Javaを用いたアクティブメッセージ交換システム

林 哲也[†] 原田 実[††]

現状の電子メールで送信されるメッセージは，単に受信者に表示されるだけのパッシブなメッセージであった．本論文で提案するJAM（Java Active Message）は，送信先で実行されるJavaプログラムが電子メールメッセージの中に埋め込まれたアクティブメッセージである．JAMにより，GUIを用いた受信者とのインタラクティブなメッセージや，ユーザのエージェントとして働く能動的メッセージの作成が可能になる．本研究では，JAMをMIMEマルチパートメッセージの一種として扱うことにしたので，JAMを表すContent-Typeを“multipart/x-activemail”，埋め込まれるJavaクラスのContent-Typeを“application/x-java”と定義した．JAMを送受信するためのメーラJAMES（Java Active Message Exchange System）を開発して，いくつかの事例JAMを送受信することにより，Javaを用いたアクティブメッセージ交換の実現可能性と有効性を確認した．プログラム記述言語にJavaを用いることは，安全性，アーキテクチャ独立，親しみなどの点で有効であることが分かった．Javaアプレットがブロードキャスト型の実行可能コンテンツ送信媒体として広く普及しているのに対応して，JAMが1対1送信形態における実行可能コンテンツ送信媒体として広く利用できるのではないかと考えている．

Java Active Message Exchange System—JAMES

TETSUYA HAYASHI[†] and MINORU HARADA[††]

Messages exchanged by traditional emails are passive, because they are merely shown to the recipient. JAM (Java Active Message) we proposed is an active message containing Java programs that are to be executed on the recipient's computer. We have become to be able to compose an interactive message with GUI or an active message functioning as a sender's agent. Because JAM is a MIME multipart message, we defined Content-Type of JAM as “multipart/x-activemail” and Content-Type of Java class in JAM as “application/x-java”. We also developed JAMES (Java Active Message Exchange System), which is the mailer system of JAM, and actually exchanged some sample application JAM by using JAMES. Therefore we proved that the active message exchange using Java is feasible and moreover effective, because Java is safe, architecture independent and close to users. We are expecting JAM is able to be used widely as a communication medium of executable contents, while Java applets have already spreaded as a broardcast medium.

1. はじめに

近年のインターネットの爆発的な普及につれ，電子メールを用いたコミュニケーションが広く社会に浸透してきた．電子メールは，高速かつ非同期なメッセージ送信，メッセージの蓄積や一斉送信を行うことができる，電話とも手紙とも違った新しいコミュニケーション手段としてなくてはならないものとなってきた．

† 青山学院大学大学院理工学研究科経営工学専攻
Department of Industrial and Systems Engineering, Graduate School of Science and Engineering, Aoyama Gakuin University
†† 青山学院大学理工学部経営工学科
Department of Industrial and Systems Engineering, Faculty of Science and Engineering, Aoyama Gakuin University

従来のインターネットメールでは通常，テキストメッセージしか扱えなかったが，MIME標準によりマルチパートメッセージや，バイナリデータを扱えるようになった．しかし現在，MIMEを用いて交換されているメッセージは，テキスト文書にせよマルチメディアデータにせよ，受信者に表示されるだけのパッシブメッセージである．

一方，WWW上のコンテンツは，以前はHTML文書やそこに埋め込まれた画像などのパッシブコンテンツだけであったが，最近ではMacromedia社のShockwaveやSun Microsystems社のJavaアプレットなどを用いてユーザとのインタラクションを持つコンテンツが作成されている．

本研究では，インタラクティブなメッセージをインターネットメールのプロトコルを用いて送受信する．

そのためには通常の電子メールを拡張して，受信者のコンピュータ上で実行されるプログラムをメッセージに埋め込んで送信する．このように受信者とのインタラクションを持つメッセージは通常，アクティブメッセージと呼ばれる．

アクティブメッセージに関する研究として，ATOM-ICMAIL[1]，Active Mail[2]，Safe-Tcl[3]などがある．しかし，これまでの研究ではActive Mailのようにマルチプラットフォームでは動作しなかったり，ATOM-ICMAILで用いるLISPやSafe-Tclのようにユーザに親しみの薄い言語であったりして，実際に利用するには問題があった．アクティブメッセージを設計するにあたり，メッセージに埋め込むプログラムの記述言語の選択は，アクティブメッセージの特性を決めるうえで重要な問題である．本研究ではJavaを用いることにし，提案するアクティブメッセージをJAM（Java Active Message）と呼ぶことにする．

Javaを採用した理由は，Javaにはセキュリティを実装するためのクラスがあり，マルチプラットフォームで動作するなど，アクティブメッセージとして好ましい性質を有しているからである．また，Javaはプログラマが習得しやすい言語であり，現在ではWWWサーバ上に数多くのJavaアプレットが存在するなどすでに多くのユーザが存在して，記述しやすさの点で評価されていることも考慮した．

JAMの設計に関しては，メーラとJAMを別々に機能改善できるように，またJAMが通常の電子メールを継承するように，JAMとメーラのためのクラスライブラリを構築した．JAMのAPIはJava Appletクラスと同様であり，既存のJavaプログラム開発環境を利用してJAMを容易に作成することができる．

JAMはJavaアプレットのようなインタラクティブなコンテンツであるだけではなく，非同期，蓄積性，同報性といった電子メールの特徴も持つ．JAMを用いると，GUIによる表示と入力を行うメッセージや，受信者の応答に基づいて処理を行うメッセージを送信することができる．たとえば，GUIを用いたアンケートやダイレクトメール，インタラクティブな地図や学習教材などである．このように，JAMは新しいメッセージ形態だけではなく，新しい電子メールサービスを生み出すものと期待される．

本論文ではJAMの構成と形式，JAMを送受信するために開発したメーラ（メールリーダ）であるJAMES（Java Active Message Exchange System）の機能と処理方式，事例JAM，JAMとJAMESの有効性について論じる．

2. JAMの設計

2.1 JAMの構成

JAMはMIMEマルチパートメッセージの一種であり，各ボディパートに埋め込まれたプログラムとそのプログラムに参照されるデータから構成される．本論文では以降，JAMに埋め込まれたプログラムを“プログラムメッセージ”，参照されるデータを“データメッセージ”と呼ぶことにする．JAMは1個以上のプログラムメッセージと0個以上のデータメッセージを持つことになる．プログラムメッセージは1個以上のJavaクラスファイルから構成される．データメッセージは1個のデータファイルと対応する．つまり，JAMは1個以上のJavaクラスファイルと0個以上のデータファイルを含むMIMEメッセージである．

2.2 JAMのMIMEタイプ

MIMEメッセージは，それ自身がどのようなデータであるのかをContent-Typeヘッダにより宣言することになっている．JAMに関しても，メーラがメッセージを表示するうえで，JAMに応じた処理を行うことができるようにJAMタイプを宣言する．本論文ではJAMについて，2つのMIMEタイプを定義する．1つはJAM全体に対してであり，もう1つはプログラムメッセージに対してである．

JAM全体に対するタイプを“multipart/x-activemail”と定義する．これはトップメッセージヘッダで宣言される．前半のmultipartの部分は，他のマルチパートメッセージと同様に，JAMがいくつかのパートに分かれていることを表す．後半のx-activemailでメッセージ内にJAMのプログラムメッセージを含むことを表している．そこでJAMのトップメッセージヘッダには次のようなヘッダフィールドを記述する．

```
Content-Type: multipart/x-activemail;
  boundary="partSeparator"
```

プログラムメッセージは1個以上のJavaクラスファイルから構成されるので，Javaクラスファイルに対してデータタイプを“application/x-java”と定義する．これはJAMを処理するメーラが，このヘッダを持つボディパートのデータを他のバイナリデータと区別するためである．application/x-javaタイプは3個のパラメータを必要とする．

“package”パラメータには，このボディパートに含まれるJavaクラスのパッケージ名を指定する．様々なユーザがクラスを作成して送信する中で，パッケージ名はクラス名の衝突を防ぐ．JAM作成者がプログラムメッセージを作成する際には，構成するクラスの

パッケージ名を，電子メールのアドレスに基づいて世界的に一意に命名する．たとえば電子メールアドレスがusername@somewhere.comならば，パッケージ名はcom.somewhere.usernameとする．

“type”パラメータには，“main”か“class”のいずれかを指定する．このボディパートに含まれるクラスが，JAMとJAMESとのインタフェースやプログラムメッセージの開始メソッドを持つ，3.2節で述べるActiveMessageクラスを継承しているならばmainを指定する．それ以外のmainと指定したクラスを補助するためのクラスであればclassと指定する．

“name”パラメータには，埋め込まれたクラスファイルのファイル名を指定する．通常は，クラス名に拡張子“.class”を付けたものである．

電子メールアドレスがusername@somewhere.comの人がSampleというクラスをJAMに埋め込む場合，このボディパートに記述するヘッダフィールドは次のようになる．

```
Content-Type: application/x-java;
  package="com.somewhere.username";
  type="main"; name="Sample.class"
```

3. メーラシステムJAMES

ここでは，本論文で提案するJAMの送受信および表示を行うために開発したメーラシステムJAMESについての説明を行う．JAMESの開発には，Java言語とその開発ツールであるJDK（Java Developers Kit）1.02を用いた．使用したコンピュータはNEC PC98と東芝製DOS/V機，OSはMicrosoft Windows95とWindows NT4.0である．Javaはアーキテクチャに依存しないので，Java Virtual Machineが実装されているシステムであれば，JAMESはいずれの環境でも動作する．

3.1 JAMESの機能と動作

JAMESはJAMの送受信を行うだけのものではなく，通常のテキストメッセージやMIMEメッセージの送受信もできる．JAMESは，メッセージの作成，送信，受信，削除，表示を行うための機能を持つ．それらは**図1**のようにGUIを用いて操作される．

3.2 ActiveMessageクラス

ユーザがプログラムメッセージを作成するには，ActiveMessageクラスを継承して，以下にあげるメソッドを再定義する．ActiveMessageクラスはJAMESにより提供され，JAMESと同じjamパッケージに含まれる．ActiveMessageクラスは抽象クラスであり，またPanelのサブクラスである．

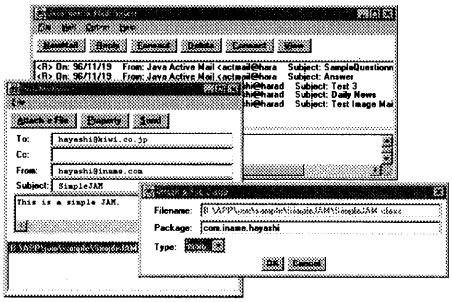

図1 JAMESの動作画面
Fig. 1 GUI of JAMES.

destroy() JAMESの終了時に行う処理を定義する．

init() GUI部品を配置したり，データの読み込みや変数の初期化などの処理を定義する．インスタンスが作成された直後に呼び出される．

start() ActiveMessageオブジェクトを表示するためのウィンドウが表示されたときに行う処理を定義する．

size() ActiveMessageオブジェクトの表示に必要な幅と高さを指定する．

stop() ActiveMessageオブジェクトを表示しているウィンドウが閉じられるときに行う処理を定義する．

このようにActiveMessageクラスはAppletクラスと同様のメソッドを持つので，Javaアプレットと同様に作成できる．さらに，ActiveMessageクラスはJAMESとのインタフェースとなる以下の変数とメソッドを持つ．

mail インスタンス変数mailは，このプログラムメッセージが埋め込まれたJAMを表しているMailオブジェクトを持つ．変数mailを用いると，JAM送信者などのメッセージヘッダにアクセスすることができる．

getDataDir() データメッセージが一時的にファイルに復元されるディレクトリパス名を返す．プログラムメッセージは，このパス名とファイル名を用いてデータメッセージにアクセスする．

ActiveMessageクラスに関連するクラス群を表すクラス図を**図2**に示した．ここで，PostOfficeクラスはメッセージの送受信と管理を行うJAMESの基幹となるクラスである．Mailクラスは電子メールメッセージを表すクラスであり，通常のテキストメッセージ，MIMEメッセージ，JAMのいずれにも対応する．MessageBodyクラスはMIMEマルチパートメッセージのボディパートを表し，HeaderMessageクラスはヘッダ部分を表す．HeaderFieldクラスはヘッダに書かれているヘッダフィールド1行を表す．

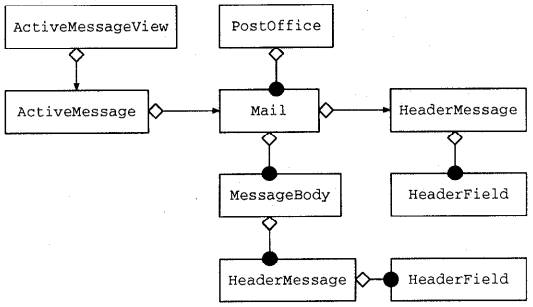

図2 ActiveMessage に関するクラス図
Fig. 2 Class diagram related to ActiveMessage classes.

3.3 プログラムメッセージの作成

3.2 節で説明したメソッドを用いて作成したプログラムの例を図 **3** に示す．作成したソースファイルは Java コンパイラでコンパイルをしてクラスファイルを作成する．作成したクラスファイルをメッセージに埋め込んで JAM を作成する．

現在では Java プログラムを開発するためのツールが，開発中あるいは発売されている．その中でも，Symantec 社の VisualCafe や Sun Microsystems 社の Java Studio はコーディングをすることなくプログラムの作成が可能である．JAM に関しても，これらのツールを利用することで，プログラムメッセージの作成が容易になる．あるいは，事前に用意されるプログラムメッセージテンプレートを利用することもできる．

3.4 JAM の作成と送信

JAMES は JAM だけではなく，テキストメッセージ，MIME マルチパートメッセージのいずれも作成し，送信できる．JAM の作成と送信は，通常の MIME マルチパートメッセージを送る方法とほぼ同様である．つまり作成されたプログラムメッセージをファイルとして添付する．JAMES で新規作成コマンドを選択すると，メッセージ作成のためのウィンドウが表示される．そこでは次のように，メッセージの作成と送信を行う．

(1) 送信先，同報先，題目のヘッダ情報を入力する．
(2) 添付ファイルコマンドを選択し，先に作成したプログラムメッセージのクラスファイルを，作成中のメッセージに添付する．
(3) 添付したクラスファイルを選択し，プロパティコマンドを選択する．プロパティダイアログが開くので，package パラメータと type パラメータの値を設定する．package パラメータにはパッケージ名を指定する．type パラメータには添付するクラスが ActiveMessage クラスを継承したものであれば main，それ以外は class を指定する．
(4) 送信コマンドを選択し，作成した JAM を送信する．

3.5 JAM の送受信形態

通常のインターネットメッセージと同様に，JAMES は JAM を SMTP により送信し，POP3 により受信する．その際に JAM はサーバと図 **4** のように送受信される．これは通常の MIME メッセージと同様であり，クラスファイルは BASE64 エンコーディングされてメッセージに埋め込まれている．メッセージが JAM であることを示すために，2.2 節で述べたように，トップメッセージヘッダで multipart/x-activemail を，クラスファイルが埋め込まれているボディパートでは application/x-java を宣言している．

3.6 JAM の解釈と表示処理

JAMES は POP サーバから図 4 のようなメッセージを受信する．これを解釈して，JAM として受信者に表示を行うまでの JAMES が行う処理を以下で説明する．

(1) 受信したメッセージのトップメッセージヘッダを調べ，MIME-Version ヘッダがないまたは Content-Type ヘッダがない場合は，テキストメッセージとして処理をする．
(2) トップメッセージヘッダの Content-Type が multipart/x-activemail ではない場合は，通常の MIME メッセージとして処理を行う．
(3) 各ボディパートを読み込み，Content-Type が application/x-java であれば，埋め込まれているファイルをパッケージディレクトリへ復元する．その他のタイプであればデータディレクトリへ復元する．ここでデータディレクトリとはデータメッセージの復元に用いられるディレクトリで，通常は JAMES が指定するテンポラリディレクトリである．パッケージディレクトリとはプログラムメッセージの復元に用いられるディレクトリで，パッケージ名により決まる．JAMES が指定するパッケージルートディレクトリが，たとえば “c:\mailclasses”（Windows の場合）であり，復元するクラスのパッケージ名が com.somewhere.username であるならば，パッケージディレクトリは “c: \mailclasses \com \somewhere \username” となる．
(4) 再度各ボディパートのヘッダを調べ，Content-Type が application/x-java であり，かつ type パラメータの値が main であるならば，該当するクラスのインスタンスを生成する．

```
package jp.ac.aoyama.ise.haradalb.hayashi;

import jam.*;
import java.awt.*;

public class Example extends ActiveMessage {
    static String[] day = {"Sunday", "Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday"};
    Button buttonSend;
    Choice choiceDay;

/** 内部変数を初期化し，GUIコンポーネントを配置します */
public void init() {
    this.setLayout(new BorderLayout());
    Panel panelCenter = new Panel();
    panelCenter.setLayout(new GridLayout(0, 1));
    panelCenter.add(new Label("When are you free?"));
    choiceDay = new Choice();
    for(int i=0; i¡day.length; i++) choiceDay.addItem(day[i]);
    panelCenter.add(choiceDay);
    this.add("Center", panelCenter);
    buttonSend = new Button("Send");
    this.add("South", buttonSend);
}

/** アクションイベント処理 */
public boolean action(Event event, Object arg) {
    if (event.target == buttonSend) {
        selectedSend();
        return true;
    }
    return super.action(event, arg);
}

/** 「Send」ボタンが押されたときの処理 */
public boolean selectedSend() {
/* メールの送信者のアドレス */
    String sender = mail.header.get("from").body;
/* メール受信者（回答者）のアドレス */
    String me = mail.postOffice.getProperty("mailaddress");
/* メールメッセージの作成 */
    String mes = "#Answer\n";
    mes = mes + "I'm free on " + choiceDay.getSelectedItem() + "\n";
/* 新しいMailオブジェクトを作り送信する */
    Mail reply = new Mail(sender, "", me, "Answer", mes, mail.postOffice);
    reply.send();
/* このフォームを閉じて，プログラムを終了する */
    this.close();
    return true;
}

/* 表示領域のサイズ */
public Dimension size() {return new Dimension(200, 100);}
}
```

図3 JAMプログラムメッセージの例
Fig. 3 Sample program of JAM program message.

(5) 生成したインスタンスに対して初期化（init()）メソッドを呼び出し，JAMを表示するためのウィンドウを開き，開始（start()）メソッドを呼び出す．

以上，JAMの作成，送信，受信，表示までの過程を簡単に図示すると**図5**のようになる．

```
From: Tetsuya Hayashi <hayashi@haradalb.ise.aoyama.ac.jp>
Message-Id: <9702110453.AA22719@haradalb.ise.aoyama.ac.jp>
X-Mailer: JAMES ver 0.31
Mime-Version: 1.0
Date: 11 Feb 1997 04:53:14 GMT +900
To: actmail@haradalb.ise.aoyama.ac.jp
Cc:
Subject: Example
Content-Description: "Mail with base64 encoded attachement"
Content-Type: multipart/x-activemail;
    boundary="JAM_4351_k$c3Pm3hT7bs2q"

This is a MIME multipart message.

-JAM_4351_k$c3Pm3hT7bs2q Content-Type: application/x-java;
    package="jp.ac.aoyama.ise.haradalb.hayashi";
    type="main";
    name="Example.class"
Content-Transfer-Encoding: Base64
Content-Disposition: attachment;
    filename="Example.class"

yv66vgADAC0AwwgAsAgAfAgAfQgAwggAwQgAiggAnggAe
owgAbggAagcAnAcAvAcAkQcAfgcAjwcAewcAsgcAcgcAmgc
        — 中略 —
AAYACADAAJIAAQC1AAAASQAEAAAAAAAAtEAe9ABhG
DlNZEAYSAlOzACixAAAAAAAQB

-JAM_4351_k$c3Pm3hT7bs2q-
```

図4 JAMの送受信形態
Fig. 4 Message transactions exchanged by SMTP and POP3.

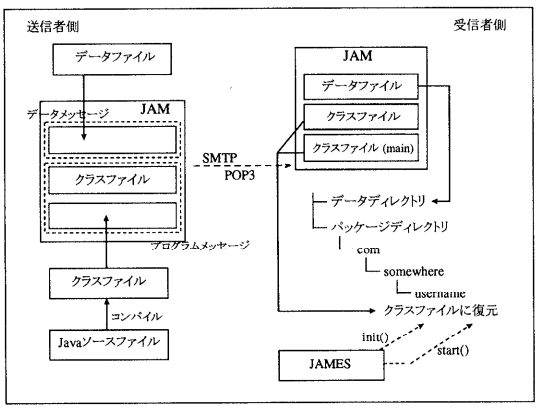

図5 JAMの作成から表示までの処理
Fig. 5 Processes from composing to viewing JAM.

4. 利用事例

JAMおよびJAMESの有効性を確認するために，事例としてのいくつかのJAMを作成した．本論文ではその中から2つを取り上げて説明する．

4.1 GUIを用いたアンケート

インターネット上でアンケートを行うには，電子メールとWWWの2つの手段がある．WWWを利用したアンケートの長所として，GUIを用いた容易な回答の入力があげられる．また，CGIを用いて動的に質問内容を作成することや，回答の集計も行うことができる．しかしWWWを利用したアンケートは待ち受け型だといえる．アンケートを作成したWWWサーバにユーザがアクセスしない限りは，回答が得られないからである．そこで現状ではアンケートの存在をユーザに認識させるために，WWWサーバのURLを電子メールで送信することが多い．

一方，電子メールを利用したアンケートには，公知性があるという長所がある．アンケートはユーザの電子メールボックスに届けられるために，ユーザに対してアンケートの存在を認識させることができる．しかし電子メールの場合の回答は，テキストで書かれたアンケートを編集して行うために入力が大変であったり，自動集計のために削除してはならない文字があったりした．

JAMを用いてアンケートを行うと，GUIを利用できることと公知性があることの，それぞれの特長を生かすことができる．例として作成したアンケートJAMは**図6**のように回答者に表示される．

作成した事例JAMにより，JAMを用いたアンケー

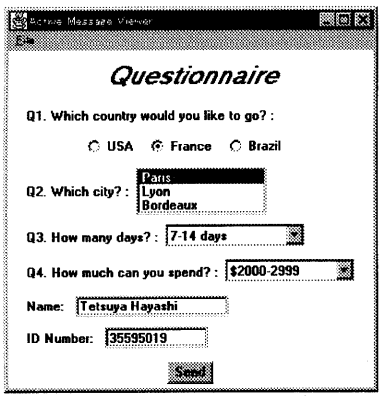

図6 アンケートJAMの表示
Fig. 6 GUI of a sample questionnaire JAM.

トには次の利点があることを確認した.

- マウスを使った回答など, GUIを用いた分かりやすい表示と容易な入力が可能になる.
- 入力した値が妥当でない場合は, 警告ダイアログを出して値の変更を促すなど, 送信前に入力内容に対する妥当性の評価を行うことができる.
- 条件分岐を用いて回答内容に応じた質問を動的に作成することができる.
- 回答内容に対して, 分析などの何らかの処理を行った結果を返信することができる.

4.2 作画ツールとサンプルデータ

MIME メッセージや FTP などを利用すると, ネットワークを通してアプリケーションソフトのデータを他のユーザに送信することができる. しかし, これらの手段を用いたデータの送信では, データを編集するためのエディタや見るためのビューワが, 受信者のコンピュータ上にインストールされている必要があった.

この問題を解決するために, マウスを用いて作画を行うツールとそのサンプルデータをJAMを利用して送信することを行った. このメッセージは**図7**のように伝達され, 処理される.

(1) データ送信者のコンピュータ上で作画ツールを実行してデータを編集する.
(2) JAMを作成して送信する. ここでプログラムメッセージは作画ツールであり, データメッセージは作成したデータである.
(3) 受信者のJAMES上で作画ツールが実行される. 受信者はデータを編集して, そのデータを返信することができる.

作画ツールを含むJAMの作成により, 次にあげる有効性を確認した. このようなJAMの利用はソフトウェアの管理コストを低減させる1つの方法であると

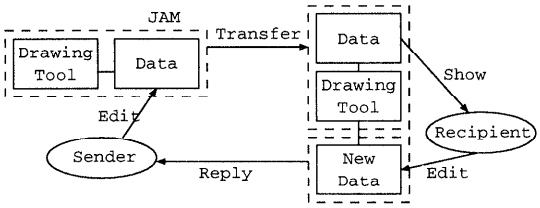

図7 作画ツールとサンプルデータを含むJAMの動作
Fig. 7 Sample JAM including a drawing editor and a sample data.

考える.

- 送信者は, 受信者のソフトウェア環境を気にせずにデータを送信することができる.
- 受信者は様々なデータに対応するための様々なソフトウェアをインストールする必要がなくなる.

5. 評価と考察

5.1 アクティブメッセージにおけるJavaの有効性

アクティブメッセージの設計に関して最も重要なことは, メッセージに埋め込むプログラムの記述言語を何にするかである. 本研究ではこの問題に対してJavaを選択し, JAMを設計した. JAMESおよび事例JAMの開発を通して考察したアクティブメッセージにおけるJavaの有効性について以下に述べる.

安全性 Java標準クラスライブラリにはSecurityManagerクラスがあり, JAMESにセキュリティを実装することで, 悪意のあるユーザからのJAMに対して機能制限をすることができる.

アーキテクチャ独立 ここでアーキテクチャ独立とは2つのことを意味する. 1つはマルチプラットフォームで動作することであり, もう1つはマルチユーザインタフェースに対応していることである. JAMはJavaを用いることで, アーキテクチャ独立なアクティブメッセージとなることができた. また既存のアクティブメッセージは, テキスト入力フィールドやボタンなどを提供するGUI部品の数が少なかった. しかしJavaを用いるとGUI部品を自作することができるので, 従来のアクティブメッセージに比べて高機能なGUIを提供できるようになった.

親しみのある言語 従来のアクティブメッセージで用いられているLISPやSafe-Tclなどのプログラミング言語は一般のユーザになじみが薄いものであった. このことはアクティブメッセージの利用が広まらない1つの原因と考えられる. Javaは従来から親しまれているCやC++と同様の構文を多数採用しながら, ポインタを用いないことや自動ガーベッジコレクションを実装していることなど, プログラマが容易に習得し

表1 JAMとアプレットの比較
Table 1 Comparing JAM to Applet.

	JAM	アプレット
送信形態	1:1（個人宛て送信）	1:多（ブロードキャスト型）
プロトコル	SMTP	HTTP
保管サーバ	受信者側メールサーバ	送信者側WWWサーバ
削除	受信者による	送信者による
送信開始	送信者の要求	受信者の要求
保存	可能	不可能
ネットワーク	一括	継続的

やすい言語になっている．またJavaアプレットの普及につれ，Javaは多くの人にとって親しみのある言語になった．

高速 従来のスクリプト言語で書かれた方式とは異なり，JAMのプログラムメッセージはアーキテクチャ独立なバイトコードにコンパイルされている．今後のジャストインタイムコンパイラやJava仮想マシンの高速化により，JAMはそのメリットを享受することができ，インタプリタ方式に比べて性能の改善余地が大きい．

オブジェクト指向 Javaはオブジェクト指向言語であるので，あるJAMのために作成したプログラムメッセージを別のJAMに再利用することが容易である．たとえばユーザが作成したGUI部品を，いくつものJAMの中で使用できる．

マルチスレッド Javaはマルチスレッドをサポートしているので，複数の処理を同時に扱うことができる．マルチスレッドにより得られる1つの利点は，GUIを用いたユーザインタラクションの性能が改善されることである．

5.2 JAMとJavaアプレットの比較

JAMもJavaアプレットも，ネットワークを通して実行可能なコンテンツ，すなわちJavaプログラムをリモートコンピュータに送信するという点では同じである．JAMはアプレットと同様のAPIを持ち，似ている点も多い．そこでJAMとアプレットとを比較し，利用形態などの違いについて言及する．

根本的な違いは送信形態にある．JAMは基本的に1対1の送信形態であり，個人宛てに送信される．同報先を指定すると複数の人に送信することができるが，送信先は送信者が指定する．一方，アプレットは1対多のブロードキャスト型の送信形態である．送信者はアプレット受信者を特定することができない．また，コンテンツの送信は，JAMの場合は送信者の要求により行われるが，アプレットの場合は受信者の要求により行われる．

アプレットを利用すると，クラスファイルや使用するリソースを必要に応じてサーバから読み出すので，連続的なネットワーク接続が必要である．JAMは受信者側で保存可能であり，またメールサーバから一括して読み出されるので，実行時にはネットワーク接続の必要がない．

JAMとアプレットとの比較を**表1**にまとめた．アプレットはブロードキャスト的な実行可能コンテンツとしてインターネット上で普及しているが，JAMは個人宛に送信される実行可能コンテンツとして普及することが期待できる．また携帯情報端末などの通信回線の安定性が低い環境におけるアクティブコンテンツの送信にも役立つことが期待できる．

5.3 モーバイルエージェント

インターネット上でJAMを安全に使用できるように，5.1節で述べた必要なセキュリティについてさらに研究し，JAMESに実装する必要がある．しかし，信頼できるユーザから送信されたJAMに対してセキュリティを緩和すると，JAMの応用範囲が広がると考える．送信者を信頼できるかは，送信者のメールアドレスと電子署名により判断する．

社内利用などの信頼できるユーザ間では，携帯情報端末から社内データベースを検索するJAMを送信して，検索中はネットワーク接続を切断することができる．また，発信者が指定した何人かのユーザを経由するルーティングJAMや，さらには受信者が不在の場合はページャやFaxへの転送なども考えられる．これらの場合，データベースへのアクセスや第三者へのメッセージの送信をJAMが行うことを認める必要がある．

このようにJAMに高度な機能を持たせると，JAMはモーバイルエージェントとなりうる．General Magic社のTelescriptやIBM東京基礎研究所のAgletsなどの既存のモーバイルエージェント環境との違いは，既存のエージェントは場所から場所，つまりマシンからマシンへ移動するのに対して，JAMは人から人，つまり個人のメールボックスに対して非同期に移動することである．

6. まとめ

本論文では，現在最も注目されているプログラミング言語であるJavaを用いたアクティブメッセージJAMを提案した．JAMを送受信するためのメーラJAMESを開発し，いくつかの事例JAMを送受信することにより，JAM交換の実現可能性を実証した．アクティブメッセージのプログラム記述言語にJavaを用いることは，安全性，アーキテクチャ独立，親しみなどの点で有効であることが分かった．JAMを用いると，GUIを用いたインタラクティブなメッセージや，エージェントとして受信者とのインタラクションを代行するメッセージなど，今までの電子メールにない新しい機能を持つメッセージの作成が可能になる．

今後の課題として，セキュリティの実装，プログラムメッセージを一時ファイルではなくメモリ空間に復元すること，ユーザ認証機構の導入などがあげられる．

参考文献

1) Borenstein, N.: Computational Mail as Network Infrastructure for Computer-Supportcd Cooperative Work, *Proc. CSCW '92 Conference*, Tronto (1992).
2) Goldberg, et al.: Active Mail – A Framework for Implementing Groupware, *Proc. CSCW '92 Conference*, Tronto (1992).
3) Borenstein, N.: EMail With A Mind of Its Own: The Safe-Tcl Language for Enabled Mail, *ULPAA '94*, Barcelona (1994).
4) Borenstein, N. and Freed, N.: MIME (Multipurpose Internet Mail Extensions), Part One: Mechanisms for Specifying and Describing the Format of Internet Message Bodies, RFC1521, Bellcore, Innosoft (1993).
5) Moore, K.: MIME (Multipurpose Internet Mail Extensions), Part Two: Message Header Extensions for Non-ASCII Text, RFC1522, University of Tenessee (1994).
6) Flanagan, D.: *Java in a Nutshell – A Desktop Quick Reference for Java Programmers*, O'Reilly & Associates (1996). 永松健司（訳），日本サンマイクロシステムズ（監訳）：JAVAクイックリファレンス，オライリージャパン (1996).
7) JAVA WORLD編集部：Javaの基本コンセプトと可能性，*JAVA WORLD*, IDGコミュニケーションズ (1996).
8) OPEN DESIGN（編）：電子メールシステム完全マスタ，*OPEN DESIGN*, No.8, CQ出版社 (1995).
9) 西田豊明：エージェントオブジェクトの次に来る基盤技術，日経コンピュータ 1995.11.27, 日経BP社 (1995).

(平成9年 2 月28日受付)
(平成9年10月 1 日採録)

林 哲也

1995年青山学院大学理工学部経営工学科卒業．1997年同大学院理工学研究科経営工学専攻博士前期課程修了．1997年SAPジャパン（株）入社．

原田 実（正会員）

1975年東京大学理学部物理学科卒業．1980年同大学理学系大学院博士課程修了．理学博士．同年（財）電力中央研究所担当研究員．1989年より青山学院大学理工学部経営工学科助教授．自動プログラミングシステム，ソフトウェアの要求理解や設計の自動化，オブジェクト指向CASE，株式投資エキスパートシステム，自律ロボットシステムなどの研究を行う．1986年（財）電力中央研究所経済研究所所長賞受賞．1992年人工知能学会第6回全国大会優秀論文賞．訳書「ソフトウェアの構造化設計法」（日本コンピュータ協会），編著書「自動プログラミングハンドブック」（オーム社），監修書「CASEのすべて」（オーム社）など．IEEE, ACM, AAAI, 人工知能学会，日本ロボット学会，OR学会，経営工学会各会員．