実装した文書管理システムのシステム構成を示し，動作概要を述べた．

謝辞

本研究を進めるにあたり，懇切丁寧なご指導をしていただきました乃村能成准教授に心より感謝の意を表します．また，研究活動において，数々のご指導やご助言を与えていただいた谷口秀夫教授，田端利宏准教授，後藤佑介助教に心から感謝申し上げます．

また，日頃の研究活動において，お世話になりました研究室の皆様に感謝いたします．

最後に，本研究を行うにあたり，経済的，精神的な支えとなった家族に感謝いたします．

参考文献

[1] “ OMA Instant Messaging and Presence Service V1.3, ”Open Mobile Alliance, 23 January 2007

[2] “ OMA Push to talk over Cellular V1.0.4, ”Open Mobile Alliance, 3 December 2009

[3] “ XML Document Management Specification V1.1, ”Open Mobile Alliance, 27 June 2008.

[4] J. Rosenberg,“ The Extensible Markup Language (XML) Configuration Access Protocol (XCAP), ”IETF RFC4825, May 2007.

[5] J. Rosenberg, “ SIP: Session Initiation Protocol, ”IETF RFC3261, Jun 2002.

[6] “ OMA IP Multimedia Subsystem V1.0, ” Open Mobile Alliance, 19 September 2005.

[7] “ IANA, ”http://www.iana.org/

[8] J. Rosenberg, “ Presence Authorization Rules, ”IETF RFC5025, December 2007

[9] “ W3C XML Schema, ”http://www.w3.org/XML/Schema

[10] “ Fraunhofer FOKUS XDMS, ”http://www.fokus.fraunhofer.de/en/fokus_testbeds/open_ims_playground/components/xdms/index.html

[11] Iñaki Baz Castillo,“ xcapclient, ”http://xcapclient.rubyforge.org/