Mobile MIME Agents: A Visual Approach to Multimedia Mail Authoring and Presentation

by

Marcel R. Karam

A Thesis Submitted to the
Faculty of Computer Science
in Partial Fulfillment of the Requirements for the Degree of

MASTER OF COMPUTER SCIENCE

TECHNICAL UNIVERSITY OF NOVA SCOTIA

Halifax, Nova Scotia

1997
The author has granted a non-exclusive licence allowing the National Library of Canada to reproduce, loan, distribute or sell copies of this thesis in microform, paper or electronic formats.

The author retains ownership of the copyright in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author's permission.

L’auteur a accordé une licence non exclusive permettant à la Bibliothèque nationale du Canada de reproduire, prêter, distribuer ou vendre des copies de cette thèse sous la forme de microfiche/film, de reproduction sur papier ou sur format électronique.

L’auteur conserve la propriété du droit d’auteur qui protège cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

0-612-31597-5
To my parents.....
TABLE OF CONTENTS

LIST OF FIGURES ............................................................................................................. vii
LIST OF ABBREVIATIONS ............................................................................................... ix
ACKNOWLEDGMENTS ........................................................................................................ x
ABSTRACT .......................................................................................................................... xi

1. INTRODUCTION ............................................................................................................. 1
   1.1 What is multimedia mail? .......................................................................................... 3
   1.1.1 Current Limitations ............................................................................................. 5
   1.2 Why Multimedia Mail? ............................................................................................... 6
   1.3 Some Existing Prototypes ......................................................................................... 7
       1.3.1 The Andrew and Eudora Systems ...................................................................... 7
       1.3.2 Montage ............................................................................................................. 8
       1.3.3 Multimedia Office System (MOS) .................................................................... 14

2. MULTIMEDIA MAIL STANDARDS: ISSUES AND DISCUSSIONS ......................... 20
   2.1 Background ............................................................................................................... 21
   2.2 Multimedia Mail Standards (Message, Media, and Document) ............................. 24
       2.2.1 MIME ............................................................................................................... 25
       2.2.2 MIME: Content-Types ...................................................................................... 26
       2.2.3 Encoding and Decoding with MIME ................................................................. 31
       2.2.4 Compression with MIME ................................................................................ 32
       2.2.5 Security with MIME ......................................................................................... 33
       2.2.6 MIME as a Standard ......................................................................................... 33
4.4 Authoring the Temporal and Spatial Synchronization ....................... 80
4.5 Implementation Details ..................................................................... 80
  4.5.1 MIME-Agent-Parser (MAP) ......................................................... 81
  4.5.2 MMAIDB .................................................................................. 82
  4.5.3 The “Orchestrator” ...................................................................... 83
4.6 Mail Presentation ............................................................................... 86
4.7 Document Flexibility ......................................................................... 87

5. AN AUTHORING AND PRESENTATION EXAMPLE .......................... 89

6. CONCLUSION .................................................................................... 95
  6.1 Future Work .................................................................................. 97

7. REFERENCES ..................................................................................... 99

8. APPENDICES .................................................................................... 105
  Appendix A - MIME Content-Types: A complete up to date list ....... 105
  Appendix B - Samples of the Prograph CPX code .............................. 108
LIST OF FIGURES

Figure 1.1 The MOS presentation window ................................................................. 19
Figure 2.1 An example of an SMTP message ............................................................ 22
Figure 2.2 An example of a MIME message ............................................................. 22
Figure 2.3 An example of an X.400 message ............................................................. 23
Figure 2.4 Message exchange between UA and MTA ................................................. 24
Figure 2.5 An example of a multipart/related MIME content-type ............................ 29
Figure 2.6 The X.400 message structure; adapted from [18] .................................... 34
Figure 2.7 Encapsulating the message in one body part ........................................... 45
Figure 2.8 Extended mapping .................................................................................. 46
Figure 2.9 The architecture of the MMMS ................................................................. 49
Figure 3.1 The Mobile MIME agents and their iconic representations ....................... 56
Figure 3.2 An overview of the authoring/composing models ..................................... 59
Figure 3.3 An example of a textual document format .............................................. 60
Figure 3.4 Attachments window .............................................................................. 61
Figure 3.5 ActiveM^3 mail structure ........................................................................ 65
Figure 3.6 A panel to compose and insert multimedia elements ............................... 67
Figure 3.7 The "backpack" view .............................................................................. 67
Figure 4.1 The Mobile MIME agents and their iconic representations ....................... 72
Figure 4.2 The interface of the authoring view ......................................................... 73
Figure 4.3 The document format in (a) and its sub-form in (b) .................................. 76
Figure 4.4 The Mobile MIME Agent mail structure ................................................ 78
Figure 4.5 A typical Mobile MIME Agent message format ...................................... 79
Figure 4.6 A sample MMAIDB database file ......................................................... 82
Figure 4.7 The general algorithm of the "Orchestrator" ........................................... 84
<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.8</td>
<td>The presentation in a MIME Agent mail tool</td>
<td>86</td>
</tr>
<tr>
<td>5.1</td>
<td>The dialog box that incorporates the media into the agents</td>
<td>90</td>
</tr>
<tr>
<td>5.2</td>
<td>Assigning a multipart for a generic container</td>
<td>91</td>
</tr>
<tr>
<td>5.3</td>
<td>The Multipart/parallel window composition</td>
<td>92</td>
</tr>
<tr>
<td>5.4</td>
<td>A sample mail presented in Eudora™ without the Java interpreter</td>
<td>93</td>
</tr>
<tr>
<td>5.5</td>
<td>A sample mail presented with the Java interpreter</td>
<td>94</td>
</tr>
<tr>
<td>7.1</td>
<td>The MIME agent classes in Prograph</td>
<td>108</td>
</tr>
<tr>
<td>7.2</td>
<td>The method calls for extracting MIME agent information</td>
<td>109</td>
</tr>
<tr>
<td>7.3</td>
<td>The method calls for parsing the document and sending the mail</td>
<td>110</td>
</tr>
</tbody>
</table>
LIST OF ABBREVIATIONS

ASCII  American Standard Characters International
CCITT  International Telegraph and Telephone Consultive Committee
EBP    X.400 External Body Part
FTP    File Transfer Protocol
GUI    Graphical User Interface
HCL    Hyperpicture Command Language
HTML   Hyper Text Markup Language
HTTP   Hyper Text Transfer Protocol
MAP    MIME-Agent-Parser
MMAIDB Mobile MIME Agent Internal Database
MIME   Multiple Internet Mail Extensions
MMMS   MultiMedia Mail System
MMTP   Internet Multimedia Mail Transfer Protocol
MMTSS  Multimedia Mail Transfer and Storage System
MOS    Multimedia Office System
MTA    Message Transfer Agent
NCM    Nested Context Model
ODP    Open Distributed Process
SMTP   Simple Mail Transfer Protocol
SMHMS  Simple Multimedia Hypermedia Mail system
URL    Uniform Resource Locator
UA     Use Agent
WWW    World Wide Web
WYSIWYG What You See Is What You Get
ACKNOWLEDGMENTS

With all my respect and gratitude, I would like to thank my supervisor, Dr. Trevor Smedley for his continuous support and guidance throughout the research work of this thesis. I am also grateful to Dr. A. Farrag of Dalhousie University for his constant encouragement and helpful advice.

I would also like to thank all the faculty members at the School of Computer Science (TUNS) and Dalhousie University who taught me in the Masters program.

Last but not least, I like to thank all my family members for their, help, support, and love.
Electronic mail (e-mail) has become one of the most useful and powerful tools for communication. Through attachments and similar primitive forms of mail composition, traditional e-mail systems are able to send and receive text, voice, or text/graphics that is often adequate to achieve, in principle, the objective of the mailing service. However, the introduction of multimedia and its widely spread applications gives rise to the need for a multimedia mail system that is capable of orchestrating the authoring, delivery, and presentation of unconventional media such as audio, video, and images. A multimedia mail system should provide the user with an environment where messages can be interactively authored and manipulated. It is also of paramount importance that a system that is capable of delivering documents of such diverse media presents them in an organized, flexible, and comprehensive fashion.

Multimedia mail has attracted a substantial amount of interest in recent years [5] [13] [25] [28]; however, with a wealth of prototypes and products that have demonstrated both feasibilities and capabilities of such technology, multimedia mail systems have yet to mature. The delay stems primarily from the standardization constraints of media components, Internet transfer protocols, documents, and multimedia message formats. Another important problem is the lack of tools and prototypes that allow “What You See Is What You Get” (WYSIWYG) manipulation during both the authoring and presentation phases.

This research examines multimedia electronic mail from a standardization point of view. It concentrates on the implementation of an authoring tool that will facilitate the authoring
process as well as, without resorting to format conversion mechanisms, simplify the problem of message navigation during the presentation phase in a multiplatform environment. This approach will help achieve a greater level of usability, simplicity, navigation, and understanding of multimedia messages.
1. INTRODUCTION

Electronic mail (e-mail), a central element in today's modern office, is composed of text or voice oriented messages widely used in communications; however, textual e-mail restricts itself to a single medium environment. This constraint prohibits users from exchanging multimedia documents containing text, audio, video, and other multimedia forms. To overcome this conventional e-mail deficiency, the concept of multimedia mail was introduced.

Multimedia mail is an e-mail service by which users can exchange electronic documents containing several media types via different communication channels. Unlike textual mail, multimedia mail systems can transmit multiple media such as text, audio, graphics, animation, and video sequences. With this definition in mind, one wonders why multimedia mail is only slowly coming into use. The main problem with multimedia mail stems from the heterogeneous nature of the infrastructure on which multimedia mail systems are implemented [25]. In general, a heterogeneous system has a set of different workstations and personal computers that operate on different platforms over the same network. Software and hardware differences like audio, video, and display devices have individually contributed to the difficulties of processing multimedia documents. Therefore, creating multimedia documents on one system often requires translations or conversions before they can be presented on different platforms or operating systems. Moreover, high speed networks with standard mail transfer protocols are also necessary for the success of this technology. Finally it is necessary to standardize a well defined message format that will help users compose, decompose, view, and navigate multimedia messages.
In Chapter 1, we discuss what multimedia mail is, its importance and benefits, and its current limitations. We then look at some of the existing prototypes that have been developed in recent years to give the reader an insight into how multimedia mail tools operate. In Chapter 2, we examine in detail the two leading Internet carriers; Multiple Internet Mail Extension (MIME) and X.400. Both of these carriers address the standardization issues of the message interchange format. We also look at the advantages and disadvantages of some of the research approaches to solving the issues of media and document format standards, storage capacity, and multimedia system architecture.

Chapter 3 looks at general multimedia authoring tools and draws a distinction between them and multimedia mail authoring tools. The distinction, as we will see in Chapter 3, can be attributed to factors, such as standard document format, interoperability, document flexibility, and the less interactive nature of multimedia mail in comparison with general multimedia applications. We then introduce three different multimedia mail authoring models: the Linear, Hypermedia, and Active models. In each model, the authoring and presentation styles are carefully examined.

In Chapter 4, we introduce the graphical user interface of our Mobile MIME Agent authoring tool, outlining the steps we have taken to ensure that the introduced authoring tool provides the user with an authoring model that is interactive, easy to learn and easy to use. We then describe the implementation of our approach, detailing the structure of the message interchange format, as well as the document interchange format that follows the metaphor of a paper-like format. The contents of the paper-like document format are orchestrated on the receiver's site with the help of two components that we have introduced. The first is a Java program which we call the "Orchestrator", and is based on a general algorithm, the second is a database which we call "MMAIDB" (Mobile MIME Agent Internal Database), which contains inputs to the "Orchestrator". The use of the
Java technology in our approach helps achieve a "WYSIWYG" presentation in a heterogeneous environment.

Chapter 5 shows an authoring example that reflects the use of our authoring approach in the area of collaborative work. Finally, in Chapter 6, we conclude with a summary and a brief discussion of the future and direction of multimedia mail.

1.1 What Is Multimedia Mail?

The term "multimedia mail" generally refers to the transmission of messages between two or more users that contain: text with composite effects; sound clips; video sequences; images with 2D or 3D graphical representations; animation; structured data such as spreadsheet documents; and the temporal and spatial relationships between the media. In addition to this brief definition, it should be noted that the current state of affairs in the field of multimedia mail authoring requires users (in most authoring tools) to repeatedly attach documents to the mail. Tools that allow interactive authoring, require users to have a certain level of skill and knowledge of temporal and spatial media synchronization to be able to accomplish a somewhat moderate task such as a simple synchronized slide show. It is our objective to address the different issues pertaining to authoring and presentation of multimedia mail. Our new approach to mail authoring and presentation is based on the following assumptions:

- One should not have to be a professional multimedia author to send and receive multimedia messages.
- Multimedia elements should be visible and easily manipulated at the user interface level during both authoring and presentation of the mail.
The message format should comply with MIME standards, and support the use of any MIME-capable mail reading tool. (see Appendix A for more information on MIME).

Media transmission is achieved by a simple upgrade of plain e-mail via attachments. Attachments are uncorrelated multimedia components that are "attached" to textual mail messages. The unconventional structure of attachments is the result of many complications facing the growth of multimedia mail. These complications are the result of conflicting message, media, and mail transfer protocol standards, and they will be examined in detail in Chapters 2 and 3.

Although not ideal, multimedia message attachments have the advantage of being simple and extensible. They allow a wide range of encoding standards and the inclusion of arbitrary documents such as those produced by spreadsheets, word processors, and other packages. MIME, an extension of Internet mail, was defined by an Internet Engineering Task Force Group. The group provided a simple and standardized way of representing, encoding, and delivering a wide range of media, including non-ASCII character sets. MIME specifications affect only the format of text messages, not the protocol on which they are delivered. MIME is an extension to Simple Mail Transfer Protocol (SMTP) which is the recognized standard for mail delivery.

Each media type in an attachment should have its own specific manipulation and rendering tool. These rendering tools are special software applications provided through intuitive user interfaces that facilitate the formulation of the temporal and spatial relations among the different components of the multimedia message [5]. Moreover, the interface of a multimedia mail system should provide the same flexibility as that offered by its textual e-mail counterparts. The interface, therefore, should enable the user to create, broadcast, retrieve, display, archive, delete, and graphically manipulate messages. In
addition, it is important to give more attention to the human factors involved in the interface design because of the more complex nature of the multimedia messages.

When creating or editing a multimedia mail message, users should be able to scan images, digitize audio and video segments, and subsequently import them into the mail document. The editing facilities should also support Cut, Copy, Paste for audio, video, text, and graphic segments, and Scale for images. Received messages should be seen in a visual index organized according to the user’s preferences. When replying to a message, the user should have the choice of using part of the message or creating a new message. In both cases, the user switches to the edit mode where facilities needed to compose or modify a message are available. Every multimedia Personal Computer (PC) or Workstation should be equipped with a bit-mapped high resolution screen, image scanner, audio input/output, and video production and digitizing units. A multimedia message, as the case in its textual counterpart, contains a number of basic parts: a unique message code; date and time stamp; recipient(s) list; information about the sender; and a message body.

1.1.1 Current Limitations

The current limitations of multimedia mail are message transport systems such as MIME or X.400, and the common problems shared with all multimedia systems. The difficulties that message transport systems encounter include large volumes of audio and video data that increase the size of multimedia messages. The size of such messages ranges from a few hundred bytes to several megabytes of data. Dealing with large data can lead to numerous problems ranging from waiting for the mail server to download the message to message segmentation. The latter can be handled using a mechanism to reassemble the parts of the message at the recipient’s site.
In principle, multimedia mail shares the common problems of all multimedia systems, including inadequate processing power and a shortage of memory. These problems can lead to poor quality presentation of both audio and video in the message.

From an authoring point of view, most multimedia mail authoring tools are not interactive, and do not provide feedback to the user during the authoring phase. The authoring model in most of the modern tools is based on attaching documents in a linear fashion to the mail file. Two problems arise from the current authoring model; first, in the authoring phase, users cannot compose hierarchical messages based on some temporal or spatial structure. Users also have no control over where the media elements are positioned in the mail document. Second, in the presentation phase, attachments, in the form of icons corresponding to a particular media type are manually retrieved, and there are no mechanisms to provide a multimedia presentation or playback.

1.2 Why Multimedia Mail?

In the last two decades, electronic mail has become the communication medium of choice. The growing demand and increased popularity of applications that support or create multimedia elements have forced the need to extend mail systems to support and exchange a wider range of media types. This method of exchanging information has numerous advantages over the conventional text-oriented e-mail. From the human senses point of view, textual e-mail is very limited. For example, it is virtually impossible to precisely describe a complex architectural drawing by exchanging textual mail; nor it is possible to convey facial and vocal expressions.

Moreover, exchanging multimedia mail that contains alphabets different from Roman is possible. This is done by writing the text in the language of the author’s choice, capturing
it in the form of an image (PICT or TIFF file), and then exchange that image in lieu of the original non-Latin text.

In many applications, concepts are better understood when presented in a mixed or multimedia mode [27]. Collaborative work, an area that is completely dependent on the exchange of different media across a network, is definitely enhanced through the use of multimedia mail. For example, if two or more people from different geographic locations are working on a particular project involving complex drawings, then exchanging ideas among them (pictures, some text and audio explaining one's approach) via multimedia mail services can help achieve a greater level of cooperation among project members.

1.3 Some Existing Prototypes

Numerous prototypes and products have emerged in the last two decades. They have individually shown the power of communication by multimedia mail systems. These systems include Eudora™, Andrew [14], Montage [24], and Multimedia E-mail System (MOS) [34]. We will briefly describe the Andrew and Eudora systems and examine in detail the major functions and deficiencies of both the Montage and MOS systems. The purpose of introducing these prototypes at this stage is to provide the reader with an insight into the use of modern multimedia mail tools and their support for message authoring, presentation, and navigation.

1.3.1 The Andrew and Eudora Systems

The Andrew Message System enables compound multimedia messages that include formatted text, pictures, sound, spreadsheets, animation, hierarchical drawings, hypertext links, and even embedded programs to extend the set of available media types. Interactive
authoring and presentation are poorly supported in Andrew. Presentation of media objects is accomplished by individually retrieving the attachments, thereby making the system unattractive from a multimedia presentation point of view.

Eudora's support for authoring multimedia mail messages is limited to attaching media elements to messages. Eudora supports the transmission of audio, video, complex drawings, and formatted text. However, from an authoring point of view, neither Andrew nor Eudora allows users to position media elements directly into the mail document. This approach to message authoring prohibits users from including the media elements in specific positions in the mail document. Moreover, the Andrew system and Eudora do not allow hierarchical composition. Hierarchical or nested composition of media elements gives users a mechanism to group a number of media elements that are temporally or spatially interrelated. Therefore, it is important to develop an authoring tool that will solve the authoring problems encountered in these systems through an intuitive interface that allows the average textual e-mail user to compose or author complex multimedia messages.

1.3.2 Montage

Montage is an experimental multimedia electronic mail system developed at the Georgia Institute of Technology Software Engineering Research Center. Four major design specifications influenced the Montage system:

- Custom multimedia message model: The distinction between multimedia messages and more complex general multimedia documents has led to the design of a custom message model which assumes that, unlike multimedia documents which have a
potentially large number of path through the information, Montage’s messages should proceed along one basic stream of control.

- Compatibility: Montage is built on top of existing lower-level mail routing protocols, and thus should work on most systems which support the UNIX operating system, X Windows, and Simple Mail Transport Protocol (SMTP) mail transfer system.

- Extensibility: By definition, multimedia mail should allow a wide variety of multimedia elements to be exchanged, thus Montage was designed with a high level of extensibility. Montage is completely runtime extensible by the user in the media domain. Users can choose their favorite tool for “recording” text. For example, users can choose a word processor or a spreadsheet to enter text into the mail. Similarly, users have their choice of tools for viewing received audio and video. The system provides some means for playing and recording certain simple media. However, because the system is not limited to those media that have built-in handlers, users can automatically add support for new media by specifying external programs to be used for playback and recording.

- Platform: While most e-mail tools are implemented on a variety of platforms, including PC’s and Macintoshes, Montage was implemented on a UNIX workstation running the X Window System. The platform was chosen because the developers felt that the interprocess communication abilities of UNIX and the high level of flexibility in X Windows can greatly simplify the implementation of Montage’s prototype.

- Message Model

General purpose multimedia documents tend to present the user with a large body of information in various multimedia types. The user then “navigates” through the document along a path that he or she chooses. General purpose multimedia system tend to be quite
interactive – the user is presented with almost all relevant information about a topic then browses the document to find interesting information. Contrast this model to a typical mail message in which the author is generally trying to convey some small number of central tenets. Unlike more general hypermedia documents, mail messages tend to be less interactive. The author has an idea to present and in some sense defines the reader’s path through the message at composition time.

Still, however, since various multimedia elements will be combined within a single message, navigating the multimedia message in Montage is achieved by categorizing the multimedia elements in a message into two classes: static or dynamic. The defining characteristic of a dynamic multimedia element is that the information presented changes with time. Examples of dynamic class elements include video and audio clips. Static multimedia do not have this temporal component. Examples of static class elements include simple text, formatted (rich) text, and still images.

To support the different characteristics in the multimedia element classes, Montage introduces the primary media class and the attachments that are explained in the next subsections.

- **Primary Media Class**

Each message in Montage has a primary media class. This class is either static or dynamic. The primary media class is the type of multimedia elements presenting the “main thrust” of the message to the reader. Various multimedia elements with the same class (either static or dynamic) as the primary media class may be combined freely. For example, if a message’s primary media class is dynamic, the author may freely intersperse audio and video in the message. All the components of the primary media class which compose the principal part of the message are collectively called the primary component.
The primary component is presented to the user in a single window. The window contains mechanisms for controlling the presentation of the primary components contained therein. In the case of dynamic media, the controls may be buttons for “play”, “pause”, “fast forward”, and “reverse”. In the case of static media the controls may be scrollbars to view parts of the message.

The concept of the primary component is to project a single “path” through the message. Although the primary component can be either static or dynamic, it cannot be a mix of these. Therefore, to increase the flexibility of the mail contents, Montage introduces the concept of attachments, allowing the user to include media which are not of the same type (static or dynamic) as the primary component.

- **Attachments**

Attachments, or sub-messages, are placed on any message of a given primary media class and can contain either static or dynamic multimedia elements. Attachments are not central to the message but still convey useful information. They are not fully presented to the user upon reading the mail; rather, they are represented by icons. Attachments give users a way to convey additional information while retaining an easy-to-playback, easy-to-understand main message thread. Attachment icons are connected to a particular location in the primary component message. Attachment icons are “connected” to a particular location in the primary component of the message. Attachments may be used to provide annotation or supplementary information to the central idea contained in the primary component.

In the case of static primary media, attachments are connected to a certain physical point in the message. Thus, a voice attachment may be connected to a particular line number in a text message. In the case of dynamic media, attachments are connected to a time range in
the primary message. Thus, as a dynamic message "plays back", the various attachments are presented to the user during the time they are relevant. Compared to ordinary mail programs, attachments in Montage serve as secondary information, while in mail tools such as ELM\(^1\) or Pine\(^2\), they may also serve as the primary media or central ideas.

- **Implementation and Interchange Format**

Montage provides full support for interchange of text, still images, audio, and video. The interchange format and its components are explained in the following sub-sections.

- **Interchange Format:** Montage uses a custom format for the bodies of e-mail messages. This format was created because Montage’s developers felt that existing standards were either inflexible in the types of media they allow, or were too unconstrained to present media in an understandable format appropriate for mail messages. The bodies of Montage messages are encoded into ASCII to allow transmission over SMTP.

In Montage, the message body consists of several discrete units called “chapters”. When mail is received by Montage, the body is separated from the header, converted to binary, uncompressed, and then broken down into its components. Each “chapter” is stored in a separate file. One special “chapter” is called the “table of contents” (TOC) which contains information on the layout of the entire message.

- **Chapters:** As stated before, a message consists of one or more message “chapters” along with a TOC. The TOC specifies the relations between the various “chapters” (relative placement within message, “chapter” format, etc.) Montage treats each “chapter” as raw data, thus it associates no real semantic information with each

---

\(^1\) ELM is a MIME compatible mail tool. Media are retrieved and viewed separately by external viewers.

\(^2\) Pine is a mail tool capable of understanding MIME, in a similar manner to ELM.
“chapter”. Instead, the “chapter” data is handed to a playback module or handler that is external to Montage. The handler invoked on a particular “chapter” depends on the multimedia type.

Multimedia types are identified to Montage by “tags” that are associated with each “chapter” via the TOC. Montage does not associate any meaning with “tags”; rather, they are defined by users of the system. When a “chapter” is encountered, the “tags” are searched and the user-specified handler is invoked on the “chapter”.

- **Table of contents**: The table of contents (TOC) is a special “chapter” that specifies the relations of the other “chapters” to one another. The format of the TOC is line-oriented ASCII format. Each TOC must have at least three records. Each record gives some piece of information and each appears on a separate line. The required records are:

  - `TOCVersion` which is the version identifier for the TOC format.
  - `Class` which is the class of the primary message component, either static or dynamic.
  - `Primary` which is the name and type of the message chapter containing a part of the primary component of the message.

Several other optional records may be included and they are: `attachment`, `author`, `CreationDate`, `Subject`, `ID`, and `Comment`.

- **Performance and evaluation**

The performance of Montage depends on the following interconnected factors: network; message size; and user’s habits and expertise in multimedia authoring. For example,
composing large unusual messages can cause both network latency and message fragmentation. It is therefore desirable to incorporate interface agents that will help users in constructing messages. Interface agents are semi-intelligent systems that assist users with daily computer-based tasks. Recently, various researchers have proposed a learning approach towards building such agents and some working prototypes have been demonstrated [42]. Such agents learn by “watching over the shoulder” of the user and detecting patterns and regularities in the user’s behavior.

From a design point of view, the custom built message format in Montage makes the tool unattractive for the purpose of interoperability because it does not follow standards. Since standards are not followed, traversal of messages across the Internet can lead to the loss information. Montage is not platform-independent. Moreover, if a multimedia mail is composed using Montage, is received by a different mail tool, retrieving and viewing the media mail can lead to a poor and misleading presentation. This is due to Montage’s custom built message format.

1.3.3 Multimedia Office System (MOS)

The Multimedia Office System (MOS) was developed by the Communication and Multimedia Laboratory of the National Taiwan University. Its architecture is an integrated environment of authoring, sending, receiving, and displaying multimedia documents. The architecture can be divided into two parts: the sending and the receiving sites.

- Sending Site

At the sending site, the user can prepare a new multimedia document, or modify an existing one, using separate media editors, and with the help of a simple Graphical User
Interface (GUI), import the result into the integrated environment. Upon completion of document authoring, the system scans the multimedia document and constructs an ASCII message format. The message is then sent through conventional e-mail.

- **Receiving Site**

Upon detection of incoming mail, the system retrieves the message, decodes it, and displays the contents of the message depending on the media type. The decoder displays the text type on the screen directly; however, the remaining media are represented by small proper icons. To activate a particular medium, the user clicks on the icon, the appropriate media player is invoked, and the contents of the media are then viewed.

- **Design and Specifications**

The MOS was built with the following design specifications:

- **Compatibility**: MOS has to be fully compatible with traditional e-mail so as to avoid interference with the e-mail of the operating system.

- **Integration**: The system has to be more than just a simple media composer; rather, it has to support an integrated environment and all of the functionalities of a text mail system.

- **Format Specification**

One of the main issues in developing MOS was to agree on a letter format. Two types of letter formats exist: The first, called *message format*, is an intermediate form used just for mailing. The second, called *document format*, is used for authoring documents. The major
difference between the two formats is that the media data must be put into a message for
delivery, whereas for an authoring document, how and where to get the media is the main
concern.

- Message Format: The MOS uses the MIME standard in its mailing message format.
  Therefore, the mailing message format contains in its header the following verbatim
text: MIME-Version 1.0. The line in the header denotes which version of MIME is
being used (in this case it is 1.0); it also indicates that the message conforms to RFC
1521[12] and RFC 1522 [37]. Next, the system uses a multipart/mixed (a MIME
content-type for grouping media elements in a mixed fashion) to bundle the whole top
level body part, and all other bodies with private header and body respectively, are
embedded in it. Every part in the mail is separated by a boundary. Boundaries are
parameters that define delimiters between body parts. Boundaries start on a new line
and consist of two hyphens followed by the boundary value. The final boundary,
which indicates the end of the last part, also has a suffix of two hyphens. Since the
MOS system follows the MIME standard, it supports the inclusion of external media
by referring to their File Transfer Protocol (FTP) sites.

- Document Format: The authoring document format is similar to that of the mailing
message except that each medium section is coded in a string indicating the type of the
medium, its storage location, and other information. The format of each medium
section in an authoring document is:

  %[Media Type[Audio or Image or Video] : [directory]
  filename — Access Type [Normal, Local-file, ANON-
  FTP, FTP] — IP-address — Caption ]
• **Conceptual Software Architecture**

A major problem during the design of a multimedia system, or of any large software project, is to avoid or minimize correlation among modules. MOS divides its modules into five subsystems:

• **Authoring/View module**: The authoring module is used to build, modify, and maintain a letter by adding, deleting, and replacing information in the letter. The Viewing module is used to extract information from the received letter.

• **Interface module**: This is the user interface of the MOS E-mail. It is constructed with the OPEN LOOK Intrinsic Toolkit (OLIT) and the X-Window system. The reason behind using a toolkit is to ensure portability to other GUI toolkits.

• **Media editor module**: This module is needed for creating and editing the media. It includes editors for sound, audio/video, images, and text.

• **Mail spool interface module**: This is a routine for manipulating the mail spool, including mailing letters, deleting letters, and detecting received multimedia mail.

• **Compose/Decompose module**: The function of this module is to control the data flow pipeline from the authoring document to the internal representation, then to the mailing message, and vice versa. This module parses the authoring document to get the global information of the media objects and constructs an internal database for the authoring document. On an individual level, the Compose module’s basic work is to scan the authoring document, convert the non-text data, and prepare a fixed format multimedia message for delivery. On the other hand, the Decompose module plays the inverse role of the Compose module. To add more flexibility to the system in terms of
convenient usage and time consumption when decoding, the following implementations were added to the Decomposing module:

- **Decoding on Demand**: Decoding is often time consuming, especially for video. A user might want to only survey the text and play few media in the document for the purpose of testing. The Decoding on Demand module first extracts all the text and records all media attributes, but decodes them only when a user selects the medium he or she wants.

- **Simple Player Media**: Instead of invoking the complete editor of a particular media, this module launches only the player that provides basic functions such as stop, play, pause, volume, and others.

- **Evaluation of MOS**

The style of the document authoring determines the multimedia system's "look and feel" [24]; however, MOS does not allow users to position media at arbitrary locations in the document. The mail is displayed in a form of text and icons in an integrated window depicted in Figure 1.1. It is necessary to develop a scenario to guide the user in navigating the mail to improve both usability and understanding of the multimedia mail.

Our authoring approach will allow media agents to be placed in any location in the document, thereby providing the user with a free style of authoring. Moreover, our approach also allows hierarchical authoring. For example, the user can place a folder in the document, place media elements in it, and subsequently assign a relationship between the media elements. This approach to hierarchical authoring in our opinion help users in both the authoring and the presentation/navigation phases.
In addition, our approach to mail presentation does not distinguish text from the rest of the media. For example, regardless of the number of text files received, the user is always presented with the text files in the shape of icons or agents in their appropriate positions (inside a container or as a separate media). This approach helps solve the ambiguity problem that could be encountered in MOS when more than one textual file are received and presented to the user in the same window. From an interoperability point of view, messages in MOS can be retrieved and viewed by any MIME-capable mail tool.

Figure 1.1: The MOS presentation window.
2. MULTIMEDIA MAIL STANDARDS: ISSUES AND DISCUSSIONS

Research and commercial prototypes have both demonstrated success in exchanging multimedia mail over homogenous environments; however, they have not yet succeeded to make it as a standard part of e-mail systems. Multimedia mail requires well defined and accepted standards for exchanging multimedia documents over the Internet. With this in mind, one should realize that standard interchange and presentation formats for message components (audio, video, graphics, text) are first required to ensure that received messages will actually be readable by their recipients, and to avoid conversion of multimedia types. Moreover, a standard message transfer protocol is needed to ensure that temporal, spatial, structural, and procedural relationships between the multimedia elements can be clearly and easily defined. Finally, a standard compression algorithm is needed to avoid message fragmentation over the network. Security issues are also a concern; however, they will only be briefly mentioned for they are outside the scope of work in this thesis. The above prerequisites are of paramount importance for the success of this promising technology.

In section 2.1 of this chapter, we provide the reader with a background on the differences in message formats between SMTP, MIME, and X.400. We then discuss the roles of the User Agent (UA) and Message Transfer Agent (MTA) in message transmission. In section 2.2 we divide the multimedia mail standard issues into three different categories: message exchange format; media interchange format; and document format. Sections 2.2.1 through 2.3.1 examine the message exchange format category by providing an in-depth discussion on both MIME and X.400. Section 2.3.1 gives a comparison between MIME and X.400.
The remainder of this chapter looks at some research approaches that have been taken to solve the standard problems associated with the three standard categories previously mentioned.

2.1 Background

To adequately understand the concepts behind e-mail, one should think of a postal office where each letter must have a specific format to ensure successful delivery. The format is usually composed of [name, street, city, state/province, and zip/postal code]. This format is designed to ensure that any agency in charge of handling this mail knows the destination of the recipient. Message transfer systems such as X.400, SMTP, and MIME operate in a very similar fashion. For example, SMTP follows the format defined in RFC 822 [21], and it uses the following format specifications:

- Message characters are standardized as a 7-bit ASCII.
- An information log is added to the start of the delivered message which indicates the path the message has followed.

Restrictions to the contents of SMTP can be summarized as follows:

- Transmission of executable files or other binary objects is not allowed.
- Characters represented by 8-bit codes cannot be transmitted.
- Messages have to be limited to a certain size, otherwise they are rejected by the server. An example of SMTP message is depicted in Figure 2.1.
Figure: 2.1. An example of an SMTP message.

MIME is an extension to RFC 822 [21], it was designed to solve the limitations of the SMTP standard, described in section 2.2.1. An example of a MIME message is depicted in Figure 2.2.

An Example of X.400 on the other hand, is depicted in Figure 2.3. It uses the X.400 envelope as its transport mechanism, while the heading and body parts are basic X.420 components that contain characteristics of the message header and the information being carried by the message body parts. X.420 is part of the family of protocols defined by X.400 [40]. X.420 is the Interpersonal Messaging User Agent Layer, it defines the services provided by the Interpersonal Messaging and the procedures for providing those services.
The previously mentioned examples depicted in Figures 2.1, 2.2, and 2.3 clearly show the differences between the MIME, SMTP, and X.400 messages structures. The differences in message formats creates a dilemma when exchanging multimedia mail messages on different platforms using different mail tools.

In e-mail methodology, the person sending and the person receiving a mail message are called the originator and the recipient respectively. Both originator and recipient use their local user interfaces to compose, manipulate, read, send, and receive messages. The interface capabilities are provided by the User Agent (UA) which is a software program that provides translation from the user interface to the local network, and vice versa.

As depicted in Figure 2.4, to send an e-mail, the UA submits the message to the Message Transfer Agent (MTA) which functions as the store and forward engine. The MTA is equivalent to individual postal offices, it provides the routing information required to reach the recipient’s MTA and UA subsequently. Once received by the UA, the message can be viewed by the local user interface.
2.2 Multimedia Mail Standards (Message, Media, and Document)

In the last two decades, mail systems have generally supported the exchange of textual messages in a heterogeneous environment. Recently, the growing popularity and increased number of applications that can create and utilize multimedia information raises the awareness of the possibility of a multimedia mail system that can exchange media information in a heterogeneous, multiplatform environment. After years of research and non-standard prototypes, multimedia mail has yet to become part of the Internet. To understand the standard issues facing the growth of multimedia mail over the Internet, one should examine both the capabilities and deficiencies of existing message transfer systems with respect to their support in exchanging multimedia documents. We divide the standard problems into three different categories:

- The message exchange format, such as MIME, SMTP, or X.400.
- The media interchange format, such as WAV or mu-LAW for a sound file.
- The document interchange format, such as ODA, HyTime, etc.

We examine the standard issues with regards to the message exchange by comparing the capability of both MIME and its main Internet opponent X.400. Second, we look at some proposed media interchange format standards. Finally, we examine the problems and proposed solutions in solving the document interchange format standards.
2.2.1 MIME

In order to appreciate the design of the Internet multimedia mail facilities, one must first recognize the need for a strong compatibility with the existing message formats in the Internet mail world. The message format as defined by RFC 822 [21] describes a message as a structured header followed by a single textual body that is limited to 7 bit US-ASCII characters. With this definition, messages containing more than one body, including non-textual media elements, could not be exchanged over the Internet. To overcome these limitations without both modifying all the mail servers and changing the basic model that defines the interchange message, MIME extended RFC 822 [21]. MIME is completely backward-compatible to SMTP, yet flexible and open to extension [23]. MIME provides a standardized way to represent and encode a wide variety of media types, including textual data in non-ASCII character sets, for transmission via Internet mail.

MIME was defined by the Internet Engineering Task Force Group. It is a fully available specification that offers a way to interchange messages containing multiple media elements and text in languages with different character sets. It supports several pre-defined types of non-textual messages such as 8-bit 8 kHz-sampled mu-LAW audio, GIF image files, and PostScript programs. Moreover, users are allowed to have external parts to be referenced; the message can contain a “pointer” to a part available via FTP or other means.

In MIME there are registered and unregistered types. Unregistered types are user-defined and they usually begin with an “X”. Their interpretation generally depends on private agreements between the originators and the recipients. There are seven valid registered content-types that will be discussed in detail in the next section. To see a complete list on MIME types and subtypes, refer to appendix A.
2.2.2 MIME: Content-Types

MIME contains seven main pre-defined content-types and numerous subtypes. The main types are: text; audio; image; video; application; message; and multipart. In general, a content-type declares the general type of data, and the subtype specifies a particular format for that type of data. For example, if the mail contains a GIF image, then the main type would be the token "image" and the subtype would be the token "gif", and they are represented in the mail by Content-Type: image/gif. The following is a detailed description of MIME types and subtypes:

- The token “text”, or default content-type, is intended for sending messages of textual nature. The primary subtype is “text/plain”; other subtypes such as “text/richtext” are intended for experimenting with multifont formatted text. A critical parameter used in the content-type field for “text/plain” is the character set that is specified with “charset” and might be included like: content-type: text/plain; charset = us-ascii.

- The token “image” indicates that the body contains an image. The subtype names are case sensitive for they represent specific image formats. Two initial subtypes are “jpeg” for the JPEG format, and “gif” for GIF format. The formal grammar for the content-type header field for data of type image is given by: content-type : = “image” “/” (“gif” / “jpeg” / extension-token).

- The token “audio” indicates that the body contains audio data. Although there is no consensus yet on a standard audio format for use with computers, there is a pressing need for a format capable of providing interoperable behavior. The initial subtype of “basic” is specified to meet this requirement by providing an absolutely minimal lowest common denominator audio format. It is expected that richer formats for higher quality and/or lower bandwidth audio will be defined in the future RFC 1521 [12].
The content of the “audio/basic” subtype is audio encoded using 8-bit ISDN mu-LAW. When this subtype is present, a sample rate of 8000 Hz and a single channel is assumed. The formal grammar for the content-type header field for data of type audio is given by: content-type: = “audio” “/” (“basic” / extension-token).

- The token “video” indicates that the body contains time-varying-picture images, possibly with color and coordinated sound. The term “video” is generically used, rather than with reference to any particular technology or format, and is not meant to preclude subtypes such as animated drawings encoded compactly. The subtype “mpeg” refers to video coded according to the MPEG standard. It is recognized that the so-called “video” formats include a representation for synchronized audio, and this is explicitly permitted for subtypes of “video”. The formal grammar for the content-type header field for data of type video is given by: content-type := “video” “/” (“mpeg” / extension-token).

- The token “message” is used for sending a mail message that encapsulates another message(s). For example, If the user wants to collect one or more received messages to a certain destination, then the use of this type is appropriate. The primary subtype, “message/rfc822”, has no required parameters in the Content-Type field. Additional subtypes, “partial” and “external-body”, do have required parameters. These subtypes are explained below.

  - the token “message/rfc822” (primary) subtype indicates that the body contains an encapsulated message, including header and body. It is not required that each “message/rfc822” body includes a ‘From’, ‘Subject’, and at least one destination header. It should be noted that, despite the use of the number “822”, a “message/rfc822” entity can include enhanced information. In other words, a “message/rfc822” message may be a MIME message.
- the token "message/partial" is defined in order to allow large objects to be delivered as several separate pieces of mail and automatically reassembled by the receiving UA. This mechanism can be used when intermediate transport agents limit the size of individual messages that can be sent. Therefore, "message/partial" indicates that the body contains a fragment of a larger message.

- the token "message/external-body" is used to include a reference number to bulk data residing on a server and is accessible via FTP.

- The token "multipart" specifies one or more different sets of data to be combined in a single body. The body must contain at least one body part. Each body part consists of a header, blank line, and a body source. All subtypes of "multipart" share a common syntax, therefore guaranteeing that all implementations can successfully break a "multipart" message into its component parts. There are several subtypes to "multipart", and some are described below:

- the token "multipart/mixed" is intended for use when the different body parts are independent, but need to be bundled and transmitted together. The body parts should be presented to the user in the order they appear in the mail message.

- the token "multipart/parallel" is syntactically identical to that of "mixed". It is intended for use when body parts in a message need to be simultaneously displayed. If using this subtype, composing agents should be aware that mail readers may lack this compatibility and may show the parts sequentially [12].

- the token "multipart/related" in RFC 1872 [32], is intended to represent compound objects consisting of several interrelated body parts. The presentation of the compound object starts at the root (a body part that gets executed before
the rest of the media parts in the compound object) and proceeds according to a
pre-defined relationship among the media parts that form the compound object.

```
Content-Type: Multipart/Related; boundary=example-1
start=":<950120.aaCC@Xlson.com>"
type="Application/X-FixedRecord"
start-info="-o ps"

--example-1
Content-Type: Application/octet-stream
Content-Description: The fixed length records
Content-Transfer-Encoding: base64
Content-ID: <950120.aaCB@Xlson.com>
T2xkIElhY0RvbmFsZCBoYWQgYSBmYXJtCkUgSSBFIEkgTwpBbmQgb24gaGlzIGZhc0GugGagaGFk
IHvbWUgZHVeja3MKRSBIEUgSSBPCldpdGggY29sb3IgY29sb3IgYWJvdXQgY29sb3IgY29sb3Ig
BmxWFjhayBzdWFRjayBoZXJjLaiphIFBHYWNrIHFJ
YWNrIHRoZXJjLaiphmdVyeSB3aGVyZSBhIHFJYW
NzIHF1YWnrCkUgSSBFIEkgTwo=

--example-1
Content-Type: Application/X-FixedRecord
Content-ID: <950120.aaCC@Xlson.com>
25
10
34
10
25
21
26

--example-1--
```

Figure 2.5: An example of a multipart/related MIME content-type.

The relationships among the media parts in a compound object are defined using
the following parameters: type; start; and start-info. To illustrate how these
parameters work, Figure 2.5 uses a single data block that the sender processes
automatically to generate the record length list. Consequently, the list appears
after the data. It should be noted here that UAs that do not recognize
"multipart/related" shall, in accordance with MIME, treat the entire entity as a
"multipart/mixed".
the token "multipart/report" in RFC 1892 [41], is a general family or container type for e-mail reports of any kind. It is used for producing delivery status reports and contains either two or three subparts in the following order:

1- Required. The purpose of this subpart is to provide an easy-to-understand description of the condition(s) that caused the report to be generated. This subpart is a human readable message.

2- Required. The purpose of this subpart is to provide a machine-readable description of the condition(s) that caused the report to be generated along with details not present in the first subpart.

3- Optional. This subpart contains the returned message or a portion thereof. This subpart may be useful in helping experts in diagnosing problems.

- The token "application" is used for data that does not fit in any of the other categories, and particularly for data to be processed by mail-based application programs. Expected uses of content-type applications include mail-based file transfer, spreadsheets, data for mail-based scheduling systems, and languages for "active" (computational) e-mail. This type will be used in our prototype to include a Java program that we call the "Orchestrator". The "Orchestrator" will be responsible for constructing the media mail on the receiver's site.

- the token "application/octet-stream" is the primary subtype of "application" and it is used to indicate that the message body contains binary data. It simply offers to put the data in a file, with any Content-Transfer-Encoding undone, or perhaps to use it as input to a user-specified process.
- the token “application/PostScript” indicates a PostScript program. The PostScript language definition provides facilities for internal labeling of the specific language features that a given program uses. The use of document structuring conventions, while not required, is strongly recommended as an aid to interoperability. The execution of general-purpose PostScript interpreters entails serious security risks, and implementors are discouraged from simply sending PostScript e-mail bodies to “off-the-shelf” interpreters. Message-sending software should not make use of nonstandard extensions; they are likely to be missing from some implementations. Message-receiving and message-displaying software should make sure that any nonstandard PostScript operators are secure and do not present any kind of threat.

2.2.3 Encoding and Decoding with MIME

Encoding message bodies is a process by which binary and raw data are converted into ASCII so that they can be delivered in MIME. Encoding is also a scheme that provides a reliable delivery across the largest range of environments. The following encoding schemes are defined in RFC 1521 [12]:

- “7bit” is the default encoding and is used for ordinary flat-ASCII text messages and formats, such as enriched and rich-text. The source must have lines of less than a thousand characters.

- “8bit” is similar to 7 bit in that line length limits may be imposed, but it allows for non-ASCII or extended character sets.

- “binary” encoding means that the message body can contain arbitrary binary data, without any regard for carriage-return or linefeed semantics. At this time, “8bit”
support is infrequent, and binary support non-existent, but in the future it may be possible to send binary files without converting them into ASCII.

- “quoted-printable” and “base64” are both encoding schemes that map binary images to RFC 822 [21] compatible text (short lines of 7-bit ASCII); however, they differ in that they are optimized for different types of data. The “quoted-printable” encoding is designed for text that is mainly composed of ASCII text, with small amounts of non-ASCII characters, such as PostScript files. The ASCII portion of the message is readable even if the receiver does not have MIME-compatible software.

- “Base-64” is used for pure binary data: integrity is insured and data is only expanded by about 33 percent. In comparison, “quoted-printable” encoding scheme can expand data by as much as three times in worst-case scenarios.

2.2.4 Compression with MIME

The need to encode in “base64” begs the presence of a compression algorithm to reduce the overhead that results from both including and encoding bulk data in mail messages; however, finding a truly interoperable compression algorithm can be difficult. This difficulty stems from both the uncertain legal patent situation regarding some commonly used compression algorithms and the overheads that they tend to introduce (large data space, the requirements that dictate that input should be treated as a stream of bits, etc.) Currently, there is no specified compression algorithm adopted by MIME because of the lack of compression expertise among the authors of MIME [15]. In an effort to correct this defect, work is being done to develop a compressed 64 encoding mechanism.
2.2.5 Security with MIME

Like any other network applications, mail traffic using MIME generally comes into host systems without being monitored or tested against any viruses. Messages, such as executable files, have the risk of propagating viruses. Mail users are advised to use secure computing practices and are warned against executing mail-based applications from within mail programs without the appropriate use of safeguards, such as virus protection programs [12].

Software applications are being developed to deal with the issue of security in MIME. Some of these applications include Secure/Multipurpose Internet Mail Extensions (S/MIME) that is a specification for secure electronic mail. S/MIME was designed to add security to e-mail messages in MIME format. The security services offered are authentication (using digital signatures) and privacy (using encryption).

2.2.6 MIME as a Standard

In conclusion, the e-mail community has been moving quickly to implement the new standards. A publicly-available implementation of MIME has been added to support over a dozen of the most common UNIX and DOS mail readers, including Berkeley Mail, Eudora, Elm, and Emacs. MIME allows e-mail sent via SMTP to contain multiple objects and information types; however, it lacks the proper representation of the structuring capabilities. All data types (audio, video, etc.) in MIME are encoded as 7 bit ASCII prior to submission. This encoding scheme increases the size of the message by 35% [5]. With the absence of a compression algorithm, the increase in message size adds to the problem of network delay and message fragmentation which is already caused by the inclusion of bulk data. The main advantages of MIME are the public domain implementation and the
integration with other Internet services. Moreover, MIME allows users to reference FTP sites and sends a reference in the message instead of sending the bulk data.

2.3 X.400

The International Telegraph and Telephone Consultative Committee (CCITT) (now the International Telecommunication Committee (ITU-T)) recommendation on X.400 has been defined to provide an electronic messaging service that is similar in principle to the current (hand-delivered) mail system. It consists of a family of protocols, each of which performs a specific function in relation to the complete message handling system. The structure of an X.400 message is defined in the Interpersonal Messaging System (IPMS) which in turn is defined in X.420. As depicted in Figure 2.6, the envelope is the X.400 transport mechanism while the heading and the body parts are basic X.420 components that contain the characteristics of the message header and the information being carried by the message (body parts).

Figure 2.6: The X.400 message structure; adapted from [18].
The standards on the nature of the body parts that can be included in an X.400 message were revised in 1984, 1988, and 1992. These updates are called X.400/84, X.400/88, and X.400/92 respectively. Since its inception in 1984, X.400 has included the ability to transmit multiple media in e-mail. The 1988 standard defined the following message body parts: ia5-text; voice; g3-facsimile; g4-class1; telex; videotex; encrypted; message; mixed-mode; bilaterally-defined; nationally-defined; and externally-defined. We will describe in detail the most used and useful body parts:

- ia5-text: this body part is used for sub-messages containing plain text (ASCII).
- voice: this body part is used for sub-messages containing 8-bit quality sound.
- g3-facsimile: this body part is used for sub-messages containing images or graphics.
- message: this body part is used to bundle one or more X.400 messages.
- mixed mode: this body part is a container or a folder used to group diverse media types in a mixed fashion. UAs retrieve the contents of this body parts sequentially in the order they are presented.
- externally defined: this body part is very useful in defining externally defined body parts. For example, X.400 does not support audio or video media types; the externally defined body part can be chosen to include either media type.

X.400 provides for the inclusion of several body parts of different representation forms in a single message; however, structural, temporal, and spatial relationships between the body parts cannot be expressed. Some have suggested that, since X.400 was designed with multimedia in mind, the demand for multimedia should simply be used to force the transition to X.400; this is an oversimplified view [15]. The established base of Internet mail users and the perceived complexity of X.400 create substantial resistance to such a transition, particularly in North America. Moreover, X.400 systems currently exist mainly as islands in a sea of Internet mail. In order to interoperate, X.400 mail must often go through gateways and then out to the Internet. Gateways are physical machines that
provide mapping from one message format such as MIME to another, such as X.400. Such gatewaying, in the absence of Internet multimedia mail standards, loses information because Internet mail has no standard representation for image, audio, or even non-ASCII textual data.

Despite the various updates to X.400 capabilities over the last ten years, widespread use of X.400-based multimedia mail has been limited. This is largely due to the fact that, although the X.400 defines many message body parts, it defines little in the way of content-type encoding [18]. Moreover, the absence of body parts that represent both audio and video has a severe setback to the widespread use of X.400 in the domain of multimedia mail.

2.3.1 Comparison: X.400 and MIME

MIME has the capability to include different media type representations (audio, video, text, graphics, and formatted text). In addition, it has the ability to use a file reference to an external body part that indicates only where and how to obtain that body part. That will be helpful, or even necessary, where file size limitations do not permit the transport of multimedia messages containing large amounts of data, such as video clip(s). This capability does not exist in X.400. Moreover, the inclusion of a linking mechanism to synchronize the body parts in time that allows the parallel or sequential execution of body parts is available in MIME. X.400, on the other hand, lacks this capability. However, it is important to note that although MIME defines a parallel execution through its multipart/parallel type, the UA may execute the body part in a sequential fashion. The main disadvantage with MIME is its lack of support to transfer raw data. For example, when using MIME, media parts (audio, video, and graphics) must be converted to ASCII using different encoding mechanism.
X.400 supports the transmission of text, graphics, and formatted text; however, no formal X.400 body parts have been defined to represent either audio or video. X.400 incorporates important features that MIME does not support, among them the ability to transfer data without encoding, the external body part (EBP) where information of any type can be represented. Moreover, X.400 provides guaranteed delivery, security and accountability of mail services.

In comparison, it is easy to see that X.400 lacks the representation of structural, spatial, and temporal relationships among message objects. This deficiency can be solved by allowing different document architectures such as ODA, MHEG, and HyTime to be included in the message (as an external body part). However, presentation and interpretation of such documents would be possible only if the recipient has the required software application to view and interpret the information in the document.

Once again the problem becomes the interchange format and standardization problems. The next section describes some research approaches that were taken to overcome the interchange format dilemma.

2.4 Multimedia Document and Media Interchange Format: Problems and Suggested Solutions

Different views and approaches have been proposed to solve standardization issues associated with multimedia mail. The solutions to these issues, as we have mentioned in section 2.2, are the key factors to the success of multimedia mail in a heterogeneous environment. This section examines the approaches taken by researchers to improve multimedia mail interoperability. Section 2.4.1 describes the “bottom-up” approach suggested by Borenstein [13] which is a rather naive approach that has been taken to address the media interchange format problems. Section 2.4.2 examines the approach
taken by Gay, Kervella, and Horlait in [25] that divides the problems of multimedia mail into five different categories: enterprise; information; computation; engineering; and technology. Section 2.4.3 describes the approach taken by Baveco, Kruithof, and Hoppner in [7] that defines the heterogeneous problem of multimedia mail to be related to either the document structure or the system architecture. Finally, section 2.4.4 introduces the Mediator, a mechanism introduced by Hofrichter, Moeller, Scheller, and Schurman in [28] to help solve the document interchange problem.

### 2.4.1 The Bottom-Up Approach

Borenstein, in the early stages of multimedia mail development, has suggested in [13] the possibility of obviating the need for a standard media interchange format, at least temporarily. In this approach which might be thought of as the “bottom-up”, multimedia mail users are not required to change their habits and begin using a new mail-reading program. Instead mail reading programs or UAs are themselves modified in a straightforward manner to notice when an “unknown” media type is encountered; the UA will call a single external program to display the message. The benefits of the “bottom-up” approach are at least threefold. First, at the cost of a small modification to the mail reading interfaces, this approach avoids the need to convince users to change their mail reading habits or the set of tools they use for authoring and editing messages. Second, by centralizing the handling of non-text mail in a single configurable program, this approach simplifies the administrative burden of administering multimedia mail in a heterogeneous environment. Third, by allowing the set of recognized data representation formats to grow and individually configured at each site.

This approach is somewhat naive for it does not take into consideration media conversion. For example, user A sends user B a media type that is “unknown” to user B’s platform. Then viewing that media type would require either a conversion program to
convert that media type and make it visible to user B using the locally installed facilities, or installing the appropriate software application to be able to view the "unknown" media type.

Another important point that this approach has overlooked is the standards in exchanging multimedia documents containing the actual presentations and orchestration of the media elements in the mail.

2.4.2 Multimedia Mail based on Open Distributed Process Framework

Gay, Kervella, and Horlait [25] studied the standards of multimedia mail using the framework provided by an emerging standard for Open Distributed Processing (ODP). The ODP provides a framework of abstraction based on five different categories from which various features of distributed applications can be studied. The five categories are enterprise; information; computational; engineering; and technology.

The enterprise category is concerned with the question of user-friendly interfaces. The information category is linked to the media formats, spatial and temporal information in the document, and the message transfer protocols that support the transmission of such information. The computational category is the question concerned with the application design and answers the question: "what needs to be done?". The engineering category is handled by operating systems and communication experts, and it answers the question of "how it should be achieved?". Finally the technology category is handled by the system engineers and it answers the question "with what?". The following sections take each category and study the standards that can be used.
- **Enterprise**

Implementation in this category is concerned with the design of friendly user interfaces where messages containing different media elements can be easily navigated, manipulated, and exchanged with a maximum number of users in a heterogeneous environment with a minimal delay. In addition, users may want to know what the recipient has effectively received because if a part of the message is missing, its meaning can be lost.

- **Information**

The information in the multimedia e-mail mainly corresponds to the interchange of multimedia messages that contain diverse media types with the possibility of synchronization relationships among the different parts. Another important information element is acknowledgment.

The existence of too many standards implies the need of conversion among the media objects that in turn requires time and introduces processing problems. The standards examined in the information category relates to the media and the message transport protocol standards that carries them.

- **Graphics:** A standard like JPEG is the first international compression standard for high definition color, or black and white pictures. Its compression ratio can reach 200:1.

- **Video:** MPEG is a generic standard for color video compression at low speeds. MPEG reached the International Standard status in 1995. The applications envisaged by MPEG are electronic publishing, video games, video mail, and multimedia application in general. H.261, also called P x 64, is a real time video
compression standard, its compression ratio ranges from 100:1 to 2000:1. Few applications use this standard, among them, videophone and videoconference.

- Audio: The audio situation is no different than that of the graphic one. There are standards like PCM (14-16 bits at 44 kHz for laser disks), and the PCM algorithm (8 bits at 8 kHz for telephone). The latter can be used for mail but it is four times longer than a suitable coding at 16 Kbits/s, for an equal quality.

- Information in X.400: X.400 supports text, facsimile, and other important body parts such as externally-defined. However, there is no explicit definition of body parts for video or audio.

- Information in SMTP: The message syntax is defined in RFC 822 [21]; it only allows for the transmission of textual body parts.

- Information in MIME: The message transfer protocol allows for the transmission of audio, video, graphics, text, structured data (spreadsheet), and animation. Media objects have to be converted to ASCII using transfer encoding mechanism such as Base64, quoted printable, and 8-bit. The need to encode into ASCII results in the increase of the message size by a ratio of 135%. MIME also supports the definition of temporal structure via its multipart parallel and multipart related content-type.

- Information in MMTP: Messages in the Internet Multimedia Mail transfer Protocol (MMTP) can contain different data types (text, voice, picture, facsimile, graphic, and spreadsheet). There is no formal body part for video.
• **Computational**

The required functionalities for the different part of a multimedia message is to be able to write a scenario to synchronize the parts among themselves. Another functionality example is for a user to be able to express the choice for message restitution such as quality of picture. Standards such as MHEG and HyTime can be used to support the required functionalities. However, it should be noted that the use of HyTime and MHEG does not guarantee that the UA at the recipient site will compute and process the information regarding temporal and spatial relations among the media objects.

• **Engineering**

The characteristics needed to support this category are guaranteed message delivery, low error rate, and mechanisms to code, compress, uncompress, and decode the media elements. The following standards are examined:

- X.400: The series of recommendations X.400 provides a good support for the message transfer.

- Internet mail: SMTP provides store-and-forward service for textual e-mail messages.

- MMTP: The MMTP protocol allows routing, transmission, and message delivery. It also supports reliable message transfer.

- MIME: Since MIME is built on top of SMTP, the message transport mechanism is not changed. The changes come from the special program that is invoked to display the message parts that are not text/plain. MIME also supports
message reassembly in the case where the message is big and was fragmented through the network.

- **Technology**

This category does not require the examination of standards because it looks at the physical support and configuration required to support the engineering category mechanisms.

The study of multimedia e-mail using the ODP framework of abstraction will allow a further study and modeling in each category by the corresponding specialists. For example, the application designer will be interested in the modeling of the elements of the computational category. This will facilitate the future integration of multimedia e-mail in an open distributed system together with other distributed applications such as multimedia conferencing systems.

**2.4.3 Maximizing Interoperability: Extended Mapping**

X.400 is widely used because it provides features such as delivery and non-delivery notifications, useful audit traces, guaranteed delivery, and built-in support for encryption. Many companies and governments departments have implemented, or are in the process of implementing, X.400-based messaging systems. Therefore, it is fair to conclude that X.400 mail tools are here to stay. On the other hand, the Internet is already a de facto standard in the Internet community [18], and it has established MIME as its multimedia mail standard. MIME authoring tools are becoming widely available. MIME-enabled UAs for basically all platforms are being developed around the world. In addition, numerous commercial e-mail tools are adding MIME capabilities to their products.
The interoperability that is supported by existing gateways between X.400 and MIME is defined in RFC 1327 [29], RFC 1494 [1], RFC 1495 [2], and RFC 1496 [3]. The followings is a summary on the mapping capabilities provided by the previously mentioned RFCs.

- RFC 1327 [29] defines the mapping between standard SMTP messages and X.400 messages. This mapping does not include support for messages containing multimedia elements.
- RFC 1494 [1] defines the equivalence between X.400 and SMTP.
- RFC 1495 [2] defines the mapping of message bodies between X.400 and SMTP.
- RFC 1496 [3] defines the mapping of message bodies and the rules for downgrading messages from X.400/88 to X.400/84 when MIME content-types are present in the messages.

Carrier and Georgnas [18] took the mapping mechanism provided in RFC 1327[29], RFC 1494 [1], RFC 1495[2], and RFC 1496 [3] one step further. Their approach provides standard multimedia support for X.400 and interoperability with the Internet without any modification to the existing X.400 infrastructure. To achieve the goal of their project, Carrier and Georganas decided to examine the following considerations:

- Use of a proprietary format that would include all information and all media and be read by specific software tool.
- Use of a set of newly defined body parts that would allow the transport of the information in an open fashion.
- Use of the MIME message format and content-types (the equivalent of body parts in X.400) for encapsulation in an extended body part (EBP, body part).
- Use of the MIME content-types for mapping in EBPs.
In the case of the use of a proprietary file format, any multimedia authoring tool can be used to assemble a presentation that would contain all the media in addition to the information regarding the synchronization in time and space. This option is easy to implement, however, it will defeat the purpose of standards. The standard is meant to provide message service to users in a heterogeneous environment. This approach requires the recipient to have certain software applications that might not be available at their platforms.

In the case of the use of newly defined body parts, the approach would be to select the best encoding schemes and make them the new body parts. Although this approach will provide independence of existing schemes and standards, interoperability with the Internet can be limited. Moreover, standardizing the newly defined body parts can be an issue. In the case of the use of MIME format, two approaches are taken: encapsulation and extended mapping. Encapsulation is the process of encapsulating a whole MIME message in a body part as depicted in Figure 2.7, and extended mapping is the process of mapping all MIME content-types in corresponding body parts as depicted in Figure 2.8.

![Figure 2.7: Encapsulating the message in one body part.](image-url)
Each consideration is assessed based on the following factors: gateways or physical machines that provide mapping mechanism between X.400 and MIME; tunneling (when a message of a given type goes through a system of another type by being encapsulated within a message of the other type); performance; ease of implementation; and interoperability.

- **Encapsulation**

As depicted in Figure 2.7, encapsulation is achieved by taking the whole message of one type and placing it in a new message of the other type. Existing gateways will not be able to perform such a task without modifications. Existing gateways can take various parts of an X.400 message and replace them by the appropriate parts in an Internet message, but they cannot take one part only and create a new message. Encapsulating could be faster since the body parts are not modified part by part. This performance improvement can be overlooked and outweighed by the size increase in the message due to the maintaining of
the transfer encoding since the message might have to traverse a gateway whose mail tool is MIME-supported.

Interoperability is not well served by encapsulating. Although MIME is not a proprietary format, it is different than X.400. Therefore, if someone sent a multimedia message to another X.400 user, the recipient could have some problems. The problems stem from the fact that the whole message is encapsulated in one body part, and retrieving even a plain text could be problematic.

- Extended Mapping

Extended mapping takes the standard mapping mechanism provided by RFC 1327[29], RFC 1494 [1], RFC 1495 [2], RFC 1496 [3] (previously described) one step further by introducing the mapping of audio and video parts, that are not supported by the above mentioned RFC's, from MIME content-type to X.400 body part. Both the audio and video MIME content-type are mapped into X.400 EBPs. As previously mentioned, extended mapping would require significant amount of work, and in the case of tunneling, it might be done twice.

In the case of implementation, the need to develop a composer/reader for the newly defined body parts (audio, video) that are not yet available in X.400 might be necessary. Viewing the newly defined parts is made possible by using "mail cap". The concept of mail cap is to use a specific file to inform multiple mail reading UA programs about the locally installed facilities for handling mail in various format.

In the case of interoperability, extended mapping would be the ideal situation. All the contents of the X.400 message would appear in their native X.400 format therefore
making the retrieval of any message body part possible regardless of the multimedia capability of the recipient.

2.4.4 Standards based on System Architecture

The gaps between a multimedia mail service and existing standards can be related to either the document structure or the system architecture [7]. The European RACE project carried out a research on the Coordination, Implementation, and Operation of multimedia services (CIO 2060). Four of the partners in this project (PIT Research in the Netherlands, GMD FOKUS and Acotec Gmbh in Germany, and Telephonia in Spain) focused on the specifications and development of a prototype for a Multimedia Mail Service (MMMS).

The approach chosen for the system architecture is to use existing standards like X.400 as much as possible without changing them. The requirements of multimedia mail services can be met by combining and coordinating existing standards. This approach leads to a full backward compatibility with existing e-mail system. The requirements of the MMMS, its general architecture, and its implementation will be discussed in the following subsections.

- MMMS Requirements

Large enterprises generally have a corporate e-mail network that is based on X.400. On the other hand, most universities implement SMTP and MIME mail tools. Therefore, the system architecture should be flexible to support both X.400 and MIME based e-mail systems. MIME is capable to send a reference to an FTP site instead of sending the bulk data. X.400 on the other hand does not support that. Therefore, it is important that the
MMMS system architecture supports the transmission of referenced in lieu of the bulk data.

- **MMMS Architecture**

As depicted in Figure 2.9, the MMMS architecture is based on the X.400 standard, and is decomposed in a number of objects. The multimedia mail environment contains the objects User and Multimedia Mail Service. The User is either a human or an application program using the services of the MMMS. The MMMS provides the following multimedia mail services: Create, Present, Submit, List, Retrieve, Local-store, and Global-Store.

![Figure 2.9: The architecture of the MMMS.](image)

The Create, Present, and Local-Store operations have only local impact, and they are provided to the User by the Service Agent which acts as the UA in the X.400 architecture. Other operations such as, Submit, Retrieve, and Global-Store are distributed
and they are provided to the Service Agent that is part of the Multimedia Mail Transfer and Storage System (MMTSS). The MMTSS provides the following distributed mechanisms (teleservices) to support the above mentioned distributed operations: the messaging; the storage; and the directory teleservices.

The messaging teleservice provides a mechanism for the transfer of small amounts of data such as a simple drawing objects or a text file. This teleservice can be seen as the standard X.400 Message Transfer System (or any similar message transfer service).

The directory teleservice provides for the mapping of user friendly names onto „real” mail addresses and for the retrieval of authentication information of distribution lists.

The storage teleservice provides the storage facility for bulk data. This teleservice was introduced so that in the case of large content parts, bulk data is taken out of a message and replaced by a reference before the transmittal phase. The reference number is generated using the Distinguished Object Reference (DOR) standard. When receiving the message, the reference indicates where the data is stored. (this is equivalent to the FTP referenced site in MIME).

From an interoperability point of view, the architecture of the MMMS (storage teleservice) only helps the different partners in the RACE project exchange bulk data with the benefit of a reference object instead of bulk data. However, when an MMMS message containing referenced objects is sent to MIME user, the referenced object cannot be retrieved.

2.4.5 Mediator

Supporting interworking with the user's local mail tools is the key strategy to improve acceptance of the multimedia mail teleservice. For example, it is not easy to use an X.400
mail tool and use the Internet mail protocol for transmission of messages directly without gateways. If the mail tool and the message transfer protocol are not compatible, the message transport system will differ in the interface to the mail tool, addressing scheme, and other service elements. Therefore, if the mail tool on both the sender's and the receiver's sites is incompatible, the message generated on one site cannot be interpreted on the other site. The Mediator is an extended conversion facility discussed in [8], and it includes three main sub-components: mail adapter; message transport system adapter; and parser and formatter.

- Mail adapter: There are two kinds of mail tool adapters; the send-adapter; and the receive adapter. The send-adapter enables the mail tool to pass formatted messages to the Mediator for further processing, the receiver-adapter feeds incoming messages to the mail tool by either adding messages to the local Message Store or by accessing the system spool area.

- Message transport system adapter: The message transport adapter connects the Mediator to the message transport system, and is done by a protocol process, by using a UNIX/SMTP tools and conventions.

- Parser and formatter: The most interesting components of the Mediator are the parser and the formatter. The Mediator Base Format, a sub-structure of the Mediator, allows representation of message structures that are independent of existing formats. The other sub-structure of the Mediator, the parser, converts incoming messages form the mail tool or the transport system to the internal representation. The mediator can handle different message formats by employing different parsers and formatters. For example, if the user's mail tool provides a message in the format used by the Sun's mail tool, the appropriate parser converts the message to the Mediator Base Format. The user can advise the Mediator to generate, for example, an X.400 or MIME coded
representation of this message by selecting the appropriate formatter, or any other interchange format.

With the approach suggested by the RACE project, there are no restrictions on the interchange format used nor on the network setup; however, it is costly to implement and its usability has been only proven among the partners that undertook this project.

The interchange format standardization problem is solved by employing the Mediator; however, this is a lossy solution, for conversion of interchange format can loose a lot of unrecoverable information that subsequently leads to complete ambiguity of the mail.
Authoring multimedia documents can be easily implemented using general multimedia authoring tools. Documents generated by these tools contain information on the temporal and spatial relationships between the multimedia elements included therein. However, general multimedia authoring tools produce documents with proprietary file formats that make exchanging and interpreting these documents in a multiplatform environment a difficult task. For example, if a mail document is authored using a tool such as, Director™ or Authorware™, its interpretation on the receiver’s site can easily fail if the UA does not recognize the document format, or if the application that supports its playback is simply not available. Therefore, it is important to understand that, only after a standard international interchange format (exchange, document, and media) is reached, can the exchange of formatted multimedia documents via Internet be fruitful.

With a standard interchange format far from realization, one should note two important observations. First, the need for multimedia mail authoring tools that do not generate proprietary document formats, but instead follow a standard such as MIME. Second, is the distinction in the authoring style between general purpose hypermedia/multimedia authoring tools and that of multimedia mail messages. This distinction can be attributed to the less interactive nature of multimedia mail messages.

This chapter reviews the common approaches to the design of general multimedia authoring tools. It concentrates on the design concepts of multimedia mail authoring tools by providing an in-depth discussion on the key factors that answer the following
questions: (a) What can you author? (b) How much work, time and skills are needed? and (c) How flexible is the resulting document?

Finally, we examine three different multimedia mail authoring models, in each model, we provide an insight into both the authoring and presentation style, and examine the advantages and disadvantages.

3.1 Overview

The requirements for authoring systems are rapidly changing as the gap between the possibilities envisioned for multimedia technology and the authoring tools available to realize them grow wider [30]. Multimedia mail authoring tools are communication-oriented applications that allow users to compose and exchange messages containing multiple media elements. In general, there are three main questions in evaluating a multimedia mail authoring system:

- What does the system let you author?
- How much work and skills are needed or involved?
- What can be done with the result?

3.1.1 What Does a Multimedia Mail Authoring System Produce?

Multimedia mail is a communication-oriented application constrained by one major factor: standards, and subsequently the resulting multimedia message which is less interactive in comparison to a multimedia document produced by a full-blown general multimedia authoring tool. Authors want to create mail messages usually containing a variety of media elements – video, text, audio, and so on. In addition, authors want to define some relations between the media elements included in a mail message. The current state of
affairs in the authoring style of most multimedia mail tools, does not easily allow authors to define spatial or temporal structures among multimedia elements included in the message. Therefore, authors have two main concerns: importing media elements into the mail document; and defining the synchronization relationships between them.

- Importing the media elements: This includes the content data for each component and its presentation format. For example, should an audio clip be included in the multimedia mail message, the author's responsibility is to inform the system from which storage location and what desired media format to represent it.

- Defining the relationships among the various multimedia elements: The relationships can be expressed in terms of individual, sequential, or parallel execution. In the individual category, the author's responsibility lies in placing the media elements arbitrarily in the document. On the other hand, in both the sequential and parallel categories, the author is responsible to explicitly specify the relationships. For example, a sequential relationship between elements $x$ and $y$ can be defined so that during the course of the mail presentation, if the user initiates the execution of element $x$ at time $t = 0$ for example, and element $x$'s presentation is terminated at time $t = 5$. Element $y$ then starts its execution at time $t = 5$.

### 3.1.2 Time and Skills

Time, skill, and experience are inevitably prerequisites for every user regardless of the authoring tool at hand. When designing an authoring tool, it is important to take into consideration the level of experience needed and the time required to comfortably accomplish a certain task. Moreover, when designing an authoring tool, certain operations are considered important, and therefore they are optimized by the designer. This design approach is a process of abstraction where it becomes quite efficient to do certain tasks
based on the "target domain". For example, in the area of multimedia mail, the target domain can be the media types (audio, video, etc.) that can be inserted in the message, and the procedure to define the structure among the multimedia elements. In our approach to multimedia mail authoring, the MIME pre-defined types (audio, video, multipart, application, etc.) are given higher preferences during the interface design by making each MIME pre-defined type a visual icon or agent in a floating palette as depicted in Figure 3.1.

![Figure 3.1: The Mobile MIME agents and their iconic representations.](image)

Another important concept in the design of multimedia mail systems is to examine the simplicity/boredom tradeoff. Simplicity is an important feature of any authoring tool because it allows the system to be used by a wider range of users. However, the best authoring system is not necessarily the simplest, but rather the one that helps its users grow in experience most comfortably and increase their productivity [30]. On the other hand, interactivity is also very important for any authoring tool; however, with more interaction comes more complexity. In our approach, we provide a good level of interactivity while minimizing the level of complexity.
3.1.3 Resulting Document Flexibility

In an authoring system, the user is called the author. On the other hand, in a presentation or "playback" system, the user is called the reader. The success of a multimedia document depends on the success of two processes: authoring and presentation. The author is usually provided with a richer tool set than that of the reader. This discrepancy raises the question of the level of both the reader's document manipulation and the distinction between authoring and presentation systems.

A very useful and important feature of a multimedia document is inclusion; it provides a mechanism to allow insertion of one document or an interactive controller into another document. An example of an interactive controller is the Play, Pause, and Stop buttons that allow the user to control the play of an audio or video clip. Searching the content of a multimedia document is currently restricted to textual contents only; images, audio, and video remain a great challenge with respect to this important feature. Finally security issues in authoring systems are twofold. First, on an individual level, security is maintained by locking readers out of some or all information. Second, on a distributed level, mailing interactive multimedia documents introduces numerous security problems, among them the access to destructive system files during the execution of the document presentation.

In general, authoring tools provide an alternative to custom programming. Custom programming is a process that is usually complicated, time consuming, and requires expert skills and knowledge. There are software programs that contain pre-programmed objects that are used in the development of interactive multimedia applications. Authoring tools differ among each other, each of which presents its own orientation, capabilities, and learning curve. Both authoring and presentation systems are needed to create and present
information on the screen. There is a difference between authoring and presentation tools because of the difference in their objectives and intended users [17].

Authoring tools are usually designed to provide users with a mechanism to create and organize different media types according to certain spatial, temporal, and procedural relationships. The resulting document is a structure of multimedia elements linked in various paths. Most authoring systems require well trained authors in order to accomplish moderate to complex interactive tasks.

On the other hand, presentation systems are designed to allow the user to interactively view and navigate multimedia documents. However, during the presentation process, the end user may not add or modify the content of these documents, and in fact, will interact with them on the terms set by the author. Some new presentation packages do however allow users to put together an interactive presentation thus providing the user a limited authoring experience [30].

3.2 Multimedia Mail Authoring Models

One of the challenges of multimedia mail is to create an authoring tool that supports the easy, interactive creation of messages for non-expert users. There are several authoring models in existence: the Linear; Hypermedia; and Active models. The authoring model is tightly coupled with the document or message form metaphor. Among the most popular message forms is the two-dimensional paper metaphor. The form metaphor has significant importance; first, it is the “look and feel” of the mail message during both the authoring and the presentation processes. Second, it simplifies the mapping process of the form onto a different platform or application such as the World Wide Web (WWW). Figure 3.2 gives an overview of the different authoring models discussed in this thesis.
Figure 3.2: An overview of the authoring/composing models.

3.2.1 The Linear Model

Authoring or composing multimedia mail using the Linear approach has been introduced through applications such as Netscape™ and Pegasus™. In the Linear model, the document format is usually of a textual nature therefore allowing only textual data to be inserted and edited inside the document. Figure 3.3 depicts the Linear authoring style in the Netscape™ browser environment. Note that the document format is paper-like; however, only textual data are allowed to be directly inserted in the document.
Multimedia elements (audio, graphics, video, applications, etc.) are not directly inserted into the document; rather, they are linearly attached to the document via a dialog box that helps locate them on either local directories where files already exist or through the Uniform Resource Locator (URL) service as depicted in Figure 3.4.

Using the Linear model has many disadvantages: poor interactive authoring style; lack of structure among the multimedia elements; and poor presentation and navigation of the mail.

- Poor interactive authoring style: During the composing or authoring process, the user is presented with a single window through which attachments that contain multimedia elements are sequentially inserted into the mail message. Authoring with this model is simply a non-interactive repetitive task. Moreover, since attachments are not directly inserted in the document, the author has no control over the position of attachments.
in the resulting document, and subsequently this leads to a poor message navigation at the recipient’s site.

![Attachments window](image)

**Figure 3.4:** Attachments window.

- Lack of structure among the media elements of the message: Since attachments are added sequentially, the author cannot specify any spatial relationships. Moreover, temporal relationships which ensure sequential or parallel execution among the attachments are also not supported by this model.

- Poor presentation of the media mail: Upon receiving the mail, the reader can only manually retrieve the attachments, and there is no equivalent of a presentation or a playback. Moreover, pursuing the mail is achieved sequentially by individually reading the attachments, this style results in a poor navigation capability.
3.2.2 The Hypermedia Model

The main idea behind the concept of the Hypermedia model is to take a multimedia document whose format is a standard such as, MHEG, and map it onto the WWW. Using the WWW paradigm, hypermedia messages are composed using Hyper Text Markup Language (HTML), and the documents that form the message are retrieved using Hyper Text Transfer Language (HTTP). As its name implies, “text markup” is closely related to the notation format of a text. It is used as a technique for describing information structure, especially when a paper-like format is in use.

The approach to using the WWW paradigm can be attributed to the large number of sites where WWW browsers are installed, and the interoperability services they provide on different platforms. An example of such an authoring model can be found in [6] where a Simple Multimedia Hypermedia Mail System (SMHMS) is extended to provide mapping of its document onto the WWW browser. SMHMS uses the Nested Context Model (NCM) as a document format.

NCM is a document model in accordance with the MHEG standard. In NCM, a document is composed of nodes (fragments of information) and links. There are two classes of nodes: terminal nodes and composite nodes, the latter being the central concept of the model.

A composite node is used to group together nodes and links between these nodes. This concept therefore permits organizing, hierarchically or not, sets of nodes, and defining different logical views over the same contents.

A link connecting two nodes is defined in a composite node containing these two nodes. This means that the same terminal node may be a source of a link in one composition and
not in another; that is, the contents of a document are independent of its links. The NCM approach in defining composite nodes and links contradicts the WWW model where document contents are not independent of their links.

From an implementation point of view, provision of information by servers is exactly like in WWW, which allows the system to use existing HTTP servers. Browsing, however, is slightly different. In the WWW environment, documents are retrieved when the user selects a link, this implies real-time transmission of data, which in the case of sound and video may take a long time. Moreover, in the WWW environment, links are embedded in documents. So, it is impossible to reuse the document without inheriting all the links. This means that if the same multimedia element is used in two messages with different links defined in each of them, the node that represents that element will have to be replicated. This overhead requires extra memory space.

Another limitation stems from the fact that links are embedded in nodes and therefore, it is impossible to define spatial and temporal synchronization for presentation of nodes present in a composition (as in MHEG or NCM), because nodes in different compositions have different synchronization requirements.

The Hypermedia model offers support for creating, transferring, and accessing hypermedia documents in a heterogeneous environment. Navigation of the message components is done through links, and in case of large messages, users are forced to navigate the document by going through many links which in turn might cause some ambiguity in the message’s meaning. This problem is often referred to as being “lost in hyper space”. Moreover, using HTML requires a high level of skills that inhibits average e-mail users from making changes to the mapped document.
3.2.3 The Active Model

The concept of active multimedia mail can be divided into two issues: active mail and multimedia mail. Active mail simply means to send executable programs within a mail message. On a more abstract level, active mail messages can be regarded as active objects traveling through the network: searching for, distributing, and collecting information; interacting with receivers; making decisions based on context knowledge; and replicating itself. Active mail has been used in areas such as remote program installation.

In addition to its support for structured multimedia contents, data and user interface, active mail could, for instance, build a form on the receiver's site, consisting of text fields (for editing purposes) and anchor buttons for embedding multimedia elements in the document. The conjunction of active mail and multimedia mail leads to the concept of active multimedia mail.

The general structure of active multimedia mail is divided into two main components: the "data" part that contains the included multimedia elements; and the "behavior" part that contains the program which is executed at the receiver's site. From an implementation point of view, an active multimedia mail message is basically a program written in a suitable interpreter language. At the receiver's site, the interpreter is started automatically upon message access, executing the program contained in the message. This terminology implies that creating an active multimedia message is equivalent to writing a program. However, the average user is not expected to do a programming job when the task is simply to compose a message to a friend or colleague.

Therefore, one of the challenges in the Active model approach is to provide an intuitive, user-friendly interface that allows, in simple fashion similar to its textual counterpart, the
creation of active multimedia mail messages. The following sections describe the user interface and structure of ActiveM³ [38] which is based on the Active model.

- **The ActiveM³ Model**

ActiveM³ is a multimedia mail tool based on the MIME standard. It combines the concepts of both active mail and multimedia mail. The mail document in ActiveM³ follows the two-dimensional paper-like format. The document allows multiple choice fields, static and editable text field, and labels to be arbitrarily placed on the document form. In addition, the integration of multimedia elements is also possible. As depicted in Figure 3.5, media elements integrated in the mail document are stored in the “backpack”, on the other hand, the behavior or the active part is stored in the “hiker”. The hiker is a Hyperpicture Command Language HCL-script (a lisp dialect) designed to organize, orchestrate, and present the display of the backpack at the destination.

![Figure 3.5: ActiveM³ mail structure.](image)

- **ActiveM³ Authoring Model**

ActiveM³ is an authoring system for active multimedia mail that is based on the metaphor of direct manipulation. The user creates an active message by interactively assembling
multimedia elements onto the mail form (mail document). The mail form may contain presentation/interaction elements such as buttons, text areas, or in-line images that are used to present multimedia elements in external presentation tools. ActiveM³ provides a number of different views during the authoring phase to simplify both the creation and visualization of the mail. The following views are available:

- The page view: It includes the initial empty form or document in which the user can assemble multimedia elements using the available presentation/interaction buttons. These elements are picked from an associated panel and are positioned on the form. In addition, the page view may contain “anchor” buttons, leading to multimedia content objects that are presented in external presentation tools, such as image, audio, or video.

- The backpack view: As depicted in Figure 3.7, the backpack view shows the multimedia elements currently contained in the message. The backpack control panel, as depicted in Figure 3.6, contains a palette of available multimedia elements that can be inserted into the backpack. Moreover, the backpack conceptually supports the editing and visualization of the interrelation between the multimedia elements.

- The script view: displays the HCL script created by ActiveM³ for the current form. Experienced users may modify the script to give more functionalities to the presentation.
Temporal and Spatial Authoring

As depicted in figures 3.6 and 3.7, authoring basic synchronization is provided in a graphical interactive manner. The backpack view and its associated control panel are separated into two fields:

- Multimedia elements: the panel buttons enable linking of multimedia elements into the mail.

- Temporal relations: the temporal buttons allow the user to choose between several synchronization operators, such as parallel or sequential. The synchronization among objects can be seen as a directed graph. A timer mechanism is also available, and permits the user to explicitly define the duration of a compound action.
The synchronization is embedded using a language based on path-expressions that in turn is based on the logic of temporal intervals. A simple example is the description of a slide show consisting of two pictures (P1 and P2) and two audio clips (A1 and A2). Using the following path expression: path (P1 ∧ A1) ; (P2 ∧ A2) end, the slide show starts executing by displaying P1 and A1 simultaneously. The parallel execution is denoted by the "∧" operator. The ";" character is a chop operator and its job is to terminate the first part of the slide show. The second part of the slide show, (P2 ∧ A2), is started after the execution of the chop operator and ends with the end of the path.

HCL is based on regular path-expression. It should be mentioned that path-expressions are regular expressions which are transformable into a "deterministic finite state-machine". The implementation of the synchronization mechanism is achieved by taking these regular path-expressions and transforming them into deterministic finite state machine. The finite state machine is described in ActiveM³ as follows: \( M = (Q, \Sigma, A, \partial, q_0, F) \) where \( Q \) is the set of states, \( \Sigma \) is the list of events, \( A \) is the set of actions, \( \partial \) is the transition-function defined as: \( Q \times \Sigma \rightarrow Q \times A \), \( q_0 \) is the initial state, and \( F \) is the set of final states. The HCL abstract presentation functions used to achieve synchronization are:

\[
\begin{align*}
\text{CreateMMobject("Picture1" "obja" "image" refBackpack time X Y)} \\
\text{CreateTransitionTable( state listOfActions)}
\end{align*}
\]

where CreateMMobject is needed to create different instances of the same backpack components. The second function is automatically invoked to build a transition table where path expressions are translated into finite state machine expressions.

Temporal and spatial synchronization are embedded in the document via buttons that enable these relationships on the screen. These buttons can be picked up from the media editing panel. For parallel execution, ActiveM³ does not abide by the multipart/parallel
subtype provided by MIME; rather it uses a finite state-machine. This approach stems from the fact that HCL does not support multi-threading. Although important features such as hierarchical composition and parallel media presentation are available in this model, the authoring style can be very difficult especially for novice users that are migrating from textual mail based tools to multimedia ones. Therefore, it is of a paramount importance to provide a mail tool that support easy and interactive composition style in addition to the important features available to the Active model.

Moreover, the execution of the script is greatly dependent on the presence of an interpreter at the receiver’s site. Since the availability of the HCL interpreter is scarce, the mail presentation on most platforms is not supported. This delinquency makes this approach non-supportive of one of the main goals, that is interoperability.
4. MOBILE MIME AGENTS: IMPLEMENTATION AND DESIGN SPECIFICATIONS

This chapter presents our new visual authoring model that is based on the visual agency concept. It also describes the interface design and functionalities that support the introduction of this new multimedia mail authoring tool.

In this chapter, we discuss the goals that influenced the design specifications of our visual Mobile MIME Agents multimedia mail authoring tool. The following issues will be dealt with: the visual authoring approach; a detailed description of the interface elements; the document format; the mail message; and a description of the implementation. The latter will include the MIME-Agent-Parser (MAP), the Mobile MIME Agent Internal Database (MMAIDB), and the “Orchestrator”. The MAP is responsible for scanning the mail document and collecting MIME agents information. The MMAIDB is where the information collected by the MAP is stored. The MMAIDB acts as a database that is used by the “Orchestrator” which is a general algorithm written in the Java language. The “Orchestrator” orchestrates the MIME agents on the receiver’s site and supports the user interaction with the agents.

4.1 Mobile MIME Agent: Our Visual Authoring Approach

User interfaces for authoring tools have evolved from string driven, to menu driven windows environment, and finally to an icon-based interface. In the latter, multimedia documents can be authored by directly inserting, editing, and visually manipulating the multimedia components on the computer screen. The most powerful contemporary
authoring tools use programming paradigms to accomplish difficult tasks by easily allowing complex interaction and synchronization relationships to be defined.

There are three main goals that have influenced the design of our Mobile MIME Agent multimedia mail authoring tool. The first goal is to ensure the system’s compatibility with the traditional Internet e-mail environment. In so doing, we take full advantage of the MIME capabilities, including parallel execution of the multipart/parallel multimedia components. The second goal is to provide the user with an intuitive interface that allows interactive authoring in accordance with the drag and drop Macintosh™ guidelines [4]. Finally, the third goal is to make our multimedia mail tool interoperable by taking advantage of the Java platform independent byte code and its capability of carrying and executing programs on heterogeneous platforms.

The authoring approach we propose in this thesis is based on the direct manipulation metaphor and the visual agent approach. The intersection of these ideas simplifies the complexity of the authoring task and takes multimedia mail applications to a level of sophistication not formerly achieved. For example, in Eudora™, it is impossible to compose a multipart/parallel message body; whereas in our visual agent floating palette, as depicted in Figure 4.1, and especially the visual concept of agents, (especially containers) the mode of user interaction becomes one where the user visually manipulates and assigns temporal and spatial relationships to the agents. Thus, using the direct manipulation metaphor and the visual agent approach not only increases interactivity during the authoring phase, it also improves the user’s productivity.

Our main objective in using the visual MIME agent approach is to develop a direct manipulation metaphor for a multimedia mail authoring tool. To accomplish that, we introduced the pre-defined MIME types as visual agents in a floating palette as depicted in Figure 4.1. Therefore, the authoring task becomes one where the user composes a
multimedia mail message by interactively assembling multimedia elements onto the mail document. The mail document form may contain any MIME agent type.

![Multimedia mail message composed form](image)

Figure 4.1: The Mobile MIME agents and their iconic representations.

### 4.1.1 The Interface: A Closer Look

As we have mentioned in the beginning of this chapter, a major challenge for any multimedia mail application is the introduction of an interactive tool that allows inexperienced users to easily author or compose multimedia messages in a direct interactive fashion. It is our belief that the authoring environment that the interface provides concentrates on the specific presentation functionality required for visual iconic representation and the agency concept.

To give an impression of how authoring is accomplished using the visual MIME agent approach, consider the composed form in Figure 4.2. The authoring tool is divided into four main parts: the Address-Subject Area; the Document-Area; the Control-Area; and the MIME-Agent-Area depicted in figure 4.1.
- The Address-Subject-Area: This area contains two main components: the address(s) of the recipient and the subject of the message. They are both editable text boxes where the user can edit, cut, paste, etc.

- The Document-Area: This is a paper-like format and it is the initial empty form in which the user can assemble a message using the MIME agent elements. Agents are selected from the MIME-Agent-Area and interactively positioned on the form. In the case of audio, video, graphics, or application agents, the system does not launch any editor to author the media; rather the media has to be already prepared. Assigning multimedia elements to their prospective MIME agents can be achieved by either double-clicking on the agent and using a dialog box mechanism, or by selecting the agent and then pressing the Import button found in the Control-Area. In both cases, a dialog box mechanism is used to provide information on the type, location, and name
of the media component. On the other hand, in the case of a text agent, as soon as the author drops the agent on the form, the system launches a simple text editor. In the case of a container agent, the user still drags and drops the container on the form; however, a double click will open up another composition window with an empty “form or sub-form” where the user can drop any agent type, including containers. Assigning relationships to agents inside the container agent is achieved by selecting the appropriate relationship from the Container menu. In both the form and sub-form, the user can arbitrarily manipulate elements by changing their position and other attributes.

- The Control-Area: This is an area composed of buttons; it provides a way for the user to manipulate the contents of a mail message. For example, the user can save the mail file to a particular folder, send the mail, and edit a particular multimedia element in the mail by selecting the agent and pressing the Edit button. To give a visual control to the user, standard buttons are offered. Our prototype includes the following buttons; however, more could be added to increase the functionalities of the Control-Area.

  - **Cancel:** During the process of media authoring, users often like to terminate the composition, done by clicking on the Cancel button.

  - **Save:** This button offers the possibilities of saving the mail file to either an In-mailbox, Out-mailbox, or some other folder.

  - **Send:** Allows the composed mail to be sent. If the mail is not already saved, a mandatory dialog will force the user to make the save choice before sending the mail.
- *Edit:* This button allows the contents of the visual MIME agents to be edited by launching their corresponding applications. For example if an audio agent has been assigned an audio component but later decided to alter the sound file, the user chooses that particular agent and clicks on the *Edit* button and the appropriate application is launched.

- *Import:* This button allows the user to import a particular media element into the document.

- The MIME-Agent-Area: This area contains the MIME agents in a floating palette. Agents are divided into two categories: single media agents; and the generic container agent. The single media agents are: Audio; Video; Text; Application; and Graphics. The generic container agent can be configured to become a multipart or a message container type.

For a complete example on both the authoring and presentation tasks, refer to Chapter 5.

### 4.1.2 Working with Containers

MIME messages consist of one or more body parts. In the simplest case, these body parts are in sequential order. The user can create body parts that, in turn, contain other body parts. These are referred to as Container body parts. Containers provide structure to the MIME messages. For example, to group a number of mail messages in a message/mixed container, the user by drags the generic container and drops it on the document. The next step is to define the type and subtype of the container. This is done by using the *Container* menu and its sub-menu selections. Once the type and subtype are
established (multipart/mixed), a label indicating the type of the container appears under the depiction of the container agent. After the type is set, the user is presented with a dialog to specify the name of the container. The name is separated from the type of the container by a "-". Both the type/subtype, and name of the container provide valuable information during the presentation phase. To include agents in the container, the user double-clicks on the container agent, and is presented with another composition window or a sub-document to add media into the container; simply, the user can drag and drop any agent type. This approach to authoring with containers allows hierarchical structure of the mail.

4.2 Document Format

Our suggestion is simple, we use a two-dimensional paper-like form. The document follows the MIME standards (no overheads like MHEG). Our prototype allows MIME agents of any type to be arbitrarily placed in the document. Furthermore, it allows the author to create a path by following a top-down or left-to-right path. With this flexibility, the reader can peruse the document with more understanding and flexibility. For example, if a mail consists of an audio element followed by some text and then a synchronized audio/graphics presentation, the document would resemble the layout in Figure 4.3.

Figure 4.3: The document format in (a) and its sub-form in (b).
Our main goal is to allow the user to write letters in a free style and without any restriction or limitation. Users are free to acquire, edit, and manipulate multimedia elements in the editor of their choice. Once a multimedia element is prepared it can be imported to the document via a dialog box that allows the user to provide information such as, its location (local storage or external such as FTP). For example, if a sound file is stored on local storage and is needed for the multimedia mail, the user drags the sound agent from the MIME agent palette and drops it onto the form; then by double-clicking on the agent, a dialog box is displayed which allows the user to provide the necessary information.

4.3 Mail Message Format

Our model follows the MIME standard as its mailing message format, it supports all the seven MIME pre-defined parts and some of their important subtypes. It is extensible in the sense that any new subtype can be defined and added to the system; however, the seven pre-defined MIME agent types are not extensible. The restriction on the extensibility of the seven pre-defined MIME types is primarily due to our strong belief that standards should be followed. As depicted in Figure 4.4, the mail message consists of three main parts. The first part is a standard ASCII text message parsed as a text/X-Message subtype. The text is designed to work as a message to inform a recipient whose mail tool is different from that of our Mobile MIME Agent, on how to view the mail. The second part is a multipart/mixed that comprises of two sub-parts that are the "Orchestrator" and the Mobile MIME Agent Internal Database (MMAIDB). The third main part of the mail is called the "MediaPack", and is the linear list of the multimedia elements present in the mail document.
The “Orchestrator” is a Java program based on a general algorithm to construct the MIME agents at the receiver’s site. The MMAIDB is an internal database collected by the MAP, it contains information on the MIME agents in the mail document. The information collected by the MAP is stored in the MMAIDB, and contains data such as the instance name of the agents and their coordinates on the composer’s site. The data is used by the “Orchestrator” to organize and orchestrate the display of the “MediaPack” on the receiver’s site. The complete message is sent as a MIME application/X-MMagent. The introduced (application/X-MMagent) type indicates to the UA at the recipient’s site that the mail contains an executable application part that needs to be executed first [38]. The application/X-MMagent consists of three parts: the text message as an X/Message, the “MediaPack”, and a multipart/mixed containing the “Orchestrator” as an application/X-MMagent plus the MMAIDB as a text/X-MMAIDB. Upon the execution of the “Orchestrator”, the UA makes the contents of the “MediaPack” available to the “Orchestrator” for retrieval.

An example of a mail format is depicted in Figure 4.5. The mail message contains, as a standard, the MIME version header field to indicate which MIME version is being used. Next, the system uses the multipart/mixed content-type to group all the multimedia
MIME-Version: 1.0
From: Nathaniel Borenstein<nnb@bellcore.com>
To: support@bellcore.com
Subject: A Mobile MIME agent Example
Content-Type: application/X-MMagent;
boundary =x-mmagent
--x-mmagent
--unique-boundary-1
Content-Type: text/X-Message; charset = US-ASCII
Message
This multimedia mail message has been composed....
--unique-boundary-1--
--x-mmagent
Content-Type: multipart/mixed;
boundary=unique-boundary-2
"Orchestration"
Content-Type: application/X-MMagent
Content-Transfer-Encoding: Base64
+ .... While not the end of file in the MMIADB
scan the objects;
call the graphics constructor;
......
MMIADB
--unique-boundary-2
Content-Type: text/X-MMaidb
unique-boundary-1. Text Object. "Proposal". (50, 45)
--unique-boundary-2--
--x-mmagent
Content-Type:multipart/mixed
boundary=unique-boundary-3
--unique-boundary-3
Content-Type: text/plain
...... the fox that jumped
"MediaPack"
--unique-boundary-3
Content-Type: multipart/parallel
boundary=unique-boundary-4
--unique-boundary-4
Content-Type: audio/basic
Content-Transfer-encoding: base64
...... mu-Law format audio data goes here
--unique-boundary-4
Content-Type: image/gif
Content-Transfer-encoding: base64
...... Base64 image data goes here
--unique-boundary-4--
--unique-boundary-3--
--x-mmagent--

Figure 4.5: A typical Mobile MIME Agent message format.
elements. At the receiver’s site, if the recipient’s system differs from that of the sender’s, the UA retrieves the media linearly and places them in the form of icons on the recipient’s screen. The UA retrieves and distinguishes the media objects via a pre-defined boundary in the MIME version header (Boundary = xxxxx) where xxxxx can be any ASCII representation.

4.4 Authoring the Temporal and Spatial Synchronization

Our approach is to use the synchronization technique promised by the MIME technology. This type of synchronization among media objects during presentations is achieved by placing the media objects to be executed in parallel in the multipart/parallel container. However, Borenstien in RFC 1521 [12] reminds us that parallel execution is not always performed correctly by the UA. Spatial relations can be achieved by placing the objects in a document or sub-document following either a top-down or a left-right path. The reason we chose these paths is twofold; First in a hand written mail, people usually read the letter by scanning it starting left-to-right and then jumping to a top-to-bottom, then left-to-right again. Users are not obligated to follow either path; however, the path will help organize the mail and make it easy to navigate. For example, if the author wants the reader to sequentially peruse a set of graphic files, the user can place the image agents in a multipart/mixed container one after the other in either a horizontal or vertical fashion.

4.5 Implementation Details

Two important issues of implementation are: how to build an internal database that could adequately represent information on the MIME agents present in the mail document; and the development and implementation of a general algorithm to support the
“Orchestrator”. The Java language is chosen for its platform independent capability. To accomplish this task we use the MIME-Agent-Parser that we will discuss in the next section.

4.5.1 MIME-Agent-Parser (MAP)

Although available to most multimedia mail tools, parsers are mainly used to scan the document and assemble the different body parts of a media mail. The MIME-Agent-Parser or MAP on the other hand, is designed to perform two functions that include converting MIME agents to body parts and building an internal database (MMAIDB) of the different MIME agents in the mail document.

The MAP is activated when the mail is ready to be delivered and the user clicks on the Send button. The basic work is to scan the authoring document and convert each part (Video, Audio, etc.) into a corresponding body part, and finally assemble the body parts in a whole message. To simplify the job of the receiver’s UA in dividing the message into its appropriate components, the MAP places the first main part of the message (“Orchestrator” and MMAIDB) at the top of the message and the linear list of the multimedia elements or the “MediaPack” at the bottom. This is done intentionally to improve time efficiency during both data retrieval and agent orchestration on the user’s screen.

The second function that the MAP is responsible for is to collect the necessary information on the MIME agents present in the mail document. This information will include agent type or class, agent id, and finally the coordinates (x, y) of each agent in the document. The MAP is able to collect nested agent information such as the different agents that reside in a container. In the next section, we will discuss in detail the mechanism behind both the MMAIDB and the “Orchestrator”.
4.5.2 MMAIDB

MMAIDB is an internal database collected by the MAP and contains the complete layout of the message, such as the instances of each object, the name given to a particular media, its location, its parent composition window, the Boundary, and a unique id. The unique id is used to avoid duplicate copies of the object instance during both the authoring and presentation phases. A sample MMAIDB database file is depicted in Figure 4.6. The database file defines the multimedia mail agents and their hierarchical structure.

```
TOOL "&MMATool"
MIME_AGENT WINDOW
{
  AGENT "&Audio"
  {
    Name "Sound1"
    Coordinates "(35 45)"
    Unique_Id "345F4E"
    Unique_Boundary "xxxxx"
  }
  CONTAINER_AGENT "&Mixed"
  {
    Name "Proposals"
    Coordinates "(45 55)"
    Unique_Id "395G4E"
    Unique_Boundary "yyyyy"
    CHILD_WINDOW "Proposals"
    {
      AGENT "&Graphics"
      {
        Name "Graph1"
        Coordinates "(30 40)"
        Unique_Id "34014K"
        Unique_Boundary "xxxxy"
      }
    }
    AGENT "&Audio"
    {
      Name "Audio2"
      Coordinates "(20 40)"
      Unique_Id "398O4D"
      Unique_Boundary "xzzyy"
    }
  }
}
```

Figure 4.6: A sample MMAIDB database file.
The MMAIDB is constructed by the MAP that parses agents information from the authoring or composing view. Information inside the MMAIDB are divided into two main categories: AGENT and CONTAINER_AGENT. The token “AGENT” indicates to the “Orchestrator” that this is a single media agent with Name, Coordinates, Unique_Id, and Unique_Boundary, and should be constructed in the main reading window or view. The token “CONTAINER_AGENT” on the other hand, indicates that there is a hierarchical structure. Container agents have the following attributes: Name; Coordinates; Unique_Id; Unique_Boundary; and CHILD_WINDOW. The latter is the child window where the agents inside the hierarchy are constructed. Agents inside the hierarchy can be either single agents or container agents.

4.5.3 The “Orchestrator”

The binary code of the “Orchestrator” which is depicted in Figure 4.7, is encoded and placed inside at the top of the MIME file. The “Orchestrator” is a general algorithm implemented in the Java language. The “Orchestrator” introduced subtype (application/X-MMagent) indicates an active part consisting of the Java program, that will be executed using the Java virtual machine. Upon receipt of the mail, if the recipient’s mail tool is a Mobile MIME Agent, then both systems are compatible, and therefore the “Orchestrator” is activated. The code in the “Orchestrator” starts reading the MMAIDB database file to retrieve the necessary information of the MIME agents included therein. The code then asks the local system’s graphical constructor to construct instances of agents in their appropriate windows and folders, and associates each agent instance with its Unique_Boundary value in the MIME file (“MediaPack”) that contains the actual data. The interaction methods of the agents in the reading window are already set so that no extra work is needed to define methods of interaction.

It is important to note here that when the systems of both the sender’s and the receiver’s
if (Check_System ()) // Checks for compatibility in the recipient's system
{
    Open_Mmaindb (); // Opens the MMAIDB to read the MIME Agents information
    while (!EOF)
    {
        Token = Get_Token ("AGENT", "CONTAINER_AGENT ");
        if (Token == AGENT) // Single media agent
        {
            Agent_Info = Retrieve_Agent_Info (); // Get Agent info. (name, coordinates..)
            Construct_Agent (Agent_Info); // Ask the Graphical constructor to construct a single media agent
        }
        else
        {
            Container_Info = Retrieve_Container_Info (); (name, Coordinates, ChildWindow...) (name, Coordinates, ChildWindow...)
            Construct_Container (Container_Info);
            for (each AGENT inside the CONTAINER_AGENT)
            {
                Agent_Info = Retrieve_Agent_Info ();
                Construct_Agent (Agent_Info);
            }
        }
    }
}
else
{
    Generate_Message (); // Generates a message to inform the user on how to view the mail
    Open_Mmaindb ();
    while (!EOF)
    {
        Build_Window ("name", Coordinates);
        Token = Get_Token ("AGENT", "CONTAINER_AGENT ");
        if (Token == AGENT)
        {
            Agent_Info = Retrieve_Agent_Info ();
            Build_Icon (Agent_Info); // Builds an icon inside the main window
            Set_Double_Click_Agent (Boundary); // Sets the method of interaction inside the main window so,
            // that when the user double clicks on that icon the media associated with that Boundary inside
            // the MIME file gets displayed according to its type.
        }
        else
        {
            Container_Info = Retrieve_Container_Info ();
            Build_Folder (Container_Info);
            Set_Double_Click_Container (); // This operation gives the user the option to either open the
            // folder and retrieve the media separately or display the media presentation.
            for (each AGENT inside the CONTAINER_AGENT)
            {
                Agent_Info = Retrieve_Agent_Info ();
                Build_Icon (Agent_Info);
            }
        }
    }
}

Figure 4.7: The general algorithm of the "Orchestrator".
are Mobile MIME Agent, the text/X-Message subtype does not get constructed and displayed; rather, it is ignored by the recipient’s system.

On the other hand, if the recipient’s system is not compatible (Eudora™ for example), the text/X-Message, “Orchestrator”, MMAIDB, and the media elements inside the “MediaPack” are retrieved and, with the exception of the text/X-Message, are constructed in the form of icons. The text/X-Message on the other hand is displayed on the receiver’s screen. The message contained in the text/X-Message provides useful information on how to activate the presentation of the mail. The mail presentation in this case is greatly dependent on the presence of a Java virtual machine interpreter. Therefore, if a Java interpreter is available, the user, by double-clicking on the “Orchestrator” icon, can activate its code which is designed to build its own windows, icons, folders, and methods of interactions with the agents. This approach in building agents icons and containers with their pre-designed methods of interactions on the recipient’s screen, is designed to give the recipient the flexibility to peruse and navigate the multimedia mail with the same flexibility expected when the recipient’s mail tool is compatible with that of the sender’s.

On the other hand, if the Java virtual machine is not available at the recipient’s site, the user can still individually view the multimedia elements – by either double-clicking on their icons or by calling external viewers – depending on the mail tool at hand.

It is important to note that the “Orchestrator” only allows the user to view the message. In particular, it is not possible to compose a reply that includes all or part of the received multimedia message (except, of course, as attachments). For full flexibility in composing replies, the user must use the native mail tool.
4.6 Mail Presentation

Our approach to mail presentation takes advantage of the Java independent platform capability and makes the presentation available in practice in a heterogeneous system. Mail presentation in our approach can be divided into two main categories: compatible mail tool (Mobile MIME Agent) and non-compatible tool such as Eudora™. In the first category, the Java code in the "Orchestrator" is automatically executed, and as depicted in Figure 4.8, the user is presented with the same layout authored at the originator site. The user can then proceed by viewing the mail contents by double clicking on the MIME agents and calling the corresponding application to be launched.

Figure 4.8: The presentation in a MIME Agent mail tool.
In the second category, in a non-compatible tool such as Eudora™, the mail presentation starts by a generated message that informs the user that the mail was composed in a non-compatible tool, and asks whether or not a Java interpreter is available at the site. At this point, if a Java interpreter is available, the user is instructed to double-click on the icon titled “Orchestrator”. Upon its activation, the “Orchestrator” starts by retrieving the appropriate information from the MMAIDB, and start to build the presentation by constructing windows, agents and containers (if any) on the receiver’s screen. On the other hand, if a Java interpreter is not available at the recipient’s site, the user cannot activate the “Orchestrator”, and therefore the user can double-click on the individual media icons and view them individually.

For a complete example and illustration on both mail authoring and presentation, refer to Chapter five.

4.7 Document Flexibility

The resulting document of our Mobile MIME Agents multimedia mail authoring tool can be seen as “WYSIWyG” from both the originator and the reader point of views. The flexibility at the presentation stage is rigid, and does not allow users to change the contents of a MIME agent body part nor its position. This approach is intended to preserve the properties of each MIME agent in the mailing document. The only useful function that the reader is able to perform during the presentation stage is the rendering and navigating of the media parts in the message. On the other hand, with a choice to reply to the message, the presentation stage changes into an authoring task. This implies that all media parts, with their characteristics, are moved into the authoring window and placed in the new document where they can be manipulated (Cut, Paste, Edit). This makes the mail document extremely flexible.
From an authoring point of view, as we have mentioned in section 3.1.2, skill and experience are inevitably necessary for comfortably using an authoring tool. In the case of our prototype, the visualization of the MIME agents, the drag and drop facilities, and the container approach definitely minimize the level of skill required for using a sophisticated multimedia authoring tool. Moreover, we believe that our approach makes authoring simple it avoids repetitive tasks and provides interactivity while minimizing the level of complexity of the authoring process.
In this chapter, we will look at an authoring example of a multimedia mail using our Mobile MIME Agents prototype.

An example in the domain of collaborative work is the following: a user “Bill” working in California is modifying his first draft of a graphical piece that his boss did not like; Bill however wants to approach the modification process from two different angles therefore making two different versions of the final draft. As a final package, Bill wants to mail his boss a text document containing a project proposal followed by the two graphic drafts and a audio file explaining the difference between the two modification processes.

After preparing the contents of the multimedia mail, Bill activates the Mobile MIME-Agent multimedia mail authoring tool. He opens the composition window where he will begin authoring the mail document. Bill drags the MIME text agent from the palette and drops it into the upper left corner of the document. At this point, the system invokes the system’s local text editor where bill can write his proposal; however, his proposal is already stored on a local file. Therefore, Bill cancels the textual editor. To import the text into the document, Bill can either double-click on the text agent, or select it and click on the Import button. In either case, a dialog box pops-up, as depicted in Figure 5.1.
Bill then fills out the necessary information on the text document such as its subtype, name, and storage. At this point the MIME text agent displays its subtype and name in the document.

The next step is to author a synchronized structure of two graphic items and a sound file. To accomplish that, Bill grabs the generic MIME container and places it either to the right of the first agent following a left-to-right path, or under the first agent following top-down path. After the positioning of the generic container, Bill then proceeds to specify the type and subtype of the container. Bill selects the container if not already selected, and chooses the multipart/parallel from the Container menu as depicted in Figure 5.2.
After the type and subtype are defined, the user is presented with a dialog box to add a name to the container. In this case, Bill calls it “Graphdiff”. Bill double-clicks on the container and he is presented with a new composition window depicted in Figure 5.3, where he can drop two graphic agents and a sound agent and follow the same steps in importing the media into the document. This is a simple example. A more complicated and nested structure can be applied to a mail document. For example, to obtain a sequential slide show of two parallel entities, the user places two containers, along with their media contents, of type multipart/parallel into a container of type multipart/related.
After the authoring process is complete and the mail is ready for delivery, Bill presses the Send button. The MAP parses the document and assembles the media objects in the "MediaPack", collects the necessary information on the MIME agents inside the document, and places it in the MMAIDB. The parser places the "Orchestrator" along with the MMAIDB inside a multipart/mixed package.

The whole message is sent as an application/X-MMagent. At the receiver's site, if the receiving mail tool is compatible to that of the sender's, the agents as depicted in Figure 4.8, are presented with the same characteristics; such as spatial and hierarchical structures and names. The receiver can then peruse the document by double-clicking on the agents. On the other hand if the mail tool is not compatible the mail presentation as depicted in Figure 5.4 displays the mail contents including the "Orchestrator" and the MMAIDB to the user.
This multimedia mail message has been composed using the MOBILE MIME Agent tool. If you have a Java interpreter, double-click on the icon titled "Orchestrator", else if you wish to navigate the media objects separately double click on the media icons.

Orchestrator
MMAIDB
Proposal
Graph_(1).gif
Graph_(2).gif
Why?

Figure 5.4: A sample mail presented in Eudora™ without the Java interpreter.

The user is then asked whether or not a Java interpreter is available. In the presence of a Java interpreter, the user is instructed to double-click on the "Orchestrator’s" icon to launch the Java program which will build the mail presentation as depicted in Figure 5.5. The multimedia mail presentation in this case is similar, as depicted in Figure 4.8, to that of our tool. Activating the multimedia components is simply achieved by double clicking on the media icons or folders. This is possible due to the pre-defined methods of interaction with the agents set by the "Orchestrator".
Marcel Karam, 1:02 PM +0200, So what about that

This multimedia mail message has been composed using the MOBILE MIME Agent tool. To view the media, double click on the icons.

In our opinion, our approach in using the visual mobile agency concept not only solves the interoperability and platform independency problem; rather, it strongly promotes the ease of practical use in the domain of collaborative work.
6 CONCLUSION

The work in this thesis started with a soft introduction to the concept of multimedia mail systems. We discussed the reasons behind the need of such systems, the different components that form a multimedia mail message, and the current limitations, such as dealing with large data, message fragmentation, shortage of processing power, and the problems associated with authoring and presentation in a multiplatform environment.

The work then progressed to an in-depth search on the leading cause of the delay in the "take-off" of this technology – the standardization issues. In examining the latter, we provided a comprehensive literature survey on the Internet leading carriers: MIME and X.400. We also examined some proposed solutions that aimed at solving the interchange format dilemma.

Dealing with the concepts of multimedia mail authoring and presentation, we studied the following: the Linear, Hypermedia, and Active models. The prototypes and/or applications associated with the mentioned models are Netscape™, NCM over the WWW, and ActiveM³ respectively. In each model, we examined the document structure, the message structure, and the authoring and presentation styles. We then introduced our Mobile MIME Agent multimedia mail authoring tool and looked at different areas of its graphical interface design.

Finally, we introduced the conceptual structure of our mail tool which included: the "Orchestrator", a Java program responsible for loading the MIME agents at the receiver's site; the MMAIDB, an internal database collected by the MIME-Agent-Parser and used by the "Orchestrator"; and the "MediaPack", a linear list of the multimedia elements.
Our work on the Mobile MIME Agent authoring model is based on our belief that visual interactivity and direct manipulation of interface objects can greatly enhance the usability of a system. In addition, our approach facilitates the navigation of media components during both the authoring and presentation phases.

The interface in our prototype was implemented using the Application Builder Classes (ABCs) of Prograph CPX™ [33]. Our decision to use Prograph for the implementation of the interface stems primarily from the fact that Prograph is a great tool for rapid prototyping. The ABCs classes in Prograph provided us with effective and efficient built-in procedures that allowed us to easily create screen objects, such as menus, windows, floating palettes, and items within windows. In the following paragraphs, we will discuss the advantages of our authoring and presentation model in comparison with that of the following: the Active; Hypermedia; and Linear models.

Our prototype interface and authoring style provide the user with an interactive style during both the authoring and presentation phases; whereas users in the Linear model are not provided with the same feature. In the Linear model, users are forced to compose by linearly adding attachments, and individually retrieve and view them without any sense of presentation or playback.

In comparison with the Hypermedia model, our mail tool not only achieves interoperability, it also simplifies the authoring, modification, and navigation of the mail contents. Moreover, in the Hypermedia model, navigating a large mail message can lead the user to follow many links which in turn might cause some misunderstanding of the message’s meaning; whereas in our prototype, the media components contained in the mail message are visually manipulated and navigated.

In the Active model, active multimedia mail is a program that is executed on a remote
site - a mail composer in this case is therefore a tool for creating such a program. While this approach is intuitive, the composing tool has to be extremely sophisticated to accurately interpret the list of spatial and temporal events and their corresponding functions that constantly take place during both the authoring and presentation phases. The system then translates all the actions in the document into a program that might not execute correctly on a heterogeneous system. Moreover, authoring a hierarchical multimedia mail structure using the Active model can be a difficult task especially for novice users who are migrating from textual to multimedia mail.

In comparison with our approach the introduction of containers and agents in a drag and drop fashion makes the composition of hierarchical structures such as multipart/parallel or multipart/related an easy task. Moreover, the implementation of a general algorithm ("Orchestrator") that is written in Java assures – in case of mail tool incompatibility and the presence of a Java interpreter – the presentation and navigation of the mail will still comply with our prototype. This characteristic in the "Orchestrator" allows the user to see the presentation in a "WYSIWYG" fashion and navigate the mail with the agent/container convention. Therefore, the "Orchestrator" can be thought of (in the presence of a Java interpreter) as a mail viewer that ensures the presentation of a multimedia mail in a heterogeneous environment.

On the other hand, even if the recipient’s site has an incompatible mail tool and the Java interpreter is absent, the user can still view the media separately.

6.1 Future Work

In this thesis, we have outlined the general concept of MIME agent mail tool. In addition, we have stressed the need for a multimedia mail tool that provides the average mail user
with an authoring environment where messages can be visually, interactively, and easily composed. We then introduced our prototype of a composing tool for authoring multimedia messages based on the visual mobile agency concept. There are, for example, open questions concerning the optimization in the design of the algorithm of the "Orchestrator" for the efficient communication and effective presentation of the multimedia message objects.

Our decision to use a paper-like document format was based on the fact that such format does not interfere with the MIME standard nor does it require a special representation in the MIME message file. Moreover, scanning, parsing, and presenting the multimedia mail document using the paper-like format is easier and more manageable.

The paper-like format comes with its own problems. For example, the initial document form is empty and there are no restrictions on where to place the MIME agents. This characteristic in the document might lead novice users to place MIME agents in an unorganized fashion. Therefore, further studies and research are suggested to solve this problem. One approach to solve the paper-like document format is to investigate the effect of incorporating a visual compiler and/or interpreter that monitors the activities in the document during the authoring phase. A possible task for the visual compiler and/or interpreter is to provide the user with the next possible position in the document following either a top-down or a left-to-right path. Our future research will focus on the above issues.
7. REFERENCES


[23] Djamal-Eddine, B., “Attaching Documents to E-mail Messages Sent Across the Internet”, <<bensize@un.org>>, 1996.


8. APPENDICES

Appendix A - MIME Content-Types: A complete up to date list

<table>
<thead>
<tr>
<th>Type</th>
<th>Subtype</th>
<th>Typical File Extensions</th>
</tr>
</thead>
<tbody>
<tr>
<td>text</td>
<td>enriched</td>
<td>html htm</td>
</tr>
<tr>
<td></td>
<td>html</td>
<td></td>
</tr>
<tr>
<td></td>
<td>plain</td>
<td></td>
</tr>
<tr>
<td></td>
<td>txt</td>
<td></td>
</tr>
<tr>
<td></td>
<td>tab-separated-values</td>
<td></td>
</tr>
<tr>
<td></td>
<td>richtext</td>
<td></td>
</tr>
<tr>
<td>multipart</td>
<td>alternative</td>
<td></td>
</tr>
<tr>
<td></td>
<td>appledouble</td>
<td></td>
</tr>
<tr>
<td></td>
<td>digest</td>
<td></td>
</tr>
<tr>
<td></td>
<td>header-set</td>
<td></td>
</tr>
<tr>
<td></td>
<td>mixed</td>
<td></td>
</tr>
<tr>
<td></td>
<td>parallel</td>
<td></td>
</tr>
<tr>
<td>message</td>
<td>external-body</td>
<td></td>
</tr>
<tr>
<td></td>
<td>news</td>
<td></td>
</tr>
<tr>
<td></td>
<td>partial</td>
<td></td>
</tr>
<tr>
<td></td>
<td>rfc822</td>
<td></td>
</tr>
<tr>
<td>application</td>
<td>activemessage</td>
<td></td>
</tr>
<tr>
<td></td>
<td>andrew-inset</td>
<td></td>
</tr>
<tr>
<td></td>
<td>applefile</td>
<td></td>
</tr>
<tr>
<td></td>
<td>atomicmail</td>
<td></td>
</tr>
<tr>
<td></td>
<td>commonground</td>
<td></td>
</tr>
<tr>
<td></td>
<td>cybercash</td>
<td></td>
</tr>
<tr>
<td></td>
<td>dca-rft</td>
<td></td>
</tr>
<tr>
<td></td>
<td>dec-dx</td>
<td></td>
</tr>
<tr>
<td></td>
<td>eshop</td>
<td></td>
</tr>
</tbody>
</table>
iges
mac-binhex40
macwriteii
mathematica
msword
news-message-id
news-transmission
octet-stream
tar
dump
readme
bin
uu
exe
oda
oda
pdf
pdf
postscript
ps
eps
ai
remote-printing
riscos
rtf
rtf
slate
wita
wordperfect5.1
dx-dvi
dx-pdf
dx-tar
dx-tex
dx-www-form-urlencoded
dx-www-pgp-request
x-www-pgp-reply
x-www-local-exec
zip
zip

image
gif
gif
iew
ief
jpeg
jpeg
jpg
jpe
rgb
tiff
tif
xhm
xhm
xpm
xpm
x-xwindowdump
xwd
x-pict
pict

audio
basic
au
snd
x-aiff
aif
aiiff
aiffc
x-wav
wav
<table>
<thead>
<tr>
<th>video</th>
<th>mpeg</th>
<th>mpeg mpg mpe</th>
</tr>
</thead>
<tbody>
<tr>
<td>quicktime</td>
<td>qt mov</td>
<td></td>
</tr>
<tr>
<td>x-msvideo</td>
<td>avi</td>
<td></td>
</tr>
<tr>
<td>x-sgi.movie</td>
<td>movie</td>
<td></td>
</tr>
</tbody>
</table>
Appendix B - Samples of the Prograph CPX code

Figure 7.1: The MIME agent classes in Prograph.
Figure 7.2: The method calls for extracting MIME agent information.
Figure 7.3: The method calls for parsing the document and sending the mail.