

著作権管理のための Java による画像データカプセル化

木俣豊† 田中克己†† 上原邦昭†††

†通信・放送機構 神戸リサーチセンター

††神戸大学大学院自然科学研究科知能科学専攻

†††神戸大学工学部情報知能工学科

本稿では、Java を用いてデジタル画像データをカプセル化し、著作権管理のためのデータアクセスをコントロールする手法について提案する。Java は、プラットフォームに独立なオブジェクト指向型言語である。Java で記述されたアプレット内にデジタル画像データをカプセル化することで、様々な環境で動作するデジタル画像オブジェクトが構築できる。このデジタル画像オブジェクトへのデータアクセスや表示は、そのメソッドファンクションを通じて行われるために、カプセル化された画像データの様々な状況に対応した表示が可能となる。その機能の実証例としてユーザ認証機能を付加したデジタル画像オブジェクトを考案し、著作権管理オブジェクトを試作する。それにより、画像データのオブジェクト化による有効性や、Java アプレットを用いたデジタル画像のカプセル化の有効性について検証を行う。

The encapsulation of image data for the copyright management by Java

Yutaka Kidawara† Katsumi Tanaka†† Kuniaki Uehara†††

†Kobe Research Center, TAO

††Division of Intelligence Science,

Graduate School of Science and Technology, Kobe University

†††Department of Computer and Systems Engineering,

Faculty of Engineering, Kobe University

In this paper, we propose an access control method of the digital image for copyright management which is encapsulated by Java. Java is a platform independent object oriented language and is able to construct the encapsulated objects for the digital image data on the various computer environment as the applets. They can control the behavior of the digital image data encapsulated in the object using method function. Therefore they can display the various method and operate data on the various environment. We study about the digital image object including user identification function and develop the copyright management image object. We will verify the usefulness of the digital image data encapsulation and Java applet encapsulation for digital image data.

1 はじめに

近年、コンピュータネットワークが急速に普及し、各コンピュータ間での情報交換が高速かつ容易に行えるようになってきた。特にインターネット上での情報の流通は目覚ましい発展を遂げ、メールやftpを用いたプログラムの流通が主であった時代から、画像や音声の情報が流通する時代となっている。現在

では、様々なマルチメディアデータがWWWを通じて大量に流れており、専門家達の情報交換の場であったインターネットが一般の人々が気軽に情報交換の場として使えるネットワークとなりつつある。また、これらのインフラストラクチャの整備に伴って、ネットワークを使った様々なマルチメディアデータを提供する新しいビジネスも立ち上がりつつある。

しかしマルチメディアデータは、コンピュータ上のデジタル情報として提供されるために品質を損なうことなくコピーすることが容易で、著作権を保護することが難しい。

ネットワーク上で著作権の存在するマルチメディアデータが広く流通するためには、そのデータが、永久的に保護される必要があり、認証されていない環境下での使用を制限することが必要不可欠である。著作権を保護するためには、不正なコピーを制限する必要があるが、ネットワーク上のすべてのコンピュータにコピー制御機能を付加することは、困難である。特に、ネットワーク上からローカルのコンピュータ上にダウンロードされたデータに対してその使用を制限することは、事実上不可能である。著作権保護の技術としては、画像情報の中に、人間には識別できないように著作権情報を入力する電子透かしの技術 [1][2] がある。この手法は、著作権情報を画像データに埋め込むことによって、制作者の権利を保護するものである。その一方で制作者が使用を許諾した者以外には、そのデータを見せないといったより強い管理手法が求められている。我々は、版權管理機能を実現するためには、データをカプセル化するオブジェクト指向技術が有効な技術であると考えている。本稿では、以下の点について報告する。

- マルチメディアデータカプセル化の基本概念と有効性
- 版權管理のためのマルチメディアデータオブジェクトの概念
- Java を用いたカプセル化の有効性と問題点

2 マルチメディアデータカプセル化の基本概念と有効性

マルチメディアデータをオブジェクトにカプセル化した場合には、以下の機能が得られる。

1. マルチメディアデータの保護
2. 複数のメソッドによる表現の多様化

このマルチメディアデータオブジェクトの基本概念を図 1 に示し、詳細を以下に述べる。

2.1 マルチメディアデータの保護

ユーザによる二次的な編集を禁止したマルチメディアデータの場合には、そのデータの編集を制限する必要がある。GIF や MPEG などの一般的なマルチメディアデータは、データ側からアプリケーション

の制御が不可能である。そのためアプリケーションで二次的な加工が行われ、制作者の意図が損なわれたり、データの不正使用や不正加工が行われる問題があった。このような問題に対してオブジェクト指向技術の特徴の一つであるデータの隠蔽化は、有効な解決手段となる。オブジェクトはカプセル化されたデータへのアクセスを特定のメソッドだけに制限する事が可能である。従って、内包されたマルチメディアデータは不正なアクセスから保護され、不正な編集等を防止する事ができる。

2.2 複数のメソッドによる表現の多様化

もう一つのオブジェクト指向技術による利点は、多様な処理機能をメソッドとして付加できることである。

たとえば、現在のシステムで作成者がユーザに応じた異なる表現でマルチメディア情報を表示する場合には、それぞれのユーザ用のデータを作成し、アプリケーションでそれを選択させる必要があった。しかし、表示メソッドにユーザを特定する機能を付加したオブジェクトを用いれば、画像の処理方法を変化させることが可能となる。

このように、従来のシステムでは、アプリケーションが処理をしていた機能をオブジェクトに組み込むことができるので作成者が意図する多様な表現を組み込むことが可能となる。

3 版權管理のためのマルチメディアデータオブジェクト

メソッドに版權管理のためのユーザ認証機能を付加したマルチメディアデータオブジェクトの基本概念を図 2 に示し、以下にその詳細を述べる。

3.1 版權管理のために必要な機能

ネットワーク上でマルチメディアデータを配布する場合、以下の手順で手続きが行われる。

1. 希望データの選択
2. ユーザ認証、決裁
3. ローカルコンピュータにダウンロード
4. ローカルコンピュータで使用

しかし、従来のこのような方法では、決裁されたデータが完全な形でローカルコンピュータ上にダウンロードされるために、データの不正な変更やコピーが可能であった。我々が提案するマルチメディアデータ

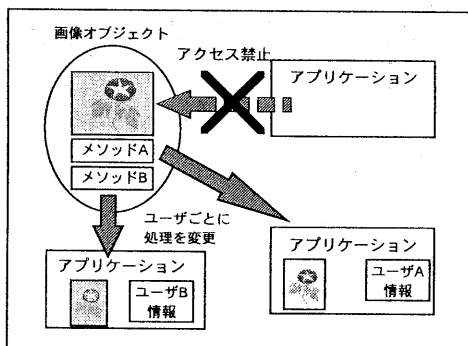


図 1: マルチメディアデータオブジェクトの基本概念

オブジェクトでは、以下の機能を付加する事によって、マルチメディアデータの版權を管理する。

- ユーザ認証機能
- マルチメディアデータ処理機能
- ユーザ非認証時のデータ処理機能

3.2 Java による実装

先に述べた基本概念に基づいた版權管理機能を持つマルチメディアデータオブジェクトを作成するには、データをカプセル化したオブジェクトとして表現できるプログラミング言語が必要となる。但し、ネットワーク上には、様々なコンピュータが接続されているのでマルチメディアデータオブジェクトのメソッドは、機種に依存せずに動作することが必要となる。

Java はネットワーク上に接続された様々なコンピュータ上で動作するプログラムの作成に適した言語である。また、Java はオブジェクト指向言語であり、マルチメディアデータをカプセル化する言語としては最適であると考えられる。

今回、我々は、Java を用いて GIF や JPEG の画像データをカプセル化し、版權管理のための機能を付

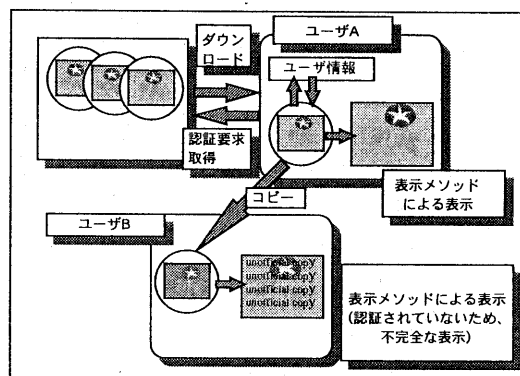


図 2: 版權管理を実現するマルチメディアデータオブジェクトの概念

加した画像オブジェクトを開発した。この開発を通して判明した Java によるカプセル化の有効性と問題点について以下に述べる。

3.2.1 画像オブジェクトとしてのアプレット

Java アプレットをネットワーク上で流通するオブジェクトとして考えるとその中に画像データをカプセル化する事で画像オブジェクトとして用いることができる。

但し、Java を用いたカプセル化において問題となるのが Java アプレットの永続化である。アプレットは、ローカルディスクへのアクセスが禁止されているので、メモリ上に存在しているインスタンスの情報をインスタンス自身がローカルディスクに記録して永続化することが不可能である。また通常の画像を表示する Java アプレットでは、メモリ上にロードされた初期化時に API を用いて別の画像ファイルを読み込む。しかし、この場合には、メモリ上では、画像オブジェクトとして存在するが、記憶装置上では、プログラムと画像データが分離した状態で存在しており、データの隠蔽化に問題がある。

我々は、画像データとメソッドを含んだクラスを分離させることなく画像オブジェクトを一つのファイ

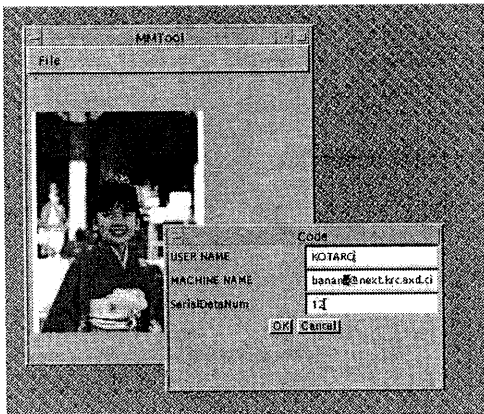


図 3: 画像オブジェクト生成ツール

ルとして画像オブジェクトを永続化するために画像データをアプレットのクラス変数として組み込んだ。これにより、画像オブジェクトのメソッド部分と画像データ部分は記憶装置上でもクラスファイルとして一体化され、画像データの隠蔽化と永続化が実現することができた。

3.2.2 画像オブジェクト生成ツール

通常の画像データを Java でカプセル化するためには、画像データを解析し、その値を組み込んだ Java のソースコードを作成する必要がある。しかし、画像データは多くのピクセル情報から構成されており、手作業でコードをプログラミングすることは、困難である。そこで、我々は、Java で任意の画像データをカプセル化するための画像オブジェクト生成ツールを開発した。このツールは、JPEG や GIF 等の画像データを読み込み、画面上のピクセル情報を記録する。その後、任意の作成者データ等と共に Java のクラス変数として定義した Java のソースコードを自動生成する。作成されたコードはコンパイルを行った後に画像データがカプセル化されたアプレット、つまり画像オブジェクトとなる。この画像オブジェクトは、表示機能、ユーザ認証機能を組み込んだ親クラスから継承されたクラスとして作成されている。これによって、Java の知識を持たないユーザでも、必要な画像データを自由にカプセル化する事ができる。このツールによって作成されたソースコード例を図 4 に示す。

```
class DataNode0 extends PixelNode{
    static int iTmpPixelImage[]={
        // カプセル化された画像データ
        -8158081,-7829116,-7500151,-7565944,-7894909,
        -7960702,-7697530,-7368565,-7565944,-7565944,
        -7565944,-7565944,-7565944,-7565944,-7565944,
        (省略)
    };
    DataNode0(){
        // コンストラクタ
        imageWidth =200;
        imageHeight=31;
        OrgXValue= 0;
        OrgYValue= 0;
    }
}
```

図 4: 画像オブジェクトソースコード例

3.2.3 ユーザ認証メソッドを付加した表示機能の検証

カプセル化ツールを用いて作成した画像オブジェクトの使用例を図 5 に示す。この例では、ユーザがワードプロセッサを用いて、認証された 3 つの画像オブジェクトと 1 つの認証されていない画像オブジェクトを使用したドキュメントを製作している場面を想定している。認証されていない画像オブジェクトでは、unofficial copy というメッセージが原画像上に表示されて使いものにならないが、認証された画像オブジェクトでは、それらのメッセージは表示されず、完全な画像として表示されている。

このドキュメントを他の認証を受けていないユーザが使用した場合には、すべて unofficial copy メッセージが出力されるために、不正なコピーの使用は、不可能となる。

これらの処理は、画像オブジェクト内の表示メソッドが行っており、ワードプロセッサ側では、画像オブジェクトに対して表示メッセージを送信しているだけである。また、この機能は、ワードプロセッサ、WWW ブラウザ等の様々な Java が動作するソフトウェア上で有効である。更に、ハードウェア等にも依存せず、ワークステーションや、PC 等で動作するため、従来の画像データと同様の感覚で使用することができた。

これによって、Java を用いることで有効な画像オブジェクトが作成できる事が検証できたと共に、ユーザ認証機能を組み込んだ画像オブジェクトが版權管理に有効である事が検証できた。



図 5: 使用例

3.3 現在の問題点

今回、画像データを Java でカプセル化してアプレットを画像オブジェクトとして使用したが、以下の問題点が明らかになった。

1. ネットワークが必要な認証機能
2. クラスファイルの複数化
3. ファイルサイズの増大

これらについて、以下に詳細を述べる。

3.3.1 ネットワークが必要な認証機能

Java のアプレットは、ネットワーク上で流通され、様々なコンピュータ上で動作する。そのため、セキュリティを脅かすプログラムの記述を困難にするために様々な制約を課している。それらの制約は場合によっては、システムを作成するに当たって機能を制限することとなる。ユーザ認証を行う場合には、以下の点が問題となる。

- ローカルコンピュータのリソース情報取得制限
- Java がロードされたサーバ以外へのネットワーク接続の禁止
- ローカルディスクへのアクセスの禁止

我々が開発した現在のシステムでは、画像オブジェクトが格納されているサーバ上で認証情報の管理を行っている。また、ユーザ情報や、ローカルコンピュータの情報は、それらの情報を格納するオブジェクトで管理を行っているが、ネットワークが切断された環境では、認証情報の記録や管理ができず、使用が困難である。

3.3.2 クラスファイルの複数化

今回行ったカプセル化は、Java のクラス変数に画像データを初期値として定義することで実現している。画像データは、Java アプレットの定数を格納する定数プールと呼ばれる領域に格納されるが、この領域は 32K 個までしか管理することができない。そのため、それらの数を超えるピクセル数を持つ情報では、複数のクラスに分割して格納する必要がある。これは結果として、複数のクラスファイルに分割されることとなる。つまり、一つの画像データを表現する画像オブジェクトは、複数のクラスファイルから構成されることとなり、オブジェクトの管理効率が悪くなる。

3.3.3 ファイルサイズの増大

カプセル化された画像オブジェクトのデータサイズを調べてみると、元情報と比較して約 5 倍弱のサイズとなっていることがわかった。例えば、200X260 ピクセルのデータ (約 208KB) を格納した結果は、約 990KB 程度の大きさになっている。

当初からメソッドを付加することによるオーバーヘッドを予想していたが、これは、我々が予想していた以上の増加量であった。メソッドのオーバーヘッドは、約 5 KB 程度でありデータが増加している原因は、データの初期化プロセスに起因していると考えられる。この原因を解析するために画像データを格納したクラスファイルを逆コンパイルして、その内容を調査した。その逆コンパイル結果を図 6 に示す。

```

class DataNode0 extends PixelNode {
    static int iTmpPixelFormat[];
    DataNode0();
    static void <clinit>();
Method void <clinit>()
0 sipush 6200
  2 バイトの符号付き整数をスタックに積む
3 newarray int
  スタックに積まれている値 (6200) の個数を
  int 型で配列を割り当てる
5 dup
  スタックの先頭要素のコピーをスタックに積む
6 iconst_0
  スタックに int 型の値 0 を積む
(iconst_0 から iconst_5 まである)
7 ldc #32 <Integer -8158081>
9 iastore
  3 で格納された配列をコピーした (4) に対して、
  要素番号 0 のところに値 (-8158081) をセット。
10 dup
  配列の参照値をコピー
11 iconst_1
  スタックに 1 を積む
12 ldc_w #292 <Integer -7829116>
  スタックに定数を積む
15 iastore
  配列 [1] に -7829116 をセット
(省略)
47744 sipush 6198
47747 ldc_w #586 <Integer -15461104>
47750 iastore
47751 dup
47752 sipush 6199
  スタックに 2 バイト符号付きの値を積む
47755 ldc_w #586 <Integer -15461104>
47758 iastore
47759 putstatic #907
<Field DataNode0.iTmpPixelFormat [I>
  スタティック領域に配列をセットする。
47762 return
}

```

図 6: 逆コンパイルリスト

この結果を見ると Java ではクラス変数の初期化を行うための初期化関数 `clinit()` をコンパイル時に付加している。この `clinit()` 関数は、画像データの格納領域を確保した後に定数プールにある定数をスタックを用いて転送している。このルーチンでは、一つのピクセル情報を配列に格納するために 4 ステップ使用していることがわかった。また、これらの処理は、配列の場合でもループではなく、すべて一つずつ処理を行っている。

このことから、データサイズの増大は、Java のクラス変数を初期化するルーチンが非常に大きくなることが原因であると判明した。

4 おわりに

本研究では、データをカプセル化する事によって得られる機能について検証を行った。そして、Java を用いてカプセル化する事によって、従来の画像データと同様に様々な環境下で使うことが可能となることが実証できた。また、メソッドにユーザ認証機能を付加することによって、従来の手法では困難

であった二次的なコピーによる不正な使用を制限する機能を持った著作権管理オブジェクトが構築できることが実証できた。

しかし、その一方で Java によるカプセル化の問題点も明らかになった。これらの問題点は、現状の Java の仕様によるところが大きくその解決には、様々な機能を改良する必要がある。Java の時期バージョンである JDK1.1 の仕様を調べてみると今回判明した認証やクラスファイルの複数化等の問題点の解決に有効な機能が追加されており、正式リリースが期待される。今回提案した画像オブジェクトとその著作権管理の手法をより実用的なものとするために以下の機能の研究、開発が必要であると考えている。

1. 分散オブジェクト CORBA/Java RMI を用いた認証
2. データ圧縮によるデータサイズ減少
3. 画像オブジェクトデータベース
4. 画像オブジェクト配布システム
5. 著作権を考慮した画像オブジェクト編集システム

今後は、これらの項目に対して、研究を行っていく予定である。

謝辞

本研究は、一部、文部省科学研究費重点領域研究(課題番号 08244103) による。

参考文献

- [1] Watermarking Technology:
<http://www.digimarc.com>
- [2] Data Hiding:
<http://www.trl.ibm.com/projects/s7730/Hiding/index.htm>
- [3] QUE: Special Edition Using JAVA
- [4] Ken Arnold,James Gosling:The Java Programming Language,JAVASOFT
- [5] Ken Arnold,James Gosling:The Java Application Programming Interface,Volume 1 Core Packages ,JAVASOFT
- [6] Ken Arnold,James Gosling:The Java Application Programming Interface,Volume 2 Window Toolkit and Applets ,JAVASOFT