特 別 研 究 報 告 書

題 目

グループワークを支援するメーリングリスト機構の提案

研究室担当教員の署名

提 出 者

藤原 啓輔

岡山大学工学部 情報工学科

平成 21 年 2 月 9 日 提出

要約

メーリングリスト (以下 ML) は，同好の不特定多数を結ぶツールとして発展してきた．このため，特定の話題についての議論や情報収集は，ML によって行われることが多かった．しかし，現在では，このような議論や情報収集は，Wiki や掲示板で行われるのが一般的となっている．一方で，現在 ML は，小さな部署や特定のプロジェクトにおいてコミュニケーション手段や文書の共有スペースとしてさかんに利用されている．つまり，ML が大規模で不特定多数向けの利用から，小規模で特定のグループ向けの利用へと変化している．

ML 管理や運営の負担を軽減するツールとして，Mailman や fml などの ML 管理ツールと呼ばれるものがある．ML 管理ツールは，メンバ登録作業の自動化など，主に大規模な ML の利用に適した機能を提供している．しかし，小規模で特定のグループ向けの利用に特化した機能は提供していない．

そこで，本論文では，小規模で特定のグループ向けの利用を支援する ML サービスの柔軟な構成を支援するため，グループワークを支援するメーリングリスト機構を提案した．

提案機構は，従来の ML 機構の問題点に対処するため，ML サーバ自身が積極的にメールを投稿して利用者を支援する機能，Web コンテンツとのマッシュアップ，および組織に合致した自由度の高いアーカイビングといった機能を提供する．

こうした機能を提供するため，提案機構は，主要機能群と外部連携モジュール群，および両者を連携させる連携処理部からなる．主要機能群は，メールの配送，メールのアーカイビング，および ML メンバの管理を行う．外部連携モジュール群は，連携する外部の Web コンテンツやソフトウェア群とのインタフェースを統一して選択可能な形で拡張機能を提供する．これらのモジュールにより，ユーザに様々なサービスを提供する．

また，提案機構は，現在，所属する研究室内で研究・開発・実験的に運用されているグループウェアである LastNote との連携や，Zimbra という豊富なマッシュアップ機能を持つ Web メーラとの連携が可能である．また，一部の機能を Ruby on Rails を用いて実装した．

目次

図 目 次

表 目 次

第 1 章

はじめに

メーリングリスト (以下，ML) は，複数の人に同時に電子メールを配送する機構である．このため，同好の不特定多数を結ぶツールとして発展してきた．例えば，Linux に関する特定の話題についての議論や情報収集をするため，同好者による ML を利用するといったことが一般的であった．しかし，現在では，このような議論や情報収集は，Wiki や掲示板で行われることが多くなっている．

一方で，現在 ML は，小さな部署や特定のプロジェクトにおいてコミュニケーション手段や文書の共有スペースとしてさかんに利用されている．つまり，ML が大規模で不特定多数向けの利用から，小規模で特定のグループ向けの利用へと変化している．また，ML 管理や運営の負担を軽減するツールとして，Mailman[1] や fml[2] などの ML 管理ツールと呼ばれるものがある．ML 管理ツールは，メンバ登録作業の自動化など，主に大規模な ML の利用に適した機能を提供している．

しかし，小規模で特定のグループ向けの利用に特化した機能は提供していない．そこで，特定のグループでの利用を支援する ML サービスの柔軟な構成を支援するため，グループワークを支援するメーリングリスト機構を提案する．提案機構の提供する機能例として，ML サーバ自身が積極的にメールを投稿し，利用者を支援する機能，Web コンテンツとのマッシュアップ，および組織に合致した自由度の高いアーカイビング機能，が挙げられる．

第 2 章

従来の ML とその機構

2.1 ML の現状

ML は，特定の話題に関心を持つグループなどで議論や情報交換を行うツールとして発展してきた．例えば，オープンソースの開発や使いこなしに関する ML がその代表例である．具体的には，GNU/Linux の開発や利用者向けの ML がある．参加者は，多い ML では，数千人をゆうに越える．そのため，メンバ管理や配送エラー処理の手間を軽減するために ML 管理ツールが利用されることが多い．

しかしながら，近年では，このような大規模な ML は減少し，既存の ML も流量が減少しつつある．その代わりに Web の掲示板や Wiki の設置が増加し，これらに議論の場が移されつつある．例えば，FreeBSD の開発プロジェクトは，2008 年 11 月 16 日付けのアナウンスで，Web ベースのディスカッションフォーラムを告知している [3]．アナウンス中では，このフォーラムは，あくまでも ML の補完であるとしつつも，これまでにない動きである．また，Mandriva Linux 開発者向け ML(cooker ML)[4] の流量は，2007 年まで年間 4-5 万通あったものが，2008 年 12 月の時点で 1 万 4000 通弱となっている [5]．また，GNU/Linux の一般ユーザ向けの ML は，流量，加入者共に減少傾向にある．これには，様々な理由の説明があるものの，ML というツールそのものからの利用者離れであると指摘する声もある [6]．

一方，会社内や研究室内の小さな部署や特定のプロジェクトのグループにおいて，ML は，コミュニケーション手段や文書の共有スペースとしてさかんに利用されている．こうした小規模で特定のグループによる ML は，以下の特徴を持つ．

(1) 参加者は少数で固定的

ML に参加しているメンバは数十人程度である．また，メンバはお互いを認識しているため，メールアドレスから個人を特定できる．

(2) コミュニケーションは ML 以外にも発生

情報の交換は ML 以外でも行われる．例えば，メンバ同士は直接会って会話し，情報を交換する．また，会議日程といったグループの予定をスケジュールとして登録し，情報を共有する．

(3) 文書の共有スペースとして利用

議事録や会議資料をメールで送信することで，文書を共有する．

このように ML の利用は大規模で不特定多数の向けの利用から，小規模，特定のプロジェクト向けの利用へと変化している．

本章の以降では，従来の ML 機構の例として MTA と ML 管理ツールを取り上げる．また，それぞれの手法での ML 管理方法について説明する．

2.2 MTA による ML

2.2.1 エイリアスの特徴

MTA(Message Transfer Agent) とは，ネットワーク内でメールを配送するソフトウェアである．MTA を用いて ML を作成する場合，エイリアスを定義する必要がある．ここで，エイリアスとは「別名をつけること」を意味する．例えば，A というユーザのメールアドレスに B というエイリアスを定義すると，B 宛のメールは，A のメールアドレスに届く．また，1 つのエイリアスに複数のメールアドレスを指定することも可能である．この場合，B 宛のメールは，複数人のメールアドレスに送られる．このため，エイリアスを定義することで ML としての機能を実現できる．以降では，ML にメールアドレスが登録されているユーザを ML メンバと呼ぶ．

本節の以降では，MTA の例として Postfix[7] を取り上げ，Postfix のエイリアスによる ML 管理方法について詳細を述べる．

2.2.2 エイリアスの定義方法

エイリアスは，エイリアスファイルを編集することにより定義する．MTA が Postfix の場合，使用するエイリアスファイルの場所は，Postfix の設定ファイル (/etc/postfix/main.cf)

に記述されている (デフォルトでは，/etc/postfix/aliases に設定されている)．エイリアスによる ML の定義方法として，エイリアスファイルに直接メールアドレスを記述する方法，および，メールアドレスを列挙したファイル (以下，リストファイル) を読み込む方法がある．以下でこれらの詳細を述べる．

(1) エイリアスファイルに直接メールアドレスを記述する方法

エイリアスファイルに直接メールアドレスを記述する方法の場合，以下をエイリアスファイルに記述することで ML を作成できる．

```
<エイリアス>：<ML メンバのメールアドレス>，<ML メンバのメールアドレス>
```

<ML メンバのメールアドレス > の代わりに < ユーザ名 > を指定することもできる．

<ML メンバのメールアドレス >，および < ユーザ名 > の指定方法を以下に示す．

(A) メールアドレスを指定

例えば,「user-a@example.com」のようにメールアドレスを指定する．

(B) ユーザ名を指定

例えば,「user-a」のように，ユーザ名のみを指定する．このとき，Postfix に設定された自サーバのドメイン名を，メールアドレスのドメイン名として補完する．(例えば，自サーバのドメイン名が「example.com」の場合「user-a@example.com」と認識される．)

(2) リストファイルを読み込む方法

リストファイルを読み込む方法の場合，まず，リストファイルを用意する必要がある．リストファイルの記述例を以下に示す．

```
user-a
user-b@example.com
```

このように，ユーザ名やメールアドレスを改行区切りで列挙する．

次に，エイリアスファイル内で読み込むリストファイルを指定する．

例えば，/etc/postfix/list/test というリストファイルを用意したとする．このとき，エイリアスファイルへの記述例は，以下のようになる．

```
test： :include:/etc/postfix/list/test
```

2.2.3 新規 ML の作成例

以下の (1) と (2) を実行することで，ML：test@example.com を作成できる．

(1) /etc/postfix/aliaces にあるエイリアスファイルを編集

エイリアスの定義方法は，2.2.2 項を参照のこと．

(2) 編集内容を保存し，newaliases コマンドを実行

newaliases コマンドを実行しないと変更したエイリアスがシステムに反映されない．このため，コンソール上で以下のコマンドを実行する．

```
# newaliases
```

2.2.4 ML の管理方法

2.2.2 項で示したエイリアスの定義方法における ML の管理方法を以下に示す．

(1) エイリアスに直接メールアドレスを記述する方法

エイリアスファイルに直接メールアドレスを記述する方法で ML を管理する場合，ML メンバの追加や ML メンバのメールアドレスの変更がある毎にエイリアスファイルを編集し，newaliases コマンドを実行する必要がある．ここで，エイリアスファイルの記述方法を誤ると，当該のエイリアスは動かなくなる．さらに，エイリアスファイルにおいて設定を誤ったエイリアス以外にも，設定を誤ったエイリアス以降に記述されているエイリアスも動かなくなる．また，エイリアスファイルの編集には管理者権限が必要となることから，ML 管理における計算機管理者の負担が大きくなる問題がある．

(2) リストファイルを読み込む方法

リストファイルを読み込む方法で ML を管理する場合，変更内容はリストファイルに記述する．この際，変更内容を反映するため，newaliases コマンドを実行する必要はない．また，リストファイル内で記述方法に誤りがあったとしても，他のエイリアスの動作に影響を与えることはない．さらに，この方式の場合，リストファイルの保管場所をユーザのホームディレクトリに設定することができる．このため，リストファイルの管理を計算機管理者以外のユーザに任せることで，ML における ML メンバ管理の負担を分散させ，計算機管理者の負担を軽減できる．

以上より，MTA においては，リストファイルを読み込む方法で ML の管理を行うのが望ましい．

しかし，どちらの方式もエイリアスファイルの管理は計算機管理者が行わなければならない．また，メールのアーカイビング機能が無いため，ML に途中から参加した ML メンバは，過去のメール内容を参照できない．

2.3 ML 管理ツールによる ML

2.3.1 特徴

ML 管理ツールとは，ML の管理や運営する管理者の負担を軽減するための機能を提供するものである．主な機能として以下の機能がある．

(1) メンバ登録作業の自動化

ML 登録用のメールアドレス宛にメールを送信すると，自動でメールアドレスが登録される．

(2) メールのアーカイビングと全文検索

送信されたメールを過去ログとして保存し，メール内容を検索できる．

(3) ダイジェスト (まとめ送り) 機能

ML 宛に送られてきたメールをすぐに ML メンバに送信せず，1 日分のメールをまとめて送信する．

こうした機能を提供することで ML 管理者の負担の軽減し，ML メンバによる ML の利用を支援している．ML 管理ツールの代表的なものとして Mailman や fml と呼ばれるソフトウェアがある．

Mailman は，Web インタフェースからの ML の作成や削除する機能，配送エラーとなったメールを自動で処理する機能，および作成した ML を外部に紹介するための ML 紹介ページを自動作成する機能を提供している．また，メンバの登録や修正といった ML 管理は，Web 上とコマンドラインのどちらからでも行うことができる．

2.3.2 Mailman の設定方法

Mailman を動作させるには，HTTP サーバと MTA，および Python の動作する環境が必要となる．表 2.1 に今回構築した Mailman の動作環境を示す．本節の以降では，Mailman による ML 管理方法について述べる．

表 2.1 Mailman の動作環境

環境	バージョン情報
ML 管理ツール	Mailman 2.1.9-7
OS	Debian Etch
HTTP サーバ	Apache 2.2.3-4
MTA	Postfix 2.3.8-2
使用言語	Python 2.4.4-2

2.3.3 新規 ML の作成例

以下の (1) から (4) を実行することで，ML：test@example.com を作成できる．なお，Mailman では ML をリストと呼ぶ．

(1) サイトパスワードを設定

サイトパスワードとは，Web インタフェースによる ML の作成や管理を行う際に必要なパスワードである．以下のコマンドを実行することでサイトパスワードを設定できる．

```
# /var/lib/mailman/bin/mmsitepass パスワード
```

(2) 管理用の ML の作成

Mailman で作成した全ての ML を管理するため，管理者用の ML を作成する必要がある．新規に ML を作成した場合,「ML の作成が完了した」という通知を行うメールが作成した ML の管理者に送信される．管理者用の ML の作成は，以下のコマンドを実行する．

```
# /var/lib/mailman/bin/newlist mailman
```

(3) ML を作成

ML 作成画面 から ML を作成する．ML 作成画面を図 2.1 に示し，以下で説明する．ML 作成画面は，Web ブラウザで `http://<ドメイン名>/cgi-bin/mailman/create` からアクセスできる．作成する ML のドメイン名は Mailman の設定ファイルで設定できる．ここでは ”example.com” とする．

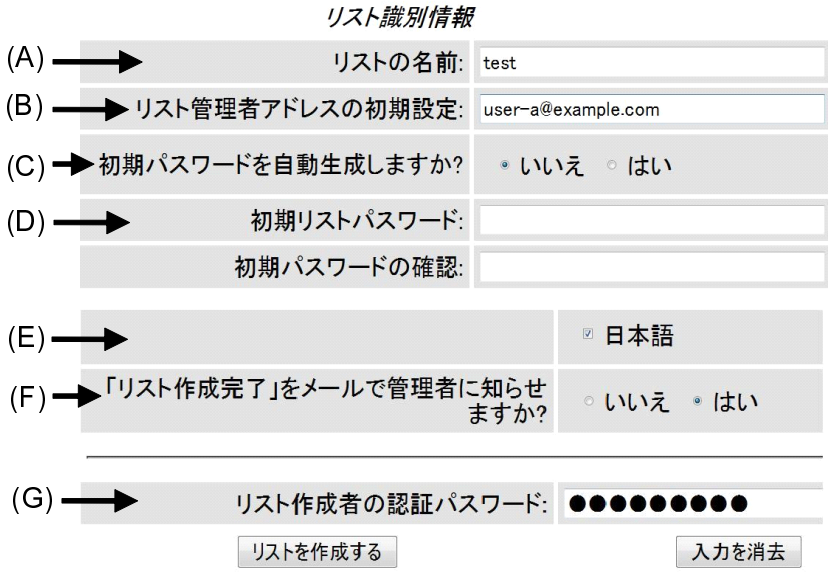

図　2.1　ML 作成画面

ML 作成画面に以下の (A) から (C)，および (E) から (G) の項目を入力する．

(A) リストの名前：test

(B) リスト管理者アドレスの初期設定：user-a@example.com

(C) 初期パスワードの自動生成：はい

自動生成された初期パスワード (以下，リストパスワード) は，上記の項目 (B)：「リスト管理者アドレスの初期設定」で入力したメールアドレスに送信される．初期パスワードを自分で設定する場合は「いいえ」と設定し，(D) にパスワードを入力する．

(E) 言語の設定：日本語

(F) 「リスト作成完了メール」をメール管理者に知らせますか ?: はい

ML 作成が完了した場合，作成完了通知を「リスト管理者アドレス」に送信する．

(G) リスト管理者のパスワード：サイトパスワード

(4) メンバの追加

作成した ML に ML メンバを登録する．作成した ML の情報を `http://<ドメイン名>/cgi-bin/mailman/admin` から確認する．ML の一覧が表示されるので作成した ML を選択する．ML 管理者かどうか Basic 認証を求められるので，リストパスワードを入力する．認証を行うと，ML 管理画面に移動する．ML 管理画面を図 2.2 に示し，以下で説明する．会員管理を選択することで複数のメールアドレスを一括して ML メンバに登録できる．一括登録画面を図 2.3 に示し，以下で説明する．図 2.3 で表示されている記入方式に従って，メールアドレスを記述するか，リストファイルのアップロードを行う．ここでのメールアドレスの記述は，2.2.2 項のエイリアスファイルに直接メールアドレスを記述する方法に相当する．また，リストファイルのアップロードは，2.2.2 項のリストファイルを読み込む方法に相当する．これにより ML メンバを登録できる．

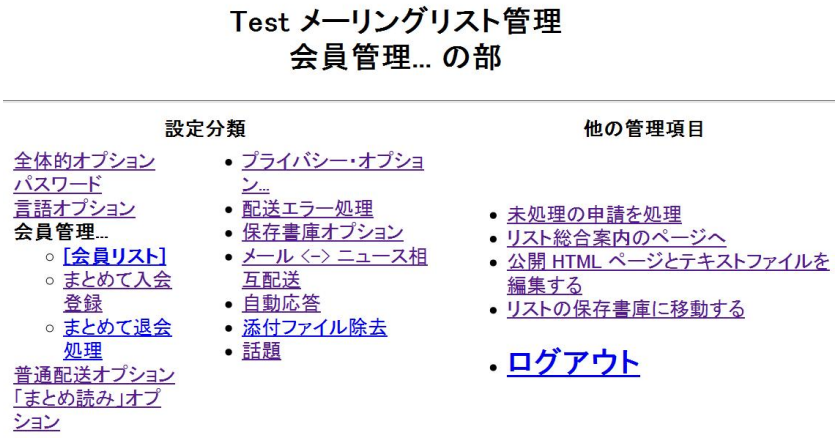

図　2.2　ML 管理画面

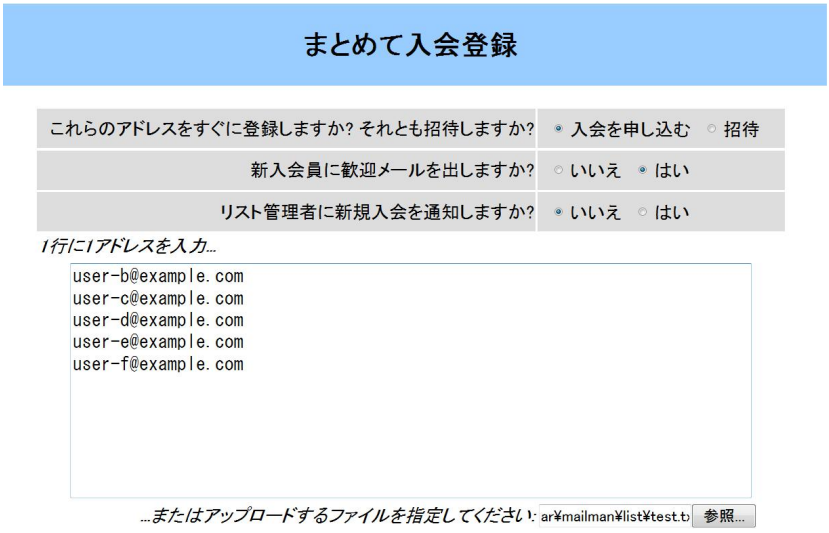

図　2.3　一括登録画面

2.3.4　ML の管理方法

Web インタフェースによる管理は，2.3.3 項の ML 管理画面 (図 2.2) から行う．

管理ページでは，メール配送の設定，メンバの追加，削除，およびメールアドレスの修正を行うことができる．こうした設定の変更を行う際，入力ミスを防ぐためのインタフェースを Mailman は提供している．また，MTA による ML 管理とは異なり管理ページでの設定の変更を反映させる場合，Mailman が自動で newaliases 相当の操作を行うことから，管理者が newaliases コマンドを実行する必要は無い．さらに，ML ごとに管理を行っているため，他の ML の影響を受けない．また，ML の管理者を計算機管理者とは別に設定できるため，管理にかかる負担を分散することができる．この場合，ML 管理者には作成した ML のリストパスワードを，計算機管理者にはサイトパスワードを割り当てる．

また，アーカイブ機能により ML メンバ宛に送信したメールは，アーカイブに保存されている．このアーカイブは，全ての人が閲覧可能，あるいは ML メンバのみ閲覧可能，といった公開，非公開の設定ができる．

第 3 章

提案機構

3.1 従来の ML 機構の問題点

MTA のエイリアス機能によって ML を作成する場合，2.2.4 項で述べた問題がある．例えば，エイリアスファイルの編集は，計算機管理者のみ実行可能である．このため，新規 ML の作成は，必ず計算機管理者が行う必要があるため，計算機管理者の負担は大きい．また，メールのアーカイブ機能がないため，ML に途中から参加した ML メンバは，過去のメール内容を参照できない．

このような問題に対処しているのが，ML 管理ツールである．2.3.4 項で述べたように，Mailman は，計算機管理者の ML を管理する負担を分散できる．また，アーカイブ機能によって過去に送信されたメールは計算機上に保管され，ML メンバは Web ブラウザを通じて外部から，保存されたメールを閲覧できる．

しかし，従来の ML 管理ツールの提供している機能は，不特定多数を相手にした ML 向けの機能である．具体的には，2.1 節で述べたような近年の ML の利用形態を支援するような機能は提供されていない．このため，従来の ML 管理ツールには，以下の問題がある．

(1) 利用者を特定できない

各 ML メンバの嗜好に合わせたサービスを提供できない．

(2) グループ内で共用している API と連携できない

ML 単体での運用を支援しているため，Wiki や掲示板，スケジューラとの連携できない．

(3) 添付ファイルの管理機能が不十分

ML を文書の共有スペースとして利用することを想定しておらず，添付ファイル一覧の閲覧といった機能を提供していない．

3.2 要求

3.1 節で述べた問題点から，ML 機構に対する要求を以下に示す．

(1) 利用者に合わせたサービスを提供

小規模，特定のプロジェクトでの ML の利用では，ML メンバ同士は互いを認識しているため，個人に合わせたサービスが提供できる．

(2) グループ内で利用しているサービスと連携

コミニュケーションや情報の共有方法は，グループ内で共通している．こうしたコンテンツとの連携を行うことで利用者の支援を行う．

(3) 添付ファイルを整理

添付ファイルの管理システムを提供することで，文書の共有スペースとしての ML の利用を支援できる．

3.3 特徴

3.2 節の要求に対処するため,「グループワークを支援する ML 機構」を提案する．この提案機構は，以下の 3 つの機能を提供する．

(1) ML サーバ自身が積極的にメールを投稿し，利用者を支援

ML メンバが少数で固定的であるという点から，メールアドレスを基に利用者を特定できる．この特性を利用し，ML サーバから利用者に合わせて内容をカスタマイズしたメールを送信できる．

(2) Web コンテンツとのマッシュアップ

情報交換やコミュニケーションが ML 以外でも発生するのであれば，スケジューラといった他の外部の Web コンテンツと連携を行うことで，利用者の嗜好に合わせたサービスを提供できる．例えば，スケジューラと連携を行う場合，会議日程といったスケ

ジュール情報をメールで送信した際に，自動でスケジューラに登録するといった機能が考えられる．

(3) 組織に合致した自由度の高いアーカイビング

ML が文書の共有スペースとして利用されている点に着目して，会議資料として添付された文書ファイルを部署内の文書管理システムに自動で登録するといった連携を行う．また，個々のメールによって公開，非公開の設定を行う．これにより複数のグループで提案機構を利用していても必要な情報だけ共有できる．

第 4 章

設計方針

4.1 フレームワーク

3.3 節で述べた提案機構の特徴より，提案機構は，従来の ML 機構と比較して，柔軟な機能を提供する必要がある．

なぜなら，外部の Web コンテンツとのマッシュアップを行う場合，連携を行いたい全ての Web コンテンツを提案機構のシステム内に組み込むのは非常に手間がかかる．また，組織によって連携が必要なコンテンツとそうでないコンテンツが存在する．このため，システムと外部の Web コンテンツとの連携は，必要に応じて，連携の取りやめ，連携する Web コンテンツの変更といった操作を自由にできる必要がある．また，提案機構では，組織に合致した自由度の高いアーカイビングを提供する．このため，組織によってアーカイビングを支援するための機能を変える必要がある．このような理由から提案機構は従来の ML 機構と比較して，柔軟な機能を提供する必要がある．

こうした柔軟な機能を提供するため，提案機構では，決まった動きをする部分と組織によって大きく変わった動きをする部分に分ける必要がある．決まった動きをする部分とはメールの配送部分である．メールの配送部分とは，特定の ML に送信されたメールをシステムが受信し，それぞれの ML メンバ宛に配送するまでの動作を指す．組織によって大きく変わる部分は，外部で連携を行う Web コンテンツを指す．この Web コンテンツは Web サービスとして，提案機構と連携する形をとる．つまり提案機構は，ML フレームワークのような形になる．

4.2 API

4.2.1 準拠する方式

外部の Web サービスとの連携にあたり，連携する API を決定する必要がある．Web API を公開する際の方式として，SOAP(Simple Object Access Protocol) と REST(REpresentational State Transfer) がある．

REST とは，HTTP を使って通信する手法である．明確な仕様はないが，一般的に「HTTP の GET メソッドを使ってある URL にアクセスすると XML が返ってくる」ものを REST と呼ぶ．REST の「ある特定の URL にアクセスして XML を得る」という動作は，Web ブラウザが URL にアクセスして HTML を得る動作とまったく同じである．このため，Web ブラウザに URL を入力するだけで，動作を確認できる．また，HTTP と XML があれば利用できるため，多くの開発言語で利用できる．以上をまとめると，REST は「確認し易い」「利用し易い」という特徴を持っているといえる．

SOAP とは，SOAP メッセージという XML によってメッセージ交換をおこなう方法である．REST の対応するプロトコルは HTTP のみなのに対して，SOAP は SMTP などのプロトコルでも実現可能である．SOAP は，エンドポイントと呼ばれる場所に SOAP メッセージを送ると，結果の SOAP メッセージが返ってくる．SOAP は「拡張性が高い」という特徴を持っている．

提案機構は，Web インタフェースを提供しており，HTTP を用いて簡単に情報を授受できる REST と相性が良いと考えられる．このため，提案機構は REST インタフェースを用いて，外部と連携する．

4.2.2 外部の Web コンテンツとの連携

本項では，提案機構が外部の Web コンテンツに提供する情報，提案機構が取得する情報，および実際の外部の Web コンテンツとの連携方法について述べる．

提案機構が外部の Web コンテンツに提供する情報は，ML 宛に送信されたメールである．このため，外部の Web コンテンツは，メール本文，あるいは添付ファイルからスケジュール情報，会議資料といった文書ファイルを取得する．これに対して，提案機構が取得する情報は，外部の Web コンテンツの処理結果である例えば，スケジューラによって登録されたスケジュールの閲覧先や文書管理システムによって文書として保存された添付ファイルの保存先，といった情報を取得する．

こうして，情報を相互に交換するため，提案機構側と外部の Web コンテンツ側の両方に REST インタフェースを実装する必要がある．また，連携を行う Web コンテンツとの選択は，システムによって決定されるのではなく，提案機構を利用するユーザによって決定できるようにする．

第 5 章

システム設計

5.1　システム構成

提案機構のシステムの構成図を図 5.1 に示す．以下に，システムの各構成要素について詳細を述べる．

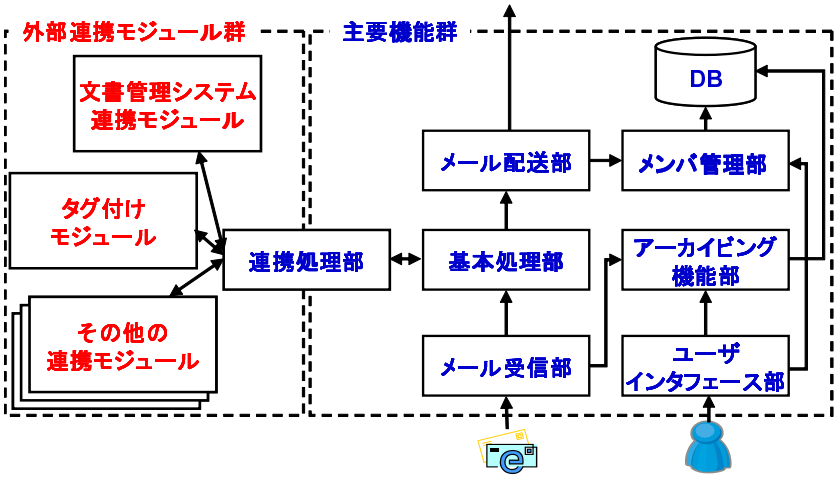

図　5.1　システムの構成

(1) 外部入力

外部入力は，ユーザからのアーカイブ閲覧や ML メンバの確認，および MTA からのメールの転送を表す．

(2) メール受信部

メール受信部は，MTA から転送されたメール全体を受信し，基本処理部，アーカイビング機能部に転送する．

(3) 基本処理部

基本処理部は，メール受信部から転送されたメール全体を連携処理部に転送する．転送されたメール全体は，連携処理部から各外部連携モジュールに転送され，内容が加工される．内容が加工された後，連携処理部から加工されたメール全体を受け取り，メール配送部へ転送する．

(4) 連携処理部

連携処理部は，基本処理部から転送されたメール全体を各外部連携モジュール全てに転送する．各外部連携モジュールは，それぞれの処理を行った後，必要に応じてアノテーションを処理結果として連携処理部に返す．アノテーションとは，与えられたデータに対して注釈として付与する関連情報である．例えば，スケジューラ連携モジュールは，スケジュール登録先の Web ページの URL を返す．

また，モジュールによってはアノテーションを返さない．

連携処理部は受け取ったアノテーションをメール本文の末尾に追記する．アノテーションを追記したメール全体を基本処理部へ返す．

(5) 外部連携モジュール群

外部連携モジュール群は，提案機構を支援するモジュールの総称である．各外部連携モジュールは，連携処理部からメール全体を受け取り，アノテーションを連携処理部に返す．例えば，スケジューラ連携モジュールは，メール本文からスケジュール情報を抽出し，スケジューラに登録する．スケジュール登録後，メール本文の末尾に登録されたスケジュールの確認先の URL をアノテーションとして連携処理部に返す．

(6) メール配送部

メール配送部は，基本処理部から転送されたメールを各 ML メンバに配送する．各 ML メンバのメールアドレスはメンバ管理部を通じて DB から取得する．

(7) ユーザインタフェース部

ユーザインタフェース部は，アーカイビング機能部と連携し，アーカイブされているメールを表示するインタフェースを提供する．また，メンバ管理部とも連携を行い，ML

メンバの確認や追加，および登録されているメールアドレスの変更を行うインタフェースを提供する．

(8) アーカイビング機能部

アーカイビング機能部は，メール受信部から転送されたメール全体を DB に保存する．また，ユーザインタフェース部の要求に応じて，DB から保存しているメール情報を取得し，結果を返す．

(9) メンバ管理部

メンバ管理部は，メール配送部やユーザインタフェース部からの要求に対して，DB から必要な情報を取得し，結果を返す．また，ユーザインタフェースを通じて ML メンバの情報が変更された場合，DB の更新を行う．

(10) DB(データベース)

DB は，ML 宛に送信されたメール全体，ML メンバの氏名やメールアドレス，および所属しているグループといった情報を保存する．

5.2 動作概要

提案機構の動作概要について説明する．図 5.1 の矢印に記されている英字は，以下で述べる動作概要の説明と対応している．

メール配送の流れは，以下のようになる．

(A) MTA からのメールの転送

(B) 受信したメール全体を保存するため, アーカイビング機能部に転送

(C) DB にメール全体を保存

(D) メール全体をメール受信部から基本処理部に転送

(E) 連携処理部にメール全体を転送し，処理結果を受信

(例: 過去ログの保存先の URL を追記したメール全体を処理結果として受け取る．)

(F) メール全体を各モジュールに渡し，処理結果を基本処理部に返却

(G) (F) で処理されたメール全体をメール配送部に転送

(H) メンバ管理部から ML メンバの情報の取得を要求

(I) ML メンバの情報を DB から取得し，結果を返却

(J) (I) で取得したアドレスに基本処理部から転送されたメールを配送

ユーザの ML メンバの確認や修正要求に対する処理の流れは，以下のようになる．

(K) ユーザから ML メンバの確認と修正の要求

(L) ユーザがメンバの確認，編集を行いたい ML を指定すると，ユーザインタフェース部はメンバ管理部にこの旨を通知

(I) 指定された情報を DB から取得し，結果を返却

(M) 修正要求を受けて，修正内容を DB に反映

ユーザのアーカイブ閲覧要求に対する処理の流れは以下のようになる．

(N) ユーザから過去ログの閲覧要求

(O) ユーザが指定した ML の情報をアーカイビング機能部に通知

(P) 要求されたメール情報を DB から取得し，結果を返却

5.3 試作

提案機構と連携を行う外部の Web コンテンツとして，現在，所属する研究室内で研究・開発・実験的に運用されているグループウェア (以下 GW) である LastNote との連携を行う．LastNote は，Ruby on Rails[8] で開発された Web アプリケーションである．また，Zimbra[9] という豊富なマッシュアップ機能を持つ Web メーラとの連携も行う．提案機構の機能の一部を Ruby on Rails を用いて実装した．実装環境を表 5.1 に示す．

作成した機能の一部を以下に示す．

(1) ML の作成機能

ML の新規作成を行う．作成した ML の一覧は，図 5.2 のように表示される．作成した ML は，メンバ編集や ML の削除，およびアーカイブの閲覧といった操作ができる．

表　5.1　提案機構の実装環境

OS	Debian Etch
使用言語	Ruby 1.8.2-1
Web アプリケーションフレームワーク	Ruby on Rails 1.2.6
データベース管理ソフト	SQLite 3.3.8-1.1
MTA	Postfix 2.3.8-2
使用コンポーネント	TMail 2.4.4-2

(2) アーカイビングの閲覧機能

各 ML のアーカイブを閲覧できる．アーカイブは，図 5.3 のように表示される．また，添付ファイルもアーカイビングされており，選択することで内容が確認できる．

図　5.2　ML 一覧表示画面

図　5.3　アーカイブ閲覧画面

第 6 章

おわりに

本論文では，小規模で特定のグループ向けの ML を支援する ML 機構について述べた．まず，従来の ML 機構の方式を示し，従来の ML 機構の問題点を明らかにした．そして，問題点に対処するため，グループワークを支援する ML 機構を提案した．提案機構は，従来手法の問題点への対処として，ML サーバ自身が積極的にメールを投稿し，利用者を支援する機能，Web コンテンツとのマッシュアップ，および組織に合致した自由度の高いアーカイビングといった機能を提供する．このため，提案機構は，従来の ML 機構と比較して柔軟な機能を提供する必要がある．

こうした柔軟な機能を提供するため，提案機構は，主要機能群と外部連携モジュール群，および両者を連携させる連携処理部からなる．

また，提案機構は現在，所属する研究室内で研究・開発・実験的に運用されている GW である LastNote や，Zimbra という豊富なマッシュアップ機能を持つ Web メーラとの連携を行っており，一部の機能を Ruby on Rails を用いて実装した．

残された課題は，評価方法の検討とその評価について，および連携する外部モジュールの検討とその実装である．

謝辞

本研究を進めるにあたり，懇切丁寧なご指導をしていただきました乃村能成准教授に心より感謝の意を表します．また，研究活動において，数々のご指導やご助言を与えていただいた谷口秀夫教授，田端利宏准教授に心から感謝申し上げます．

また，日頃の研究活動において，お世話になりました研究室の皆様に感謝いたします．

最後に，本研究を行うにあたり，経済的，精神的な支えとなった家族に感謝いたします．

参考文献

[1] Barry Warsaw, “Mailman,” http://www.gnu.org/software/mailman/

[2] 深町 賢一, “fml,” http://www.fml.org/index.html.ja

[3] “FreeBSD News Flash,” http://www.freebsd.org/news/newsflash.html#event20081116:01

[4] “Mandriva Linux 開発者向け ML(cooker ML),” http://lists.mandriva.com/cooker/

[5] Vincent Danen, “Sad metrics on community decline,” http://linsec.ca/blog/2008/12/05/sad-metrics-on-community-decline/

[6] 八田真行, “メーリングリストの死,” http://sourceforge.jp/magazine/08/12/08/0850251

[7] Wietse Venema, “Postfix,” http://www.postfix.org/

[8] David Heinemeier Hansson, “Ruby on Rails,” http://rubyonrails.org/

[9] “Zimbra Collaboration Suite 5.0,” http://www.zimbra.com/