Republic of Iraq
Ministry of Higher Education and Scientific Research
Al-Nahrain University
College of Science

Personal Assistant Email Agent

A thesis
Submitted to the College of Science, Al-Nahrain University in Partial Fulfillment of the Requirements for the Degree of Master of Science in Computer Science

By
Zaid H. Alibadi
(B.Sc. 2006)

Supervisors
Dr. Loay E. George Dr. Sawsan K. Thamer

January 2011 Sufar 1423

Dedication

To the two women, that they have most impact in my life,

my mother and grandmother.

Zaid

Acknowledgment

I express a special thank to Dr. Loay E. George, beside his valuable guidance and ideas to complete this work, I thank him for the interesting discussions that we had, which included various issues and topics of life. For those hours and ideas that you shared with me, I show my sincere thanks and gratitude.

To Dr. Sawsan K. Thamer, for her valuable guidance and ideas to present this thesis as best as possible, have my sincere gratitude and appreciation.

Grateful thank to both Dr. Haithem A. Al-Ani and Dr. Taha S. Bashaga for helping me in all the administrative matters that accompanied the years of study and research.

Finally, to everyone who helped me and supported me, I say "thank you".

Abstract

Email is one of the most useful communication tools over the Internet; Email can be an effective knowledge management tool which conveniently enables fast and accurate communication. On the other side, the increasing volume of email threatens to cause a state of “email overload” at which the volume of messages exceeds individuals’ capacity to process them.

The recent phenomena of email-overloading in daily life and business have created new problems to users. It becomes a personal headache for users because they have to process a large number of daily received emails. Also, it is a financial issue because the user checks and read for large amount of email messages needs long online communication connection. Therefore, there is a practical need for developing a software assistant to facilitate the management of personal and organizational emails and to enable users to complete their jobs or tasks smoothly.

Personal **A**ssistant **E**mail **A**gent (PAEA) assists the user to send all his/her email messages to their recipients and automatically download the user email message from the email server. The agent is designed to be able to classify the incoming email messages into folders, and to prioritize them so that the user can focus on more important emails first. The agent prioritizes messages according to user profile and his historical reaction. PAEA can instantaneously update his learning from the user behavior to be more effective and adaptive in doing the email sorting task.

PAEA was tested to check if its results are rational and match the user prediction, also, to focus on the power points of the PAEA. The system minimized the internet connection time that needed to only the real active time to upload and download email messages. Also, it minimizes the processing time to manage the user's email messages.

List of Abbreviations

AREMS	Auto-Reply Email Message Service
ASCII	American Standard Code for Information Interchange
Bcc	Blind carbon copy
CR	Carriage Return
Cc	Carbon copy
CM	Contacts Module
DC	Downloader Component
DLL	Dynamic Link Libraries
Email	Electronic mail
EB	Email Browser
EC	Email Composer
EMC	Email Management Component
FEDS	Filter Email Downloader Services
HTML	Hyper Text Markup Language
ICU	Internet Check Unit
IP	Internet Protocol
IMAP	Internet Message Access Protocol
IR	Information Retrieval
IRC	Initialization and Reconfiguration Component
II	Information Integration
IMM	Internet Mail Module
IT	Information Technology
KM	Knowledge Management
LAN	Local Area Network
LF	Linefeed
LU	Login Unit
MUA	Mail User Agent
MTA	Mail Transfer Agent
MDA	Mail Delivery Agent
MSA	Mail Submission Agent
METAL	MEssage TAsk Linking
MCA	Multitask Coordination Assistance
Magi	Mail agent interface
MIME	Multipurpose Internet Mail Extensions
NN	Nearest Neighbor

PAEA	Personal Assistant Email Agent
PC	Personal Computer
PT	Power Tools
POP3	Post Office Protocol version 3
PU	Profile Unit
SGML	Standard Generalized Markup Language
SMTP	Simple Mail Transfer protocol
SSL	Secure Sockets Layer
SVM	Support Vector Machine
TCP	Transmission Control Protocol
TF-IDF	Text Frequency-Inverse Document Frequency
UBE	Unsolicited Bulk Email
UCE	Unsolicited Commercial Email
UC	Uploader Unit
URL	Uniform Resource Locator
XML	Extensible Markup Language
WWW	World Wide Web

Table of Contents

CHAPTER ONE
Introduction

Chapter One
Introduction

1.1 Overview

World Wide Web (www, w3), communication networks, and widespread computation and storage are significantly participating creating a world in which information is ubiquitous and inexpensive; however, the disadvantage of its access is inundation. Popular media describe an alarming situation in which individuals are spending more and more time filtering information. The flood of irrelevant, often unwanted, data has created a need for intelligent information management systems that automate the selection, sorting, presentation, storage, and retrieval of emails [GwDi97].

Email foldering is a rich and multi-faceted problem, with many difficulties that make it different from traditional topic-based categorization. Email users create new folders, and let other folders fall out of use. Email folders do not necessarily correspond to simple semantic topics, sometimes they correspond to unfinished to-do tasks, project groups, certain recipients, or loose agglomerations of topics. It is also interesting to note that email content and foldering habits differ drastically from one email user to another; so while automated methods may perform well for one user, they may fail horribly for another.

Furthermore, email arrives in a stream over time, and this causes other significant difficulties. Some email messages only make sense in the context of previous messages. Occasionally all messages in a thread should go to the same folder, but with time shift the topic in a thread may drift. The topic associated with a certain email folder can also shift over time. For example, a folder about funding opportunities may at first contain only messages about the

National Science Foundation, but later only get new messages about industrial partnerships, each of which may have very different word distributions [BeMc04].

1.2 Agents as User Assistants

Agent systems have been proposed as solutions to the problem of information overload, particularly regarding email and internet searches. Most of the current implementations aiming to ease the burden of dealing with email are text classifiers or keyword extractors, often working as email client plug-ins [AbMc01].

The basic idea behind software agents is to perform tasks similar to those that a human assistant would carry out. Gruen et al. (1999) conducted field studies and an analysis of the types of assistance provided by human assistants. They found that these included: pre-processing, filtering/ prioritizing, adding relevant information, performing a number of steps in response to a single request and peripheral awareness/pointing out information [GrSi99].

Murch, in 1999, had dedicated a chapter in his book for email agent issue. He suggested that they may successfully be employed to perform the following eight activities [MuJo99]:

1. Controlling any unwanted email or “spam”.
2. Alerting users by voice if a certain message arrives.
3. Automatic mail forwarding.
4. Consolidating mail from numerous sources.
5. Searching the internet for new sources of news, stocks and deals and then delivering them by email.
6. Distinguishing between private/personal and Corporate/business emails.
7. Automatically answering email and responding according to conditions.

8. Carrying out regular administrative tasks such as archiving and indexing for future searching.

1.3 Email Client

An email client, email reader, or more formally Mail User Agent (MUA), is a computer program used to manage email. Specifically, the term email client may refer to any agent acting as a client toward an email server, regardless of it being a mail user agent, a relaying server, or a human typing on a terminal. In addition, a web application providing message management, composition, and reception functionality is sometimes considered an email client [Par08].

1.3.1 Retrieving Messages from a Mailbox

Like most client programs, an MUA is only active when a user runs it. Messages arrive on the Mail Transfer Agent (MTA) server. Unless the MUA has access to the server's disk, messages are stored on a remote server and the MUA has to request them on behalf of the user.

In the case, shared disk, a user logs on a server and runs an MUA on that machine. The MUA reads messages from a conventionally formatted storage, typically mailbox, within the user's HOME directory. The MTA uses a suitable Mail Delivery Agent (MDA) to add messages to that storage, possibly in concurrence with the MUA. This is the default setting on many Unix systems. Webmail applications running on the relevant server can also benefit from direct disk access to the mail storage [MyRo96].

1.3.2 Submitting Messages to a Server

As a basic function, an MUA is able to introduce new messages in the transport system. Typically, it does so by connecting to either Mail Submission

Agent (MSA) or MTA, two variations of the Simple Mail Transfer Protocol (SMTP). The client needs to put a message quickly without worrying about where the message eventually will be delivered: that's why a transport system exists. Thus it always connects to the same preferred server, however, how does that server know that it should accept and relay submissions from that client? There are two ways. The older method recognizes the client's Internet Protocol (IP) address, e.g. because the client is on the same machine and uses internal address 127.0.0.1, or because the client's IP address is controlled by the same internet service provider that provides both internet access and mail services. The newer method, since the SMTP protocol has an authentication extension, is to authenticate the access. The latter method eases modularity and nomadic computing.

Client settings require the name or IP address of the preferred outgoing mail server, the port number (25 for MTA, 587 for MSA), and the user name and password for the authentication, if any. There is a non-standard port 465 for Secure Sockets Layer (SSL) encrypted SMTP sessions that many clients and servers support for backward compatibility. Transport Layer Security (TLS) encryption can be configured for the standard ports, if both the client and the server support it.

Email servers and client use the Transmission Control Protocol (TCP) port numbers, listed in table (1.1), by default, unless configured for specialized installations [Par08].

Table (1.1) TCP port numbers

Protocol	Purpose of use	Port Number
POP3	incoming mail	110
IMAP4	incoming mail	143
SMTP	outgoing mail	25

1.4 Webmail

Webmail (or Web-based email) is an email service intended to be primarily accessed via a web browser, as opposed to through a desktop email client (such as Microsoft Outlook, Pegasus Mail, Mozilla's Thunderbird, or Apple Inc.'s Mail). Very popular webmail providers include Gmail, Yahoo! Mail, Hotmail, and AOL.

One advantage of webmail over application-based email is that a user has the ability to access their inbox from any Internet-connected computer around the world. However, the need for Internet access is also a drawback, in that one cannot access old messages when not connected to the Internet (with the exception of some newer technologies, such as Gmail's "Offline Mail" feature). On the other hand, if one uses the IMAP protocol through an application-based email client, all contents of the mailbox will be consistently displayed in both the webmail and the PC email client contexts [BrEk98].

In 1997, before its acquisition by Microsoft, Hotmail (now Windows Live Hotmail) introduced its service, which became one of the first popular web-based email offerings. Following Hotmail's success, Google's introduction of Gmail in 2004 sparked a period of rapid development in webmail, due to Gmail's new features such as JavaScript menus, text-based ads, and bigger storage [And09].

1.5 Email Files and Folders

There are three kinds of text documents. The first kind has no specific structure, such as plain-text documents. The second kind has a well-defined structure, such as Standard Generalized Markup Language (SGML) documents. The structure of a document can provide clues for classification. The last kind of documents has a partial structure. For example, email messages and news articles are semi-structured, each with a structured header

part and an unstructured body of text. The structured part of the document can help us identify specific attributes useful for classification [SaWo97].

Email provides an example of a rich information management domain, Email is typically short, less than a hundred lines and contains a limited amount of structure. The body of an email is usually unstructured text, while the headers provide some tagged information. For example, an agent knows a priori the meaning of a "From" header field it can define the sender; similarly, the "Date" header field should reflect signs about when the email message is received.

The "Subject" header field is sometimes problematic. It says something about the contents of the email, but not always. Even for the headers with known content, the utility of their information is limited. Knowing the sender of an email is useful, but often the same sender may discuss different topics with same recipient. So, some form of content understanding is required [Boo98].

In graphical user interfaces, such as Windows and the Macintosh environment, a folder is an object that can contain multiple documents. A folder is a named collection of related files that can be retrieved, moved and otherwise manipulated as one entity. In the DOS and UNIX platform, folders are called directories. A folder is considered to be non-topical if email messages are stored in this folder regardless of their content. This category includes folders such as Inbox, Send, Trash and Drafts. It makes more sense to remove the non-topical folders, because no automatic system is going to assist the user in classifying messages into these folders. Non topical folders belong to two email categories [BeMc04]:

A. Automatically created folders of an email application (such as "Send items" of MS Outlook).

B. Archiving folders that are standard for all users belong to a certain organization (such as; "all-documents" and "discussion-threads" that can be found in the folder hierarchies of all former Enron employees).

1.6 Spam

Spam, officially called Unsolicited Bulk Email (UBE) or Unsolicited Commercial Email (UCE), is rapidly becoming a major problem on the Internet, because it is account for more than 81% of the total daily message traffic. To handle this increasing load of junk email, several spam filtering techniques exist to automatically classify incoming email as spam, and to reject or discard email classified as such [Rad09].

For users, receiving spam is quite a nuisance and costs money. A study, conducted on the European Community, had estimated the cost for receiving spam for an average Internet user is in the order of 30 euro in a year. But the cost of spam goes well beyond the total costs of all recipients. Each ISP (Internet Service Provider) pays for each email message received, because it must be stored in a mail box and it takes up a certain amount of bandwidth. The total cost has been estimated in the order of 10 billion euro per year [GaDr01].

A second problem with spam is the impact it has on the Internet backbone. Spam sent over the Internet backbone causes delays for all Internet users. Furthermore, because most spammers use mailing lists that have outdated addresses on them, many messages are rejected (“bounced"). This mandates the operator of the intended destination to send a return response, wasting even more bandwidth [HaLu99].

There are at least three fundamentally different ways to counter spammers. First, bulk mailers can be prevented to send spam by blocking or limiting access to mail servers. Another method is make spamming less profitable, for example by incurring a cost on every email message sent. The third method aims to detect and remove all spam once it is sent by applying different types of filtering techniques that use the special characteristics of spam to recognize it [AnKo00].

1.7 Literature Survey

A variety of approaches have been taken to address the problem of automating email classification. Most of these approaches address, either the general classification of email or the specific filtering of junk mail. Although this task can be viewed as special case of text categorization, a number of studies have introduces and/or investigated different ways of classifying email using machine learning and information retrieval (IR) approaches. Some of these systems are described below in approximate chronological order:

- In 1996, Cohen [Coh96b] had used the RIPPER learning algorithm, because it can induce rules that spot keywords for classifying email. His argue was the keyword spotting is more, as it induce, an understandable description of the email filter.
- Magi by Payne and Edwards [PaEd97], in 1997, have developed **M**ail **ag**ent **i**nterface (Magi) application to work on top of a UNIX mail system. They have examined two different techniques for sorting emails, CN2 a rule induction algorithm and IBPL1 a modified version of the K-nearest Neighbor algorithm which uses Memory Based Reasoning. These approaches have been used to predict action (e.g. forward, delete and file in folder X). They indicated that depending on the confidence "Magi" it can carry the action out automatically, suggest the action to the user and see if they agree, or make no suggestion at all. Payne and

Edwards found that 26% of the time when using CN2 no suggestion was made and 22% of the time when using IBPL1. Also they found that when Magi did suggestions it is 65% (for CN2), and it is better (i.e., 57%) when using IBPL1.

- Bonne [Boo98], in 1998, had introduced "Re:Agent" email tool, it is unlike other approaches. It is implies into two distinct stages. In the first stage, features are either learnt from collections of email messages using a Term Frequency–Inverse Document Frequency (TF-IDF) approach, or they are constructed by users providing keywords. This permits the user to guide the agent without explicitly formulating rules. The second stage uses the features, constructed by the first stage, for learning actions. Boone investigated both the neural networks and nearest neighbor approaches for this learning tasks. He found that both learning approaches are useful for making significant reduction in the dimensionality gained in the first stage (i.e. features extraction stage). However unlike other older approaches "Re:Agent" classifies emails into two categories only, 'work' or 'other', so its results was not compared with the work of other authors. However, Boone found that his introduced approach "Re:Agent" can achieve 98% accuracy, while the standard IR approach had 91% accuracy.
- Segal and Kephart [SeKe99], in 1999, introduced "MailCat" system. Their system used TF-IDF approach which computes weighted vectors for each folder based on word frequencies, and then a distance measure is used to estimate the similarity a new message has with each folder. They referred that, when new messages were directory filtered into the most similar folder an error of 20% to 40% resulted.
- Rennie [Ren00], in 2000, had used a naïve Bayes approach for text classification. He called his introduced method "iFile" filter. It works as filter for the EXMH mail client. The system applies stemming and

makes use of a stop-list. A number of iFile users were provided a program that performs a series of experiments on their own email folders; this helped address privacy issues relating to email. However, it also limits the ability of other researchers to perform a comparison with other approaches. On these folders Rennie achieved 89% accuracy using his "iFile" classifier.

- Mock [Moc01], in 2001, had introduced an experimental system to test different ideas for categorization and management within a real email environment. The project includes utility-like routines such as extracting features from email, assigning numeric values to features, stop listing, putting email into categories, or detecting when email arrives. The classifier is a simple Nearest-Neighbor (NN) classifier. Given a target message to classify, its features are extracted and compared to all messages in the classifier using the cosine coefficient. The top three matches are averaged as the similarity measure for the classifier. The message is then put into the classifier with the largest similarity measure. It was used to aid users that wish to search for emails; the add-in provides the capability to quickly display a list of messages ranked by relevance (using the similarity metric) to the selected message. In this manner, other messages in the same thread or in the same topic will be displayed at the top of the list.
- Moreale and Watt [MoWa03], in 2003, have introduced a system that works with several lists, giving users archiving and retrieval assistance through an intuitive and dialectic interface: users can email their query directly to the agent and receive a prompt reply day or night. Alternatively, users can post their query publicly to a forum (monitored by the agent) or run a web-like search over the monitored lists. Through the application of IR and a novel Information Integration (II) technique to the mailing lists, Sentinel automatically links email into a tangled

network of stories, and arranges them in a meaningful way (digests, queries asked to date).

- Freed and his colleagues [FrCa08], in 2008, have introduced a so-called "RADAR" system which consists of three sets of components: Message-Task Linking (METAL) components analyze email messages to identify new tasks, distill out other task relevant content and establish user-navigable links between task representations and message text. Using METAL output, Multi-task Coordination Assistance (MCA) components provide task-management support to the user such as maintenance of a to-do list and guidance on which item on the list to do next. Power Tools (PT) are application programs similar to the tools people normally use to produce work products, but are enabled by METAL and knowledge from past interactions to provide task aware user assistance.
- Fawzi [Faw08], in 2008, had introduced "EMFA" system which used machine learning to classify the email messages into two list, Negative list (that contains unwanted messages) and Positive list (that contains the messages that must be forwarded or replied).

1.8 Aim of work

The aim of this work is to implement a friendly automated system that help user to automatically manage their email messages according to his\her personal profile. It is design to send all user composed email messages to their recipients, download the new email messages from the user account in server to his/her local storage then sorting the email messages according to user interesting. Also, the system offer a number of services to the user (like, give the user the ability to filter-in and download the new email message according three email messages attributes, auto-reply email message and compose new email message with the ability to attach files).

1.9 Thesis Layout

Beside chapter one, the remaining part of this thesis consists of the following chapters:

- **Chapter Two: (Theoretical Background)**

  Discuss the concepts of the agent, the structure of the internet email message and internet mail model. Also, it discusses the email management and the techniques that used in email overload managing.

- **Chapter Three: (PAEA System)**

  In this chapter, the proposed system design and implementation steps are given. The PAEA system modules are described in details.

- **Chapter Four: (PAEA Interfaces and Evaluation)**

  This chapter presents the interfaces of the established system and state a user guide for how to user PAEA system properly. Also, a simulation is applied on PAEA to see the system results.

- **Chapter Five: (Conclusion and Future Work)**

  This chapter presents a list of derived conclusions and suggestions for future work.

CHAPTER TWO
Theoretical Background

CHAPTER TWO
Theoretical Background

2.1 Introduction

Although, there is no universally accepted definition of the term agent, an "agent" can be defined as an entity that can be viewed as perceiving its environment through sensors and acting upon its environment through effectors [RuNo03]. Software agent may viewed as a program that engage in dialogs, negotiate and coordinate the transfer of information [Coe95], or viewed as a hardware and/or software-based computer system that displaying the properties of autonomy, social adeptness, reactivity, and pro-activity [WoJe95]. There is a consensus that autonomy (i.e., the ability to act without the intervention of humans or other systems) is a key feature of an agent. Beyond that, agent has other different attributes; which take different importance based on the domain of the agent. Figure (2.1) depicts a high-level view of an agent within its environment. An agent receives input from its environment, and through a repertoire of actions available to it, reacts to it in order to modify it [Rud04].

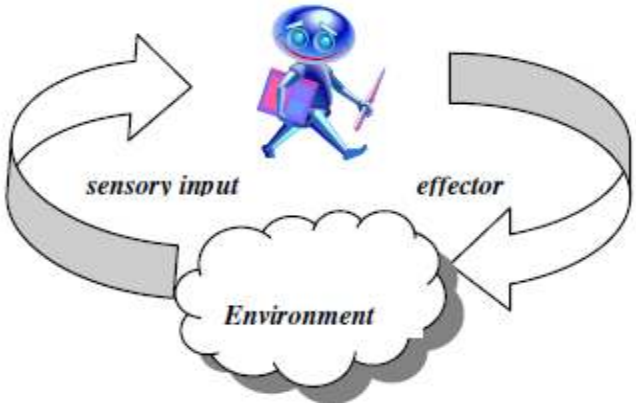

Figure (2.1) Agent interacting with its environment

Generally, in domains of reasonable complexity, an agent may not have total control over its environment. Thus, the same action performed twice in seemingly identical situations might appear to have completely different outcomes. Failure is also a possibility; that is, the action taken by the agent may not produce the desired effect at all [Rud04].

2.2 Agent Environments

The critical decision an agent faces is "determining which action should performed to best satisfy its design objectives". Agent environments are classified according to different properties; each can affect the complexity of the agent's decision-making process. They include [RuNo03]:

- Accessible vs. inaccessible: An accessible environment is one in which the agent can obtain complete, timely and accurate information about the state of the environment. The more accessible an environment, the less complicated it is to build agents to operate within it. Most moderately complex environments are inaccessible.
- Deterministic vs. non-deterministic: Most reasonably, complex systems are nondeterministic, the state that will result from an action is not guaranteed even when the system is in a similar state before the action is applied. This uncertainty presents a greater challenge to the agent designer.
- Episodic vs. non-episodic: In an episodic environment, the actions of an agent depend on a number of discrete episodes with no link between the performance of the agent in different scenarios. This environment is simple to design since there is no need to reason about interactions between this and future episodes; only the current environment needs to be considered.
- Static vs. dynamic: Static environments remain unchanged except for the results produced by the actions of the agent. A dynamic environment

has other processes operating on it (e.g., by changing the environment outside the control of the agent). A dynamic environment obviously requires a more complex agent design.

- Discrete vs. continuous: If there is a fixed and finite number of actions and percepts, then the environment is discrete. A chess game is a discrete environment while driving a taxi is an example of a continuous one.

2.2.1 Intelligent Agent

An intelligent agent is defined as "an agent capable of flexible autonomous action to meet its design objectives". The word "Flexible" in above definition means [WoJe95]:

- Reactivity: intelligent agents perceive and respond in a timely fashion to changes that occur in their environment in order to satisfy their design objectives. The agent's goals and/or assumptions that form the basis for a procedure that is currently executing, may be affected by a changed environment and, such case, a different set of actions may be need to be performed.
- Pro-activeness: reacting to an environment by mapping a stimulus into a set of responses is not enough. Goal directed behavior is needed in intelligent agents. In a changed environment, intelligent agents have to recognize opportunities and take the initiative if they designed to produce meaningful results. The challenge to the agent designer is to integrate effectively goal-directed and reactive behavior.
- Social ability: intelligent agents are capable of interacting with other agents (and possibly humans), through negotiation and/or cooperation, to satisfy their design objectives.

Other properties sometimes mentioned in the context of intelligent agents include:

- Mobility: is the ability to move around an electronic environment.
- Veracity: an agent will not knowingly communicate false information.
- Benevolence: agents do not have conflicting goals and every agent will therefore always try to do what is asked of it.
- Rationality: an agent will act in order to achieve its goals in so far as its beliefs permit.
- Learning/adaptation: agents improve performance over time.

Users of the web are facing information overload problem; that is the amount of their available data doubles annually. Individuals can analyze only about 5% of the data and most efforts do not provide real meaning. Thus, the need for intelligent agents is critical to assist in searching, filtering, and deciding what is relevant to the user [Coo01].

2.2.2 Automated Intelligent Agent

Under a certain memory constraint for each email client, the administrator of the information system (i.e., the mail server) may choose to adopt a "brute-force" approach, based on structured information such as time stamp or size of the message. Consequently, clients, without their consent, may often realize their emails unavailable at times, once they have reached the memory quota set by the system. However, this aggressive method is not effective, due to its user-service implications if not legal.

Thus, an automated system that may advise the clients in real time about their email usage patterns is considered as an attractive alternative for information resource management. Moreover, all clients of the information system may share the workload of managing the memory requirement for the mail server. Once the client is logged onto the system, the automatic intelligent

agent generates a list of email messages that are to be removed, as well as those that are highly likely to be candidates for local archives. In addition, another agent system is suggested for the server to advice administrator(s) for potential preventative measures [Sug07].

2.3 Internet Email Message

An email message is a piece of information that an internet user can send it to another. It may include multiple parts, including binary files that may be attached to the message. An internet email message consists of a number of header fields and a body. The body of a message is actually optional.

At the most basic level, a message is defined as a series of characters. A message that is conformant with this standard is comprised of characters with values in the range 1 through 127 and interpreted as US-ASCII characters. There are other documents, specifically the Multipurpose Internet Mail Extensions (MIME) document series that extend this standard to allow for values outside of that range [Res01].

MIME is an Internet standard that extends the format of email file to support [Bac07]:

- Text in character sets other than ASCII.
- Non-text attachments.
- Message body with multiple parts.
- Header information in non-ASCII character sets.

The blank line separating the header fields from the body is, in reality, an ASCII carriage return character (CR) followed by an ASCII linefeed character (LF) on a line by itself. A program receiving email can also note the location of this blank line by performing a regular expression match on the headers to find the first instance of two CRLF combinations in sequence, one to end the last header field and the other to constitute the blank line [Woo99].

Great feature of MIME is that it allows messages to have structure. A mail client that cannot display Hyper Text markup Language (HTML) can skip that part of the message and just display the plain text. This allows message content to gradually migrate towards new technology. In the near future, it is likely that similar logic will be used for messages that contain Extensible Markup Language (XML), HTML, and plain text [Moo96].

Because of the capability to structure messages, MIME can be used for multimedia and unified messaging. A single email message can contain one or more movies, sound files, text files in a variety of formats, binary files such as word processing documents, calendar events, fax images, and so on. The MIME structure tells the recipient software which parts of the message contain particular types of data, as well the relationship between the parts (such as "this part contains three different alternative sound formats") [PaHo99].

2.3.1 Email Message Header

Header fields are necessary for any standards compliant message. Header fields contain information such as where the message came from, where it is going, when it was sent, and more.

However, only two header fields are non optional for standards-compliant messages:

From: indicating the originator of the message.

Date: indicating the origination date of the message.

All header fields consist of the following, ordered, parts [Los99]:

1. Field name: Field names must consist entirely of printable US-ASCII characters except for the colon character.
2. Colon (“:”): A colon sets off the field name from the field body.
3. CRLF: A CRLF indicates the end of the header field.

However, approximately two dozen different header fields are specified in the standard. Headers can be categorized as optional or required, as structured or unstructured. More specific field categories include originator and destination fields, identification and informational fields, resent, trace, and optional fields [Los99].

Table (2.2) Email header fields

Header Filed Name	Description	Categorize
Date	Indicates the date and time when the creator of the message considered it to be complete.	Message Origination.
From	Indicating where the message came from.	Message Origination.
Sender	Used when the entity placing the message into the mail delivery system is different from the entity that created the message.	Message Origination.
Reply-to	Contains mailbox addresses or groups (for lists of addresses), indicating where replies to the message should be directed.	Message Origination.
To	Contains the address (es) of the primary recipient(s) of the message.	Message Destinations.
Cc	Contains the address(es) of recipient(s) who will get copies of the message as a courtesy.	Message Origination.
Bcc	Unlike the "To" and "Cc" header fields, indicating that blind copies were sent, but without any indication of to whom they were sent.	Message Origination.
Message-id	Contains a unique identifier issued by the host, and referencing the host, on which the message was created.	Message Identification
In-reply-to	Contains the message ID of the message to which the current message is replying.	Message Identification
References	Contains the message IDs of all other messages in the same thread.	Message Identification
Subject	Contains a brief description of the contents of the message body.	Message Information.
Comments	Contain additional information about the body of the message.	Message Information.
Keywords	Used by the recipient to classify the message or for searching through message stores.	Message Information.

2.3.2 Email Message Body

Originally, the body of email messages consists of plain ASCII text. This was sufficient for the inventors of email, who spoke mostly English and had access to other information transfer mechanisms such as FTP to move binary data around. The body of a message is simply lines of US-ASCII characters. The only two limitations on the body are as follows [Res01]:

1. CR and LF must only occur together as CRLF; they must not appear independently in the body.
2. Lines of characters in the body MUST be limited to 998 characters, excluding the CRLF. (As was stated earlier, there are other standards documents, specifically the MIME documents)

Probably the biggest advance in internet email in the past years is in the format of email messages, not in their transport. In the early 1990s, internet email went from being text-only to allowing the transfer of non-text messages and parts of messages. MIME revolutionized the usefulness of internet email by allowing senders to include files with messages, to use styled text, to give their messages useful structure, and to provide the first interoperable support for international email [FrBo96a].

Internet email went from being text-only to allowing the transfer of non-text messages and parts of messages. Both plain text and HTML are used to convey email. Advantages of HTML include [FrBo96b]:

1. The ability to include inline links and images.
2. Set apart previous messages in block quotes.
3. Wrap naturally on any display.
4. Use emphasis such as underlines and italics, and change font styles.

2.4 Internet Mail Model

The Internet Mail Model (IMM), like the internet itself, is a collection of standardized components all acting with a common goal. In the case of email, the goal is to provide the framework for carrying electronic messages between one user and another. Each of the end users may be on very different platforms. Their respective sites may have vast geographic, technological, and social differences. Such differences demand that the framework be at once both robust and flexible. The internet's email framework consists of agents, mail stores, and standards. Figure (2.2) illustrates how the agents, mail stores and standards work together [MuMu00].

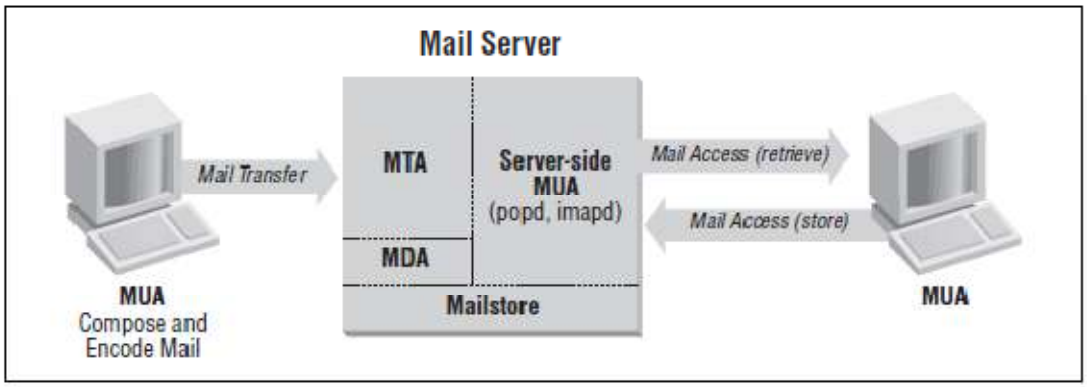

Figure (2.2) Email life cycle

The most important elements of the Internet mail system are [Woo99]:

1. Mail User Agent (MUA): client program used by a user to send or receive email. An MUA could, also, be a program or script that emulates the behavior of a typical MUA by sending or receiving email.
2. Mail Transfer Agent (MTA): A server program that transfers email from one machine on the Internet to another mail server.
3. Mail Delivery Agent (MDA): A small program used by an MTA to write a message into a user's mailbox.

4. The Mailstore: It is the filing cabinet of the mail system. When a user receives a piece of email, it's deposited into his portion of the Mailstore, by using an MUA to peer into the Mailstore the email is retrieve and the user can reviews it.

2.4.1 Mail Transfer

Some standards are needed for describing how that message is conveyed upstream to other hosts on the internet. The most important of these is the RFC 821, "Simple Mail Transfer Protocol (SMTP)". SMTP defines how mail is transported from one place to another, whether that transport is from your MUA to the MTA or between two MTAs.

SMTP defines how the client and server initiate a connection, how they transfer mail, and how they close the connection. How email gets from a file system or directly from a user to an SMTP client is of no concern to SMTP. This is the job of the mail client software, or of some other mechanism, that is capable of placing the message into the SMTP transport environment.

In practice, the mail client software may use SMTP to submit the message to a fully functional SMTP system, but how the message gets to an SMTP system is irrelevant to the protocol itself. When an SMTP system receives a message, the SMTP entity can forward the message [MuMu00].

2.4.2 Mail Access

A mail access protocol is a means by which the mail client software may perform operations on messages that have already made it to the Mailstore. Not just "read" messages that are in the Mailstore, Although POP3 is a "read-heavy" protocol, IMAP permits users to add messages to the Mailstore, move them around, and change their attributes or the degree of access other users have to them [Los99].

A. POP3 (Post Office Protocol, Version 3)

POP3 provides a standard mechanism for retrieving email from a remote server. The messages stored on the server will be deleted. This allows many millions of home internet users to have email sent to a mail server at their internet service provider, and then retrieve it when they are online.

Small offices can use POP in the same manner to pull mail for many users from a mail server at a home office or an internet service provider. These remote mail servers are known as mail drops. In short, any user that wishes to receive email on his own machine or Local Area Network (LAN), but does not have a permanent internet connection ،may use POP3 [Woo99].

B. IMAP (Internet Message Access Protocol)

IMAP is a way to retrieve messages from one or more mailboxes on a central server, without ever having to download a single message to local hard disk. The messages remain on the server at all times. By design, IMAP was intended to provide the same level of functionality for mailbox and message access and management that exists with a mailbox located on a local hard drive. Consequently, IMAP has server operations, such as "search for messages matching such-and-such criteria," that are normally associated with mail clients. The advantages of IMAP can be seen very clearly if you work from several computers (e.g., home computer, office computer, and laptop). With IMAP, the user doesn't have to wonder on which computer, when will downloaded and read a given message. The user will know it's still on the server. With the right IMAP client, all of the following could be done:

1. Learn when new messages arrive in any of the mailboxes of the user.
2. Share the user's mailboxes with anyone or everyone.

3. Move messages from one mailbox to another.
4. Mark messages with flags (such as "Important") that are preserved between IMAP sessions.

Another distinguishing feature of IMAP is that, the mail folders (mailboxes, in IMAP parlance) are stored in a central location on the IMAP server not the inbox only. As long as the user has got IMAP client software, not only can a user read the incoming email from just about anywhere on the net, but a user can access all the mail that archived in the mailboxes, too [MuMu00].

2.5 Knowledge Management

Information Technology (IT) may support knowledge management (KM) in two classes: codification and personalization. In essence, the codification approach manages structured knowledge, whereas personalization manages unstructured, tacit knowledge. Because email usage patterns for end users may entail both types of knowledge. That is, there may be common patterns of email management among end users (such as removing messages which are more than 36 months old) or organizing their mail folders every 30 days, and so on.

On the other hand, each user may have highly subjective patterns that may not be consistent with codified knowledge [WoJe95].

The general stages of any KM process are:

1. Knowledge elicitation.
2. Knowledge packaging.
3. Distributing or disseminating knowledge.
4. Reusing knowledge.

To this end, building knowledge bases in conjunction with automatic agents is considered one of the good methods for managing user knowledge associated with email usage in real time [Mar01].

2.5.1 Attributes for Knowledge Management

The process of information retrieval in the human mind is fundamentally different from a filing or library system, in which items are accessed by location rather than their meaning. The first notion is that people recall chronological information about information: what else was happening at roughly the same time. Consequently, the "time stamp" of emails may be a good source of structured information or knowledge. Association is another means by which humans retrieve information.

Each email message consists of four pieces of tacit information: recipient or sender, event or subject, attachment(s), and significance of the message. Table (2.1) summarizes the attributes for KM concerning email usage.

Table (2.1) Attributes of knowledge-management for email usage

Attribute	Type of knowledge
Time stamp	Structured
Size (kb)	Structured
Recipient/Sender	Tacit/Unstructured
Event/Subject	Tacit/Unstructured
Attachment	Tacit/Unstructured
Significance	Tacit/Unstructured

A set of six generic attributes associated with emails are mentioned in above table. Notice that the first two attributes (i.e., time stamp and size) are structured information, which may be available for both the server and clients. By contrast, each client may manage his/her email messages based on one or more of the four tacit attributes. Given a certain restriction of memory size, say

100MB per email account holder, one client may choose to either remove or archive (on local memory store) emails based on: (1) Recipient/sender, (2) Event/subject, (3) Attached file(s), (4) Significance, (5) Or any combination of the four.

So far as tacit knowledge management is concerned. Alternatively, he/she may simply choose to archive or remove emails with regards to structured information such as time stamp and/or size [Sug07].

2.5.2 Techniques for Managing Email Overload

Techniques for managing email overload include rule-based techniques and automatic clustering techniques. Rule-based techniques require an email user to define a set of rules to sort the incoming messages into existing folders. Many commercial products, such as Microsoft Outlook and Eudora, are using rule-based techniques. As it is difficult for non-technical users to create folders and rules, rule-based techniques can only be an auxiliary way to manage email overload problem [Xia08].

Automatic clustering techniques classify incoming emails and reduce the effort required for rule creation and folder maintenance. Currently there are few techniques used to manage email overload For example;

1. Magi records each email interaction and uses a machine learning algorithm to classify the new messages according to the user's previous behavior, this requires a training phase involving all previous records to be trained to perform the classification [PaEd97].
2. Prototype system was developed to scan all emails in existing folders and creates a nearest-neighbor (NN) classifier. Then the system compares the incoming email messages to the classifier and categorizes them. Although the training of NN classifier is fast, it is not effective when training data is insufficient [Moc01].

3. A system was proposed to automatically identify messages belonging to the same structured activity. It is a system that supports the use of email as a task manager from a high level point of view. Email-based activities are formalized as finite state automata, where messages represent state transitions [KuLa05].

2.5.3 Text Clustering Techniques

The main goal of clustering is to partition a given set of objects into homogeneous groups based on given features, such that objects within a group are more similar to each other and more different from objects that are in other groups [ChHa96]. The task of text clustering is to assign an electronic document to one or more categories, based on its contents. Text clustering has been used in many areas such as ranking the Web search results [HuCh06], automatic generation of taxonomy of Web documents [ChCh02], and Spam detection [SaSh05].

Many text clustering techniques have been explored. The main two categories of clustering techniques are: (1) hierarchical and (2) partitioning-based clustering techniques. Hierarchical text clustering techniques first classify the documents into a few broad classes, each of which is further divided into smaller classes, and each of these further partitioned, and so on until terminal classes are generated which are not further subdivided. Hierarchical clustering techniques can be further divided into agglomerative methods (such as the nearest neighbor and centroid cluster analysis) and divisive methods (such as association analysis). Hierarchical text clustering techniques inherently generate a hierarchical tree structure, thus have been used in many taxonomy applications. Partitioning-based text clustering techniques create a non nested partitioning of the data by using iterative partitioning process.

The Partitioning based techniques are firstly partition the data into some specified number of clusters, compute the centroids of these clusters; secondly, allocate all the data into the clusters that have the nearest centroids; thirdly, compute the new centroids of the clusters; and then repeat the second and third steps until no data change occur in the clusters. The typical partitioning based clustering techniques are the k-means algorithm and its variants. The time complexity of this method is linear in the number of documents. However, then number of clusters, k, must be known before the partitioning [LaAo99].

2.5.4 Email Management

Most email mining tasks are being accomplished using email classification at some point. In general, email classification confronts the assignment of an email message to one from a pre-defined set of categories. Automatic email classification aims building a model which will undertake this task on behalf of the user. Typically, email classification is achieved by utilizing machine learning techniques examples of applications are automatic mail categorization into folders, spam filtering and author identification. But there are also other applications, where the classification is used in the process (like automatic email summarization).

Actually, there are two kinds of classification. The first and simplest one is the flat classification; it is applied when one level of classes is specified. The other category is known as hierarchical, where a hierarchy of classes and subclasses are specified [BeMc04].

These models (or classifiers) can be built using various machine learning techniques, such as Naïve Bayes [SaDu98], Support Vector Machines [KlYa04], Rule Learning [Paz00]. Most of the classification algorithms are compatible with the vector representation model. The stage of building the classifier is called training. To build a classifier, a set of training examples is

required. An example is a message that has already been categorized, usually by a user or a domain expert. Each example is usually represented as a vector $e = [w_1,\ldots,w_n,c_e]$ where "**c**" is the class that the example "**e**" belongs to.

What is important in email classification is the fact that email is a dynamic environment and messages are constantly arriving. This means that although there might be a training set in availability, the classifier needs to be able to adapt new knowledge while new examples arrive [KaTs05].

Depending upon the mechanism used, email classification schemes can be broadly categorized into:

A. Rule based classification

One of the rule learning algorithm that used is the RIPPER algorithm, it is a propositional learner designed for efficient performance on large, noisy datasets. RIPPER builds a rule set by repeatedly adding rules to an empty rule set until all positive examples are covered. Rule are formed by greedily adding conditions to the antecedent of a rule (starting with an empty antecedent) until no negative examples are covered. After a rule set is constructed, an optimization post passes messages the rule set so as to reduce its size and improve its fit to the training data. Combinations of cross-validation and minimum-description length techniques are used to prevent over fitting [Coh95a].

RIPPER is designed to handle-set and bag-valued attributes equivalently by generating “keyword-spotting rules.” These rules have the form

cs328 ⟵ "utexas" ∈ from ∧ "utexas" ∈ to

This rule states that a message belongs to the folder "cs328" if the word “utexas” appears in both the "From" and "To" headers. Rules like this are highly suitable for email classification and filtering because many email reading programs are already equipped to use rules of this type in

classification, thus integrating this kind of rule-learning system into existing mail systems becomes a question of converting the syntax of the rules into one understood by the mail reader [Coh95b].

B. Information Retrieval Based Classification

The Term Frequency-Inverse Document Frequency (TF-IDF) is the most common weighting method used to describe documents in the Information Retrieval (IR) problems. Regarding text categorization, this weighting function has been particularly related to SVM. The TF-IDF function weights each vector component (each of them relating to a word of the vocabulary) of each document on the following basis. First, it incorporates the word frequency in the document. Thus, the more a word appears in a document (e.g., its term frequency, TF, is high) the more it is estimated to be significant in this document. In addition, IDF measures how infrequent a word is in the collection.

This value is estimated using the whole training text collection at hand. Accordingly, if a word is very frequent in the text collection, it is not considered to be particularly representative of this document (since it occurs in most documents; for instance, stop words). In contrast, if the word is infrequent in the text collection, it is believed to be very relevant for the document. TFIDF is commonly used in IR to compare a query vector with a document vector using a similarity or distance function such as the cosine similarity function [YaLi99].

C. Machine Learning Based Classification Techniques

The Naïve Bayes Classifier had been used many times for email classification. It is computationally cost effective and easy to implement. Beside to these characteristics, its flexibility and considerable performance are the basic characteristics that made it so popular, not only in email applications, but in text classification in general. The Naïve Bayes Classifier is based on the

simplifying assumption that "the attribute values are conditionally independent" [Ren00].

First email messages will be represented in a structured “bag of words” representation. Each email message is represented as vectors of features, in which the message body and each individual message header are represented as separate features. The content of each feature is all the words that appear in that feature, with repeated words counted multiple times. In tokenizing the messages, all letters are converted to a single case, all punctuation is removed, and email addresses, domain names, URLs, etc, are broken down into their constituent “words.” For example:

From: Jefferson Provost

jp@cs.utexas.edu

Would become

from = {Jefferson, Provost, jp, cs, utexas, edu}.

Many high-frequency, low-information content words, such as “a,” “an,” “the,” most prepositions and conjunctions, and all single-character words are removed from the token stream before bagging, The ability of this classifier to utilize the word counts in the bags of words in calculating its probability tables, The disadvantage of a Naïve Bayes classifier is difficulty of integration with existing mail reading software, because of the lack of a rule-based representation of the classification [Pro02].

CHAPTER THREE
PAEA SYSTEM

CHAPTER THREE
PAEA System

3.1 Introduction

This chapter presents the design aspects of the established personal email agent system, named **PAEA** (**P**ersonal **A**ssistance **E**mail Agent); it assists the user to automatically manage the growing volume of his/her email messages. The agent system establishes tables to assist the agent in managing and archiving user's email messages. It works to automatically:

1. Upload the user's composing email messages and sends them to their recipients.
2. Download then classify the user's incoming messages into folders.
3. Prioritize each incoming message according to user's preferences.

The agent does the above jobs by utilizing a set of learning rules for classifying and managing emails; such that it can assist user by automatically filter in his/her email messages. The developed classification algorithm is a keyword-based method. The agent prioritizes user's email messages depending on his/her profile; the profile is constructed and updated by tracking the personal-responses taken by the user over time.

3.2 System Requirements

Some requirements are needed to implement all adopted design considerations, which can be summarized in the following points:

1. Any valid email account belongs to any internet service provider that supports IMAP and SMTP protocols (e.g., Gmail or any paid email account).

2. The agent core engine utilizing some services offered by three emails Dynamic Link Libraries (DLLs), these services are utilized to handle the email structures and protocols. The used email DLLs are:
   a. **Rebex.Mail.dll**: is referenced in this work to be able to use the features of email for .NET. It contains classes that enable PAEA to create, read, process and save email messages in MIME format using the "MailMessage" class. Also, it contains the "Rebex.Mime.Headers" namespace with a number of classes that represent mail message headers, which will be very useful to extract all the information about the email message (i.e. sender address, subject, arrived time and etc).
   b. **Rebex.Net.Imap.dll:** it contains classes which make it easy to connect and authenticates the agent to the IMAP server, downloads a specified part of the message or the whole message with a specified unique ID, gets information about a message with a specified unique ID, search a specific folder on the user's account (i.e. Inbox) for email messages that match a specified searching criteria and modifies the flags of an email message with the a sequence number.
   c. **Rebex.Net.Smtp.dll**: it contains "Smtp" class that makes it easy to read an email message from a specified local file and send it using a specified SMTP server.

   For more details, appendices (A, B, C) present more details about the services offered by the above libraries.

3. A Programming language that supports internet protocols operations. In this work Visual Basic.Net is adopted to implement this agent.

3.3 PAEA Architecture

PAEA consists of five components: (i) Initialization and Reconfiguration component, (ii) Uploader component, (iii) Downloader component, (iv) Email Management component and (v) Email Services component; as illustrated in figure (3.1).

The proposed agent system is developed to be reactive and automatic. Its structure consists of two sets of components; the first set of components deal with email account as long as there is an internet connection, while the second set of components offers off-line services (like, email composing, prioritize, archiving and browsing) whether there is an internet connection or not. These two sets are designed to work in an asynchronous and collaborative way. PAEA components are:

1. Initialization and Reconfiguration Component: It initializes the agent system information. It consists of three units:
   a. Profile unit: It defines from the user the interesting subject list to the user and all relevant configuration information of the agent system.
   b. Connection Check unit: It checks if a connection to the internet is available or not.
   c. Login unit: It defines all the information needed from the user to accomplish the login and authentication process.
2. Uploader Component: Automatically sends all scheduled email messages which are saved on the user local storage (Outbox folder) to their recipients.
3. Downloader Component: It monitors the user's account inbox in the server and downloads the newly coming email messages to its local storage (after being registered in the email-records table).
4. Email Management Component: it prioritizes and sorts the incoming email messages based on the user interests.

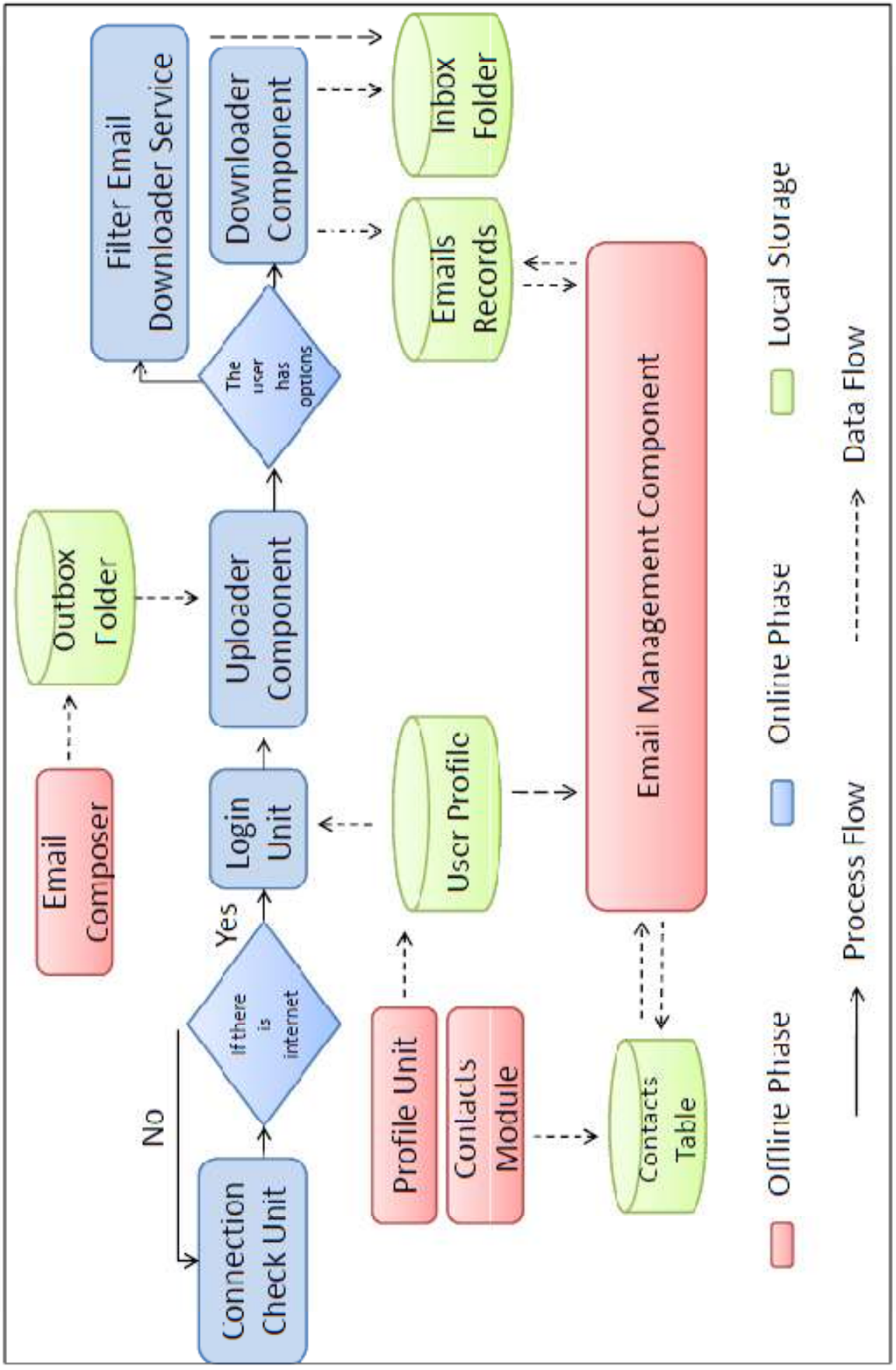

Figure (3.1) PAEA Architecture

5. Email Services Component: it offers the following services:
   a. Contacts module: Creates records for contacts that the user will exchange email messages with them, then save the records in a contacts table.
   b. Email Browser: Browses the email messages and downloads any files that attached to the downloaded email message.
   c. Email Composer: Composes a new email message, it has the ability to attach more than one file to it.
   d. Auto-Reply Email Message Service: Automatically sends an auto-replay email message to the email addresses that the user exchange email messages with them.
   e. Filter Email Downloader Service: Filters all received email messages and download only the emails that match the user's criteria.
6. User Profile: Holds the user's personal information and all relevant information necessary for account's login process.
7. Email-Records Table: Holds records of information to each downloading email message.
8. Contacts Table: Holds a record of information for each contact (person or website) that the user interested to exchange email messages with them.

3.3.1 Initialization and Reconfiguration Component (IRC)

Predefined information and actions from the user are needed: (i) to make sure that a successful login process to the user account will be accomplished and (ii) to achieve results that match the user's expectations. This component consists of three units:

1. Profile Unit (PU): When a new user uses PAEA system, a user profile will be created, the user profile contains the connection's information which should pre-assign by the user to initiate the connection. Also, the

user profile contains a brief description about what are the most interesting subjects to his\her which can be used later as keywords in email management process, and this information can be modified by the user whenever he/she wants. The involved user profile information includes:

a. Login Information: It contains the server name, account username and password; this set of information is a required to enable an access to the user' email account.
b. Interest Subjects List: It is a list of "interest subjects" for the user; it could be pre-assigned by him. This list is changeable and could be modified by the user during his continual interaction with the agent system.
c. Actions Setting: The user can adjust the agent how to react with the uploaded and downloaded email messages (delete the email messages after it is downloaded from the server or not, specifying the folder on local storage where the email messages are saved and if the agent after downloading each new email, will send an auto-replay email message to email's sender).

The user profile is treated as a record consists of the following fields:

a. Address: A string of 50 characters holds the email address of the user.
b. Username: A string of 20 characters holds the user ID of the user's email account.
c. Password: A string of 20 characters contains the password of the user's email account.
d. Incoming: A string of 50 characters holds the name of the server used to download email messages from it.

e. Outgoing: A string of 50 characters holds the name of the server that used to upload the email messages to it.
f. Del_flag: A Boolean valued flag; if this flag is marked "true" the email messages will be deleted from server side after downloading it and saved in the local storage media; otherwise a copy of the email messages will be left on the server storage media. This is an important option offered by PAEA system because the users who login to their email accounts from different stations (i.e. PC in internet café, mobile phone) still can see a copy of their old email messages saved on the server.
g. Rep_flag: A Boolean valued flag; when it is set "true" then an auto-replay email message will be issued and send to the sender after downloading its send message.
h. Body: A string of 100 characters holds the auto-replay email message's content.
i. Path: A string of 100 characters holds the path directory in the local storage, where the downloaded email messages are saved.

2. Internet Check Unit (ICU): As a first main stage for all on-line operations a connection with internet must be established to reach the user's email account on an email server; this will let PAEA capable to make the required access to user email account. This unit is continually checks if there is an internet connection to automatically activate Login Unit operation to establish a connection with the user email account. Algorithm (3.1) illustrates how the Internet Check unit works.

Algorithm (3.1) ICU Algorithm
Goal: Check the existence of internet connection. **Input**: internet connection signal **Output**: **Flag**: return (true or false) indicate the connection status
Step1: //initiate a web-request for a specific URL. **req** = WebRequest.Create(url) **Step2**: //"GetResponse" system function returns a response to an // Internet request. If there is response then **Flag**= True Else **Flag**= False **Step3**: return **Flag**

3. Login Unit (LU): after supplying the user credentials to login method the PAEA, by using IMAP protocol, will automatically choose the available authentication method and log in. Algorithm (3.2) illustrates how the Login unit works.

Algorithm (3.2) LU Process
Goal: Login the Agent into the user's email account **Input**: **User**: the username of email account. **Password**: the password of the email account. **Incoming**: the address of the IMAP server **Output**: None.
Step1: Load (**User, Password, Incoming**) from the user profile. **Step2**: //Connect to the User's email account using (Rebex.Net.Imap.dll) function. Connect(**Incoming**) **Step3**: //authenticate the login by using (Rebex.Net.Imap.dll) procedure. Login (**User, Password**)

3.3.2 Uploader Component (UC)

One of the gained benefits due to the usage of PAEA system is "the email messages could be composed in off-line mode (i.e., internet connection is not required instantly)". After the preparation of the email messages, they will be automatically stored in "Outbox" folder with the overhead information required to send the messages later without need to user interception. Later, the agent will send the stored messages to their recipients automatically as soon as an internet connection becomes available.

To implement all the design assumptions for this component, the following steps must be done:

First: By using "getfiles" function, all the file names of the email messages that the user had already composed and saved in "Outbox" folder, are saved in an array of string. Algorithm (3.3) illustrates how "getfiles" works.

Algorithm (3.3) getfiles Algorithm
Goal: Get the file's names of all email messages saved in "Outbox" folder. **Input**: **Path**: string represents the path of the "Outbox" folder. **Output**: **Files**(): array of string represents the file names of email messages.

Step1: Define **s, Temp_files(), files()** As String
Define **f** As FileInfo
Define **counter** As Integer
Step2: //Send "**Path**" to "GetFiles" system's function, it returns the
//names of files (as array of strings).
Temp_files= GetFiles(**path**)
Step3: For each **s** in **Temp_files**
f= New FileInfo(**s**) //Initializes a new instance of the FileInfo
//class, which acts as a wrapper for a file path.
If **f.Extension** = ".eml" Then // Check if its email message file
files(i) = f.Name()

```
            i ++
          End If
        Next
Step4: Return (files)
```

Second: Then one by one, the stored messages will be uploaded using the procedure "Load", which is a member of "MailMessage" class in "Rebex.net.mail.dll". PAEA uses the protocol SMTP for sending email messages to a pre-specific email server. This later step is done after PAEA determines the outgoing email SMTP server from its configuration then the SMTP client typically initiates a TCP connection to the server through well-known port (i.e. port number 25), which is designed for the SMTP, and finally the email message is transferred to a remote server using a series of queries and responses between the client and server. After the email message is sent to its recipient, it will be moved from "Outbox" folder to "Sent" folder. Algorithm (3.4) illustrates how the Uploader component works. To accomplish all the above mentioned actions the following information is needed:

1. To: The email address of the reception.
2. Subject: The subject title of the email message.
3. Body: The content of the email message.
4. ServerName: The address of the outgoing server for the destination user account.
5. From: The email address of the user.

The information 1, 2 and 3 are extracted from the email message file. While, information 4 and 5 are extracted from user profile record.

Algorithm (3.4) UC Algorithm
Goal: Automatically send all the email messages in "Outbox" Folder. **Input**: Email messages in "Outbox" folder. **Output**: None.
Step1: **Email_array= get files** () // Algorithm (3.3) **Step2**: For each **message** in **Email_array** Load (**message**) Send (**From, To, Subject, Body, ServerName)** Move the email file from (**Inbox**) folder to (**Sent**) folder Next

3.3.3 Downloader Component (DC)

The downloader's function is to download all newly incoming email messages from the user account area on the server to the local storage. The downloaded and saved email messages can be accessed, through the email browser component of the PAEA system. The automatic downloading of the email messages and storing them into a local storage could be useful for reducing the time needed to be on-line with internet media in order to access all the email messages (even the previous ones). Algorithm (3.5) illustrates how the Downloader Component works.

Algorithm (3.5) DC Algorithm
Goal: Download the newly incoming email messages from user account and save them in his local storage. **Input**: User email account. **Output**: Update the email-records table.
Step1: Select Folder ("**Inbox**") **Step2**: Get the number of unseen email message **Step3**: //Search for all unseen (new) emails **Tmp** =Search(HasFlagsNoneOf(ImapMessageFlags.Seen))

Step4: // saved the email and create the email's record
For Each **message** In **Tmp**
GetMailMessage(**message.SequenceNumber**)
Save the email message
Create a record for the email message with its relative info.
Next
Step5: If **Del_flag**= True then DeleteMessage
Step6: If **Rep_flag**= True then Send the auto-replay message.
Step7: Update the email records table

The operation stages of the downloader component are:

1. Current Folder Selection: The email account of any user could be organized into a number of folders, the default ones are:
   a. Inbox: Contains the newly incoming email messages.
   b. Sent: Contains all email messages that had been sent.
   c. Spam: Contains the unwanted email messages.

   Many IMAP commands operate on the email messages exist in the currently selected folder. If no folder assigned, many of the email access commands will be access denied, among these commands are the GetMessageList, GetMessage, DeleteMessages. To be able to use these commands, a folder has to be selected using the "SelectFolder" command. In addition to "folder" selection, by assign the current folder, using "CurrentFolder" property, can be used to determine various useful information about the folder, such as messages count, number of unseen messages and supported flags. The "SelectFolder" method and "CurrentFolder" property are members of the "Imap" class in "Rebex.Net.Imap.dll".
2. Message Downloading: IMAP supports "Search" command, which is a member of "Imap" class in "Rebex.Net.Imap" library. The agent can search for email messages that match a given criteria by the search

method. It accepts a variable number of search parameters. So, by sending "HasFlagNoneOf(FlagSeen)" as a parameter to "Search" command, it will return all the unseen email messages in a temporary container of message objects, then the call of "GetMailMessage" method, which is a member of "MailMessage" class in "Rebex.Net.Mail", causes the retrieval of an instance of the "MailMessage" class that contains the message corresponding to the supplied sequence number. The user can read message header, body and attachments; and save it into the local storage.

3. Create a record for each downloaded email message, it contains all the necessary information about the email message, and it will be used later in the email management process. The record's fields are:
    a. Read: It is a flag to show if the email message is previously read or not.
    b. Seen: It is a counter indicates the user level interaction with the agent work in email message prioritization; it is used as the user feedback parameter which will be used in agent learning stage. Each time a set of new email messages downloaded, the agent will increases this counter by one for all unread email messages.
    c. Weight: It is an integer value represents the overall weight of each listed email message.
    d. Name: A string, of length 100 characters, holds the name of the email message file that saved in the local storage.
    e. Size: It is an integer value represents the email message size.
    f. Dat: It is a date variable, indicates when the email message is received to user email account in server area.
    g. ID: A string, of length 100 characters, holds the message ID, which is a unique ID used to find the target email message and

load its content and header's information in email's management process.

3.3.4 Email Management Component (EMC)

Prioritizing emails according to their personal importance to the user is another function offered by PAEA system. The degree of importance of an email to a single user could be assigned by a number; this number is considered as the significance weight of the message. Algorithm (3.6) illustrates how the EMC works. The main steps for assigning of the weight number are:

1. Load the record that contains all the necessary information for each unread downloaded email message, this will be done as the following: as all the downloaded email messages are saved in "Inbox" folder on the local storage, so by using the function "getfiles" all email message files names in "Inbox" folder are saved in an array. The email-records table, which contains information about the downloaded email messages, must be loaded into an array of records to be repeatedly used in email message management process.
2. Calculate the overall weight of each email message using the following factors:
    a. Address weight: if the email's sender address is saved in user's contacts list, the value of this contact (which is saved in value's field in contact's record) will be added to the overall weight of the email message.
    b. Website weight: in a similar way to that followed with the address weight value. If the email message is received from a registered website in contact table, the weight that is assigned to this website (which is saved value field in website's record) will be added to the overall weight of the email message.

c. Subject weight: if the subject's keywords of the email message are related to the interest subject list of the user, a bounce will be added to the overall weight of the email message (in this work the bounce is equal to 10).

Algorithm (3.6) EMC Algorithm

Goal: Prioritizing emails according to their relative importance to the User.

Input: Email table records.

Output: Updating the email-records table

Step1: Load emails' records from email's table in array of records.

Step2: Load the messages files names from "**Inbox**" folder to array of String "**files** ()" using "**getfiles**" function (Algorithm 3.3).

Step3://calculate the overall email weight

For each **element** in **files** array

find the email's record of the target file in email-records table

Check the email's read status (read's flag)

If read's flag= false then

Reset email weight to calculate a new weight

//Add address weight to email weight

Find the target contact address's weight from "Contacts Table"

Add address weight to email' weight

//Add website weight to email weight

Find the target website's weight from" Contacts Table"

Add website's weight to email's weight

//Check if the email' subject is related to the user interest subject list

Add subject's weight to email's weight

Next

Step4: Sort the email's records array in descending according to Email's overall weight field.

Step5: Update the email-records table.

For each unread email message, its overall weight value is calculated, the weight equation is:

MessageWeight=AddrWeight + WebsiteWeight + SubjectWeight

Where,

AddrWeight: is a number represents the address email's significance (weight) factor to the user.

WebsiteWeight: is a number represents the website's significance factor to the user.

SubjectWeight: is a number represents a bounce that will be added to the email's weight if the subject of the email is related to the interest subject list.

The overall weight value is saved in the email -record table.

In case that the email message is read, its weight value will reset to zero.

3. After the calculation of all email messages' weights; the email messages will be sorted in descending order according to their weight value; and then the messages are listed to the user from most significant one to less significant.

3.3.5 Email Services

A number of services are offered in PAEA; they are either necessary to accomplish agent's work, or to offers extra services for the user. The offered services are the following:

1. Contacts Module (CM)

The user has the required ability to create its own contacts list. This list holds all email addresses and websites which the user has the interest to exchange emails with them. So, the main function of contact module is to create a table that contains a record for each contact.

This module deals with two sets of data; **the first set** holds all known email addresses that had made messages exchange with them. Whenever the user wants to save a new email address in the table, a new record is created to hold this new address. The fields of this record are:

A. Name: a string of 20 characters that contains the name of the email address(es).
B. Addr, Addr2, Addr3: three strings of 25 characters length. They are used to hold up to three addresses for each contact. So, the user can save more than one email address for each contact name. This will increase the performance of the downloader component when a person, who sends email messages to user email account, has more than one email address, so, the downloader can deal with all email addresses at a time as one group of email addresses.
C. Val: A field of type integer, it represents the address weight; which will be used in the calculations of the overall weight of the email message. Each new address in the contacts list is assigned a significant initial weight value (in this work the initial value will be considered as 50). This value is continually changed according to the interaction between the user and the email messages incoming from that contact address.

The second set of data is used to hold some required information about the websites which are registered by user. These websites may send email messages to the user. The fields of this set are:

A. Address: a string of 30 characters, which contains the website address.
B. Dscr: a string of 30 character, it contains a brief description for the website.
C. Val: a field of type integer, it represents the personal weight of the website. This field is used to calculate the overall weight of the email message received from the website. When a new website is registered

as a new email source in the table of the agent system, it is assigned an initial value (in this work the initial value will be considered as 50). The system makes changes on this value according to the interaction between the user and the email messages coming from the website.

2. Email Browser (EB)

PAEA offers a simple Email Browser to present the text content of the email message, and to present the most important information about the email message (i.e., the sender's address and subject's title). Also, through this browser the user can extract the attachments from the email message and save them as separated files into the local storage; this could be done by using the "Attach_dwnloder" function. Algorithm (3.7) illustrates how the EB works. Algorithm (3.8) illustrates how the Attach_dwnloder works.

Another important job of the EB is monitoring the behavior of the user with email messages. When user opens the email message, EB increases the weight value of the contact (whether it is person or website) by one unit. If the email message received from unregistered email address (i.e., the email address that is not saved in user's contact list), then PAEA will automatically add this address to user's contacts list, if the user had frequently opened the email messages received from this address. PAEA use a binary file as a "History file" to save a record of any unknown email address that the user received an email message from it. The record consists of two fields:

1. Addr: a string of 30 characters holds the unknown email address.
2. Red: an integer holds the number of times that the user had received and read email messages from that email address.

When the user read more than three email messages from an unknown email address, it will automatically create a record for this email address

and add it to the user's contacts list. Algorithm (3.9) illustrates Add_contact function.

Algorithm (3.7) EB Algorithm.
Goal: Browse the email message. **Input**: **Path**: the path where the email message saved. **Output**: **Flag**: indicate if there are files attached to the email.
Step1: Create an object of "MailMessage" using "Rebex.Net.Mail.dll" function "New". **Message**= New MailMessage **Step2**: Load the email message using "Rebex.Net.Mail.dll" procedure "Load". **Message.load** (**Path**) **Step3**: Load and find the target email's record from email-records table. **Step4**: Reset the "**read**" field to "True" and the "**weight**" field to "0", to indicate that this email message had been read. **Step5**: Increase the contact weight (address weight or website weight). **Step6**: Show the **From, Subject** and **Body** fields of the email to the user using their corresponding Textbox objects. **Step7**: If there is file(s) attach to the email then set **Flag**=True Else set **Flag**=False **Step8**: If email's sender address is not registered in contact table then **add_contact** (**sender's email address, sender's name**) **Step9**: Return (**Flag**)

Algorithm (3.8) Attach_dwnloder Algorithm
Goal: Extract the attachment files from the email message. **Input: Path**: the path of the email message file. **Output**: None.
Step1: Select the folder's path where attachment files want to be saved. **Step2:**Create an object of "MailMessage" using "Rebex.Net.Mail.dll" function "New". **Message**= New MailMessage **Step3:** Load the email message using "Rebex.Net.Mail.dll" procedure "Load". **Message.load (Path)** **Step3**: Define attach as "**attachment**" class represents attachment email. **Step4**: For each **attach** in email attachments Using "Rebex.Net.Mail.Dll" procedure "save" to save the **attachment** file in the selected folder. Next

Algorithm (3.9) add_contact algorithm
Goal: add a new email address to contact list. **Input: Addr**: string, the new unknown email address. **Name**: string, the proposed contact name for **Addr**. **Output:** Update the history file and contact table.
Step1: Load content of history file in array of records (**history_records**). **Step2:** If there is a record for **Addr** in **history_records**. increase **Red** field in **Addr** record by one unit if **Red** =3 then create a new contact record (**Name**) for **Addr** Save **Name** in contacts table **Step3**: Update history file and contact table.

3. Email Composer (EC)

The "MailMessage" class, used in PAEA system, is established to create an instance of "MailMessage" to represent each new email message. PAEA offers Email Composer to set most important attributes in header and body fields of the email message (like; From, To, Subject, date, MessageID and the content of the email message). Also, the user can attach, up to three, files with each sent email message. Finally, after user completed composing the email message, it will be saved automatically in "Outbox" folder waiting to be sent to its recipient(s).

4. Auto-Reply Email Message Service (AREMS)

PAEA offers the capability of allowing user to compose a default email message, so whenever the agent download a new email message an auto-replay email message are send to the sender. This is a useful tool if the user does not have enough time to handle his email messages, but still want to inform the sender that his message is received and his real replay message will issued later.

First an instance of "MailMessage" class must be created by using "New" command, set its properties to desired values and then send it using "Send" command. Algorithm (3.10) illustrates how the Auto-Replay Email Message Service works.

Algorithm (3.10) AREMS Algorithm
Goal: Send auto-replay message **Input**: Message Fields **Output**: None
Step1: Create an object of "MailMessage" class **Step2**: Set email message's fields **Message.From**= user's email address **Message.To**= receiver's email address **Message.Subject**= email's subject title **Message.body**= email's content **Step3**: Send the email message

5. Filter Email Downloader Services (FEDS)

As mentioned previously, IMAP supports "Search" command. So, the user can search for email messages match a given criteria by the search method. It accepts a variable number of search parameters. Table (3.1) lists the search parameters that could be used to perform various types of search operations. Therefore, the user can adjust the downloader to download only the email messages that match a given criteria.

The adopted "Search" mechanism, in this work, filters in all email messages, matches a given criteria, and lists them in a temporary container of message objects. Then by using a "GetMailMessage" method the target email message is gotten. Finally, the procedure "Save" could be used, which is a member of the "MailMessage" class, to save the email message in the local storage.

Three types of filters are available to user, and could applied separately on its new incoming email messages, they are:

1. Specific Contact Filter (SCF): it is used to download only email messages received from a specified contact. All contacts (i.e., email

addresses) saved in the "Contacts Table" are listed in List- Box and the user can choose more than one contact, after the email messages are filtered the result for each contact is save in a separate folder named as the name of the contact. Algorithm (3.11) illustrates how the SCF works.

2. Specific Website Filter (SWF): it is used to download only email messages received from a specified website. All websites address saved in the "Contacts Table" are listed in List-Box, and the user can choose more than one website address; after the email messages are filtered, the address for each filter website is saved in a separate folder named as the name of the website. Algorithm (3.12) illustrates how the SWF works.
3. Specific Time-intervals Filter (STF): it is used to download only email messages received within specific time intervals. Algorithm (3.13) illustrates how the SWF works.

Table (3.1) The list of search method parameters

Search parameter	Description
From(address)	Messages contain the specified string in their From field.
To(address)	Messages contain the specified string in their To field.
CC(address)	Messages contain the specified string in their CC field.
Bcc(address)	Messages contain the specified string in their BCC field.
Subject(queryTerm)	Messages contain the specified string in their subject field.
Body(queryTerm)	Messages contain the specified string in their body.
FullText(queryTerm)	Messages contain the specified string in their headers or body.
Arrived(on)	Messages arrived at a specific date.
Arrived(since, before)	Messages arrived at a specific date interval.
HasFlagsAllOf(flags)	Messages with all the specified flags set.
HasFlagsNoneOf(flags)	Messages have none of the specified flags set.
Deleted	Messages whose Deleted flag is set.
New	Messages whose Recent flag is set and Seen flag not set.
Size(min, max)	Messages whose size within specified interval.

Algorithm (3.11) SCF Algorithm
Goal: Filter all new incoming email message and download only ones that sent from a specific contact address. **Input**: Contact record. **Output**: Update email-records table.

Step1: Select Folder ("**Inbox**")
Step2: For each **email address** of the contact
//passing (From(email address) ,HasFlagsNoneOf (Seen flag))
//parameters to "Search" command
Tmp= search(From (email address), HasFlagNoneOf(Seen flag))
If there is **email** in **Tmp** then GoTo Algorithm (3.14)
For each **email** in **Tmp**
Save **email** in local storage
Create a record for the email
If **Del_flag**= True then delete email from the server
Next
Next
Step3: Update email-records table

Algorithm (3.12) SWF Algorithm
Goal: Filter all new incoming email messages and download only the ones sent from a specific website. **Input**: Website record, **Output**: Update email-records table.

Step1: Select Folder ("**Inbox**")

Step2: //passing (From(**website address**) ,HasFlagsNoneOf (**Seen flag**))

//parameters to "Search" command

Tmp= search(From (website address), HasFlagNoneOf(Seen flag))

If there is **email** in **Tmp** then GoTo Algorithm (3.14)

For each **email** in **Tmp**

Save **email** in local storage

Create a record for the email

If **Del_flag**= True then delete email from the server

Next

Next

Step3: Update email-records table

Algorithm (3.13) STF Algorithm

Goal: Filter all the new incoming email messages and download only ones that sent from a specific time interval.

Input: Time interval (**since**, **before**)

Output: Update email-records table.

Step1: Select Folder ("**Inbox**")

Step2: //passing (Arrived(since,before) ,HasFlagsNoneOf (Seen flag))

//parameters to "Search" command

Tmp= search(Arrived (**since, before**), HasFlagNoneOf(**Seen flag**))

If there is **email** in **Tmp** then GoTo Algorithm (3.14)

For each **email** in **Tmp**

Save **email** in local storage

Create a record for the email

If **Del_flag**= True then delete email from the server

Next

Next

Step3: Update email-records table

Algorithm (3.14) FolderExist Algorithm
Goal: Change the default directory of the VB.Net to a specific directory. **Input**: DirName, Path: the path of the "Inbox" folder. **Output**: None.
Step1: Get the name of all subdirectory of "**Path**" **Step2**: For each **subdirectory** Compare (**subdirectory name**) with (**DirName**) If True then **Flag**=1 Else **Flag**=0 Next **Step3**: If **Flag**=0 then create new directory (**DirName**) **Step4**: Change the default directory of VB.Net to (**DirName**)

3.4 System Learning

Beside to autonomous transaction of email messages, PAEA can adjusted by its learning information and, consequently, updated its knowledge. This continual updating of system parameters keeps the system capable to produce rational decisions for each user over the time.

The interaction between the user and the system depends on user behavior (i.e., how the user acts with the email messages that the agent sort and manage them), so each time the user decide to open the email or not, PAEA will interact with this action as a feedback from the user. Therefore, the reaction of PAEA will be different in case when the previous email message send by some source are already opened or ignored.

- If the user reads the email:

  The system will automatically increases the contact weight whose send messages continually opened. If the user open unread email message, the system will increase the contact weight of its sender by one unit. Also,

if the user open more than one email message from unknown email address, the system will automatically add this address to the user's contacts list.

- If the user ignores (not read) the email:

  Each time the system download a new set of email messages from the user account on server to local storage, the agent will increase the "seen" field of the email record by one unit to all unread email messages. So, when the agent recalculate the overall weight to user's unread email messages, the EMC will decrease the weight of the old unread emails by an integer value, which is already saved in "seen" field.

  After three times of user ignorance for the same email the system will decrease the contact weight of its sender by one unit. Also, the user has the ability to change the contacts weights, all the user's contacts will be list in List-View. The user can make click on any contact and change its weight to a new value. This will give more flexibility to the work of the agent because at any time the user can update the contacts weight, which is an important factor the EMC depend on it in the email management process.

3.5 System Autonomous

After the user runs PAEA system and it is connected to user account; then after 30 minutes PAEA will automatically check:

i. "Outbox" folder to see if the user had composed a new set of email messages, and they are waiting to be sent to their destinations. If there message(s), the "Uploader Component" will be activated to send these email messages.

ii. "Inbox" folder, which exists in user's account area on the mail server, to know whether new email messages have arrived. Once PAEA find new email messages it will activate the "Download Component" in order to

download the new emails and archive them on local storage. After downloading, PAEA will set the "New Email" Flag to True.

The "Email Management Component" will periodically recheck the "New Email" flag to see if there are new email messages were downloaded; in case of finding new coming emails then this component will recalculate the overall weight of the user's unread email messages and to rearrange the email messages.

CHAPTER FOUR

PAEA Interfaces and Evaluation

Chapter Four
PAEA Interfaces and Evaluation

4.1 Introduction

In this chapter, the user interfaces of PAEA application and the functions of their components are illustrated. Also, the main steps should followed by PAEA user are given. For system performance evaluation, a sequence of email messages were sent to test mail account and the system interaction with this flow of messages is tested and its rationality is evaluated by some users and they address a score according to the degree of similarity between the system response and their expectations. Also, the time taken by PAEA system to download and upload a set of email messages is measured and it is compared with the average time taken by some popular client emails.

4.2 PAEA Interface

As shown in figure (4.1), the main window of PAEA consists of the following items:

- Folders list: This located on the left side of the window. It shows all folders that "PAEA" directory contain; which they hold all user emails. The main folders are:
    1. Inbox: it holds all emails that the PAEA downloads them from the user account server area.
    2. Outbox: it holds all emails composed by user, and which are waiting their turn to be sent to their destinations.
    3. Sent: after sending the composed emails, each sent email will be moved to this folder for archiving purpose.

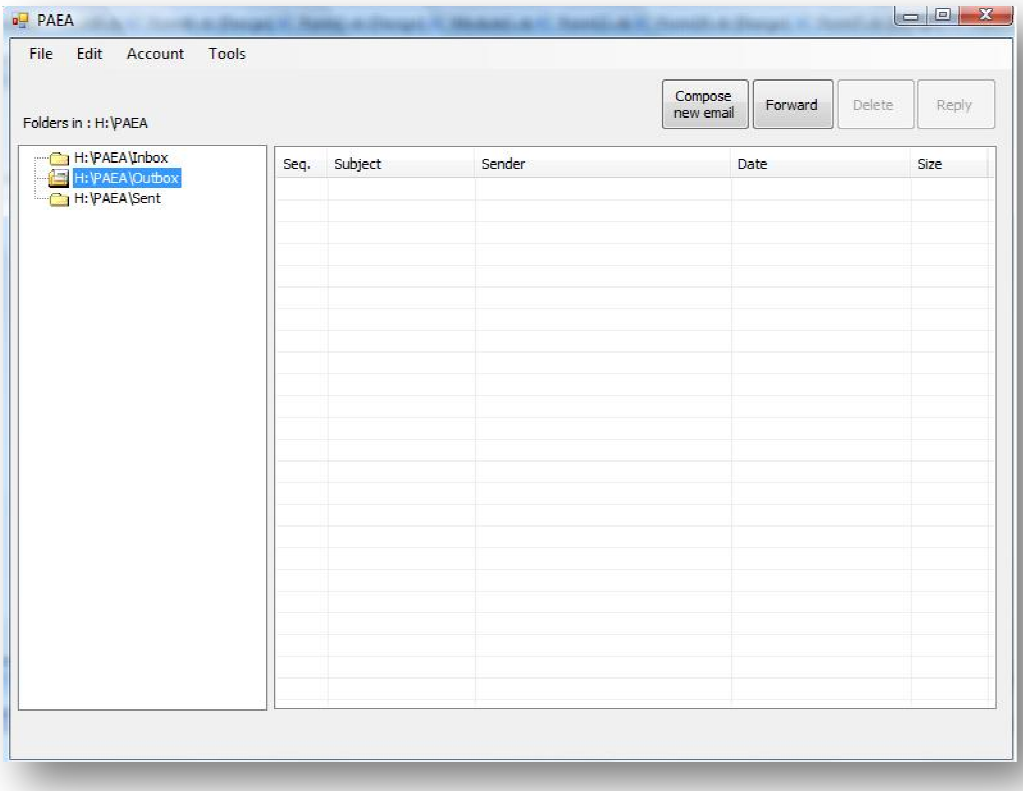

Figure (4.1) PAEA Interface

- Emails list: It is located at the right side of the main window. It lists all emails hold in a folder that preselected by the user. A set of attributes is shown for each listed email, these attributes are:
  1. Seq: show email message file name.
  2. Subject: show the email's subject.
  3. Sender: show the sender email address.
  4. Date: show the email receiving date.
  5. Size: show the email message size.
- Menu-strip: It is located at the upper side of the window. It contains the following dropdown items:
  1. File: it is consist of the following sub-menu options, as illustrate in figure 4.2:

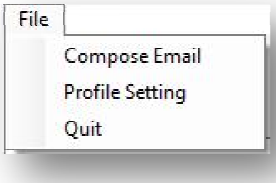

Figure (4.2) File Form

a. Compose Email: this command allows the user to compose a new email message through the "Email Composer", the composer will appear when user make a click on this sub-menu option.
b. Profile Setting: this command allows the user to create his\her profile and to adjust the way that PAEA system works.
c. Quit: to terminate PAEA system.

2. Edit: As illustrate in figure (4.3) it is consist of the following sub-menu options:

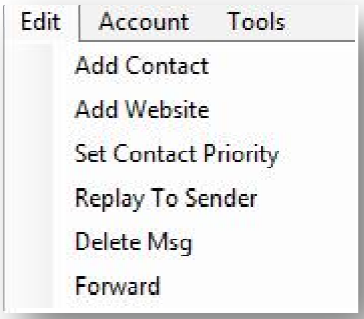

Figure (4.3) Edit Form

a. Add Contact: this command allows the user to add a new contact to his\her contacts list.
b. Add Website: this command allows the user to add the address of a website that the user will receive email messages from it.
c. Set Contact Priority: this command allows the user to manually update the contact's weight.
d. Reply To Sender: this command allows the user to compose a reply email message to a selected email message.

e. Delete Msg: this command allows the user to delete a selected email message and its corresponding record from the email-records table.
f. Forward: this command allows the user to forward a selected email message to a specific email address.

3. Connect: this command allows the user to establish a connection with the user account on the server in order to upload, download and manage the user's email messages.

4.3 PAEA User's Guide

When user run the PAEA system for the first time he\she must create his\her profile. To do this, the user should make a click on the "File" option which is listed in the PAEA menu-strip, after that he should make a click on "Profile" option; then the "User Profile" form will appear. As shown in Figure (4.4), the "User Profile" form consists of four sets of information to identify the user, they are:

1. Account information: this set of information is necessary to make an access to the user account on server. It is consist of:
   a. Email Address: it holds the email address of the user.
   b. Username: it holds the Account ID of the user's account.
   c. Password: it holds the password of the user' account.
2. Server information: this set of information is required to identify at which server the user account is located, and which protocol is used to transfer the email message to\from the user's account on server. This set of information consists of:
   a. Incoming Server Name: it holds the server address that PAEA downloads the user's email messages from it to the local storage.

b. Outgoing Server Name: it holds the server address that the PAEA system uploads the user's email messages from the local storage to it.

c. Interested Subjects: It is a list of the email subjects that the user is interested to read (in this work the user list consists of three interest subjects).

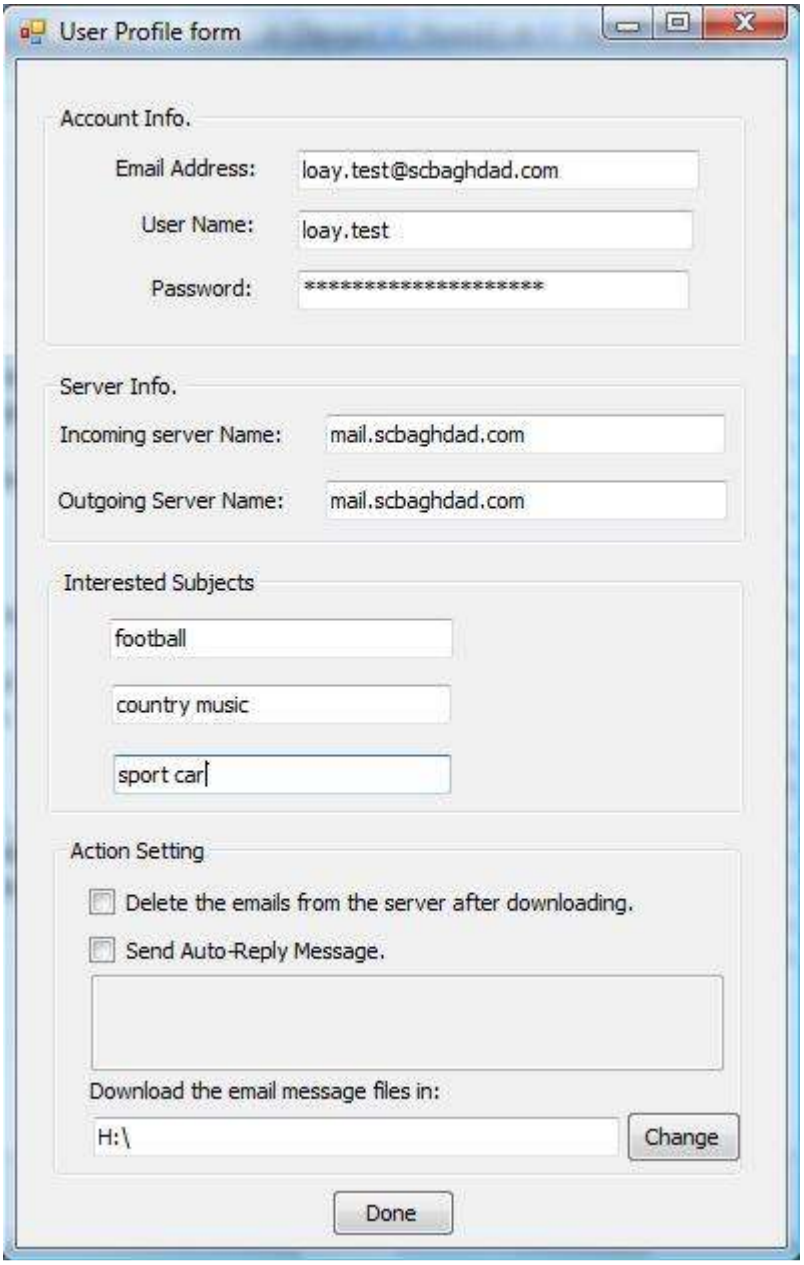

Figure (4.4) User Profile Form

3. Action setting: It is the set of information required to address how the system behaves at the cases:
   a. Leave, or not, a copy of the email messages in the user's account area on the server after downloading the new email message.
   b. Send an auto-replay email message or not; after the download instance of each new email message.
   c. Identify the directory name on the local storage to where the mail PAEA system's folders (Inbox, Outbox and sent) are created. When user doesn't identify the directory name then PAEA system will automatically create a default "PAEA" Directory on "C:" drive.

The user has the facility to update both "Interested Subject" and "Action Setting" whenever he\she want but after creating the user profile for the first time.

To complete user registration the user must add all his\her contacts addresses that he\she will exchange email messages with them, and all the websites that he\she registered in it. After making a click on "Add Contact", from "Edit" list, the "Add Contact" form will be shown, see figure (4.5). Through the "Add Contact" form the user can address the contact name and the list of his email addresses (up to 3 addresses). Each contact's name should not exceed more than (20) characters and for the email address not more than (25) characters.

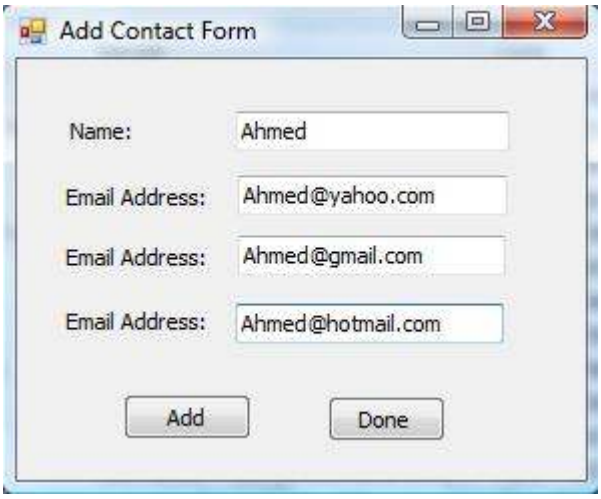

Figure (4.5) Add Contact Form

The user can manually reset the priority of his\her contacts by making a click on "set contact priority" from "Edit" menu list. The "Priority form" holds the whole user's contacts names listed in list view, as depicted in figure (4.6).

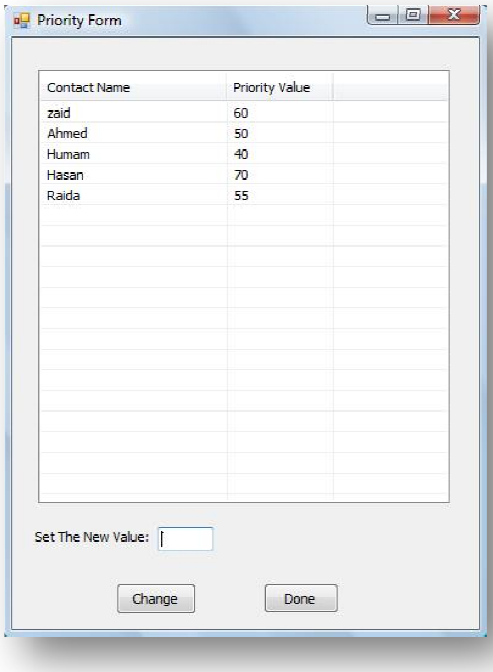

Figure (4.6) Priority Form

A click on a contact name followed by editing the type of new weight value in the textbox (i.e., within the range [1-100]), and followed by making a click on "change" button will lead to update the contact's weight.

The user can add a website URL address by making a click on the "Add Website" from "Edit" list, and then the "Add Website" form will appear. As shown in figure (4.7), the "Add Website" form offer for user the ability to edit the website address and he/she can add a description about each website. Both the website address and the description must not exceed more than (30) characters.

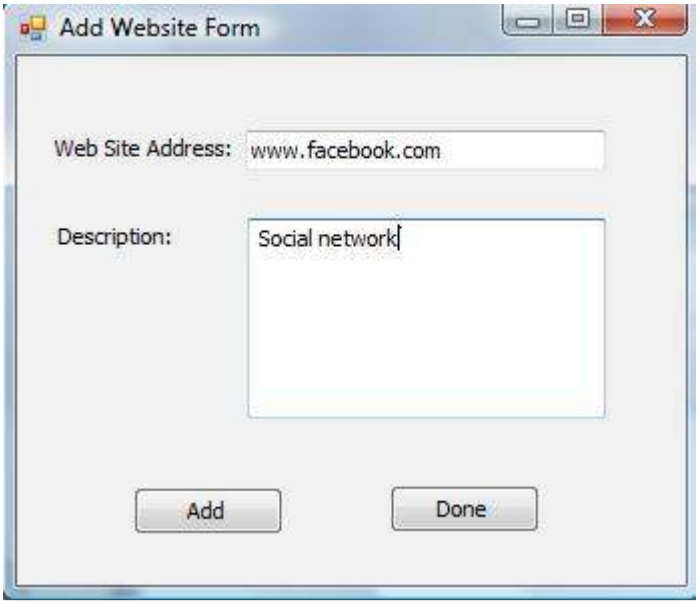

Figure (4.7) Add Website Form

The user can compose a new email message after he make a click on "Compose new email" submenu in the "File" list or on the "Compose new email" button in the PAEA main window, see figure (4.8). The following items must be defined by the user to compose new email message:

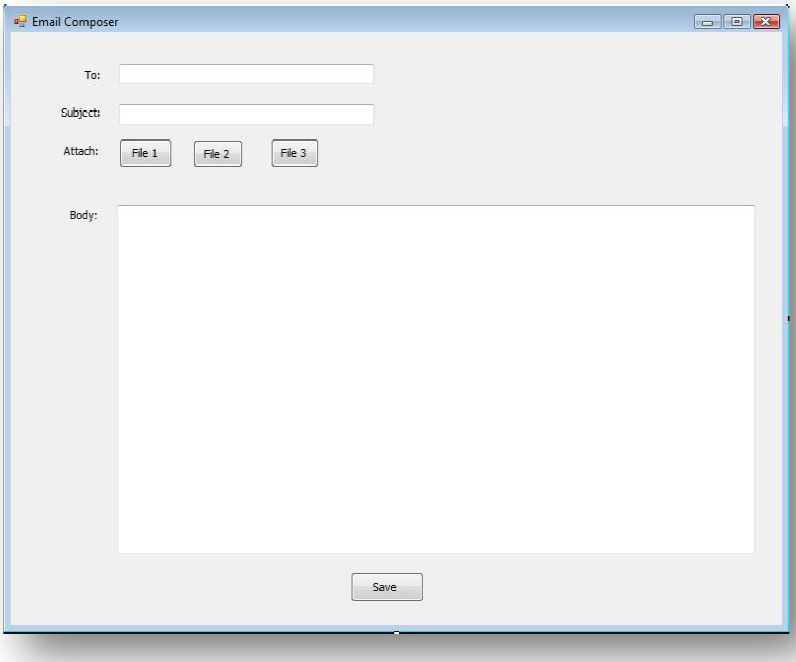

Figure (4.8) Email Composer

1. To: it holds the address of the email address' receiver.
2. Subject: it holds the subject of the email message.
3. Body: it holds the text-content of the email message.
4. Attach: The user can attach up to three files to his/her email message. When the user make a click on (file1, file2 or file3) the "Open File Dialog" will shown to allow the user to select the target file.

The "Save" button enable the user to save the email message in "Outbox" Folder.

To initiate the actual work of the PAEA system, the user must make a click on "Connect". Then the system check whether there are email messages waiting in the upload pool (i.e., "Outbox" folder), if "yes" then the system will upload them automatically. Also, the system will download the newly coming email messages in user account "Inbox" folder and save them on the local storage. After a new set of email messages are being downloaded the system will activate the "EMC" to sort the user email messages according to their

priority. After the email messages been downloaded and sorted they will appear in the system main windows, as shown in figure (4.9).

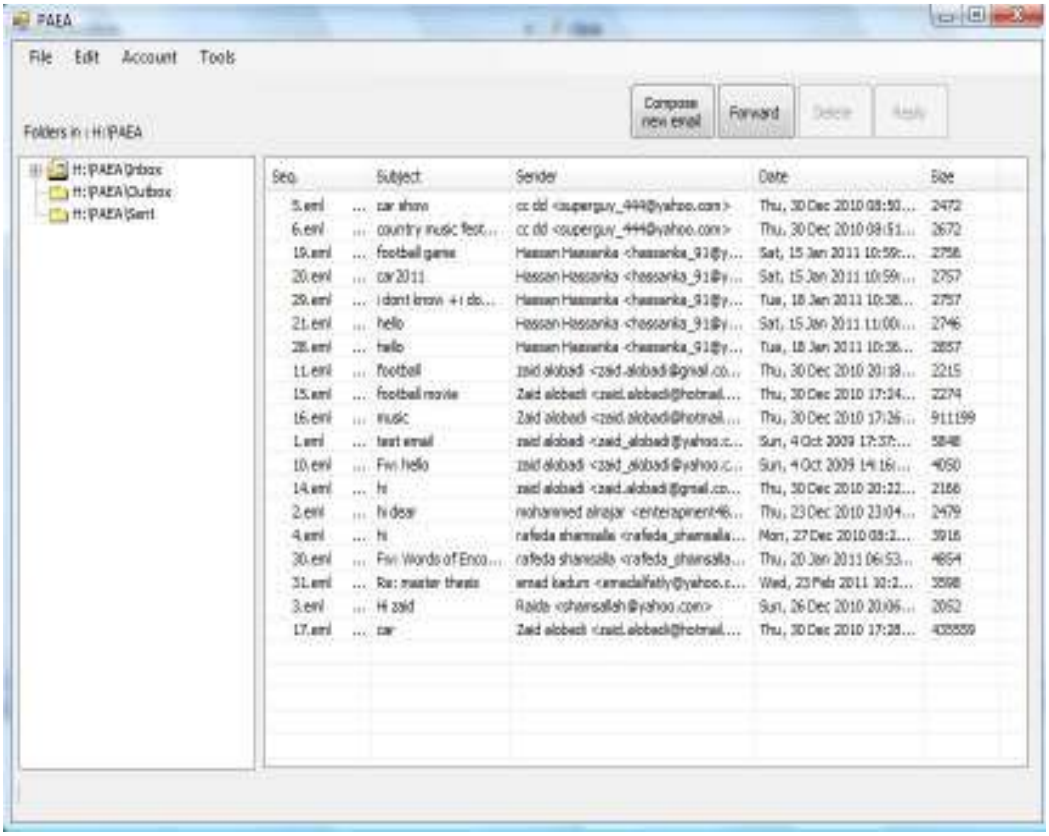

Figure (4.9) Email messages after been sorted

In order to open any email message the user must make a mouse click on the "Seq." field then the system will activate the "Email Browser" to show the content of the email to user, as shown in figure (4.10).

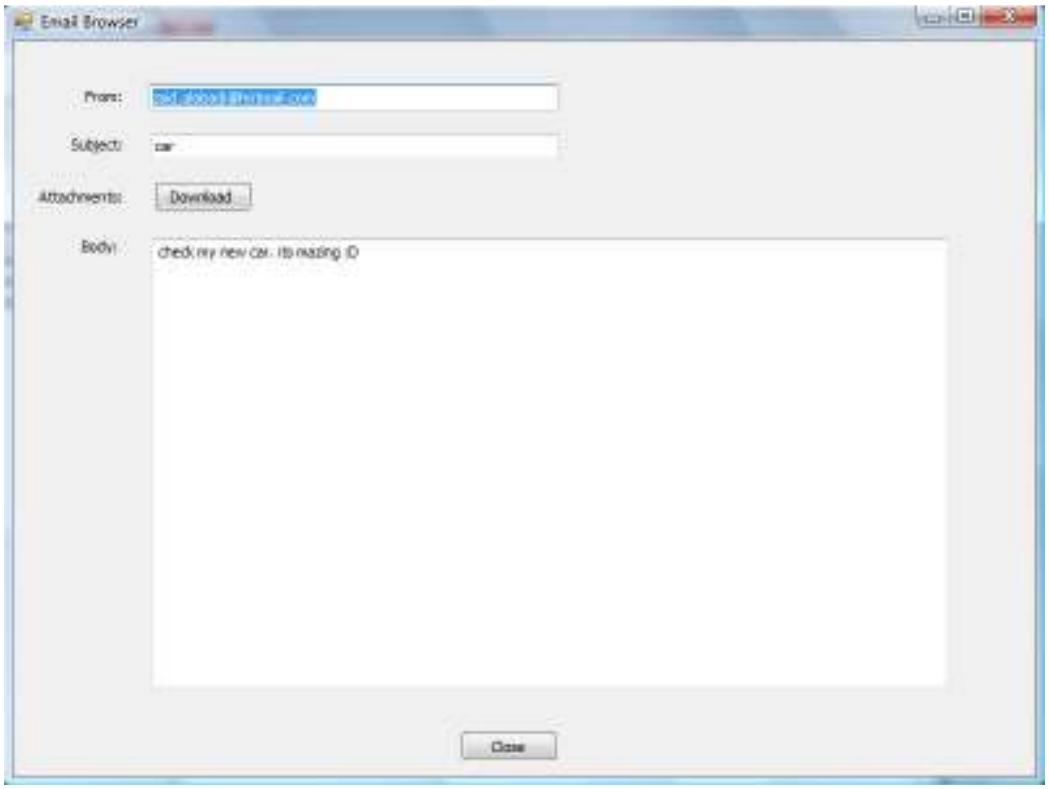

Figure (4.10) Email Browser

The Email Browser shows the following information about the selected email message:

1. From: it is the email address of the email's sender.
2. Subject: it is the title of the email message.
3. Body: is the text content of the email message.
4. Attachment: If there is a file (or files) attached to the email message then the "Download" button will be activated to enable the user to download the attachment. When user makes a click on "Download" button a "Folder Browser Dialog" is enabled to allow the user to select the path where he\she want the attachment be saved, see figure (4.11).

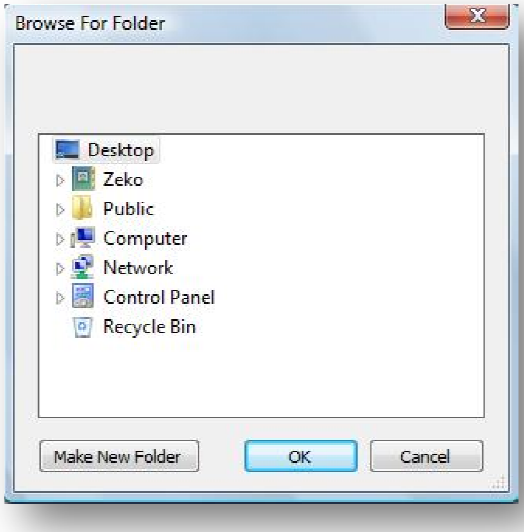

Figure (4.11) Folders Browser Dialog

The user has the facility to download only the email message that fit specific criteria. This could be done by making a click on the "Filter" from "Tools" list, then the "Filter" form is shown, see figure (4.12).

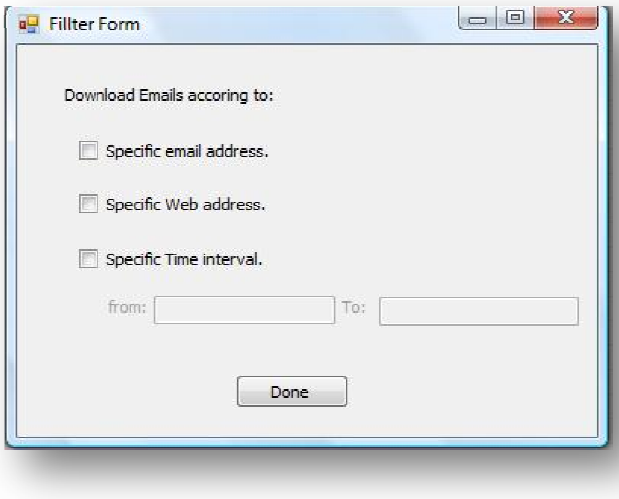

Figure (4.12) Filter Form

In "Filter" form the first checkbox "Specific email address" is checked if the user wants to download only the email messages received from a specific email address. After the user make a click on this checkbox a new form called "Contact Selection form" will appear , see figure (4.13), and it holds the user's

contacts addresses listed in checked list box to enable the user select which contact address he want to download its email messages.

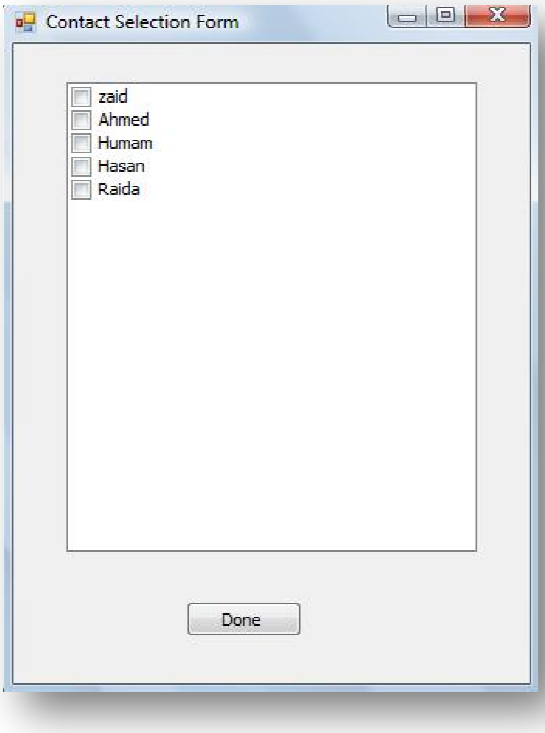

Figure (4.13) Contact Selection Form

The second checkbox "Specific website address" is checked if the user wants to download only the email messages that received from a specific website address. After the user make a click on this checkbox a new form called "Website Selection" form will appear, see figure (4.14). It holds the user website addresses listed in checked list box to let the user select which website address he want to download its email messages.

The third checkbox "Specific time interval" is checked if the user wants to download only the email messages that received within a specific time interval. Once the user make a click on this checkbox then two text boxes will be activated to let the user define the time interval (From, To).

Finally, when the user makes a click on "Done" button then PAEA will download the email messages and lists them in system main window.

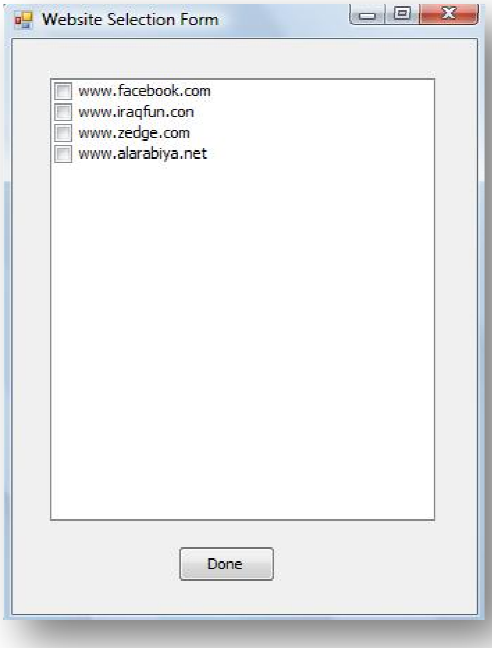

Figure (4.14) Website Selection Form

4.4 Simulation Case

A subject test through a simulated scenario of email flow was applied on the PAEA system to evaluate its work, and to see if its response is rational and match the user expectation. In this simulation a hypothetical user "Mr. X" is defined and assumes he has interests in (Football, Cars and country music). Also, all his contacts addresses added to the PAEA system database, as illustrated in table (4.1).

Table (4.1) Mr. X contacts List

Contact Name	Address
Zaid	Zaid.alobadi@hotmail.com Zaid.alobadi@gmail.com
Rafal	rafeda_shamsalla@yahoo.com
Suha	shamsallah@yahoo.com
Mohamed	enterapment485@yahoo.com

In this simulation we will track the steps of the main three components (DC, UC and EMC). For two iterations we will see the status of these components (active, wait or idle) and their effects to the user's email messages as shown in Table (4.2). Also, we notice that most of the email messages that related to Mr. X interested subjects are in the top of the emails list, and it show that the needed internet connection time is minimized (it need UC only 15 seconds to send 5 email messages, their total size is 585 Kbytes, comparing if the Mr. X using the webmail it took him 3 minutes to compose and send these email messages and it took DC 30 seconds to download 4 email messages, their total size is 913 Kbytes). Note that these results are depending on the internet connection speed.

Although email is ubiquitous, large and realistic email corpora are rarely available for research purposes due to privacy concerns. So, we use "Alshather Company" to test PAEA system and check its results; a questionnaire was applied on the company staff to check their opinion, and PAEA system get 8.0 (in scale from 1 to 10) average score.

Table (4.2) Simulation Result

#	UC Status	UC Job Description	DC Status	DC Job Description	EMC Status	EMC Job Description	Email Message		Read Status	Email Weight
							Subject	Sender		
1	Active	Sending 5 emails	Wait		Wait	No emails to manage				
	Idle		Active	Downloading 8 emails	Wait	No emails to manage	Funny football tricks Words of wisdom Vacations pics Amazing shopping offers Country music festivals Football movie Top10 country music 2010 2011 cars pics	Mohamed Suha Rafel Superguy_444 Superguy_444 Zaid Zaid Zaid	False False False False False False False False	N.D.* N.D. N.D. N.D. N.D. N.D. N.D. N.D.
	Idle		Idle		Active	Pertiorizing emails	2011 cars pics Top10 country music 2010 Football movie Funny football tricks Vacations pics Words of wisdom Country music festival Amazing shopping offers	Zaid Zaid Zaid Mohamed Rafel Suha Superguy_444 Superguy_444	False False False False False False False False	60 60 60 60 50 50 10 0
	Idle		Idle		Idle		2011 cars pics Top10 country music 2010 Football movie Funny football tricks Vacations pics Words of wisdom Country music festival Amazing shopping offers	Zaid Zaid Zaid Mohamed Rafel Suha Superguy_444 Superguy_444	True False True True False False True True	- 60 - - 50 50 - -

2	Active	No emials to send	Wait		Wait	No new email to manage	2011 cars pics Top10 country music 2010 Football movie Funny football tricks Vacations pics Words of wisdom Country music festival Amazing shopping offers	Zaid Zaid Zaid Mohamed Rafel Suha Superguy_444 Superguy_444	True False True True False False True True	- 60 - - 50 50 - -
	Idle		Active	Downloading 5 emails	Wait	No new email to manage	2011 cars pics Top10 country music 2010 Football movie Funny football tricks Vacations pics Words of wisdom Country music festival Amazing shopping offers Best football player My car pics Biz in Iraq I don't know I don't care RMD vs. BAR football match	Zaid Zaid Zaid Mohamed Rafel Suha Superguy_444 Superguy_444 Hassanka_91 Hassanka_91 Ahmed_jawad Hassanka_91 Mohamed	True False True True False False True True False False False False False	- 60 - - 50 50 - - N.D. N.D. N.D. N.D. N.D.
	Idel		Idle		Acive	Pertiorizing emails	RMD vs. BAR football match Top10 country music 2010 Vacations pics Words of wisdom My car pics Best football player I don't know I don't car Biz in Iraq 2011 cars pics Football movie Funny football tricks Country music festivals Amazing shopping offers	Mohamed Zaid Rafel Suha Hassanka_91 Hassanka_91 Hassanka_91 Ahmed_jawad Zaid Zaid Mohamed Superguy_444 Superguy_444	False False False False False False False False True True True True True	60 60 50 50 10 10 0 0 - - - - -

*N.D. is Not Determined

CHAPTER FIVE

Conclusion and Future Work

Chapter Five
Conclusion and Future Work

5.1 Conclusion

From this research, many points are noticed and concluded. The following are the most important ones:

1. One of the problems encountered during this work is the long waiting time spend by user during the send and download process of his\her email messages. So to solve problem PAEA workflow was designed to be in two phases: the first one is the online phase handled by the "email transaction" module, and the second phase is the offline phase handled by the "management" module. These two phases work in asynchronous way. So, while the "email transaction" module is monitoring the user account and do the required email messages uploading and downloading process, at the same time the user can read his\her emails and compose new emails without need to take care of the current email transactions.
2. There are two popular protocols (IMAP and POP3) for emails downloading. In this work the IMAP protocol was utilized for the following reasons:
   A. With POP3, the emails can be automatically erased from the server after they are downloaded; and this will free up the space of the user account. While, IMAP keeps all emails on the server till the user decide to erase them.
   B. POP3 doesn’t keep track of a variety of message states, it doesn’t allow the user to search through his/her mailbox, and it doesn’t even facilitate storing messages in a number of

mailboxes. With IMAP, clients not only have the option of operating in offline mode (where mail is downloaded and processed locally), but they allow storage and searching to take place on the IMAP server.

3. The most popular email providers (like, YAHOO and HOTMAIL) don't allow their free accounts to accessed via the protocols (IMAP or POP3). The user must have a paid account to be able to access his/her email messages through these two protocols. On other hand, other providers (like, GMAIL) allow their user to access their email accounts via IMAP or POP3 protocols.
4. When the keywords, mentioned in "Subject" field, not relevant to the email messages content, this will problematic if PAEA depend on "Subject" field to classify in which topics the email message belongs. So, to solve this problem, beside to the "Subject" field, the email message body will be taken into consideration.

5.2 Future Work

1. Add more features to PAEA (like; multiple user accounts, Import the user contacts from his/her email account to the user contacts list in the system).
2. The adopted email weighting equation is a linear combination of three email attributes (i.e., contact email address, website address and subject keyword). In this work, the weight of these three factors is multiply by a static value which equal to 1. An analysis for historical email usage data is required to characterize the behavior of the personal interactions with their incoming emails. This kind of study will be helpful to formulate a dynamic method for weighting the user email messages list.

3. Instead of making string-comparison between the predefine user keywords with the subject and body fields of the email message. The Agent system could be developed to be able to eliminate all the trivial words (like; a, the, he, is and etc) from the email message content then check the main keywords in the email messages with the system DB (which contains a set of lists, each one contains a combination of keywords belong to certain subject). And, the email message is classified into the subject whose list contains the maximums number of keywords in the email message.
4. In order to give more rational results for the user, the system must have more detailed profile about the user. To do that this task the system should keep track for all received email messages. Also, the email messages send by the user can give more clear perception about the user's interest subjects.
5. In each system, one of the important issues is the security; in PAEA the password of the user account must be protected to prevent that it may be stolen and used by unauthorized person. So, it is recommended that this information to be encrypted.

References

References

[AbMc01]

S. Abu-Hakima, C. McFarland and J. Meech, "An agent-based system for email highlighting", AGENTS'01 Proceedings of the 5th International Conference on Autonomous Agents, pp. 224-225, Montreal, Quebec, Canada, 2001.

[AnKo00]

I. Androutsopoulos, J. Koutsias, K. Chandrinos, G. Paliouras, and C. Spyropoulos. "An evaluation of naive bayesian anti-spam filtering, In Proceedings of the workshop on Machine Learning in the New Information Age", 11th European Conference on Machine Learning (ECML 2000), pp. 9- 17, Barcelona, Spain, 2000.

[And09]

P. Andy, "New in Labs: Offline Gmail" article, the official Gmail site, January 27, 2009.

[Bac07]

D. Backman , "Promises, Promises" article, Network Computing for IT magazine, 2007.

[BeMc04]

R. Bekkerman, A. McCallum and G. Huang. "Automatic Categorization of Email into Folders: Benchmark Experiments on Enron and SRI Corpora", in Technical Report, Computer Science department, IR-418, 2004.

[BrMe00]

C. Brutlag, J. Meek, "Challenges of the email domain for text classification", in 17th International Conference on Machine learning, pp.103-110, July 2000.

[BrEk98]

P. Brusilovsky, J. Eklund and E. Schwarz, "Web-based education for all: a tool for development adaptive courseware", Computer Networks and ISDN Systems, Volume 30, Issues 1-7, pp. 291-300, April 1998,

[Boo98]

G. Boone, "concept features in Re:agent, an intelligent email agent", in 2nd International Conference on Autonomous Agents, pp. 141-148, MAY 1998.

[ChHa96]

M. Chen, J. Han, and P. Yu, "Data mining: an overview from a database perspective", IEEE Transactions on Knowledge and Data Eng. 8(6):866--883, December 1996.

[ChCh02]

S. Chuang, L. Chien, "Towards automatic generation of query taxonomy: a hierarchical query clustering approach", in 2nd IEEE international conference on data mining, p 75, 2002.

[Coe95]

M. Coen, "SodaBot: A Software Agent Construction System", MIT AI Lab, technical report 1493, USA, 1995.

[Coh95a]

W. Cohen. "Fast effective rule induction", in Machine Learning: Proceeding of the 12th international Conference, pp. 115-123, Lake Taho, California, 1995.

[Coh96b]

W. Cohen, "Learning Rules that Classify Email", in proceeding of the AAAI Spring Simposium on Machine Learning in Information Access, pp.18-25, 1996

[Coo01]

C. Coolidge, "Best of the Web" article, Financial Planning, Forbes, 167(5), p. 84, spring 2001.

[Faw08]

H. Fawzi, "Design and Implementation of Email Filtering Agent", Master thesis, computer department, college of science, Al-Nahrain University, 2008.

[FrBo96a]

N. Freed and N. Borenstein, "Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies", RFC 2046, IETF, November 1996.

[FrBo96b]

N. Freed, N. Borenstein, "Multipurpose Internet Mail Extensions (MIME) Part Two: Format of Internet Message Bodies," RFC 2045, IETF, November 1996

[FrCa08]

M. Freed, J. Carbonell, G. Gordon, J. Hayes, B. Myers, D. Siewiorek, S. Smith, A. Steinfeld and A. Tomasic, " RADAR: A Personal Assistant that Learns to Reduce Email Overload", proceedings of the 23rd AAAI Conference on Artificial Intelligence, pp. 1287-1293, July 2008.

[GaDr01]

S. Gauthronet and E. Drouard, "Unsolicited commercial communications and data protection", Summary of Study Findings, commission of the European Communities, 2001.

[GrSi99]

D. Gruen, C. Sidner, C. Boettner and C. Rich, "A Collaborative Assistant for Email", proceedings of Human Factors in Computing Systems, pp. 196-197, New York, 1999.

[GwDi97]

S. Gwynne and J. Dickerson, "Lost in the e-mail", Time Magazine, pp. 88-90, April 1997.

[HaLu99]

S. Hambridge and A. Lunde, "Don't spew, a set of guidelines for mass unsolicited mailings and postings (spam)", RFC 2635, IETF, 1999.

[HuCh06]

S. Huang, Z. Chen, Y Yu and Y Maw, (2006) "Multitype features coselection for Web document clustering", IEEE Trans Knowledge Data Eng. 18(4):448–459, 2006

[KaTs05]

I. Katakis, G. Tsoumakas, I. Vlahavas, "On the Utility of Incremental Feature Selection for the Classification of Textual Data Streams", 10th Panhellenic Conference on Informatics, pp. 338-348, Volos, Greece, 11-13 November, 2005.

[KlYa04]

B. Klimt and Y. Yang, "The Enron Corpus: A New Dataset for Email Classification Research", Paper presented at the Machine Learning: ECML 2004: 15th European Conference on Machine Learning, pp. 237-248, Pisa, Italy, September 2004.

[KuLa05]

N. Kushmerick, T. Lau, "Automated e-mail activity management: an unsupervised learning approach", in 10th international conference on intelligent user interfaces, pp. 67–74, 2005.

[LaAo99]

B. Larsen, C. Aone, "Fast and effective text mining using linear-time document clustering", in: ACM SIGKDD 5th international conference on knowledge discovery and data mining, pp.16-22, 1999.

[Los99]

P. Loshin, "Essential email standards: RFCs and protocols made practical", John Wiley & Sons, Inc, November 1999.

[Mar01]

M. Markus, "Toward a theory of knowledge reuse: Types of knowledge reuse situations and factors in reuse success", Journal of Management Information Systems, 18(1), 57-93, 2001

[Moc01]

K. Mock, "An experimental framework for email categorization and management", in: 24th ACM international conference on research and development in information retrieval, SIGIR '01, pp 392–393, New Orleans, Louisiana, USA, September 9-12, 2001.

[MoWa03]

E. Moreale and S. Watt, "An Agent-Based Approach to Mailing List Knowledge Management", Papers from the AAAI Spring Symposium, pp. 49, Stanford, California, March 24–26, 2003.

[MuMu00]

D. Mullet and K. Mullet, "Managing IMAP", O'Reilly Media, 2000.

[MuJo99]

R. Murch, T. Johnson, "Intelligent Software Agents", Prentice Hall PTR; 1st edition, November 30 1998.

[Moo96]

K. Moore, "MIME (Multipurpose Internet Mail Extensions) Part Three: Message Header Extensions for Non-ASCII Text", RFC 2047, IETF, November 1996.

[MyRo96]

J. Myers and M. Rose, "Post Office Protocol-Version 3", RFC 1939, IETF, May 1996.

[PaEd97]

T. Payne and P. Edwards, "Interface Agents that Learn: An Investigation of Learning Issues in a Mail Agent Interface", Applied Artificial Intelligence, Vol. 11(1), pp.1-32, 1997.

[PaHo99]

J. Palme, A. Hopmann, N. Shelness, "MIME Encapsulation of Aggregate Documents, such as HTML (MHTML)", RFC 2557, IETF, March 1999.

[Par08]

C. Partridge, "The Technical Development of Internet Email", IEEE Annals of the History of Computing (Berlin: IEEE Computer Society) 30 (2): 3–29. (April-June 2008)

[Paz00]

M. Pazzani, "Representation of electronic mail filtering profiles: a user study", proceedings of the 2000 international conference on intelligent user interface, pp.202-206, New Orleans USA, January 2000.

[Pro02]

J. Provost, "Na¨ıve-Bayes vs. Rule-Learning in Classification of Email", University of Texas at Austin, Artificial Intelligence Lab. Technical Report AI-TR-99-284, 2002.

[Rad09]

The Radicati Group, INC, "2009 report", A Technology market research firm, www.radicati.com, 2009.

[Ren00]

J. Rennie, "ifile: An Application of Machine Learning to E-Mail Filtering", proceedings the KDD-2000 Workshop on Text Mining, Boston, 2000.

[Res01]

P. Resnick, "Internet Message Format", QUALCOMM Incorporated, RFC 2822, IETF, April 2001.

[Rud04]

I. Rudowsky, "Intelligent Agents", proceedings of AMCIS2004-Americas Conference on Information Systems, pp. 4588-4595, New York City, NY, August 5-8, 2004.

[RuNo03]

S. Russell and P. Norvig, "Artificial Intelligence: A Modern Approach", 2nd edition, Prentice-Hall Englewood Cliffs, 2003.

[SaDu98]

M. Sahami, S. Dumais, D. Heckerman & E. Horvitz, "A Bayesian Approach to Filtering Junk E-Mail. In Learning for Text Categorization", papers from the 1998 Workshop, pp.55-62, Madison, Wisconsin: AAAI Technical Report WS-98-05, 1998.

[SaSh05]

M. Sasaki and H. Shinnou, "Spam detection using text clustering", proceedings in International Conference on Cyberworlds,pp.316-319, 23-25 November 2005.

[SaWo97]

G. Salton, A. Wong and C. Yang, "Vector Space Model for Automatic Indexing", in K.S. Jones and P. Willett editors, Readings in Information Retrieval, pp. 273-280, San Francisco, Morgan Kaufmann, 1997.

[SeKe99]

R. Segal and M. Kephart, "Mailcat: An intelligent assistant for organization email", in Proceedings of the 3rd International Conference on Autonomous Agents, pp. 276-282, Seattle, WA, 1999.

[Sug07]

V. Sugumaran, "Application of Agents and Intelligent Information Technologies", Idea Group Publishing, January 30, 2007.

[WoJe95]

M. Wooldridge and N. Jennings, "Intelligent Agents: Theory and Practice", The Knowledge Engineering Review, 10(2), pp.115-152, 1995.

[Woo99]

D. Wood, "Programming Internet Email", O'Reilly Media, 1999.

[Xia08]

Y. Xiang, "Managing email overload with an automatic nonparametric clustering system", The Journal of Supercomputing, Vol. 48(3), PP. 227-242, July 2008.

[YaLi99]

Y. Yang and X. Liu, "A re-examination of text categorization methods", proceedings of the 22nd ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'99), pp. 42-49, Berkeley, California, USA, 1999 .

Appendixes

Appendix A: Mail Message (Rebex.Mail.dll)

It contains classes that enable you to create, read, process and save email messages

Name of Object	Type	Member of	Summery	Remark
MailMessage	Class	Rebex.mail	Represents an e-mail message that can be saved, loaded, processed or sent.	
Attachments	Property	Rebex.Mail.MailMessage	Gets the collection of attachments of this message.	
Bcc	Property	Rebex.Mail.MailMessage	Gets or sets the list of addresses of recipients that are not to be revealed to other recipients of the message.	
BodyText	Property	Rebex.Mail.MailMessage	Gets the text body of the message, or an empty string if it has no text body.	
Date	Property	Rebex.Mail.MailMessage	Gets or sets the origination date of this message. If origination date is not available, return the date specified by the first 'Received' header. The origination date specifies the date and time at which the creator of the message indicated that the message was complete and ready to enter the mail delivery system.	The origination date specifies the date and time at which the creator of the message indicated that the message was complete

From	Property	Rebex.Mail.MailMessage	Gets or sets the list of authors of this message.	This property specifies the the list of authors of the message. If this field contains multiple authors, the Rebex.Mail.MailMessage.Sender' field must be set to specify the agent responsible for the transmission of the message If this field contains a single author that is also the sender of the message, the 'Rebex.Mail.MailMessage.Sender' field should not be set.
Load	procedure	Rebex.Mail.MailMessage	Loads a mail message from the supplied file.	
MessageId	Property	Rebex.Mail.MailMessage	Gets or sets the unique identifier of the message.	
New	procedure	Rebex.Mail.MailMessage	Initializes an instance of an empty email message.	
Save	procedure	Rebex.Mail.MailMessage	Saves the mail message to the supplied file.	
Subject	Property	Rebex.Mail.MailMessage	Gets or sets the subject of the message.	This property contains a short string identifying the topic of the message.

Sender	Property	Rebex.Mail.MailMessage	Gets or sets the sender of this message.	This property specifies the agent responsible for the transmission of the message. Use Rebex.Mail.MailMessage.From to specify the list of authors of the message. This field must be set if the 'Rebex.Mail.MailMessage.From' field contains multiple authors. This field should not be set if the 'Rebex.Mail.MailMessage.From field contains a single author that is also the sender of the message.
Save	procedure	Rebex.Mail.AttachmentBase	Writes the content of the attachment or view into a supplied file.	
Add	Function	Rebex.Mail.AttachmentCollection	Adds an item to the end of the list.	
Count	Property	Rebex.Mail.AttachmentCollection	Gets the number of items in the collection.	
Attachment	Class	Rebex.Mail	Represents an e-mail attachment.	
FileName	Property	Rebex.Mail.Attachment	Gets the suggested file name of this attachment.	
DisplayName	Property	Rebex.Mail.Attachment	Gets the suggested display name of this attachment.	

Appendix B: IMAP (Rebex.Net.Imap)

It contains all the classes that are needed for the IMAP operations.

Name of Object	Type	Member of	Summery	Remarks
Imap	Class	Rebex.Net	Provides methods for communication with IMAP servers.	All members of this class are thread-safe.
Connect	Function	Rebex.Net.Imap	Connects to the IMAP server.	
CurrentFolder	Property	Rebex.Net.Imap	Gets the Rebex.Net.ImapFolder object that represents the currently selected folder, or null if no folder is selected.	Use Rebex.Net.Imap.SelectFolder (System.String) to select a folder and Rebex.Net.Imap.UnselectFolder to unselect it.
DeleteMessage	Procedure	Rebex.Net.Imap	Marks a message with the specified sequence number as deleted.	To actually remove messages marked as deleted, use the Rebex.Net.Imap.Purge method. This method represents IMAP STORE command.
Disconnect	Function	Rebex.Net.Imap	Disconnects from the IMAP server	
GetMailMessage	Function	Rebex.Net.Imap	Downloads the message with the specified sequence number and loads it into an instance of Rebex.Mail.MailMessage.	Uses IMAP FETCH command to retrieve the message.

Login	Procedure	Rebex.Net.Imap	Authenticates the user to the IMAP server using the specified authentication method.	This method tries to authenticate using the following methods, in this order of preference: CRAM-MD5, DIGEST-MD5, PLAIN, LOGIN, ClearText. The list does not contain the NTLM method - please use Rebex.Net.Imap.Login(System.String,System.String,Rebex.Net.ImapAuthentication) and specify NTLM directly if you wish to use it.
Purge	Procedure	Rebex.Net.Imap	Permanently removes all messages marked as deleted (with a Deleted flag) from the current folder.	This method represents IMAP EXPUNGE command.
Search	Function	Rebex.Net.Imap	Search the current folder for messages that match the specified searching criteria.	Searching criteria consist of one or more parameters. When multiple parameters are specified, the results is the intersection (AND) of all the messages that match.

SelectFolder	Procedure	Rebex.Net.Imap	Selects the specified folder so that its messages can be accessed.	The Rebex.Net.Imap.CurrentFolder property will contain an instance of Rebex.Net.ImapFolder that represents the selected folder after a call to this method. Only one mailbox can be selected at a time for a single connection, and only the messages in currently selected folder can be accessed. Also, most Rebex.Net.Imap.Notification only occur when a folder is selected. This method represents IMAP SELECT and EXAMINE commands.
ImapMessageCollection	Class	Rebex.Net	Provides a container for a collection of Rebex.Net.ImapMessageInfo objects.	
Count	Property	Rebex.Net.ImapMessageCollection	Gets the number of elements contained in the Rebex.Net.ImapMessageCollection.	

BodyText	Property	Rebex.Net.ImapMessageInfo	Gets the text body of the message, or an empty string if no text bodywas retrieved.	This field is only set if the Rebex.Net.ImapListFields options was specified in a call to Rebex.Net.Imap.GetMessageList(Rebex.Net.ImapListFields) or Rebex.Net.Imap.GetMessageInfo(System.String,Rebex.Net.ImapListFields).
From	Property	Rebex.Net.ImapMessageInfo	Gets the list of authors of this message.	
HasAttachment	Property	Rebex.Net.ImapMessageInfo	Returns a value indicating whether the message has an attachment (or more attachments).	This field is only set if the Rebex.Net.ImapListFields option was specified in a call to Rebex.Net.Imap.GetMessageList(Rebex.Net.ImapListFields) or Rebex.Net.Imap.GetMessageInfo(System.String,Rebex.Net.ImapListFields).
Length	Property	Rebex.Net.ImapMessageInfo	Gets the length of the message.	
ReceivedDate	Property	Rebex.Net.ImapMessageInfo	Gets the received date of this message.	The received date specifies the date and time at which the message was receied.

Sender	Property	Rebex.Net.ImapMessageInfo	Gets the sender of this message. May be null.	
SequenceNumber	Property	Rebex.Net.ImapMessageInfo	Gets the message sequence number.	
Subject	Property	Rebex.Net.ImapMessageInfo	Gets the subject of the message.	This property contains a short string identifying the topic of the message.

Appendix C: SMTP (Rebex.Net.Smtp)

It contains all the classes that are needed for the SMTP operations.

Name	Type	Member of	Summery	Remark
Smtp	Class	Rebex.Net	Provides methods for communication with SMTP servers.	All members of this class are thread-safe.
Send	procedure	Rebex.Net.Smtp	Sends a mail message constructed from the specified parameters.	Both sender and the list of recipients are extracted from the message. The sender address is taken from the "Sender" header field or from the first address in the "From" field if the "Sender" is missing. The list of recipients comes from "To", "Cc" and "Bcc" header fields. Also, the "Bcc" field is removed from message headers prior to sending.

الخلاصة

يعتبر البريد الالكتروني من اهم وسائط الاتصال من خلال شبكة الانترنيت، حيث انه يعتبر اداة فعالة لتنظيم المعلومات كونه وسيلة اتصال سريعه ودقيقه لتبادل المعلومات. ان الاستخدام الواسع للبريد الالكتروني تسبب في زياده كم الرسائل التي يجب على مستخدم البريد معالجتها بسبب اصبحت في بعض الاحيان تتجاوز طاقته.

ان ظاهرة الزيادة في عدد الرسائل الالكترونيه المستقبلة للمستخدم في عمله او حياته اليومية تسببت في خلق مشاكل جديده للمستخدم. حيث تطلب منه بذل جهد اضافي لمعالجة هذه الزيادة في عدد الرسائل، اضافة الى ان اصبحت تشكل تكلفة مادية اضافية لتنظيم وقراءة هذا العدد الكبير بالاضافة الى انها تتطلب الاتصال المستمر بشبكة الانترنيت. وعليه اصبحت هناك حاجة لبناء نظام يساعد المستخدم في عملية تنظيم وقراءة رسائلة الالكترونية لكي يتمكن من اكمال اعماله بسهولة.

التطبيق المقترح يساعد المستخدم بارسال رسائله الالكترونية الى مستقبليها اليا وتحميل كل الرسائل الجديدة من بريد المستخدم الى حاسبه الشخصي. اضافة الى ذلك يقوم التطبيق بتنظيم وترتيب الرسائل الالكترونيه بالاعتماد على السجل التاريخي لاهتمامات المستخدم وطبيعة استجابته للرسائل. كما ان التطبيق المقترح ومن خلال الاستخدام المستمر من قبل المستخدم يقوم بتحديث وتطوير عمله من اجل تقديم نتائج اكثر قبولا ومتوافقه مع توقعات المستخدم.

تم اختبار التطبيق من اجل الكشف على ان نتائجه تكون متوافقة مع توقعات المستخدم، كذلك تم اظهار الفوائد المترتبة من استخدام هذا التطبيق حيث انه قلص الزمن المتطلب للاتصال بشبكة الانترنيت. بالاضافة الى تقليص الجهد المتطلب من قبل المستخدم وذلك باظهارالرسائل التي تتعلق باهتمامات المستخدم باعلى القائمة بعد ترتيب الرسائل.

جمهورية العراق
وزارة التعليم العالي والبحث العلمي
جامعة النهرين
كلية العلوم

نظام وكيل المساعد الشخصي للبريد الإلكتروني

رسالة
مقدمة الى كلية العلوم في جامعة النهرين كجزء من متطلبات نيل
درجة ماجستير علوم في علوم الحاسبات

من قبل

زيد حيدر العبادي

(بكالوريوس ٢٠٠٦)

باشراف

د. لؤي ادورد جورج

د. سوسن ثامر

كانون الثاني ٢٠١١

صفر ١٤٣٢