2010 International Conference on Broadband, Wireless Computing, Communication and Applications

A Mailing List Management System Mashing-up with Web Services

Keisuke Fujiwara, Yoshinari Nomura, and Hideo Taniguchi
Okayama University
3-1-1 Tsushima-naka
Okayama, 700-8530, Japan
fujiwara-k@swlab.cs.okayama-u.ac.jp, nom,tani@cs.okayama-u.ac.jp,

***Abstract*—Mailing list (ML) was a popular system for discussing specific topics and was an easy tool for making birds of a feather. However, today, such kinds of MLs seem to be shrunk. Instead, we usually use SNS or Wiki systems for these purposes. Do you think ML was dead? No, you would receive many mails every day via several MLs. We think today's MLs are still alive changing their styles from mass service to small-internal service. The small-internal MLs are alive in small sections of offices and labs of universities. These kinds of MLs have different characteristics from traditional mass MLs. However, we don't have any good ML management system conforming with the new style MLs. We have to settle for using old-style popular ML administration systems in spite of the mismatch of use-case. We have developed a new ML management system, which helps the new small-internal MLs. In this paper, we explain that our ML system effectively helps users' group work mashing-up with other intranet or Internet web services.**

***Keywords*-mailing list; groupware; web service; mash-up**

I. Introduction

Mailing list (ML) is a system to discuss specific topics among the ML members. For example, open-source communities usually have their discussion MLs. Major ML such as Linux ML has many (over 1000) members.

However, recently, large MLs are gradually shrinking in number. Also, their ML-traffic is decreasing and shifting their place to web BBSs or Wiki systems. For example, FreeBSD project announced that their web-based discussion had been opened on November 2008[1]. In this announce, they added that their web-forum was opened as a complement to their **fine** mailing lists. At any rate, it was a new movement to FreeBSD community. For another example, we can see a report about traffic of Mandriva Linux cooker ML[2]. Before year 2007, they had 40,000 to 50,000 mails per year, however, in 2008, they had only 1,4000 mails[3]. Seeing the GNU/Linux ML for non-developers, both members and traffic are decreasing. We may come up with many reasons for this issue, as one thing we think, ML as a service platform is outdated.

However, more or less, we still depend on MLs and are noticed that these still-alive MLs are almost for small groups in our office or community, not for mass service. In these MLs, we discuss much confidential topics, and share some documents via attachment files.

Taking these discussion into account, we authors think that the ML management system for small groups has different requirements with the traditional ML system. However, we apt to use the traditional ML systems on making MLs.

In this paper, we propose a new ML management system, which helps the new small-group MLs mashing-up with their web systems. It will make heavy mail users migrate to the mash-up-based web world.

II. Problems of small MLs

A. Characteristics of small MLs

We think today's MLs for small-groups have these four characteristics:

1) Small members
   Every member knows one another. Almost all MLs have only tens of members. Every member can identify senders and recipients.
2) Used for sharing documents
   They tend to send their documents and minutes for their meeting. It is awful.
3) Much communication out of MLs
   They have Face-to-face communication, may have a kind of groupware, Wiki, and Intra-SNS tools.
4) Heavy ML users own MLs
   We sometimes feel the existence of enthusiasm gap about using MLs. Some (especially old = means high position in Japan) users have excessive emphasis on mail. As a result, we tend to lose a chance to select preferable alternative tools.

B. Requirements for ML management system

We authors think the ML management system for small groups should equip these three functions:

1) Customization of ML archive for each user
   Traditional mass ML systems were not conscious of each user. Therefore, their archiving style of ML is monotone and not able to create different information for each user. Since our new ML system will have small number of members and concrete membership for each user, it will be able to provide per-user customization of many aspects. such as per-user attachment file management and original archive view.
2) Provision of web API for outer web applications

978-0-7695-4236-2/10 $26.00 © 2010 IEEE
DOI 10.1109/BWCCA.2010.157

Modern communication tools provide users with rich mashing-up functions, which enable users to link their tools each other. From this point of view, the traditional ML systems appear to be a weakest link. Our Small group-oriented ML system should provide rich APIs and user-customizable archiving framework as an entrance of mash-up with other tools.

3) Encourage users for migration to web services
We have many communication tools other than mail in our office. It must be desirable for many users to introduce a calendar system instead of announcing meeting schedule via mail. Our system should encourage heavy mail users to migrate to the web world.

C. Traditional solutions for the problem

1) Using traditional ML management systems: We have discussed the requirements for modern ML management systems in section II-B. It is difficult for traditional ML management systems to fulfill the requirements, because of these two problems:

1) Lack of customization of ML archive
ML archives generated by traditional ML management systems are monotonic. They does not provide any per-user customization of archiving. Consequently, it is difficult for users to manage attached documents in archives.
2) Lack of web API for outer web applications
Traditional systems have some interfaces to users, such as the way of automatic subscription and unsubscription. However they do not have good interface to other applications.

For these reasons, traditional ML management systems do not fulfill the requirements mentioned in section II-B.

2) Mailer-centric solutions: As another solution, we have a choice to use a kind of custom-mailer for some specific purpose. Microsoft Outlook[4] might be the most heavy-hitting tool for this purpose. As you know, Outlook has its own address-book, schedule management tool, plug-ins for helping group collaboration, and much more... It is totally easy if all users use the single same rich-mailer as their boss told. However, as is often the case for heavy mail users (whom we want to care), users have difficulty accepting the change of their mailers.

Using web mailer such as Gmail[5] or Yahoo! mail[6] also might be a good solution for the problems. However, it has the same problem with the case of Outlook. On the top of that, old users have difficulty to store mail articles to outer cloud space.

III. The new ML management system

A. Main features

To solve the problems, discussed in previous sections, we propose a new ML management system. It has these functions:

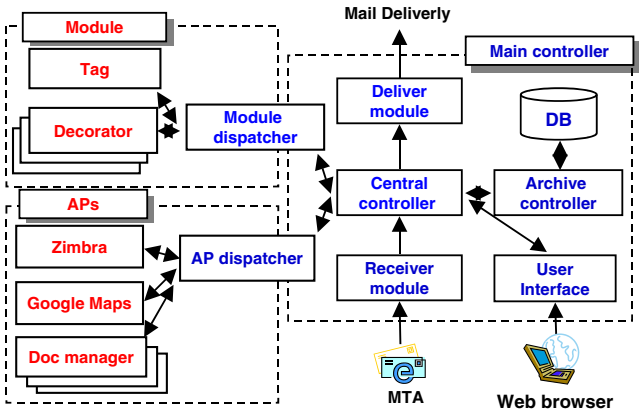

Figure 1. System diagram.

1) Flexible customization of ML archive
Traditional ML archivers are supposed to store mail articles in the as-is form. It is nothing more than a storage. Our system allows users to transform each article in many ways. It is called as *decoration*. Users can have their own set of decorators to customize the appearance of the archive.
2) Solid web service API for outer applications
Our system provide some web (REST) APIs for outer web applications. Using the APIs, the outer applications take advantage of ML member management, retrieval and use of information on the archive.
3) Acting as an entrance to web services
On delivering mails, our system inject some additional links (URLs) to their bodies of articles. These links in the body lead users to the related web service. Since, the mechanism is very simple, it will work with any mailers.

B. Design of the system

1) System diagram: Figure 1 shows the structure of our system. Our system consists of three parts as follows:

1) Main controller
Main controller consists of Receiver module, Central controller, Archive controller, Deliver module, User interface, and DB. It provides similar functions to the traditional ML management system, such as member-management, ml-creation/deletion, web page as the user interface.
2) Decorator modules
Decorator modules add useful additional information such as tags, links to outer web services, and some annotations.
3) Outer APs

Through the AP dispatcher, these outer APs can utilize the contents of the ML management system.

These three parts are connected via Central controller, Module dispatcher, and AP dispatcher:

2) Central controller: Central controller composes mail articles for delivery or for archiving. It uses proper decorator modules for each request. For example, on delivery of mails, the Central controller behaves as follows: (1) Receive an original mail article from Receiver module. (2) Forward the article to Archive controller and Module dispatcher. (3) Archive controller stores the article into DB. (4) Module dispatcher adds some additional information to the mail using proper decorator modules. We call the information as *annotation*. (5) The annotated mail is returned back to Central controller. (6) Central controller forwards the mail to Deliver module.

3) Module dispatcher: Module dispatcher controls the decorator modules. Decorator is a module to add some information to the original article. For example, the simple decorator adds the URL string which indicates corresponding archive web page. Another popular decorators add tags or URL links to outer web services triggered by matching particular string pattern in the body of the mail.

4) AP dispatcher: AP dispatcher is a gateway to outer web applications. AP dispatcher provides outer applications with archive information, member information, and APIs to manipulate the ML system. We introduce a set of REST API for this purpose.

IV. Prototype

We have implemented the ML management system introduced in section III-B. The system is built on our in-house groupware system, namely LastNote. LastNote has a function to sending mails using Postfix[7] and Ruby on Rails[8], so we added the proposed mechanism to the original mail-handling function. Currently, the ML management system has these functions:

1) Creation/Deletion of MLs
2) Manipulation of ML members
3) Add yellow-brick URLs on delivering mails:
(a) add URL bound for web archive page
(b) add URL for free/busy inquiry
4) Add decorations to archived mails:
(c) work with project management tool, Redmine[9]
(d) work with Google maps[10]
(e) work with in-house document management system
5) APIs for outer services:
(f) work with outer web mail system, Zimbra Collaboration Suite[11]

In following sections, we show some examples for function (a) to (f) with screen shots.

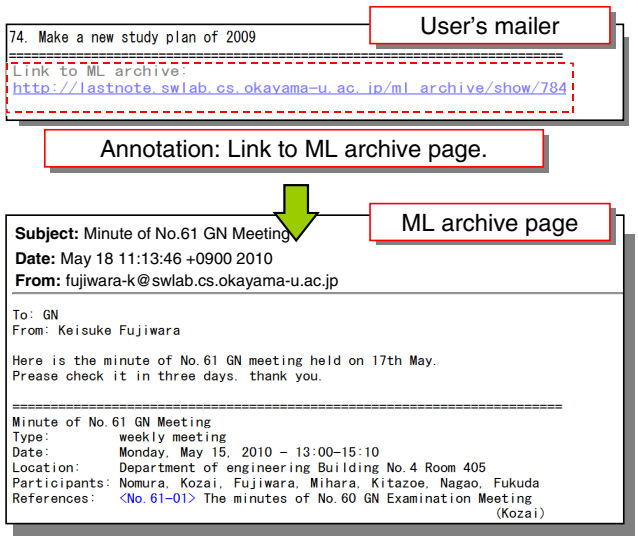

Figure 2. Annotation link to ML archive.

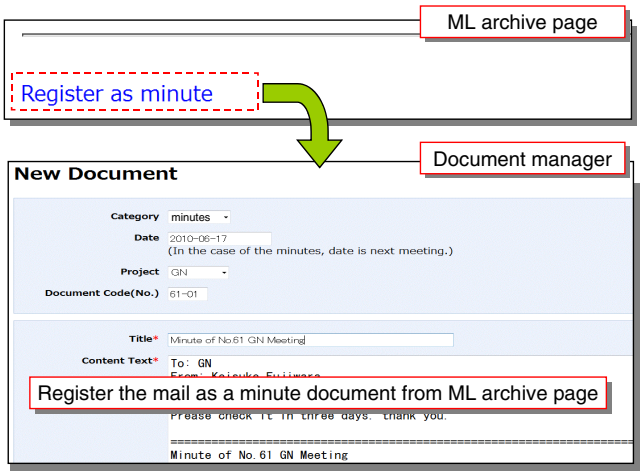

Figure 3. Register a mail as minute.

A. Add yellow-brick URLs on delivering mails

As examples for adding some annotations (URLs) on delivering mails, we show these two scenarios:

(a) add URL bound for web archive page
(b) add URL for free/busy inquiry page

By clicking these injected URLs, users will be routed to the corresponding web services. Therefore, we can deal these mails as if a part of web services.

1) Annotation link to ML archive: Figure 2 shows a mail in which the annotation link to ML archive is injected. This annotation shows the URL to the ML archive and is

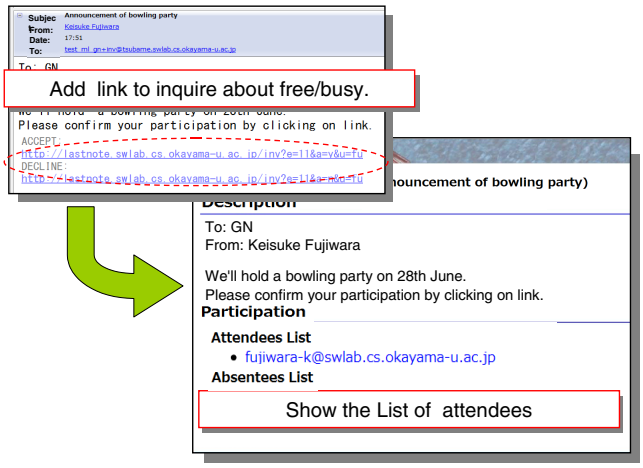

Figure 4. Free busy inquiry.

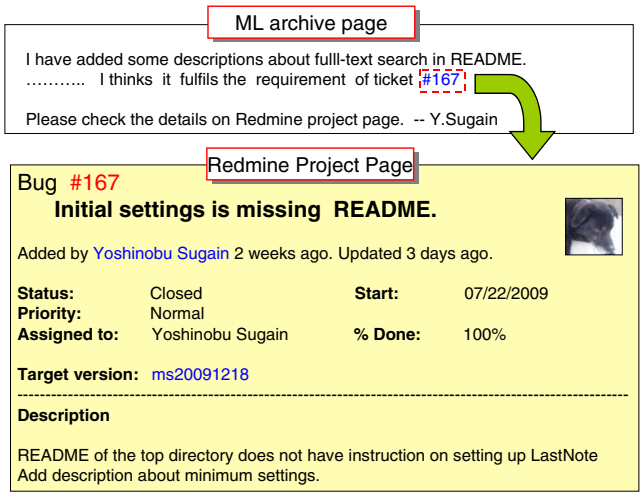

Figure 5. Link to ticket management system.

automatically injected on delivering the mail. By following the link, the user can move to the ML archive page. In the corresponding archive page, we will see more extra links than the original plain mail article. In Figure 2, it has the link "$< No.61 - 01 >$" which points to a PDF document in our document management system.

In Figure 3, ML archive has a link "Register as minute". By clicking the link, we can register the mail article as a minute of a meeting.

As seeing, the system provides the smooth path from a mailer to any web services through the archive page.

2) free/busy inquiry: The decorator is capable of injecting distinct link (annotation) for each receiver. Therefore, these links are useful to identify contexts or attributes of the visitors. Using these user specific information, our system provides various user-centric functions.

For example, we can use the annotation as a kind of inquiry. Figure 4 shows an invitation mail to a party. In this example, we can see the two inserted links. One is for accepting, another is for decline. By clicking one of them, the user is able to answer and be navigated to the corresponding web service.

B. Decorations in archive page

In this section, we show the three decorator modules effective in the archive page.

(c) Mash-up with project management tools
(d) Mash-up with the Google maps
(e) Mash-up with an in-house document management system

1) Mash-up with project management tools: Redmine is a flexible web-based project management application fully written in Ruby on Rails.

Redmine uses the term *ticket*, which express a small-short-term task in a project. Every *ticket* has its own unique ID number and some attributes such as owner, title, and description.

In the Redmine world, we frequently use the notation such as "#150" , which means "ticket ID 150". Therefore, in MLs of projects managed by Redmine, users naturally write the phrase like *#number TITLE....* in their mail articles. Redmine provides web page for each ticket in which we can manipulate the status of the ticket such as current status, assignment and description. In this situation, we want the strings like "#number" in articles of our ML archive to be clickable. That is, the string like "#number" should be annotated by the URL which navigates to the ticket page in Redmine. The URL for the ticket number *ID* has the form of:

```
http://example.com/issues/show/ID
```

Therefore, our decorator for Redmine behaves as bellows:

1) Perform string match to *#number*.
2) Enclose the matched string by the URL pointing to the *ticket*.
   URL string is like:
   http://redmine.example.com/issues/show/number

Figure 5 shows the archive page in which the above-mentioned link is added.

Our annotation scheme does not require any particular functions to the existing mailers with the exception of making the URL-like string beginning with http:// clickable. This exception ought to be affordable for every average mailers. We believe that introducing a rich mailer is a bad idea for this purpose. Since mailer is a basic tool for us,

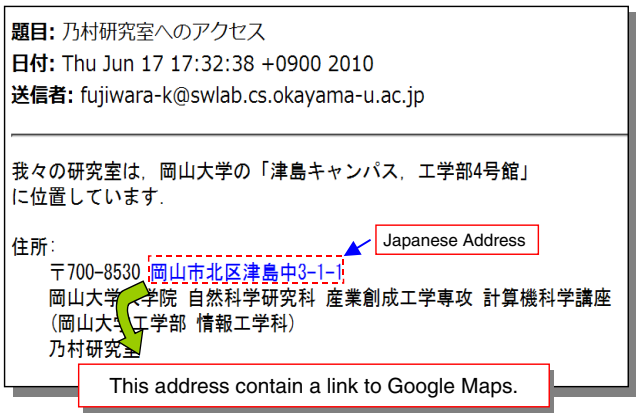

Figure 6. Link to Google maps.

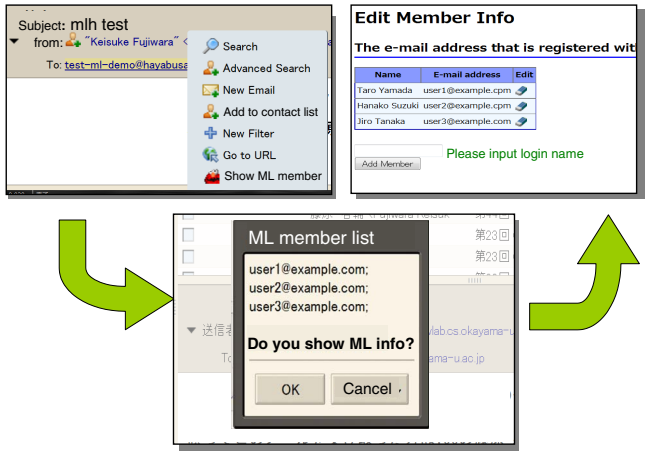

Figure 8. Mash up using Zimlet on Zimbra.

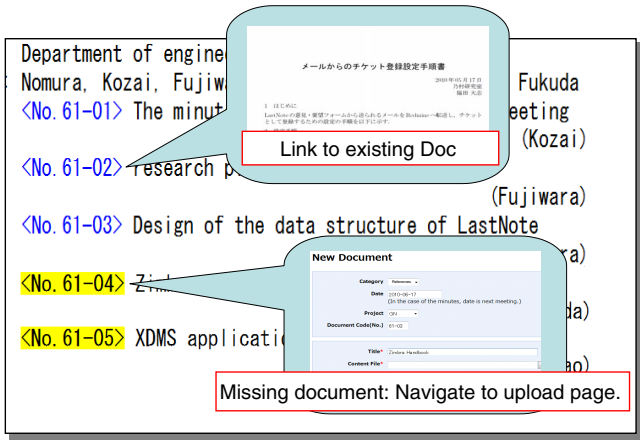

Figure 7. Link to in-house document management system.

enforcing the switch of mailer will be costly. On the contrary, our method only costs the extra one-click to the archive page.

2) Link to Google maps: Mashing-up with Google maps is the same story as the case of Redmine. Main difference is the regular expression in the decorator. We feed a regular expression which catches Japanese postal addresses.

3) Mash-up with document management system: As an another example, we show how to mash-up with our in-house document management system. Linking to strings which represent document ID is similar action to the case of Redmine. In this case, the decorator goes deep into the the document management system and acquire information about a target document for adjusting the link accordingly.

In our document management system, each document has the attribute such as title, owner, project and ID. ID has the form of *xx-yy*. *xx* and *yy* are sequence of numbers. We sometimes want to mention about the documents in our meeting minutes. In this case, we use the notation like $< No.xx - yy >$ enclosing ID by <>.

In Figure 7, we have two different kinds of links on the $< No.xx - yy >$ strings. The blue link shows a direct linkage to a stored document as usual. On the contrary, the yellow link shows a missing document in the system. By clicking the yellow link, the user will be taken to Upload-page. In the Upload-page, system will fill in the blanks of the web-form by using the information taken from the original mail article, and ask the user to upload the missing document file.

In this example, the decorator cuts the string which represents the document ID, and connect to the document management system to look up the document by the ID. Using the result of the search, it adds a proper link.

C. Using Zimbra Collaboration Suite

As an example for exporting API of our system, we show a story that the Zimbra Collaboration Suite gets a member list of a ML on our system. Zimbra has a framework for extending itself, namely, Zimlet. Zimlet is a small snippet written in JavaScript. We wrote a Zimlet to fetch a member list of a ML using the API of our ML system. Figure 8 shows the screen-shot of the zimlet.

Firstly, our zimlet send a request in the form of JSON in which target ML name is enclosed. Next, our ML system check permission of the request and validity of the ML name. If the request is valid, the member list of the ML is returned in JSON format. Finally, our zimlet shows the result adding a link to the ML-archive page.

As shown by the example, our system is capable of exporting its functions and information via Mash-up API. Traditional ML systems do not have such concept.

V. Conclusions

In this paper, we propose a new ML management system, which helps the new small-group MLs mashing-up with their web systems. Also we describe the design and the implementation of the system. Our system encourages users for migration to web services. It provides some methods for connecting mail articles with outer web services. As a future work, we are planning to perform some evaluation of our system.

Acknowledgment

This study was done in cooperation with NTT Service Integration Laboratories, NTT Corporation, and also was partially supported by the Ministry of Education, Science, Sports and Culture, Grant-in-Aid for Young Scientists (B), 21700139, 2010.

References

[1] The FreeBSD Forums Admin Team, "Official FreeBSD Forums," FreeBSD-announce ML, http://docs.FreeBSD.org/cgi/mid.cgi?20081116160403.GC79046 , Nov. 2008.

[2] "Mandriva Linux developer ML (cooker ML)," http://lists.mandriva.com/cooker/

[3] Vincent Danen, "Sad metrics on community decline," linsec.ca blog, http://linsec.ca/blog/2008/12/05/sad-metrics-on-community-decline/ , Dec. 2008.

[4] Microsoft Corp., "Microsoft Outlook," office.microsoft.com/en-us/outlook/

[5] Google Inc.,"Gmail," http://mail.google.com/

[6] Yahoo! Inc.,"Yahoo! Mail," http://mail.yahoo.co.jp/

[7] GWietse Venema, "Postfix," http://www.postfix.org/

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

[9] Redmine, http://www.redmine.org/

[10] Google Inc.,"Google Maps," http://maps.google.com/

[11] Zimbra Inc., "Zimbra Collaboration Suite," http://www.zimbra.com/