アーティクル直交化による次世代情報管理システムの提案

清水亮 †
Ryo Shimizu

†Ubiquitous Entertainment Inc.
shimizu@uei.co.jp

複数の情報をアーティクル単位で管理し、複数アーティクルの相互関係性によってより大きな情報を多面的に管理するシステムを提案する。提案するシステムは、従来ファイルとディレクトリというオントロジーで扱われていたエンドユーザの扱う情報をアーティクルというより細かな単位に分解し、メタデータを付与することで利用目的に応じた序列と集合の即時取得を実現する。このシステムでは個人および組織における情報を即座に複数の用途に供することができるため、従来考えられなかった細かな情報の再利用を可能にする。

1 はじめに

現在、エンドユーザが一般的に計算機上で扱う情報の種類は非常に多岐に渡る。エンドユーザが扱う情報はより細分化され、電子メールを初めとする極めて短い文章データや、アイデアの断片、スケジュールのひとつひとつの項目、プロジェクト管理のひとつひとつの作業項目などである。これらの情報の最小構成単位を本論文では「アーティクル」と呼ぶことにする。本研究では従来、情報の本質部分に置いては再利用への要求が強いにも関わらず、技術的には再利用性が低かったアーティクルをファイルやディレクトリといった管理単位から遊離させ、複数のアプリケーション間で相互運用するための手法を開発した。

2 関連研究

2.1 ファイルとディレクトリー

従来、エンドユーザが情報を管理する最小単位は主にファイルというまとまりであった。ファイルには文章やデータが収められ、目的に応じたアプリケーションから読み込んで使用する。複数のファイルを束ねるのがディレクトリという概念であり、ディレクトリは入れ子構造を作ってツリー状に複数のファイルを保存することができる。従来より、コンピュータによって制作されるデータが単純な文章や画像、数値計算に利用されるデータであれば、この管理手法で十分と考えられていた。しかし、近年では、インターネットの発達により、従来とは扱う情報の性質と量が大きくことなるデータを扱う必要が発生し、旧来のディレクトリとファイルという管理手法ではしだいに非効率的なやりとりが発生してくることとなった。

2.2 ファイル直交化の必要性とエイリアス，スマートフォルダによる実装例

また、ファイルとディレクトリというオントロジーには他の問題もある。ファイルというかなり具体的な単位であっても、それがどのような階層構造の中に属するか一意的には決めかねる状況が生まれてきている。たとえば、あるプロジェクト A の見積もり結果を格納した表計算ファイル「見積もり.xls」が存在するとする。このとき、この「見積もり.xls」はプロジェクト A のディレクトリに格納するのが最適なようにも思える。しかし会計担当者が見積もりを全体的に把握したい場合、これは「見積もり総合」ディレクトリを作ってそこで管理できたほうが好都合である。これを従来のファイルシステムで実現するには以下の二種類の方法がある。

1. エイリアスによる解決
2. スマートフォルダによる解決

1. エイリアスによる解決を行う場合、ユーザはなにか見積もりに関するファイルを作成するごとに「プロジェクト A」ディレクトリか、「見積もり総合」ディレクトリの一方に実体を置き、他方にエイリアスを明示的に作成しなければならない。この作業をなにかファイルを作るごとに繰り返すのは大変非効率的であるし、柔軟性もない。2. スマートフォルダは Mac OS X 10.4(Tiger) から搭載された機能で、検索エンジンである Spotlight を利用して、キーワードで検索され、まとまったディレクトリを提供するものである。スマートフォルダを利用する場合、ユーザは単

に「プロジェクト A」ディレクトリで「見積もり.xls」を作成し、スマートフォルダでは「見積もり」をキーワードにファイルを検索させれば良い。スマートフォルダの利用例を図 1 に示す。しかし、いずれの方法も従来のファイルとディレクトリというオントロジーを補完する存在として位置づけられているに過ぎず、情報の管理単位そのものを変えようと言う試みではない。エイリアスを用いる場合、エンドユーザは常に情報 (ファイル) の「真の所在地」を意識する必要がある。エイリアスはあくまでエイリアスに過ぎず、エイリアスのコピーは実体のコピーとはならないからである。スマートフォルダを用いる場合は、エイリアスよりもやや意識は異なる。スマートフォルダは常に最新の「検索結果」をフォルダに見立てるものであり、スマートフォルダ内のファイルにも実体はあるからである。しかし、スマートフォルダはあくまでも検索結果の呼称であり、通常のディレクトリ (フォルダ) とは異なる扱いになる。また、スマートフォルダではファイルにメタデータを付与することで検索効率を上げることができるが、そもそもメタデータを個別のファイルに付加する作業自体は自動化されておらず、非常に煩雑な作業を要求されることになる。

2.3 既存手法の限界

エイリアスやスマートフォルダが解決しようとした問題は、つまるところファイルの集合をディレクトリという一次的な管理単位で管理するだけでなく、ツリー構造を超越した第二の次元でファイルを管理しようとするものである。このパラダイムが意味するのは、階層的な分類法の限界である。階層的な分類は人間の直感に添っているように思えるが、人間の記憶というのは複数の階層的な分類を有しているはずである。たとえば，図 2 の (a) に示すような知識の階層構造があるとする．食べ物として分類すれば,「りんご」は「食べ物/果物/りんご」というパス (経路) になるが，色として分類すると「りんご」は「色/赤/りんご」というパスになる．

人間の記憶というのは，連想的にできており，(a) と (b) のそれぞれの分類について一意的に優劣を付けることはできない．つまり，同じ情報に対して複数の階層構造で整理分類して記憶が成立しているのである．これは従来の情報管理手法ではあまり考慮されていなかった点である．

3 情報の直交化

3.1 記憶手法の直感性と直交化

コンピュータをエンドユーザが利用しようとする場合、その情報の管理手法はできるだけ直感的になることが理想である。階層構造は人間の記憶方法の一部分を模倣しているため直感的ではあるが、人間の記憶のもう一つの側面、すなわち異なる階層構造でひとつのものを意味するという部分についての再現はできない。これをエイリアスやスマートフォルダで対処療法的に解決することはできるが、本質的な問題は、そもそも階層構造的な情報の整理手法が人間の記憶機能の一側面の再現でしかないということである。そこで発想を転換して、人間の記憶により近い手法で情報を管理する必要がある。良く知られているように、人間の記憶は、概念に複数の属性を持たせ、複数の概念を相互に連鎖的に関係させることによって行われている。例として図 3 で「りんご」という概念が、複数の属性によって成り立っていることを示す。ここで属性として提示している概念は「りんご」のメタデータであり、「りんご」という概

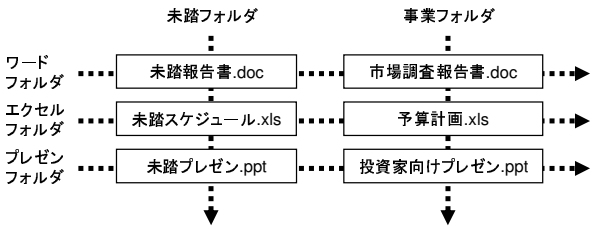

図 1: スマートフォルダの例

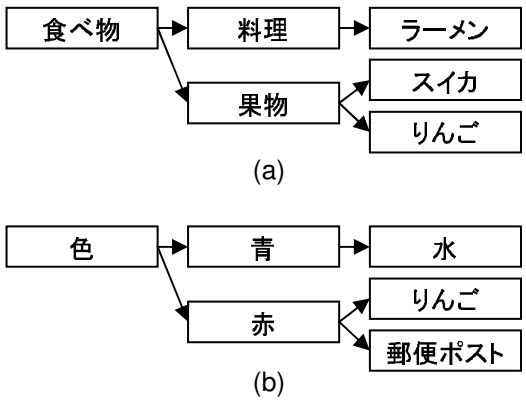

図 2: 知識階層構造

念は全てメタデータの集合として表現できる。こうすると、「りんご」を分類する時には、たとえば「食べれる/赤い」と言う分類もできるし、「腐る/甘い」という階層的な分類もできる (図 3)。このような分類をすると一見これはディレクトリによる階層構造があるかのように見える。しかし重要なのは人間の記憶において、階層関係というのはあくまでも絞り込みのための関係性であって、そこに明確な「主/従」の関係は存在しないということである。つまり、ツリー構造では、「腐る」ディレクトリの下に「甘い」ディレクトリがある場合、「甘い/腐る」では同じ情報を得ることは難しいが、人間の記憶方法ではそもそも階層に主従関係が必要なことは殆どない。

ツリー構造というのは記憶の分類法の一表現形態に過ぎず、実際には無数の分類が可能なのである。しかも、その分類はできるだけ利用目的に応じて動的に変化し、必要に応じてすぐに取り出せるものでなければならない。

情報を扱うときも同様であり、あらゆる情報は、特定の概念に基づいて集められたメタデータの集合として定義できる。情報における属性は、その数ぶんだけの次元を情報が持つことになる。その分類をなるべく属性の次元が直交するよう整理すれば、たとえある情報を織りなす属性が無数の次元に渡ったとしても、直交している故に取り出すことは容易にできる。これを実現する方法を、本論文では「情報の直交化」と呼ぶ。

3.2 情報の再利用を意識した最小管理単位

情報の分類を動的に行う主な目的は、情報の再利用にある。今日、日夜膨大な分量の情報がコンピュータに入力され、使用されているが、一度入力された情報を再利用する方法はそれほど多くない。メールでやりとりする本文、添付ファイル、Blog のエントリー、スケジューラで入力したスケジュール項目、メッセンジャーでやりとりする短いメッセージ文など、日常業務でやりとりする情報は非常に細分化され、総量としては個人の扱うべき情報は飛躍的に増大してきている。こうした情報の中でも、殆どの情報がコピーアンドペーストによって再利用される。場合によってはインターネット上の URL など、ポインタのみを示す場合もある。情報のやりとりの本質は、受信、加工、送信の三つであり、たいていの情報はどこからか受信したものを加工し、再び送信するという方法によって行われている。たとえば、メールを受信し、その内容を読んでスケジューラを起動し、メールの要件の部分をコピーして該当する日付の部分に貼付けたり、メールの内容の一部をコピーアンドペーストしてメッセンジャーで誰かに送信するなど、情報の再利用はごく無意識に行われている。この再利用性を高めることで、従来は考えられなかったコンピュータの応用法を産み出す可能性がある。情報の管理単位として、ファイルという概念が従来よく用いられてきた。ファイルとはある目的のために作られたデータの集合である。しかし、情報の再利用性の観点から考えると、ファイルは必ずしも理想的な管理単位とは言えない。情報を再利用することを想定した場合、あらゆる情報を再利用可能なまで十分小さな単位で管理するのが理想的である。

3.3 アーティクルとメタデータ

そこで本論文で扱う情報の最小単位を「アーティクル」と呼ぶことにし．次のように定義する．

1. アーティクルは単体で完結した意味を持つ情報

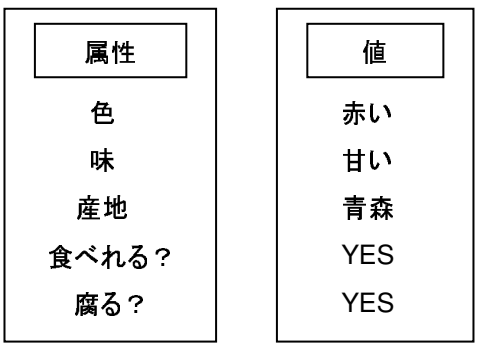

図 3: りんごが持つ属性とその値

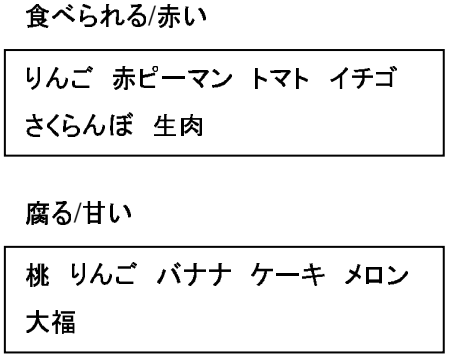

図 4: 分類の多様性

である

2. アーティクルは複数のメタデータの集合である
3. アーティクルは他のアーティクルとの依存関係を持つことができる

また．アーティクルを構成するメタデータを次のように定義する．

1. メタデータは単一のデータ型を持つ
2. メタデータはデータ型とデータの他に．その利用目的を一意的に意味するセマンティックを持つ

この構造は前述の人間の記憶方法をそのまま数理モデル化したものである．

4 システム概要

我々は既述の前提を踏まえ，これをもとにしたアーティクルの相互運用を可能とするシステムを開発した．実際に開発したシステムの要件を表 1 に示す．アーティクルとメタデータを MySQL 上にマッピン

表 1: システム要件

サーバ用 PC	
OS	Mac OS X (10.4.2)
CPU	PowerPC G5
Memory	2 GB SDRAM
HDD	100GB
Software	MySQL4.0.2
	Jakarta Tomcat 5.0
	Java 2 Standerd Edition 1.5.0
クライアント用 PC	
OS	Mac OS X (10.4.2)
CPU	PowerPC G4
Memory	2 GB SDRAM
HDD	100GB
Software	Safari
	Internet Explorer

グし，Java 用のライブラリとしてこれらの直交化したアーティクルを扱う手段を開発した．アーティクル直交化を行うために，表 2 のように Java の基底クラスを定義した．CDocument クラスは，従来でいえばファイルに当たる概念で，アーティクルの集合を意

表 2: API overview

Basic classes	
CDocument	collection of articles
CArticle	collection of attributes
CAttributte	meta date

味する．CArticle クラスは文字通りアーティクルを意味し，CAttribute クラスは属性を意味する．得に重要な CArticle クラスおよび CAttribute クラスのそれぞれが持つメソッドを，表 3, 表 4 に示す．それ

表 3: CAtricle methods

protected void addAttribute(CAttribute)
register attribute to internal array
public void read(int ID)
public void write()
public void newAtricle()
public Vector find(CAttribute attr,String query)

表 4: CAttribute methods

protected void addAttribute(CAttribute)
register attribute to internal array
public void read(int ID)
public void write()
public void newAtricle()
public Vector find(CAttribute attr,String query)

ぞれの基底クラスを継承して，実際の属性やアーティクルを作り出す．良く用いられる属性クラスとして，表 5 のようなものがある．CAttribute クラスとその継承クラスでは，コンストラクタに必ずセマンティックを渡す．セマンティックは，属性そのものの分類を決定する重要なキーワードとなる．実際にこれらの CAttribute クラスを組み合わせてスケジュール項目のアーティクルを定義したのがリスト 1 である．

リスト 1

```
public class CScheculeArticle
                extends CArticle {
  CTextAttribute subject;
```

表 5: Standard Attributes

CTextAttribute
CIntegerAttribute
CIntegerArrayAttribute
CTimeAttribute
CLinkAttribute
CLargetextAttribute
CRealnumberAttribute

```
  CTextAttribute body;
  CTextAttribute location;
  CTimeAttribute from;
  CTimeAttribute to;

  CScheduleArticle() {
    subject =
        new CTextAttribute("subject");
    body = new CTextAttribute("body");
    location =
        new CTextAttribute("location");
    from = new CTimeAttribute("from");
    to = new CTimeAttribute("to");

    super.addAttribute(subject);
    super.addAttribute(body);
    super.addAttribute(location);
    super.addAttribute(from);
    super.addAttribute(to);
  }
}
```

コンストラクタ内で CAttribute クラスの各アトリビュートにセマンティックを渡している他，基底クラスである CArticle の CArticle::addAttribute でアトリビュートを登録し，どんなアトリビュートがあるのかフレームワークに認知させている．実際にこのクラスを使用する方法をリスト 2 に示す．

リスト 2

```
CSchedule sch = new CSchedule();
sch.name.setBody(“IPA”);
sch.write();
```

5 将来の展望

本論文では，従来のファイルとディレクトリという概念に代わる情報管理の単位としてアーティクルを提唱し，その管理手法として直交的な手法を用いる実装方法を示した．将来的には未踏ソフトウエア創造事業の一環として，アーティクル直交化を用いた情報管理システムを基盤として全く新しいアプリケーションの組み合わせを実現し，ファイル指向ではない，アーティクル指向のアプリケーションの可能性を模索していきたい．