University of Innsbruck
Semantic Technology Institute

Enabling Semantic Email Addressing

Bachelor Thesis

Philippe Rödlach
Emmat 370r
A-6105 Leutasch
Matrikelnr.: 0415810

Supervised by Dr. Elena Simperl
and co-supervised by Uwe Keller

Innsbruck, June 30, 2008

Abstract

This bachelor thesis illustrates the usage of semantic email addressing. To show this concept and define it's advantages it will be compared with the concept of mailing lists and shows how efficient it can work in highly dynamic groups. To explain these concepts charts were generated.

To compare semantic addressing with the nowadays common email addressing, an email client was developed that can handle both concepts. Two solutions to send semantically addressed emails were worked out. On the one hand a client sided solution has been elaborated where the emails directly sent by the user. On the other hand a server sided solution has been designed where the server sends the emails to the recipients.

To make semantic addressing possible, attributes which define users who want to use semantic addressing have to be collected. Therefore two ways to enrol the user's data into a unit which collects these attributes will be explained. One web interface has been designed where the user can enrol himself and one web-crawler has been coded who automatically enrols users.

The most important use cases which ease the work concerning sending, receiving, forwarding and replying emails, has been explained. Furthermore solutions to use them for an email which has been semantically addressed were worked out. To show the solutions to make these use cases possible a

semantic email client was implemented.

The email client has been tested in a customer relationship business setting between a hotel and its guests and the advantages and disadvantages were explained. A possible workflow using on the one hand mailing lists and on the other hand semantic email addressing has been elaborated.

Acknowledgements

To express my gratitude I want to thank all people who supported me doing this bachelor thesis.

I want to thank the department of computer science of the Leopold-Franzens-University and Semantic Technology Institute (STI), especially Uwe Keller who supervised me in my project.

Contents

Chapter 1

Introduction

Nowadays group communication becomes more and more important, either for private issues or for business matter. It is used to exchange cogitations, feelings, knowledge and information. Group communication is the absorption, the exchange and the transmission of information between at least two persons. Communication in groups happens every day, if friends speak about the new soccer scores or about the skiing excursion of the last weekend for private reasons, or department chiefs hold a meeting to exchange the newest happenings in the department and the results they achieved.
As seen in the examples two points must be fulfilled to establish a group communication.

- a communication channel must be defined
- the recipients must be identificated

To define a communication channel in which the communication should take part is not as easy as it seems. Every recipient has to be in the ability to establish the communication channel. For example if a group communication should be established where a deaf-mute person should be involved the communication channel can not be the telephone but rather an email for example, a letter or a meeting where the communication takes part with sign

language.
The more complicated part is the identification of the recipients which should be involved in the group communication. In this process recipients have to be defined who can exchange information concerning a specific topic. The recipients have to have knowledge to be part of the communication. For example in a group communication about computer networks a car mechanic is maybe not a recipient because he can not exchange useful information with the other network professionals. So a car mechanic would be in this topic a person which would be outside the group and not be mentioned as long the communication in the group would change the topic in which the mechanic is familiar with.

One of the nowadays most common communication channels is email. Here people have the ability to exchange information to one specific person or to a whole group of people. The communication to one person is very simple, it is nearly like a phone call to a person. The person who wants to establish the communication via the communication channel has to identify the receiving person. The identification via the communication channel *telephone* is done by entering the phone number, with *email* the identification process is done by entering the email address.
As mentioned above, recipients have to have knowledge of the concerning topic to be part of the communication and the ability to take part at the communication. With one-to-one email communication, the sending person has the knowledge, that the receiving person has on the one side the ability to work with emails, otherwise he would not have an email address. On the other side is the person which can take something up with the information received, because to access the email address of the recipient, the sending part stands in a specific relation to the receiver (commercial relation: spokesperson gives his business card).
But what is with group communication via email? If it is a group where the

sending part knows everybody (knows all the email addresses) to communicate with, it is not a big problem to identify the persons and be sure that they have the knowledge to be part of the communication. But if the members in the group change very often, it can happen that the sending person does not establish the communication channel to everybody. So information can be lost or not properly distributed to all relevant receipients.
To simplify group communication, the concept of mailing lists was invented. With this concept the group size does not play a role, the sending part has not to know all the email addresses of the recipients, and fluctuation in the group is not a problem. But also with mailing lists the communication is not that flexible as it could be, if a user wants to be part in a communication process, he has to know of the existence of the mailing list and has to sign in it.

Is there an easier way to perform group communication via email?

In this thesis we take a look at and implement a new concept for addressing in email based group communication called *semantic email addressing* that has been proposed in [2] and [3].

1.1 Problem Statement

Nowadays, almost everyone is using email as a tool for communication and collaboration every day. Especially in a business setting, email became extremely important and a major means for information exchange. However, for efficient communication across highly dynamic groups of people (e.g. a team working over long time on a product and respective updates the working group) current tools are non-optimal. The primary supporting means for managing such groups are mailing lists, which are manually administrated. In particular, the management of the groups and respective lists are overly complex. Identification of people that have to be addressed is time-consuming and error-prone and not integrated with any other applications.

In order to achieve the latter, a knowledge-based approach is suitable, e.g. the creation of knowledge layer that represents, e.g. people, their current tasks, responsibilities, and roles. Such an explicit representation can be done by means of an ontology [4]. A respective reasoner then allows applications to exploit and access (explicit and implicit) knowledge from this knowledge base.

Recently, semantic email addressing has been proposed as a knowledge-based solution to the outlined problem. Essentially, semantic email addresses are logical (declarative) descriptions of recipients rather than static lists of strings. Therefore, they allow sending (with the same description) emails to highly dynamic groups of people and at the same time ensuring, that the "right" people (at this specific moment in time) are addressed. Therefore semantic emails addresses are stable, whereas the group might change very frequently. Moreover, they can be stored and shared with others, while keeping the same desirable property.

1.2 Contribution of the Project

This project illustrates the concept of semantic email addressing. To explain this concept comparisons between semantic email addressing and the usage of mailing lists are drawn in section 2.1. Also some problems and solutions to solve these problems are shown in section 2.2. The most important use cases which also semantic email addressing compared with nowadays email addressing should provide are explained in the chapter 3. In the last chapter 4 the results of the testing process in a real life environment are presented.

Chapter 2

Solution

In the first part of this chapter semantic email addressing is compared with the concept of mailing lists. The advantages and disadvantages are explained and an example is given, which should demonstrate the operation breakdown. Afterwards it will be spoken about problems which have to be solved by using semantic email addressing and possible solutions will be given. To demonstrate the problems and explain the solutions semantic email addressing will be compared with the nowadays common way to address emails.

2.1 Mailing lists vs. semantic email addressing

Nowadays mailing lists are common to discuss topics in which all enroled people are interested. They are used in business, in research and other settings where group communication takes part. If someone is enroled in many mailing lists, it can be very exhausting to administrate his appearance in the lists. The administrator of the list can limit your rights to "only allowed reading", "only allowed writing" or both rights. So wouldn't it be more comfortable to have to do none of these steps? That is one reason why semantic

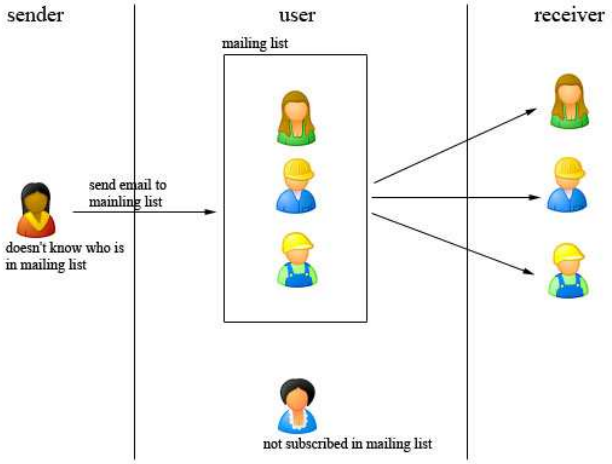

Figure 2.1: Concept of mailing lists

email addressing could become very popular.

2.1.1 Mailing lists

A mailing list offers a closed group of people the possibility to exchange messages concerning a topic in which all participants are interested. For example a mailing list where people discuss the newest Windows version, talking about bugs and approaches to solve these bugs. Figure 2.1 shows the concept of mailing lists.

The sending party, with the exception of the administrator, doesn't know who is subscribed in the mailing list. He gives his advices or asks questions which will be discussed in the mailing list. He sends his message to an allocator who sends each subscribed person the message. Figure 2.1 shows that only subscribed users receive the email from the sending party.
When users are subscribed in many mailing lists it can be very exhausting to maintain their appearance in the lists. Another disadvantage is that users who are not interested in the topic after some time they have to unsubscribe

to become deleted from the list. The next problem can arise, when users change their email address where they want to receive the messages. The first step would be to unsubscribe the old email address from the list and subscribe with the new address or he has to update his profile in all the mailing lists. Often it could be the case that a user wants to send a message to a particular subset of user in mailing lists, but this isn't possible because the message will be send to an allocator who sends the message to all subscribed users. This is another reason to think about the concept of semantic email.

2.1.2 Semantic email addressing

Everybody knows what the words *email* and *semantic* mean when they don't stay in relation. Email is a message in the form of an electronically letter. Semantic refers to the concept of meaning and discusses the question how we can deduce simple terms from complex terms to understand the meaning of the complex construction. But what do the two words stand for together?

Semantic email can be understood as a mailing list with the difference that the user has not to sign in and sign out of this list respectively. With semantic email addresses the user can avoid the maintenance of his attendance at mailing lists (described at point 2.1.1) and his email addresses.

Everybody has been in the position that he had to search for an email address of a person or received an email with the message that the inserted email address does not exist anymore. The goal of semantic email is, to write emails to a person without the knowledge of his email address. This will be achieved by inserting relevant attributes like name, address or to groups which have the same relevant attributes like the company name they are working for (to send all members of the company the email) and the company department (to send all members of the section e.g. marketing the email). Figure 2.2 shows this concept where only users with the relevant attribute, which was

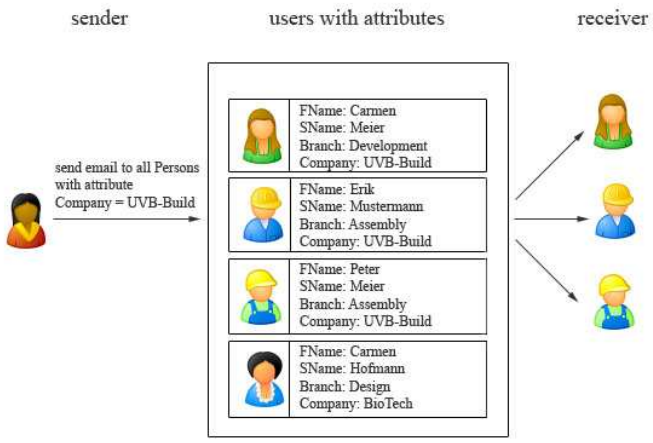

Figure 2.2: Concept of semantic email addressing

decided from the sending party, receive the email.

2.1.3 Example

Both concepts have their advantages and disadvantages. It is easier to manage a mailing list, then to collect the user's attributes in a unit, which will decide which user receives the message. In section 2.3 possible ways to collect the attributes will be explained. Semantic email is a dynamic concept, where it is as easy to send a message to one user as to send the message to a group of users only by defining the effective attribute. The following example will show the differences of both concepts.

There is a user who works for company c1 in the design branch. To receive important emails from the company he has to subscribe to the main mailing list of the company and to receive important emails from the branch he has to subscribe to the mailing list of the certain branch. After some month he had to leave the design branch and changed to the management branch.

- **Solution 1:** *Concept of mailing lists*

At first he has to subscribe to the company main mailing list. After this he also has to subscribe to the mailing list of the design branch. When he leaves his actual branch, he has to unsubscribe from the design mailing list and has to subscribe to his new branch management mailing list.

- **Solution 2:** *Concept of semantic email addressing*

  To receive the important emails of the company, the sending party has to write all users which have the company c1 as the company attribute the email. The user itself has less to do, he has only to change his personal data. To receive the important emails of the branch, the sending party has to write all users which have the branch design as the branch attribute the email. When the user changes the branch now, the branch attribute design changes to management also. Now you can see the dynamic concept of semantic email. The only exercise for the user is to define his attributes.

2.2 Problems of semantic email addressing

Semantic email addressing should technically be layered on top of standard email infrastructure, since this is a reliable and well-understood technology and software.
Due to the usage of semantic email addresses some big problems can appear, which have to be solved. In the following paragraph these problems will be explained and some possible solutions for them will be discussed.

2.2.1 Transmission of semantic addressed emails

If one considers the transmission of semantically addressed emails the following question arises.

How can the transmission of semantically addressed emails can take place and who should send the emails, the email client or an own server?

Therefore two different solutions have been worked out. One solution is based client sided, where an email client directly sends the semantic addressed emails. The other one is based server sided, where the server is responsible for the transmission of to the right receivers.

2.2.2 Client-sided solution

A vital point for the solution was that nowadays popular email clients can be easily extended and that the client has the ability on the one hand to handle the semantically addressed emails but also on the other hand to still support the *normal* addressed emails. To test the client-sided solution an email client has been developed, who can handle the semantically addressed emails as well as "normal" addressed emails. One possible design of a semantic email client shows figure 2.3.

The semantic email client looks like popular email clients, with the exception the semantic send button. To send normally addressed emails the user has the possibility to insert the email address in the *To:* field. If the user wants to address another user semantically he should input the information which identifies the receiver. It would be very comfortable to insert the semantic address into the *To:* field too but it is not possible for a program to find out, if the inserted text is the first name, the second name or any other identifier of the user without any other information of the inserted text. The next problem is that a user inserts attributes which could identify the receiving

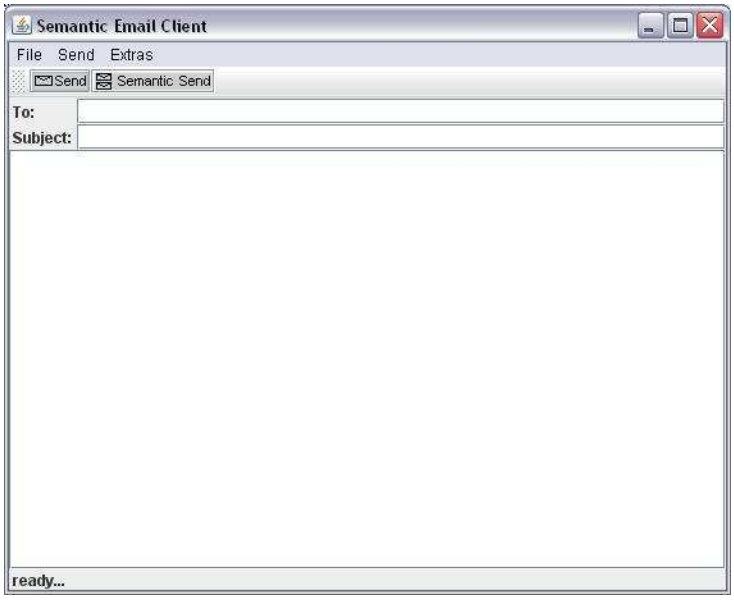

Figure 2.3: Semantic email client

party, but the attributes were not defined from the user.

A possibility to find out the attributes of the inserted text is to use a search filter which on the one hand defines the attributes for which can be searched, and on the other hand defines what the input string stands for. Figure 2.4 shows a possible search filter.

Figure 2.4 describes the concept of the search filter. The user has the possibility to search for the email address which concerns to his inputted text. In the example all people who include the attribute *Company = UVB-Build* will receive the email of the user.

After the definition of the attributes the email address/addresses of the receiving party has/have to be found out. This can be done with a database in which all users are stored with the specific attributes. When a user with the defined attributes has been found in the database the email address is

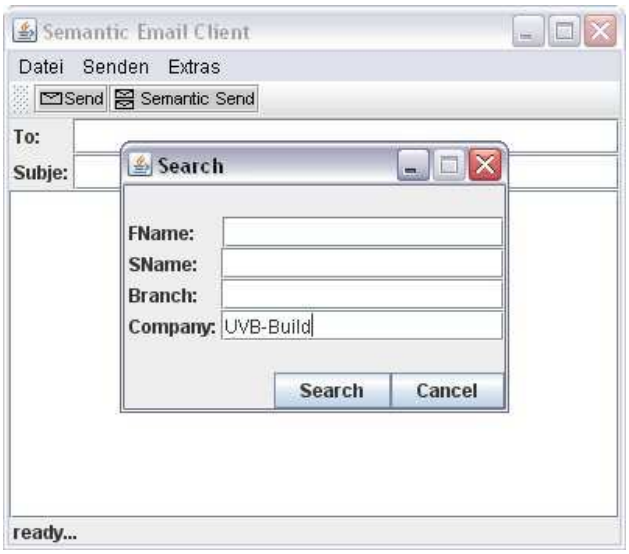

Figure 2.4: Search filter of the email client

listed in the *To:* field and the user can send the email. Figure 2.5 shows the found email addresses in the database.

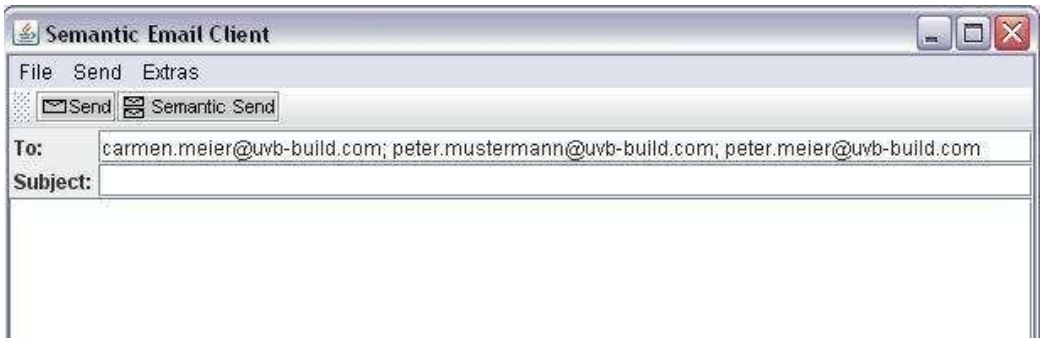

Figure 2.5: Email addresses inserted from database

Of course this solution is very helpful to send emails to people whose email address is not known by the sending user. But in case of security it would be safer not to send back the receiver's email addresses to the client to avoid that the sender gains the real addresses and to avoid receiving a lot

of spam. To achieve a safer solution a second possibility was elaborated (see point 2.2.3).

2.2.3 Server-sided solution

Unlike to the client-sided solution the server-sided solution receives a possible semantic address email from the email client, which includes a string from the the receivers attributes and the message of the email. Figure 2.6 shows on the left side the semantic email client and on the right side the semantic email server.

Figure 2.6: Server received string

The server receives the string, suspends it and searches for the email address of the receiving party with the gotten information in the database. When an email address was found, the server sends the email to this address. Figure 2.7 shows that the server sent an email to *lospippos@gmail.com* successfully.

Figure 2.7: Server successfully sent email

If the email was not sent successfully the server informs the client.

In all this possible solutions, the email address of a user has to be found out with his describing attributes. As shown in figure 2.7 the email address *lospippos@gmail.com* was found with the inputted attributes ***fname:****Philippe,* ***sname:****Roedlach*. So the question came up how to gain the email address and the user's attributes.

2.2.4 SMTP-Header extension

With the solution described at point 2.2.2 emails will be transmitted with the nowadays email-transmission routine. A more comfortable way would be to insert a semantic email address in the way to use the input filter. So the semantic email address would be dynamically generated. Afterwards the email will be send to the user without trying to search for his "normal" email address to send the email with.

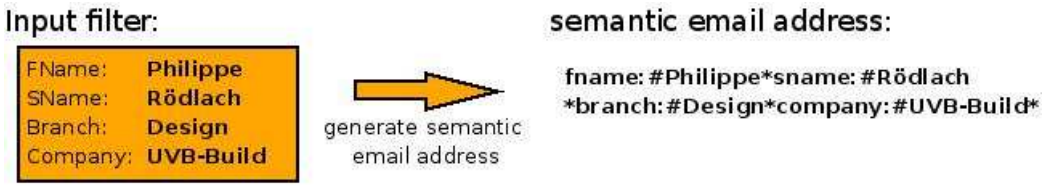

Figure 2.8: Generation of a semantic email address

To achieve such a solution, the SMTP protocol would have to be extended. In the following paragraph the SMTP protocol and the way it works will be described shortly.

SMTP is the abbreviation for "Simple Mail Transport Protocol" and is used to transmit emails. Figure 2.9 shows the model and how it is working.

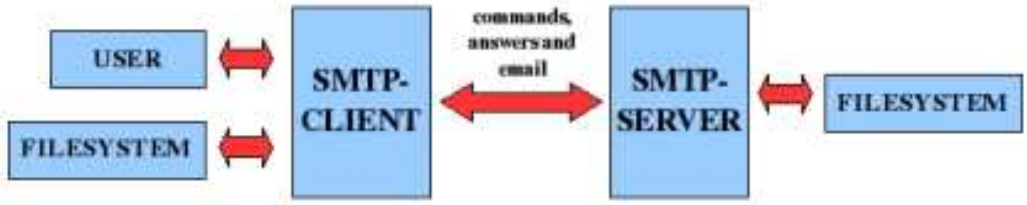

Figure 2.9: SMTP-Model

The SMTP-Client (the email program of the user who wants to send an email) sends a request to the SMTP-Server. The server sets up a two way transmission channel to the client. The SMTP-Server must not be the SMTP-Server of the email receiving party, it could also be a intermediary server. In this case it is used as a negotiator.

Afterwards the SMTP-Client generates the SMTP commands, which were send to the SMTP-Server. As a response of the client's commands, the

Figure 2.10: Telnet connection - The lines starting with client had to be entered in a shell

server sends back status signals (e.g. +OK, if the command was accepted). For example the client sends the command MAIL to the server, as soon as the transmission channel was created. As a response the client receives the status signal +OK from the server if it is in the position to handle the request (in this example to forward/send the email). After this step the addressee of the email will be send to the SMTP-Server with the command RCPT. If the server can't deliver the email, the client receives according to the circumstances, an error message and can try to send another address of the receiver.

If this step was successfully completed, the data (header an message text) can be sent with the accordant SMTP command. The end of the email will be shown over a sequence defined from the protocol.

After this the SMPT-Server handles the email, if it could send it to the receiver, the server sends the status signal +OK back to the client.

The concept of SMTP is illustrated by a concrete example in Fig-2.10. This example shows, that to send the email with a semantic email address it would be necessary to implement a field called SEMANTIC RCPT TO.

To test the transmission of a semantic addressed email, a server-sided solution has been developed. In point 2.2.3 this solution had been described and some figures of a possible design had been shown and mentioned.

2.3 Enrolment of the user's attributes in the database

As in the chapter above described, the users attributes were stored in a database. There are two major problems: How should the data-attributes concerning a user should be accessed and who should store the data of the users? One possibility is that the user himself stores the vital attributes in the database with a web interface in a simple form.

2.3.1 Enrolment by the user

A solution is to implement a php form which is responsible for the storage of the data in the database. See Figure 2.11. An important difference to mailing lists is, that the users has to enrol once into a single database storing his personal profile and not multiple times when he submits to multiple mailing lists located at different places on the web.
In the next section a more intuitive way to enrol users is described.

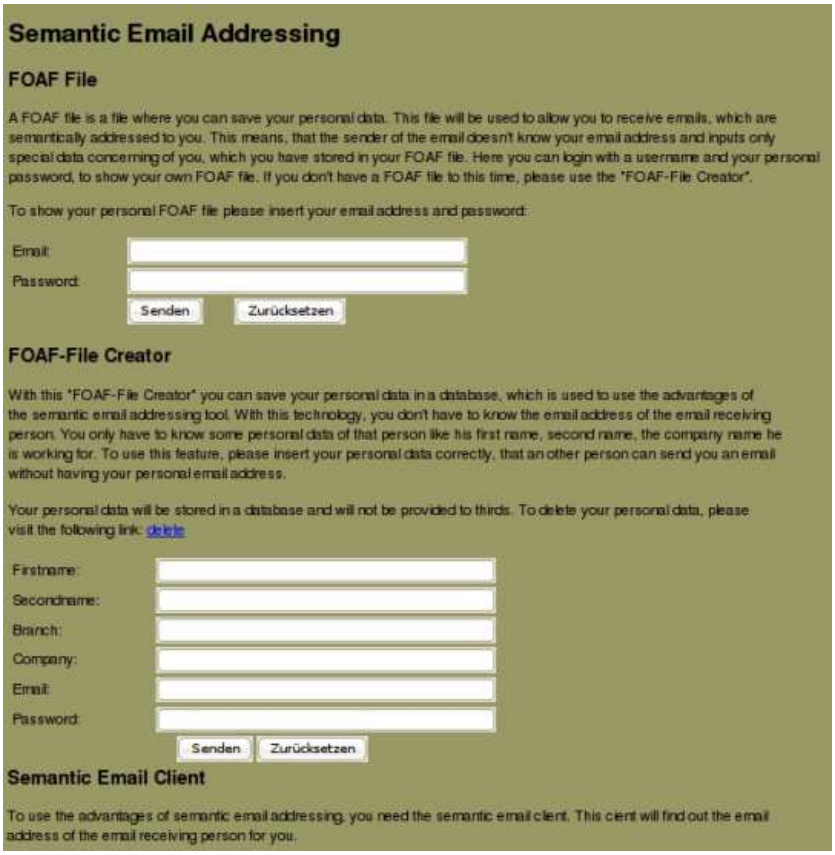

Figure 2.11: Php form to store user's data in the database

2.3.2 Automatic enrolment

To enrol the users attributes automatically in the database, any source of information concerning the user has to be found. A possible way to find such information is to search for FOAF files in the internet.

FOAF [1] stands for Friend of a Friend and is a project to generate a social network, which a machine can read. A FOAF file is a document where a person can store his information like name, email address, homepage and friends. The file can link to other FOAF files and so a network arises. That

means the machine can be programmed that it reads such files and finds out vital information of people. Through the links to other files a machine can crawl over all the files which are linked and store the information if it is finding a person in the database. A FOAF file is written in RDF which bases on XML and can so be understandable for machines.

RDF is the abbreviation for "Resource Description Framework" and is used to generate metadata in the internet. The RDF-Model consists of three object types: resources, property-elements and objects. One resource, property and object build a RDF-Triple. With this combination a conclusion of an object within a domain will be made which is also called a statement. Connecting this statements a network structure is set up. The following code part shows a FOAF-File written in RDF (see listing 2.1).

Listing 2.1: FOAF-File

```
<?xml version="1.0" encoding="iso-8859-1"?>
<rdf:RDF
      xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
      xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
      xmlns:foaf="http://xmlns.com/foaf/0.1/"
      xmlns:admin="http://webns.net/mvcb/">

  <foaf:PersonalProfileDocument rdf:about="">
      <foaf:maker rdf:resource="#me"/>
    <foaf:primaryTopic rdf:resource="#me"/>
  </foaf:PersonalProfileDocument>

  <!-- Informations to a person called Philippe Roedlach, ...
       -->
  <foaf:Person rdf:ID="me">

    <!-- foaf:name defines the full name -->
```

```
    <foaf:name>Philippe Roedlach</foaf:name>

    <!-- foaf:givenname defines the first name-->
    <foaf:givenname>Philippe</foaf:givenname>

    <!-- foaf:family_name defines the family name-->
    <foaf:family_name>Roedlach</foaf:family_name>

    <!-- foaf:Organization defines the company the person is
        working for -->
    <foaf:Organization>Uni Innsbruck</foaf:Organization>

    <!-- foaf:member defines a membership or the branch where
         the person is working -->
    <foaf:member>Student</foaf:member>

    <!-- foaf:nick defines the nickname -->
    <foaf:nick>pippo</foaf:nick>

    <!-- mbox defines an email address of the person -->
    <foaf:mbox rdf:resource="edv@klosterbraeu.com"/>

    <!-- mbox rdf:resource defines that the email address
      will not be shown on the web page -->
    <foaf:mbox rdf:resource="philippe.roedlach@aon.at"/>

    <!-- here mbox defines that the email address will be
        shown -->
    <foaf:mbox>lospippos@gmail.com</foaf:mbox>
    <foaf:mbox>csaf3266@uibk.ac.at</foaf:mbox>

    <!-- foaf:homepage defines the url of the web
        presentation -->
    <foaf:homepage rdf:resource="http://lospippos.lo.funpic.
        de"/>
```

```
    <!-- foaf:knows defines that a person knows another
        person -->
    <foaf:knows>
      <!-- the person which is a friend (ID="friend1") who is
51      known from the main person -->
      <foaf:Person rdf:ID="friend1">
        <foaf:name>Max Mustermann</foaf:name>
        <foaf:givenname>Max</foaf:givenname>
        <foaf:family_name>Mustermann</foaf:family_name>
56      <foaf:member>Student</foaf:member>
        <foaf:Organization>Uni Innsbruck</foaf:Organization>
        <foaf:mbox>max.mustermann@uibk.ac.at</foaf:mbox>
      </foaf:Person>
    </foaf:knows>
61  <foaf:knows>
      <foaf:Person>
        <!-- seeAlso defines, that in the given url the
            attributes of the person are stored and have not
            to be input by the main person (here Philippe
            Roedlach)-->
        <rdfs:seeAlso rdf:resource= "http://lospippos.lo.
            funpic.de/homepage/themen/schatz.rdf"/>
      </foaf:Person>
66  </foaf:knows>
    <foaf:knows>
      <foaf:Person rdf:ID="friend2">
        <foaf:name>Petra Testhausen</foaf:name>
        <foaf:givenname>Petra</foaf:givenname>
71      <foaf:family_name>Testhausen</foaf:family_name>
        <foaf:member>Student</foaf:member>
        <foaf:Organization>Uni Innsbruck</foaf:Organization>
        <foaf:homepage rdf:resource="xxxxxxx"/>
      </foaf:Person>
76  </foaf:knows>
    <foaf:knows>
      <foaf:Person rdf:ID="friend3">
        <foaf:name>Wolgang Brenner</foaf:name>
```

```
        <foaf:givenname>Wolfgang</foaf:givenname>
        <foaf:family_name>Brenner</foaf:family_name>
        <foaf:member>Student</foaf:member>
        <foaf:Organization>Uni Innsbruck</foaf:Organization>
        <foaf:mbox rdf:resource="wolfgang.brenner@uibk.ac.at"
            />
        <foaf:homepage rdf:resource="xxxxxx"/>
      </foaf:Person>
    </foaf:knows>
  </foaf:Person>
</rdf:RDF>
```

The best way to demonstrate a FOAF file is to draw a RDF-Graph. The following FOAF file in listing 2.2 will briefly demonstrate a RDF-Graph shown in figure 2.12.

Listing 2.2: FOAF-File

```
<?xml version="1.0" encoding="iso-8859-1"?>
<rdf:RDF
      xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
      xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
      xmlns:foaf="http://xmlns.com/foaf/0.1/"
      xmlns:admin="http://webns.net/mvcb/">
  <foaf:Person rdf:ID="me">
    <foaf:name>Philippe Roedlach</foaf:name>
    <foaf:givenname>Philippe</foaf:givenname>
    <foaf:family_name>Roedlach</foaf:family_name>
    <foaf:mbox rdf:resource="philippe.roedlach@aon.at"/>
    <foaf:mbox>csaf3266@uibk.ac.at</foaf:mbox>
    <foaf:knows>
      <foaf:Person rdf:ID="friend1">
        <foaf:name>Max Mustermann</foaf:name>
        <foaf:givenname>Max</foaf:givenname>
        <foaf:family_name>Mustermann</foaf:family_name>
        <foaf:mbox>max.mustermann@uibk.ac.at</foaf:mbox>
      </foaf:Person>
    </foaf:knows>
```

```
110     <foaf:knows>
          <foaf:Person>
            <rdfs:seeAlso rdf:resource="http://another_url/peter.
                rdf"/>
          </foaf:Person>
        </foaf:knows>
115   </foaf:Person>
</rdf:RDF>
```

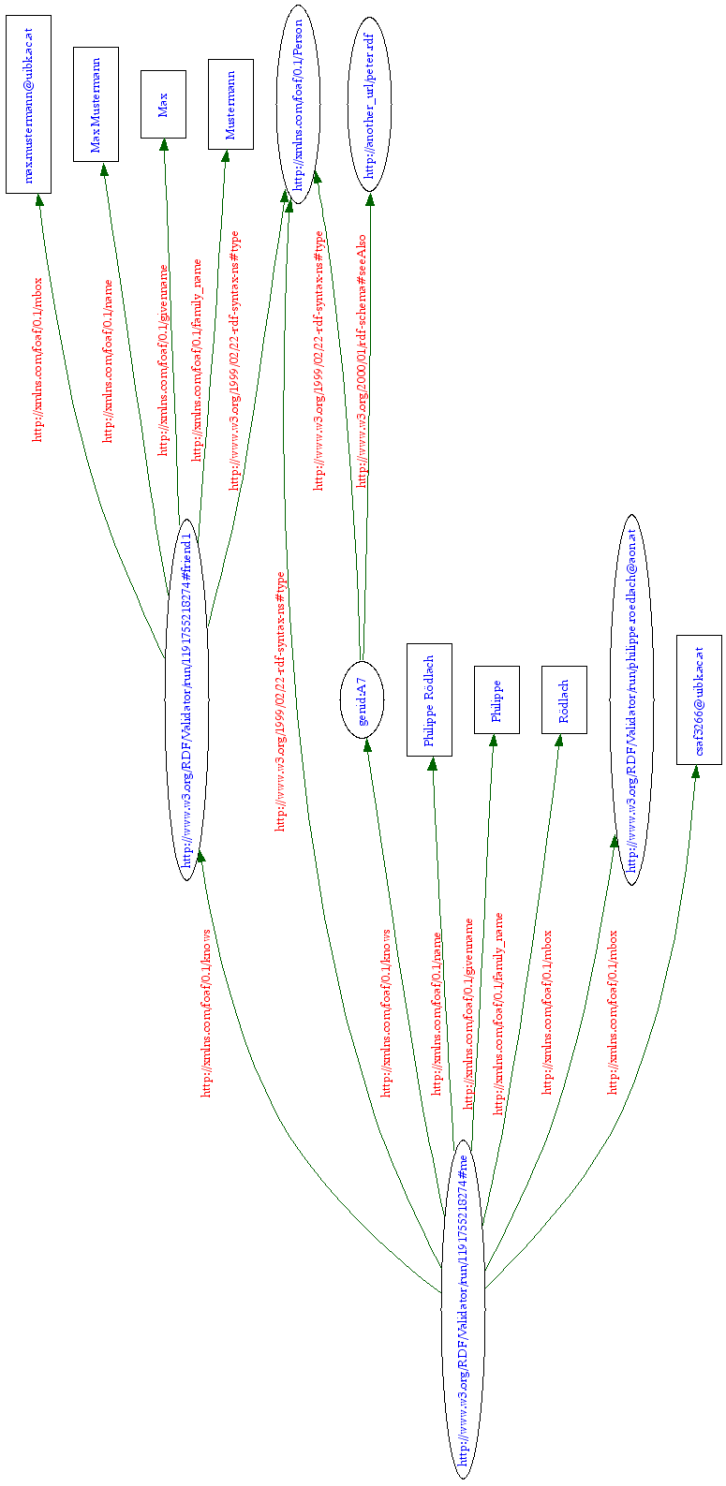

Figure 2.12: RDF-Graph

As mentioned above a statement is a triple of a *resource*, a *property* and an *object*. In a RDF-Graph a resource is defined as an ellipse for example *http://www.w3.org/RDF/Validator/run/1917552182744#me*. This stands for the namespace of RDF and the extension of the resource id *#me*. The properties of the resource stand on the arrows and are all the sub-elements of the resource. The properties of the resource *#me* are:

- *http://xmlns.com/foaf/0.1/knows*
- *http://xmlns.com/foaf/0.1/name*
- *http://xmlns.com/foaf/0.1/givenname*
- *http://xmlns.com/foaf/0.1/family_name*
- *http://xmlns.com/foaf/0.1/mbox*
- *http://xmlns.com/foaf/0.1/type*

For example the string *http://xmlns.com/foaf/0.1/givenname* stands for the namespace of the FOAF tags in the XML document and the name of the property, which is *givenname*. The last part of the RDF-Triple is the object. In this example the object of the resource *#me* with the property *givenname* is *Philippe*. Figure 2.13 shows this example with a RDF-Graph.

Figure 2.13: Example of a statement with a RDF-Graph

So one statement in the graph is the resource with the ID #me and the roperty givenname has the object Philippe.

With the property type the type of the resource is defined. In figure 2.13 the type of the resource *#me* is *Person*. So the resource *#me* is as a person and can not be for example a cat.
With the property *knows* figure 2.13 shows, that the person knows another person and that two more persons are linked. There are two ways of linking a person. The first possibility is to store the personal data of the to link person in the own RDF-File shown with the resource *#friend1*, which is also a person, where all his attributes are shown in figure 2.13. In figure 2.12 *Max Mustermann* is linked like this way.

Another possibility is to use the property *seeAlso*. With this property the graph defines that the attributes of the to linking person are stored on another website. So the user, has not to save the attributes of the to linking person in his RDF-File. In figure 2.12 the resource *gerid:A7* defines a person which is linked with the *seeAlso* attribute. The person's attributes are stored at the url *http://another_url/peter.rdf*.

If every person in the world who wants to use semantic email addressation has such a FOAF file, it could be possible to find all these files and insert them into a database. So nobody has to know all the email addresses to write an email to a specific user.
A possible way to find FOAF file is to write a web crawler which finds all web links in a page and searches for RDF files on all pages for whom it found a weblink. If the crawler finds a RDF file, it looks if a person stored his attributes on the file and stores them in the database if the crawler was successful. Otherwise it searches for the next file. When it finished the search algorithm on one page it tries to find all RDF files on the sites to which the found links link. Here it also searches for links and RDF files and does this recursively. Figure 2.14 shows a web crawler which searches for links and RDF files.

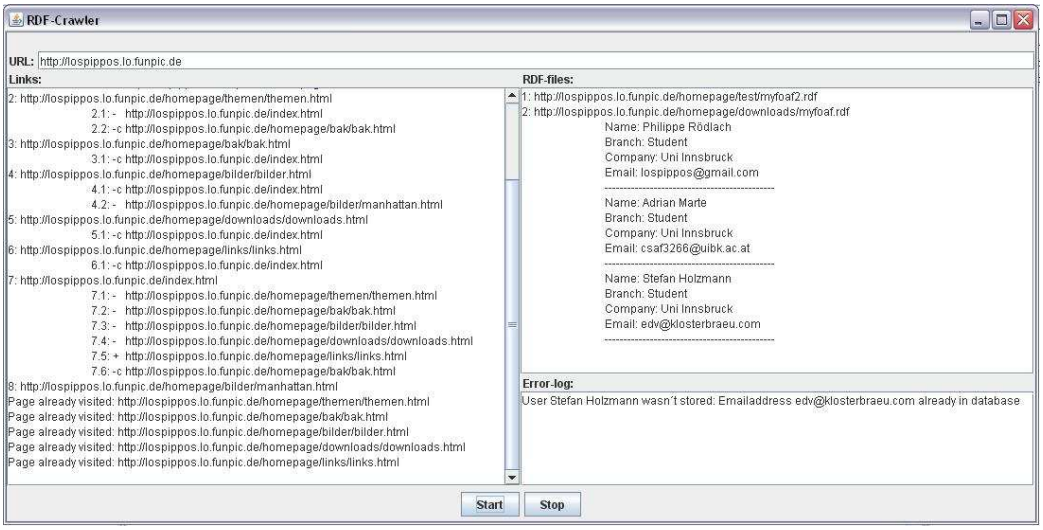

Figure 2.14: Web-crawler with search results

The crawler shows the found links in the *Links* textfield and the found RDF files in the *RDF-files* textfield. In the RDF file *myfoaf2.rdf* no user was found. This can be if really no user was found or the user has not stored all the vital attributes in his file. In the file *myfoaf.rdf* three persons were found an stored in the database. The *Error-log* textfield shows errors which appeared during the search process, or informations that a user was not stored because the email address is already stored in the database.

Chapter 3

Use case scenarios

In the following chapter the most basic use case scenarios like *WRITING*, *SENDING*, *RECEIVING*, *REPLYING* and *FORWARDING* of emails which where "normally" addressed are explained and possible solutions to solve these scenarios for emails which where semantically addressed are described. Figure 3.1 shows an email which has been received by the semantic email client. The buttons shown in the header panel of the received email are used to solve some of the use cases which are described in the following sections.

3.1 Writing semantic emails

To write an email which will be addressed semantically should not be more difficult as to write a "normal" addressed email. The user should not notice a difference when he writes on the one hand a "normal" addressed emails and on the other hand a semantically addressed emails.
With the email client shown in figure 2.3, the user can write semantic and "normal" emails without any difference. When he writes the email he has to decide if he wants to address the email semantically or "normal". So there is no difference between both concept concerning the writing of emails.

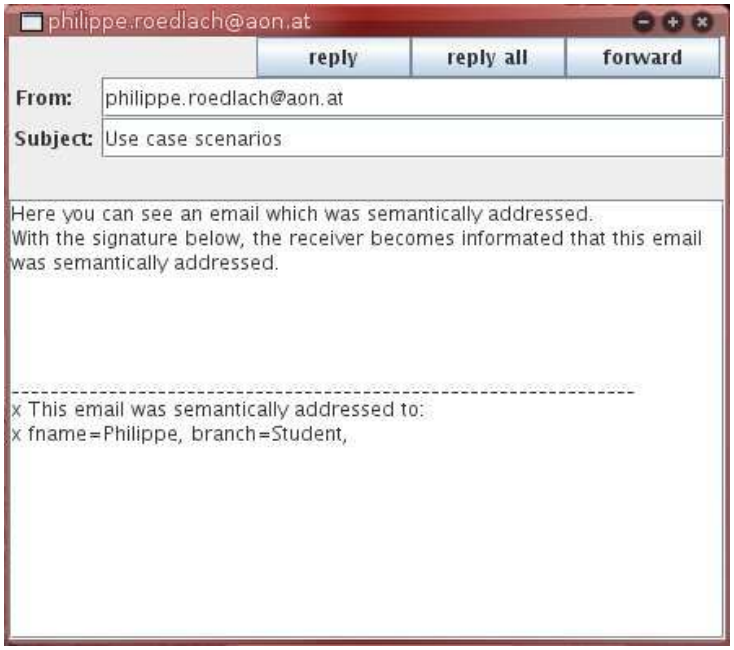

Figure 3.1: Received email - semantically addressed

3.2 Sending semantic emails

This section describes two ways of sending semantically addressed emails and will compare semantic addressing with the nowadays common addressing of emails.

Today the user writes his email message in a window. When he finished writing, he has to insert the email addresses of the email recipients. The next step is, that the user has to send the email. This happens using in the most popular email clients the *Send* button.

To make the transmission of semantically addressed emails as easy as the nowadays email addressing. There should not be a big difference between both email transmission routines. After the user wrote the email message, as mentioned above he has to insert the email addresses of the recipients, but with semantic email addressing the address is dynamic which the user has to insert to make the transmission possible. Two make the transmission

possible two solutions were worked out, which should help the user to insert the semantic email.

3.2.1 Sending with the client-sided solution

Therefore the email client in figure 2.3 has a *Semantic Send* button, which the user has to press if he wants to send his email with semantic addressing. After he pressed the button the search filter shown in figure 2.4 appears. With this filter the client can define the attributes which the recipients have to satisfy to receive the email. This was the first solution of the sending process, which was client-sided (see also subsection 2.2.2)

3.2.2 Sending with the server-sided solution

Another solution which would be more intuitive is the server-sided solution described at point 2.1.2. The advantage over the client-sided solution is that the user has only one *Send* button and one more addressing field *Semantic To:* shown in figure 2.7. With this solution the user can send emails which are addressed "normal" in the *To:* textfield and at the same time semantically addressed emails in the *Semantic To:* textfield. To insert the semantic email addresses the user has to *click* into the *Semantic To:* textfield. After the *click* the same search filter as in the client-sided solution appears where the user can define the attributes which the recipients identify.

3.3 Receiving semantically addressed emails

Receiving emails is nowadays not a big problem. The user starts the receiving routine when he uses the receiving button of his email client. After the usage of the *receive* button the in-box server of the user becomes synchronised with the local in-box of the client and the emails were listed there.
To make semantic addressing suitable for daily use the email client has to

receive semantically addressed emails as well as “normal” addressed emails. With semantic addressing one major problem arises. The user receives an email an wants to reply it, but if the email was addressed semantically the recipient has to know which attributes the sender defined to use the same attributes that all receivers will get the reply of the email. In the section concerning the *reply* of emails (see section 3.4) this problem will be solved.

3.4 Replying to semantically addressed emails

In this section possible solutions for the reply of semantically addressed emails will be explained.
To make the concept of semantic addressing useful you also have to think about the reply of emails has to be thought for. This concept is very helpful and therefore should also work for semantic email addressing.
Generally the reply to emails is not a big problem. If a user receives an email he opens it and can reply to it to send the author of the email an answer. Therefore the recipient has only to use the *Reply* button in his email client. The email client of the receiver has only to read the email address of the sender in the email header. After this step the email address of the user who send the email is automatically inserted in the *To* field of the answering email. Also the subject and the original message are inserted in the mail with special headers that the user who will get the reply knows that the message was replied. For example in the subject the extension *Re* was extended.
If an email was addressed semantically there will not be a difference about the single reply, only to the author of the email. The user who sends the email defined the attributes which the recipients defined and sends the email to 100 users for example. If any user wants to reply the semantically addressed email he also has the button *Reply* (see figure 3.1) in his email client and uses it to answer to the email. In this case the email address of the author has also been inserted in the *To* field, the subject was extended with

Replied and the original message has been inserted in the message body. A bigger problem is the *Reply All* routine which will be explained at point 3.5.

3.5 Replying to all semantically addressed emails

To answer to all users who received the "*normal*" addressed email is not very difficult. The email client of the recipient has to parse all email addresses which are defined in the *To* and the *Cc* field of the email. When the recipient uses the *Reply* button all the addresses which were written in these fields will be inserted in the *To* field of the "answering" email. After this step the replying user inserts his email and sends the email to all users. To answer to semantically addressed emails is a little bit more difficult. If a person receives a semantic addressed email, he also has to know if there are other persons who also received the email, without the knowledge of the real email addresses.
If a user sent an semantically addressed email, he had to define the vital attributes in a search filter (see figure 2.4). To make it possible for the receiving part, to reply the email to all receivers, the "*replier*" has to know, which attributes the sender defined and the type of email addressing (semantic or "*normal*"). Two possibilities to achieve the attributes and identify the addressing type will be explained.

3.5.1 Signature

Today signatures are very popular and widely used. The user can define within a signature attributes which he always wants to send with the email. For example for business purpose the user inserts his telephone number, fax number, address of the company, and some other for him vital attributes. Therefore signatures could also be used to send the attributes which defined all the recipients.
The email client of the sender adds an unique signature to the message which

the sender does not see (see figure 3.1). Only the receivers of the email see this signature. If a receiver wants to answer to all other receivers he has to push the *reply* button of the received email. The client looks for the signature in the email and parses it to gain the attributes which the sender also defined in his email. So the replying receiver automatically *defines* the vital attributes to generate the semantic email address.

3.5.2 Attachment

Another possibility is to send an added attachment with the email to all receivers. In this attachment the vital attributes of all the recipients could be listed. An advantage compared to the solution described at point 3.5.1 would be, that the email has not to be changed, only an attachment has to be added. A big disadvantage is, that some email clients can block attachments and so the user has not the possibility to reply to the email.
An attachment is always a point of trustability because it can contain a virus or it can be exchanged through another attachment where the receiver's attributes are completely different to them who actually received the email.

3.6 Forwarding semantically addressed emails

The forwarding process of semantically addressed emails is not very difficult and not a big difference to the nowadays email forwarding routine. If a user wants to forward an email he has to use the *forward* button of his email client. Afterwards the subject field is extended with an information that the email is forwarded for example with an *FWD* char sequence followed by the original subject. The original message of the email will also be inserted in the message body. The user only has to insert the email addresses of the recipients to whom the email should be forwarded.
If a user wants to forward a semantic addressed email the forwarding routine is completely the same. A little difference is if he wants to forward it

to address the receivers semantically. Therefore he has to *click* into the *Semantic To* field whereby the semantic search filter (see figure 2.4) appears with whom the user defines the attributes of the forwarding email recipients. Instead of the *forwarding* button in the most common email clients the recipient can use the *Semantic To* field to input the new attributes which the user or group of people define who should be the receivers of the forwarded email.

Chapter 4

Application and conclusions

Semantic email addressing is a very useful tool to address many people dynamically but is this really useful in the real life where spamming becomes more and more popular?

Today everyone has to anger with the daily never ending spam. With a tool like the semantic email client spammers would have a very potent tool to overfload us with spam. That is the reason why semantic email addressing in this form would not be the option. Of course, there could be implemented nowadays spam filters in the email client or users who want to use this technology can define an attribute, which allows only to them known users to address them semantically.

The real domain of application for semantic email addressing is in business settings. Where users don't have to estimate to receive spam from the other employees. The dynamic concept of this technology can be fully used in such a setting because of the fluctuation of employees, the alternation of employees to other departments and so on.

Another domain where the application can be used is where customer re-

lationship concepts take part. For example in a hotel, where guests receive newsletters, offers and other mails. In fact, we tested semantic email addressing in a concrete use case in a hotel. To use this concept the filter shown in Figure 2.4 had to be adjusted with the essential attributes for addressing. So the attributes which were defined in the hotels SQL database where added to the search filter of the semantic email client.
Before semantic email addressing was used, the hotel had the following mailing lists in which the guests where enroled by the reception employees, to have a possibility to send newsletters concerning certain offers.

- *allguests* - all guests who want to receive newsletters were enroled.
- *summerguests* - guests who visited the hotel in summer and want to receive newsletters were enroled.
- *winterguests* - guests who visited tho hotel in winter and want to receive newsletters were enroled.
- *golfguests* - guests who booked a golf package and wandt to receive newsletters were enroled.

With the semantic email client the hotel had now the possibility to define dynamic groups who would receive the emails, and was not bound to the above mentioned mailing lists. For example if the hotel wanted to send a newsletter concerning wellness offers for which it had no mailing list, an employee had to write an SQL-Query which gave as a result all the guests email addresses which fulfilled this query. Afterwards all the email addresses had to be copied into the none semantic email client. This procedure costs a lot of time and so a lot of money.
With the semantic email client the employee only had to define the wellness attribute and the newsletter was send out.

Another big disadvantage of mailing lists in this hotel was that the employees had to enrol the guests to the specific mailing lists, when a guest called the hotel and was interested on a specific offer.
A possible workflow of a guest's phone call is:

- guest calls hotel and is interested in the offer *golf unlimited* and in another offer called *learn to ski*
- the reception employee inserts the guests attributes like surname, forename, address, phone number, interests ..., into the SQL database
- guest thinks about the offer and will call in a view days
- employee enrols the guests into the mailinglists golfguest and winterguest
- guest calls again and wants to book the *golf unlimited* offer
- employee stores the reservation and has to unroll the guest of the *winterguests* mailing list

The hotel receives on average 50 offers per reception employee a day. So the above mentioned workflow is not really efficient because the employee first stores the data into the SQL database and afterwards has to do the enrolment to the specific mailing lists.
With semantic email addressing the employee only has to store the data into the database. The complete enrolment process is omitted and money is saved because the employees save a lot of time in which they can perform more important work.
The same workflow as mentioned above using semantic email addressation is:

- guest calls hotel and is interested on the offer *golf unlimited* and on another offer called *learn to ski*

- the reception employee inserts the guests attributes like surname, forename, address, phone number, interests (golf, ski) ..., into the SQL database
- guest thinks about the offer and will call in a view days
- employee stores the reservation

To send now a newsletter concerning the offer *golf unlimited* the employee defines the search filter of the semantic email client with the attributes *guests who have reservation* and *guests who booked golf unlimited*. So the database has not to be changed. This guests can also receive newsletters concerning the offer *learn to ski*, if the employee defines the search filter like *guests who are interested in skiing*.

Bibliography

[1] Dan Brickley and Libby Miller. Foaf vocabulary specification 0.91. `http://xmlns.com/foaf/spec/`, nov 2007.

[2] Lee-Ming Zen Michael Kassoff, Charles Petrie and Michael Genesereth. Semantic email addressing: Sending email to people, not strings. *AAAI 2006 Fall Symposium on Integrating Reasoning into Everyday Applications.*, 2006.

[3] Charles Petrie. Semantic attention management. *IEEE Internet Computing*, 10(5):93–96, 2006.

[4] Steffen Staab and Rudi Studer, editors. *Handbook on Ontologies*. International Handbooks on Information Systems. Springer, 2004.