特 別 研 究 報 告 書

題 目

IMS/XDMSの高度利用を実現するソフトウェアライブラリに関する検討

指導教員

報 告 者

長尾 武憲

岡山大学工学部 情報工学科

平成23年2月14日 提出

要約

次世代ネットワークである Next Generation Network(NGN) において，IP Multimedia Subsystem(IMS) は，高品質で高信頼のマルチメディア通信を提供するための仕組みを規定している．IMS は，設定情報の保存に XML Document Management Server(XDMS) を提供している．XDMS が扱う設定情報として，携帯電話の通信記録や Instant Messaging のコンタクトリストがある．XDMS 内の XML 文書の追加，更新，及び削除には，XML Configuration Access Protocol(XCAP) と呼ばれる HTTP ベースのプロトコルが使用される．XCAP はリソースの指定に URI を用いる．また，操作には GET，PUT，及び DELETE の 3 つの HTTP メソッドを文書の取得，更新，及び削除と対応付けて用いる．

増大するネットワークのトラフィックに伴って，高品質，高信頼な大容量のデータ通信に対する需要は高まっている．このため，今後 NGN が普及していき，XDMS の利用も増加すると考えられる．そこで，XDMS には「XDMS へ複雑な操作を行うシステムを容易に実装したい」という要求が生まれると考えられる．この要求を満たすためには，以下の 2 つが問題となる．1 つ目は，XDMS には，自身が保持する文書の一覧を取得する機能がないという点である．このため，多数の文書を扱うようなシステムを構築しづらい．2 つ目は，XCAP は単純な問い合わせしかできない点である．このため，整列といった複雑な処理の実現には，XDMS を利用する Application Server(AS) 側での処理が不可欠である．

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．文書の一覧が取得できない問題の対処として，既に提案されているインデックスを導入し，文書の一覧を取得する方式を実装した．複雑な操作の実装には AS 側での処理が不可欠である問題の対処として，ライブラリの導入により，XDMS を利用した AS の工数を削減する方法を提案した．インデックスを AS 内に実装するのではなく，ライブラリ内に隠ぺいすることで，AS がインデックスの動作を意識せずに処理できるようにした．さらに，ライブラリを実装し，その動作概要についても述べた．

目次

1 はじめに **1**

2 XDMS と XCAP **3**

2.1 XML Document Management Server 3

2.2 XML Configuration Access Protocol 5

2.2.1 XCAP の概要 5

2.2.2 URI 構成 5

2.2.3 リソース操作 6

3 XDMS の抱える課題 **9**

3.1 文書管理における XDMS の利点と欠点 9

3.1.1 利点 9

3.1.2 欠点 9

3.2 XDMS への要求と対処 10

3.3 課題 11

4 システム設計 **12**

4.1 システム構成 12

4.1.1 全体の設計方針 12

4.1.2 全体構成 13

4.2 設計の詳細 13

4.2.1 新たな課題への対処 13

4.2.2 インデックス 14

4.2.3 ライブラリ 16

5 実装 **18**
5.1 実装環境 18
5.2 動作概要 18
5.2.1 文書の取得 18
5.2.2 文書の追加 20
5.2.3 文書の更新 20
5.2.4 文書の削除 23

6 おわりに **24**

謝辞 **25**

参考文献 **26**

発表論文 **27**

図 目 次

1.1 NGN/IMS における XDMS 2
1.2 ライブラリの導入 2

2.1 URI の例 6

3.1 ライブラリの挿入 11

4.1 従来のシステムの構成図 13
4.2 設計したシステムの構成図 13
4.3 インデックスの例 15

5.1 登録文書 19
5.2 インデックス 19
5.3 インデックスの取得 19
5.4 指定文書の取得 20
5.5 文書の追加：新規作成 20
5.6 文書の追加：更新 20
5.7 追加後のインデックス 21
5.8 文書の更新：文書の取得 21
5.9 文書の更新：編集 22
5.10 文書の更新：更新 22
5.11 更新後の文書 22
5.12 全文書の削除 23
5.13 指定文書の削除 23

表 目 次

2.1 XCAP における HTTP メソッドの役割 7

4.1 実装する動作 16

5.1 実装環境 18

第 1 章

はじめに

IP ネットワークと電話網を統合した次世代ネットワークとして，Next Generation Network [1](以下，NGN とする) が存在する．NGN における IP Multimedia Subsystem[1](以下，IMS とする) は，高品質で高信頼のマルチメディア通信を提供するための仕組みを規定している．

IMS の中には，設定情報を保管する XML Document Management Server(以下，XDMS とする) がある．NGN/IMS における XDMS の位置づけを図 1.1 に示す．ユーザは，IMS の内外から Application Server(以下，AS とする) を通じて XDMS 内の設定情報を利用する．XDMS 内の設定情報の例としては，携帯電話の通信記録や Instant Messaging(以下，IM とする) のコンタクトリスト，及びプレゼンス監視に関する情報などがある．XDMS は，XML 文書を扱う．XML 文書とは，XML によって整形された文書である．XDMS 内の XML 文書の操作は，XML Configuration Access Protocol[2](以下，XCAP とする) と呼ばれる HTTP ベースのプロトコルが使用される．XCAP はリソースの指定に URI を用いる．また，操作には GET，PUT，及び DELETE の 3 つの HTTP メソッドを文書の取得，更新，及び削除と対応付けて用いる．

近年，動画配信サイトなどの利用の増加により，ネットワークのトラフィック量は増加 [3] している．このため，高品質，高信頼な大容量のデータ通信に対する需要は高まっており，NGN への需要も高まっている．そして，XDMS の利用も増加すると考えられる．これに伴い，XDMS に対して「XDMS で設定情報だけでなく，一般の文書も扱いたい」という要求が生まれる．また,「XDMS へ複雑な操作を行うシステムを容易に実装したい」という要求も生まれると考えられる．これらの要求を満たすためには，2 つの問題がある．1 つ目は，URI を知らない XML 文書へのアクセスが不可能な点である．このため，文書の一覧を取得する際は，AS 側に一覧取得のための処理を実装する必要があり，AS の工数が増加する．2 つ目

は，複雑な操作を実装するためには，AS の処理が通常の Web アプリケーションに比べて多くなる点である．これは，XCAP で使用できるメソッドは少なく，単純な問い合わせしかできないためである．本論文では，文書の一覧が取得できない問題の対処として，XDMS の文書一覧を取得するために，インデックス [4] を実現する．また，複雑な操作の実装で AS 側における処理が通常より多くなる問題の対処として，ライブラリの導入による XDMS を利用した AS の工数を削減する方法を提案する．外部 AS へのライブラリの導入の位置付けを図 1.2 に示す．ライブラリを通して XDMS を使用することで，AS の負担を軽減する．さらに，インデックスを導入したライブラリの設計，実装，及び動作概要についても述べる．

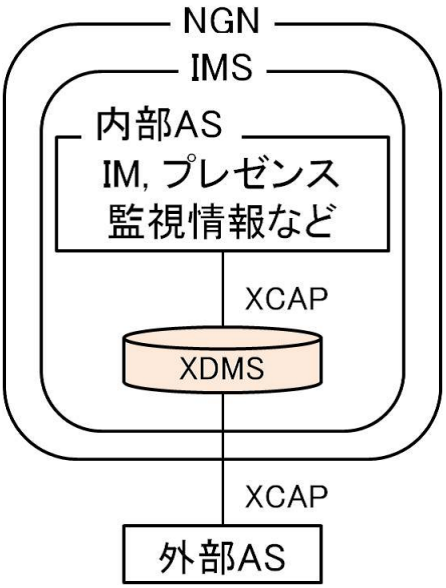

図 1.1 NGN/IMS における XDMS

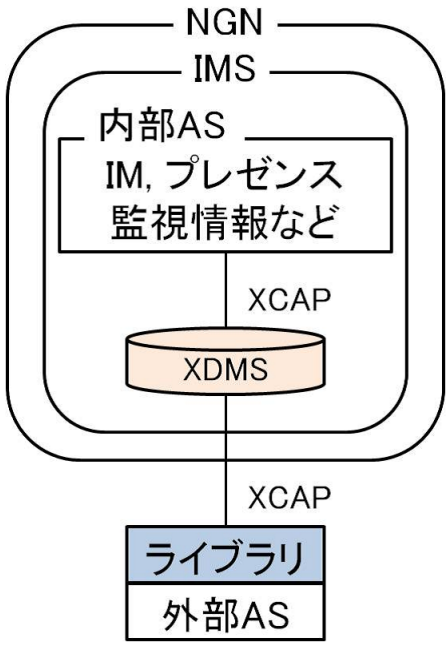

図 1.2 ライブラリの導入

第 2 章

XDMSとXCAP

2.1 XML Document Management Server

NGN とは，IP ネットワークと電話網を統合した次世代ネットワークである．NGN において，音声と動画などのマルチメディア・アプリケーションをパケット通信ネットワーク上で提供するために標準化された仕組みとして，IMS が存在する．IMS は，高品質で高信頼のマルチメディア通信を提供するための仕組みを規定しており，NGN の中核技術として世界的に導入が計画されている．IMS は以下の特徴を持つ．

(1) パケット網上でのサービス品質を保証

各サービスに適した品質や帯域となるように，セッション制御サーバがルータを制御する．これにより，アプリケーションごとに必要なサービス品質を確保している．

(2) 課金するための仕組みを提供

IMS は事業者に，課金のための必要な機能を提供している．具体的には，以下の 2 つの課金方法を提供している．

(A) オフライン課金

電話の開始や終了時刻などの詳細な記録を通じて，料金請求が行われる課金の仕組みである．料金請求後に，料金未払いにより，サービスが使えなくなるなどの影響が発生する場合もある．

(B) オンライン課金

課金の状況に応じてリアルタイムに，サービスの利用が可能かどうかを判断し，通信に影響を与えるような課金の仕組みである．例えば，限度額を超えた時点で利用ができなくなるようなプリペイドサービスが挙げられる．

(3) 高度なセキュリティ機能を提供

ユーザを一意に認証する認証機能や IP Security Protocol と呼ばれる暗号通信プロトコルを適用する．これにより，IP の偽装や反射攻撃 (パスワードや暗号鍵の盗聴によるなりすまし) を防ぎ，セキュリティを高めている．

(4) パケット網上における各種メディアへ対応

インターネット用のプロトコルを多用し，各種メディアをパケット網上で提供する．

IMS を利用するメリットとして，さらに以下のような点が期待される．

(1) 各種アプリケーションを短期間かつ低コストで開発可能

共通のプラットフォームとして IMS を利用しているためである．

(2) 異なるネットワーク間での相互接続，さらに複数ベンダからの製品を組み合わせる事が可能

国際標準として細部の規定が行われているためである．

(3) 移動や固定の各種アクセスに関するシステムを統合可能

今までネットワークごとに構築されていたシステムのうち，ネットワーク制御をサービスから分離して共通化しているためである．

IMS の中で，XDMS はユーザの設定情報を XML 文書として保管している．設定情報としては，例えば，携帯電話の通信記録，Instant Messaging のコンタクトリスト，及びプレゼンス監視に関する情報などが挙げられる．XDMS 内の XML 文書の操作には，XCAP と呼ばれる HTTP ベースのプロトコルが使用される．

2.2 XML Configuration Access Protocol

2.2.1 XCAP の概要

XCAP は，状態をサーバ上に保管するためのプロトコルである．各ユーザの設定と状態情報は XML 形式で保管される．XCAP は以下の特徴を持つ．

(1) 通信に HTTP を使用

(2) リソースの指定に URI を使用

(3) アプリケーションごとにリソースの使用方法を定義可能

ユーザがクライアントから HTTP メッセージを XDMS に送信することで，XDMS 上の XML 文書を操作する．

2.2.2 URI 構成

XCAP では，XML 文書を操作する際，以下の 4 項目を図 2.1 のように連結し，エンコードする．そしてエンコードした結果を，サーバに送る HTTP URI とする．各項目は RFC4825[2] で規定されている．

(1) XCAP ルート

XCAP ドキュメントが保存されているツリーのルートを示す．

(2) ドキュメント・セレクタ

XCAP ドキュメント（すなわち XML 文書）を指定する．

(3) パス・セパレータ

ノード・セレクタが存在する場合に，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 2 つのチルダ記号「~~」である．

(4) ノードセレクタ

XML 文書内の特定の要素や属性などを指定するクエリ式である．XPath[5] の概念に基づいている．

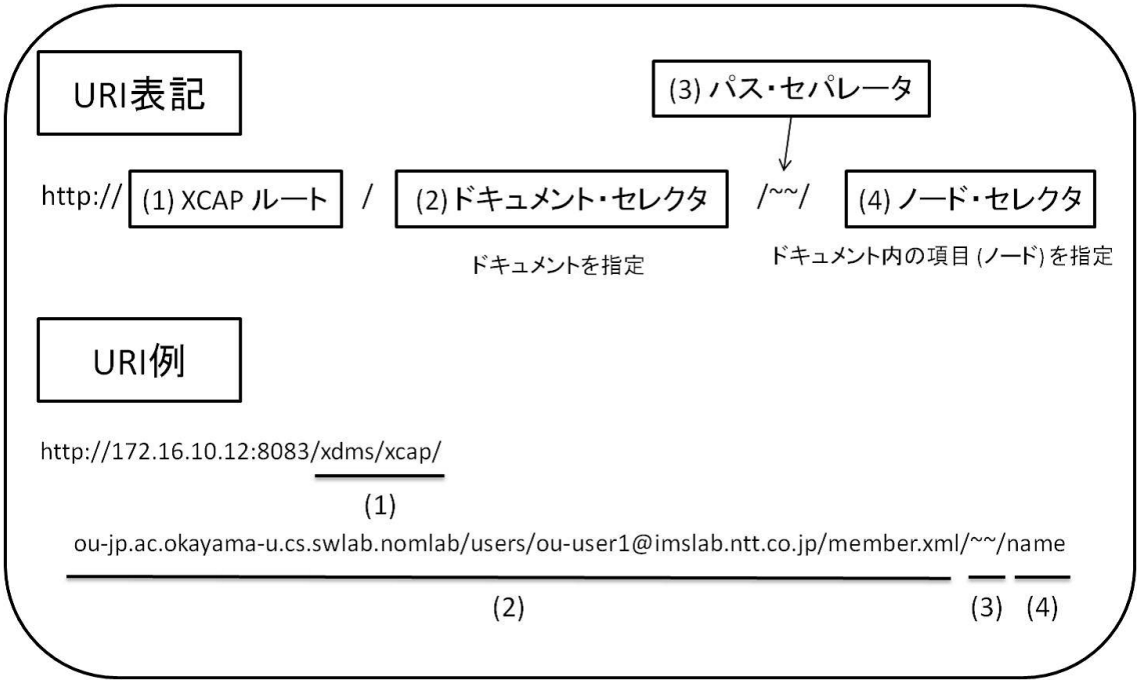

図 2.1 URI の例

図 2.1 の URI 例は，XDMS 内の member.xml 直下の name 要素を指定する URI である．(1) は XCAP ルートを表す．(2) はドキュメント・セレクタ，すなわち XDMS 内の member.xml を指定している．(3) はパス・セパレータである．(4) はノードセレクタ，すなわち member.xml 直下の name 要素を指定している．

2.2.3 リソース操作

リソースの操作内容は GET，PUT，及び DELETE といった送信する HTTP メソッドによって決定する．各 HTTP メソッドと役割を表 2.1 に示す．また，表 2.1 の降順に詳細を述べる．

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	リソースの取得
PUT	リソースの作成，置換
DELETE	リソースの削除

(1) XCAP リソースの取得

リソースの取得は下記のように，GET メソッドを用いる．

```
GET http://172.16.10.12:8083/xdms/xcap/ou-jp.ac.okayama-u.cs.
    swlab.nomlab/users/ou-user1@imslab.ntt.co.jp/member.xml
```

リソースの取得に成功すれば，XDMS はステータスコード 200 を返す．リソースが存在しなければ，ステータスコード 404 を返す．

(2) XCAP リソースの作成，置換

リソースの作成，置換は下記のように，PUT メソッドを用いる．

```
PUT http://172.16.10.12:8083/xdms/xcap/ou-jp.ac.okayama-u.cs.
    swlab.nomlab/users/ou-user1@imslab.ntt.co.jp/member.xml
```

XDMS は，PUT 要求を受け取ると以下のように処理する．

(A) 指定リソースが存在しない場合

HTTP メッセージのボディ部の内容を持つ新たなリソースを作成する．XDMS はステータスコード 201 を返す．

(B) 指定リソースがすでに存在する場合

指定リソースの内容を HTTP メッセージのボディ部の内容へ置き換える．XDMS はステータスコード 200 を返す．

(3) XCAP リソースの削除

リソースの削除は下記のように，DELETE メソッドを用いる．

```
DELETE http://172.16.10.12:8083/xdms/xcap/ou-jp.ac.okayama-u.cs.
       swlab.nomlab/users/ou-user1@imslab.ntt.co.jp/member.xml
```

リソースの削除に成功すれば，XDMS はステータスコード 200 を返す．リソースが存在しなければ，ステータスコード 404 を返す．

第 3 章

XDMS の抱える課題

3.1 文書管理における XDMS の利点と欠点

3.1.1 利点

以下に文書管理において，XDMS を使用する利点を述べる．

(1) XCAP を通信に利用

クライアントは XDMS への通信に XCAP を用いる．このため，クライアントの実装が XDMS の内部構造に依存しない．

(2) アプリケーションごとに XML 文書の利用方法を定義可能

共通の Application Usage(リソースに対して定義されたデータ構造や制約の集合) に対応していれば，アプリケーション間での情報共有が容易となる．よって，文書の共有が容易となる．

(3) 接続に特殊なプログラムが不必要

通信は，広く用いられている HTTP メソッドを利用するため，接続に特殊なプログラムを必要としない．

3.1.2 欠点

以下に文書管理において，XDMS を使用する欠点を述べる．

(1) HTTP 以外では通信不可能

XDMS とクライアントが同一計算機上にある場合でも HTTP で問い合わせる必要がある．

(2) URI を知らない XML 文書へのアクセスが不可能

全ての文書を一覧するインデックスが存在せず，URI を指定してファイルを操作するため，XML 文書の一覧を取得できない．

(3) 複雑な問い合わせが不可能

XCAP が利用できるメソッドは 3 つしかない．このため，複数文書への一括操作や整列といった複雑な処理を行う場合，AS 側で処理する必要がある．

3.2 XDMS への要求と対処

(1) XDMS への要求

XCAP は単純な問い合わせのみが可能であり，複雑な問い合わせの実現には多くのソフトウェア記述が必要である．これに対し，NGN の普及にも伴って「XDMS へ複雑な操作を行うシステムを容易に実装したい」という要求が生まれると考えられる．この要求を解決することで，少ない記述量で複雑な操作が可能となり，抽象度の高い操作で XDMS を操作できる．しかし，要求の実現のためには，3.1.2 項の (2)(3) で述べた，以下の 2 つの問題点に対処する必要がある．

(問題点 1) XML 文書の一覧が取得不可能

(問題点 2) XDMS は複雑な処理が不可能

(2) XDMS への対処

2 つの問題点の解決方法としては，以下のような方法がそれぞれ考えられる．

(対処 1) インデックスの導入

XDMS 上に存在する XML 文書の一覧を保持するドキュメントを作成する．このドキュメントをインデックスと呼ぶ．インデックスを導入する事で，XML 文書の一覧が取得できる．

(対処 2) ライブラリの挿入

ライブラリとは，ある特定の機能を持ったプログラムを，他のプログラムから利用できるように部品化し，複数のプログラム部品を一つのファイルにまとめたものである．これを図 3.1 のように，AS と XDMS との間に挿入する．AS はライブラリを通じて XDMS を操作する．これにより，AS からの XDMS の操作を簡略化する事ができる．

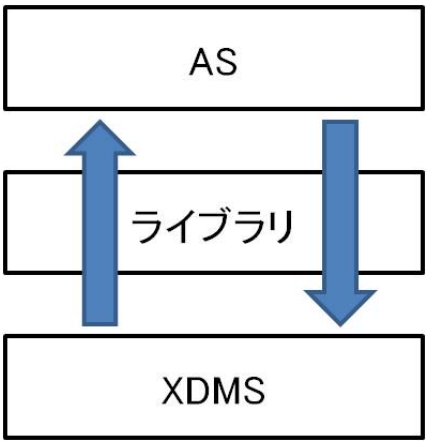

図 3.1 ライブラリの挿入

3.3 課題

3.2 項の (2) で述べたインデックスの導入とライブラリの挿入を行う場合，以下の 2 つの事項について検討する必要がある．

(1) リソースの更新方法

データの送信量と送信回数はトレードオフの関係にある．よって処理速度や実装の簡易さからリソースの更新方法について決定する必要がある．

(2) AS 側からの利用方法（インタフェースの決定）

AS 側からのライブラリの使用方法を決定する必要がある．

これらの課題への対処方法については次章で述べる．

第 4 章

システム設計

4.1 システム構成

4.1.1 全体の設計方針

本研究の実装では，既存の Web アプリケーションフレームワークである Ruby on Rails(以下，Rails とする) とその一部である ActiveResource を用いる．ActiveResource とは，データベースに対し RESTful な API を提供し，Ruby のオブジェクトにマッピングする操作を抽象化するライブラリである．この ActiveResource を用いる理由は，以下の利点が存在するためである．

(1) XML 文書をオブジェクトとして操作可能

ActiveResource は，XML 文書をオブジェクトとして扱うマッピングツールの役割を果たす．すなわち，ActiveResource の利用者が XDMS 上のデータをオブジェクトとして操作できるようになる．これにより，XDMS 上のデータを操作するための工数を大幅に削減できる．

(2) 既存の Web アプリケーションとの高い親和性

ActiveResource は，リレーショナルデータベース (以下，RDB とする) を Ruby のオブジェクトとして操作するライブラリである ActiveRecord と類似している．このため，既存の RDB+ActiveRecord に基づくアプリケーションを XDMS+ActiveResource に少ない工数で改変できる．

4.1.2 全体構成

従来のシステムの構成図を図 4.1 に示す．これに対し，設計したシステムの全体構成図を図 4.2 に示す.

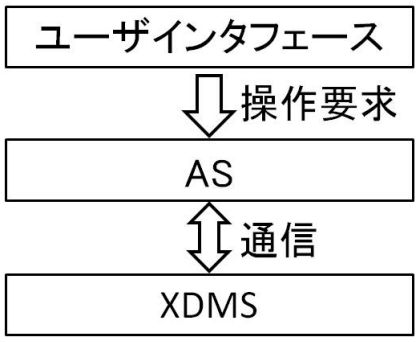

図 4.1 従来のシステムの構成図

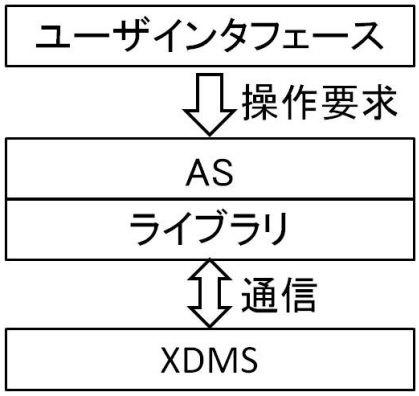

図 4.2 設計したシステムの構成図

従来のシステム構成では，AS はユーザインタフェースからの操作要求に応じた処理と，XDMS との通信処理を行う．また，文書の一覧取得も AS 側で処理する必要がある．

これに対して設計したシステムは，通信処理と一覧取得を AS ではなくライブラリで行う．ライブラリは，AS の要求に応じて，文書の参照，更新，及び削除を行う．ライブラリは XDMS 上に存在する文書をオブジェクトとして扱う．ライブラリを利用する事で，AS は XDMS 上の文書をオブジェクトとして操作できる．インデックスの詳細については，4.2.2 項で述べる．ライブラリの詳細については，4.2.3 項で述べる．

4.2 設計の詳細

4.2.1 新たな課題への対処

以下に 3.3 節で述べた課題への対処方法と設計方針について述べる．

(1) リソースの更新方法

リソースの更新には以下の 2 つの方法がある．

(A) バルクアップデート

リソース全体を送信する更新方法である．クライアントの実装が簡単になる．しかし，送受信するデータが大きくなる．

(B) パーシャルアップデート

リソースの更新したい部分だけを送信する更新方法である．送受信するデータが少なくなる．しかし，GET したリソースの一部を修正してそのまま PUT するという使い方ができなくなる．

リソースの更新は，実装の簡易さから，バルクアップデートで実装する．

(2) AS 側からの利用方法（インタフェースの決定）

方法としては，以下の 2 つが考えられる．

(A) アプリケーションに include して利用

AS 内のアプリケーションに組み込む形で利用する．アプリケーションの構造がライブラリの内部構造に依存する欠点がある．一般的なライブラリの利用方法である．

(B) 別プロセスを立ち上げて利用

WebAPI のように作成したライブラリに対し，HTTP でアクセスする．ステートフルな実装が可能となる．しかし，送信量が増えパフォーマンスの低下を招く．

処理速度から考えた場合，(A) の方法が良いと考えられる．また一般的な利用方法に近い方が，AS の作成の手間を軽減できると考えられる．よってライブラリは，AS 内のアプリケーションに include して利用する．

4.2.2 インデックス

XCAP では，XDMS 上に存在する XML 文書の一覧 (インデックス) を取得する事ができない．このため，インデックスを取得するためにはライブラリ側でインデックスを作成し，管理する必要がある．本ライブラリでは，インデックスを除く全ての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．

このインデックスは，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 4.3 に示す．また，各 XML 要素の内容を以下に述べる．

(1) root_directory 要素

仮想的なルートディレクトリを表す．

```
<?xml version="1.0" encoding="UTF-8" ?>
<head>
  <used_ids>
    <id1 type="integer">1</id1>
    <id2 type="integer">2</id2>
  </used_ids>
</head>
<root_directory>
  <document1>
    <name>member</name>
    <id type="integer">1</id>
  </document1>
  <document2>
    <name>group</name>
    <id type="integer">2</id>
  </document2>
  <used_names>
    <name1>member</name1>
    <name2>group</name2>
  </used_names>
</root_directory>
```

図 4.3 インデックスの例

(2) document 要素

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

(3) used_ids 要素

使用中の ID を子要素として保持する．

(4) used_names 要素

使用中の文書名を子要素として保持する．

4.2.3 ライブラリ

XCAP は単純な問い合わせのみが可能であり，整列や複数文書への一括操作といった複雑な処理の実現には，AS 側での多くのソフトウェア記述が必要である．このため，複雑な処理を実装する場合には，工数が多くなる．この工数を削減し，XDMS を利用した AS を容易に実装するために，ライブラリを導入する．

ライブラリで実装する動作を表 4.1 に示す．

表 4.1 実装する動作

動作	メソッド
文書の取得	find(文書の検索条件)
文書の新規作成	new(:name => 任意の文書名，文書ファイル)
文書の更新	save
文書の削除	destroy(文書の検索条件)

各メソッドについての説明を以下に示す．

(1) find(文書の検索条件)

文書を取得し，オブジェクトに変換して返す．文書の検索条件は:all か文書名をとる．:all の場合はインデックスを取得する．文書名の場合は，指定された文書名の文書を取得する．

(2) new(:name => 任意の文書名, 文書ファイル)

オブジェクトを新規作成して返す．このメソッドだけではXDMSに登録はされない．登録するには後述のsaveメソッドを使用する．name要素は文書名の決定に用いるため，必ず指定する必要がある．文書ファイルはハッシュをとる．ハッシュを使用するのは，ActiveResourceの仕様である．このハッシュは，同階層に同じ要素名がある場合，複数ではなく1つの要素として認識するため，同階層に同じ要素名を用いないようにすべきである．

(3) save

任意のオブジェクトをXMLに変換してXDMSに登録する．この際，新規に追加された文書であればインデックスを更新する．

(4) destroy(文書の検索条件)

文書を削除する．文書の検索条件は:allか文書名をとる．:allの場合は，インデックスに登録されている全ての文書とインデックスを削除する．文書名の場合は，指定された文書名の文書を削除し，インデックスを更新する．

各メソッドの動作概要については次章で述べる．

第 5 章

実装

5.1 実装環境

実装環境について表 5.1 に示す.

表 5.1 実装環境

OS	Ubuntu 10.04
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 3.0.3
ActiveResource	3.0.3
XDMS	Fraunhofer FOKUS XDMS

5.2 動作概要

5.2.1 文書の取得

図 5.1 の文書と図 5.2 のインデックスが XDMS に登録されているとする．このとき，文書を取得する際の動作概要について述べる.

(1) インデックスの参照

図 5.3 のように書くと，インデックスをオブジェクトとして取得し，参照できる．なお，#以下は，実行結果である.

```
<?xml version="1.0" encoding="UTF-8" ?>
<name>member</name>
<content>nagao</content>
<id type="integer">1</id>
```

図 5.1 登録文書

```
<?xml version="1.0" encoding="UTF-8" ?>
<head>
  <used_ids>
    <id1 type="integer">1</id1>
  </used_ids>
</head>
<root_directory>
  <document1>
    <name>member</name>
    <id type="integer">1</id>
  </document1>
  <used_names>
    <name1>member</name1>
  </used_names>
</root_directory>
```

図 5.2 インデックス

```
index = Document.find(:all)
# index => {head => {used_ids => 1}, root_directory => {document1 =>
#           {name => member, id => 1}, used_names => {name1 => member}}}
```

図 5.3 インデックスの取得

(2) 文書の取得

図 5.4 のように文書名を指定する事で，XDMS に存在する member.xml をオブジェクトとして取得し，参照できる．なお，#以下は，実行結果である．

```
document = Document.find("member")
# document => {name => "member", content => "nagao", id => 1}
```

図 5.4 指定文書の取得

5.2.2 文書の追加

図 5.1 の文書と図 5.2 のインデックスが XDMS に登録されているとする．このとき，文書を追加する際の動作概要について述べる．

(1) new メソッドで文書を作成する．要素の指定は図 5.5 のようにハッシュで行う．この際，name 要素は文書名の決定に用いるので，必ず指定する．

```
group = Document.new(:name => "group", :content => "gn")
```

図 5.5 文書の追加：新規作成

(2) save メソッドを図 5.6 のように実行する事で，文書の追加が行われる．この際，インデックスも図 5.7 のように更新される．

```
group.save
```

図 5.6 文書の追加：更新

5.2.3 文書の更新

図 5.1 の文書と図 5.2 のインデックスが XDMS に登録されているとする．このとき，文書を更新する際の動作概要について述べる．

```
<?xml version="1.0" encoding="UTF-8" ?>
<head>
  <used_ids>
    <id1 type="integer">1</id1>
    <id2 type="integer">2</id2>
  </used_ids>
</head>
<root_directory>
  <document1>
    <name>member</name>
    <id type="integer">1</id>
  </document1>
  <document2>
    <name>group</name>
    <id type="integer">2</id>
  </document2>
  <used_names>
    <name1>member</name1>
    <name2>group</name2>
  </used_names>
</root_directory>
```

図 5.7 追加後のインデックス

```
document = Document.find("member")
# document => {name => "member", content => "nagao", id => 1}
```

図 5.8 文書の更新：文書の取得

(1) 5.2.1 項と同様に，図 5.8 のように文書を取得する．なお，#以下は，実行結果である．

(2) 文書の内容を編集する．例えば，図 5.9 のように書く事で member.xml 中の content 要素を変更できる．

```
document.content = "takenori"
```

図 5.9 文書の更新：編集

(3) save メソッドを 5.2.2 項と同様に図 5.10 のように実行する事で，文書の更新が行われる．更新後の member.xml は図 5.11 である．

```
document.save
```

図 5.10 文書の更新：更新

```
<?xml version="1.0" encoding="UTF-8" ?>
<name>member</name>
<content>takenori<content>
<id type="integer">1</id>
```

図 5.11 更新後の文書

5.2.4 文書の削除

図 5.1 の文書と図 5.2 のインデックスが XDMS に登録されているとする．このとき，文書を削除する際の動作概要について述べる．

(1) 全文書の削除

図 5.12 のように書く事で，インデックスに登録されている全文書とインデックスを削除できる．

```
Document.destroy(:all)
```

図 5.12 全文書の削除

(2) 指定文書の削除

図 5.13 のように書く事で member.xml を削除できる．削除の際，インデックス内から member.xml に関する部分も削除される．

```
Document.destroy("member")
```

図 5.13 指定文書の削除

第 6 章

おわりに

本論文では，IMS/XDMS の高度利用を実現するライブラリの実現を検討した．まず，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．次に XDMS の利点と欠点を示し，「XDMS へ複雑な操作を行うシステムを容易に実装したい」という要求を示した．この要求を満たすには，XDMS の欠点である XML 文書の一覧が取得不可能であることと XDMS 自体は複雑な処理が不可能であることが課題となる．このため，対処法として，インデックスの導入によって文書の一覧を取得することとライブラリの導入によって AS の工数を削減することを述べた．

そして，インデックスとライブラリの設計について述べた．また，ライブラリの操作として，インデックスを含む文書の取得，新規作成，更新，及び削除を実装した．最後に，実装したライブラリのシステム構成を示し，動作概要について述べた．

このライブラリの実装により，AS は XDMS の文書を操作する際，通信とオブジェクトへの変換をライブラリで行うため，AS 内のアプリケーションのコード量が削減できる．また，インデックスをライブラリ内に隠ぺいすることで，AS はインデックスの動作を意識せずに処理を行える．

残された課題として，実際にどの程度の工数が削減できたのかという定量的な評価がある．さらに，排他制御とインデックス内のディレクトリ構造の実装も残されている．

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，山内利宏准教授，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] 藤岡雅宣, 小田稔周, 仲田和彦, “IMS 入門,” 株式会社インプレス R&D, July 2007.

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

[3] Sara Cicero, “Annual Cisco Visual Networking Index Forecast Projects Global IP Traffic to Increase More Than Fourfold by 2014,” Cisco Systems, June 2010, http://newsroom.cisco.com/dlls/2010/prod_060210.html.

[4] 須賀院 吉伸, 乃村 能成, 谷口 秀夫, “IMS における XDMS を利用した文書管理システムの検討,” 第 72 回情報処理学会全国大会講演論文集, DVD-ROM, March, 2010.

[5] Anders Berglund, Scott Boag, Don Chamberlin, Mary F. Fernandez, Michael Kay, Jonathan Robie, Jerome Simeon, “W3C XML Path Language (XPath) 2.0,” W3C, January 2007, http://www.w3.org/TR/2007/REC-xpath20-20070123/.

発表論文

[1] 乃村能成, 長尾武憲, 谷口秀夫, 南裕也, 並河大地, “IMS における XDMS 高度利用のためのソフトウェアフレームワーク,” 第 73 回情報処理学会全国大会講演論文集, (2011 年 3 月掲載予定).