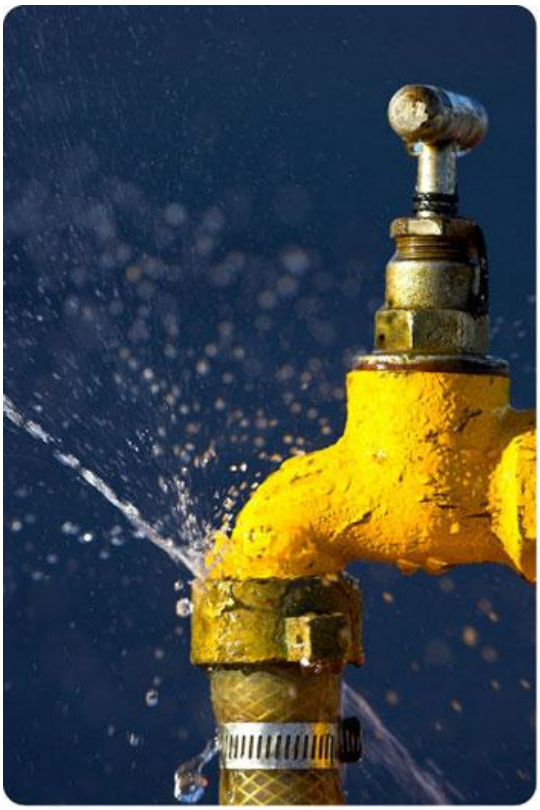
Ben Gurion University

Deutsche Telekom Labs

Preventing Data Leakage via E-mail

ARD Document



Authors: *Rostislav Pinski, Dmitry Kaganov, Eli Shtein, Alexander Gorohovski*

Version 1.1

Table of Contents

1. Introduction.....4-16
    1.1 Vision.....4
    1.2 The Problem Domain.....5-6
    1.3 Stakeholders.....7
        1.3.1 Users.....7
        1.3.2 Researchers.....7
    1.4 Software Context.....8-10
        1.4.1 Plug-in for the outlook e-mail client.....8
        1.4.2 Middle-ware.....8-9
        1.4.3 Server.....9
        1.4.4 Administrator' GUI.....10
    1.5 System Interfaces.....11-16
        1.5.1 Hardware Interfaces.....11
        1.5.2 Software Interfaces.....11-14
            1.5.2.1 Outlook mail client API and plug-ins mechanism.....11
            1.5.2.2 Exchange server API.....11
            1.5.2.3 "doc" and "pdf" files API.....11
            1.5.2.4 Middle-ware interfaces.....12-13
            15.2.5 Server Interfaces.....13-14
        1.5.3 Events.....14
        1.5.3.1 Plug-in Events.....14
        1.5.3.2 Middle-ware events.....14-15
        1.5.3.3 Server events.....15
        1.5.3.4 Administrator's GUI events..... 15-16
2. Functional Requirements.....17-21
    2.1 Plug-in.....17
    2.2 Middle- ware.....18
    2.3 Server.....19
    2.4 Administrator GUI.....20-21
3. Non-Functional Requirements.....22-25
    3.1 Performance Constraints.....22-24
        3.1.1 Speed.....22
        3.1.2 Reliability.....22
        3.1.3 Safety & Security.....22
        3.1.4 Availability.....23
        3.1.5 Capacity.....23
        3.1.6 Portability.....23
        3.1.7 Usability.....23
        3.1.8 Software Interfaces.....23
        3.1.9 Reusability - Modularity of the system.....23-24

3.2 Platform Constraints.....24
3.3 SE Project Constraints.....24-25
3.4 Special Restrictions & Limitations.....25
4 Usage Scenarios.....26-46
4.1 User Profiles — The Actors.....26
4.1.1 The E-mail client.....26
4.1.2 The Exchange server.....26
4.1.3 The Simple User.....26
4.1.4 The Administrator.....26
4.2 Use Cases.....27-46
4.3 Special Usage Considerations.....46
5 Appendices.....47-54
5.1 Appendix 1 – Catching the "send e-mail" event.....47
5.2 Appendix 2 – Glossary: meaning of professional terms used in this document.....48-49
5.3 Appendix 3 – The theoretical model used by our system.....50
5.3.1.1 Proposed solution.....50
5.3.1.2 Classification model.....51-52
5.3.1.2.1 Baseline classification model.....51
5.3.1.2.2 Proposed classification model.....52
5.3.1.2.3 Cascading the models.....52
5.4 Appendix 4 - Descriptions of similar products.....53-54

1 Introduction

1.1 Vision

Most of the people that intensely use e-mail communication can confirm that at least once they have sent an e-mail to the wrong person. Such a mistake can be very damaging. Inappropriate jokes may be sent to a supervisor, financial reports may be sent to a competitor or a broker, love letter to the wrong man or woman.

Modern business activities rely on extensive e-mail exchange. E-mail "wrong recipients" mistakes have become widespread and the severe damage caused by such mistakes constitutes a disturbing problem for individuals but for organizations such mistakes might not only cause a disturbing problem but also cost a lot of money. Various solutions attempt to analyze email exchange for preventing e-mails to be sent to wrong recipients. However there is still no satisfying solution: many email addressing mistakes are not detected and in many cases correct recipients are wrongly marked as potential addressing mistake.

In our project we define an e-mail leakage as an e-mail that intentionally or unintentionally reaches a recipient it should not reach. We present a new approach to be applied in organizations for preventing e-mail leakage. According to the proposed approach we analyze the e-mails communicated between all members of the organization, extract the topics discussed in the organization via e-mail exchange and derive groups of members that share the same topic. Consequently, each member may belong to several topic groups and a topic group may contain members that have never communicated before. When a new e-mail is composed each recipient is classified as a potentially leak recipient or a legal one. The classification is based not only on the e-mails exchanged between the sender and the recipient, but also based on the topic groups which they belong to.

The goal of the project is to develop a system for **preventing data leakage via email** for some e-mail client. The system will work using a theoretic model developed by researchers at Deutsche Telekom laboratories at BGU.

The system will work the following way:

- Each e-mail will be checked before it is sent (after the user hits the "send" button).
- In case all the recipients are valid the e-mail will be sent to all the recipients.
- A list of non-valid recipients will be shown to the user.
- The user will be able to ignore systems recommendations (e.g., send the e-mail to a recipient that the system detects as non-valid recipient).
- The user will be able to change the content of the e-mail.
- Optionally – a list of additional suggested recipients for this e-mail will be shown to the user.
- Optionally – the user will be able to send the e-mail easily to any of the recipients from the suggested recipients list.

1.2 The Problem Domain

The problem domain is to prevent leakage of data through e-mails. Our system works on Windows environment and communicates with outlook and the exchange server of the organization.

Our system will be operated in large environment that consists of 3 main components:

1. **The outlook e-mail client** – the e-mail client which the system is developed for. Our system will have a component that interacts with this e-mail client (a plug-in). This component will wait till send e-mail events occur, catch and cancel them.

The plug-in will also add to the outlook e-mail client some graphic components such as:

a. A checkbox or a button that allows deleting e-mail's data from the system's database (in case the e-mail was sent/ received by mistake).

b. A list boxes of invalid recipients, recommended recipients (optionally) and valid recipients.

c. Buttons that allow sending the e-mail to valid/ non-valid/ recommended (optionally) recipients.

2. **Exchange server** – the e-mail server of the organization which contains all the e-mails sent to, form and inside the organization. E-mails will be sent by the plug-in to the exchange server of the organization (both e-mails found by the system to be legal and e-mails sent by the user to some recipients despite the system's recommendations). The e-mail client (outlook) pulls e-mails from there.

3. **System Core** – (a) receives all the send e-mail events that occur in the e-mail client, (b) checks validity of recipients, (c) returns for each of the recipients whether it is a valid recipient or not, (d) in case all the recipients are valid it sends the e-mail to all of them, (e) suggests another recipients (optionally), and (f) deletes from the system e-mails sent by mistake.

The system core also has an administrator mode which allows configuring the system.

The external interfaces that are maintained in this product:

The outlook plug-ins mechanism: a set of interfaces, functions, and events provided by outlook to application developers, which enables them to write software applications that interfaces with the outlook e-mail client (mainly by COMs'). This API enables applications to exploit the power of outlook. Using this API, we can develop applications that run successfully on almost all versions of outlook while taking advantage of the features and capabilities unique to each version.

Exchange server Application Programming Interface: a set of functions provided by the exchange server to application developers, which enables them to write software applications that interfaces with the exchange server.

Microsoft Word document ("doc") Application Programming Interface: a set of functions and a format, which enable to write applications that manipulate a content of Microsoft Word documents. The product will be able to handle any text that is stored in "doc" documents**.**

"Pdf" document Application Programming Interface: a set of functions and a format, which enable to write applications that manipulate a content of "pdf" documents. The product will be able to handle any text that is stored in "pdf" documents.

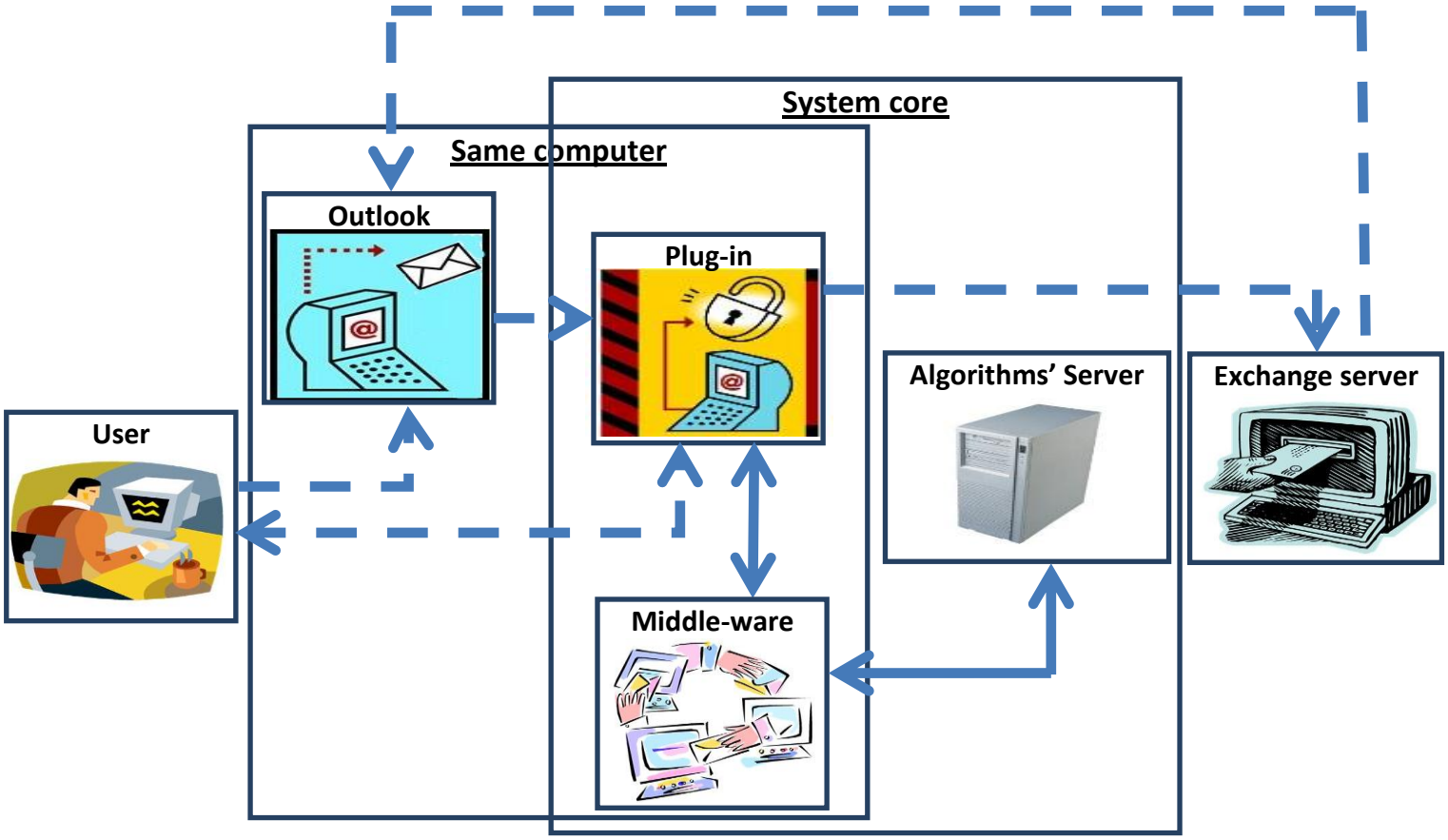


Figure 1.1 – System architecture

1.3 Stockholders

1.3.1 Users

The system's users are:

1. Employees - organization members who use the outlook e-mail client as their work related e-mail manager and want to prevent data leakage through e-mails they send.

Our system receives the e-mails from the outlook e-mail client immediately after the user hits the send button (and before the e-mail is sent), checks them for validity, ask the user about the recipients considered to be not valid, and send the e-mail to all recipients considered as valid and to recipients considered invalid but approved by the user. Moreover, it suggests the user additional recipients to the e-mail (optionally). For system configuration the system administrator is required.

2. Administrator – In addition to being a simple user, the administrator user is capable of configuring the system (selecting the threshold of the system, load balance of the system's server etc.), and deal with some other issues related to the system and described in the "User Profiles — The Actors" section of the "Usage Scenarios" part.

1.3.2 Researchers

The E-mail leak detector plug-in is developed for researchers of Deutsche Telekom Labs at BGU, for the purpose of advance the research in the field of e-mails data leakage.

1.4 Software Context

1.4.1 Plug-in for the outlook e-mail client

The plug-in is the part of the system that communicates with both the outlook mail client and the user. This component will have to interface with outlook plug-ins mechanism, and the organization's exchange server interface.

The plug-in catches the send e-mail events that occur in outlook and passes their data (the sender, the recipients, the e-mail, the subject, attached files etc.) to the "Middle-ware". It should be able to interact with "doc" and "pdf" files and to get all the textual data out of them. It also gets answers from the "Middle-ware" (for each recipient whether he is valid or not), displays a list of invalid recipients, and a list of recommended recipients (optionally). It sends the e-mail to all the recipients considered by him to be valid, to all recipients chosen by user despite of system recommendations and to users recommended by the system (optionally) and chosen by the user. It also allows the user to edit the content of the e-mail, and to perform the check again. Finally it sends to the "Middle-ware" the data of all recipients that get the e-mail.

The plug-in will also support for the users an option to cancel e-mail sent/got by mistake (e.g. delete their data from the system database). In a case a user choses to cancel an email sent by him, the plug-in will display him the list of all recipients of the mail so that the user will be able to choose the recipients he sent them the e-mail by mistake. The plug in will pass all the relevant data to the "Middle-ware".

In addition the plug-in displays error and success messages to the user.

1.4.2 Middle-ware

The part of the system which communicates between the plug-in and the server. Each client will have his own "Middle-ware" (e.g. there will be a "Middle-ware" per client).

The "Middle-ware" can get some different kinds of messages from the plug-in, and react to them differently:

a. check e-mail validity: the plug-in sends the "Middle-ware" an e-mail's data (the sender, the recipients, the e-mail, the subject, attached files etc.). The "Middle-ware" processes the e-mail, its' subject, and its' attached files the following way: it removes all the stop words, stems all the words encrypts the stemmed words and sends all that data to the server. In a case the server is overloaded, the server will send to the "Middle-ware" all the relevant data to perform the validity check, and the check will be performed in the "Middle-ware".

The "Middle-ware" will return the list of valid recipients, the list of invalid recipients and the list of recommended recipients (optionally) to the plug-in.

b. delete e-mail: The "Middle-ware" gets from the plug-in all the relevant data (e.g. the sender, the recipients, some identification data of the e-mail) of the e-mail to be deleted and passes it to the server.

c. send e-mail: The "Middle-ware" gets from the plug-in all the relevant data (the sender, the recipients, the e-mail) of the recipients who the e-mail was sent to (e.g. recipients considered by the system to be legal, recipients added due to

recommendations of the system and recipient who got the e-mail despite systems' recommendations) and passes it to the server in order to update the server's database. All this data will be also written to appropriate log files.

1.4.3 Server

The part of the system which contains all the data of the sent e-mails (the system's database), all the centroids (vectors of users that can send mails on certain topics each to the other), and system's configurations. The system will contain one server which all the clients will communicate through their "Middle-ware" component.

The server can get from "Middle-wares" some different kinds of messages, and react to them differently:

a. check e-mail validity: the "Middle-ware" sends to the server an e-mail's data (the sender, the recipients, all the word of the e-mail and its attachments being stemmed and encrypted). In case the server is not overloaded it will check the validity of the e-mail, update its' database with the data of the valid e-mails, check for additional advised recipients (optionally) and send to the "Middle-ware" lists of valid, invalid and advised recipients. In the case the server is overloaded it will send to the "Middle-ware" all the relevant data and the "Middle-ware" will perform the check.

b. delete e-mail: The "Middle-ware" sends to the server all the relevant data (e.g. the sender, the recipients, some identification data of the e-mail) of the e-mail to be deleted and the server deletes it's data from its database.

c. send e-mail: The "Middle-ware" sends to the server all the relevant data (the sender, the recipients, the e-mail) of the recipients who the e-mail was sent to (e.g. recipients considered by the system to be legal, recipients added due to recommendations of the system and recipient who got the e-mail despite systems' recommendations) and the server updates it's database with this data. The data of e-mails sent in contradiction to the system advices will be saved in the database marked in a special way ("?").

In addition once per certain time the systems database will be backed up, old e-mail's data will be deleted from the database, and the centroid vectors will be updated due to last sent mail's data.

The server will control data base for the legal e-mails sent inside or out form the organization, and the centroids (clusters of users that can pass e-mails on certain topics each to the other).

Also the server's configurations can be changed by she system administrator, and the system administrator can delete from the database emails marked by "?" which are invalid e-mails, and remove the "?" mark from data of valid e-mails.

1.4.4 Administrator's GUI

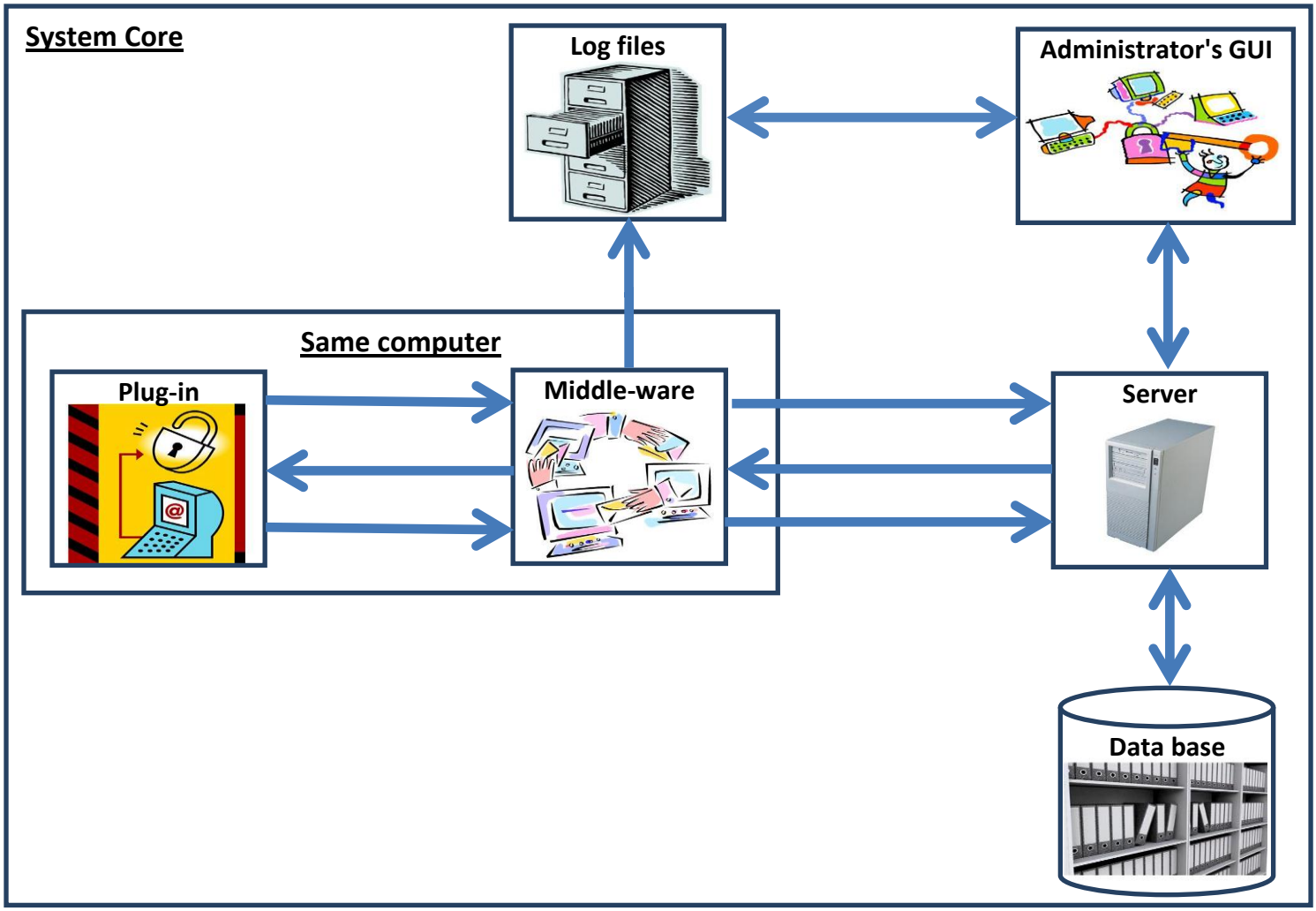
A component which is used by the administrator to configure the system, and to belong users to topics (centroids) in a manual way. This component should contain the following options:

a. log-in as the system's administrator.

b. add new user: in case a new worker joins the organization.

c. remove user: in case a worker leaves the organization.
d. add new subject group.
e. remove a subject group.
f. update user's subject groups: in case a worker is removed from some project, added to some project, his work place in the organization is changed, etc.
g. deal with e-mails marked by question marks: the system administrator can delete from the database emails marked by "?" which are invalid e-mails, and remove the "?" mark from data of valid e-mails. For this purpose the "Administrator's GUI" component should be able to read data from the log files of the system (in order to get the full information about e-mails sent in contradiction to system's recommendations). It should also be able to write data to the log files in order to updated the information stored in them due to the decisions made by the system administrator (whether some e-mails are legal or not).
h. set system's configurations: set the threshold, the period of time e-mails should be saved in the system, the load factor of the server, the amount of days to pass between centroid vectors updates.
i. log out from the system.

Figure 1.2 – System Core architecture



1.5 System Interfaces

1.5.1 Hardware Interfaces

Our system has no hardware components and interfaces

1.5.2 Software Interfaces

1.5.2.1 Outlook mail client API and plug-ins mechanism

Our plug-in should communicate with the outlook mail client through its API and plug-ins mechanism using the COM. Our application will catch all the send e-mail events that occur in the outlook, cancel the sending of the e-mail in the caught event, receive all the data and content of the e-mail (e.g. the sender, recipients, subject, text, attached files etc.), and display the user failure messages (if failures occur).

Our application will also add some graphical components to the standard outlook view such as:

a. checkbox or a button that allows deleting e-mail's data from system's database (in case the e-mail was sent or got by mistake).

b. list boxes of invalid recipients, recommended recipients (optionally), and recipients who got the e-mail.

c. Buttons that allows sending the e-mail to not recommended / recommended (optionally) recipients.

All this will be done using the "Microsoft.Office.Interop.Outlook" library of C#.

1.5.2.2 Exchange server API

Our system should communicate with the organization's Exchange server through its API. E-mails which will be found by the system to be legal or marked by the user as legal will be sent by the plug-in to the Exchange server of the organization. In order to send e-mails to the exchange server we will use the "send" method of the "MailItem" object provided by the "Microsoft.Office.Interop.Outlook" library of C#.

1.5.2.3 "doc" and "pdf" files API

Our system should be able to get the whole text of any "doc" and "pdf" documents attached to e-mails. It will do it using "doc" and "pdf" files format and API. For this purpose we will use the free Apache POI Package developed by Apache Foundation which allows working with Microsoft Word documents in Java and the free Java library for PDF from Etymon Consulting, which allows working with "pdf" files in java.

1.5.2.4 Middle-ware interfaces

Operation: pass data to check e-mail validity
Used By: Plug-in
Parameters:
- **Recipients**: The list of e-mail recipients (including the recipients appearing in CC and BCC).
- **Sender**: The e-mail sender
- **Subject**: The subject of an e-mail
- **Body**: The body of an e-mail
- **Attached files**: The files attached to an e-mail

Return Values:
- **Valid recipients**: list of valid e-mail recipients
- **Invalid recipients**: list of invalid e-mail recipients with their results
- **Suggested recipients**: list of recipients the system suggests the user to add to the recipients of the e-mail, with their results.
-**Threshold**: The bound which differentiate between valid and invalid recipients.

Operation: check e-mail validity
Used By: Plug-in
Parameters:
-**Centroids**: all the system centroids (vectors representing the typical e-mails associated with the e-mail).
-**Threshold**: The bound which differentiate between valid and invalid recipients.

Return Values:
- **Valid recipients**: list of valid e-mail recipients
- **Invalid recipients**: list of invalid e-mail recipients with their results
- **Suggested recipients**: list of recipients the system suggests the user to add to the recipients of the e-mail, with their results.
-**Threshold**: The bound which differentiate between valid and invalid recipients.

Operation: pass data to Update Database with sent e-mail data
Used By: Plug-in
Parameters:
- **Recipients**: The list of e-mail recipients (including the recipients appearing in CC and BCC).
- **E-mail**: All the content of the e-mail to be sent (subject, body, attached files).

Return Values: none

Operation: pass data to delete e-mail data
Used By: Plug-in
Parameters:
- **Recipients**: The list of e-mail recipients (including the recipients appearing in CC and BCC).
- **Sender**: The e-mail sender
- **Date:** the date an e-mail was sent
- **Time:** the time an e-mail was sent

- **E-mail**: All the content of the e-mail to be sent (subject, body, attached files).

Return Values: none

1.5.2.5 Server Interfaces

Operation: set system's configurations
Used By: System administrator
Parameters:
- **Load factor**: The bound which differentiate the system from being loaded or not.
- **Threshold**: The bound which differentiate between valid and invalid recipients.
- **Time to keep data:** amount of months e-mails data should be kept in the system
-**Time to centroid updates:** amount of days to pass between centroid vectors updates.
- **Users to add**: list of new users with the topic groups they should be assigned to.
- **Users to remove**: list of users with their old topic groups they should be removed from.
- **Users to update**: list of users with their old topic groups they should be removed from and their new topic groups they should be assigned to.
-**Subject groups to add**: list of topic groups that should be added to the system.
-**Subject groups to remove**: list of topic groups that should be removed from the system.

Return Values: none

Operation: Check e-mail validity
Used By: Middle-ware
Parameters:
- **Recipients**: The list of e-mail recipients (including the recipients appearing in CC and BCC).
- **Sender**: The e-mail sender
- **Subject**: The subject of an e-mail
- **Body**: The body of an e-mail
- **Attached files**: The files attached to an e-mail

Return Values:
In case the server is not loaded:
- **Valid recipients**: list of valid e-mail recipients
- **Invalid recipients**: list of invalid e-mail recipients with their results
- **Suggested recipients**: list of recipients the system suggests the user to add to the recipients of the e-mail, with their results.
-**Threshold**: The bound which differentiate between valid and invalid recipients.

In case the server is loaded:
-**Centroids**: all the system centroids (vectors representing the typical e-mails associated with the e-mail).
-**Threshold**: The bound which differentiate between valid and invalid recipients.

Operation: Update Database with sent e-mail data
Used By: Middle-ware
Parameters:
- **Recipients**: The list of e-mail recipients (including the recipients appearing in CC and BCC).
- **E-mail**: All the content of the e-mail to be sent (subject, body, attached files).

Return Values: none

Operation: delete e-mail data
Used By: Middle-ware
Parameters:
- **Recipients**: The list of e-mail recipients (including the recipients appearing in CC and BCC).
- **Sender**: The e-mail sender
- **Date:** the date an e-mail was sent
- **Time:** the time an e-mail was sent
- **E-mail**: All the content of the e-mail to be sent (subject, body, attached files).

1.5.3 Events

1.5.3.1 Plug-in Events

Plug-in events are events the system should handle for controlling the input from the user, which are the system's input events, and events the system should handle for controlling output to the user, which are the system's output events. The events should be handled by the plug-in:

a. Send e-mail – event that happens when user hits the "send" button. There are some different cases of that event and for each of them the system should behave differently: check e-mail validity, send e-mail all valid, considered by the system to be invalid but marked by the user as valid, and additional advised (optional) recipients.

b. Delete e-mail from system's data base – event that happens when the user hits the "got / sent by mistake" button (or marks the e-mail in any other special way).

c. Get answers from the "Middle-ware" – event of getting lists of valid, invalid and additional advised (optionally) recipients for a mail, and displaying the user lists of invalid and additional advised recipients.

1.5.3.2 Middle-ware events

"Middle-ware" events are events the system should handle mainly for performing communication between the plug-in and the server. They are inner system's events. The events should be handled by the "Middle-ware":

a. Check e-mail validity: event that happens when the plug-in sends the "Middle-ware" an e-mail's data (the sender, the recipients, the e-mail, the subject, attached files etc.). The "Middle-ware" processes the e-mail, and sends it to the server.

b. Delete e-mail: event that happens when the "Middle-ware" gets from the plug-in all the relevant data (e.g. the sender, the recipients, and some identification data of the e-mail) of the e-mail to be deleted. The "Middle-ware" passes all this data to the server.
c. Pass recipients data: event that happens when the "Middle-ware" gets from the plug-in all the relevant data (the sender, the recipients, the e-mail)of the recipients who the e-mail was sent to (e.g. recipients considered by the system to be legal, recipients added due to recommendations of the system and recipient who got the e-mail despite systems' recommendations). It passes all that data to the server and writes it to an appropriate log files.
d. Get answers to e-mail validity: event that happens when the "Middle-ware" gets from the server answer about each of the mail's recipients. In case the server is overloaded the appropriate "Middle-ware" will get all the relevant data (e.g. centroids and links) and perform the validity check by itself.

1.5.3.3 Server events

Server events are events the system should handle mainly for performing users' requests and maintaining the system to keep its data updated and safe. They are inner system's events. The events should be handled by the server are:
a. Check e-mail validity: event that happens when the "Middle-ware" sends to the server an e-mail's data (the sender, the recipients, all the word of the e-mail and its attachments being stemmed and encrypted). In case the server is not overloaded it checks the validity of the e-mail.
b. delete e-mail: event that happens when the "Middle-ware" sends to the server all the relevant data (e.g. the sender, the recipients, some identification data of the e-mail) of the e-mail to be deleted and the server deletes it's data from its database
In the case the server is overloaded it will send to the "Middle-ware" all the relevant data and the "Middle-ware" will return to it the data of valid e-mails to update the database.
c. Save recipients data: event that happens when the "Middle-ware" sends to the server all the relevant data (the sender, the recipients, the e-mail) of the recipients who the e-mail was sent to (e.g. recipients considered by the system to be legal, recipients added due to recommendations of the system and recipient who got the e-mail despite systems' recommendations), and the server updates it's database with this data.
d. Back up data base: event that happens once per certain time (defined in the system's configurations).
e. Delete old e-mails data: old e-mail's data will be deleted from the database,
f. Update centroid vectors: event that happens once per certain time (defined in the system's configurations).

1.5.3.4 Administrator's GUI events

Administrators GUI events are events the system should handle mainly for system's maintenance and configurations and for controlling the input from the administrator.
They are inner system's events. The events should be handled by the server are:

a. log-in as the system's administrator: event that happens when the administrator logs in to the system. The system insures his identity and displays him administrator's options window.
b. add new user: event that happens when the administrator adds a new user to the system (in case a new worker joins the organization).
c. remove user: event that happens when the administrator removes existing user from the system(in case a worker leaves the organization).
d. add new subject group: event that happens when the administrator adds a new subject group to the system.
e. remove a subject group: event that happens when the administrator removes an existing subject group from the system.
f. update user's subject groups: event that happens when the administrator updates some existing user's subject group(in case a worker is removed from some project, added to some project, his work place in the organization is changed, etc.).
g. deal with e-mails marked by question marks: event that happens when the administrator choses to update data of e-mails sent to recipients considered by the system to be invalid.
h. set system's configurations: event that happens when the administrator sets system's configuration.
i. log out from the system: event that happens when the administrator logs out from the system. The administrator's functionality window is being closed.

2 Functional Requirements:

2.1. Plug-in:

#	Function Requirement	Description
1.	Monitor	Ability to listen to events occurrences from the outlook client, e.g. SEND done by the user, received messages from the "Middle-ware".
2.	Prompt The User	Ability to receive messages from the Middle-Ware and do the following: • Suggest the user to add recipients he\she has forgotten to add. • Suggest the user to remove recipients he\she has added accidently. • Allow the user to change the content of the e-mail, if and only if the system returned list of invalid recipients. • Display error messages in case errors occurred e.g. lost connection with the Server, the "Middle-ware" or the exchange server. • Display success messages e.g. the mail was sent successfully.
3.	Send Monitored Data	Collect the relevant information from the mail client and send it to the "Middle-ware". Relevant information, e.g. the sender, the recipients list, the subject of the mail, the text field, the attached files of type pdf or doc, the date and the time the e-mail was sent.
4.	Activation	Ability to activate automatically when Outlook starts.
5.	Connection	Ability to connect to the email client.
6.	Cancel E-mail	Ability to choose to remove from the system an e-mail data that has been sent or received by mistake.

2.2 Middle-ware:

#	Function Requirement	Description
7.	Stemming	Ability to create a vector that contains the base words by stemming of all the words which are not stop words in the mail, the words received from the plug in.
8.	Encryption	Ability to encrypt all the stemmed words.
9.	Send Data	Ability to send data to the server, the data is the encrypted and stemmed words.
10.	Advice For Additional Recipients (optional)	Ability to recommend additional recipients for the e-mail using the data received from the server – this should be done when the server is loaded.
11.	Check Validity	Ability to check if the recipient is valid with the data received from the server – this should be done when the server is loaded.
12.	Logging	• Ability to log relevant data accordingly to defined mode. Relevant data includes events such as: user sending an email to a certain recipient despite the system's recommendation not to do so. • Ability to log all the data required for the system statistics, e.g. the content of the e-mail, the sender, the recipients, the similarity score of each recipient, the threshold of each recipient, for which of the recipients the user took the advice of the system, and for which he\she didn't. • Ability to delete from the log file logs of e-mails that were canceled.
13.	Read PDF And DOC Files	Ability to get all the text from all pdf and doc files attached to the e-mail.

2.3 Server:

#	Function Requirement	Description
14.	Back Up	Ability to make back up of the data base.
15.	Send Data	Ability to send data to a client's "Middle-ware". The data is a list of valid recipients, invalid recipients and additional advised recipients. For each of the recipients a similarity score and a threshold will be added.
16.	Check Validity	Ability to check if the recipient is valid. If the server is loaded (configured Load balance by the administrator), send data to the "Middle-ware" for further treatment.
17.	Advice For Additional Recipients (optionally)	Ability to recommend additional recipients for the e-mail using the data received from the server – done when the server is not loaded (configured Load balance by the administrator).
18.	Update Data Base	Ability to update and store all relevant information to the database with the data received from the "Middle-ware". The update is performed for valid recipients and for all additional recipients added by the user according to the system's recommendations.
19.	Update Clusters	Ability to update the clusters once in two days at 3:00 AM.
20.	Delete Old Data From The Database	Ability to delete old data from the database once in a certain period defined by the history timeframe size. During this deletion the system should still be able to serve the clients' requests.
21.	Delete E-mails Sent/Received By Mistake	Ability to delete data of e-mails that were sent or received by mistake and marked as such by the sending\receiving user.
22.	Deleting data from log files	Ability to delete from the log file data of e-mails that were marked by the administrator as legal e-mails.

2.4 Administrator GUI:

#	Function Requirement	Description
23.	Registration	Ability to register an administrator user, the following details are required: username, password, employee number, ID, phone number, etc.
24.	Login	Ability to login as an administrator user to the system, the following details required: username, password.
25.	Logout	Ability to logout as administrator to become simple user.
26.	Configuration Options	• Ability to configure the history timeframe size (after the period of timeframe the data stored before that should automatically be deleted). • Ability to configure the content similarity threshold (how similar the content of the e-mail to older e-mails on the same topic - centroid). • Ability to configure the mode, i.e. level of system restriction (the lower bound of a score defines an e-mail as legal). • Ability to configure Load balance of the server.
27.	Add New User	Ability to add a new user to the system and to add him into at least one cluster (users' group) – happens when a new user joins the organization.
28.	Remove Existing User	Ability to remove an existing user from the system and to remove him from all the clusters (users' groups) he\she was in – happens when a user leaves the organization.
29.	Update User's Groups	Ability to update a certain user's groups, e.g., to remove the user from one of the groups he\she belongs to, or to add the user to another group – happens when position is changed.
30.	Add new Cluster	Ability to add a new cluster (subject group) to the system – happens when the organizations start dealing with a new issue.
31.	Remove Cluster	Ability to remove an existing cluster from the system – happens when the organization stops dealing with a specific issue.
32.	Unmark E-mails Marked By Question Marks	Ability to remove question marks (?) from emails in the database after they were confirmed as legal by the administrator. Question marks are given to (recipients of) e-mails that have been sent despite the system's

		recommendation.
33.	Delete E-mails Marked By Question Marks	Ability to delete from the database e-mails that were marked with question marks (?) after the administrator decided they are illegal.
34.	Statistics	Ability to produce a summary containing the following data for each e-mail: • The content of the e-mail • The sender • The recipients • The similarity score of each recipient • The threshold of each recipient • For which of the recipients the user took the advice of the system, and for which he\she didn't

3 Non-Functional Requirements:

3.1 Performance constraints

3.1.1 Speed:

1. Registration/login of the administrator should take less than 5 seconds.
2. System's initialization time should be less than 5 seconds.
3. The delay in sending the e-mails should be linear to the size of the e-mail and not longer than half a minute (in extreme cases, for emails with large attachments).

3.1.2 Reliability:

4. The system should be able to detect no less than 94% of the illegal recipients.
5. The upper bound on false alerts (wrong alerts on legitimate recipients) is 30%.
6. A database backup should be done on a daily basis.
7. If the administrator has determined that the user has wrongly labeled a recipient of an email, the system should have the ability to rollback.
8. If an e-mail is tagged by one of its recipients as an e-mail that has been received by mistake, the system should have the ability to rollback.
9. If an e-mail is tagged by the sender as an e-mail that has been sent to a certain recipient by mistake, the system should have the ability to rollback.
10. The system should be capable of restoring the database and the links to at least two days before the crash (if such occurs).

3.1.3 Safety & Security:

11. The data between server and client must be encrypted with SHA1 encryption.
12. The communication channel between server and client must be secured with SSL (Secure Sockets Layer).
13. E-mail data must not be lost or corrupted if no system crash occurs.
14. Disclosure of e-mail data must not occur.
15. Changing system configuration must be password protected.

3.1.4 Availability:

16. The system must be available and functional at any time.

3.1.5 Capacity:

17. The system disk space requirement must be linear to the number of e-mails sent\received.
18. The system must function properly on a computer with 512MB of RAM or more.
19. CPU usage for the system in the client computer must not exceed 9%.

3.1.6 Portability:

20. The system must support Microsoft Windows XP/Vista/7 operation systems with Java Runtime Environment.
21. The system must support Outlook as an e-mail client.

3.1.7 Usability:

22. The system must have user-friendly GUI for administrator mode.
23. The system must be simple to manage for the common user.
24. The system must have a clear status as to when the system is running.
25. The system must have a clear status as to when errors occurred, but not overwhelm the user with redundant messages.
26. The system must support English e-mails.
27. The system must support extracting words from pdf and doc files.
28. The system should be able to sustain 2,000 client connections concurrently.

3.1.8 Software Interfaces:

29. The system must use Outlook server exchange.
30. The system should use services of Windows such as Component Object Model.

3.1.9 Reusability - Modularity of the system:

31. The module for representing and comparing texts must be easily replaceable.
32. The module for computing and representing links must be easily replaceable.
33. The module for computing and representing clusters must be easily replaceable.

34. The system should be adaptable to other e-mail clients.

3.2 Platform constraints:

35. The "Middle-ware" and the server would be developed in Java using "Eclipse" or "NetBeans" run-time environments.

36. The plug-in would be developed in C# using .NET 3.5 Framework with Graphical user interface framework supported by visual studio 9.0.

37. The system would run on Outlook e-mail client.

38. The system will run on Windows operating system only (XP, Vista, 7).

3.3 SE project constraints:

- A user, excluding the administrator, can't configure the system or activate/ deactivate it.
- The system will be tested on several types of users, part of them can exchange valid e-mails on certain topics and others can't. (Several "topic groups" will be created for this purpose).
- The system will be tested while some real people will connect to the system and send e-mails simultaneously. It will also be tested under different loads on the server (including extreme loads) to ensure the system can function with large number of clients (the tests under different loads will be done by simulated clients – e.g. a large number of threads that will send e-mails simultaneously).
- The input data will be collected from e-mails sent by the "tested users".
- The testing method will be white-box all-paths testing.
- Presenting the final system in projects presentation day:
  1. Filling some data to the system's data base.
  2. Letting a group of persons to sit by some PC's.
  3. Asking them to compose some e-mails and to send each of them to people existing in our system's database. Some of the recipients are valid recipients for this e-mail, and the others are not.
  4. Asking the participant to send the e-mail to some of the recipients considered by the system to be invalid.
  5. Asking the participant to mark some of the e-mails as sent by mistake to part of their recipients.
  6. Asking one of the participants to change system's configurations.
  7. Asking one of the participants (that will simulate the system's administrator) to mark some e-mails sent (in contradiction to system's recommendations) as legal e-mails and some of them as illegal e-mails.

8. Asking them to compose some e-mails and to send each of them to people existing in our system's database some of whom are valid recipients for this e-mail, and another – people the person knows shouldn't get this e-mail. – (this will be done also after step 7, in order to insure changes done by the administrator will affect the validity check process of our system).

3.4 Special Restrictions & Limitations:

Not applicable.

4 Usage scenarios

4.1 User Profiles — The Actors

4.1.1 The E-mail client

Referrers to the e-mail client which the user uses to work with his e-mail account (to send, read e-mails etc.).
The e-mail client passes e-mails (the sender, the recipients list, the subject field, the text field, the attached files, the date and time the e-mail was sent) to the plug-in, which is a part of our system.

4.1.2 The Exchange server

Referrers to the exchange server of the organization.
The exchange server gets all the e-mails sent by people from the organization, all the e-mails sent to people from the organization and delivers them to their target.

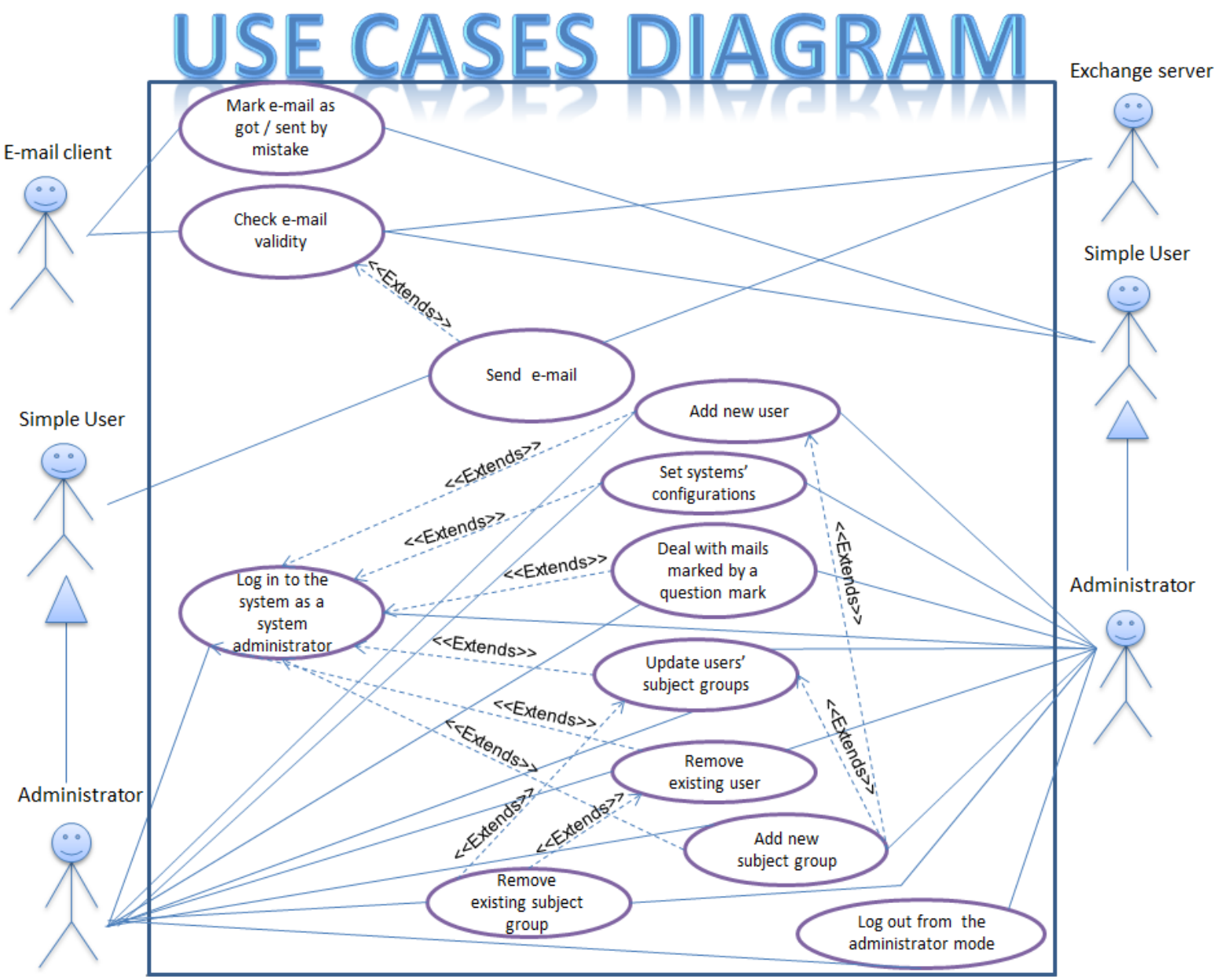
4.1.3 The Simple User

Referrers to a person who uses the system.
The user can send e-mails, ignore the recommendations of the system (send the e-mail to recipients which the system advises they should not get the e-mail), use the recommendations of the system to attach other users to the list of the recipients of the e-mail and mark e-mails as sent / received by mistake (causing their data to be deleted from the system).

4.1.4 The Administrator

Referrers to a person who configures the system.
In addition to being a simple user, the administrator user is capable of configuring the system according to his desires (e.g. selecting the threshold of the system, load balance of the system's server etc.), deal with groups of classifications (e.g. adding new group classification to the system, removing existing group classification, adding a new person to a classification group and removing a person from a classification group), looking in log files for illegal e-mails that were sent and dealing with e-mails that were sent in contradiction to the recommendations of the system (e.g. removing from the log files information about legal e-mails that were written to the log files as illegal e-mails and removing "?" from mails that were sent in contradiction to the recommendations of the system).
A special mode of working with the system is required for the administrator. The administrator should be able to log in to this mode and to log out of it.

4.2 Use-cases


USE CASES DIAGRAM
E-mail client
Exchange server
Simple User
Simple User
Administrator
Administrator
Mark e-mail as got / sent by mistake
Check e-mail validity
Send e-mail
Add new user
Set systems' configurations
Deal with mails marked by a question mark
Log in to the system as a system administrator
Update users' subject groups
Remove existing user
Add new subject group
Remove existing subject group
Log out from the administrator mode
<<Extends>>

4.2.1 UC 1 – Check e-mail validity

ID: 1
Primary actors: E-mail client, Simple user, Exchange server
Description: The simple user checks the validity of an e-mail
Trigger: The user clicks the "send" button (at the first time).
Pre-conditions:
1. The plug-in is installed on the user's e-mail account.
2. The "Middle-ware" is installed on the user's computer.
3. The server of the system is installed and runs on a server computer.
4. The network is working correctly.
5. The e-mail is not empty.
6. The list of the recipients is not empty.

Post-conditions:
In case all the recipients are valid:
1. The e-mail was sent to all of them.
2. The system's database was updated.
3. All the relevant information was written to the appropriate log files.

In case at least one of the recipients is invalid:
1. The user gets on his monitor a list of users which the system advices to add to the recipients list (optionally).
2. The user gets on his monitor a list of users which the system advices not to send them the e-mail.

Flow of events:
1. The user enters the e-mail, the recipients and attaches all the relevant files to the e-mail.
2. The user hits the send button.
3. The plug-in sends all the relevant data to the "Middle-ware".
4. The "Middle-ware" performs stemming of all the words which are not stop words in the e-mail.
5. The "Middle-ware" encrypts all the stemmed words.
6. The "Middle-ware" sends the encrypted words and the e-mail recipients to the systems server.
7. For each recipient the server checks if it is a valid recipient.
8. In case all the recipients are valid:
    8.1. The system's database is updated with all recipients and e-mail data.
9. The server returns a list of valid recipients to the "Middle-ware".
10. The server returns a list of invalid recipients to the "Middle-ware".
11. The server returns a list of additional advised recipients to the "Middle-ware" (optionally).
12. In case all the recipients are valid:
    12.1. The "Middle-ware" writes the e-mail data to the appropriate log files.
13. The "Middle-ware" sends to the plug-in a list of valid recipients of the e-mail.
14. The "Middle-ware" sends to the plug-in a list of invalid recipients of the e-mail.
15. The "Middle-ware" sends to the plug-in a list of additional advised recipients of the e-mail (optionally).
16. In case all the recipients are valid:
    16.1. The plug-in sends the e-mail to all the valid recipients.
    16.2. End use case.

17. The plug-in displays to the user all the invalid recipients of the mail.
18. The plug-in displays to the user all the additional advised recipients of the e-mail (optionally).

Alternative flows:

7.a **Case:** The server is loaded.

Actions: 7.a.1 The server sends to the "Middle-ware" all the data needed to decide which recipients are valid and which are not.

7.a.2 For each recipient the "Middle-ware" checks if it is a valid recipient of the e-mail.

7.a.3 In case all the recipients are valid: The "Middle-ware" sends the valid recipients to the system's server.

7.a.4 For each valid recipient the database of the server is updated.

7.a.5 Continue to step 12.

6.a, 9.a, 10.a, 11.a, 12.a **Case**: There are network problems.

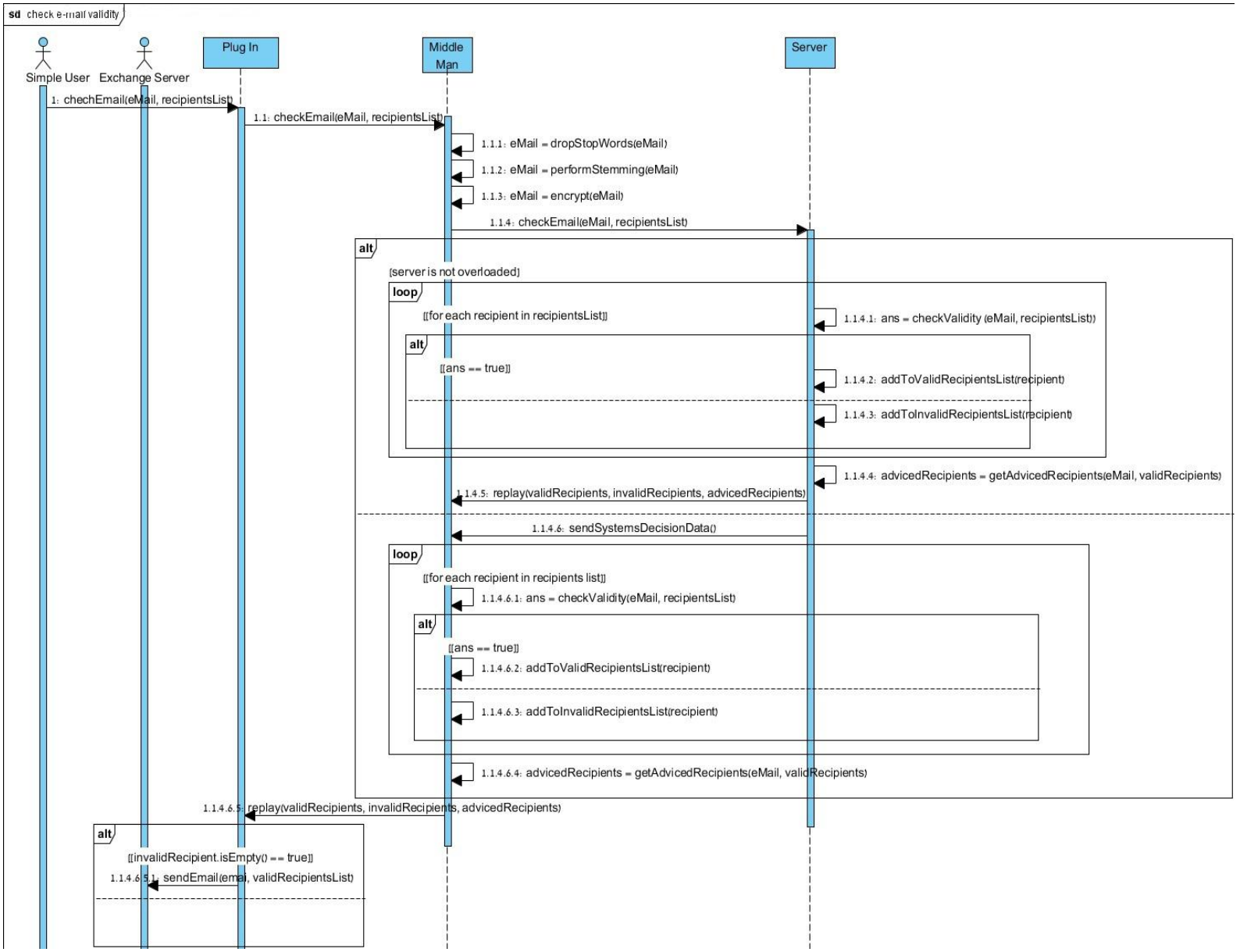
Actions: 6/9/10/11/12.a.1 The "Middle-ware" gets an error message.

6/9/10/11/12.a.2 The "Middle-ware" sends an error message to the plug-in

6/9/10/11/12.a.3 The plug in displays the error message to the user.

Covered Requirements: 1, 2, 3, 4, 5, 7, 8, 9, 10, 11, 12, 13, 15, 16, 17.

Sequence Diagram:



4.2.2 UC 2 – Send an e-mail

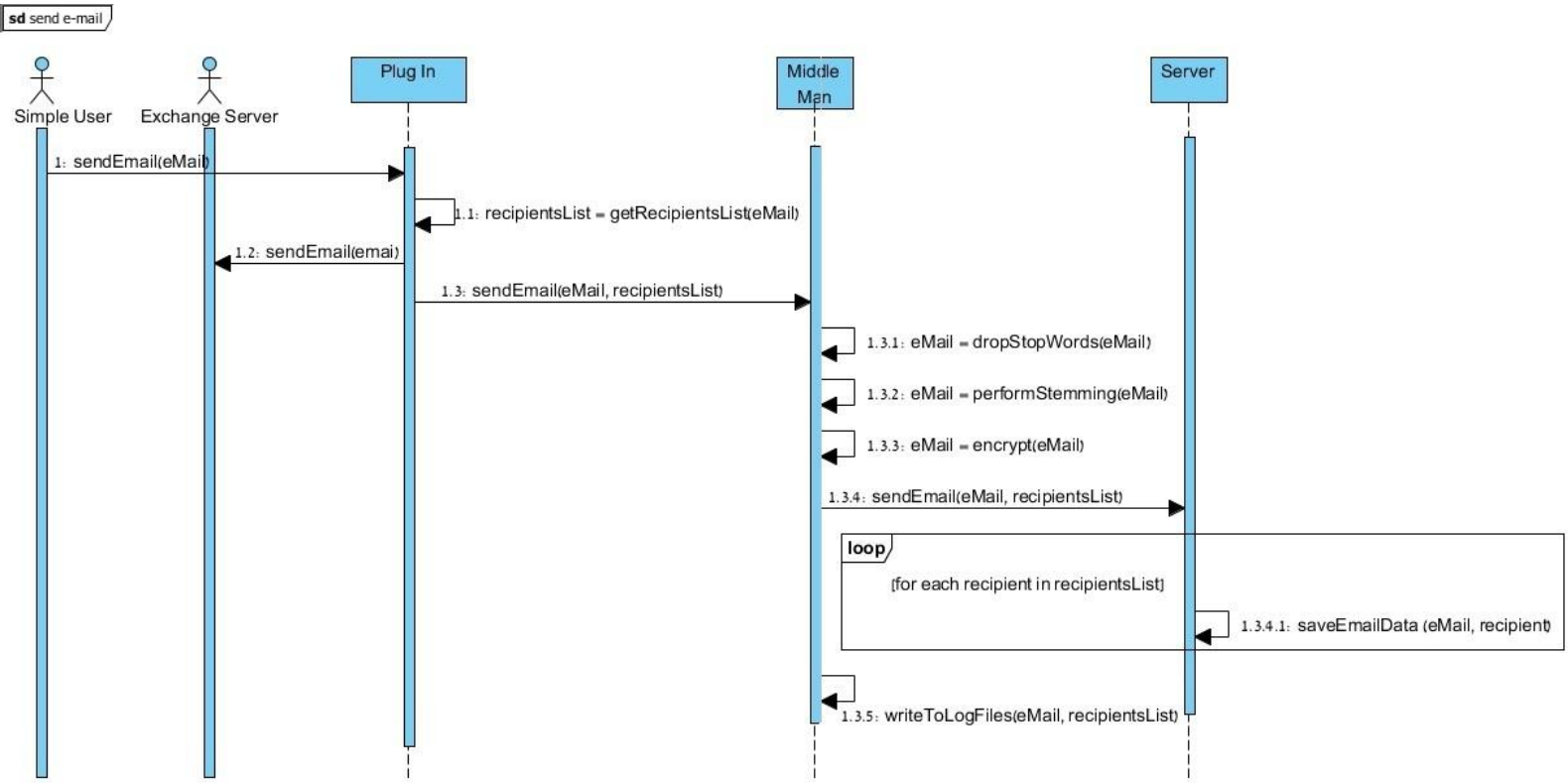
ID: 2
Primary actors: Simple user, Exchange server
Description: The simple user sends the e-mail to all valid recipients, maybe to some of the users considered by the system as invalid, and maybe to some of the advised users.
Trigger: The user clicks the "send button (at the second time).
Pre-conditions:
1. The plug-in is installed on the users e-mail account.
2. The "Middle-ware" is installed on the user's computer.
3. The server of the system is installed and runs on a server computer.
4. The network is working correctly.
5. The users monitor shows him a list of users which the system advices not to send them the e-mail.
6. The user's monitor shows him a list of users which the system advices to send them the e-mail.
Post-conditions:
1. The e-mail was sent to all the valid recipients of the e-mail.
2. The e-mail was sent to all the recipients the user has chosen despite the system recommendations.
3. The e-mail was sent to all the recipients the user has chosen accordingly to the system recommendations (optionally).
4. All the relevant information was written to the appropriate log files.
5. The system's database was updated (data of the recipients chosen despite the system recommendations is marked with "?").
Flow of events:
1. The user choses the list of recipients to whom he wants to send the e-mail despite of the systems recommendation.
2. The user choses the list of recipients to whom he wants to send the e-mail accordingly to the system's recommendation (optionally).
3. The user hits the "send" button.
4. The e-mail is sent to all the valid and chosen recipients.
5. The plug-in sends all the relevant data to the "Middle-ware".
6. The "Middle-ware" writes the e-mail data to the appropriate log files.
7. The "Middle-ware" performs stemming of all the words which are not stop words.
8. The "Middle-ware" encrypts all the stemmed words.
9. The "Middle-ware" sends the encrypted words and the e-mail recipients to the system's server.
10. For each of the recipients the server's database is updated (in case the recipient was previously considered as invalid this data is marked with "?").
Alternative flows:
3.a **Case**: The user hits the "cancel" button.
Actions: 3.a.1 The lists of invalid and recommended e-mail recipients are closed.
3.a.2 Go to use case 1.
2.a **Case**: No recipients were chosen.
Actions: 2.a.1 Display an error message to the user.
2.a.2 Continue to step 1.
4.a, 6.a, 9.a **Case**: There are network problems.
Actions: 4/6/9.a.1 The "Middle-ware" gets an error message.

4/6/9.a.2 The "Middle-ware" sends the error message to the plug-in.

4/6/9.a.3 The plug in displays the error message to the user.

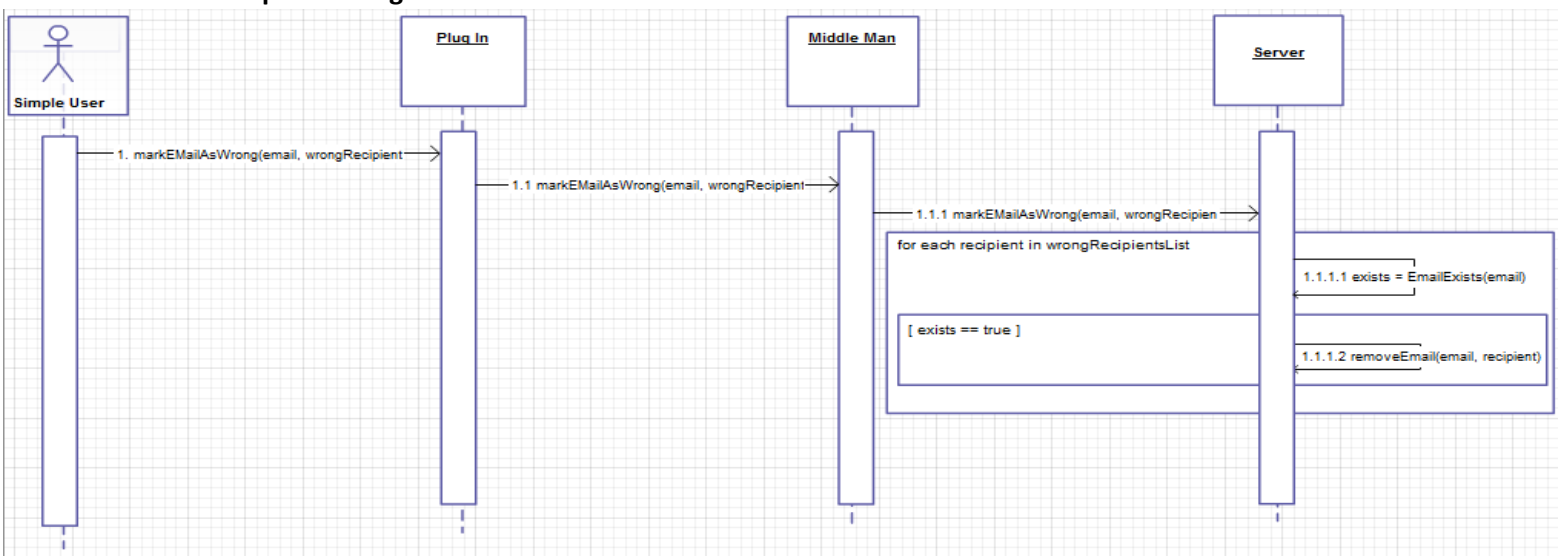
Covered Requirements: 1, 2, 3, 5, 7, 8, 9, 12, 13, 18.

Sequence Diagram:



4.2.3 UC 3 – Mark an e-mail as got / sent by mistake

ID: 3
Primary actors: E-mail client, Simple user.
Description: The simple user sends the e-mail accordingly to recommendations of the system.
Trigger: The user marks the e-mail as sent / got by mistake or the user hits the "remove data" button.
Pre-conditions:
1. The plug-in is installed on the user's e-mail account.
2. The "Middle-ware" is installed on the user's computer.
3. The server of the system is installed and runs on a server computer.
4. The network is working correctly.
Post-conditions:
1. All the chosen e-mail's data was deleted from the database.
Flow of events:
1. The user marks an e-mail as sent / got by mistake.
2. The user chooses the recipients who got the e-mail by mistake (only in case the mail was sent by mistake).
3. The user hits the "remove data" button.
4. The plug-in sends all the relevant data to the "Middle-ware" (only in case the mail was sent by mistake).
5. The "Middle-ware" sends all the relevant data to the system's server.
6. The e-mails' data is removed from the system's data base.
7. The server sends a success message to the "Middle-ware".
8. The "Middle-ware" sends a success message to the plug-in.
9. A success message is displayed to the user.
Alternative flows:
3.a **Case**: The user hits the cancel button.
Actions: 3.a.1 The list of recipients e-mails is closed.
3.a.2 The e-mail is unmarked from being got / sent by mistake (end use case).
3.b **Case**: no recipients were chosen.
Actions: 3.b.1 Display an error message to the user.
3.b.2 Continue to step 2.
5.a, 7.a **Case**: There are network problems.
Actions: 5/7.a.1 The "Middle-ware" gets an error message.
5/7.a.2 The "Middle-ware" sends the error message to the plug-in.
5/7.a.3 The plug in displays the error message to the user.
Covered Requirements: 1, 2, 3, 5, 6, 9, 12, 21.
Sequence Diagram:



4.2.4 UC 4 – Log in to the system as the system administrator

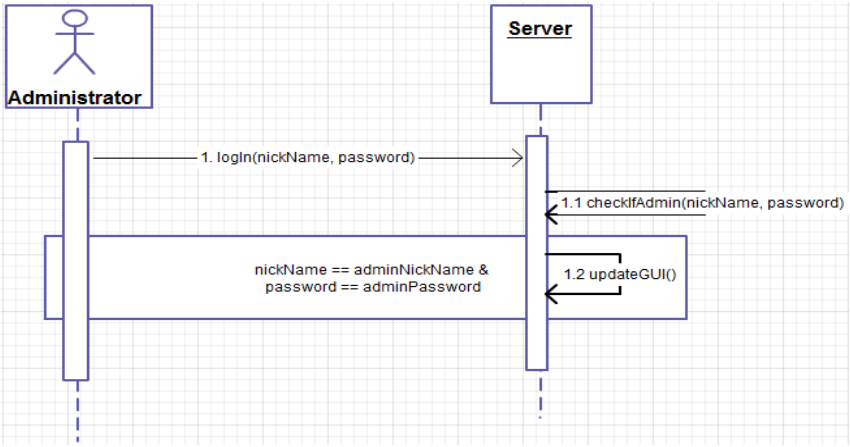
ID: 4
Primary actors: Administrator.
Description: The Administrator logs in to the system.
Trigger: The Administrator clicks the "log in" button.
Pre-conditions:
1. Administrator's module which interfaces with the system is installed and is running.
2. The administrator is logged in to this module as the system administrator.
3. The server of the system is installed and runs on a server computer.
4. The network is working correctly.
5. The "log in" window is displayed.

Post-conditions:
1. The administrator is logged in to the system.
2. The "admin functionalities" window is shown.

Flow of events:
1. The administrator enters his username and password.
2. The administrator hits the log in button.
3. The system shows the administrator the "admin functionalities" window.

Alternative flows:
2.a **Case**: The user entered wrong username or password.
Actions: 2.a.1 The system displays a proper error message.
2.a.2 Continue to step 1.
2.b **Case**: The user didn't enter username or password.
Actions: 2.b.1 The systems displays a proper error message.
2.b.2 Continue to step 1.
2.c **Case**: There are network problems.
Actions: 2.c.1 The systems displays a proper error message.
2.c.2 Continue to step 1.

Covered Requirements: 23, 24.
Sequence Diagram:



4.2.5 UC 5 – Set system's configurations

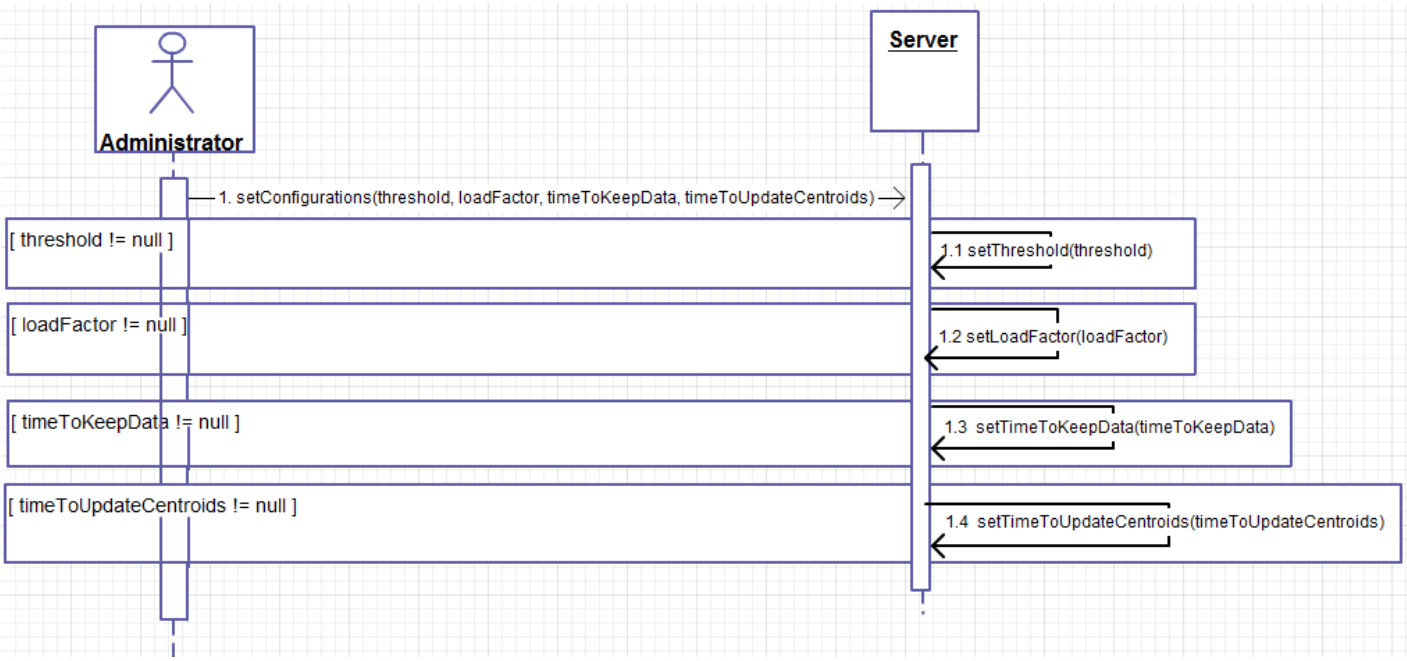
ID: 5
Primary actors: Administrator.
Description: The Administrator sets system's configurations.
Trigger: The Administrator clicks the set configurations button.
Pre-conditions:
1. Administrator's module which interfaces with the systems is installed and running.
2. The administrator is logged in to this module as the system administrator.
3. The server of the system is installed and runs on a server computer.
4. The network is working correctly.
5. The "admin functionalities" window is displayed.

Post-conditions:
1. All the system's configurations are set.
2. The "admin functionalities" window is shown.

Flow of events:
1. The administrator choses the "set configurations" option.
2. The system displays all the current configurations.
3. The administrator enters the load factor of the system.
4. The administrator enters the threshold of the system.
5. The administrator enters the amount of months e-mails data should be kept in the system.
6. The administrator enters the amount of days to pass between centroid vectors updates.
7. The administrator clicks the set configurations button.
8. The system displays a success message.

Alternative flows:
7.a **Case**: No load factor was entered.
Actions: 7.a.1 The systems displays a proper error message.
7.a.2 Continue to step 2.
7.b **Case**: No threshold was entered.
Actions: 7.b.1 The systems displays a proper error message.
7.b.2 Continue to step 3.
7.c **Case**: No amount of months mails data should be kept in the system was entered.
Actions: 7.c.1 The systems displays a proper error message.
7.c.2 Continue to step 4.
7.d **Case**: The administrator hits the cancel button.
Actions: 7.d.1 The systems displays the "admin functionalities" window (end use case).
7.e **Case**: There are network problems.
Actions: 7.e.1 The systems displays a proper error message.
7.e.2 The systems displays the "admin functionalities" window (end use case).

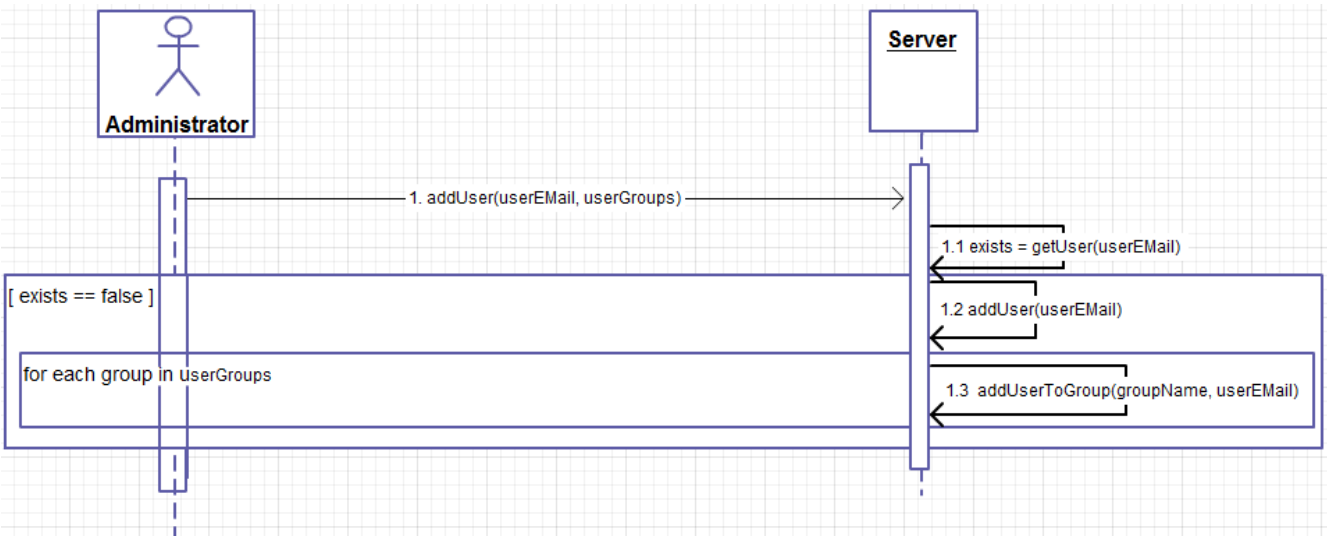
Covered Requirements: 26.
Sequence Diagram: (at the next page).


Administrator
Server
1. setConfigurations(threshold, loadFactor, timeToKeepData, timeToUpdateCentroids)
[threshold != null]
1.1 setThreshold(threshold)
[loadFactor != null]
1.2 setLoadFactor(loadFactor)
[timeToKeepData != null]
1.3 setTimeToKeepData(timeToKeepData)
[timeToUpdateCentroids != null]
1.4 setTimeToUpdateCentroids(timeToUpdateCentroids)

4.2.6 UC 6 – Add new user

ID: 6
Primary actors: Administrator.
Description: The Administrator adds a new user to the system.
Trigger: The Administrator clicks the add user button.
Pre-conditions:
1. Administrator's module which interfaces with the system is installed and running.
2. The administrator is logged in to this module as the system administrator.
3. The server of the system is installed and runs on a server computer.
4. The network is working correctly.
5. The "admin functionalities" window is displayed.
Post-conditions:
1. New user was added to the system.
2. The new user exists in all its subject groups.
3. The "admin functionalities" window is shown.
Flow of events:
1. The administrator choses the "add new user" option.
2. The system displays all the users that exist in the system.
3. The system displays all the subject groups that exist in the system.
4. The administrator enters the e-mail of the new user to be added.
5. The administrator choses all the subject groups which the new user will be included in.
6. The administrator clicks the add button.
7. The system displays a success message.
Alternative flows:
5.a **Case**: One or more of the new users subject groups is not included in the list of subject groups that exists in the system
Actions: 5.a.1 For each such a subject group.
5.a.1.1 Perform use case number 10.
5.a.2 continue to step 6.
6.a **Case**: No user e-mail was entered.
Actions: 6.a.1 The system displays a proper error message.
6.a.2 Continue to step 4.
6.b **Case**: The entered user e-mail already exists in the system.
Actions: 6.b.1 The system displays a proper error message.
6.b.2 Continue to step 4.
6.c **Case**: The administrator didn't choose any subject group for the new user.
Actions: 6.c.1 The system displays a proper error message.
6.c.2 Continue to step 5.
6.d **Case**: The administrator hits the cancel button.
Actions: 6.d.1 The system displays the "admin functionalities" window (end use case).
6.e **Case**: There are network problems.
Actions: 6.e.1 The system displays a proper error message.
6.e.2 The system displays the "admin functionalities" window (end use case).
Covered Requirements: 27.

Sequence Diagram:



4.2.7 UC 7 – Remove existing user

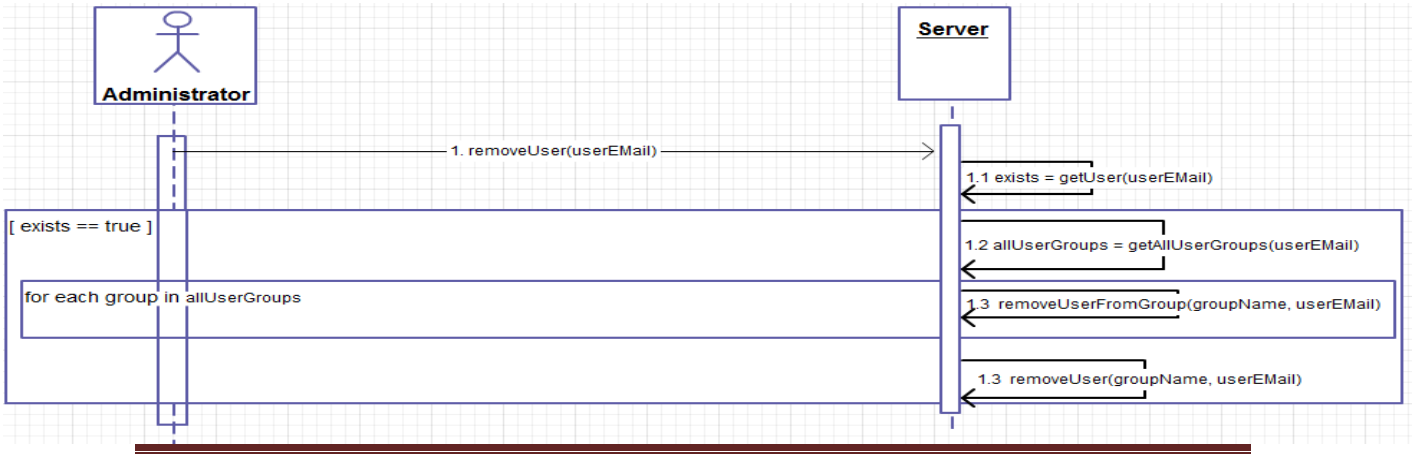
ID: 7
Primary actors: Administrator.
Description: The Administrator removes existing user from the system
Trigger: The Administrator clicks the "remove user" button.
Pre-conditions:
1. Administrator's module which interfaces with the system is installed and running.
2. The administrator is logged in to this module as the system administrator.
3. The server of the system is installed and runs on a server computer.
4. The network is working correctly.
5. The "admin functionalities" window is displayed.

Post-conditions:
1. A user was removed from the system.
2. All the removed user's previous subject groups don't include his e-mail.
3. The "admin functionalities" window is shown.

Flow of events:
1. The administrator choses the "remove user" option.
2. The system displays all the users that exist in the system.
3. The administrator selects the e-mail of the new user to be removed.
4. The administrator clicks the remove button.
5. The system removes the user from all of its' groups.
6. The system deletes the user.
7. The system displays a success message.

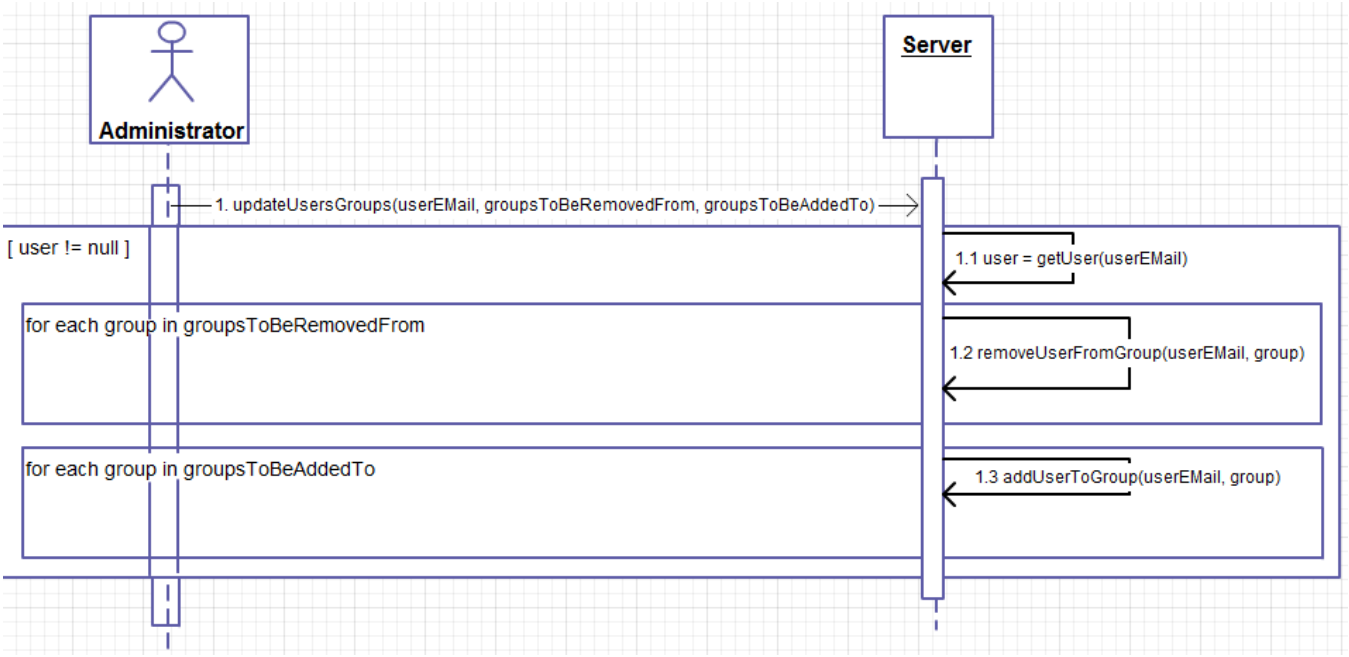
Alternative flows:
4.a **Case**: No user e-mail was selected.
Actions: 4.a.1 The system displays a proper error message.
4.a.2 Continue to step 3.
4.b **Case**: The administrator hits the cancel button.
Actions: 4.b.1 The system displays the "admin functionalities" window (end use case).
4.c **Case**: There are network problems.
Actions: 4.c.1 The system displays a proper error message.
4.c.2 The system-displays the "admin functionalities" window (end use case).

Covered Requirements: 28.
Sequence Diagram:



4.2.8 UC 8 – Update user's subject groups (user's cluster)

ID: 8
Primary actors: Administrator.
Description: The Administrator updates user's subjects groups.
Trigger: The Administrator clicks the "update user's subject groups" button.
Pre-conditions:
1. Administrator's module which interfaces with the system is installed and running.
2. The administrator is logged in to this module as the system administrator.
3. The server of the system is installed and runs on a server computer.
4. The network is working correctly.
5. The "admin functionalities" window is displayed.
Post-conditions:
1. The user exists in all its subject groups.
2. The "admin functionalities" window is shown.
Flow of events:
1. The administrator chooses the "updates user's subjects groups" option.
2. The system displays all the users that exist in the system.
3. The administrator selects the e-mail of the wanted user.
4. The system displays a list of all the subject groups of the wanted user.
5. The system displays a list of all the subject groups that the wanted users can be added to.
6. The administrator choses all the subject groups which the user will be removed from.
7. The administrator choses all the subject groups which the user will be added to.
8. The administrator clicks the update button.
9. The system displays a success message.
Alternative flows:
7.a **Case**: One or more of the user's new subject groups is not included in the list of subject groups that exists in the system
Actions: 7.a.1 For each such a subject group.
7.a.1.1 Perform use case number 9.
7.a.2 continue to step 8.
8.a **Case**: No user e-mail was chosen.
Actions: 8.a.1 The system displays a proper error message.
8.a.2 Continue to step 3.
8.b **Case**: The administrator hits the cancel button.
Actions: 8.b.1 The system displays the "admin functionalities" window (end use case).
8.c **Case**: There are network problems.
Actions: 8.c.1 The system displays a proper error message.
8.c.2 The system displays the "admin functionalities" window (end use case).
Covered Requirements: 29.
Sequence Diagram: (at the next page).


Administrator
Server
1. updateUsersGroups(userEMail, groupsToBeRemovedFrom, groupsToBeAddedTo)
[user != null]
1.1 user = getUser(userEMail)
for each group in groupsToBeRemovedFrom
1.2 removeUserFromGroup(userEMail, group)
for each group in groupsToBeAddedTo
1.3 addUserToGroup(userEMail, group)

4.2.9 UC 9 – Add new subject group

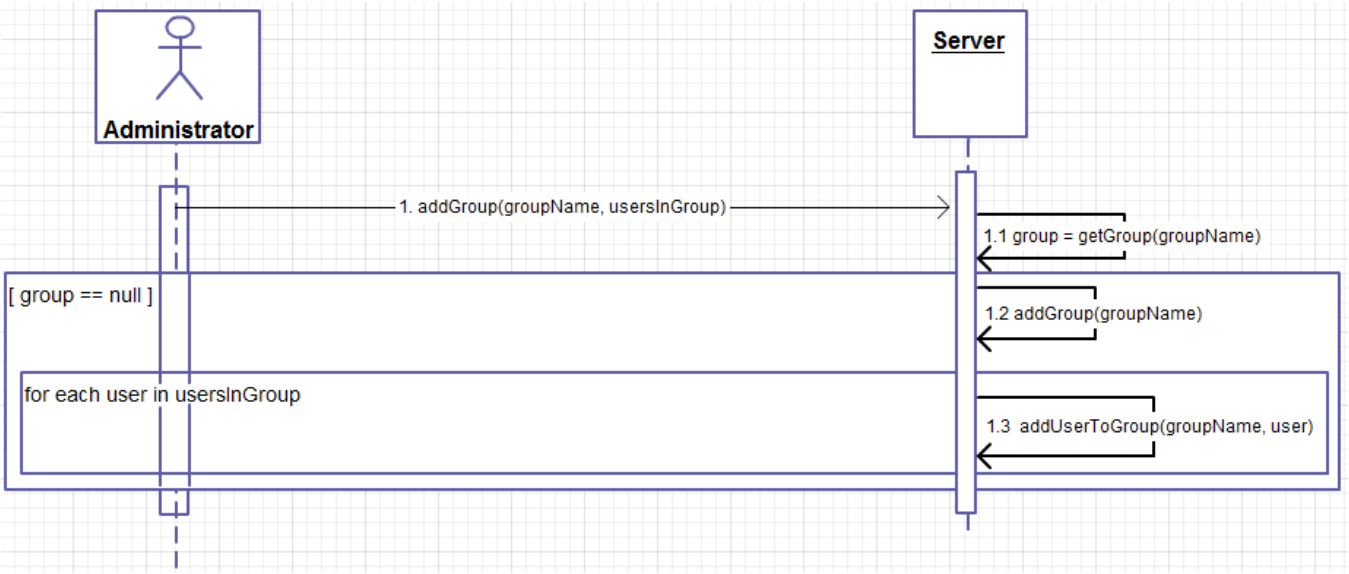
ID: 9
Primary actors: Administrator.
Description: The Administrator adds a new subject group to the system.
Trigger: The Administrator clicks the add subject group button.
Pre-conditions:
1. Administrator's module which interfaces with the system is installed and is running,
2. The administrator is logged in to this module as the system administrator.
3. The server of the system is installed and runs on a server computer.
4. The network is working correctly.
5. The "admin functionalities" window is displayed.

Post-conditions:
1. New subject group was added to the system,
2. The new subject group contains all the e-mails of its all members.
3. The "admin functionalities" window is shown.

Flow of events:
1. The administrator choses the "add new subject group" option.
2. The system displays all the subject groups that exist in the system.
3. The system displays all the users that exist in the system.
4. The administrator enters the new subject to be added.
5. The administrator chooses all the users that will be in the new subject group.
6. The administrator clicks the add subject button.
7. The system displays a success message.

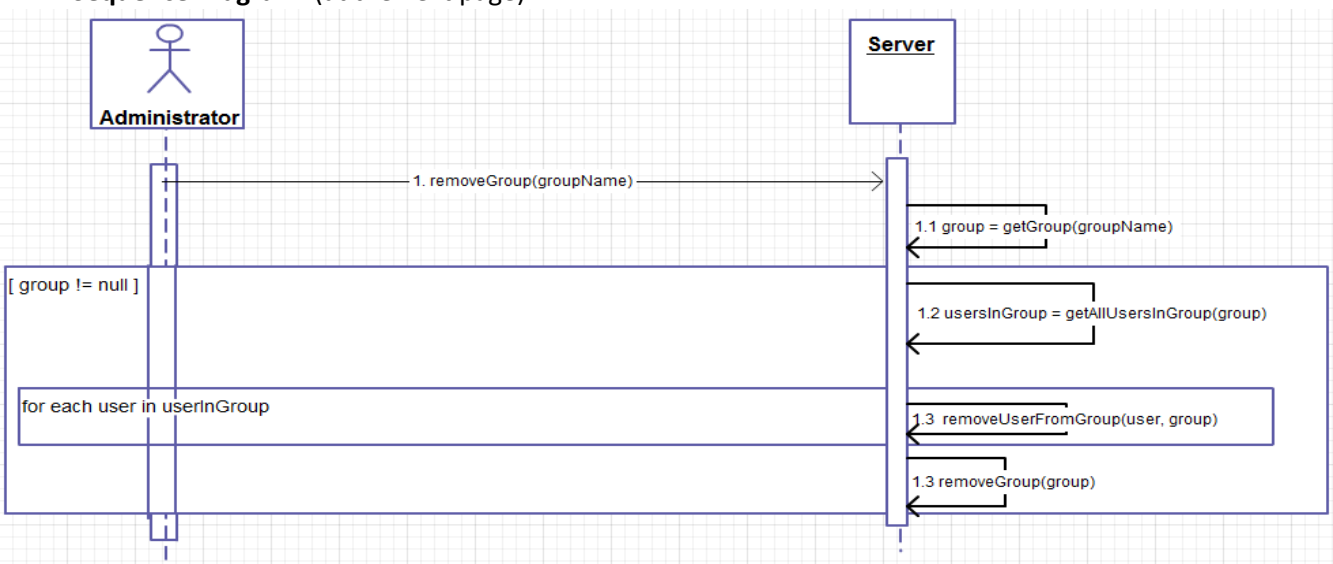
Alternative flows:
6.a **Case**: No subject was entered.
Actions: 6.a.1 The system displays a proper error message.
6.a.2 Continue to step 4.
6.b **Case**: The entered subject already exists in the system.
Actions: 6.b.1 The system displays a proper error message.
6.b.2 Continue to step 4.
6.c **Case**: The administrator didn't choose a subject group for the user.
Actions: 6.c.1 The system displays a proper error message.
6.c.2 Continue to step 5.
6.d **Case**: The administrator hits the cancel button.
Actions: 6.d.1 The system displays the "admin functionalities" window (end use case).
6.e **Case**: There are network problems.
Actions: 6.e.1 The system displays a proper error message.
6.e.2 The system displays the "admin functionalities" window (end use case).

Covered Requirements: 30.
Sequence Diagram: (at the next page).


Administrator
Server
1. addGroup(groupName, usersInGroup)
1.1 group = getGroup(groupName)
[group == null]
1.2 addGroup(groupName)
for each user in usersInGroup
1.3 addUserToGroup(groupName, user)

4.2.10 UC 10 – Remove existing subject group

ID: 10
Primary actors: Administrator.
Description: The Administrator removes existing subject group from the system
Trigger: The Administrator clicks the "remove subject group" button.
Pre-conditions:
1. Administrator's module which interfaces with the system is installed and is running.
2. The administrator is logged in to this module as the system administrator.
3. The server of the system is installed and runs on a server computer.
4. The network is working correctly.
5. The "admin functionalities" window is displayed.
Post-conditions:
1. A subject group was removed from the system.
2. The "admin functionalities" window is shown.
Flow of events:
1. The administrator chooses the "remove subject group" option.
2. The system displays all the subject groups that exist in the system.
3. The administrator selects the subject group to be removed.
4. The administrator clicks the remove button.
5. The system displays a success message.
Alternative flows:
4.a **Case**: No subject group was selected.
Actions: 4.a.1 The system displays a proper error message.
4.a.2 Continue to step 3.
4.b **Case**: The administrator hits the cancel button.
Actions: 4.b.1 The system displays the "admin functionalities" window (end use case).
4.c **Case**: There are network problems.
Actions: 4.c.1 The system displays a proper error message.
4.c.2 The system displays the "admin functionalities" window (end use case).
Covered Requirements: 31.
Sequence Diagram: (at the next page).



4.2.11 UC 11 – Deal with mails marked by a question mark

ID: 11
Primary actors: Administrator.
Description: The Administrator removes a "?" from an existing e-mail.
Trigger: The Administrator clicks the "deal with question marked e-mails" button.
Pre-conditions:
1. Administrator's module which interfaces with the systems is installed and is running.
2. The administrator is logged in to this module as the system administrator.
3. The server of the system is installed and runs on a server computer.
4. The network is working correctly.
5. The "admin functionalities" window is displayed.
Post-conditions:
1. A question mark was removed from all proper e-mails in the system's database.
2. All proper e-mails were removed from the log file.
3. All improper e-mails are marked in the log file as improper e-mails.
4. All improper e-mails were removed from the system's database.
5. The "admin functionalities" window is shown.
Flow of events:
1. The administrator chooses the "deal with e-mails sent in contradiction to the system's recommendation" option.
2. The system displays the list of e-mails that were sent in contradiction to the system's recommendation.
3. The administrator choses all the proper e-mails.
4. The administrator clicks the update proper e-mails button.
5. The system clears all the "?" marks from the improper e-mails in the database.
6. The system deletes all the information about the selected e-mails from the log file.
7. The system displays a success message.
8. The system displays the updated list of e-mails that were sent in contradiction to the system's recommendation.
9. The administrator choses all the improper e-mails.
10. The administrator clicks the update improper e-mails button.
11. The system clears all the improper e-mails from the database.
12. The system changes the information about the selected e-mails in the log file (to be definitely improper).
13. The system displays a success message.
Alternative flows:
4.a **Case**: No e-mail was chosen.
Actions: 4.a.1 The system displays a proper error message.
4.a.2 Continue to step 3 .
10.a **Case**: No mail was chosen.
Actions: 10.a.1 The system displays a proper error message.
10.a.2 Continue to step 4.
13.a **Case**: The administrator hits the cancel button.
Actions: 13.a.1 The system displays the "admin functionalities" window (end use case).

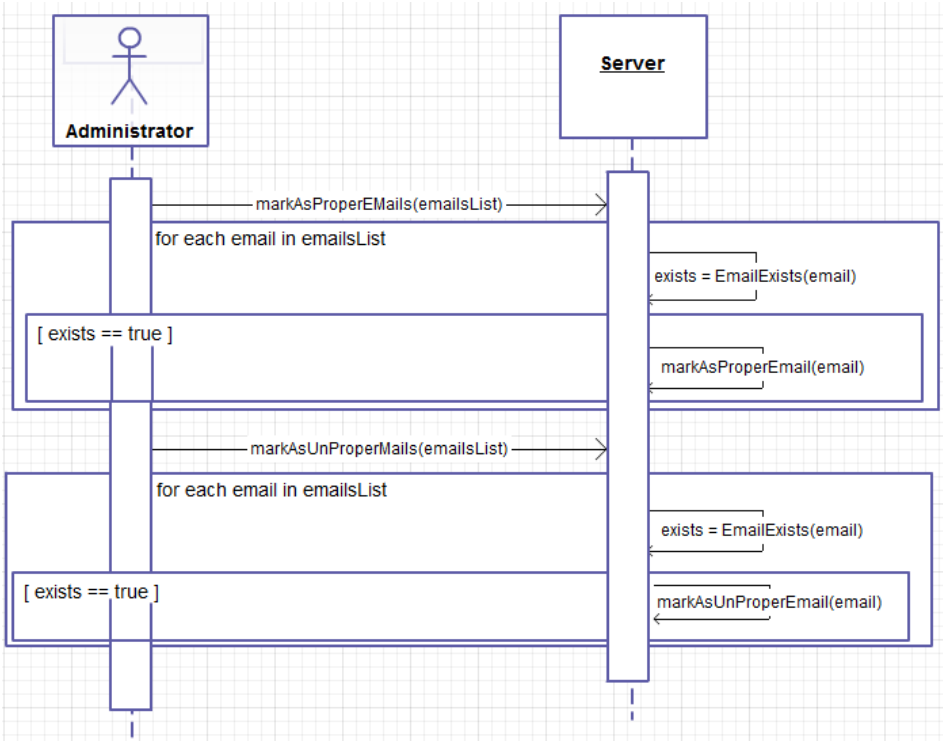
4.b, 10.b **Case**: There are network problems.

Actions: 4/10.b.1 The system displays a proper error message.

4/10.b.2 The system displays the "admin functionalities" window (end use case).

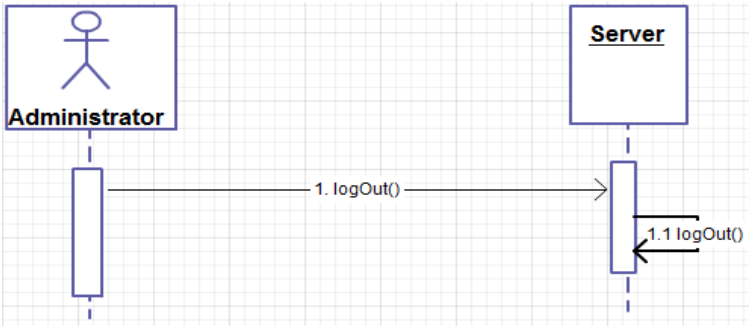
Covered Requirements: 32, 33.

Sequence Diagram:



4.2.12 UC 12 – Log out from the administrator mode

ID: 12
Primary actors: Administrator.
Description: The Administrator logs out form the administrator mode.
Trigger: The Administrator clicks the "log out" button.
Pre-conditions:
1. Administrator's module which interfaces with the system is installed and is running.
2. The administrator is logged in to this module as the system administrator.
3. The server of the system is installed and runs on a server computer.
4. The network is working correctly.
5. The "admin functionalities" window is displayed.
Post-conditions:
1. The administrator is logged out from this module.
2. Administrators' mode is closed.
Flow of events:
1. The administrator chooses the "log out" option.
2. The system displays an approval message.
3. The system logs' the administrator out.
4. The system closes the administrator mode.
Alternative flows:
2.a **Case**: The administrator hits the "no" button.
Actions: 2.b.1 The system displays the "admin functionalities" window (end use case).
3.a **Case**: There are network problems.
Actions: 3.a.1 The system displays a proper error message.
3.a.2 The system displays the "admin functionalities" window (end use case).
Covered Requirements: 25.
Sequence Diagram:



4.3 Special Usage Considerations

Not applicable.

5 Appendices

5.1 Catching the "send e-mail" event

All events of send e-mail that occur in outlook should be "caught" by our system, temporary cancelled, and maybe activated after e-mail's data will be checked by our system.

We thought about some possible ways to catch the send e-mail event:

1. To catch this event in the operation system level – however this solution is not so good, because in this case all the send e-mail events from all the person's e-mails (including his\her personal e-mail) will be caught. It is very difficult to handle such a volume of events because for each event we will have to check from which e-mail client it came, and to cancel it only in the case it came from the outlook e-mail client (we don't even know if there is a way to perform such a check). Moreover hooking the "send" button in the operation system level will cause us to catch all the events tied to send button (and not only the event of "send e-mail"), so we will have to check manually for each caught event that is not the "send e-mail" event and to ignore it.
2. To add a "check validity" button. After this button will be pressed our system will perform a validity check for the e-mail. The user will have to press this button before he\she presses the "send" button. However this solution doesn't promise the validity check will be performed before the e-mail is sent (because the user can simply hit the "send" button without hitting the "check validity" button before.
3. To override the outlook "send" button by our "send" button (code will be written by us). This solution is not elegant. Moreover not careful usage of it can it can potentially cause to security problems.
4. To catch this event in the outlook level. The problem we see with this solution that not careful usage of it can potentially cause to security problems.

We decided to check at first the option to catch the send e-mail event in the outlook level because it looks like the most elegant and proper solution.

5.2 Glossary – meaning of professional terms used in this document

Leak	A mail that is being sent to the wrong recipient
centroid of a link	The *TF-IDF* vector representation of the link, which will be further denoted as *TF-IDF*(*link*). The centroid represents a typical email associated with the link.
Link	Every two users that have exchanged emails in the past define a link,
User	A person who uses the system
System administrator	A person who has the ability to change system's configurations
E-mail	A method of exchanging digital messages across the Internet or other computer networks
E-mail client	A computer program used to manage a user's email.
Outlook e-mail client	A personal e-mail manager from Microsoft, available both as a separate application as well as a part of the Microsoft Office suite
Outlook plug-ins mechanism	A mechanism of adding specific capabilities to outlook
Data leakage	A data that is sent by mail to a wrong recipient
Valid e-mail recipient	A recipient whose e-mail receive is not considered to be a leak
Invalid e-mail recipient	A recipient whose e-mail receive is considered to be a leak
Recommended e-mail recipient	A recipients that our system recommends to add him to the list of e-mail recipients
Plug-in	A set of software components that adds specific capabilities to a larger software application.
Plug-in for outlook e-mail client	A software component for the outlook e-mail client
Topic group	A group of users that can send each other e-mails on certain topics
Classification	Deciding for each a-mail recipient whether he is a valid recipient or not
Partners of an organization	People from other organizations that cooperate with people from the organization
Members of the organization	People who work in the organization
Exchange server	The server side of a client–server, collaborative application product
Validity check	A check whether some recipient is a valid e-mail recipient or not
Threshold	The bound which separates between a recipients to be valid or invalid
Database	An organized collection of data
Stop words	Words which are filtered out prior to, or after, processing of natural language data (text)
Stemming	The process for reducing inflected (or sometimes derived) words to their stem, base or root form
Encryption	The process of transforming information (referred to as plaintext) using an algorithm to make it unreadable to anyone except those possessing special knowledge, usually referred to as a key.

Overload (server)	A situation on which the server has more clients asking for service than the server can serve in efficient way
Logger (log file)	A file containing all the actions done in the system
Server	• a computer program running as a service, to serve the needs or requests of other programs (referred to in this context as "clients") which may or may not be running on the same computer. • a physical computer dedicated to running one or more such services, to serve the needs of programs running on other computers on the same network. • a software/hardware system (i.e. a software service running on a dedicated computer) such as a database server, file server, mail server, or print server.
Client	An application or system that accesses a remote service on another computer system, known as a server, by way of a network
"?" e-mails data mark	When the user choses to send an e-mail to some person also it is considered to be a leak the data of this e-mail will be marked in our database by the "?" mark.
Back-up	Making copies of data so that these additional copies may be used to *restore* the original
GUI	A graphical user interface (GUI) - is a type of user interface that allows users to interact with programs in a graphical way
Period of time e-mails should be saved in the system,	The amount of time e-mails' data is being saved in our system's database. After this time period mail's data will be deleted
Load factor of the server,	The maximal amount of clients
Amount of days to pass between centroid vectors updates.	The centroid vectors are updated each certain time. This time is given to the system by the administrator through the "set configurations" option.
API	An interface implemented by a software program that enables it to interact with other software. It facilitates interaction between different software programs similar to the way the user interface facilitates interaction between humans and computers
COM	A binary-interface standard for software componentry. It is used to enable interprocess communication and dynamic object creation in a large range of programming languages
cluster	reflects the typical email associated with the topic; and,
rollback	Return the system's database to its previous state
AES encryption	The Advanced Encryption Standard (AES) is a symmetric-key encryption standard adopted by the U.S. government.
SSL	Is a cryptographic protocol that provide security for communications over networks

5.3 The theoretical model used by our system

5.3.1 Proposed solution

The proposed classification scheme is based on email exchange traffic among members of the organization (further on members of the organization will be referred to as *users*). The proposed scheme consists of two phases. In the first phase, groups of users that exchange emails with similar content, i.e., common topic, are identified. It is assumed that a user may belong to several groups working on distinct topics (typical for a manager). This phase will be further referred to as „*Group Communication Analysis*".
In the second phase, each new email that is about to be sent is analyzed as follows: For each recipient of the email it is checked whether the recipient and the sender of the email belong to (at least one) common topic group. If such a group does not exist, it may be concluded that there is no common topic for the two users to discuss, and the aforementioned recipient is a wrong recipient. Otherwise, the content of the email is compared to the content of emails exchanged in the group. If the similarity score is high enough, the recipient may be considered as a legal recipient. For example, assume Alice and Bob belong to the same group that communicates topic *T*, and Bob sends an email with content *T* to Alice. Alice won't be considered a wrong recipient, even if Alice and Bob have never exchanged communication with content *T* before (see Figure 1). Group communication analysis provides additional information about potential connections between users who discuss similar topics, but do not necessarily communicate with each other. Thus, it better reflects the "real picture" of topics common to different users, than the sole analysis of the individual user communication with other users.

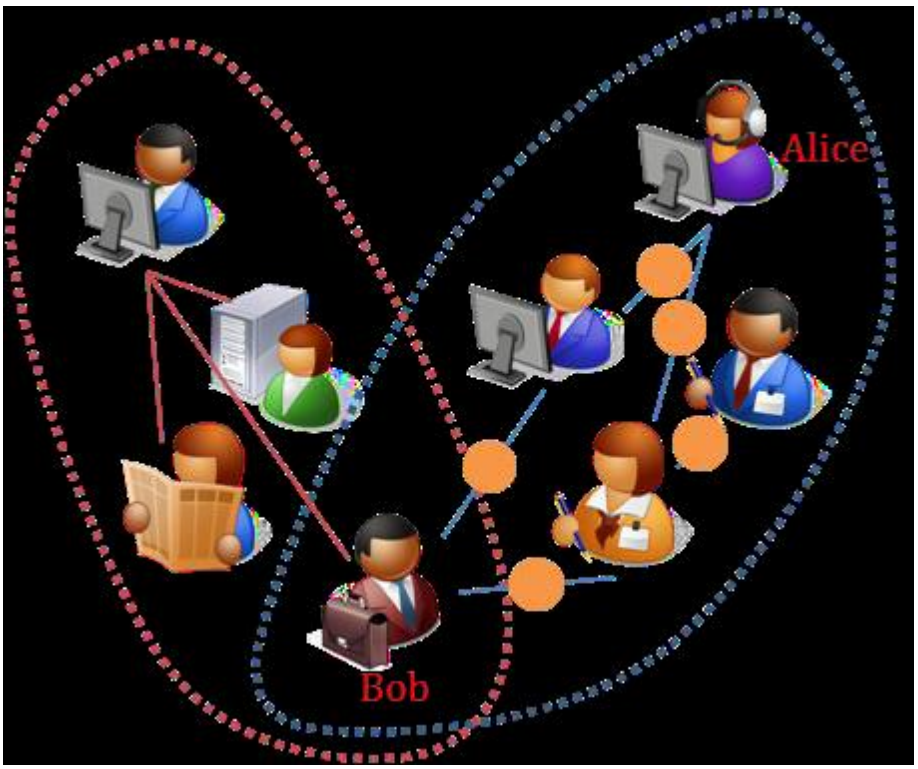


Figure 1: Orange circles represent the emails taken into account when classifying an email sent from Bob to Alice.

5.3.2 Classification model

5.3.2.1 Baseline classification model

Every two users that have exchanged emails in the past define a **link**, and all emails exchanged between these two users are associated with the link. During the training phase we compute: (i) **centroid** of the link – the *TF-IDF* vector representation of the link, which will be further denoted as *TF-IDF*(*link*). The centroid represents a typical email associated with the link. (ii) **threshold similarity score** of the link – determines how similar a new email should be in order to be associated with the link. Setting the threshold score to such that considers all emails associated with the link as "normal", may result in high false-negative classifications (i.e., emails that do not belong to the link, falsely associated with the link since their similarity to *TF-IDF*(*link*) is high enough). The classification phase of an email with content *c* sent from *s* to *r* represented by a query *s*, *r*, *c* is performed as follows: 1. Compute *TF-IDF*(*c*). 2. Retrieve the *TF-IDF*(*link*) of the link defined by the users *s* and *r*. 3. Compare, using Cosine similarity function, *TF-IDF*(*c*) and *TF-IDF*(*link*).

If the received similarity score is lower than the link's threshold similarity score, then sending *c* to recipient *r* is considered a potential leak. If *s* and *r* exchanged no emails in the past, then no corresponding analyzed link exists, and sending any content *c* to recipient *r* is considered a potential leak. Overview of the classification phase:

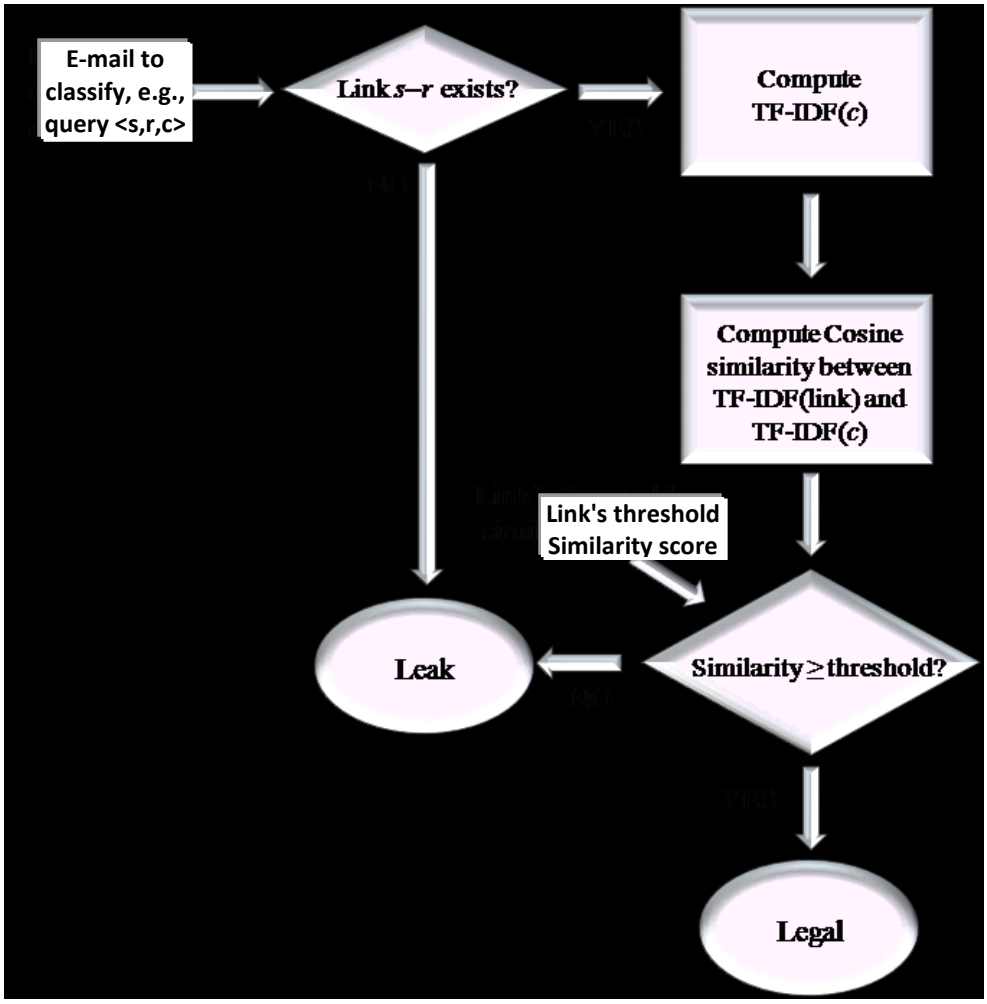


Figure 2: Link based classification phase

5.3.2.2 Proposed classification model

The training phase consists of grouping the users based on the emails they exchange. The members of each group exchange emails with similar content (i.e., topic). This phase is divided into two main sub-phases: (i) **Identifying, via fuzzy clustering, the topics discussed in an organization**. For each identified topic compute: (a) cluster centroid – reflects the typical email associated with the topic; and, (b) threshold similarity score – determines how similar a new email should be in order to be associated with the topic. (ii) **Projection of the different topics on the users**. This process includes counting for each user how many of his/her sent/received emails are associated with each one of the topics. Users that have emails associated with the same topic constitute a group of users with common topic. A user may belong to several groups.

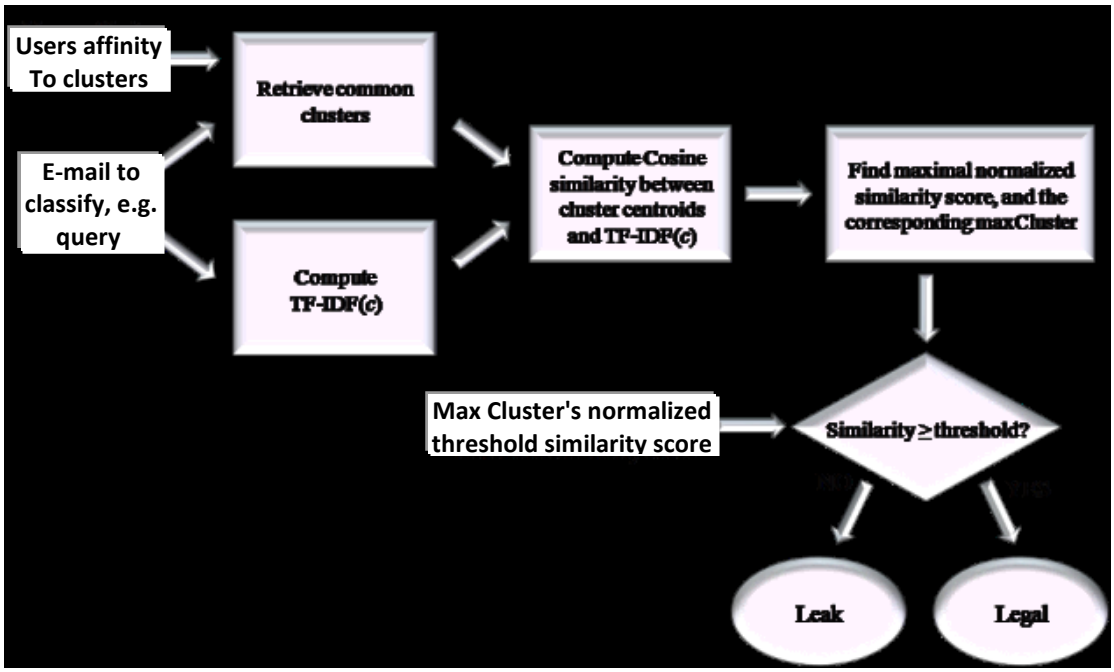


Figure 3: Group based classification phase

5.3.2.3 Cascading the models

Apparently, cascading the group-based and link-based classifiers will take advantage of the "strong" points of both classifiers, and eliminate their "weak" points. A simple cascading is described by a flow chart in Figure 4. In this cascading, a known recipient is classified by the link-based classifier while an unknown recipient is classified by the group-based classifier.

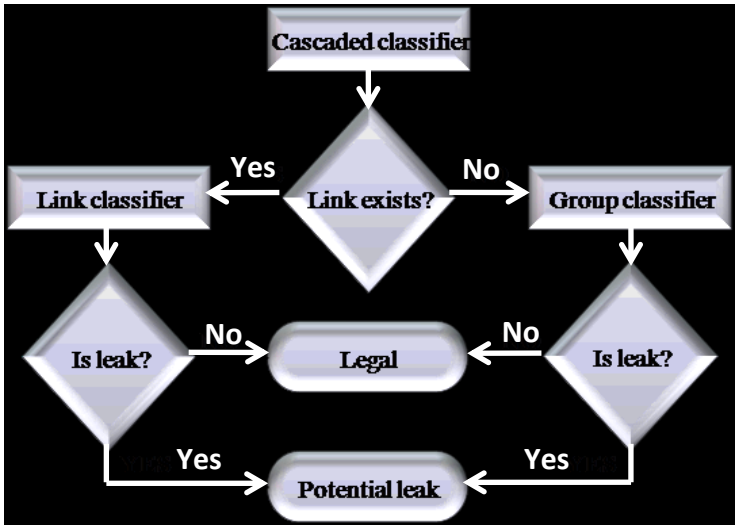


Figure 4: Cascaded classifier flow chart

5.4. Descriptions of similar products

Our review includes advanced commercial products and the latest academic research results.

Commercial products, such as Symantec (http://www.symantec.com/business/products/family.jsp? familyid=data-loss-prevention), Websense (Websense Data Security Solutions, White Paper.),McAfee (http://www.sans.org/reading_room/analysts_program/M cAfee_Total_Protection_Jun09.pdf), RSA, and Vericept (http://www.entrepreneur.com/tradejournals/article/1737 09078.html), aim to prevent sensitive data from leaking via electronic communication channels. Their solutions are embedded at the network level where an email is inspected and a policy can be applied. A policy may define groups of users that are allowed to be exposed to certain contents. Therefore, it can identify recipients that should not receive the email. The commercial products address mainly emails that are sent out of the organization.

Google provides to its users an application named "Got The Wrong Bob" (http://www.paklinks.com/gs/rss-downloads/360830- gmails-got-the-wrong-bob-helps-avoid-misfired-emails.html) that aims to prevent an email from reaching the “Wrong Bob”. To the best of our knowledge Google‟s “Wrong Bob” is based on analyzing the groups of people a user usually exchange emails with, and alerting the user if an unexpected person has been added to the email. The disadvantage of Google's application is that it only works on group emails. “Got the Wrong Bob” won't know if a user has got the wrong Bob when he or she is sending an email to a single recipient**.**

Kalyan and Chandrasekaran (Kalyan C., Chandrasekaran K., “Information Leak Detection in Financial E-mails Using Mail Pattern Analysis under Partial Information”, *Proc. of the 7th Conf.on 7th WSEAS International Conf.on Applied Informatics and Communications*, pp. 104-109, 2007.) propose email pattern analysis to detect data leaks via email. Theikelihood that an email has been sent by mistake is determined by analyzing attributes of emails previously exchanged between the sender and the recipients of the email. The attributes include time, attachment size, salutation and ending, existence of BCC recipients, etc. The proposed technique has been used on real-life emails, with detection accuracy close to 92%.

Carvalho and Cohen (Carvalho V., Cohen W., “Preventing Information Leaks in E-mail”, *SIAM International Conf.on Data Mining 2007*, 2007, http://www.cs.cmu.edu/ wcohen/postscript/sdm-2007-leak.pdf). Predicted whether a sent email is a leak or not based on the textual content of the email, and how likely that the email recipient should receive a particular message. Messages sent to past recipients are modeled into <*message*,*recipient*> pairs, and a <*message*,*recipient*> pair is considered to be a potential leak if the message is sufficiently different from past messages sent to that recipient.

The solution of Carvalho and Cohen proposed two different techniques for leakage detection. The first technique relies strictly on the message's textual content. It measures the similarity between two vector-based representations of email messages. The first vector is a *TF-IDF* [11] representation of all previous messages from the user to the specific recipient (a different vector is created for each recipient). The second vector

is a *TF-IDF* representation of the current message about to be sent. The distance between two vectors is measured using one of two suggested algorithms: *Cosine-Similarity* or *k-Nearest Neighbors* (*KNN*). If the computed similarity is smaller than a predefined threshold, then a warning message is issued to the user who is about to send the message. This comparison is done separately for each recipient of the message about to be sent. The second technique is a classification-based method and has been implemented by using social network information (such as the number of received and sent messages, the number of times two recipients were addressed in the same message, etc.). The idea is to perform the leak prediction in two steps. In the first step, textual similarity scores are calculated using a cross-validation procedure in the training set. In the second step, network features are extracted and then a function is calculated, that combines these features with the textual scores.

The method was able to detect email leaks in almost 82% of the test cases. The advantage of thisapproach is that it can be easily implemented for an email client and it does not use any information that is only available to the server.

In a later study by Carvalho et al. (Carvalho V.R., Balasubramanyan R., "Information Leaks and Suggestions: A Case Study using MozillaThunderbird", *Proc. of 6th conf. on email and anti-spam*, 2009.), the authors present a case study of the solution on the Mozilla Thunderbird. They also expanded the proposed solution not only to detect undesired recipients, but also to suggest recipients that the user has forgotten to address. The new method uses various machine learning and data mining techniques. These techniques study past email exchanges and suggest, according to the learned model, adding or removing a recipient. It has been proposed installing the solution as a plug-in to the Mozilla Thunderbird engine. Stolfo et al. (Stolfo S. J., Hershkop S. Hu C.W., Li W.J., Nimeskern O., Wang K., "Behavior-based modeling and its application to Email analysis", *ACM Trans. Internet Technol.*, Vol. 6, No. 2, pp. 187-221, 2006.) demonstrate how the Email Mining Toolkit (EMT) (Hershkop S., Wang K., Lee W., Nimeskern O., Creamer G., Rowe R., "Email Mining Toolkit Technical Manual", Department of Computer Science, Columbia University New York, NY, June 2006, (http://sneakers.cs.columbia.edu/ids/emt/software/EMTTechManualv 3 6 8.pdf),) can detect the beginning of a viral propagation in emails without content-based or signature-based analysis. EMT is a data mining system that is applied online to email files gathered from email clients or server logs. It retrieves models of user email accounts and of groups of accounts, including the social cliques embedded in the user's email behavior patterns. EMT aggregates statistical information from groups of accounts and provides the means for detecting malicious users.