Multimedia Content Object Model 及びその実装

鈴木 正† 池田 哲夫†† 青木 輝勝† 安田 浩†

東京大学先端科学技術研究センター†
日本電信電話株式会社 NTT サイバースペース研究所††

円滑なコンテンツ流通を阻害する要因の一つとして，ディジタルコンテンツの権利関係などを調べる手段が乏しく，既にあるディジタルコンテンツを素材として利用することが困難である, という問題が挙げられる. これを解決するために本稿では, Multimedia Content Object Model (MCOM) を提案し，その実装である Java Multimedia Content (JMC) について報告する.

The Multimedia Content Object Model and Its Implementation

Tadashi Suzuki†, Tetsuo Ikeda††, Terumasa Aoki† and Hiroshi Yasuda†
Research Center for Advanced Science and Technology, The University of Tokyo†
NTT Cyber Space Laboratories, NTT Corporation††

Content distribution is one of the most potent network applications, but currently it has some problems to solve. In this paper we focus on:

No mechanism allowing content recipients to examine the copyrights of received contents. People who rework digital contents for resale have difficulty in ensuring the legality of re-using them.

In order to solve this problem, we propose the Multimedia Content Object Model and report its implementation.

1.はじめに

インターネットの普及, 音声, 画像などのディジタル符号化技術の向上に伴い, ディジタルコンテンツを容易に利用可能な環境が整いつつある. しかし一方で, コンテンツを安心・安全に流通させるという点においては, いくつかの阻害要因が指摘されている[1].

A) 「劣化が少なく複製が容易」というディジタルの特徴により, 著作権者が安心してディジタルコンテンツをネットワークに流せない.

B) 課金・決済のしくみが乏しく, 著作権料の支払い方法が複雑.

C) コンテンツの権利関係などを調べる手段が少なく, 特に要素片の組みこみを中心に, 再利用（編集・加工を含む）が不安でできない.

D) ディジタルコンテンツを大量にデータベース化して相互利用する際, アクセス・検索する共通的なコンテンツの識別体系がない.

我々はこれらの問題を主として C)をコンテンツ自身の問題，A)，B)，D)をコンテンツ流通基盤の問題と捉え，本研究においては C)を解決するために，新たなコンテンツモデルとして Multimedia Content Object Model（以下，MCOM と略）を提案する．ただし，問題 A)，B)，D)においてもコンテンツ自身の改良により解決可能な部分問題については，MCOM の要求項目として取り入れ，課題とする．

また，本稿では MCOM を提案した後，その実装である Java Multimedia Content (JMC)と，そのメタ情報格納方式について報告する．

2. 課題と従来研究の問題

新たなコンテンツを定義するために，解決すべき問題 C)を以下の課題に分解した．

① メタ情報の参照可能化：新たに作成したコンテンツ, また素材として利用したコンテンツのメタ情報を容易に参照可能であること．

② コンテンツの再利用可能化：既存の多種多様なコンテンツファイルからコンテンツを作成できること．また, 作成したコンテンツは素材として再利用可能であること．

また，より高度な課題として，以下を設定した．

③ 編集・加工条件の記述と保証：著作権者の意図する編集・加工条件がメタ情報及びコンテンツに対して許可される操作として正確に記述され, その操作を通じてのみ編集・加工が可能であること．

④ 素材コンテンツの自動更新可能化：素材として利用されているコンテンツの最新状態を, コンテンツを作り直すことなく, 反映することが可能であること．

上記課題①，②，③，④について，具体例と従来研究の状況を示して説明する．

課題①：ディジタルコンテンツを再利用する場合, そのコンテンツの権利関係を調べる必要がある. また, それが別のコンテンツを素材として利用して作成されている場合は, その素材コンテンツの権利関係も調べる必要がある．

従来研究[1]では素材コンテンツへのリンク情報をもとに権利関係を調べることができる. 本研究で提案する新たなコンテンツモデルにおいても, 再利用するコンテンツ, また, それが含む素材コンテンツの権利関係を容易に調べることができることを要求項目とする．

課題②：新たなコンテンツを作成する際に, 既存の多種多様なファイルフォーマット, MPEG, JPEG, GIF，WAV，MP3 などのファイルが再利用されることが考えられる．

例えば，コンテンツ a を MPEG ファイルをもとに作成し，コンテンツ b を JPEG ファイルをもとに作成するといったことが考えられる．この際, 再利用を可能にするため，これらコンテンツ a と b を用いて MPEG ファイルと JPEG ファイルが混在したコンテンツ c を作成するといったことが可能でなければならない．

多くのカプセル化コンテンツの研究[3][4][5][6]は主として課題 A)，もしくは課題 B)の解決を目的しているため, コンテンツを素材として再利用することができない（素材として再利用するためにはカプセル化されていないコンテンツを必要とする）．本研究で提案する新たなコンテンツモデルは, 素材として再利用可能であることを要求項目とする．

課題③：コンテンツによっては，再利用の際，コンテンツの改変に条件を課している場合がある．

例えば, 芸術作品の画像の場合は，その作品の芸術性を損なわないように, 一定の解像度以下に落とすことを制限する，あるいは，白黒画像に変更することを制限する場合などが考えられる. 更に複雑なものとして, 通常は白黒画像に変更することを許可するが, 解像度を落とした場合には白黒画像に変更することを制限する場合などが考えられる．

従来研究[1][2]のように著作権情報を文書によって記述した場合, 変更操作の実行を利用者に任せることになり，意図的にその変更制限を無視する，また，意図的でない場合であっても，操作の内容が複雑になった場合などには, 著作権者の意図したとおりに変更の操作が行われないなどの問題が起こりうる．

課題④：素材として利用されているコンテンツの最新状態の反映の要求には, 以下のような場合が考えられる．

- 素材利用側が素材の最新状態を反映したいと考える場合．
- 素材提供側が素材の最新状態を反映したいと考える場合．

従来研究[1]の方式では，前者のような要求があった場合には, メタ情報に記述された素材へのリンク情報をもとに, 更新されているかどうかを確認し, 更新されていれば新たにコンテンツを作成しなおすなどの方法しかない. また後者の要求に対する解決策は提供されていない. 本研究で提案する新たなコンテンツモデルは，前者，後者の要求を同時に解決することを要求項目とする．

3. Multimedia Content Object Model

Multimedia Content Object Model ではオブジェクト指向に基づき, メタ情報, コンテンツ(以下, コンテンツデータ), 及び操作を備えたオブジェクトを新たなコンテンツ(以下, MCOM オブジェクト)として再定義することにより前述の課題を解決する. 3.2 の Composite パターンを用いたクラス構造により課題①を, 3.3 の実行時合成により課題②を, 3.4 の操作とシナリオにより課題③を, 3.6 の分散型コンテンツにより課題④を解決する.

3.1 カプセル化とインターフェース

MCOM オブジェクトでは, メタ情報とコンテンツデータの両方をオブジェクトの属性として取り扱う. 属性はカプセル化[7]され, MCOM オブジェクトを利用する側（以下, クライアント）が直接アクセスすることはできない. クライアントはインターフェースを通じてのみ MCOM オブジェクトの内部にアクセスできる. 図 1 にカプセル化とインターフェースを通じたアクセスの概念図を示す.

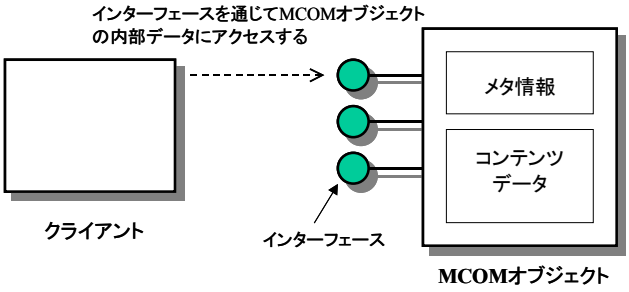

図 1：カプセル化とインターフェースの概念

3.2 Composite パターンを用いたクラス構造

MCOM の主要なクラスに以下のものがある.

・PrimitiveContent クラス

既存のコンテンツデータとメタ情報を格納し, カプセル化するための MCOM オブジェクトとして, PrimitiveContent クラスを設計した. PrimitiveContent オブジェクトに格納するコンテンツデータには MPEG, JPEG, GIF, WAV, MP3 などが考えられる.

・ContentContainer クラス

作成された PrimitiveContent オブジェクトを再利用するため, ContentContainer クラスを設計した. ContentContainer オブジェクトは PrimitiveContent オブジェクト, または ContentContainer オブジェクトを素材として利用して作成する. ContentContainer オブジェクトはコンテンツデータを含まないが, 後述する, 素材を利用するための記述を含むオブジェクトであり, MCOM では PrimitiveContent オブジェクトと同様, コンテンツとして扱う.

・Material クラス

PrimitiveContent クラス, ContentContainer クラスともに素材としての性質を持つため, 抽象クラスとして Material クラスを導入し, Composite パターン[8]を用いて, 再帰的にオブジェクトを構成することが可能であるように設計した. Composite パターンを用いることにより, MCOM オブジェクトを利用するクライアントは, 利用するオブジェクトが PrimitiveContent オブジェクトか ContentContainer オブジェクトであるかを意識する必要がない.

図 2 に MCOM の主要なクラス構造を示す.

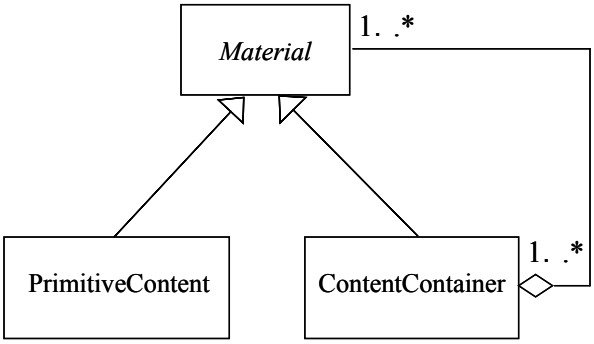

図 2：MCOM の主要なクラス構造

MCOM オブジェクトは抽象クラス Material を継承したサブクラスのインスタンスであり, PrimitiveContent クラスのインスタンスもしくは ContentContainer クラスのインスタンスである. ContentContainer オブジェクトは 1 つ以上の Material オブジェクトの参照を保持する. また, 後述する分散型コンテンツを実現した場合, Material オブジェクトは, 複数の ContentContainer オブジェクトから利用されることが想定されるため ContentContainer クラスと Material クラスを N 対 N 関連とした. 図 3 に MCOM の典型的なオブジェクト構造を示す. 図 3 に見られるような階層構造を MCOM オブジェクト自身が持つことにより, その参照先をたどることで, 新しく作成されたコンテンツのメタ情報の取得はもちろん, 素材として利用されたコンテンツから再利用に必要な権利関係などのメタ情報を取得することができる.

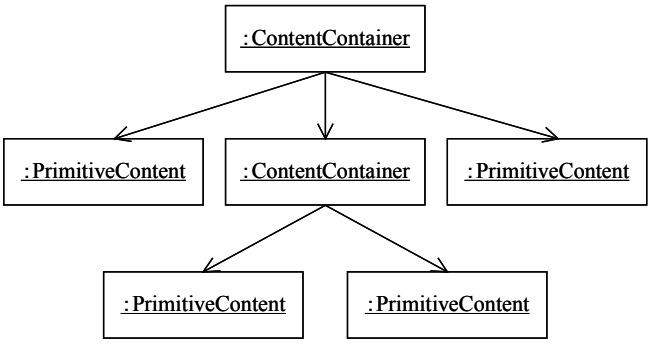

図 3：MCOM の典型的なオブジェクト構造

3.3 実行時合成（Run-Time Composition）

ContentContainer オブジェクトは，それが保持する Material オブジェクトへの参照を利用して，実行時に Material オブジェクトにコンテンツデータを要求する．コンテンツデータを要求された Material オブジェクトが PrimitiveContent オブジェクトの場合は，それが保持するコンテンツデータを返し，ContentContainer オブジェクトの場合は，更にそれが保持する Material オブジェクトにコンテンツデータを要求する．

コンテンツデータはリストの形で ContentContainer オブジェクトを利用するクライアントに返される．図 4 にコンテンツデータの要求の例を示す．

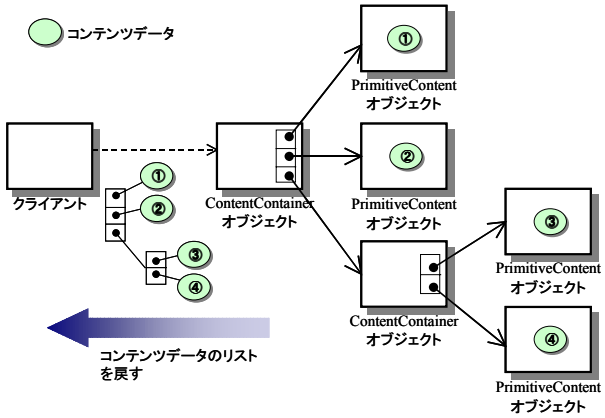

図 4：コンテンツデータの要求の例

このように，実行時にコンテンツデータを収集する仕組みを用いることによりにより，動画，静止画，音声などといった，複数種類のファイルフォーマットのコンテンツを素材として利用した場合でも 1 種類のファイルフォーマットに変換することなく再利用可能になる．

3.4 操作とシナリオ

クライアントはインターフェース中に定義され，外部に公開された操作を通じてのみ MCOM オブジェクトの内部データにアクセスすることができる．

また，ContentContainer は利用する Material オブジェクトの参照とともに，利用する際に実行する Material オブジェクトの操作とその実行順序を保存できなければならない．つまり，ContentContainer はどの素材を利用するかということだけでなく，どの素材を，どのように利用するかということを記述する．この ContentContainer に含まれる，Material オブジェクトに対して実行する操作，またその順番などの素材を利用するための記述をシナリオと呼ぶことにする．また，ContentContainer は利用するすべての Material オブジェクトのシナリオを保持する．PrimitiveContent がコンテンツデータを保持するのに対し，ContentContainer はシナリオを保持する．

例えば，ContentContainer オブジェクト a が Material オブジェクト b と Material オブジェクト c を利用する場合を考える．Material オブジェクト b が公開している操作，「白黒にする」，「解像度を 72pixels/inch に変更する」を順に実行して Material オブジェクト b を利用し，Material オブジェクト c が公開している操作，「回転する」を実行して利用する場合，「白黒にする」，「解像度を 72pixels/inch に変更する」という操作を順番を保ったまま Material オブジェクト b のためのシナリオとして保存し，「回転する」を Material オブジェクト c のためのシナリオとして保存しておかなければならない．また，解像度の変更操作に関しては「72pixels/inch」という引数情報も Material オブジェクト b のシナリオに含まれる．

その他にシナリオには，再生時に後述するプレーヤが利用するものがある．例えば，画像の描画位置，音声のボリュームなどの情報が挙げられる．これら再生時に利用するシナリオは，クライアントに渡すコンテンツデータのリストに含めることを検討している．また，各コンテンツの開始・終了時間や，各コンテンツデータの再生順序といった時間軸方向の，より高度なシナリオのモデル化も検討中である．

3.5 再生

再生は MCOM オブジェクトを再生するための専用プレーヤを用いる．これを MCOM プレーヤと呼ぶことにする．MCOM プレーヤは MCOM オブジェクトからコンテンツデータのリストを受け取りを再生する．ContentContainer から受け取る場

合は, シナリオに沿って操作を適用した後のコンテンツデータのリストを受け取る. また, 受け取ったコンテンツデータをどのように再生するかはMCOM プレーヤが決定する.

図 5 にコンテンツデータとして複数の静止画を含むリストを受け取った場合の MCOM プレーヤのコンテンツデータ再生の例を示す. この例では, MCOM プレーヤはリストに格納されている順にコンテンツデータを読み込み, 表示画面に描画している.

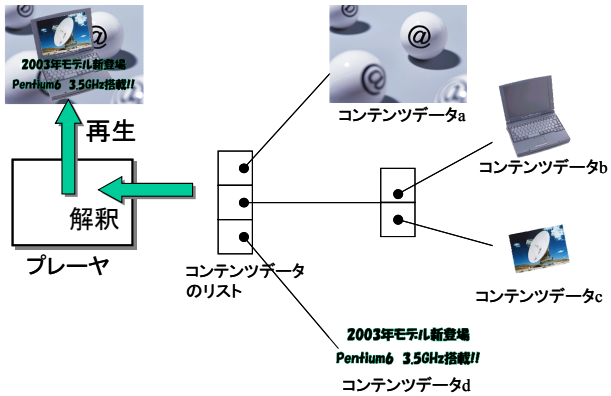

図 5：コンテンツデータ再生の例

3.6 分散型コンテンツ

MCOM は実行時合成を行うので, コンテンツをインターネット上に分散させることを可能にすることにより, 各素材コンテンツを提供者側では提供するコンテンツを一元管理することが可能になり, 古いコンテンツが散在することを防ぐことが可能になる. また, 素材利用者は常に最新の状態で素材を利用することが可能になる. 図 6 に各 MCOM オブジェクトをインターネット上に分散させた例を示す.

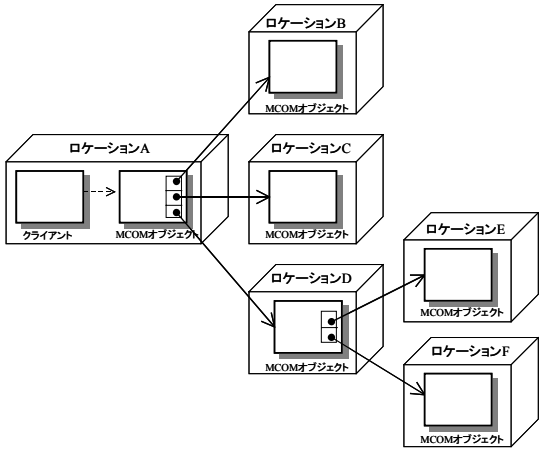

図 6：コンテンツを分散させた例

4.Java Multimedia Content

MCOM の実現可能性, 妥当性を検証するために, その実装である Java Multimedia Content(以下, JMC と略）を開発した. 現在の JMC の実装では MCOM の特徴のうち, 操作とシナリオ, 分散型コンテンツについては対応していないが, Composite パターンを用いた階層構造, 実行時合成の機能を実装し, 課題①, ②を解決している. また, 課題③, ④の実装については, 今後の課題とし, 別途報告する.

4.1 開発言語の選択

JMC では MCOM に従ったコンテンツを実現するために, 主としてプログラミング言語を利用した. また, 後述するように XML(Extensible Markup Language)[9]を一部で利用した. XML ではなく, 主としてプログラミング言語を利用したのは, 課題③を達成するために, データ記述だけではなく, 属性に対する操作を正確に記述する必要があるからである. また, プログラミング言語には Java を選択した. Java の特徴[10]には以下のようなものがある.

- 汎用のオブジェクト指向言語である.
- 一度記述したプログラムはインターネット上のどこでも実行できる.

この他, Java を選択した理由として以下がある.

- 豊富なライブラリが提供されている.
- 仮想マシンなどの実行環境が既に広く利用されている.

4.2. 基本的な仕組み

JMC では MCOM オブジェクトを通常の Java オブジェクトとして実現する. 生成した JMC における MCOM オブジェクト（以下, JMC オブジェクト）は保存される際に, Java のシリアライズ機能を用いて, 拡張子 jmc のファイルに変換される. また, この JMC オブジェクトの操作（生成・参照・保存）は後述する操作支援ツールを用いて行う.

4.3 Composite パターンを用いたクラス構造

JMC のクラス構造は MCOM の Composite パターンを用いたクラス構造（図 2)をそのまま実現している. クラス名も同じであり, 以下のように宣言

される.

```
public abstract class Material implements
Serializable{
     public abstract Vector getContent();
     public abstract int getChildCount();
     public abstract Material getChildAt();
     ・・・
}

public class PrimitiveContent extends
Material{
     //content data
     private ImageIcon picture;
     ・・・
}

public class ContentContainer extends
Material{
     //material objects
     private Vector materials;
     ・・・
}
```

ここで Material クラスには, JMC オブジェクトをシリアライズ可能にするために java.io.Serializable インターフェースを実装している．また，PrimitiveContent クラスはコンテンツデータを格納するために，フィールドとして javax.swing.ImageIcon クラスを，ContentContainer クラスは Material オブジェクトへの参照を保持するために java.util.Vector を宣言している．現在の JMC の実装では，PrimitiveContent オブジェクトに格納できるコンテンツデータのファイルとして, JPEG ファイルと GIF ファイルに対応している.

4.4. 拡張可能なメタ情報格納方式の検討と実装

コンテンツのメタ情報には，コンテンツがどの素材コンテンツから作成されたか, といった情報の他に，コンテンツのバージョン，ID，名称，作成日，更新日などの情報がある．JMC では前者の情報を主として，オブジェクトの構造で表している．例えば，ある JMC オブジェクトがどの JMC オブジェクトから生成されるかは，ContentContainer が保持する Material オブジェクトの参照をたどることで知ることができる.

また, 一般的なオブジェクト指向言語を用いた設計では，後者のタイプの，バージョン，ID，名称，作成日，更新日といった情報は以下に示すように Material オブジェクトのフィールドに格納する.

```
public abstract class Material implements
Serializable{
     private String version;
     private String id;
     private String name;
     private Date date;
     private Date update;
     ・・・
}
```

このような方式を用いると, 新たな項目を追加したい場合，Material クラスを作成し直す必要がある．後者のタイプの情報が様々な要求のもとで，新たな項目が追加されやすい性質を持つものであることを考慮すると, 一般的なオブジェクト指向言語を用いた設計では，十分に使い易いとは言えない．そこで我々は後者のタイプの情報のために XML 文書を利用するメタ情報格納方式を導入した.

4.4.1 方式の比較と検討

メタ情報を格納するにあたってコンテンツ作成者が自由にメタ情報の項目を追加可能であることを課題とした．そこで，その要件を満たす，項目を動的に追加，削除することができる方式として，Dynamic Properties[11]を用いた方式と XML 文書を用いた方式を検討した.

・Dynamic Properties を用いたメタ情報格納方式

クラスに動的に項目を追加, 削除する方式として，Dynamic Properties パターンが知られている.

この方式は，Dictionary オブジェクトを利用することによって, 動的に属性を追加することを可能にするものである．Java を用いて実現する際のプログラムは以下のようになる.

```
public abstract class Material implements
Serializable{
     private Hashtable ht;
     public String get(String key){・・・};
     public String put(String key, String
  Value){・・・};
     Enumeration elements(){・・・};
     Enumeration keys(){・・・};
     ・・・
}
```

また，ここでは Dictionary オブジェクトとして，抽象クラス java.util.Dictionary のサブクラスである java.util.Hashtable を用いている.

・XML 文書を用いたメタ情報格納方式

XML 文書は文書作成者が自由に項目を追加，削除できるという特徴を持つ.

XML 文書を用いた方式では XML 文書そのものを Material オブジェクトのフィールドに格納するため，XML 文書の特徴を利用でき，Material オブジェクトに格納されるメタ情報の項目を動的に追

加，削除することが可能になる．

4.4.2 方式の選択

我々は動的にメタ情報の項目を追加，削除可能にする方式としてこの 2 つの方式を検討し，以下の理由により XML 文書を用いた方式を利用することにした．

- 既存研究[1][2]などにおいて XML 文書によるメタ情報記述する方式を用いており，これらをそのまま利用することを可能にすることで，コンテンツ作成者がメタ情報を再入力する手間を省くことが期待できる．

- 方式を改良することなく，メタ情報を階層的に表現することが可能である．

また，JMC の現バージョンにおいては，後述する操作支援ツールが利用するいくつかの必須の項目については，通常の固定のフィールドを用いた．

4.4.3 実装

XML 文書を用いたメタ情報格納方式では，XML 文書を Material オブジェクトのフィールドに java.io.File オブジェクトとして格納する．また，格納された File オブジェクトをクライアントが取り出し可能にするためのメソッドも定義した．プログラムは以下のようになる．

```
public abstract class Material implements
Serializable{
     private File metadata;
     public File getMetadata(){…}
     …
}
```

5.操作支援ツール The JMC Content Explorer

MCOM オブジェクトとその階層構造の作成と参照を支援するための操作支援ツール The JMC Content Explorer(以下，JCE と略)を開発した．JCE は以下の機能を持つ．

1) 既存のコンテンツ(JPEG，GIF)から，MCOM オブジェクトを生成する．

2) MCOM オブジェクトの階層構造をグラフィカルに示すことにより MCOM オブジェクトが素材として利用している MCOM オブジェクト（もしくは，その MCOM オブジェクトを素材として利用している MCOM オブジェクト）との関連を容易に参照可能にする．（図 7 (a)）

3) 階層構造中において指定した MCOM オブジェクトが直接利用している MCOM オブジェクトを容易に参照可能にする．（図 7 (b)）

4) 各 MCOM オブジェクトのメタ情報を容易に参照可能にする．（図 7 (c)）

5) 各 MCOM オブジェクトの実行時合成の結果を表示する（図 7(d)）

2)，3)の機能により MCOM オブジェクトを 2 次利用する際，利用する MCOM オブジェクト及び，その MCOM オブジェクト中に素材として利用されている MCOM オブジェクトの権利関係などを容易に調べることが可能になる．

また，4)の機能により，MCOM オブジェクトのメタ情報を容易に参照することが可能になり，5)の機能により，MCOM オブジェクトの実行時合成の結果を容易に見ることが可能になる．現在，JCE に実装されている MCOM プレーヤは，再生する際に，リストとして渡された静止画データを先頭から取得し，描画領域に描画する．また，現在の JMC は，操作とシナリオを実装していないため，MCOM プレーヤは画像を，初期値である描画領域の左上隅にそろえて描画する．図 7 に操作支援ツール JCE の画面を示す．

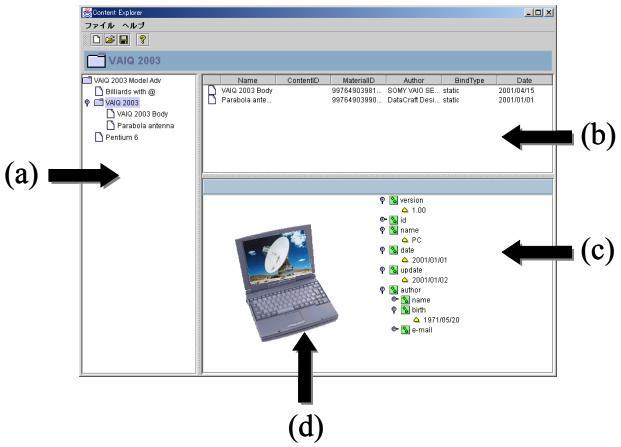

図 7：操作支援ツール JCE(コンテンツ参照)

JCE は 4)の機能を実現するために，MCOM オブジェクトに格納されている XML 文書を読み取り，表示する機能を持っている．これは DOM[12]を用いて実現した．Java で DOM を操作する API として JAXP[13]を，XML パーサには crimson[14]を用

いている. 以下のような XML 文書を読み込んだ場合, 図 8 のように JCE 画面に階層的に表示される.

```
<SampleMetadata>
    <version>1.00</version>
    <id>123456</id>
    <name>PC</name>
    <date>2001/01/01</date>
    <update>2001/01/02</update>
    <author>
        <name>Tadashi Suzuki</name>
        <birth>1971/05/20</birth>
        <e-mail>
        suzuki-t@mpeg.rcast.u-tokyo.ac.jp
        </e-mail>
    </author>
</SampleMetadata>
```

図 8：階層的に表示されたメタ情報

6.まとめと今後の課題

本稿では円滑なコンテンツ流通を実現するために, コンテンツ自身が解決すべき問題を挙げ, それらの問題を解決するための新たなコンテンツモデルを MCOM として提案した. また, MCOM の実現可能性, 妥当性を検証するための実装である JMC について報告した.

JMC により課題①, ②として挙げた, メタ情報の参照可能化とコンテンツの再利用可能化を同時に達成可能であることを示した.

また, 今後の課題としては, 課題③, ④を解決するために, MCOM の以下の機能を JMC に実装することが挙げられる.

- 操作とシナリオ
- 分散型コンテンツ

JMC における実装では前者を Java のメソッド, またそのメソッドの呼び出し情報を格納するリストとして, 後者を Java のリモート機能である RMI[15]を用いて実装することを検討している.

参考文献

[1] コンテンツ ID フォーラム
http://www.cidf.org/

[2] XrML
http://contentguard.com/index.htm

[3] 谷口,森賀,久松,櫻井："マルチメディア情報ベースとその格納単位 Matryoshka", 情処 DICOMO シンポジウム,1999

[4] 谷口,阿部,塩野入："Java を用いた動画配信用著作権保護カプセル", 情処研報 DPS 98-6, 2000

[5] 木俵,田中,上原："版権管理のための Java による画像データカプセル化", 情処研報 DBS 111-11,1997

[6] RightsShell
http://www.digigacha.com/setumei/index.html

[7] G.Booch. *Object-Oriented Design with Applications*. Redwood City:Vebhanub/ Cummings,1991

[8] E.Gamma, R.Helm, R.Johnson and J.Vlissides, オブジェクト指向における再利用のためのデザインパターン, ソフトバンク, 1995

[9] XML
http;//www.w3.org/XML/

[10] J.Gosling, B.Joy, G.Steele and G.Bracha, *The Java Language Specification.* 2nd Edition, Addison-Wesley, 2000.

[11] M.Fowler, *Dealing with Properties*, http://www.martinfowler.com/apsupp/properties.pdf

[12] DOM
http://www.w3.org/DOM/

[13] JAXP
http://java.sun.com/xml

[14] crimson
http://xml.apache.org/crimson/

[15] RMI
ftp://ftp.java.sun.com/docs/j2se1.3/rmi-spec-1.3.pdf