DEWS2004 I-5-03

インターネット上の異種メッセージの統合

荒木　学† 江本　守†† 国島　丈生† 横田　一正†

† 岡山県立大学 情報工学部 情報通信工学科　〒 719–1197 総社市窪木 111
†† 岡山県立大学 情報系工学研究科 電子情報通信工学専攻　〒 719–1197 総社市窪木 111
E-mail: †{manabu,emoto,kunishi,yokota}@c.oka-pu.ac.jp

あらまし　近年メーリングリストや電子掲示版のように, メッセージのやりとりの手段として E メールや Web サイトなどが多く利用されている. これらのメッセージはほぼ同じ形式を持つ半構造データであり, メーリングリストのアーカイブのように同一ソースから異なる形式に変換されたものも少なくない. 本研究では, インターネット上にある意味的に同じ形式の情報をすべて同様の形式のメッセージとして統合し, 管理する手法を提案する. 統合したメッセージを格納する形式には, 汎用性を重視するために XML を利用する. さらに統合したメッセージの応用として, Web サービスと連携した情報検索や, RSS(RDF Site Summary) を利用したメッセージの共有について提案する.

キーワード　XML, Web とインターネット, 情報統合, 半構造データ, 情報検索

Integrated Management of Messages in Different Format on the Internet

Manabu ARAKI†, Mamoru EMOTO††, Takeo KUNISHIMA†, and Kazumasa YOKOTA†

† Okayama Prefectural University, Faculty Office of Computer Science and System Engineering　111 Kuboki, Soja, Okayama 719-1197 Japan
†† Okayama Prefectural University, Graduate School of Systems Engineering　111 Kuboki, Soja, Okayama 719-1197 Japan
E-mail: †{manabu,emoto,kunishi,yokota}@c.oka-pu.ac.jp

Abstract　Recently, many messages are exchanged through the Internet such as e-mails, BBSs (bulletin board system), and Web sites. As these messages are semistructured data and in the almost same form, messages in the same source are frequently converted into another form as in mailing-list archives. However there is no standard format. In this paper, we propose a unified format and message box of various messages, which are semantically same but syntactically different, on the Internet. The format is based on XML from a wide-use point of view. Furthermore, we propose some applications such as cooperation with Web services and message sharing based on RSS (RDF site summary).

Key words　XML, Web and Internet, infomation integration, semistructured data, infomation retrieval

1. 序　　論

1.1 研究の背景

近年のインターネットの普及には目をみはるものがある. もはや E メールや Web サイトを利用することは特殊なことではなく, メッセージのやりとりの手段としても活用されている. インターネット上では電子掲示版やメーリングリスト, Weblog など様々な形態のメッセージの集合が存在する.

ここで, メーリングリストについて考えてみたい. メーリングリストとは, 特定のグループに属する人々に対し E メールを同時に送信する仕組みである. このアーカイブを HTML 文書に変換した状態で Web 上に公開してあることは少なくない. ユーザが途中から参加しているメーリングリストにおいて, メッセージを検索したいと考えたとする. 参加後の E メールはローカルマシンの MUA(Mail User Agent の略で一般的にメールクライアントと呼ばれるソフトウェアのこと) に蓄積されているが, 参加以前のものはアーカイブにのみ存在する. このアーカイブが HTML 文書で提供されていた場合は MUA で扱うことはできず, Web ブラウザを使わざるを得ない. 元々同じであったデータを扱うのに別のソフトウェアを使用しなければならなくなると, 検索や管理を行うにも一貫性が取れない上に手間が増えて不便である (図 1). この不便さを解決するためには元々同じであったデータを扱う際, 統合して扱えたらよいのではないかと考えた. さらにインターネット上の様々な形態のメッセージも

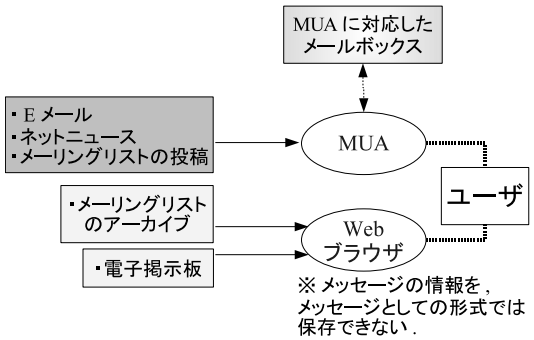

図 1　現状の概略図

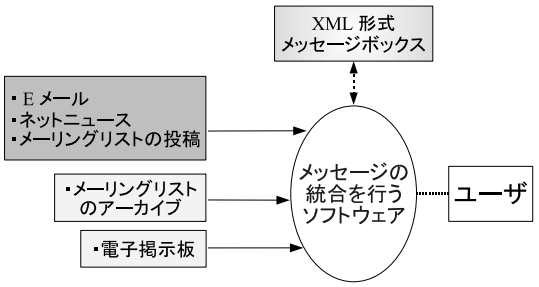

図 2　実現したい仕組み

統合して扱えるような方法を模索しようと考えたのが, 本研究の主な背景である.

もう一つの研究背景に, 既存の MUA の E メール蓄積方法に関する問題がある. E メールのシステム (POP3) における E メール受信は, 端末が MUA を使って受信メールサーバから E メールを取得し, MUA が扱えるメールボックスに蓄積するというプロセスを踏む. E メールはメールボックスに蓄積され MUA からデータとして扱えるが, 問題点もある. その一つに, 既存のメールボックスの種類が多く, それぞれの互換性があまりないことが挙げられる. メールボックスには規格化された形式はなく, あえて汎用的な形式を挙げるならば UNIX mbox 形式という慣習的に広く使われてきたものがある程度である. メールボックスに互換性が無ければ, ある MUA で蓄積した E メール資産を他の MUA に移行することができない. 現状では移行の度にメールボックスの変換を行って取り込むという方法が取られており, 不幸にも使用していたメールボックスの変換が行えない場合には取り込めない. また一般的にメールボックスを他のソフトウェアから扱うことは考慮されておらず, 仕様が非公開であるものも多い. オープンソースという流れもある今日, 非互換や非公開といった既存のメールボックスの状況は改善していくべきだと考えた.

1.2　研究の目的

本研究は, インターネットで様々な形式で提供されているメッセージをほぼ同じ形式である半構造データとして認識し, すべて同様の形式のメッセージとして統合すること, さらにはそれを管理することを目的としている. メッセージとして認識し取得する手法も, より汎用的に利用できる手法を模索する. また既存のメールボックスの問題点を解決できるように, 汎用性の高いデータ形式を使ってメッセージの格納することも, 本研究では重点を置く. 最終的に, 図 2 のようなシステムを実現することが目標である.

2.　XML 形式メッセージボックス

ここではメッセージの定義を述べた上で, 統合したメッセージの格納 · 保存に利用するメッセージの格納形式について述べる.

2.1　メッセージの定義

まず, 本論文で扱うメッセージというものを明確にする必要がある. 英語の “message” には主に「伝言」「通信文」といった意味がある. それを踏まえ, 「メッセージとは, インターネットに存在するキャラクタベースの “message” であり, 本文に加え少なくとも件名 · 発信元 · 発信日時に準じるものが明記されているもの」と定義する. 本論文においてメッセージと書いてあるものは, すべてこの定義を満たすものとする. 表 1 に, メッセージに該当するものの例を挙げる.

表 1　メッセージに該当するものの例

メッセージに該当する	メッセージに該当しない
E メール	チャットによるやりとり
メーリングリストの投稿	動画や音声による情報
電子掲示版の書き込み	画像による情報
ネットニュースの記事	
RSS	

件名 · 発信元 · 発信日時の 3 項目を必須フィールドとし, これを含むヘッダー情報は, 単にフィールドと言う. 必須フィールドに宛先を含めなかったのは, 不特定多数に向けたメッセージが存在するからである. チャットによるやりとりがメッセージに該当しないのは, 件名が無いのが最大の理由である. またチャットの性質上やりとりは一連のログとして取っておかなければ, 文脈が汲み取れないことが多く, やりとりの一言一句をそれぞれメッセージとして扱うことが最善の策とは思えない. そのため, 今回はメッセージとして扱わないものとする. E メールの添付ファイルは独立したメッセージとは言えないがメッセージの一部として重要な位置付けの場合も多い. そこで E メールの添付ファイルは本文情報の一部として扱うものとする.

2.2　メッセージの格納における要件

メッセージを統合する前に, 統合したメッセージをどのように格納し, 管理するのか考える. 今回の提案では格納したメッセージに対して検索を行うということを重視し, ローカルに保存するような形を想定する. 格納するための要件は, 次のようになる.

要件 A　**メッセージの持つ情報を欠落することなく格納できること**

メッセージは, 必須フィールド以外のフィールドを持つ場合がほとんどである. さらにメッセージの本文も必ず含まれる. これらを柔軟な形で, なおかつ欠落しないように情報を格納できることが求められる.

要件 B　**格納されたメッセージを，汎用性のある厳格な形式で保存できること**

1.2 でも述べたが, 既存のメールボックスには汎用性を軽視したものが多い. 一番汎用性がある UNIX mbox 形式に関しても仕様は統一されておらず(注1), 厳格な形式とは言えない. 異なるオペレーティングシステムや MUA で格納されたメッセージを扱えるようにすることを考えると, これも要件と含むべきであると考えた.

上記の要件を満たす仕様が標準化団体によって策定されている場合はそれを使うべきであるが, 残念ながらそのような仕様は見あたらなかった. そこで本要件を満たすようなメッセージ格納形式を, IETF の RFC2822 [4] を参考に策定した.

2.3 XML 形式メッセージボックスの仕様

メッセージを格納するフォーマットとして, メタ言語である XML を用いることにした. 理由は XML 自体が汎用性と文法の正確さを持っており, 自然に要件 B を満たすことができるからである. 本節で述べる仕様を満たす形式を XML 形式メッセージボックスと呼ぶこととする. メッセージボックスには複数のメッセージを格納でき, 必須フィールド以外の情報も格納できる. メッセージボックスのタグ構造を RELAX NG Compact Syntax [6] を用いて仕様 1 に示す.

仕様 1 XML 形式メッセージボックスのタグ構造

```
start =
  element messagebox {
    attribute version { text },
    element message {
      element header {
        element from { addressObject }
        & element subject { text }
        & element orig-date { xsd:dateTime }
        & element to { addressObject+ }?
        & element references { addressObject+ }?
        & element cc { addressObject+ }?
        & element message-id { text }?
        & element full-header { text }?
      },
      element body {
        attribute uri { text }?,
        text
      }
    }*
  }
addressObject =
  element alias { text }
  & element address { text }?
```

メッセージ 1 通に message が対応する. message には, header にある発信元の from, 件名の subject, 発信日時の orig-date の 3 つと, 本文の body が必須である. to や message-id などは, 格納可能であるが必須ではない. from や to のようにメールアドレスを扱う可能性のあるものは, メールエイリアスの alias とメールアドレスの address を子要素に持つことができ, alias を必須とする. メールアドレスを必須としないのは, 発信元情報にメールアドレスを持たないメッセージを格納可能にするためである. orig-date は, タイムゾーンを考慮した発信日時情報を格納する. full-header は, メールヘッダがある場合にメールヘッダ全体を CDATA セクションとして格納するための要素である.

次に E メールとメーリングリストアーカイブから取得したメッセージの格納例を例 1 に示す. 日本語エンコーディングは euc-jp としたが, XML で扱えるエンコーディングなら何を用いても構わない.

例 1 XML 形式メッセージボックスへの格納例

```
<?xml version="1.0" encoding="euc-jp"?>
<messagebox version="draft">
  <!-- E メール -->
  <message>
    <header>
      <from>
        <alias>Manabu ARAKI</alias>
        <address>manabu@c.oka-pu.ac.jp</address>
      </from>
      <to>
        <alias>Mamoru EMOTO</alias>
        <address>emoto@c.oka-pu.ac.jp</address>
      </to>
      <orig-date>2003-12-24T22:00:00</orig-date>
      <subject>Test</subject>
      <message-id>
        m3brqxoou9.wl@opal.c.oka-pu.ac.jp
      </message-id>
      <full-header><!-- 省略 --></full-header>
    </header>
    <body><![CDATA[This Mail is Test.
By Manabu ARAKI]]></body>
  </message>

  <!-- メーリングリストアーカイブからのメッセージ -->
  <message>
    <header>
      <from>
        <alias>hoge</alias>
        <address>hoge@c.oka-pu.ac.jp</address>
      </from>
      <orig-date>2003-12-25T22:00:00</orig-date>
      <subject>Hello</subject>
    </header>
    <body
      uri="http://hoge/mailinglist/msg0001.html"/>
  </message>
</messagebox>
```

本文はユーザに参照されるか全文検索を行われるという用途がほとんどであり, 情報の欠落というリスクを負うよりはメッセージ全体を本文と扱う方が妥当である. E メールのように本文が明確に判断できる場合は, 本文のみ取得すればよい. そこで本文にはメッセージファイルの URI を格納するか, ファイルの一部または全部を CDATA セクションとして格納することを提案する. それを実現するため, body タグの要素に CDATA セクションのテキストまたは属性に URI を格納することを可能とし, 両方を格納することも可能とする.

2.4 関連研究との比較

蓄積された E メールを XML 化する研究が過去に坪井祐太氏

(注1): mboxo 形式 ·mboxrd 形式 ·mboxcl 形式 ·mboxcl2 形式など.

によって行なわれている [1]. 研究では E メール専用の保存データ形式として XML を用いることを提案し, その有用性を示している. しかし提案している形式例では, E メールの各ヘッダ情報をそのまま要素として持たせているだけ (例えば日付情報は “Sun, 12 Dec 1999 05:14:21 -0900” のまま格納) で, ソフトウェアからは非常に扱いにくい. またスキーマ言語による構造の定義も行われておらず, 細かい仕様がやや不明瞭である.

本研究では E メールを含むインターネット上のメッセージを複数格納できるデータ形式の基盤として XML を用いる. XML 形式メッセージボックスは必須フィールドと参照頻度の高い情報 (宛先, MessageID 等) のみを各要素として持つことができ, 参照頻度の低い情報 (Recieved 等) はまとめて 1 つの要素として持つ. 少なくとも必須フィールドがあれば, 情報の多少に関わらず 1 つのメッセージとして一律の形で格納できる. 本文情報にメッセージの URI を格納できたり, メッセージの日付を XML Schema の dateTime 型 (W3CDTF [5] に準拠した日時フォーマット) で管理できたりするなど, 統合を意識した形式である.

3. Web 上のメッセージの統合

3.1 処理の流れ

メッセージが掲載されている Web サイトに対してメッセージの統合を行う大まかな流れは次のようになる.

手順 1　ユーザが指定した Web サイトにあるメッセージの URI を取得する.

手順 2　取得した URI を対象にメッセージを構成するフィールドの情報を取得し, メモリに一時的に保持する.

手順 3　保持した情報をもとにメッセージの統合を行う. 必要に応じて, XML 形式メッセージボックスに格納する.

手順 1 については 3.2 で, 手順 2 については 3.3 で, 手順 3 については 3.4 で詳細を述べる.

3.2 メッセージの URI の取得

Web でメッセージであることを認識するためには, まず各メッセージの URI(メッセージ URI) を取得する必要がある. 1 つのメッセージの内容を取り込む場合 (図 3) は URI は明らかであるが, 複数のメッセージ URI のインデックスの場合 (図 4) はそれぞれのメッセージ URI を取得した上で取り込まなければならない.

メッセージ URI を取得する時, 簡単に行なえる場合と行えない場合がある. 簡単に行なえる場合というのは, 公開されている段階で各メッセージに対して URI が割り当てられている場合である. 新規投稿が行なわれた場合でも, 新しいメッセージには新しい URI が割り当てられるので既存のメッセージ URI に変更はない.

複数のメッセージ URI のインデックスの場合 (図 4) であれば, HTML 文書内の件名に張ってあるリンク先の URI を取得する. `a` 要素の `href` 属性を全て取得して URI を提示し, 掲示された中から URI を取捨選択する方法で実現できる. 各メッセージの URI には, 通し番号を利用した命名法などの規則性がある場合が多い.

図 3　1 つのメッセージの内容 [2]

図 4　複数のメッセージ URI のインデックス [2]

URI を得ることが簡単に行なえないのは, 電子掲示版のように HTML 文書内に複数のメッセージが含まれており, 各メッセージに対応する URI が存在しない場合である. しかも新規投稿が発生すると, HTML 文書内でのメッセージの掲載位置は変わる. そこで, このような場合はまずメッセージの区切りを判断し, メッセージ毎の文書 (HTML タグも含んだ文書) に分割する. 区切りは, ユーザがメッセージの開始と終了の判断基準となる HTML タグを入力してメッセージの位置を指示する方法が妥当だと考える. 例えば水平線が区切りである場合, その表示に `hr` 要素を使っているか `img` 要素を使っているかはソースを見て判断する. こうして分割された文書をそれぞれファイルとしてローカルに保存し (図 5), 各ファイルに対して 3.3 の手法を行う.

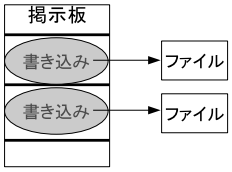

図 5　ファイルとして保存

3.3 メッセージの取得

メッセージには, 必須フィールドを含むフィールドの情報が複数存在する. 1 つのフィールドの情報を取得する方法を 3.3.1 で, それを踏まえメッセージの統合に必要となる複数のフィールドを取得する方法を 3.3.2 で述べる.

3.3.1 フィールドの情報の取得

メッセージ毎の URI が分かれば, その URI に存在するフィー

ルドの情報の取得を行う.

まず, Web で公開されているメッセージについて調査した. すると多くの場合, 発信元情報の前には “From”, “Name”, “投稿者”, “名前” 等のキーワードになり得る言葉が見つかった. 発信日時情報の前にも同様に “Date”, “投稿日”, “投稿日時” 等が見つかった. そのキーワードと記述された情報との間には, 空白やデリミタとなる記号 “:” 等があり, さらに発信日時情報の記述の仕方も “2004/1/1 12:00:00” や “1 Jan 2004 12:00:00” 等, ある程度限られていた. このような状況の場合に, 各フィールドの記述された情報を取得したい. 1 つのフィールドに対してキーワードとその情報記述パターンをいくつか想定しておけば, 高い確率で情報を取得できると考えた (図 6).

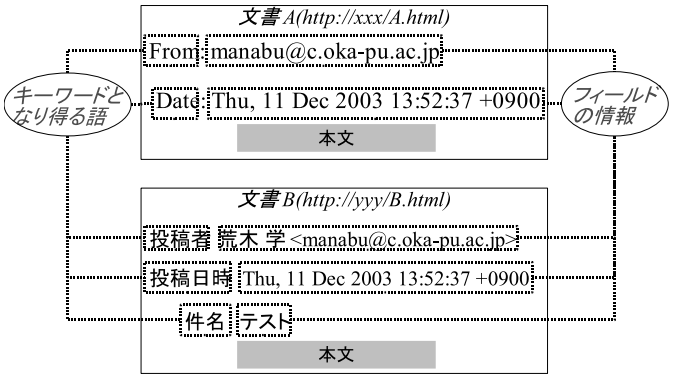

図 6　Web で公開されているメッセージの記述例

パターンの記述には正規表現を用い, 各フィールドに対してキーワードを Key と置き換えると, 例えば表 2 のように記述できる. フィールドの情報は.*の部分に存在すると考える. 不要な情報も含まれる場合もあるが, 取得後に情報の解析を行い必要な情報のみを取り出す. キーワードとパターンはそれぞれ外部情報として持ち, ユーザからの追加も可能とする.

表 2　図 6 の文書 A,B の各フィールドをパターン記述した例

Web での記述	キーワード	パターン記述
“From: manabu@c.oka-pu.ac.jp”	From	“Key:\s.*”
“Date: Thu, 11 Dec 2003 13:52:37 +0900”	Date	“Key:\s.*”
“投稿者 荒木 学<manabu@c.oka-pu.ac.jp>”	投稿者	“Key\s.*”
“投稿日時 Thu, 11 Dec 2003 13:52:37 +0900”	投稿日時	“Key\s.*”
“件名 テスト”	件名	“Key\s.*”

次にキーワードとパターンを用いてフィールドの情報を取得する手法を示す. Web 上のメッセージにおいてフィールドの記述されている箇所は特定されているものとする. その方法は 3.3.2 で述べる.

手順 1　外部情報から得たキーワードとパターンの候補を複数試し, フィールドの情報を取得する. ただし, 取得したフィールドの情報が求める情報とは限らない.

手順 2　各パターンの取得結果をユーザに提示する. ただし, 有効なフィールドの情報が取得できなかった場合 (空白のみの場合等) のパターンは提示しない.

手順 3　ユーザが, 提示された中から正しくフィールドの情報を取得できているパターンを選択する.

この手法を用いてフィールドの情報を取得した後, 使用したキーワードとパターンを蓄積する. 蓄積したものを優先利用すれば, 少なくとも同じサイトに対しては素早く情報の取得ができる.

3.3.2　隠れマルコフモデルによる複数のフィールドの取得

3.3.1 で, フィールドの情報の取得を行う方法を述べた. しかし実際にメッセージとして扱うためには, 複数のフィールドを取得する必要がある. 複数のフィールドを取得する方法として, 外部情報から得たキーワードの候補を用いて検索を行い, フィールドの記述箇所を特定する方法 (キーワード検索方法) を考えた. この方法は実現が容易だが, キーワードの候補の数の回数だけ Web 上のメッセージに対して全文検索を行うことになり, 効率が悪い. そこで隠れマルコフモデル (hidden Markov model; HMM) [3] を用いて出現順序の予測を行い, 効率良く複数のフィールドを取得する方法を考えた. HMM は事象の発生過程を状態遷移ネットワークを利用して統計的に予測するモデルの名称であり, 音声認識などのテクニックに利用され機械の音声認識率が飛躍的に増大されたことで有名である.

必須フィールドを状態, キーワードを出力記号とする HMM を図 7 に示す. なお必須フィールドは複数出現しないため, 同じ状態への遷移は考えない. 出力確率に関しては, 既存の Web サイトを対象にモデル · マージング法を使って学習してモデルを構成し, 値を求める. モデル · マージング法とは学習データのみを受理するような特殊なモデルから始め, モデル中の状態を次々と統合化していくことにより一般的なモデルを構成する方法である.

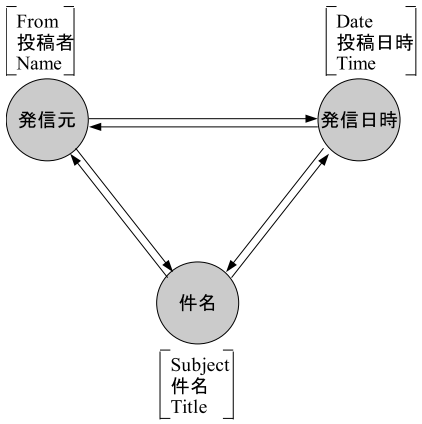

図 7　必須フィールドを状態, キーワードを出力記号とする HMM

実現する際は, あらかじめいくつもの Web サイトの記述の特徴を学習して HMM を構成しておく. それをベースに, ユーザは指定した Web サイトの複数のフィールドを取得を試み, 成功した場合のモデルをモデル · マージング法で学習することで, よりユーザに適した HMM が構成できる. 失敗した場合, まずキーワード検索方法で情報取得を行う. それと同時にモデルの学習して HMM を構成する. こうすることで次回から, そのサイトに対して HMM を使った方法を適用可能となる.

3.4　取得したメッセージの統合

Web サイトにあるメッセージを取得したら, 統合を行う. 統合とは, XML 形式メッセージボックスに格納可能な状態にする

ことを意味する．統合は，3.2 で利用したキーワードと要素を対応させることで実現する．表 3 の例で説明すれば，キーワード “From”，“投稿者”，“Name” を使って取得した情報は `from` と統一して扱うということである．

表 3　必須フィールドとキーワードの対応例

必須フィールド	`subject`	`from`	`orig-date`
キーワードの例	“Subject” “件名” “Title”	“From” “投稿者” “Name”	“Date” “投稿日時” “Time”

また，`from` のようなメールアドレスを扱うものは，フィールドの情報を解析して `alias` と `address` に分けて統合する．分けられない場合はすべて `alias` とする．`orig-date` には，日付を dateTime 型に変換して統合する．図 6 の文書 B を統合した例を，図 8 に示す．統合を行うと，ユーザは統合したメッセージの内容すべてを閲覧可能となる．必要ならば XML 形式メッセージボックスに格納することが可能となる．

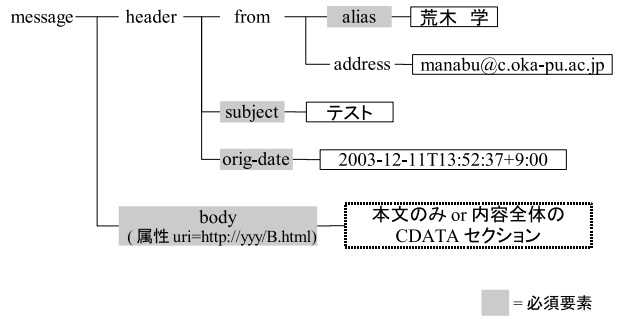

図 8　図 6 の文書 B を統合した例

メッセージを統合する際に，取得したメッセージすべてを統合したいとは限らない．ある条件に該当するものだけを統合したり排除したりする必要もあると考える．正規表現を利用してメッセージの各要素に対して条件を設定し，統合の際にフィルタリングを行う事で，メッセージの取捨選択が実現可能である．

4.　実装と評価

4.1　実　　装

Web サイトに対してメッセージの統合を行うソフトウェアを，Java (JDK 1.4.2) により図 9 のようなインタフェースで実装した．ビューワでメッセージを表示する際，メッセージ URI の有無によって HTML 文書と `CDATA` セクションのテキストのどちらを表示するか自動的に判断する．

4.1.1　XML 形式メッセージボックスの入出力

XML 形式メッセージボックスの仕様を RELAX NG を用いて策定し (2.3 仕様 1)，Relaxer [7] を用いて入出力機能を実現した．メッセージボックスの実体である XML ファイルを選択すると，DOM で XML ツリーを解析し，図 10 のように表示する．表示中のリストからメッセージボックス (XML ファイル) に出力することもできる．

4.1.2　Web 上のメッセージの取得と統合

複数のメッセージ URI が公開されているページ (3.2 図 4)

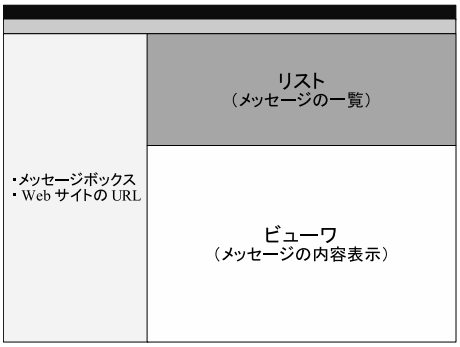

図 9　インタフェース

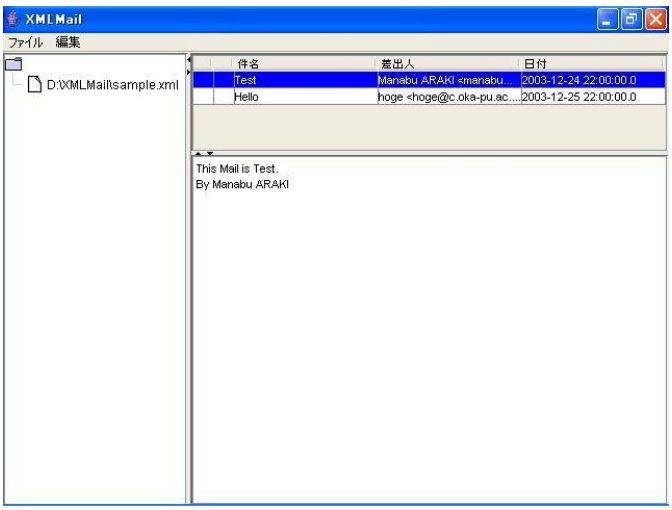

図 10　2.3 例 1 のメッセージボックスを表示

の URI を入力することで，各メッセージを取得して統合を行い，リストに出力できる (図 11)．ソフトウェア内部では，次のような

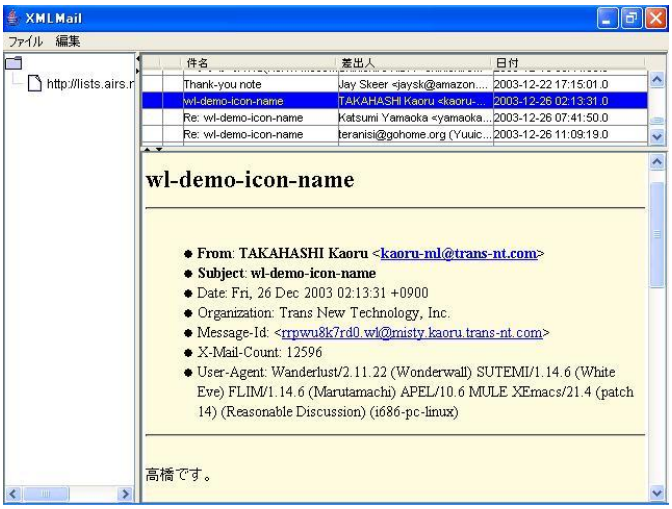

図 11　Web サイトにあるメッセージ [2] を取得して日付順に表示

な処理を行う．

（1）　入力された URI から各メッセージ URI を取得する．

（2）　各メッセージ URI に対して処理を行う．

(a)　HTML 文書の内容すべてを取得する．

(b)　各行に対して正規表現を使って HTML タグを除去

する.

(c)　HMM は使用せず, キーワードを使った全文検索を行いフィールドの記述してある箇所を探す.

(d)　ソフトウェア内に組み込んだキーワードとパターンを用いて, 各フィールドの情報を取得する. 発信日時情報の場合は, dateTime 型に変換する. これは, 複数のソースからメッセージの統合しても一貫した日付管理するためであり, 将来はタイムゾーンの違いも判断して統合するよう拡張する.

(e)　取得したフィールドの情報をもとに, メッセージとして統合する.

（3） 統合に成功したメッセージの一覧を, リストに取得順に出力する. ユーザがメニュー選択することで, 日付順にソートすることも可能である.

メッセージボックスと Web サイトにあるメッセージを, 同時に統合することも可能である (図 12). 日付順に並べることで, そのメッセージがメッセージボックスにあったものか Web サイトにあったものかを, 意識することなく管理できる.

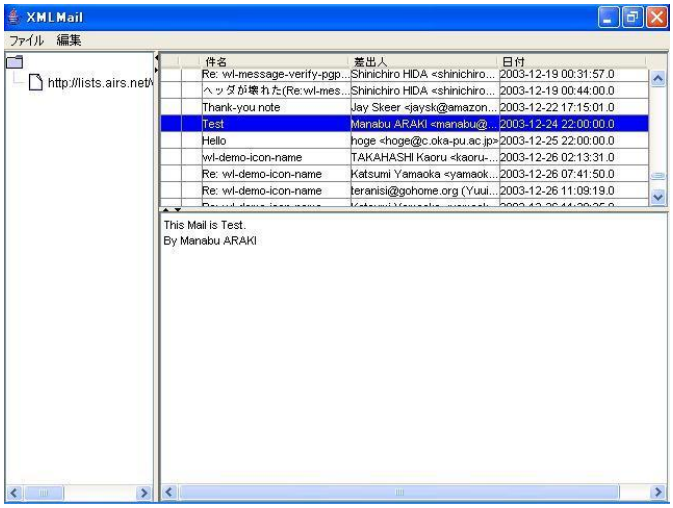

図 12　2.3 例 1 と Web サイトにあるメッセージ [2] を取得して日付順に表示

4.2　評　　価

本論文の提案をもとに実装を行い, Web サイト [2] を対象にメッセージの統合を行った. 汎用性を重視して策定した XML 形式メッセージボックスを基盤に, Web サイトのメッセージの統合を実現した. 統合したメッセージを同じインタフェースで扱うことで, 一貫性のある管理を行う事ができる.

課題として, キーワードとパターンを外部情報として持たすべき点を, 統合の実現を優先したためにソフトウェアの内部に持っている点が挙げられる. これは外部情報やパターンの仕様が実装できる段階までに至っていないことが原因である. また HMM を使った複数のフィールドの取得に関しても実装できておらず, 効率の悪いキーワード検索方法を利用している. この部分を実装すれば学習機能も確立され, 効率があがり処理速度の向上が見込める.

今回は独立したソフトウェアとして実装したが, 既存の MUA やソフトウェア (Emacs 等) に組み込める形態の方が, ユーザにとってさらに使いやすいと感じた. 今後はそういった方向性も探っていきたい.

5.　統合したメッセージの応用

5.1　複数の Web サイトを統合

メーリングリストアーカイブが複数の Web サイトに存在するような場合, 各サイトをすべて統合することで同じメッセージ集合として管理が可能になる. その結果ソートや検索が一元的に行えるため, 各サイトを巡回してメッセージの検索を行うよりも効率が良い. ユーザがテーマ別にアーカイブを統合することもできる.

5.2　Web サービスと連携した情報検索

自分の管理しているメッセージに対して検索を行った結果, 思うような情報が見つからないという場合が考えられる. そういった場合は外部の情報も検索対象にしたい. そこで Web サービスと連携した情報検索を行うことを提案する. Web サービスとは, オープンな技術を使って記述 · 呼び出し · 公開 · 発見が可能な, ネットワーク上に存在するソフトウェア部品のことである. 本研究では例として, 広く仕様が公開されている AmazonWebService と GoogleWebAPIs の利用について述べる. AmazonWebService では書籍, GoogleWebAPIs では Web サイトの情報を検索 · 取得できる. それぞれの Web サービスから提供される 1 つの書籍 ·Web サイトの情報と 1 つのメッセージを, 表 4 のように対応させることが可能である.

表 4　書籍, Web サイトとメッセージの対応

メッセージ	書籍	Web サイト
件名	書籍名	タイトル
発信元	著者 or 出版社	-
発信日時	出版年月	-
本文	要約	要約
URI	ISBN コード	URL

AmazonWebService で検索した書籍情報はメッセージとして十分な情報を持つので, メッセージとして統合をすることも可能である. 一方 GoogleWebAPIs で検索した Web サイト情報には, 発信元と発信日時に該当する情報が無いのでこのままではメッセージとは言えない. そこで一時的に, 発信元に GoogleWebAPIs を利用したことが分かるダミーのアドレスを与え, 発信日時には検索した日時を与える. そうすれば, 一時的にメッセージの形となる. 上記の方法で書籍と Web サイトの情報を一時的にメッセージの形にすれば, 自分の管理するメッセージに対する検索結果と同じ形で表示できる. さらにその検索結果をメッセージボックスに保存できる.

1 つの書籍 ·Web サイトと 1 つのメッセージを対応させる方法の他に, 1 つの Web サービスから提供される情報を 1 つのメッセージとして対応させる方法もある. 例えば発信元に Web サービス名, 本文に複数の書籍 ·Web サイトの詳細を記述したものを対応させると, あたかも Web サービスから送られたメッセージのように直観的に捉えることができる. この他にも Web サービスから得られる情報を利用すれば, 様々な応用ができる可能性がある.

5.3 RSS を利用したメッセージの共有

RSS とは RDF Site Summary の略で, Web サイトの見出しや要約などのメタデータを構造化して記述する XML ベースのフォーマットである. 主に Weblog などのサイト更新情報を公開するのに使われている. RSS は, 各記事 (更新情報) ご

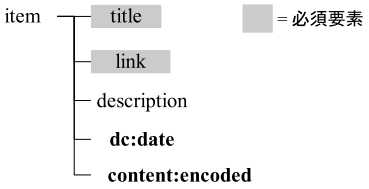

図 13 RSS1.0 の item 内タグ構造の概要

とに item 要素を持ち, 図 13 のようなタグ構造(注2)を持つ. 一般的には title には記事のタイトル, link には記事の URI, description には記事の要約を格納してある. この item にメッセージの情報を納め, 自分の管理しているメッセージを他人と共有することを実現することを提案する.

メッセージを納める方法を, 図 5 のように考えた. 件名は

表 5 RSS とメッセージの対応

メッセージ	RSS
件名	title
発信元	description
発信日時	dc:date
本文	content:encoded
URI	link

例 2 RSS での発信元記述例

```
<description><![CDATA[
  Alias: 荒木 学
  Address: manabu@c.oka-pu.ac.jp
]]></description>
```

title, URI は link に格納する. link は RSS において必須要素なので, URI を持たないメッセージの場合は公開サーバに本文内容を記述したファイルをアップロードして, そのファイルの URI を格納する. 発信日時は dc:date に dateTime 型で格納する. 本文は content:encoded に格納する. content:encoded に CDATA セクションで本文内容を記述すれば, Content モジュールに対応した RSS リーダを使うことで HTML タグ等を解釈して表示することも可能である. 発信元に相当する要素は見当たらないので, description に例 2 のように記述する. この書式は E メールのヘッダに準じた書式である.

RSS にメッセージを格納し, 図 14 のように公開する. ユーザは RSS リーダを用いて, 共有メッセージを閲覧することが可能となる.

(注2): 正確には Dublin Core モジュールと Content モジュールを用いて RSS を拡張したもの.

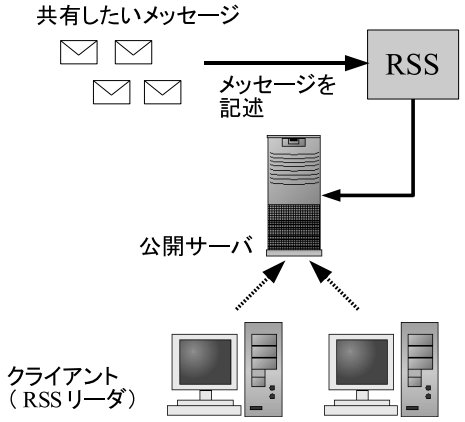

図 14 RSS での共有方法

6. 結論と今後の課題

本研究では XML 形式メッセージボックスの仕様を策定し, それを基盤にした Web 上のメッセージの取得 · 統合の手法を提案した. また実装を行い, Web 上のメッセージを統合した. さらに統合したメッセージの応用として, Web サービスと連携した情報検索や, RSS を使った共有方法を提案した. 統合したメッセージを XML 形式メッセージボックスに出力できることで, 他のアプリケーションから XML ファイルとして利用できる可能性も広がった.

課題として, Web サイトや MUA のメールボックスに対する統合の手法の改良や, 統合したメッセージの応用や問い合わせについての提案がある. またメッセージとして扱う範囲を広げ, 明確にしていく必要がある. 今後はこれらの課題を解決するように研究と実装を進めていく.

文 献

[1] 坪井 祐太: 蓄積された電子メールの XML 化の勧め. http://home9.highway.ne.jp/~tsuboi/index-j.html
[2] 寺西 裕一: Wanderlust Mailing List. http://lists.airs.net/wl/archive/
[3] 北 研二: 確率的言語モデル, 東京大学出版会, 1999.
[4] The Internet Society: RFC2822 Internet Message Format. http://www.ietf.org/rfc/rfc2822.txt, 2001.
[5] W3C: Date and Time Formats. http://www.w3.org/TR/NOTE-datetime
[6] James Clark: RELAX NG Compact Syntax. http://www.oasis-open.org/committees/relax-ng/compact-20021121.html
[7] ASAMI Tomoharu: Relaxer. http://www.relaxer.org/
[8] (株) 日本ユニテック DigitalXpress 編集部: SOAP UDDI WSDL Web サービス技術 基礎と実践 徹底解説. 技術評論社.