MASSEY UNIVERSITY

TOWARDS THE PAPERLESS OFFICE: AN INTRODUCTION TO ELECTRONIC STRUCTURED DOCUMENT INTERCHANGE

A THESIS SUBMITTED TO

THE FACULTY OF TECHNOLOGY
IN CANDIDACY FOR THE DEGREE OF MASTER OF TECHNOLOGY
(COMPUTING)

DEPARTMENT OF PRODUCTION TECHNOLOGY

BY
P.J. FRASER

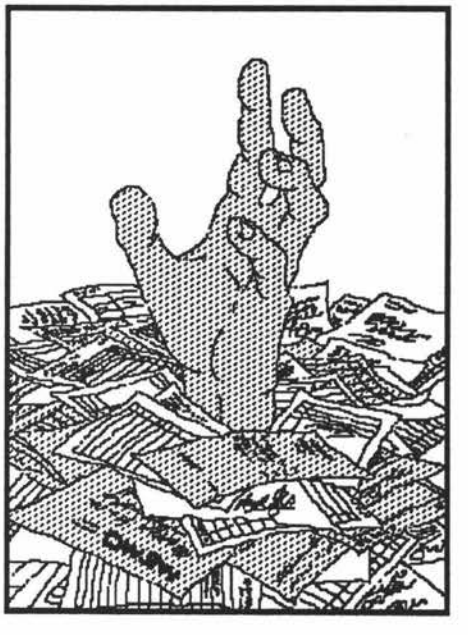
PALMERSTON NORTH
NEW ZEALAND

NOVEMBER, 1992

Towards the Paperless Office:

An Introduction to Electronic Structured Document Interchange

P.J. Fraser



To Mum and Dad.

Without whose love and support
this would not have been possible.

Abstract

Despite the advances in computing technology and office automation, and forecasts of the paperless office having become a reality by now, there remains the fact that very few companies would face less paper today than they did five years ago. Offices today are still deluged with paper because current office automation technology has failed to address one aspect of paperwork common in the office environment: the electronic equivalent to structured internal paper-based documents.

Electronic structured document interchange (ESDI) has been proposed as the last remaining technology in providing the complete infrastructure for the "paperless office." Complementing current electronic office system technology, including imaging technologies, electronic mail, and electronic data interchange, ESDI was designed to provide the electronic equivalent to structured internal paper-based documents.

Electronic structured document interchange is the intra-company computer-to-computer processing of business transactions in a format that allows the receiver to process the transaction by traditional business practices. Fundamentally, ESDI is a data processing concept that spans a single business enterprise, providing the complete electronic equivalent to the handling and processing of internal paper-based documents. The rationale being to take advantage of the benefits of electronic processing and delivery, while retaining traditional business practices. In some respects, ESDI systems have the potential to improve business practices by providing capabilities that are simply not possible with traditional paper-based systems.

This emerging technology, the justification for such a technology, and features of the technology, including details of the administrative ESDI system implemented at Massey University, are discussed.

Acknowledgements

My sincere thanks go to my two supervisors, Dr. R.I. (Bob) Chaplin and Dr. I.F. Boag, and to Associate Professor Mr. C.A. Freyberg, Director of Computing Services, and Mr. G.W. Bahlman, (now) Acting Director of Computing Services for making this project possible. To the visionary Ian Boag, whose experience in data communications and electronic mail inspired my interest in the project. To Chris Freyberg and Gerrit Bahlman who, knowing Ian's commercial achievements in this area, saw potential for the project at Massey, and provided the required resources and access to the Campus Network. And to Bob Chaplin, without whose direction this project may never have reached completion.

I would also like to thank the many other people who have made this project possible. In particular, I would like to thank Mr. S.K. Quinn (Systems Programmer) and Mr. A.G. Eustace (Systems Software Manager) for their help in implementing the various stages of the project and their patience, particularly when trying to explain to me some of the technical aspects. Finally, I would also like to thank Dr. E.A. (Ted) Drawneek (Senior Consultant), Mr. R.D. (Bob) Lambourne (Senior Consultant), and Mr. R.M. van Laar (Workshop Technician), for their technical assistance.

Contents

CHAPTER

1

Introduction

Whatever happened to the "paperless office?" For the last decade, forecasts have suggested that the "paperless office" was to have been a reality by now. In other words, we should all be using computers at work and enjoying the advantages of sharing information electronically.

Despite the advances in computing technology and office automation, and forecasts of the paperless office having become a reality by now, there remains the fact that very few companies would face less paper today than they did five years ago. There is no doubt, however, that automation has changed our work lives. But the changes are occurring very differently from the way the office pundits predicted.

The most obvious change has taken place at the desktop level. In the early 1980s, very few office workers had personal computers. Today there are more than 35 million personal computers in use, many of them on office desktops. As more business people adapted to personal computers, many firms started to interconnect them enabling users to share information and printers. Others provided office automation applications from terminals connected to a host computer.

Today most of us rely heavily on computers. They allow us to be more productive through the use of electronic mail, word processing, calendaring, spreadsheets, and access to external databases. Our personal and office productivity has improved, even if we continue to process more and more paper.

In moments of frustration, we can visualise the ideal office of the future that provides access to information at our fingertips, eases global communication, and allows a full exchange of media (text, data, voice, images, videos, handwritten notes) with a customised personal interface.

Whether that ideal soon becomes a reality, we cannot become so infatuated with it that we ignore our current office environment. Useful and sophisticated technology that is

either currently available or emerging can make the office more productive and eat away at the paper mountain.

Storage technologies, imaging technologies, electronic mail, and electronic data interchange are some of the technologies currently available to automate the office and move it closer to the paperless environment. These technologies are discussed in Chapter 2, "Towards the Paperless Office."

Through the use of storage technologies and other records and text management tools, much of the proliferation of paperwork in the office has been contained. Records and text management tools allowed paper documents to be classified, organised, circulated, retained, stored, and easily retrieved, reducing the necessity for multiple copies of a particular document to be in circulation.

Imaging technologies and image processing provided the greatest potential to shrink the paper mountain. Image processing was a similar concept to the records and text management tools associated with paper-based storage technologies. Instead, however, paper-based documents were scanned through an optical scanner, which sent the digitised image to a computer. The computer then controlled the storage and management of these electronic equivalents to the paper-based document.

As more people adapted to personal computers and businesses realised the potential in networking computers, new technologies developed making office workers more productive. However, companies continued to generate enormous amounts of paper documents.

Electronic mail eliminated some of the paper documents generated within an office. Office workers simply keyed in the message at a terminal, added the person's name and a heading, and the message was transmitted to the recipient. Electronic mail offered improvements in delivery speed, reliability, security, and communication, compared to standard mail delivery.

Electronic data interchange, which enabled firms to conduct business directly by computer, virtually eliminated the generation and transfer of paper documents between companies. Using electronic data interchange a company generated standard business transaction data, transmitted it electronically to the receiving company, and the receiving company used the data as input to the receiving company's application, just as if it had been manually entered. Electronic data interchange provided the opportunity to improve productivity for the firm as a whole, while reducing the handling of paper documents.

Why then, were offices still deluged with paper?

Current office automation technology has failed to address one aspect of paperwork common in the office environment: the electronic equivalent to structured internal paper-based documents.

Structured internal paper-based documents are often transitory and proprietary in nature and hence are unsuitable to be replaced by any currently available electronic office system. These documents, being transitory in nature, often being passed from person to person during processing, are not suited to image processing, which is used solely for static documents. Neither are these documents suited to electronic mail systems because of their structured nature. Electronic mail systems allow for the delivery of free format text, where the layout of such text is inconsequential as no further processing is required. Electronic data interchange technologies would provide the closest electronic equivalent to structured internal paper-based documents. However, electronic data interchange is a well-defined business practice, and because the documents are for internal business use and are not sent between trading partners, and because of the often proprietary nature of the documents (providing no standard format), electronic data interchange cannot be used.

A new technology is required to provide an electronic equivalent to structured internal paper-based documents. A new technology, electronic structured document interchange (ESDI), has been proposed as the last remaining technology in providing the complete infrastructure for the paperless office.

Electronic structured document interchange is the intra-company computer-to-computer processing of business transactions in a format that allows the receiver to process the transaction by traditional business practices. Fundamentally, ESDI is a data processing concept that spans a single business enterprise, providing the complete electronic equivalent to the handling and processing of internal paper-based documents.

This emerging technology, the justification for such a technology, and the features of the technology are discussed in the final section of Chapter 2.

Realising the potential benefits that could be gained from electronic structured document interchange, Massey University, through initiatives from Management Information Services, proposed an ESDI implementation to facilitate the administrative functions of the University. The aim of the project was to set up at least one, and possibly more, fully functional systems utilising ESDI.

Chapter 3, "The Massey University Electronic Structured Document Interchange Initiative," describes the environment of the ESDI initiative, highlighting the needs for such a system and the appropriateness of the University for such a system.

Having set the aim of the initiative, to create at least one, and possibly more, fully functional systems utilising electronic structured document interchange, a development strategy for the on-Campus ESDI initiative was established.

The idea behind ESDI was to provide the complete electronic equivalent to the handling and processing of internal paper-based documents. The rationale being to take advantage of the benefits of electronic processing and delivery, while retaining traditional

business practices. In some respects, an ESDI system had the potential to improve business practices by providing capabilities that were simply not possible with the traditional paper-based business systems.

While it is easy to envisage the ideal paperless environment, an environment that initiates, processes, and archives electronic transactions as an integral part of traditional business practice, it is not as easily achieved.

It was decided that a systematic approach was required for the development of the on-Campus ESDI system. The idea was to implement the system in stages, with each stage increasing in functionality. The purpose of such a strategy would allow the system requirements and system development (design and construction) of the envisaged final stage to be adequately planned for, throughout successive stages, from early on in the development. The staged development is discussed in Chapter 4, "The On-Campus Electronic Structured Document Interchange Development."

The electronic equivalents of internal paper-based documents are a fundamental part of an ESDI system, and their simplicity and ease of use are critical to its success. These "electronic forms" must provide an effective and efficient replacement of paper-based documents in order to retain traditional business practices and to be preferred to the use of paper-based systems. To be effective, the electronic forms need to be easily constructed and should provide all the content of its paper-based counterpart. While, to be efficient, the electronic forms need to be easily edited and manipulated, providing all the functions common to the processing of paper-based transactions.

A mechanism was available within the Department of Production Technology to provide the desired electronic equivalent to paper-based documents. The mechanism was simple, in that an electronic representation of a paper-based document could easily be constructed, and powerful, in that the mechanism could effectively provide the functions common to the processing of paper-based transactions. This "forms engine" was the fundamental component of the ensuing on-campus ESDI development, and is discussed in Chapter 5, "Paper's Counterpart: The Electronic Form."

The intention of the first stage of the on-campus ESDI development, referred to as Single Form / Single Destination (SFSD), was to implement a particular electronic form whose destination was predetermined. The ESDI system, for this first stage, was very simplistic. The ESDI system would display a single rudimentary electronic form, the user would fill in the form details, and when transmitted, the form details would be delivered to the predetermined destination.

Cognisant of the aims of the initiative, the Class Roll Request Form was implemented. The purpose of the Class Roll Request Form was to initiate the extraction of class roll information residing on a database that was not a part of the Campus Network.

Using the ESDI system a lecturer or faculty administrator filled in the Class Roll Request Form details, and when transmitted, the form details were delivered to Management Information Services. Details of the SFSD implementation are given in Chapter 6, "A SFSD Example: The Class Roll Request Form."

Since ESDI replaces the paper document environment with an electronic one, management and auditors face the challenge of how to implement ESDI technology to attain business objectives and appropriately control its associated risks. Many of the controls used in paper document processing are simply not effective in the ESDI environment. Chapter 7, "Security Aspects of Electronic Structured Document Interchange," establishes the business risks associated with ESDI, and examines the associated control and security aspects.

As the on-campus ESDI implementation evolves, the need to reassess the controls over the existing environment and applications becomes more apparent, and the additional risks, that may derive from the implementation of ESDI because of the changes to systems, procedures, and operations, should be addressed.

The aim of the second stage of the on-campus ESDI development, referred to as Multiple Forms / Multiple Destinations (MFMD), was to increase the functionality of the ESDI system implemented in stage one by making a facility available that would offer a choice of electronic forms and a choice of destinations. Each electronic form would provide the equivalent of an internal paper-based document and could have a predetermined destination.

The ESDI system would function as follows: if the forms do not have a predetermined destination (or if there is more than one destination provided), a menu of destinations is displayed and the user selects the appropriate destination; a menu of available forms is displayed and the user selects the appropriate form; the form is displayed, and the user fills in the form details; finally, when transmitted, the form details are delivered to the proper destination.

The details of the MFMD implementation, and the Financial Transaction Request Form, are presented in Chapter 8, "A MFMD Example: The Financial Transaction Request Form." The purpose of the Financial Transaction Request Form was to enable a department to request the account status and transaction reports, for a nominated account number, over a specified period. Again, this information resided on a database that was not a part of the Campus Network.

In summary, the effectiveness of ESDI as a viable technology for the office is discussed in Chapter 9, "Discussion and Conclusions," along with recommendations for future work.

CHAPTER

2

Towards the Paperless Office

A purchasing manager recently asked if my company's purchasing system could print three copies of the purchase order. "Certainly," I said, "but why don't you use a three-part form?"

"You don't understand," he said. "We already use a five-part form, but we need 15 copies of the purchase order."

"Why do you need so many copies?"

"We send two copies to the vendor. The vendor keeps the original and returns one copy as an acknowledgement. We send four copies to receiving to allow for partial receipts, four more copies to accounts payable to allow for partial invoices, and one copy goes back to the inventory planner to notify him that the order has been placed."

"That leaves four copies unaccounted for," I said.

"We file those in different order," he said. "We file one copy by PO number, one copy by part number, one by vendor number, and one by PO date. This makes it easy for us to locate a particular purchase order when we only have one piece of information."

Luber [1]

What's the magnitude of the paper problem? Statistics indicate that United States corporations generate over 30 billion original documents annually [2], and, depending on the nature of the document, those originals are reproduced many times: a purchase order may be reproduced as many as fifteen times, for various internal reasons, as in the above extract from Luber [1]. It can cost as much as US$25,000 to generate documents that fill one four-drawer file cabinet and an additional US$2,000 annually to maintain that file cabinet [2].

Whatever happened to the "paperless office?" For the last decade, forecasts have suggested that the "paperless office" was to have been a reality by now [3,4,5,6,7]. In other words, we should all be using computers at work and enjoying the advantages of sharing information electronically.

There would be very few companies world-wide facing less paper today than they did five years ago. The American Paper Institute reports that the amount of paper used for file folders alone doubled during the 1980s [3]. It also claims 95% of records continue to be paper-based [8]. While Moore [9] goes further, stating that the 95% of all information stored on paper stacks up to 1.3 *trillion* documents per year.

In a competitive business environment, the ability to move up-to-date information may literally be the difference between the life and death of a company. For decades, industry pundits have painted a picture of the paperless office. Yet, companies continue to generate enormous amounts of paper documents. There is no doubt, however, that automation has changed our work lives. While a panacea still remains far away, many companies are successfully using various techniques to automate many of their operations [10,11,12,13,14,15,16,17].

In the early 1980s, very few office workers had personal computers. Today there are more than 35 million personal computers in use [3], many of them on office desktops. As more people adapted to personal computers, many firms started to interconnect them enabling users to share information and printers. Others provided office automation applications from terminals connected to a host computer. Connected personal computers are fast becoming the rule and standalone personal computers the exception [18].

Today most of us rely heavily on computers. They allow us to be more productive through the use of electronic mail, word processing, calendaring, spreadsheets, and access to external databases. Our personal and office productivity has improved, even if we continue to process more and more paper [3].

In moments of frustration, we can visualise that ideal office of the future that provides access to information at our fingertips, eases global communication, and allows a full exchange of media (text, data, voice, images, videos, handwritten notes) with a customised personal interface.

Whether that ideal soon becomes a reality, we cannot become so infatuated with it that we ignore the current office environment. Useful and sophisticated technology that is either currently available or emerging can make the office more productive and eat away at the paper mountain.

The following sections describe some of the technologies currently available to automate the office and move it closer to the paperless environment.

The first section, Storage Technologies, discusses records and text management tools used to assist in the classification, organisation, circulation, retention, storage and retrieval of files, documents, and text.

Imaging Technologies discusses image processing, which has perhaps the greatest potential to shrink the paper mountain. Image processing uses a computer to store and manage entire documents. These documents are scanned through an optical scanner, which sends the digitised image to a computer. The computer processes it and displays it on a screen, sends it around to individuals for consultation and review, prints it, or stores it in an electronic archival system.

As companies generated electronic documents, they needed mechanisms to move them from one department to another. Electronic mail, which is discussed in the third section, filled this void.

The fourth section, Electronic Data Interchange (EDI), describes the technology that allows trading partners in two or more organisations to exchange business transaction data in structured formats so that they may be processed by computer application software.

Finally, the last section describes the emergence of a new technology that attempts to fill the apparent void that exists between electronic mail and electronic data interchange. This technology I call electronic structured document interchange (ESDI).

2.1. Storage Technologies.

Ten years ago, it was quite fashionable to talk about the office of the future as a paperless entity, one in which workers would have CRT screens and keyboards at their desks. According to the wisdom of the day, instead of passing paper around, we would create documents on our systems, send them electronically, and print them out remotely as needed.

At that time, however, experts overlooked two items, which today explains why the paperless office has not become a reality. One is that people like hard copy. Although things may change in the future, right now it is more convenient to lug around hard copy than a terminal and hard copy can be toted in one's briefcase.[1] People also like to play with hard copy: write on it, spread it out, reorder it. Second is the increasing number of terminals and personal computers on individual desks. These devices all generate originals, which people then take to copiers to reproduce [6].

The phenomenal growth of microcomputers and micrographics has resulted in more integration of paper systems with automated and micrographic systems. Technology does not, however, reduce the amount of paper used. On the contrary. American business now produces 600 million computer printouts, over 230 million photocopies and 76 million letters every day, twice what it was ten years ago [19]. The increased paper load needs proper handling. The result of electronic and micrographic technology has been to increase dramatically the amount of information that can be processed.

Larger quantities of information processed leads to more information to review or sift through, and increases the need to distil the vast quantities of it accurately.

In spite of advances in office automation, paper is still the most popular information medium. So much so that the field of records management has developed into a profession to handle the increasing amounts of paper, with records managers making use of advances in office technology to store and retrieve vital records effectively [7].

Today's records management options range from manual set-ups based on highly effective colour coding [20] to automated systems designed around the latest advancements in electronic technologies [21].

On the surface, the increasing use of computers and other components of today's electronic office would seem to have placed paper storage systems in jeopardy. But in fact,

---

[1]Notebook computers, computers small enough to fit inside a briefcase and with all the processing power of desktop personal computers will eventually fill this niche. The cost of notebook computers, however, remains a stumbling block to general acceptance.

the opposite has become the norm. Instead of reducing or replacing paper, advances in office automation have actually increased the production of paper records [22,23].

Not only are computer printouts increasing the volume of records, word processing fosters ever-growing numbers of traditional letters, memoranda, reports and other documents. And with copiers spitting out mountains of paper, the volume of documents filed can be monumental.

Increased volumes of paper records do not mean we simply must resign ourselves to slow and efficient filing. Shredders have played at important part in cutting down on waste storage space [24], while the development of records management systems have been designed to handle today's larger-than-ever office demands.

The technology of micrographics, for example, provides a means of preserving documents in a fraction of the space they would occupy on paper. Records can be converted to microfilm or a related microform and then stored for recall as needed, requiring much less space than traditional documents.

In situations involving large numbers of records, the micrographic process can be enhanced through computerisation. Computer-assisted retrieval (CAR) and computer output microfiche (COM) systems bring even more flexibility by adding the speed of microprocessing to the mass storage capabilities of micrographics.

Another option is optical disc technology. Instead of using microfilm, optical disc systems store information on a rotating disc through the use of a laser or similar light source. Because a single disc can store many thousands of records, optical disc technology provides an attractive alternative where large-volume storage is required.

2.2. Imaging Technologies.

When the computer first appeared in the workplace, many visionaries boldly predicted the beginning of the paperless office. The computer evolved, became smaller and more powerful, but paper still cluttered desktops and tested the limits of overtaxed filing systems. The computer, though equipped with powerful processing capabilities, was hamstrung by traditional data storage methods that relied on magnetic media.

With the advent of the laser and optical disk storage, however, the dream of a paperless office has moved an important step closer to reality [25], and many insurance and finance companies are now realising its potential [26,27,28,29,30].

Systems that convert paper documents into electronic images, which can be stored on computer and retrieved on a monitor or laser printer, have been available for several years now [31]. A typical image processing system includes scanners to process documents, workstations with large, crisp monitors to read them, and a central processing unit hooked up to gigabytes of tapes and disks to store them. Office workers can ask for any page in the system and have it on their screens in seconds. Several people can look at the same document at once, and no papers can be misfiled or doused with coffee. If the system includes a network, electronic files can be sent back and forth between co-workers [32].

Although laser optic filing systems borrow heavily from computer technology, they are not computer systems in the typical sense. Instead, computer system technology is combined with laser optics and laser printing to provide accurate transformation from document originals to optical disk storage and back again. The quality of this transformation due to the high precision digitising process is better than photocopying, the speed of access is virtually instantaneous, and the accuracy of the reproduced document, depending on the system, can be the legal equivalent of the original [33,34].

The traditional motivation for using laser optic filing has been to reduce paper handling and storage, costly to companies both for its labour and its space requirements. A C.Itoh study of paper storage requirements aboard a U.S. Navy aircraft carrier found that a laser optic filing system could significantly reduce the space presently used for paper and microfilm storage, leaving enough room to add two more fighter jets or 100 more crew members [35].

Laser optic filing addresses the vast middle ground between micrographics and on-line magnetic data storage. Microfilm and microfiche are suitable for storing huge amounts of data that is rarely accessed, while on-line magnetic computer storage is better suited for applications that involve small amounts of data with repeated access by many people.

Laser optic filing permits many people (depending on the system) the opportunity to access large amounts of data regularly, and reproduce the paperwork with ease when necessary.

At the outset, optical filing systems had been developed as mass-storage filing systems with the idea of achieving a paperless office. New developments in optical technologies may finally deliver the paperless office. Today, the more sophisticated systems can accommodate electronic files with a mix of handwriting, word processing, and electronic mail. And since many systems use facsimile file formats for storing documents, they can accept facsimiles directly and send out a facsimile copy of any page on file [32].

2.2.1. Optical Technology.

Optical media refers to several different types of optical disks and optical cards that are read by a low-power laser beam [9]. They all feature very high density storage, resistance to damage, and immunity to electromagnetic influences that can destroy information on magnetic tapes, disks, or hard drives. Access to information is fast, and error rates are very low, when compared to manual-based systems. Because the laser device used to read the disks or cards does not contact the recorded surface, the wear and tear common to magnetic media is eliminated.

Initial market growth for optical media has been slower than originally predicted. This may be due to misconceptions that optical media represent merely a substitute for other storage media. Optical document storage is not a replacement for floppy disks, hard drives, or even microfilm. In fact, it is a new technology with special characteristics suited to different types of applications than magnetic storage. A description of the media is presented below.

2.2.1.1. CD-ROM (Compact Disk - Read Only Memory).

CD-ROM disks are a derivative from the digital audio compact disks used by the music industry. The physical disks are the same as those used for recorded music, but additional levels of error protection are added when they are to be used for data applications.

The music applications of CD-ROM have helped to reduce the cost of this type of media. One 4.75-inch CD-ROM disk can hold 600 megabytes, which is roughly the equivalent of 250,000 typed pages or 1,500 double sided floppy disks [36].

Information on a CD-ROM disk is represented by an outward spiral of pits moulded into one surface. The surface is first coated with a reflective metal layer, then coated with a protective lacquer. The outer surface of a disk is made of polycarbonate - the transparent plastic used to make bulletproof windows. Even if the polycarbonate surface is scratched, the data remain intact because lasers shine through the outer surface to reach the underlying layers.

Because it is a read-only medium mastered at high expense, CD-ROM is most appropriate for storing large multiple copies of information that do not require frequent updating. CD-ROM has advantages over paper and magnetic media because it can handle images, video, and voice applications simultaneously.

2.2.1.2. WORM (Write-Once, Read-Many).

WORM disks come in two major sizes, 5.25 inches and 12 inches. The 5.25 inch size holds 200 to 800 megabytes, while the 12-inch can store two to three gigabytes. For the 5.25 inch size, that translates to approximately 25,000 A4 size photographs (at 300 dots/inch), which would fill two four-drawer filing cabinets, or, for text only, 320,000 pages (at 2,600 characters/page) requiring 27 file cabinets. A 12-inch disk would hold five file cabinets of image data or 67.5 file cabinets of text [36]. In addition, the 12-inch disk can be mounted in a multi-disk machine called a jukebox to provide virtually unlimited storage.

At present, magnetic tape as a back up media is cheaper, but WORM disks have the advantage of quicker access and higher capacity. WORM media are particularly useful for storage of data that are no longer expected to be used frequently but which must be accessible in real time. Unlike tape archives, WORM media can be mounted jukebox style and remain accessible without wasting expensive direct access computer memory.

Multiple recording standards exist for WORM media. Therefore, they are used mainly for distribution of data within a single firm. WORM disks are ideally suited for storage of images. Because WORM media can be updated by software, but not physically erased, they leave permanent audit trails and can decrease time spent on magnetic tape back up. Today WORM media are the most widely used optical media for commercial document storage and retrieval systems.

2.2.1.3. Laser Cards.

Laser cards, the size of standard credit cards, are a new form of optical application. The laser card is used when data should stay with the individual or application. Similar to magnetic strip cards in function, but with greater capacity, laser cards can hold up to 200 megabytes of data on one side.

Because cards do not rotate, they are more appropriate than CD-ROM in environments subject to shock and vibration. Laser cards can also serve as a substitute for other high capacity storage media if arranged in a carousel for rapid access. They are particularly well suited for identification because they can carry a picture, fingerprints, and even a voice print if desired.

2.2.1.4. Erasable Optics.

Erasable optics are a recent development. Three experimental versions of erasable media are magneto-optic, phase change, and dye polymer. So far, only the magneto-optic version has proven commercial viability [36]. Because magneto-optics is the slowest of the three methods, further improvements and changes in erasable media standards are sure to be on the horizon.

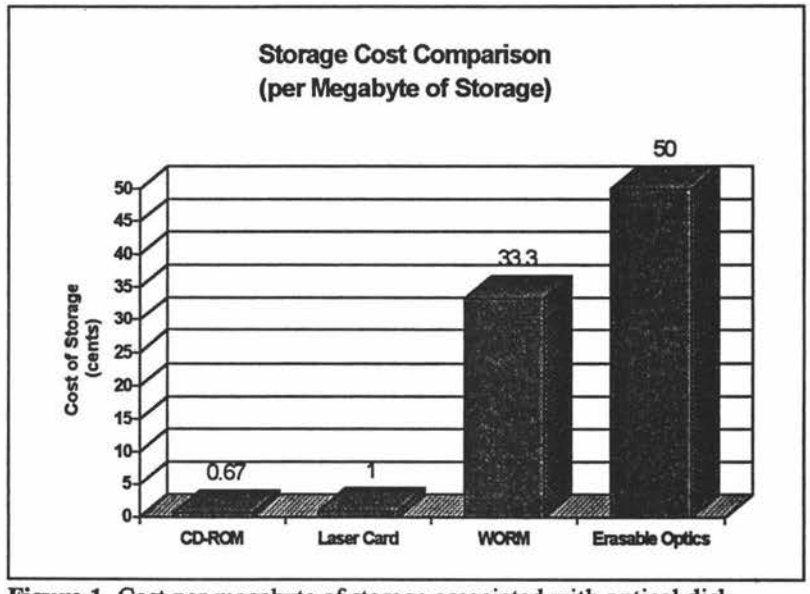


Figure 1. Cost per megabyte of storage associated with optical disk technologies.

Erasable optics come in cartridges with a capacity of 500 megabytes to one gigabyte. Erasable disks have the capacity and very nearly the speed of a hard disk. The capacity of the disks can be constantly and inexpensively expanded by adding cartridges. Data on cartridges are easier to transport and to secure under lock and key. Unfortunately, erasable media are, at present, rather expensive (Figure 1 shows a storage cost comparison between the different types of optical media).

2.3. Electronic Mail.

In the past there were two main types of communication within the office, the telephone and the letter. Both of which have their characteristic advantages and disadvantages [37].

The first characteristic of the telephone is that it is immediate: information is exchanged at the speed of light. It is also interactive: it makes possible a conversation between two people, each able to reply to the other, rather than a one-way communication. However, because it is interactive, it needs both parties to be available simultaneously. There is no permanent record, and no easy means of passing the message on to a third party without writing it down or saying it again. Each party knows that the other has heard the conversation - but not that they have understood. Finally, there are generally only two people involved: the person who initiates the call and the person receiving the call.

Electronic mail offers nearly all the advantages of the letter, with some of the advantages of the telephone. There are many different technical approaches, but the general principle is that each office worker has a "mailbox" on the system, to which mail can be sent. To send a memo you no longer need to involve the postal system. You simply key in the message at your terminal (or get a secretary to do it for you) and add the person's name and a heading. The next time they sign on at their terminal, they will find a message saying "you have mail." They can then examine their electronic "in-tray": they will see a list of the messages they have received, giving the name of the sender, the subject heading and the date and time.

At their leisure, they can then ask for any of the messages to be shown on the screen. After reading the message, they can then reply to it, discard it into an electronic "wastepaper bin," file it for future reference, hold it in their mailbox, or send it on to someone else with their own comments added. Often the fact that they have looked at it will be recorded on file, so that from time to time you can check your own listing of outgoing messages and see whether the recipient has looked at it yet.

2.3.1. Electronic Mail Features.

Electronic messages, like their paper equivalents, usually have two parts - an envelope and contents [38]. The envelope carries information (e.g., name and address) needed to route the message to the correct mailbox. The contents, comprising information that the sender wishes to transmit, can be separated into two parts - a header and a body. Header information contains predefined fields associated with the messaging process; such as "subject," "time sent," and "reply to." The body of the message is free-format text, input by the sender. The system will request any essential envelope and header information before sending a message. Figure 2 shows the basic structure of an electronic message. Basic mailbox facilities are discussed briefly below [39,40,41].

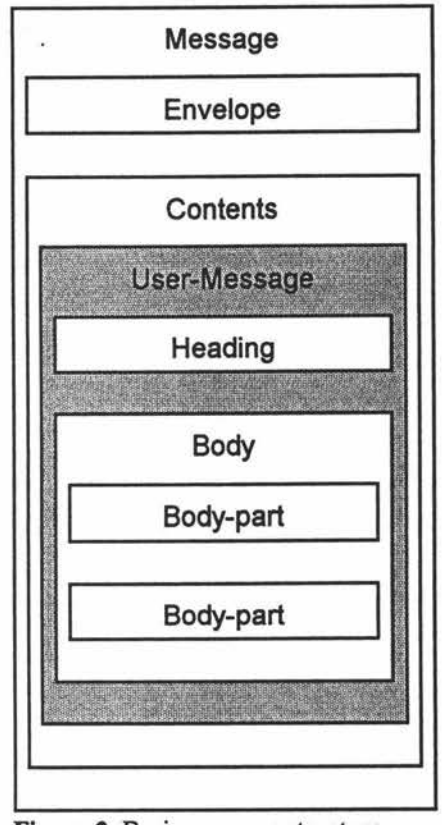


Figure 2. Basic message structure (according to the CCITT Message Handling System).

2.3.1.1. Creating the Message.

The primary difference between traditional mail and electronic mail systems is the medium of communication. Instead of delivering a piece of paper with information on it, an electronic mail system delivers just the information itself. In order to do this, the information must be translated into a format that can be transmitted through a communications network. Instead of typing text onto a piece of paper, the sender keys the text into a device, which translates it into an electronic format. Typically, this device is a computer terminal, with a keyboard much like a typewriter keyboard, and the capability to display text either on a screen or on a printer.

2.3.1.2. Transmitting the Message.

Once the message has been translated into an electronic code, it can be transmitted in that format over communications lines. A communications line can be just an ordinary telephone line, the most widespread and easily available type of network. However, there are also several types of communications networks designed specifically for electronic data communications. The most commonly used ones employ a technology called "packet switching," which insure high-speed, error-free data transmission. These networks support all kinds of communications, whether between an individual and a computer or from computer to computer.

Although packet switched networks can accommodate very high-speed communications, transmission speeds available for personal communication through electronic mail systems are generally limited. This is because of the limitations imposed by:

1. accessing the data communications network through voice grade telephone lines, and
2. the technology of the equipment currently available for use with standard phone lines [40].

Although voice grade phone lines continue to support most data communications, many new technologies for data communications are now coming into use, including microwave, satellite, fibre optics, and coaxial cable. The use of data communications technology that supports higher transmission speeds and higher capacity will be increasingly important as an organisation's volume of electronic message traffic increases.

2.3.1.3. Receiving the Message.

The methods by which mail is delivered in an electronic mail system may be divided into two general categories: direct delivery and centralised delivery [42].

Direct Delivery.

Information may be delivered directly from the sender to the recipient, or from one point in the communications network to another (see Figure 3). Such direct connect systems offer the advantage of immediate delivery of mail, since messages are sent and received simultaneously. However, these services also tend to require specialised, dedicated equipment.

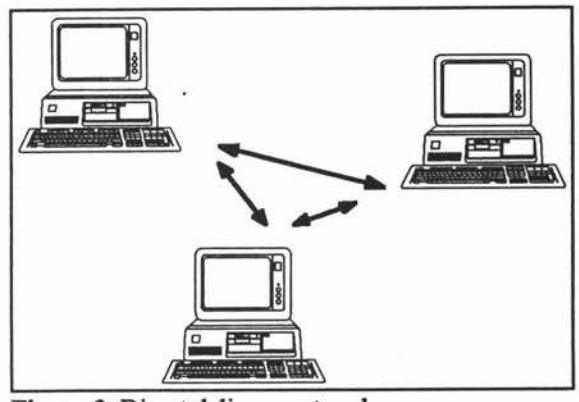


Figure 3. Direct delivery network.

Centralised Delivery.

The other general category of mail delivery is the centralised or "pick-up" delivery system in which all messages are sent to a central storage area rather than directly to a recipient. A host computer generally performs this storage function, acting like an electronic post office to hold the messages until the recipient is ready to pick them up (see Figure 4). In some cases, the computer can be instructed to forward the messages to the recipient automatically or at a specified time. In either case, the sender is relieved of the necessity to make a direct, real-time connection with the recipient.

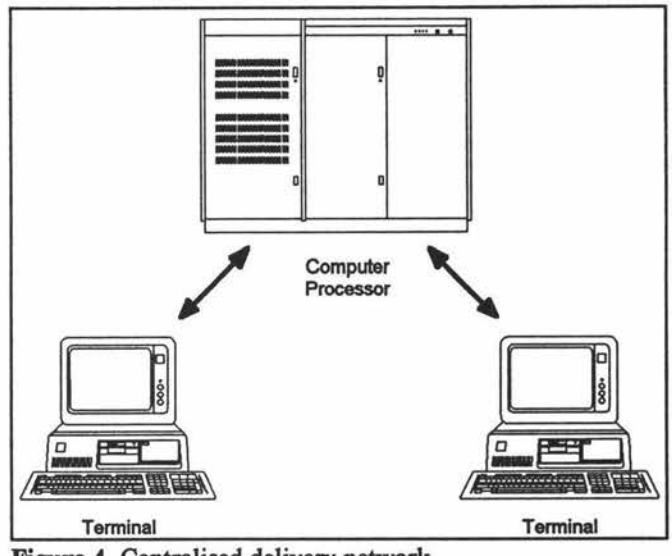


Figure 4. Centralised delivery network.

2.3.1.4. Filing and Retrieval.

In order for the recipient to read the transmitted message or information, it must be displayed in some format. The receiving point may be a printing terminal, a microcomputer, a word processor, or some other device. One of the significant differences

between standard mail and electronic mail systems is that the format of the received electronic mail message depends very much upon the type of equipment used to receive the message.

If the message is printed out onto paper, it is once again a separate physical item that can be stored in a file cabinet, interfiled with mail received through other channels. However, it is also possible to maintain a copy of the message in the electronic format, either in a local storage device, or through the services offered by the mail system used.

The advantages of maintaining on-line files are numerous. For example, if you have received a message that you know will be of interest to others, you can transmit a copy of the message without having to re-key the text. You can also add a note of your own to the original message, or perhaps add different comments for the different people to whom the message is being forwarded. You can also reply to the sender and include a copy of the original message to provide a context for your remarks. Basically, on-line storage allows repeated revision, manipulation, and reuse of the same information over and over again as it is needed in different situations.

2.3.2. Benefits of Electronic Mail.

The primary purpose of any mail system, electronic or otherwise, is the delivery of information from one person to another, as quickly as possible. A mail system must also be dependable, with procedures that ensure delivery of an item to the correct recipient, and must offer some measure of security. Because the information to be delivered may be in any format a mail system must be flexible. Finally, the system must be universal; that is, it must support the delivery of mail to any destination.

Electronic mail can offer improvements in delivery speed, reliability, security, and communication, compared to standard mail delivery [42,43,44,45]. However, there are still two other basic features of a mail system: the capability for universal delivery, and the capability to deliver materials in all kinds of formats. In these two areas, electronic mail services can present problems.

At present, there is no universal electronic mail system that can cover the wide delivery range of the regular postal service. However, there is more and more evidence that the process of standardising electronic mail formats, which started recently, will continue and eventually offer widespread possibilities for interconnection of mail systems.

The capability to deliver mail in all kinds of formats is another area where electronic mail systems have limitations compared to traditional mail. Special technologies are required to transmit graphics information and other specialised formats, and these usually require special equipment or transmission facilities.

As mentioned previously, benefits can be experienced through electronic mail in the areas of:

- reduced costs,
- improved speed of delivery,
- improved reliability of delivery,
- improved security,
- improved internal communications, and
- improved external communications.

Indeed, an electronic mail service has the potential to improve services by providing capabilities that are simply not possible with traditional mail systems [40].

2.3.2.1. Reduced Costs.

Cost is an important factor in the increasing popularity of electronic mail communication services. The traditional cost of a business letter typically includes the cost of dictation time by an executive, secretarial time to type and file, overhead and fixed costs, materials, and postage and mail handling costs. Of course, a number of these cost elements are present in any type of mail service, whether electronic communications methods are used or not. For example, overhead and fixed costs are likely to remain the same. Also, even if an executive keys in a message directly instead of dictating it, their time is still a cost factor. However, electronic options provide the opportunity to create a different mix of cost elements by reducing secretarial support requirements or handling costs. With many short or informal communications, electronic mail systems are both less expensive, in terms of total personnel cost, and more efficient.

Other cost savings (for example, earlier payment of bills or faster response) are more subjective, but are often the main reason for adopting an electronic mail solution. These more subjective cost savings are associated with the benefits discussed below.

2.3.2.2. Improved Speed of Delivery.

Because of the delays inherent in travel time and handling, it is rare for mail to be delivered through a postal system in less than two days, and regular delivery across country may take longer. Special overnight delivery services are available, but their labour-intensive nature makes such services very expensive. In comparison, one of the most important characteristics of electronic mail is the capability to transfer information instantly. Because electronic delivery is immediate, deadlines can be met more easily, questions can be answered without delay, and work can be completed faster and more efficiently.

2.3.2.3. Improved Reliability of Service.

Speed is an important consideration, but perhaps even more important is the assurance that a piece of correspondence has actually been delivered. Traditional mail systems provide no receipt capability, except in the special case of registered mail. To determine when an important letter or order has arrived at its destination, it is necessary to wait for a reply or, if it is really important, to contact the recipient. Many electronic mail systems, however, provide a reporting mechanism that allows verification of the precise time that the mail was read by the recipient as a standard feature. This is of crucial importance in a business environment where verification of a receipt or bid can be the difference between making or losing a sale.

2.3.2.4. Improved Security.

Another basic requirement for mail systems is protection from having private data exposed to the wrong eyes. While traditional mail systems provide adequate privacy protection for most applications, mail delivery through an electronic mail system can be designed for additional security features, so only authorised personnel can gain access to certain types of information. This kind of protection can be particularly valuable at higher management levels.

2.3.2.5. Improved Internal Communications.

In most business environments, there are two basic categories of messages: internal communications between departments or divisions of the same company or organisation; and external communications with other organisations, customers, suppliers, etc. Traditionally, the first type of communication is handled through personal delivery systems within the same building, combined with courier or postal service for intraorganisational mail to other locations. This mixture may result in discrepancies between the time in which information is received on-site, and when it is received in field offices and other off-site locations. With electronic mail, information is delivered to everyone on the distribution list at the same time, regardless of the location of recipients.

Another feature of internal mail is the need to route materials among a group of people to gather comments and generate a response. When one physical piece of mail is being routed, this process is slow, and annotating the same document can become messy. In an electronic mail system, it may be possible to store the document that is being reviewed in the system so that everyone can review it and add comments. In this way, one slow person on the routing list does not hold up the review; everyone gets a chance to look at it at the same time, and a more thorough review process can take place in a shorter time frame.

2.3.2.6. Improved External Communications.

External mail presents a different set of considerations, since it requires the essentially universal delivery capability of a standard postal system. It is far more difficult to develop a coherent approach to the implementation of an electronic mail system among diverse organisations than to develop it within an organisation. However, the development of electronic communications capabilities with major suppliers or large customers can enhance relationships with these key external contacts, by facilitating delivery of orders and by creating a direct connection for the quick resolution of problems.

2.4. Electronic Data Interchange.

The paperless office that has been predicted for so long may finally emerge this decade with the increasing use of electronic data interchange (EDI), which enables firms to conduct business directly by computer, virtually eliminating the transfer of paper documents [46].

Transmitting business data within a firm using communication networks is now commonplace in all industries. However, it is anticipated that by 1995, half of all intercorporate business documents will move between firms by EDI [47], documents that were previously prepared and despatched by traditional paper-based processes and postal services [48].

Electronic data interchange is the inter-company computer-to-computer communication of standard business transactions in a standard format that allows the receiving trading partner to process the transaction data by computer application software [49,50,51,52].

2.4.1. Electronic Data Interchange Features.

Fundamentally, EDI is a data processing concept that spans multiple business enterprises. It does this by relying on data communications methods and standard formats for the transmission, acceptance, and understanding of business data.

The following features characterise EDI:

2.4.1.1. Inter-Company.

Inter-company specifically refers to the electronic transmission of data between companies. This implies that the trading partner companies must have common communications capabilities. As this is not always the case, the companies involved can either purchase communication hardware and software, as required, or use the services provided by a third party to act as a buffer between themselves and their trading partners [53].

2.4.1.2. Computer to Computer.

Not only are the data travelling between companies, but they are travelling from computer to computer between computer applications.

Today the sender application generates a paper-based transaction and mails it to the receiver. In the EDI environment the sending application generates the business transaction, transmits it to the receiver, and the receiver uses the data as input to the receiving application. This places a great deal of responsibility on the sending application to generate a complete and accurate business transaction, and on the receiving application to interpret and use the data received. While paper-based transactions have sometimes required human intervention to complete or correct them prior to processing, machine readable transactions cannot be interpreted correctly by the receiving computer application if they contain ambiguities or errors. These exceptions must be processed by hand, which eliminates many of the benefits of transmitting them electronically.

2.4.1.3. Standard Business Transactions.

EDI does not refer to the transmission of electronic mail or other free form messaging capabilities. The information referred to here is intended to permit the receiver to perform a standard business transaction. Typical business transactions include purchase orders, purchase order acknowledgements, requests for quotations, quotes, invoices, bills of lading, and shipping notices.

2.4.1.4. Standard Format.

Finally, in order to be recognised by the receiving computer application, the transmitted transactions must be in a standard data format.[2] As a result of the commonality of data requirements among companies to perform a business transaction, several standards have been developed that have industry specific or cross-industry application. The standard that any one company supports for a specific business transaction is dependent on the industry in which it trades primarily.

---

[2]EDI standards primarily address the format of the business data. Thus, they address the application layer - layer 7 - of the OSI model.

Members of industries that have developed standards for their own use need only adhere to the format, syntax, and usage rules for that format to begin trading electronically with any of their trading partners in that industry.

Recently, two standard business transaction sets have gained widespread acceptance as cross-industry standards: the American National Standards Institute (ANSI) X.12 and, what will eventually form the basis of all international EDI systems, the UN/ECE EDIFACT (United Nations/Economic Commission of Europe Electronic Data Interchange for Administration, Commerce, and Transport) board standard, which submits messages it approves to the CCITT [48].

What makes EDIFACT the preferred standard is its focus on business function instead of replacement of a specific paper document. With the focus on function, transaction sets are developed as a series of predefined segments containing the information strings required to transact a particular business transaction. Each party to the transaction will find, in the segments, all the information it needs to complete a business transaction.

Since the standard format is meant to serve as a communication medium, no change of any internal application is required in order to implement EDI. Instead, a piece of software must be developed by both the sender and receiver, which acts as an application link between their internal application and the standard format. The sending application still generates the data for the paper-based transaction. However, rather than print it, the data necessary for the standard format of the transaction is mapped into appropriate fields. The transactions are communicated to the receiving party.[3] They are fed directly into the receiver's application link software, which maps the standard data fields into the input format required by the receiver's application software.

[3]As paper documents are carried in paper envelopes, EDI transactions are carried in electronic envelopes. Although the envelopes are not themselves business data, their formats are defined by the EDI standard groups. The envelopes have much the same content as the data elements standardised in 1984 and 1988 by X.400 and are analogous to the OSI presentation layer. Sample contents are sender and receiver ID, time and date stamp, and control numbers [54].

2.4.2. How Does Electronic Data Interchange Work?

The concept of EDI is simple, machine readable data is transmitted between trading partners' computers. Figure 5 represents the interchange of electronic data. It shows two trading partners, in this case a buyer and a seller, and an EDI data stream transmitted from the buyer's computer to the seller's computer.

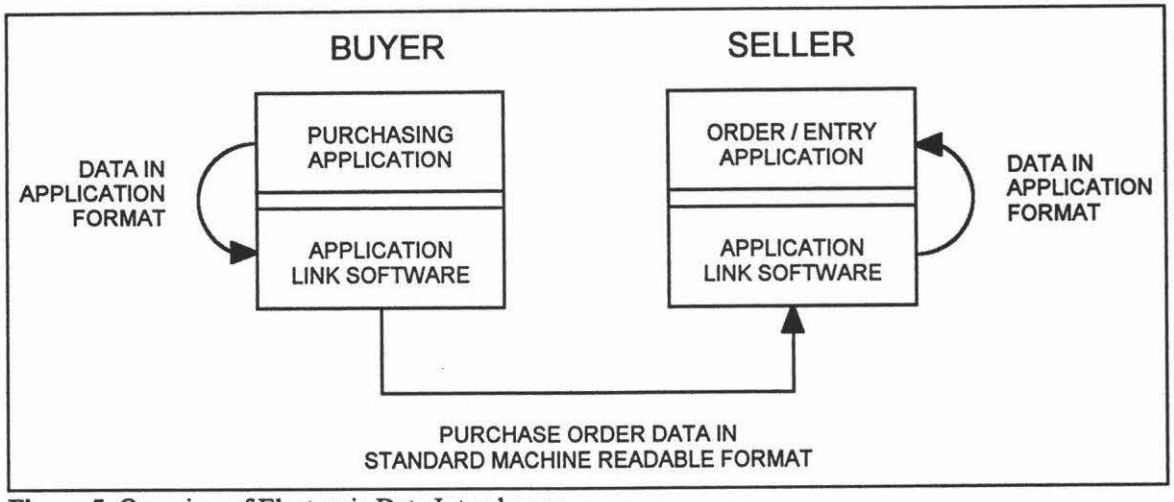


Figure 5. Overview of Electronic Data Interchange.

Supposing the data stream is a group of purchase orders; the buying company generates the transactions in its "gateway application," purchasing, as it always did. However, instead of using the purchase order data to develop the traditional paper transaction, it passes the data through an "application link" piece of software that maps it into a standard machine readable data format. That format is then transmitted to the seller's site, where it is passed through the seller's "application link," which maps it to the internal format expected by the seller's order/entry "gateway application." Order/entry processes it just as it would any incoming purchasing order.

2.4.3. Benefits of Electronic Data Interchange.

The EDI growth across their industry is reason enough for many enterprises to actively evaluate EDI. However, the ability of firms to generate real dollar savings, compress business time, and, in some cases, gain competitive advantage, is creating

substantial growth in the area of electronic data interchange. Often, the broadest objectives for migrating to EDI are to improve productivity for the firm as a whole, while reducing the handling of paper documents.

Very real benefits can be experienced through the interchange of electronic data in the areas of [55]:

- reduced costs associated with business transactions,
- reduced order-cycle-pay period,
- improved trading partner relationships,
- improved intra-company flow of data, and
- improved forecasting.

Some of these benefits happen as a direct result of implementing EDI, while others are derived from making effective use of EDI data [56,57,58].

2.4.3.1. Reduced Costs Associated with Handling of Business Transactions.

The Electronic Data Interchange Council of Australia reports that 70% of computer output becomes input to another computer elsewhere [59]. The report also indicates 25% of the cost of a business transaction is in data entry and re-entry [60]. Each time data is entered into a system there is a chance of error, and the more often data is rewritten, the greater the chance of mistakes. One recent study estimated that more than US$70 billion is lost each year by companies in Europe because of inefficiency or errors in international trade documentation and paperwork procedures [61]. With EDI, key data entry and errors introduced by the receiver during that activity are eliminated. Also eliminated are manual sorting, matching, filing, reconciling, and mailing tasks.

2.4.3.2. Reduced Costs for Materials and Services to Support Paper Transactions.

Upon implementing EDI the costs for paper, envelopes, and mailing materials, as well as for telephone and courier services used to support transmission of orders and paper documents, will decrease. Additionally, storage space for paper and material supplies and, eventually, for filing of paper transactions will be freed.

2.4.3.3. Reduced Order-cycle-pay Period.

By eliminating use of regular mail and by decreasing the time needed to process one order transaction, products can be shipped and received sooner. Therefore, the buying company can delay the ordering of new product and can lower its level of inventory. This rapid and accurate transmission of data can lead to just-in-time inventory management, which involves shipping parts to factories and assembly facilities as they are needed.

The selling company not only ships sooner, but can invoice sooner as well. By receiving an electronic invoice in a timely manner, the buyer has the opportunity of taking advantage of discount terms offered, thereby paying less for product. The seller, in turn, can receive payments sooner, improving its cash position and allowing it to pay less for its supplies by also taking advantage of discount terms.

2.4.3.4. Improved Trading Partner Relationships.

In order to successfully transmit, interpret, and process transmissions automatically, trading partner cooperation is required, particularly as each party is counting on the other to provide accurate data or to process and act on the data received.

Also, by receiving accurate and complete data and by eliminating key entry errors on the receiving end, suppliers can be assured of making more accurate and timely shipments to their customers, as well as eliminating the need for charges associated with return shipments of incorrect product.

With this higher level of customer service, the supplier can look forward to more and larger orders from its regular customers and attracting a larger customer base.

2.4.3.5. Improved Intra-company Flow of Data.

With the generation and receipt of more accurate business transactions, the ability exists to collect information from all EDI transactions, and to develop and maintain a complete audit trail of that activity.

By disseminating that data to managers within the company you can provide the means to develop forecasts for future business opportunities, track vendor performance, evaluate cost savings, justify additional EDI activity, improve marketing strategies, and improve management control.

2.4.3.6. Improved Planning and Forecasting.

In an electronic environment, rapid receipt of accurate and complete business transactions is the norm. Orders can be processed quickly and shipments can be scheduled accurately. Companies can plan more accurately for receipt of goods as well as scheduling assembly line manufacturing tasks.

In the area of cash management, managers can plan cash flow more precisely by receiving and making payments sooner. As more business is converted to EDI, the cash position improves as assets are freed up from reduced stock and storage space requirements.

2.5. The Emergence of a New Technology: Electronic Structured Document Interchange.

Despite the advances in computing technology and office automation, and forecasts of the paperless office having become a reality by now, there remains the fact that very few companies would face less paper today than they did five years ago.

Through the use of storage technologies and other records and text management tools, much of the proliferation of paperwork in the office has been contained. Records and text management tools allowed paper documents to be classified, organised, circulated, retained, stored, and easily retrieved, reducing the necessity for multiple copies of a particular document to be in circulation.

Imaging technologies and image processing provided the greatest potential to shrink the paper mountain. Image processing was a similar concept to the records and text management tools associated with paper-based storage technologies. With image processing, however, paper-based documents were scanned through an optical scanner, which sent the digitised image to a computer. The computer controlled the storage and management of these electronic equivalents to the paper-based document. The computer, through the use of optical technologies, could provide an office worker with any document in the system in seconds, send it around individuals for consultation and review, print it, or store it in an electronic archival system.

As more people adapted to personal computers and businesses realised the potential in networking computers, new technologies developed making office workers more productive. However, companies continued to generate enormous amounts of paper documents.

Electronic mail eliminated some of the paper documents generated within an office. Office workers simply keyed in the message at a terminal, added the person's name and a heading, and the message was transmitted to the recipient. The recipient, on examining their electronic "in-tray," would see a list of the messages they had received, giving the name of the sender, the subject heading, and the date and time. At their leisure, the recipient could ask for any of the messages to be shown. After reading the message, they could reply to it, discard it in an electronic "wastepaper bin," file it for future reference, hold it in their "mailbox," or send it on to someone else with their own comments added. Electronic mail offered improvements in delivery speed, reliability, security, and communication, compared to standard mail delivery.

Electronic data interchange, which enabled firms to conduct business directly by computer, virtually eliminated the generation and transfer of paper documents between

companies. Using electronic data interchange a company generated standard business transaction data, transmitted it electronically to the receiving company, and the receiving company used the data as input to the receiving company's application, just as if it had been manually entered. Electronic data interchange provided the opportunity to improve productivity for the firm as a whole, while reducing the handling of paper documents.

Why then, are offices still deluged with paper?

In order for the paperless office to become a reality there must exist an electronic equivalent for every type of paper-based document within the office. Within a company paper-based documents can be categorised as:

1. documents having originated from external sources,
2. internal documents, and
3. documents destined externally.

Documents originating from sources external to the company are ideally handled by imaging technologies. The electronic equivalent of paper-based documents, produced by optical scanners, can be transmitted through networks, and can accommodate a mix of handwriting, word processing, and electronic mail. And since many systems use facsimile file formats for storing documents, they can accept facsimiles directly and send out a facsimile of any page on file.

Internal documents and documents destined externally can be further categorised as being of a structured type or of an unstructured type.

Unstructured documents, whether they are internal documents or documents destined externally, are typically sent through traditional mail systems. There are two reasons for this. Firstly, any mail system must provide for universal delivery, and secondly, must provide the capability to deliver material in a variety of formats. Electronic mail, which is suited to the unstructured type of mail, can present problems in these areas. Electronic mail systems, through the process of standardisation, especially in the area of interconnection of mail systems, will eventually reach the level of maturity such that it becomes a viable alternative to traditional mail systems.

Structured documents destined externally typically include standard business transactions such as purchase orders, purchase order acknowledgements, requests for quotations, quotes, invoices, bills of lading, and shipping notices. Such standard business transactions are increasingly being sent between trading partners through the use of electronic data interchange. Electronic data interchange, while reducing the amount of externally destined paper-based documents, also reduces the amount of paper-based documents that originate from external sources.

There remains one stumbling block to the achievement of the completely paperless office: the electronic equivalent to structured internal paper-based documents. The electronic equivalent to structured internal documents, together with the current developments in electronic office systems and the maturing developments in electronic delivery systems, would provide the complete infrastructure for the paperless office.

Structured internal paper-based documents are often transitory and proprietary in nature and hence are unsuitable to be replaced by any previously discussed electronic office system. These documents, being transitory in nature, often being passed from person to person during processing, are not suited to image processing, which is used solely for static documents. Neither are these documents suited to electronic mail systems because of their structured nature. Electronic mail systems allow for the delivery of free format text, where the layout of such text is inconsequential as no further processing is required. Electronic data interchange technologies would provide the closest electronic equivalent to structured internal paper-based documents. However, electronic data interchange is a well-defined business practice, and because the documents are for internal business use and are not sent between trading partners, and because of the often proprietary nature of the documents (providing no standard format), electronic data interchange cannot be used.

A new technology is required to provide an electronic equivalent to structured internal paper-based documents. A new technology, electronic structured document interchange (ESDI), has been proposed as the last remaining technology in providing the complete infrastructure for the paperless office.

2.5.1. Electronic Structured Document Interchange Features.

Electronic structured document interchange is the intra-company computer-to-computer processing of business transactions in a format that allows the receiver to process the transaction by traditional business practices.

Fundamentally, ESDI is a data processing concept that spans a single business enterprise, providing the complete electronic equivalent to the handling and processing of internal paper-based documents.

The following features characterise ESDI:

2.5.1.1. Intra-Company.

Intra-company specifically refers to the electronic transmission of electronic documents within a company. This implies that the company must have some underlying communications network and electronic delivery system.

2.5.1.2. Computer to Computer.

Not only are the electronic documents travelling from computer to computer, but they are travelling between electronic mailboxes, as a part of ESDI computer applications.

Today a paper-based transaction is either generated manually or through some computer application and mailed to the receiver. In the ESDI environment the sending ESDI application generates the business transaction, transmits it to the receiver, and the receiver uses the electronic document as input to the receiving ESDI computer application to carry out further processing. Unlike electronic data interchange, there is not a great deal of responsibility placed on the sending ESDI application to generate a complete and accurate business transaction. For, unlike electronic data interchange, the electronic document generated in an ESDI computer application is taken at face value by the receiver. If the receiver interprets some problem with the electronic document then, like its paper-based equivalent, it is returned to the originator. While, like electronic data interchange, these business transactions are transmitted electronically, they do not initiate a formal business transaction on which a business may be bound legally.

2.5.1.3. Business Transactions.

ESDI refers to the transmission of the electronic equivalents of internal paper-based business documents. Unlike electronic data interchange, however, it can also refer to the transmission of electronic mail or other free form messaging. In fact, the information referred to here is intended to permit all internal paper-based business transactions to be conducted electronically. Typical business transactions include central stores' orders, departmental budgets, etc.

2.5.2. How Does Electronic Structured Document Interchange Work?

In its most basic form the ESDI concept is simple.

As an example, imagine a department within a company placing an order to their central stores. These orders must go through the departmental manager who is ultimately responsible for the department's budget. The department raises a central stores' order as it always did. However, instead of using the central stores' order to develop the traditional paper transaction, an electronic equivalent on a computer monitor is used to develop the transaction. By entering information into predefined fields, i.e., item number, quantity, etc., the transaction is raised and when complete is transmitted electronically to the departmental manager's mailbox.

The departmental manager is notified that a business transaction is waiting to be processed and can view that document by selecting it from their mailbox. The departmental manager has the option of approving the order, in which case the transaction is transmitted instantaneously to central stores where it is processed, or can return it to the originator if there is a problem, i.e., the budget has been exceeded.

In a more complex example the raising of the central stores' order could pose other possibilities. The central stores' order may access the central stores' database for item number descriptions, up-to-date prices, and stock availability. The raising of a central stores' order may also generate multiple copies, each of which must be administered. For example, the department may require a copy, the head office may require a copy, and a copy may be sent to central stores.

2.5.3. Benefits of Electronic Structured Document Interchange.

No matter how complex the example, the possible benefits that can be experienced through the use of ESDI include:

- reduced costs,
- improved speed of delivery,
- improved reliability of delivery,
- improved security,
- improved internal communications, and

- improved intra-company flow of data.

Indeed, an ESDI system has the potential to improve services by providing capabilities that are simply not possible with traditional paper-based business systems.

CHAPTER

3

The Massey University Electronic Structured Document Interchange Initiative

Realising the potential benefits that could be gained from electronic structured document interchange (ESDI), Massey University, through initiatives from Management Information Services, proposed an ESDI implementation to facilitate the administrative functions of the University. The aim of the project was to set up at least one, and possibly more, fully functional systems utilising ESDI.

From an administrative point of view, the University is a relatively complex system. Massey University is New Zealand's second largest tertiary institution, with approximately 25,000 internal and extramural students, and comprises eight faculties, as well as the schools of aviation and information sciences. As Chief Executive Officer, the Vice-Chancellor is accountable to the University Council, as governing body, and to the Academic Board, as the board responsible for academic affairs, for the implementation of policy and the general management of the University. He is supported in these duties by the University Registrar, as the Chief Administration Officer, who is responsible for the functions and activities of the Registry, which comprises the central administrative structure of the University. Accountable to the Registrar are Deputy Registrars and Directors that head various sections and offices in the Registry, while Heads of Department, Research Centre Directors, Unit supervisors, and the University Librarian are each accountable for their respective functions and activities. The structure of the University is shown in Figure 6 [62]. Most of the administrative functions of the University are conducted in the Registry, although other administrative staff, such as Faculty Administrators, Heads of Department, Research Centre Directors, etc., are distributed about the Campus and possess quite a high level of autonomy in many areas.

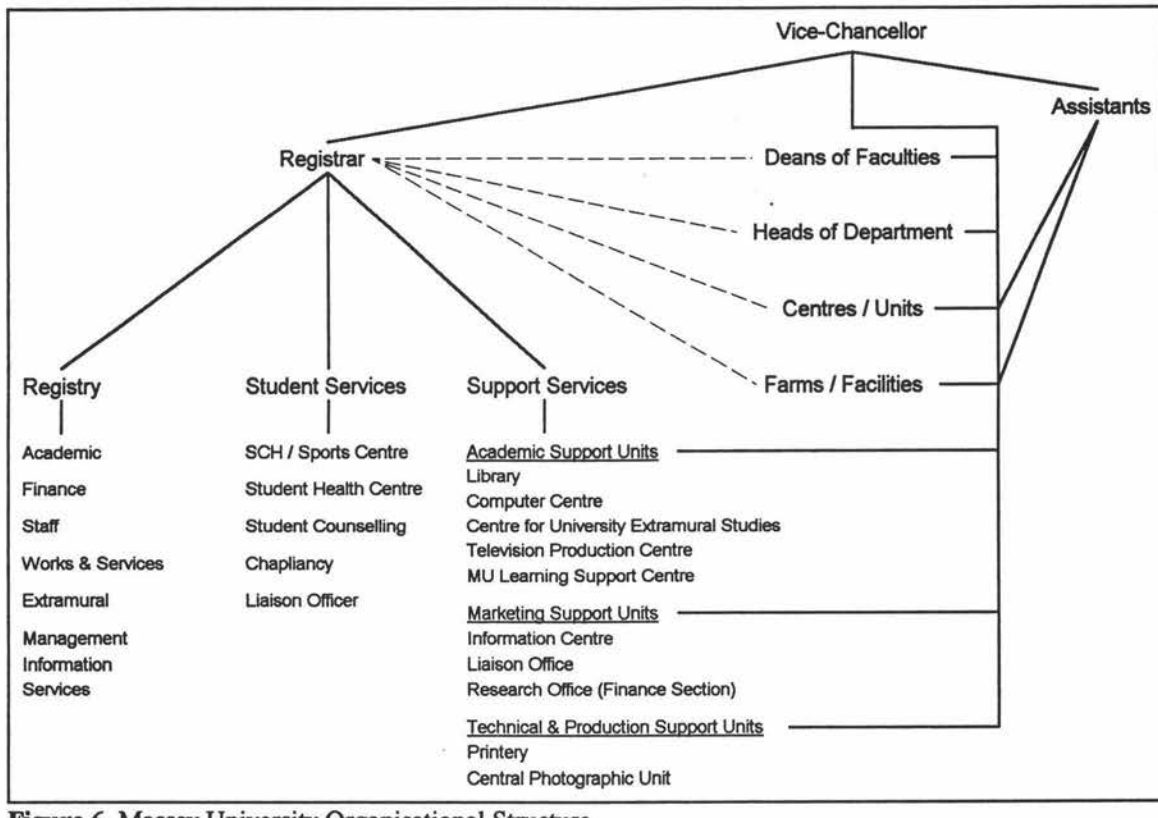


Figure 6. Massey University Organisational Structure.

The complexities of the University's administrative structure result in the generation of an enormous amount of paperwork. So much paper is circulated around the Campus that a recent study estimated that the University generated a staggering six tonnes of waste paper each week [63]. When a conservative estimate suggests that the amount of non-waste paper circulating around the Campus is at least double that, the benefits that could be realised from an ESDI system are obvious.

Before looking at the benefits that could be obtained through ESDI initiatives, it is necessary first to look at the infrastructure of Massey University Computing Services.

3.1. Massey University Computing Services.

The Computing Services section is responsible for providing the Academic Support Services and Administrative Support Services essential to the efficient running of the University. Management Information Services (MIS), through the Director of MIS, is accountable to the Registrar for the administrative computing services necessary for the functioning of the Registry. While, academically, the Director of Computing Services is accountable to the Vice-Chancellor for providing the computing services required by academic staff and students through the Computer Centre.

The Computing Centre is a central service of the University set up to meet general academic computing needs in research and testing. To meet these needs the Centre offers a range of facilities and services to staff, postgraduate students, and undergraduate courses using computers. The Centre attempts to provide general computing facilities for numbers of users rather than specific facilities for individuals or departments.

3.1.1. The Campus Network.

The Computer Centre, while providing a range of support, advisory, and computational services to over sixteen hundred users, administers and maintains a Campus-wide high-speed data network that provides distributed computing facilities, as well as internal and external communications, for personal computers and other workstations.

The distributed computing facility is based on personal computer workstations connected to various host computing services via the high-speed Campus Network. Host services are available 24 hours per day, 7 days per week.

The Campus Network features a number of bus networks that are interconnected, with the resulting topology resembling that of a tree structure. The root of the tree structure originates in Computing Services with the CISCO bridge, which links the university with Victoria University in Wellington (and other institutions throughout the world) through both a PSDN (packet-switched data network) and the PSTN (public-switched telephone network). The CISCO then branches out to three separately administered networks. The Massey University Registry Network and the Massey University Campus Network are connected through respective bridges. The third network, the Crown Research Institute Network at the Fitzherbert Science Centre in Palmerston North, is fibre-optically linked to the CISCO.

Through a multiport repeater in Computing Services, the Campus Network branches out into a number of bus networks to encompass the Department of Maori Studies and the Faculties of Business Studies (Business Studies Central and Business Studies West), Technology (Food Technology and Production Technology), and Science. Another multiport repeater in the Science Faculty serves as a hub for the bus networks to the former Computer Centre (Bernard Chambers), the Faculties of Veterinary Science and Agricultural and Horticultural Sciences, the Department of Botany and Zoology, and the University Printery. It also provides a thick ethernet bus network that links the Cafeteria, Library, the Departments of Social Sciences and Psychology, and the School of Mathematical and Information Sciences. The physical layout of the Campus Network is shown in Figure 7.

The Computer Centre offers external access to institutions world-wide providing electronic mail, news, and other services. The electronic mail system is provided by a dedicated host computer, a DEC 3100 (cc-server4) running Ultrix, which also provides the electronic mailboxes for the registered users.

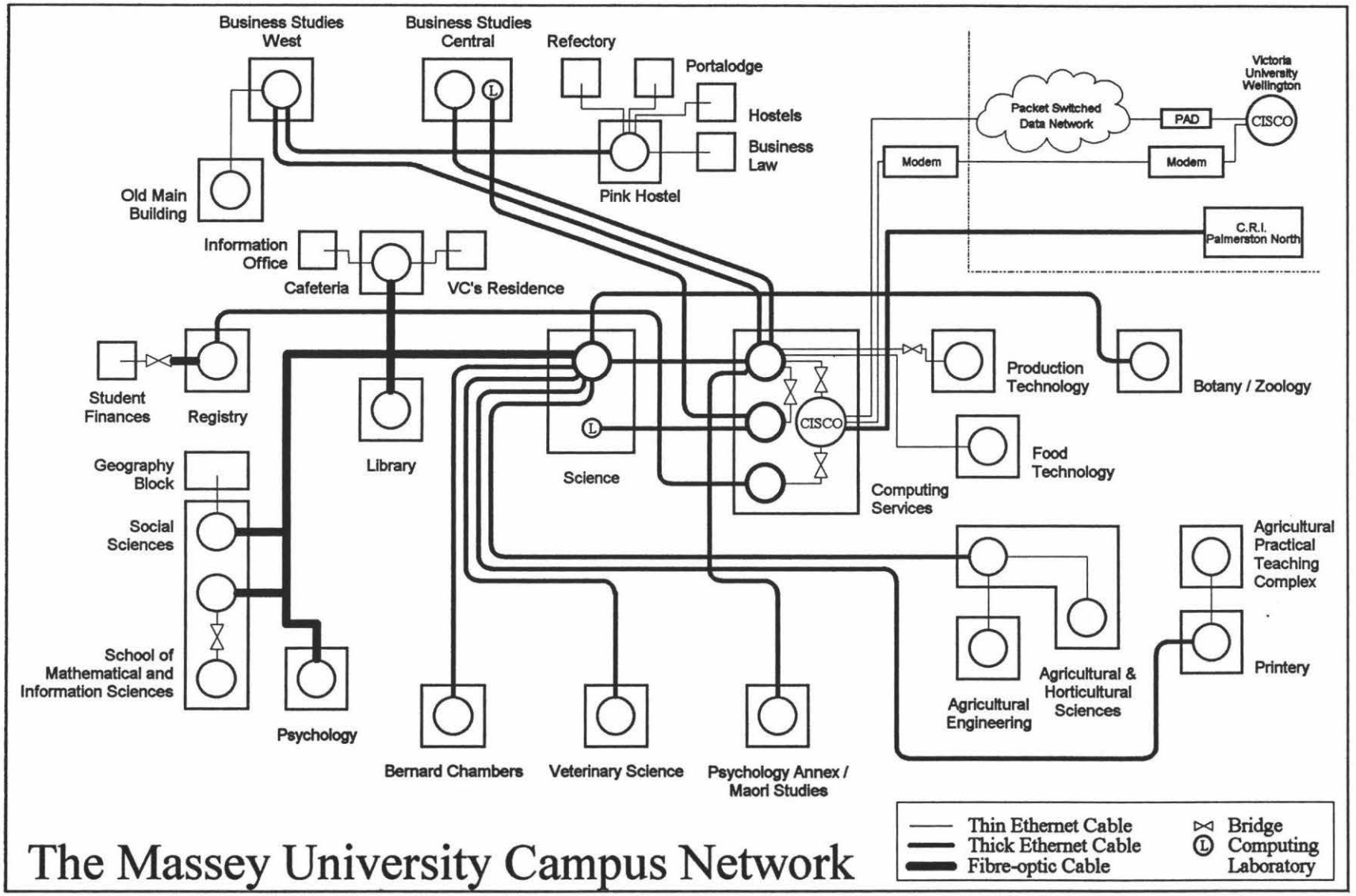


Figure 7. The Massey University Campus Network [reproduced with permission of Massey University Computing Services].

3.2. Initiative Aims.

The aim of the project was to set up at least one, and possibly more, fully functional systems utilising ESDI. The fundamental objective being to pass the electronic equivalents of paper forms around by means of some electronic delivery system, running on some underlying communications network. The use of the Campus Network, which provides a comprehensive coverage of the Campus, and the Computer Centre's electronic mail system (NETMAIL), which provides the necessary store-and-forward delivery system, provided the obvious mechanisms for the passing and delivery of the electronic equivalents of paper forms.

Making use of the Computer Centre's electronic mail system it was established that the initial aims of the project would be to:

1. develop and document a system to enable a lecturer to request a class roll from the main Registry computer,
2. develop and document a system to enable a department to place a central stores order, and
3. develop and document a system to enable a department to request account status or transaction reports.

There were many rationales behind the implementation of an administrative ESDI system, perhaps the greatest being the realisation of the potential cost savings. Not only could cost savings be achieved because of the elimination of paper, printing, and postage costs, but also because of the reduced labour costs brought about through reduced paper handling and data entry. Other rationales, including the improved speed of delivery and the improved reliability of delivery, were great incentives.

CHAPTER

4

The On-Campus Electronic Structured Document Interchange Development

Having set the aim of the initiative, to create at least one, and possibly more, fully functional systems utilising electronic structured document interchange (ESDI), a development strategy for the on-campus ESDI initiative was established.

The idea behind ESDI was to provide the complete electronic equivalent to the handling and processing of internal paper-based documents. The rationale being to take advantage of the benefits of electronic processing and delivery, while retaining traditional business practices. In some respects, an ESDI system has the potential to improve business practises by providing capabilities that are simply not possible with the traditional paper-based business systems.

While it is easy to envisage the ideal paperless environment, an environment that initiates, processes, and archives electronic transactions as an integral part of traditional business practice, it is not as easily achieved.

It was decided that a systematic approach was required for the development of the on-campus ESDI system. The idea was to implement the system in stages, with each stage increasing in functionality. The purpose of such a strategy would allow the system requirements and system development (design and construction) of the envisaged final stage to be adequately planned for, throughout successive stages, from early on in the development.

The major factor in the success of such a system is user acceptance. A staged development is well suited to encouraging user acceptance: each stage would be implemented only after extensive testing, would provide a consistent user interface, and would be seen to be offering tangible benefits to users over previous stages. It was also important that each stage be implemented only after a set period of time, a time in which the users would have had the chance to become relaxed and familiar with the previous implementation.

Three stages were defined for the on-campus ESDI development. These were:

Stage 1:	Single Form / Single Destination (SFSD)
Stage 2:	Multiple Forms / Multiple Destinations (MFMD)
Stage 3:	Accretive Forms.

The increasing functionality in each of the successive stages in the development is discussed next.

4.1. The Staged Development.

The fundamental objective of the project was to pass the electronic equivalents of paper-based documents around by means of some electronic delivery system, running on some underlying communications network. It had already been established that the Campus Network, which provides a comprehensive coverage of the Campus, and the Computer Centre's electronic mail system (NETMAIL), which provides the necessary store-and-forward delivery system, would be used for the passing and delivery of the electronic equivalents of paper-based documents. What had yet to be established was the nature of these electronic equivalents and how they would be processed.

The electronic equivalents of internal paper-based documents are a fundamental part of an ESDI system, and their simplicity and ease of use are critical to its success. These "electronic forms" must provide an effective and efficient replacement of paper-based documents in order to retain traditional business practices and to be preferred to the use of paper-based systems. To be effective the electronic forms need to be easily constructed and should provide all the content of its paper-based counterpart. While, to be efficient, the electronic forms need to be easily edited and manipulated, providing all the functions common to the processing of paper-based transactions.

A mechanism was available within the Department of Production Technology to provide the desired electronic equivalent to paper-based documents. The mechanism was simple, in that an electronic representation of a paper-based document could easily be constructed, and powerful, in that the mechanism could effectively provide the functions common to the processing of paper-based transactions. This "forms engine" was the fundamental component of the ensuing on-campus ESDI development, and is discussed more fully in Chapter 5, "Paper's Counterpart: The Electronic Form."

The intention of the first stage of the on-campus ESDI development, referred to as Single Form / Single Destination (SFSD), was to implement a particular electronic form whose destination was predetermined. The ESDI system, for this first stage, was very simplistic. The ESDI system would display a single rudimentary electronic form, the user would fill in the form details, and when transmitted, the form details would be delivered to the predetermined destination.

Cognisant of the aims of the initiative, the Class Roll Request Form was implemented. The purpose of the Class Roll Request Form was to initiate the extraction of class roll information residing on a database that was not a part of the Campus Network. Using the ESDI system a lecturer or faculty administrator filled in the Class Roll Request Form details, and when transmitted, the form details were delivered to Management

Information Services. Details of the SFSD implementation are given in Chapter 6, "A SFSD Example: The Class Roll Request Form."

Since ESDI replaces the paper document environment with an electronic one, management and auditors face the challenge of how to implement ESDI technology to attain business objectives and appropriately control its associated risks. Many of the controls used in paper document processing are simply not effective in the ESDI environment. Chapter 7, "Security Aspects of Electronic Structured Document Interchange," establishes the business risks associated with ESDI, and examines the associated control and security aspects.

As the on-campus ESDI implementation evolves, the need to reassess the controls over the existing environment and applications becomes more apparent, and the additional risks, that may derive from the implementation of ESDI because of the changes to systems, procedures, and operations, should be addressed.

The aim of the second stage of the on-campus ESDI development, referred to as Multiple Forms / Multiple Destinations (MFMD), was to increase the functionality of the ESDI system implemented in stage one by making a facility available that would offer a choice of electronic forms and a choice of destinations. Each electronic form would provide the equivalent of an internal paper-based document and could have a predetermined destination.

The ESDI system would function as follows: if the forms do not have a predetermined destination (or if there is more than one destination provided), a menu of destinations is displayed and the user selects the appropriate destination; a menu of available forms is displayed and the user selects the appropriate form; the form is displayed, and the user fills in the form details; finally, when transmitted, the form details are delivered to the proper destination.

The details of the MFMD implementation, and the Financial Transaction Request Form, are presented in Chapter 8, "A MFMD Example: The Financial Transaction Request Form." The purpose of the Financial Transaction Request Form was to enable a department to request the account status and transaction reports, for a nominated account number, over a specified period. Again, this information resided on a database that was not a part of the Campus Network.

While the electronic forms and the forms engine provided the capability of paper-based documents, the forms mechanism lacked some of the functions common to the processing of paper-based transactions. The third, and final, stage of the on-campus ESDI development, referred to as Accretive Forms, would implement the concept of transaction streams.

A transaction stream is effectively a predetermined path that a form must follow as a part of its inception, i.e., the stages of processing that the form must go through before it is regarded as a complete business transaction. Occurring as an integral part of the form itself, each instance of the form would have a transaction stream associated with that form. This allows the system considerable flexibility. The same form can have two instances, each with unique transaction streams, and so can be used for two totally different purposes.

Transaction streams are a concept that allows businesses to utilise ESDI, while helping to retain as much of their traditional business practises as possible. For paper-based transactions that must travel between various people and functions to gather specific information as a part of its inception, transaction streams and electronic forms are ideally suited. Transaction streams allow the forms mechanism to automate the process of delivering a form along its predetermined path, passing it on to the next stage in the path after the form has been processed sufficiently. Once at the end of the transaction stream, when complete, the form details can be delivered to its proper destination. The concept of transaction streams could also be taken a step further, so that a form at any predefined stage in the transaction stream may initiate one or more occurrences of another form, a form that follows a separate predefined transaction stream.

Through this final stage, ESDI would have the potential to provide the complete electronic equivalent to the handling and processing of internal paper-based documents, thus enabling businesses to take full advantage of the benefits of electronic processing and delivery, while retaining their traditional business practices.

CHAPTER

5

Paper's Counterpart: The Electronic Form

Any electronic equivalent of internal paper-based documents must provide an effective and efficient replacement in order to retain traditional business practices, and to be preferred to the use of paper-based systems.

This chapter describes a mechanism, available within the Department of Production Technology, that provides the desired electronic equivalent to paper-based documents. This mechanism is simple, yet powerful: simple, in that an electronic representation of a paper-based document can be easily constructed (while providing all the content of its paper-based counterpart); and powerful, in that the mechanism can effectively provide the functions common to the processing of paper-based transactions.

The mechanism, referred to as the "forms engine," has been designed to be called from an application program. An application program (i.e. an ESDI application) firstly describes the processing that is required of the forms engine on some "electronic form," then calls the forms engine, which processes the form as specified by the application. The application must provide the forms engine with the operation that is to be performed on the form and, based on this operation, details that include the nature of the form to be processed (i.e. where it is located and how it is stored), the nature of the output of the processing (i.e. where should the output be placed and how should it be stored), where the form is to be displayed, and how the form is to be edited.

The forms that are processed by the forms engine are described through Form Definition Files and/or Form Data Files, either of which may reside in memory. These files are ASCII text files. The Form Definition File contains the form header, the information on how the form is to be displayed and printed, display lines, which define the on-screen and printed image of the form, and field definitions, which define a field's type and attributes, and possibly, a default value for the field. An instance of a Form Data File may exist for any Form Definition File. A Form Data File, or a Form Definition File containing default

values for one or more fields, provides an instance of an electronic form. The Form Data File, unlike the Form Definition File, contains only the values for each field in the Form Definition File, as well as a reference to the Form Definition File within the form header.

Form Definition Files, Form Data Files, and the forms engine are described in more detail in the following sections, using the replacement of the paper-based invoice as an example.

5.1. The Form Definition File.

The electronic representations of internal paper-based documents, or electronic forms, can be simply and easily constructed through the Form Definition File, using nothing more than an ASCII text editor. The Form Definition File contains the form header, the information on how the form is to be displayed and printed, display lines, which define the on-screen and printed image of the form, and field definitions, which define a field's type and attributes, and possibly, a default value for the field.

Information contained within a Form Definition File is stored as ASCII text. The Form Definition File can have up to 400 lines, with each line containing up to 128 characters. The form itself can have up to 256 displayable lines and 800 fields, and may include any of the IBM graphics characters.

An example of a Form Definition File is shown in Figure 8. The Form Definition File describes an electronic representation of the paper-based invoice.

5.1.1. Line Classes.

A Form Definition File consists of a number of line classes. These designated line classes, that identify the form header, display lines, field definitions, etc., can be recognised by the first character in the line. The line recognition characters are:

"h"	Header Lines
"*"	Display Lines
"="	Field Definition Lines
":"	Label Lines

Any other characters, apart from "x", ">", and "<", which are reserved for future use, are treated as a line recognition character for a comment line, and the line will be ignored.

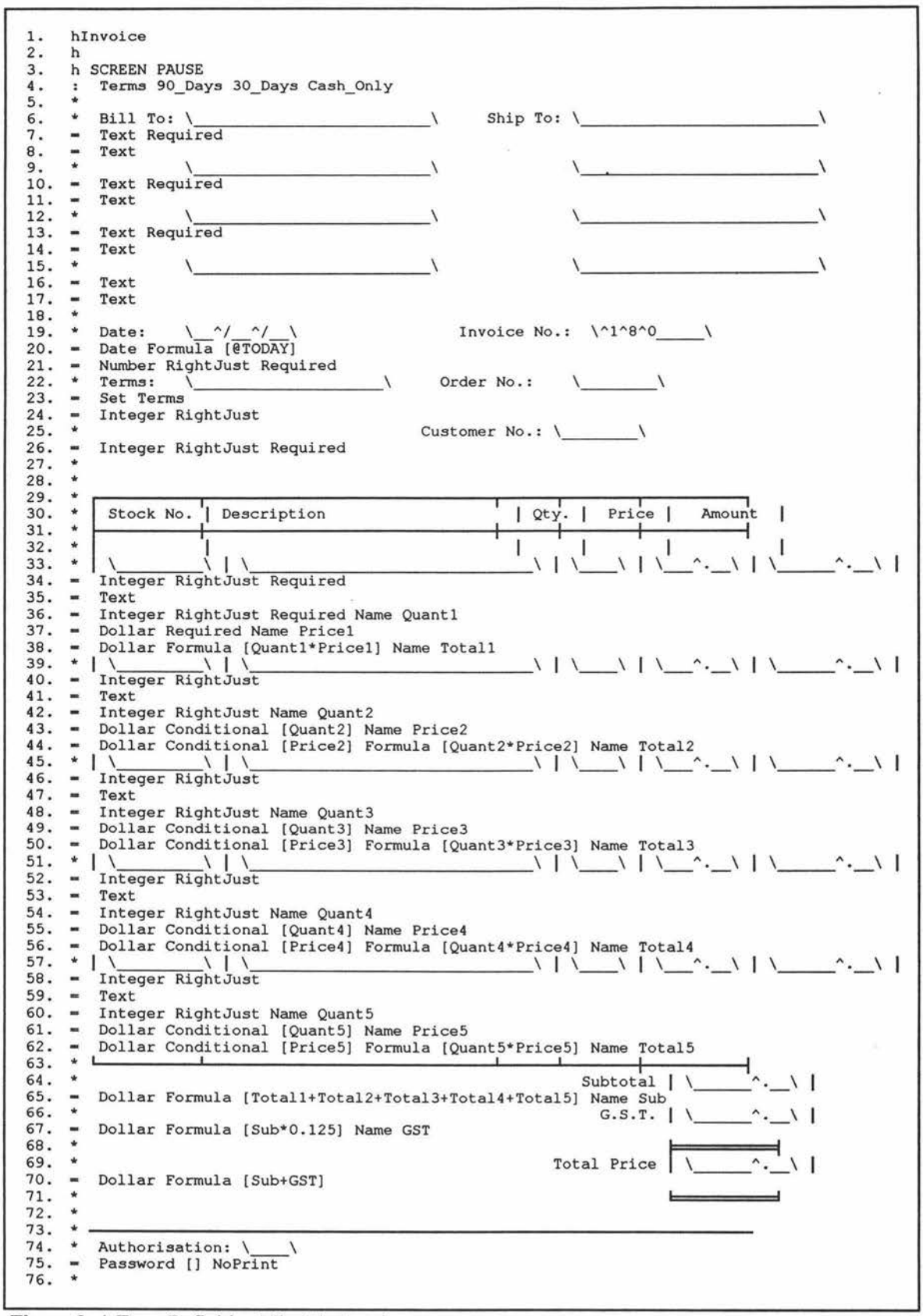


Figure 8. A Form Definition File: The Invoice.

5.1.1.1. Header Lines.

The header lines distinguish an ASCII text file as being a Form Definition File or a Form Data File. There must be three header lines within a Form Definition File and these will usually be the first three lines in the file, although they may be preceded by comment lines. Lines 1-3, of Figure 8, show the header lines for the Invoice Form Definition File.

Each of the three header lines has a specific purpose. The first header line contains the form title that will be displayed at the top of the form window. The title begins with the character immediately after the line recognition character or the first non-blank character, i.e., the form title for the Invoice Form Definition File is the word "Invoice". The second header line is reserved for Form Data Files, in which case it will contain the keyword FDATA, followed by a square bracketed file name, the file name being the Form Definition File associated with the Form Data File. Form Data Files are discussed more fully in Section 5.2. The third header line contains parameters concerning the printing of the form, examples of which are given in Figure 8, line 3. These parameters are described in Section 5.1.3., Global Parameters.

5.1.1.2. Display Lines.

The display lines are the only lines of the Form Definition File that the user sees. The form outlines, form layout, field headers, fields, and any other displayable information will be in these lines, defining the on-screen image of the form, as well as the printed image. The display image starts with the character immediately following the line recognition character. The entire line will be displayed except for the following characters:

1. the line recognition character,
2. field delimiter characters, "\", and
3. mask character precursors, "^".

Field delimiter characters and mask character precursors are discussed further in Section 5.1.2., Fields.

5.1.1.3. Field Definition Lines.

A field's type, its attributes, and any parameters required by these types or attributes, are found in the field definition lines. A field definition line is associated with a field by order of occurrence. It is required that a field definition line be after (but not necessarily immediately after) the display line in which a field occurs. For the visual association of the field and its definition, it is usual to follow a display line containing fields with a field definition line for each field in the display line. There must be a field definition line for every field in the form.

5.1.1.4. Label Lines.

Label lines are used to contain the set of strings that are displayed on the screen when a SET field is entered. These lines start directly after the colon with a unique name, the name being the label for the set. Section 5.1.2.2., Field Attributes, describes the SET field in more detail.

5.1.2. Fields.

An entry field in a form consists of two parts: the field display image, as represented in a display line; and the field type and attributes, as defined in its field definition line.

A field is indicated within a display line by leading and trailing backslashes, "\", called field delimiters. These delimiters separate a field from the displayable characters of the display line. They therefore determine a field's length and, through their position on the display line, the field's displayed position within a form. Within a display line, the field delimiter characters and mask character precursors will not be displayed or printed. They must therefore be taken into account when determining a field's length and displayed position.

Within the field delimiters, two characters have a special meaning: the underscore character, "_", and the mask character precursor, "^". Between field delimiters, each position within a field is represented by an underscore character. These characters signify that a field has not been filled, and, as a result, the field will be assigned the value NIL.

Mask character precursors, however, are used to improve a field's appearance on the screen and printed form. They may also be used to delimit subfields, or to define literals within a field. Mask character precursors have been used throughout the Form Definition File, shown in Figure 8. For example, line 19 shows mask character precursors being used to improve the appearance of a DATE field, as well as being used to define literals for the invoice number.

A field's type and its attributes are defined in field definition lines, one line for each field. A field can only have one type, however, it may have many attributes. The field type and attributes are assigned through the use of case-insensitive keywords, separated by spaces, which may be shortened to their first three characters.

Field types and attributes are described in the following sections.

5.1.2.1. Field Types.

Field types are described below. Where appropriate, the syntax for the field types has been described using Backus-Naur Form.

TEXT

A TEXT field is a general purpose field, accepting any sequence of printable character except the "-" character.

In formula or conditional calculations the field is always evaluated to zero or NIL. Formula and conditional calculations are discussed in Section 5.1.2.2., Field Attributes.

Examples of TEXT fields are shown in Figure 8, lines 7, 8, and 35.

NUMBER

A NUMBER field is a general number field, as described in Figure 9. The numbers entered can be of type integer, fixed point, or floating point, however, an exponential format cannot be used. For fixed point numbers, the position of the mask character "." fixes the definition of the decimal point.

Mask characters, such as may be used in a telephone number field, but which excludes the decimal point, will cause the field to evaluate to zero or NIL in formula or conditional calculations.

An example of a NUMBER field is shown in Figure 8, line 21.

```
<text> ::= <any sequence of printable symbols not containing "-">

<number>  ::= <unsigned number> | <sign> <unsigned number>
<integer> ::= <unsigned integer> | <sign> <unsigned integer>
<real>    ::= <unsigned number> | <sign> <unsigned number>
<dollar>  ::= <unsigned number> | <sign> <unsigned number>

<unsigned number>  ::= <unsigned integer> | <unsigned real>
<unsigned integer> ::= <digit> | {<digit>}
<unsigned real>    ::= <unsigned integer>.<digit> | {<digit>}
<sign>  ::= +|-
<digit> ::= 0|1|2|3|4|5|6|7|8|9

<date> ::= <date> <month> <year>

<date>  ::= <digit> {<digit>}
<month> ::= <digit> {<digit>}
<year>  ::= <digit> {<digit>}
```

Figure 9. Field types expressed in Backus-Naur Form.

INTEGER

An INTEGER field, unlike the NUMBER field, has values that are a subset of the whole numbers. As shown in Figure 9, an INTEGER field will not accept a decimal point, ".".

An INTEGER field returns a long (32 bit) value and should not contain mask characters.

Examples of INTEGER fields are shown in Figure 8, lines 24, 26, 34, and 36.

REAL

A REAL field, as described in Figure 9, returns single precision, floating point numbers. It can therefore be used for numbers too large for a field of type INTEGER.

DOLLAR

The DOLLAR field is a decimal currency field, and is described in Figure 9.

When formatting a DOLLAR field, the cents subfield (two character positions) should be specified with a mask character, for example, "___^.__", while the currency symbol should, for purposes of clarity, immediately precede the field delimiter and should not be a part of the field.

Examples of DOLLAR fields are shown in Figure 8, lines 37, 38, 65, 67, and 70.

DATE

The DATE field is a numeric date field that follows the European format (days before months), that is dd mm yy with specifiable separating characters. These separating characters, which must be present in order for the field to be automatically validated, are positioned within the field as mask characters, for example "__^/__^/__\".

An example of a DATE field is shown in Figure 8, line 20.

PAUSE

The PAUSE field is used to arrest the cursor when moving to the next field. It allows the user to observe the form when fields with automatic attributes (i.e., FORMULA, INVISIBLE, etc.) would cause the form to scroll in the window.

The PAUSE field only accepts cursor movement keys, the position at which the cursor rests being determined by two consecutive field delimiter characters in the desired position in the display line.

```
PASSWORD []
PASSWORD [filename]
```

The PASSWORD field is a special text field for the entry of passwords, where the entered characters appear on the screen as an IBM graphics character. A parameter must follow the keyword, optionally specifying a file containing encoded passwords. If it is desired to have the passwords validated immediately, then the parameter field must specify a file containing encoded passwords (e.g. PASSWORD [ENCRYPT.PWD]), otherwise, an empty parameter field (e.g. PASSWORD []) indicates that no validation is needed and any entered string will be accepted as a valid password.

The following should be noted about the PASSWORD field. Firstly, all fields following a password field are temporarily given the attribute INVISIBLE, therefore, these fields will not be displayed or entered until the password is correctly given, and secondly, the NOPRINT field attribute can be used to stop the transmission of the encoded password with the form data.

An example of a PASSWORD field is shown in Figure 8, line 75.

5.1.2.2. Field Attributes.

REQUIRED

A field with a REQUIRED attribute must be filled before the form can be accepted as having been sufficiently processed. If a required field has not been filled, the user will be warned and the cursor will be returned to the first unfilled REQUIRED field.

Examples of fields with the REQUIRED attribute are shown in Figure 8, lines 7, 10, and 13.

DISPLAY

A field with a DISPLAY attribute will be displayed, however it cannot be modified by the user. The cursor will enter a field with a DISPLAY attribute, but only cursor movement keys will be accepted within the field.

LEFT
RIGHT

The LEFT and RIGHT attributes specify justification within a field, the omission of both attributes leaving a field as entered. It should be noted that a DATE field, with a "/" mask character, and DOLLAR and NUMBER fields, with a "." mask character, are automatically justified.

Examples of fields with justification attributes are shown in Figure 8, lines 34, and 36.

NOPRINT

The NOPRINT attribute will allow a field to be filled, however the entered string will not be written to the output file.

An example of a field with the NOPRINT attribute is shown in Figure 8, line 75.

UPCASE

The UPCASE attribute causes characters entered into a field to be converted to upper case.

NAME *field_name*

The NAME attribute assigns a name to a field, allowing the field to be referenced within formula and conditional expressions. The field name is specified as a parameter consisting of no more than 7 characters, the only stipulation being that it must start with an alphabetic character, for example, NAME COST, NAME D1.

Formula and conditional expressions are discussed below.

Examples of fields with the NAME attribute are shown in Figure 8, lines 36, 37, 38, 65, and 67.

SET *label_name*

The SET attribute assigns a short static pick list to a field. The label name that follows the SET keyword must reference a label line that contains a set of strings, separated by spaces, which constitute the allowable entries for the field. The user picks an entry from the displayed list, the selection becoming the field entry.

An example of a field with the SET attribute is shown in Figure 8, line 23. The label line that has been referenced is shown in line 4.

CONDITIONAL [*conditional_expression*]

The CONDITIONAL attribute makes use of a string parameter to determine whether or not the field will be entered. Before entering the field the parameter string is parsed, as an expression, and evaluated. The field will be skipped if the returned value of the expression is less than or equal to zero, or NIL.

The CONDITIONAL attribute allows fields to be passed over if previous fields have not been filled or have not been filled with a specified value.

Examples of fields with CONDITIONAL attributes are shown in Figure 8, lines 43, 44, 49, and 50. Conditional expressions are discussed below.

FORMULA [*formula_expression*]

The FORMULA attribute makes use of a string parameter to determine the value of a field. The parameter string is parsed, as an expression, and evaluated. The resulting field value will be the result of the formula expression, formatted to suit the field type.

The FORMULA attribute allows field values to be calculated automatically. This is particularly useful for fields such as G.S.T., totals, etc.

Examples of fields with FORMULA attributes are shown in Figure 8, lines 38, 44, 65, 67, and 70. Formula expressions are discussed below.

Formula and Conditional Expressions.

A formula or conditional expression is an ASCII string bracketed by "[" and "]". The string, when parsed and evaluated, returns a REAL value, the value being used to determine whether or not the field will be entered (CONDITIONAL attribute) or to determine the value of the field (FORMULA attribute). The field will be displayed in the format that is determined by the field type.

The formula and conditional expressions can contain:

Numbers,
References to other fields (by name),
Operators, or
Functions.

The syntax for formula or conditional expressions is described in Figure 10.

```
<formula>     ::= [<expression>]
<conditional> ::= [<expression>]

<expression> ::= <term> | <term> <adding operator> <term>
<term>       ::= <factor> | <factor> <multiplying operator> <factor>
<factor>     ::= <unsigned factor> | <unary operator> <unsigned factor>
<unsigned factor> ::= <number> | <field reference> | <function> |
                      (<expression>)

<adding operator>      ::= +|-
<unary operator>       ::= <adding operator> | nil
<multiplying operator> ::= *|/|%
<field reference>      ::= <name string> | ?<name string>
<function>             ::= @<name string>

<name string> ::= <any sequence of symbols, of no more than 7 characters
                   starting with an alphabetic character>
<number>      ::= <unsigned number> | <unary operator> <unsigned number>

<unsigned number>  ::= <unsigned integer> | <unsigned real>
<unsigned integer> ::= <digit> {<digit>}
<unsigned real>    ::= <unsigned integer>.<digit> {<digit>}
<digit>            ::= 0|1|2|3|4|5|6|7|8|9
```

Figure 10. Backus-Naur Form for formula and conditional expressions.

NUMBERS.

The numbers may be of type integer, fixed point, or floating point, however, an exponential format cannot be used.

FIELD REFERENCES.

Fields are referenced by name, therefore it is necessary to assign a name to any fields used in the formula or conditional expression. Names may be assigned to a field through the NAME attribute. A field name consists of no more than seven characters, the only stipulation being that it must start with an alphabetic character.

A field's value will be calculated at the time that the expression is evaluated. Fields not yet filled in will return zero (NIL), while TEXT and PAUSE fields always return zero (NIL), and NUMBER, INTEGER, REAL, and DOLLAR fields return their respective values as a REAL type. As TEXT fields always return zero (NIL), the use of "?" before a field reference (i.e. ?<name string>) can be used to determine whether a field has been filled or not, only returning zero (NIL) if the field has not been filled.

OPERATORS.

The usual mathematical operators are provided, addition "+", subtraction "-", multiplication "*", and division "/", as well as the truncated modulus operator, "%". The truncated modulus operation performs:

A % B := truncate(A) MOD truncate(B)

The division operator is protected from divide by zero errors, and the result returned, if the divisor is zero, is zero (NIL).

FUNCTIONS.

A number of functions can be used within formula and conditional expressions. Functions are differentiated from field references by having their names immediately preceded by an "@" character, and may require one or more arguments.

Arguments should be enclosed in parentheses, "(" and ")", with each argument separated by commas. Note, that if an argument is missing the function will return zero (NIL) (except for the CASE function), that if there are too many arguments the unused

arguments will be ignored, and that there must be no spaces between or within the arguments.

Recursion within functions is allowed. Therefore arguments of a function may be expressions.

The following functions have been implemented.

`ABS(argument)`

Returns the absolute value of the argument.

`AVG(argument1,argument2,argument3,...)`

Returns the average of any number of argument values.

`CASE(selector,case0,case1,case2,...)`

The truncated value of the selector argument is used to select one of the remaining arguments, which will be evaluated and returned. If the value of the selector argument should be less than zero, or greater than the number of arguments, the first or last argument will be selected respectively.

`COUNT(argument1,argument2,argument3,...)`

Returns the position, decremented by one (so that the first position is zero), of the first argument with a value of zero.

`IF(expression1,true_expression,false_expression)`
`IF(expr1 boolean_op expr2,true_expression,false_expression)`

The value of "true_expression" is returned if the value of "expression1" is greater than zero or if the value of the operation "expr1 boolean_op expr2" is true, that is greater than zero, otherwise, "false_expression" is evaluated.

If either the required true or false expression is missing, the value zero (NIL) will be returned.

INT(*argument*)

Returns the integer part of the argument's value.

MIN(*argument1*,*argument2*,*argument3*,...)

Returns the minimum value of any number of argument values.

MAX(*argument1*,*argument2*,*argument3*,...)

Returns the maximum value of any number of argument values.

TODAY

Returns today's date, which can be used in DATE fields.

5.1.3. Global Parameters.

Global parameters set attributes that are global to the form, concerning the way the form will be printed. They are placed in the third header line of the Form Definition File.

The following global parameters have been defined.

SCREEN

The SCREEN parameter indicates that the printed form will be an image of the form displayed on the screen.

DEFAULT

The DEFAULT parameter is designed to be used with pre-printed forms, that is, it specifies that only the field data is to be printed as all other necessary information is expected to be on the pre-printed form.

The field's position on the pre-printed form is specified by a pair of integers (row, column) following the field definition line recognition character. For example, "= 21 56

TEXT" specifies that this TEXT field is on line 21 and starts at column 56 of the pre-printed form. Although the order and layout of the fields on the screen and pre-printed form may be different, the fields will be ordered by line and column ready to be printed.

NOFF

The NOFF parameter is a directive not to insert form feed characters within the printed form.

TOP *lines*

The TOP parameter specifies the number of lines from the top of the page at which printing is to begin.

FORMLEN *lines*

The FORMLEN parameter specifies the number of lines to be printed on the page. This includes TOP.

LEFT *spaces*

The LEFT parameter specifies the number of blank spaces to be inserted to the left of the page before each line is printed.

PAUSE

The PAUSE parameter specifies that printing is to pause at the end of each page for the next sheet of paper to be inserted.

5.2. The Form Data File.

A Form Data File is an instance of an electronic form, containing default values for each field defined in a Form Definition File.

Unlike a Form Definition File, which contains the form header, display lines, and field definitions, a Form Data File consists only of a form header, and default values, one per line, for each field definition within a Form Definition File. Since a Form Data File contains no information that defines the on-screen and printed image of the form or defines each field's type and attributes, a Form Data File must be able to specify a Form Definition File from which this information is to be obtained. A Form Data File does this through its form header.

```
1.   hInvoice .
2.   h FDATA [invoice.fdf]
3.   h
4.   Dept. of Production Tech.
5.   Central Stores
6.   Massey University
7.   Massey University
8.   Private Bag 11222
9.   Palmerston North
10.  Palmerston North
11.  ____________________________
12.  __/__/__
13.  180_____
14.  90_Days
15.  ________
16.    987281
17.  _________
18.  ____________________________________
19.  _____
20.  ___.__
21.       0.00
22.  _________
23.  ____________________________________
24.  _____
25.  ___.__
26.       0.00
27.  _________
28.  ____________________________________
29.  _____
30.  ___.__
31.       0.00
32.  _________
33.  ____________________________________
34.  _____
35.  ___.__
36.       0.00
37.  _________
38.  ____________________________________
39.  _____
40.  ___.__
41.       0.00
42.       0.00
43.       0.00
44.       0.00
45.  ____
```

Figure 11. A Form Data File: An instance of the Invoice Form Definition File.

Like a Form Definition File, header lines distinguish an ASCII text file as being a Form Data File. There must be three header lines within a Form Data File and these will usually be the first three lines in the file, although they may be preceded by comment lines. The second of these header lines is reserved for use by Form Data Files. It contains the keyword FDATA, followed by a square bracketed file name, the file name being the Form Definition File associated with the Form Data File.

Form Data Files, which provide an instance of an electronic form, are used specifically for two purposes. Firstly, one or more instances of an electronic form can be created for/by users to eliminate repetitive data entry. Using the electronic invoice as an example, if a user of the form has a number of high volume customers, a number of Form Data Files could be created with such fields as Bill To, Ship To, and Customer No., having the customer details as default values. Secondly, the delivery of the Form Data Files over a network is far more efficient than the delivery of Form Definition Files containing field values. This is not only true in terms of traffic volume, but also in terms of the temporary storage capacity required for these forms.

An example of a Form Data File, an instance of the Invoice Form Definition File, is shown in Figure 11. Note the second header line, line 2, which references the Invoice Form Definition File.

5.3. The Forms Engine.

The forms engine is a library routine that can be incorporated into an application program. An application program (i.e. an ESDI application) firstly describes the processing that is required of the forms engine on some "electronic form," then calls the forms engine, which processes the form as specified by the application. The application must provide the forms engine with the operation that is to be performed on the form and, based on this operation, details that include the nature of the form to be processed (i.e. where it is located and how it is stored), the nature of the output of the processing (i.e. where the output should be placed and how it should be stored), where the form is to be displayed, and how the form should be edited.

The forms engine is used in an application program through references to two Turbo Pascal units [64,65]. A unit is a collection of constants, data types, variable procedures, and functions, which is compiled not to an executable file, but to a special library or unit file that can be incorporated into an application without recompiling. The forms engine makes extensive use of the Blaise Power Tools Plus libraries, published by Blaise Computing [66]. These library routines can be incorporated into Turbo Pascal programs, supporting the rapid development of application programs, while taking advantage of the advanced hardware and software features of the IBM PC environment.

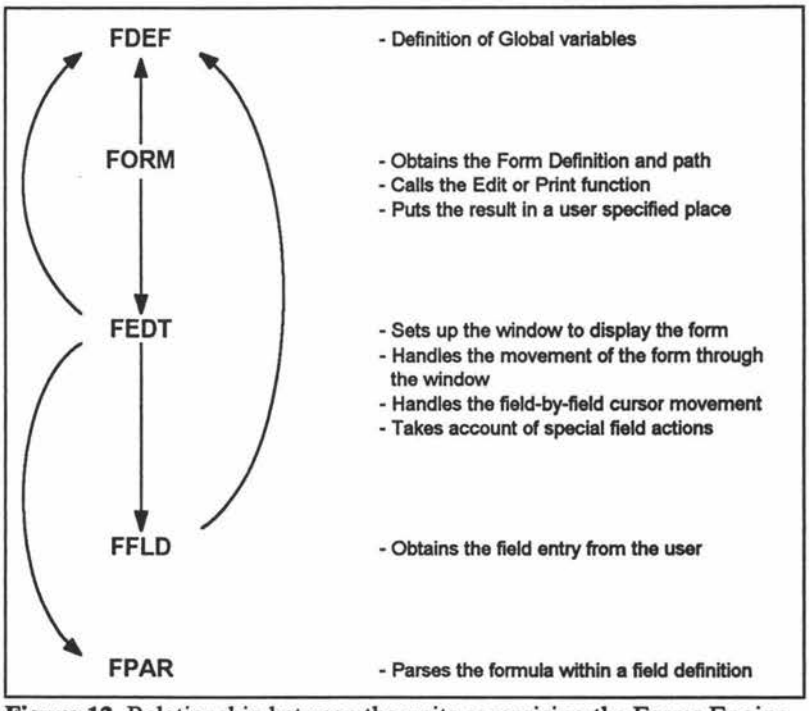


Figure 12. Relationship between the units comprising the Forms Engine.

The forms engine, though referenced by two units, is comprised of five units. The relationship between these units is shown in Figure 12 (complete program listings for these units are provided on the accompanying diskette). Of the two referenced units, the processing that is required of the forms engine, on some electronic form, is first described by an application program through a Form Manipulation Record, which is defined in the

FDEF unit, a unit that defines the global variables for the forms engine (the interface section of the FDEF unit is provided as Appendix A). The application then calls the forms engine (which processes the form) by referencing a library routine in the other unit, the FORM unit (the interface section of the FORM unit is provided as Appendix B). Although only a single library routine is called from a single unit, the two units must be referenced, as the processing of the form, as carried out by the FORM unit, is governed by the Form Manipulation Record, as defined in the FDEF unit.

The application program has complete control over the forms engine through the Form Manipulation Record. By manipulating this record the application program can direct the forms engine to, among other things, process the entire electronic form, or to process only specific fields of an electronic form. Dependent on the actions of the user, the forms engine will either "accept" the form if the user initiates an accept sequence, "abort" the form if the user initiates an escape sequence, or "return" from processing the form if the partial processing of a form is complete. Even once control is returned to the application program, the Form Manipulation Record can be used to determine how the processing of the electronic form terminated, and hence, whether to initiate any further action.

The Form Manipulation Record, its fields and their function, is described in the following section.

5.3.1. The Form Manipulation Record.

The forms engine is called through a single function call with a single parameter: that of the Form Manipulation Record. The Form Manipulation Record, which is declared as the Pascal record type *FormStruct*, is declared in the FDEF unit. The Form Manipulation Record declaration is shown in Figure 13.

Fields within the Form Manipulation Record are described below.

```
Window : _WindowPtr
```

The WINDOW variable is a facility that allows the application program to specify a window in which the processing of the electronic form will be displayed. The WINDOW variable is a Blaise Window Descriptor Pointer. A Window Descriptor is a record structure, having all the necessary information needed to control a windows display and

removal and any input and output within the window. When a window is constructed using the Blaise utilities, a pointer to the Window Descriptor record is returned. By assigning this pointer to the WINDOW variable, the window can be accessed by the forms engine, allowing the display window size and position to be under the control of the application program. If the WINDOW variable is assigned the NIL pointer, the forms engine will use its discretion in displaying the electronic form.

```
FormStruct = record
                Window     : _WindowPtr;
                FormOp     : FormOps;
                FirstFld   : integer;
                NumFlds    : integer;
                FldStr     : LineStrType;
                LstKey     : _KeySeq;
                KeepScn    : boolean;
                AcptKyStr  : KeyWrd;
                FdfDir     : ^_Path;
                Account    : string[40];
                IOSpec     : IOSpecStruct;
             end;

where:

FormOps = (Create, View, Print, Release, Save, XtData);

KeyWrd = string[7];

LineStrType = string[128];

FdfData = array[0..MAXFORMLEN] of LineStrType;

StrArray = record
              DataPtr    : ^FdfData;
              FirstLine,
              LastLine   : integer;
           end;

Format = record
            Where : byte;
            case byte of
                0 : (Name    : ^_Path);
                1 : (Address : ^StrArray);
                2 : (FilVar  : ^text);
         end;

IOSpecStruct = record
                  Input,
                  Output : Format;
               end;
```

Figure 13. The Form Description Record Structure.

```
FirstFld : integer
NumFlds : integer
```

The FIRSTFLD and NUMFLDS variables can provide for the partial processing of an electronic form. Both the variables are of type integer, the FIRSTFLD variable being used to specify the field number of the first field to be processed and the NUMFLDS variable being used to specify the number of fields inclusive to be processed. It should be noted that when referencing fields, the first field number is zero and that to reference all the fields within a form, FIRSTFLD should be assigned zero and NUMFLDS must be at least the number of fields defined within the Form Definition File.

```
FldStr : LineStrType
```

The FLDSTR variable is a means for an application program to alter fields within a form, which a user may subsequently accept. The variable, which is of *LineStrType*, can be assigned a string, which is placed in the field referenced by FIRSTFLD. The string will be displayed in the format that is defined by the FIRSTFLD field definition.

```
LstKey : _KeySeq
```

The IBM PC family of computers supports a keyboard buffer where, every time a keystroke is completed, the BIOS keyboard interrupt service routine places the keystroke. Each keystroke recorded in the keyboard buffer is really a key sequence consisting of a character and a key code. The key code is a code determined by the keyboard service routine indicating the actual key that was pressed.

The LSTKEY variable is a variable that is instantiated by the forms engine. It is a record, defined within the Blaise utilities, that contains the key sequence (character and key code) of the last key pressed before the forms engine returned to the application program.

The LSTKEY variable enables an application program to determine the intent of the user as the forms engine returns from the partial processing of an electronic form. During partial processing the forms engine may either "return" from processing once the specified fields have been processed, "accept" the form if the user initiates an accept sequence (presses the Alt-E or Ctrl-C key sequence), or "abort" processing if the user initiates an escape sequence (presses the Esc key). The key sequence contained in the LSTKEY variable enables the application program to determine how the processing of the electronic form terminated and hence, what further action needs to be initiated.

```
KeepScn : boolean
```

The KEEPSCN variable is a boolean variable that allows the application program to specify whether or not an image of the electronic form should remain on the screen after the form has been processed by the forms engine. If KEEPSCN is set to TRUE, an image of the electronic form will remain on the screen after processing, otherwise, if KEEPSCN is set to FALSE, the window that the electronic form was displayed in will be removed, restoring the screen to the condition it was in prior to the calling of the forms engine.

```
AcptKyStr : KeyWrd
```

During the processing of electronic forms the forms engine may "abort" processing if the user initiates an escape sequence (presses the Esc key) or "accept" processing if the user initiates an accept sequence (presses the Alt-E or Ctrl-C key sequence). The variable, ACPTKYSTR, which is of type *KeyWrd*, allows an application program to specify up to seven key sequences that will be used by the forms engine as alternative acceptance sequences.

```
FDFDir : ^_Path
```

The FDFDIR variable is a pointer to a Blaise General Support type, *_Path*, which is a full DOS path type, that is STRING[79]. This variable enables an application program to specify the location of Form Definition Files to be used by the forms engine. This is of particular use when Form Data Files are being processed and their associated Form Definition Files must be accessible. The FDFDIR variable is also useful should a particular user not be allowed to access all available Form Definition Files. In this case, the application program can direct the forms engine to another location, a location that contains a subset of the available Form Definition Files.

```
Account : string[40]
```

The ACCOUNT variable allows an application program to pass a user account string to the forms engine. This is a useful feature for the auditing and transaction processing of electronic forms. Using this feature each instance of processing of an electronic form could be time and date stamped with a user account string, while recording details of fields entered, etc.

This feature has yet to be implemented.

5.3.1.1. Form Operations.

The FORMOP variable, which is of type *FormOps*, is used by the application program to specify the operation the forms engine should perform on an electronic form. The *FormOps* can be subdivided into two classes; those that "compile" a Form Definition File or a Form Data File, resulting in the creation of an image of the form in memory, and those that operate on the compiled image of a form already residing in memory.

CREATE, VIEW, PRINT, and XTDATA are all form operations that "compile" a Form Definition File or a Form Data File and, if no errors are encountered, create an image of the electronic form in memory. These operations can effectively be further subdivided into two subclasses; those that perform an operation on a form while displaying the image of the form on-screen, and those that perform an operation on a form without the need to have it displayed.

CREATE and VIEW are the only two form operations that display the image of the form on-screen, the image being displayed in a window determined by the WINDOW variable. The *FormOp* CREATE enables a user to navigate the form, field-by-field, entering and modifying field values, dictating whether the form is "accepted" or "aborted" based on their actions. Like CREATE, the other *FormOp*, VIEW, enables a user to move around the form, field-by-field, however, unlike the CREATE operation, field values cannot be modified.

The *FormOps* PRINT and XTDATA also create an image of the form in memory, however, there is no need for these operations to display the image of the form on-screen. The *FormOp* PRINT is an operation that allows a hard copy of the electronic form to be obtained by printing the image of the form to a standard printer. Using this operation the forms engine looks for an optional printer definition file, which contains printer command strings specific to the printer being used, and creates a printer file of the form suitable to be copied to the printer. The other operation, XTDATA, enables field data to be extracted from the image of the form and saved to a Form Data File.

The remaining *FormOps*, SAVE and RELEASE, operate on the compiled image of a form already residing in memory. These operations are used after a form has been partially processed and control has been returned to the application program. They allow the application program, after determining how the processing of the electronic form terminated, to perform any necessary "house-keeping" operations on the image of the form. The *FormOp* SAVE enables the application program to write the image of the form to the user-requested destination, while the *FormOp* RELEASE allows the application program to release the memory used by the image of the form. Using the SAVE operation, the application program can save the field data to a Form Data File or the image of the form, which may include field data, to a Form Definition File.

5.3.1.2. Input/Output Specifications.

The IOSPEC variable, which is of type *IOSpecStruct*, is a record consisting of two variables, INPUT and OUTPUT, of type *Format*. This record, *Format*, allows an

application program to specify where and in what format the input can be obtained and the output should be placed.

The input/output of the forms engine can be from/to a Form Definition File or a Form Data File (depending on the operation, it could also be neither of the two). These files can reside as ASCII text files, in which case they are located by a pointer, of type *_Path*, to a file name, as open text files, in which case they are located by a pointer to a Pascal text file, or in memory, in which case they are located by a pointer to a data structure maintaining the memory. This data structure, a record, maintains a pointer to an array of strings, as well as the line numbers of the first line and last line of the array.

5.4. The Electronic Form.

It has been stated that any equivalent of internal paper-based documents must provide an effective and efficient replacement, in order to retain traditional business practices and to be preferred to the use of paper-based systems. The electronic equivalent of paper-based documents, discussed in this chapter was simple, yet powerful: simple, in that an electronic representation of a paper-based document could be easily constructed (while providing all the content of its paper-based counterpart); and, powerful, in that the mechanism could effectively provide the functions common to the processing of paper-based transactions.

Throughout this chapter the replacement of the paper-based invoice has been used as an example for describing the concepts of Form Definition Files and Form Data Files. The electronic representation of the paper-based invoice was simply and easily constructed, through a Form Definition File, using nothing more than an ASCII text editor. The Invoice Form Definition File was shown in Figure 8. An instance of an electronic invoice, containing default values for each field defined in the Form Definition File, was provided in the Form Data File shown in Figure 11. Continuing with this example, the forms engine can be called from an application program, and, using the examples of the Form Data File and Form Definition File, the electronic invoice can be generated as a replacement for the traditional paper-based invoice.

An application program, by calling the forms engine with a single parameter, that of the Form Manipulation Record, can generate, on-screen, paper's counterpart: the electronic form. An application program, using the Form Manipulation Record detailed below, can generate the equivalent of the paper-based invoice on-screen.

```
with FormDesc do
begin
    Window := NIL;
    FormOp := create;
    FirstFld := 0;
    NumFlds := 255;
    FldStr := '';
    KeepScn := TRUE;
    AcptKyStr := '';
    FdfDir := @Path;
    with IOSpec do
    begin
        Input.Where := DATFILE;
        Input.Name := @InFile;
        Output.Where := OPNDATFILE;
        Output.FilVar := @OutMail
    end;

    Check := Process_Form(FormDesc);
end;
```

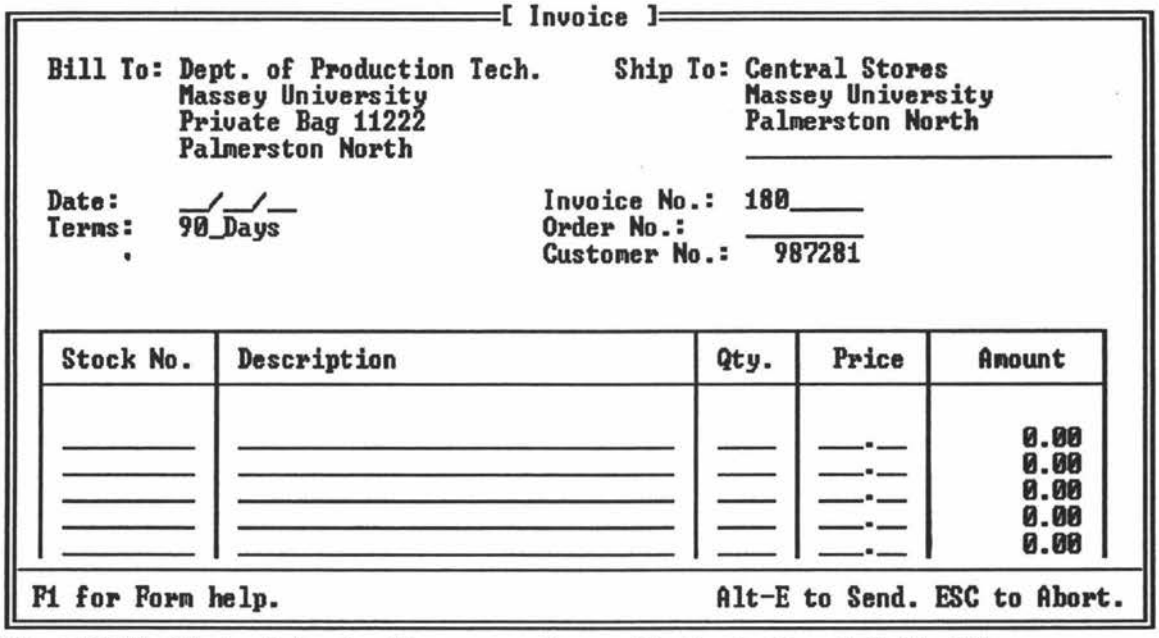


Figure 14. The Electronic Invoice: The on-screen image of the Invoice Form Definition File.

This Form Manipulation Record, which governs the processing carried out by the forms engine, directs the forms engine to create and display, in a window that is determined by the forms engine, an image of an instance of a Form Definition File. Directed to process all the fields within the form, the forms engine takes as its input a Form Data File, stored as a Pascal text file, and produces, if the user initiates an accept sequence, another Form Data File, which exists as an open Pascal text file. If the Form Data File that was input to the forms engine was the instance of the Invoice Form Definition File given in Figure 11, the image of the electronic form would appear on-screen as shown in Figure 14.

CHAPTER

6

A SFSD Example: The Class Roll Request Form

The intention of the first stage of the on-campus ESDI development, referred to as Single Form / Single Destination (SFSD), was to implement a particular electronic form whose destination was pre-determined. The ESDI system, for this first stage, was very simplistic. The ESDI system would display a single rudimentary electronic form, the user would fill in the form details, and when transmitted, the form details would be delivered to the predetermined destination.

Cognisant of the aims of the initiative, the Class Roll Request Form was implemented. The Class Roll Request Form was essentially a request for information. Its purpose: to initiate the extraction of class roll information residing on a database that is not a part of the Campus Network.

Traditionally, lecturing staff and administrative staff would obtain printed class rolls from Management Information Services (MIS). As non-Registry staffs do not have access to Registry computers, where student records and administrative data are held, they were reliant on the accuracy and timely arrival of printed class rolls that were printed and distributed to departments in bulk. These departments would photocopy the class rolls as required and staff would re-key the student records into their own private databases and spreadsheets. Any new requests for class rolls were collected by the Enrolments Office, and when sufficient records had been received, were processed by MIS. The provision of student record information took significant amounts of Registry staff time, estimated to be several person-months. However, it was not only significant amounts of Registry staff time being wasted. Lecturing staff and administrative staff were faced with the near impossible task of trying to obtain timely and accurate class roll information, at a time when this information was highly variable, while dealing with the delays inherent in the system.

The ESDI system then, displayed a single electronic form, the Class Roll Request Form, a lecturer or faculty administrator would fill in the Class Roll Request Form details,

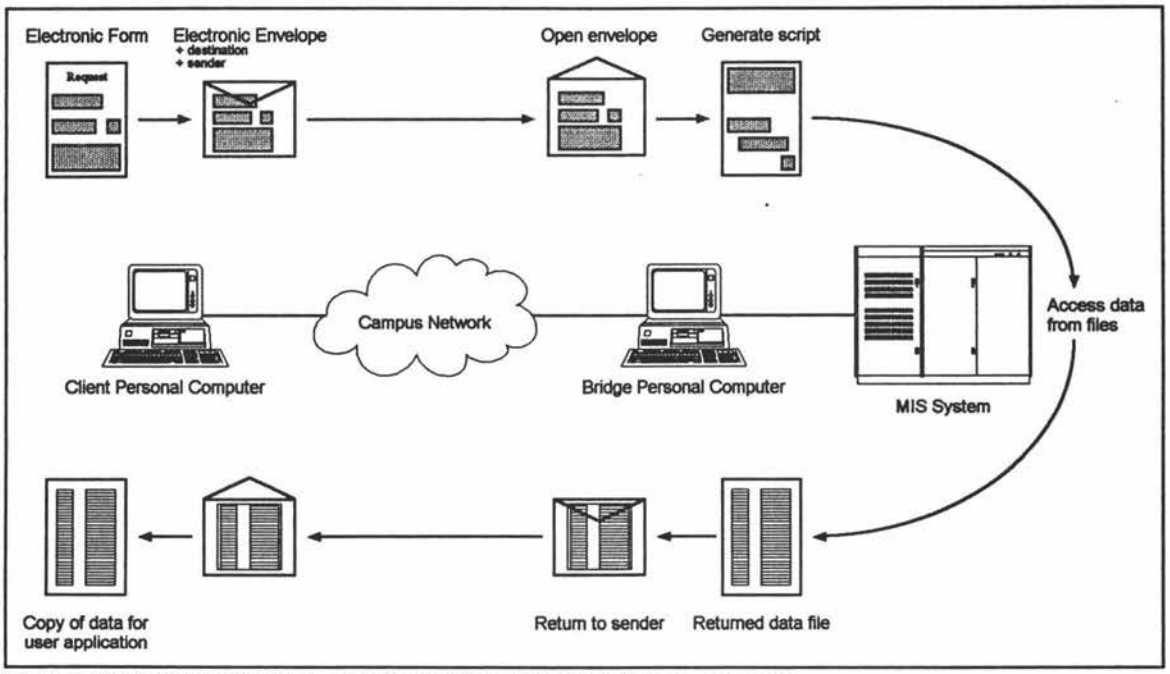


Figure 15. Relationships between the SFSD implementation components.

and when transmitted, the form details would be delivered to the predetermined destination, Management Information Services (MIS).

The SFSD implementation consisted of two components: the *Server* software and the *Bridge* software. The *Server* software was the actual ESDI software that controlled the processing and handling of the electronic form and was implemented as an integral part of the Computer Centre's electronic mail system. The other component of the implementation, the *Bridge* software, was an application program that drove a dedicated personal computer, referred to as the "Bridge" machine, which provided the physical link between the Campus Network and the main Registry computer, which processes the student records system. The *Bridge* software would periodically interrogate a dedicated mailbox for electronic mail containing a Form Data File (created as a result of the successful processing of Class Roll Request Forms by the *Server* software). Any requests for class roll information would result in the extraction of the information from the student database and the return of the results to the originator.

The components of the SFSD implementation, the *Server* software (including the Class Roll Request Form) and the *Bridge* software, are described in more detail in the following sections. The relationship between these components is shown pictorially in Figure 15.

6.1. The Server Software.

The *Server* software, in conjunction with the Computer Centre's electronic mail system, NETMAIL, provided a complete ESDI system. The *Server* software controlled the processing and handling of the electronic form, while the electronic mail system provided both comprehensive coverage of the Campus, through the Campus Network, and a store-and-forward delivery system.

The *Server* software was implemented as an integral part of the Computer Centre's electronic mail system, effectively providing users of NETMAIL with a choice of two editors: one for the sending of unstructured mail, and one for the sending of structured form mail. To send unstructured mail, the user would select the standard electronic mail option, enter the address of the recipient, compose the message, and enter the subject, transmitting the mail when complete. To send structured form mail, the user would select the electronic form option, the Class Roll Request Form would be displayed, and the user would fill in the form details, transmitting the form when complete (its destination having been predetermined).

To use the Class Roll Request Form facility a user had simply to select the electronic form option, which would direct NETMAIL to run the *Server* software. The *Server* software was executed as a child process under NETMAIL, with a single parameter, that of a unique file name. NETMAIL expected two things from this process. Firstly, if the processing was successful then the results (a Form Data File) would be found in a flat file, the name of this file being that specified by NETMAIL by way of the parameter string. Secondly, that, if this file existed, the first line of the file would contain the address that NETMAIL would send the results to. NETMAIL would encase the results in an electronic mail envelope, placing the destination address, sender address, and a standard subject line in the header, and would place the envelope in the electronic mail system, subsequently delivering it to the destination.

The *Server* software was very elementary, displaying a single electronic form, the Class Roll Request Form. The input to the forms engine was the Form Definition File for the Class Roll Request Form, and the output was an open data file, its name having been specified, by way of a parameter string, by NETMAIL. Before the Class Roll Request Form was processed the predetermined destination for the results was written to the first line of the open data file. The predetermined destination, MISEDI, was an electronic mailbox, operated by Management Information Services, dedicated to the ESDI process. The Class Roll Request Form was then processed by the forms engine, and if the user initiated an accept sequence after filling in the form details, the Form Data File, containing

the field values from the Class Roll Request Form, was written to the open data file. However, if the user initiated an escape sequence during the processing of the form, the data file was closed and erased. The program listing for the *Server* software in shown in Figure 16.

```
Program Server;

uses Crt, Unt_Fdef, Unt_Form;

var
    InFile : string;
    OutFile : text;
    Accepted : boolean;
    FormDesc : FormStruct;


begin
    ClrScr;
    InFile := 'CLASROLL.FDF';

    Assign(OutFile, ParamStr(1));
    Rewrite(OutFile);
    Writeln(OutFile, 'MISEDI');

    with FormDesc do
    begin
        Window := NIL;
        FormOp := create;
        FirstFld := 0;
        NumFlds := 255;
        FldStr := '';
        KeepScn := TRUE;
        AcptKyStr := '';
        with IOSpec do
        begin
            Input.Where := FDFILE;
            Input.Name := @InFile;
            Output.Where := OPNDATFILE;
            Output.FilVar := @OutFile
        end;

        Accepted := Process_Form(FormDesc);
    end;

    Close(OutFile);
    if not Accepted then Erase(OutFile);
end.
```

Figure 16. Program listing of the SFSD Server software.

6.1.1. The Class Roll Request Form.

While the aim of this stage was to develop and document a system to enable a lecturer to request a class roll from the main Registry computer, it had also been stipulated that the results should be returned in either a columnated or comma-delimited format. The Class Roll Request Form, therefore, had to provide for four items of information: the course number, the enrolment type, the return format, and the authorisation (the Form Definition File for the Class Roll Request Form is shown in Figure 17). These fields would enable a user to obtain a copy of a class roll, as recorded in the student system, for a particular paper and student category (internal/extramural/cost recovery), in one of two file formats.

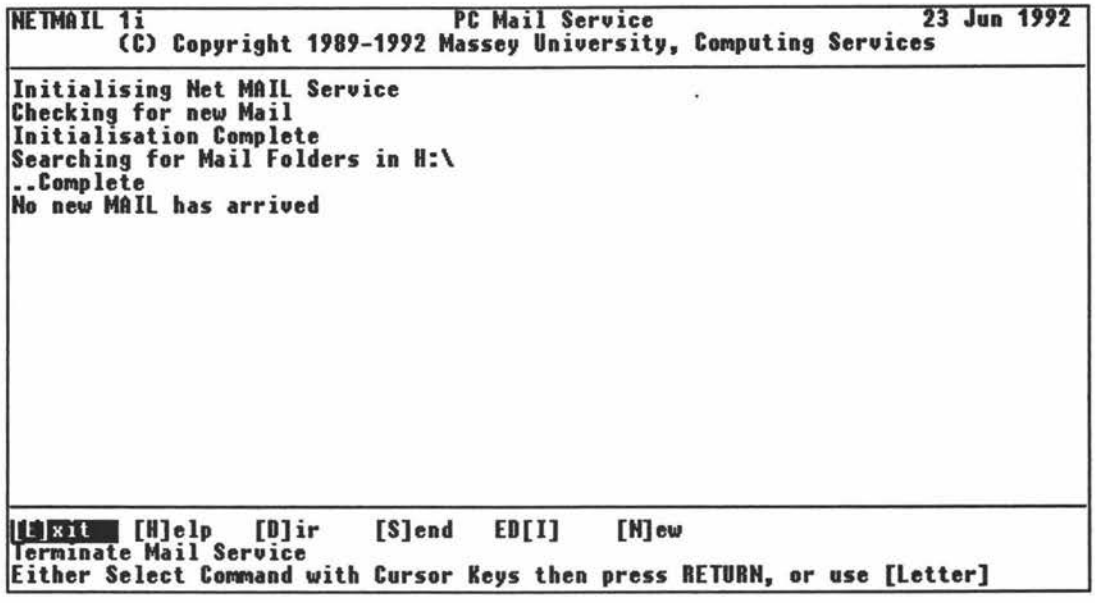
```
hClass Roll Request Form  .
h
h SCREEN
: S1  1 2
: S2  I E C
*
*  Course Number:    \__.___\      e.g. 43.401
=  Numeric req
*
*  Enrolment Type:       \_\       I - Internal
=  Text set S2
*                                  E - Extramural
*                                  C - Cost Recovery
*
*  Return Format:        \_\       1 - Columnated
=  Integer set S1
*                                  2 - Comma Delimited
*
*  Authorisation: \________\
=  Password [] req
*
*
```

Figure 17. The Class Roll Request Form Form Definition File.

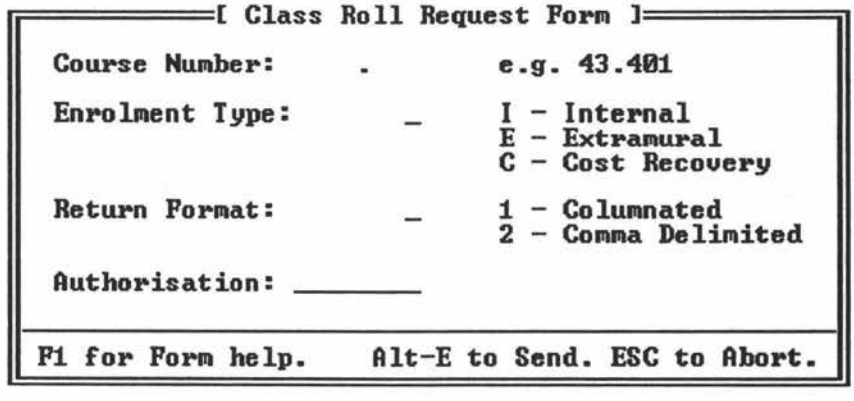
The Class Roll Request Form was essentially a request for information. Its purpose: to initiate the extraction of class roll information residing on a database that is not a part of the Campus Network. To use the Class Roll Request Form facility to obtain an individual class roll any member of staff registered as a network user had simply to select the electronic form option from the Computer Centre's electronic mail system.

The following steps detail how a class roll for a particular paper and student category can be obtained in one of two file formats.

1. Once logged on to the Campus Network, invoke NETMAIL, the electronic mail service.

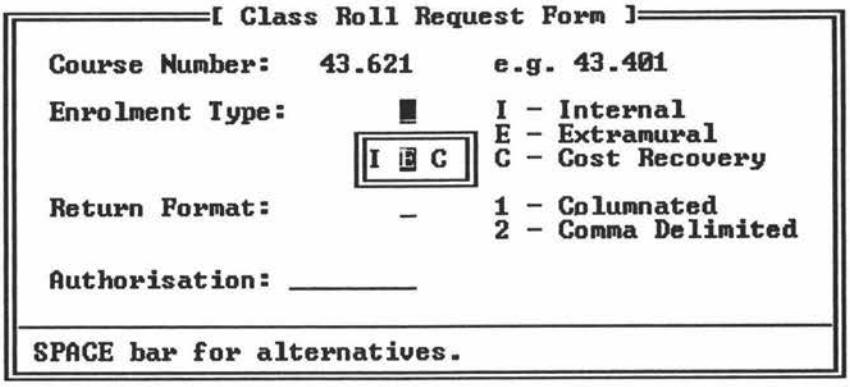


2. Select the ED[I] option to enter the ESDI system and obtain the Class Roll Request Form facility.



3. Enter the course number. Press `Enter`.

4. Select the enrolment type. A *pick* list appears, from which the alternatives may be selected by pressing the spacebar. The default enrolment type is Internal. Press `Enter` to select your choice.

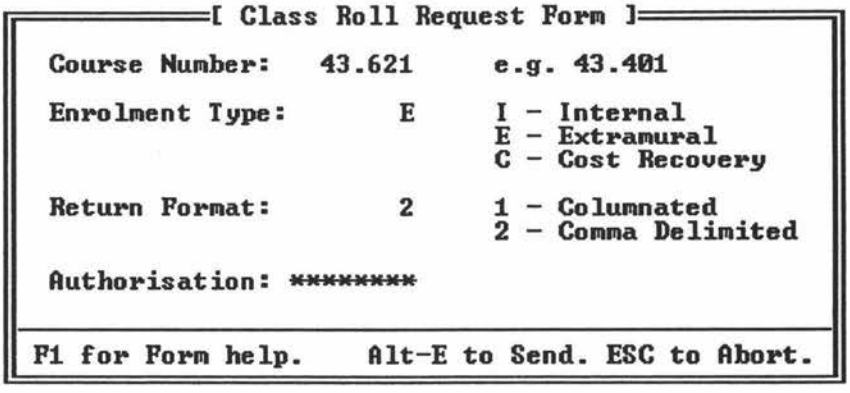


The internal, extramural, and cost recovery categories of students are mutually exclusive. Consequently, if a member of staff wants a class roll that includes all internal and all cost recovery students, two requests will need to be lodged and the resultant files merged and sorted.

5. Select the return format. The default return format is columnated. Press `Enter` to select your choice.

   The class roll can be requested as a columnated, or printable, format or as a comma-delimited format, suitable to be imported into a spreadsheet or database.

6. Enter the authorisation. The authorisation will not be displayed as you enter it. Press `Enter` to confirm the authorisation.



To use the Class Roll Request Form facility a member of staff will need to obtain authorisation, in the form of a password, from the Head of Department responsible for the course. The authorisation confirms that a member of staff is entitled to the information being requested.

7. Any editing corrections can be made to the fields of the form by using the up and down arrow keys to locate the field to be corrected and then re-entering the field contents.

8. Either press `Alt-E` to activate the request for the class roll or press `Esc` to cancel the request.

   After the form has been processed the user is returned to NETMAIL.

6.2. The Bridge Software.

The Massey University Campus Network, to which registered members of staff are connected, and the Registry Network, where the student records system is maintained, are operated as two separate networks for reasons of security. In order to be able to extract class roll information from the student records system, a dedicated personal computer, referred to as the "Bridge" machine, was used to provide a physical link between the two networks. This Bridge machine, which had a network connection to the Massey University Campus Network and a serial link to the Registry student records system, was controlled by the *Bridge* software.

The *Bridge* software was implemented as a recursive DOS batch file (BRIDGE.BAT), calling itself upon completion. A flow diagram for the *Bridge* batch file is shown in Figure 18. This batch file would periodically interrogate the ESDI system's dedicated mailbox for electronic mail containing a Form Data File (created as a result of the successful processing of Class Roll Request Forms by the *Server* software). Any requests for class roll information would result in the extraction of the information from the student records system and the return of the results to the originator of the request.

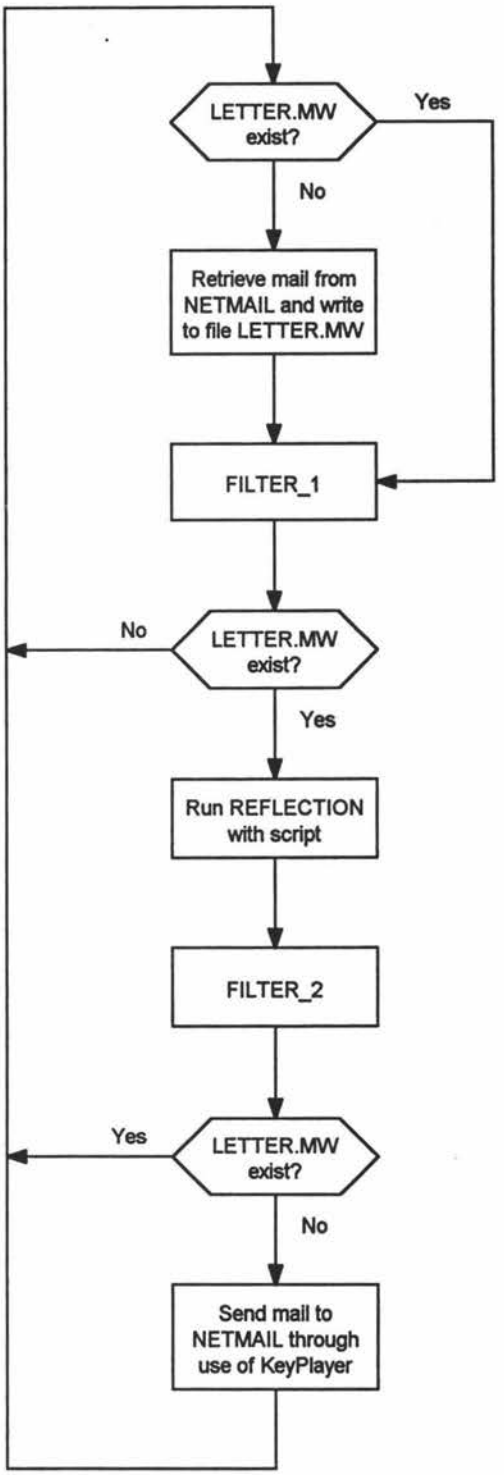


Figure 18. Flow diagram of the *Bridge* batch file.

Since the requests for information from the ESDI system were being delivered through the Campus Network, the batch file had to somehow interrogate the mailbox, to establish whether any mail was waiting to be processed, and, if so, retrieve the items of mail from the electronic mail system. Network users would normally do this by running NETMAIL,

```
1.   @echo off
2.   if exist c:\ccnet\misedi\letter.mw goto next
3.   kp /p /q /i c:\ccnet\misedi\rx_mail.job p:\appl\netmail
4.   :next
5.   \ccnet\misedi\filter_1
6.   if exist c:\ccnet\misedi\temp\haltflag goto prehalt
7.   if not exist c:\ccnet\misedi\letter.mw goto end
8.   if exist c:\ccnet\misedi\process.bat call c:\ccnet\misedi\process
9.   :prehalt
10.  \ccnet\misedi\filter_2
11.  if exist c:\ccnet\misedi\temp\haltflag goto halt
12.  if exist c:\ccnet\misedi\letter.mw goto end
13.  kp /p /q /i c:\ccnet\misedi\tx_mail.job p:\appl\netmail
14.  del c:\ccnet\misedi\tx_mail.job
15.  :end
16.  \ccnet\misedi\bridge
17.  :halt
18.  del c:\ccnet\misedi\temp\haltflag
```

Figure 19. The *Bridge* software: A recursive DOS batch file (BRIDGE.BAT).

which would inform the user if any new mail had been received. NETMAIL was the obvious choice to retrieve the items of mail from the electronic mail system, however, NETMAIL was an interactive program, unable to be effectively executed from a batch file. Because NETMAIL could not be command-line driven, and in order to make use of it from within the batch file, it was run in conjunction with a crude piece of software called KeyPlayer. The KeyPlayer utility was executed using two main parameters; a KeyPlayer generated file, containing keystrokes that KeyPlayer would place in the keyboard buffer when the second parameter, the desired application program, was executed as a child process. Line 3 of the *Bridge* batch file, shown in Figure 19, shows how NETMAIL is called through the use of KeyPlayer. KeyPlayer takes keystrokes from the job file (RX_MAIL.JOB) and places them in the keyboard buffer when NETMAIL is executed. The job file (RX_MAIL.JOB) takes the first item of mail from the queue of mail waiting in the ESDI system's mailbox, saving it to a text file (LETTER.MW), and then deletes the item from the queue. If there were no items of mail in the queue, NETMAIL would terminate without the creation of a mail file (LETTER.MW). The use of the KeyPlayer utility allowed us to make use of NETMAIL from the DOS batch file, even though it was an interactive program.

The item of mail, having been saved to a file, then needed to be processed. Ideally, this would have involved a single program that verified that the mail contained a Form Data File, extracted the required field information from the Form Data File, verified that the request for information was authorised, processed the database script with the appropriate parameters (a script that extracted the required information from the remote student database), and prepared the results for the subsequent delivery to the originator of

the request. Unfortunately, when the system was first built, difficulties were experienced trying to process the database script as a child process from a parent application. The *Bridge* batch file, therefore, evolved with some eight command lines where, ideally, there should have been only one. These command lines, lines 5-12 of the *Bridge* batch file, are described next.

The first of these command lines, line 5, runs a program designed to process the mail arriving at the Bridge machine. This filter (FILTER_1.EXE), which is provided as Appendix C, first checks to see whether or not a mail file (LETTER.MW) exists, if not, it displays a message indicating that it is waiting for mail, subsequently waiting a predetermined amount of time (ten minutes) before the batch file rechecks the mail queue. During this time, if the user presses any key, a flag is created to indicate to the batch file that the user wishes to return to DOS. However, if a mail file (LETTER.MW) exists, then the filter verifies that the item of mail contains a Form Data File, extracts the required field information from the Form Data File, verifies that the request for information is authorised, and creates a batch file to process the database script with the appropriate parameters. The filter, somewhat basically, establishes that a mail file contains a Form Data File after finding a single form header line (a line beginning with "h"). Having established the fact, the filter extracts the sender (from the "From: " line in the electronic mail envelope), the form type (from the first header line), and from the field data, the course number, enrolment type, required format, and authorisation. Using the course number and enrolment type, the authorisation supplied by the originator is checked against a file of security codes for different courses and, if the authorisation is cleared, a batch file for the processing of the database script is created (PROCESS.BAT). This batch file is simply a command line that runs the REFLECTION software, a software package that enables queries on a host system to be run remotely from a connected personal computer. The REFLECTION software processes a script (CLSROLL.CMD) that retrieves the requested class roll information, taking course number, required format, and enrolment type as parameters.

Following the processing of an item of mail arriving at the Bridge machine, an attempt is made to call the batch file that processes the database script (PROCESS.BAT). This process is line 8 of the *Bridge* batch file. If the request for information was authorised, then this batch file will exist and be called. This batch file runs REFLECTION, which processes the script (CLSROLL.CMD) that will extract the required information from the remote student database. The script logs on to the remote student records system, extracts the required class roll information, and then logs off the remote system. Given that the script is successfully processed, a file is created on the Bridge machine, by the host system, containing the class roll information of the requested course

(CLASSR.DOC). Subsequent processing can determine whether this transaction was successful or not by checking a flag (MAILXFER.RES) created by the host system on the Bridge machine.

Whether a batch file exists to process the database script or not, a program designed to process mail leaving the Bridge machine is run subsequently (line 10 of the *Bridge* batch file). This filter (FILTER_2.EXE), which is provided as Appendix D, first checks to see if a flag exists to halt the processing. This flag would exist if, while during the processing of the first filter, the user indicated that they wished to return to DOS after no mail had been received at the Bridge machine or an item of mail arrived that did not contain a Form Data File, in which case it would need to be handled as an exception. Provided no flag is present, the filter determines subsequent processing based on the contents of a temporary file (TEMP) created by the first filter. This temporary file contains the return address of the originator of the request (the sender), the form type, and either the enrolment type and course number or an error message. If this temporary file contains an error message, an electronic mail response is prepared detailing the nature of the error. The error could be due to an incorrect authorisation code for a course number, no access to a requested course number (either the course number doesn't exist or it hasn't been set up in the security file by MIS), or the item of mail was not recognised as being a request for class roll information. If no errors were encountered the filter checks the remote processing flag (MAILXFER.RES) set by the host system, which indicates whether or not the requested class roll information was successfully transferred from the host system to the Bridge machine and, hence, whether or not the request had been satisfied. If the request had been satisfied, then the mail file (LETTER.MW) could finally be deleted, otherwise, the mail file was retained while the filter, once again, subsequently waited a predetermined amount of time before attempting to re-establish a connection with the remote student records system. Given that the request had been satisfied, the results were prepared to be returned to the originator of the request, via the electronic mail system.

The returning of the results to the originator of the request via NETMAIL was a rather involved process. NETMAIL was designed for the interactive sending of unstructured mail. An item of mail would be created using an editor from within NETMAIL, the mail would be saved to a standard mail file (LETTER.DOC), and NETMAIL would enclose the contents of this file in an electronic envelope, subsequently delivering it to its destination. Essentially, to return the results to the originator, we wanted to use NETMAIL as an electronic delivery mechanism, passing NETMAIL the file that we wanted delivered, along with the destination that we wanted it delivered to. We also wanted to run NETMAIL from a batch file and be able to distinguish between the returning of error messages (which would be delivered as electronic mail) and class roll

information (which would ideally be delivered as a data file, with an accompanying electronic mail message advising of the successful transaction, i.e. the delivery of the data file - users shouldn't have the burden of having to remove electronic mail headers from a response, in order to obtain the desired data file).

By making some changes to the way NETMAIL was configured and by making use of KeyPlayer we were able to use NETMAIL as an electronic delivery mechanism. Firstly, NETMAIL was configured to prompt for a third item when sending unstructured mail, a new entry in the electronic mail header. NETMAIL already prompted for the destination (the "To: " line) and the subject (the "Subject: " line). Through the configuration file (NETMAIL.CFG), NETMAIL was configured to also prompt for an "X-File: " header line. The "X-File: " header line enabled NETMAIL to distinguish between standard electronic mail messages and ESDI mail that contained a standard electronic mail message and an associated data file. If the X-File line was empty, the contents of the standard mail file (LETTER.DOC) were treated as an electronic mail message. However, the X-File line could contain two parameters:

```
X-File: <message file> <data file name>
```

Of the two parameters, the contents of *<message file>* would be treated as an electronic mail message and the contents of the standard mail file (LETTER.DOC) would be saved in the file name specified by *<data file name>*, in the recipient's network directory. The X-File header line provided a method whereby NETMAIL could be used to deliver a data file containing the class roll information for the requested course. NETMAIL, however, expected to run an editor to create the standard mail file (LETTER.DOC), therefore one further change was made to the NETMAIL configuration file (NETMAIL.CFG). NETMAIL could be configured to run an editor of the user's choice, so NETMAIL was configured to run a dummy program as an editor, that is, from NETMAIL's perspective, an editor that lacked all functionality. Secondly, the filter created a KeyPlayer job file (TX_MAIL.JOB), from a template of that file (TX_MASK), to enable NETMAIL to return the results to the originator of the request. This job file (TX_MAIL.JOB) had to be created from a template as the destination (the address of the originator of the request) for the results and the X-File line were specific to each item of mail. If an error was encountered by the filter, the filter would write the error message to the standard mail file (LETTER.DOC) and specify an empty X-File line when creating the job file (TX_MAIL.JOB), which would result in the error message being delivered to the originator of the request as electronic mail (an example of a transaction that has resulted in an error message is shown in Figure 20). However, if the request had been satisfied, the

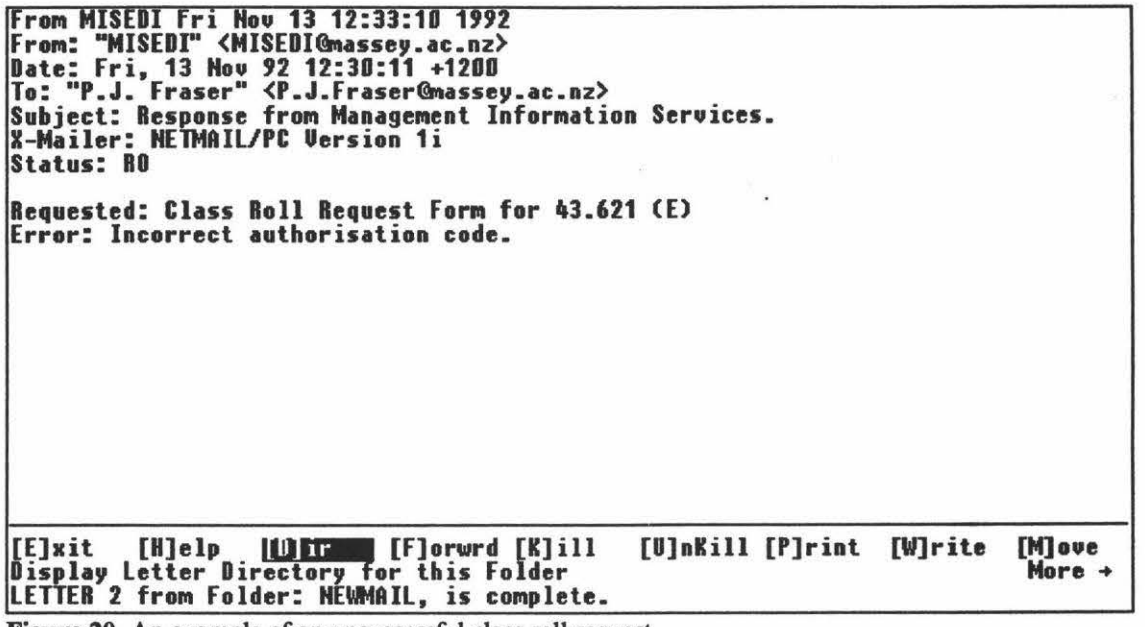


Figure 20. An example of an unsuccessful class roll request.

filter would copy the contents of the class roll information file (CLASSR.DOC) to the standard mail file (LETTER.DOC) and specify a message file (CLSROLL.MSG) and a data file name (a concatenation of the enrolment type and course number) as parameters of the X-File line when creating the job file (TX_MAIL.JOB), which would result in the contents of the standard mail file (LETTER.DOC) being saved in the specified data file in the recipient's network directory, and the contents of the message file being delivered to the originator of the request as electronic mail (an example of a successful class roll request is shown in Figure 21). The electronic mail, as well as notifying of a successful transaction, would indicate the name and location of the data file containing the results.

Finally, in line 13 of the *Bridge* batch file, the results of the request for information were returned to the originator. The job file (TX_MAIL.JOB) directs NETMAIL to send an item of unstructured mail, specifying the destination (the "To: " line), the subject (the "Subject: " line), and the mail type ("X-File: " line), and then exits NETMAIL.

There was a considerable amount of concern expressed about the fact that a machine connected to the Campus Network had access to student record files maintained on the Registry system. In order to safeguard the Registry system from unauthorised access, a number of features were implemented in the system. Firstly, information could only be obtained from the Registry system, that is, the connection established with the Registry database was read-only, therefore, in the unlikely event of someone getting into the system, they were not in a position to modify any records. Secondly, to process a query through the serial link from the Bridge machine, the REFLECTION software had to be used, restricting access to databases through the built-in security features. Thirdly, the

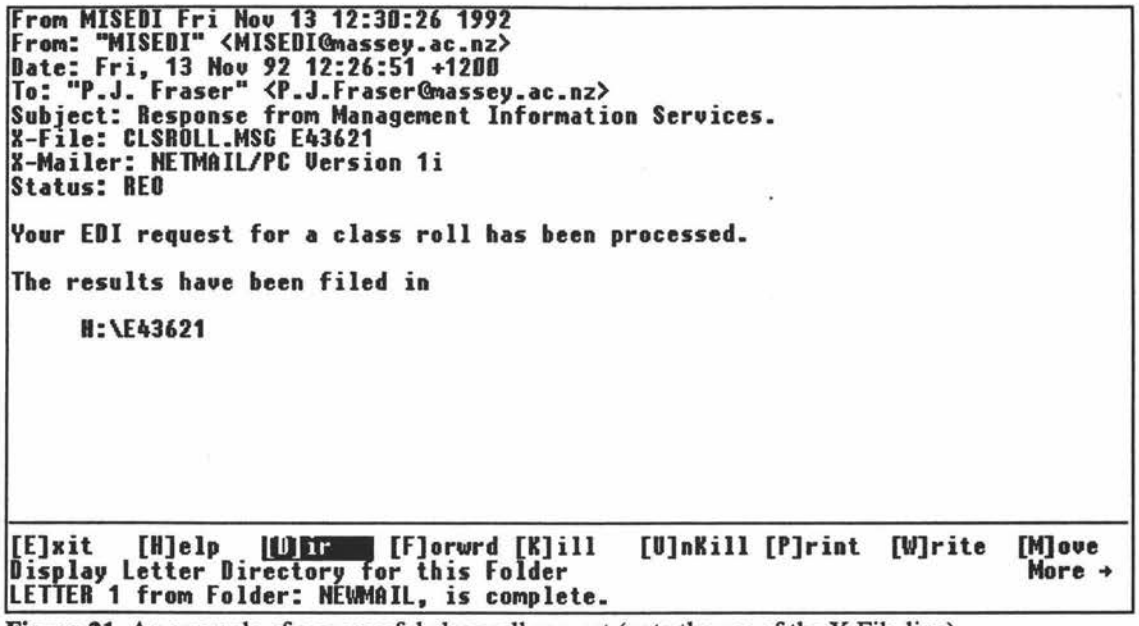


Figure 21. An example of a successful class roll request (note the use of the X-File line).

bridge machine maintained a full log of the transactions that it received, as well as details of the processing of the transaction (FORMS.LOG). And, finally, in order to use the system in the first place, the user had to be a registered network user with enough privileges to use the electronic mail system. A comprehensive discussion on aspects of security issues and ESDI is discussed in the following chapter.

CHAPTER

7

Security Aspects of Electronic Structured Document Interchange

"Security is the protection of data from accidental or deliberate threats that might cause unauthorised modification, disclosure, or destruction of the data; and the protection of the information system from degradation or non-availability of service.

They all involve the firm in some kind of loss:

1. *loss of data integrity,*
2. *loss of availability of service, and/or*
3. *loss of confidentiality."*

Kemp [67]

In general, moving a transaction through the ESDI process involves three functions (steps) within each ESDI application computer system. These can be described as:

- **The Communications Handler**, which transmits data between trading partners;
- **The Application System**, which processes the data to be sent to, or received from, the trading partner, and;
- **The ESDI Interface**, which manipulates and routes data between the application system and the communications handler.

These functions comprise a generalised ESDI architecture. The generalised architecture is generic as it is not dependent on any specific hardware, software, communications protocol, or processing environment. Regardless of whether a transaction

is sent or received by a microcomputer, minicomputer, or mainframe, transactions are processed through the ESDI architecture. Consequently, the processing of those transactions through each step in the ESDI process requires the appropriate security and control.

By reviewing the data flow through this generalised ESDI architecture, establishing the business risks associated with ESDI, and by examining the associated control and auditing aspects, in terms of application and general controls, the security aspects associated with the technology are discussed.

Appendix E and F contain papers, written by the author, that are specific to electronic data interchange. "Security Issues in Electronic Data Interchange (EDI)," contained in Appendix E, was submitted as part of the internal assessment for *58.467 Information Systems Security* and "Management Recommendations For Handling Electronic Data Interchange (EDI) Risk Aspects," contained in Appendix F, was submitted as part of the internal assessment for *26.381 Information Systems Management*. This discussion is closely tied to these papers, as would be expected, and is hence a summary of the two, highlighting the major security aspects concerning electronic structured document interchange.

7.1. Communication Issues.

As indicated earlier, the communications handler transmits data between trading partners. There are several options available to transmit electronic documents between trading partners. The implementation generally involves adopting a network configuration, technical network standards, functional network standards, and building the appropriate translation software.

As electronic documents are transmitted intra-company there is generally not the same emphasis placed on communications issues as for inter-company transactions. However, depending on the nature of the company, and the means to which ESDI is used, these issues may become increasingly important.

7.1.1. Network Configurations.

For companies that have divisions that are either geographically dispersed, or operate as separate business entities that are not on a shared network, there are four general configurations that can be adopted for an ESDI services network.

A Point to Point configuration involves direct communication between business partner computing facilities by either leased lines or dial up services. Point to Point configurations are discussed further in Appendix E (section 3.1.1.).

Other approaches utilise a public telecommunications network for communication services. Either the basic public telecommunication network can be used for communication services or mailbox services, which can be used for the sending or receiving of transactions, can be provided in addition to the basic services. These configurations are further described in the Public Network Configuration and Value Added Networks (VANs) sections of Appendix E (sections 3.1.2. and 3.1.3., respectively).

Finally, Appendix E (section 3.1.4.) describes Third Party EDI Service Networks. These third party networks, often known as Clearing House Systems, serve as an EDI services bureau.

7.1.2. Technical Network Standards.

Two sets of technical standards must be implemented within an ESDI network: interchange standards and communication standards.

Interchange standards are responsible for the interchange of electronic messages across a network. The emerging standard for this operation is the electronic messaging standard ANSI X.400, which is described further in Appendix E (section 3.2.2.).

The setting of communications standards involves adopting a set of standards related to communications speed and low level communications protocols (e.g. packet switched network using X.25).

7.1.3. Functional Network Standards.

Establishing the functional standards within an ESDI network requires agreement between trading partners on transaction, message, security, and data storage standards.

Setting transaction standards entails agreeing with trading partners what business transactions will be conducted using ESDI, and any restrictions to be enforced. For example, trading partners may wish payment transactions of only up to $500 be involved within the ESDI system.

Establishing the message standards determines the form and content of each ESDI transaction. These standards will dictate what data is to be transmitted as part of each ESDI transaction, and the format of the data within each transaction.

Commercial transactions often involve some data that is confidential or sensitive (e.g. discounts offered to specific customers). Trading partners will have to determine how sensitive data is to be protected whilst in transit from trading partner to trading partner. This is the role of security standards.

Trading partners will also have to agree upon the storage of sensitive data and the length of time that back-ups of transactions and transaction logs should be retained.

7.1.4. Translation Software.

Application programs never automatically generate or accept data files in standard format. This means that translation to/from standard format may have relevance in ESDI implementations if the electronic documents are to be used in conjunction with different application software. Translation software is discussed further in Appendix E (section 3.4.).

7.2. The ESDI Architecture.

7.2.1. The Communications Handler.

The communications handler transmits and receives electronic documents between trading partners and/or VANs. Organisations frequently use their existing communications facilities as a communications handler for ESDI transactions. The software used in the communications handler is usually designed to detect errors to protect against transmission error or interruption.

7.2.2. The ESDI Interface.

The ESDI interface usually consists of the ESDI translator and the application interface.

The ESDI translator translates between the standard format and a trading partners proprietary format (internal application formats). It may generate and send functional acknowledgements (standard ESDI transactions that tell the trading partner that their electronic documents were received) and verify the identity of partners and check the validity of transactions by checking transmission information against a trading partner master file.

The application interface moves electronic transactions to, or from, the application systems and performs data mapping. Programs separate and deliver inbound transactions to the application systems and gather transactions from the application systems for outbound transmission. Controls need to be incorporated into this function to ensure that transactions are processed completely and accurately between the communications interface and the application systems.

7.2.3. The Application Systems.

Many companies have substantial investments in application systems that process transactions that originated as paper documents. Many companies are able to preserve those investments by leaving the existing application systems unchanged. This is usually accomplished by designing "front end" procedures that enable ESDI transactions to utilise existing applications systems. Such front end procedures become the ESDI interface, and replace the manual procedures previously used to process paper transactions. Although new controls should be developed for the ESDI interface, the controls for existing applications will usually remain unaffected.

7.3. ESDI Risks.

Section 5 of Appendix E briefly outlines some of the business risks associated with electronic data interchange, many of which are applicable to ESDI. Management and auditors need to be aware of these risks as they provide a framework for placing the appropriate security and control mechanisms within the ESDI application or the ESDI environment.

The risks relevant to ESDI include:

- loss of business continuity - "Going Concern problem";
- loss of confidentiality of sensitive information;
- increased exposure to fraud;
- overpayment;
- loss of audit trail;
- concentration of control;
- errors in the information system, and;
- not achieving anticipated cost savings.

These risks and exposures will be addressed by examining the control and auditing issues related to ESDI.[4]

[4]Although many of these risks and exposures are not directly related to security, the vulnerabilities faced by businesses are diminished through the appropriate implementation of security controls.

7.4. Control and Auditing Issues.

Internal accounting controls are procedures that ensure accurate and reliable financial reporting as well as the safeguarding of the company assets. These controls are designed to prevent and detect financial statement inaccuracy that might affect the entity's financial position. Although internal and external auditors may have different objectives, both may find it necessary to evaluate internal accounting controls.

Internal audit is an appraisal function that examines and evaluates an organisation's activities, and encompasses the organisation's system of internal control to determine how well control objectives are met. External audit is an independent appraisal function that results in an opinion about an organisation's financial statements. The opinion is based on evidence regarding assertions made by management in the financial statements that have been evaluated by the auditor.

Generally, internal control procedures are divided into two major categories: application controls and general controls. Application controls are *specific controls over each separate computer application that ensure only authorised data is processed completely and accurately by that system*. Application controls can be automated or manual in nature.

Since many application's controls in today's complex information processing systems are automated, it is critical to ensure that adequate controls exist over the functioning of the programs. This is the purpose of general controls, which relate to the development, implementation, maintenance, and operation of the applications systems and the security of data files and programs. They are designed to ensure that *programmed accounting and control procedures are developed properly, performed consistently, are modified only with proper authorisation, and that appropriate division of duties and responsibilities is enforced.*

7.5. ESDI Controls.

Since ESDI replaces the paper document environment with an electronic one, existing internal control procedures may need to be changed to accommodate the ESDI processes. More control will now need to be exercised by the user's computer systems to replace manual controls such as sight reviews of documents. More emphasis will need to be placed on control over data transmission, with selection of document standards and appropriate protocols and use of acknowledgements becoming very important.

There will be reduced opportunities for review and authorisation. For example, organisations may pay on receipt of goods rather than on invoice. This will eliminate the current invoice authorisation review. As payment will probably be approved on the basis of sighting of a list of payments due, rather than on review of authorised documentation for each payment, more emphasis will be placed on the initial authorisation of orders and on evidence of receipt.

Transactions will be actioned more quickly. For example, orders received will be processed and filled almost immediately.

This means that any controls will need to operate far more quickly to highlight errors and inconsistencies and correct these before they impact production or delivery.

Controls need to be in place to address:

- transmission errors (these should be addressed by the protocol and document standard selected and by the use of acknowledgements);
- errors in the generating applications of the message sender (the receiving organisation needs to assess whether the transmissions received are consistent with transactions normally received from that trading partner), and;
- manipulation of data, either within the sending or receiving application, on transfer to/from applications, while held in storage, or on transmission. This will include controls within the applications, procedures to address the risks associated with specific transaction types, consideration of message authentication and encryption techniques, key management procedures, etc.

Often the design of controls will require some imagination in order to implement a regime that will protect the organisation against itself and against other potential weak links in the ESDI chain. This may mean increased use of expert systems as front end transaction processors and editors.

The nature of controls may not change significantly, but the way they are executed is likely to undergo major change and the computer is likely to be utilised more and more to provide this control.

All this means that organisations will place more reliance on the electronic data processing (EDP) control environment. If an organisation cannot be sure that controls will be applied consistently, that its data is not adequately protected, and that it cannot maintain operations, it is at risk.

At a minimum, key issues that must be considered include:

- ESDI/systems access security;
- encryption;
- transaction and management audit trails;
- implementation of an information security policy;
- data communications policy, and;
- adequate user control procedures in end user areas.

The next sections address the application and general controls that are affected by ESDI. Application controls will be discussed for each function in the generalised ESDI architecture. General control issues unique to ESDI systems will also be identified.

7.6. Application Controls.

Many controls used in paper document processing are not effective in the ESDI environment. ESDI application controls are usually automated and perform procedures that ensure transaction data is processed correctly throughout the ESDI architecture. Those controls should be used throughout the ESDI process, and address the completeness and accuracy of input as well as the authorisation of the input. Additional controls are required to monitor the generation of other ESDI transactions such as functional acknowledgements. Note that once the data is in the application system it is subject to the usual controls applied to data processed by that application system.

Completeness of input controls, which are discussed in Appendix E (section 8.1.), ensure that all transactions are input and accepted for processing *only* once. Techniques to accomplish this typically include batch totalling, sequence numbering of electronic documents, and one for checking against a control file (functional acknowledgements processing).

Accuracy of input controls ensure key transaction data is input accurately and accepted for processing, and are discussed in Appendix E (section 8.2.). Techniques may include edit checking of key data or matching transaction data to master files.

Authorisation of transaction controls are designed to ensure that only valid and properly authorised transactions are processed. Generally, controls over authorisation performed by the communications handler involve typical sign on procedures, including ID and password verification of the trading partner or VAN. Authorisation of transactions is discussed further in Appendix E (section 8.3.) with discussions on the generation of transactions, file continuity, asset protection, and the updating of data.

7.7. General Controls.

General controls ensure data integrity by controlling development, maintenance, and operation of computer systems, as well as data and program security. Although existing general controls are applicable for ESDI systems, new issues need to be addressed as a result of ESDI processing.

The implementation of ESDI systems should be controlled during the development process and should include controls over design, programming, testing, implementation, and modification. Just as with an application system, the computer operation controls are also important in ESDI processing. In addition, it is strongly recommended that an internal and/or external auditor knowledgeable about ESDI participate during the entire system development process.

Another issue to be considered with ESDI systems is that the definition of "the system" may be expanded. The processing of a transaction by the "ESDI system" may include VANs or multiple trading partners. Therefore, additional data security, confidentiality, and integrity considerations need to be addressed when the external environment is included in the system definition. The following discussion details general control objectives and related ESDI control issues.

7.7.1. Implementation and Maintenance.

Implementation controls are designed to ensure the appropriate development, testing, and implementation of a system. Once the system is operational, maintenance controls ensure that changes to the system are designed and implemented correctly. Frequently, ESDI systems are implemented or modified according to the same procedures used for application systems. Appendix E (section 9.1.) describes some important implementation and maintenance tasks that are applicable to ESDI.

7.7.2. Data File and Program Security.

Security controls are designed to help prevent or detect deliberate, unauthorised changes to data files and programs. Security controls may include physically securing hardware, data, and programs, and restricting logical access to system data and programs.

7.7.3. Operational and Management Control Issues.

ESDI raises operational and management control issues that relate to an organisation's efficient use of resources, achievement of management's goals and objectives, and adherence to legal and regulatory policies and procedures. Failure to recognise the importance of ESDI on these controls may reduce the strategic advantages of ESDI even if application and general controls are effective.

Some operational and management control issues result from the elimination of paper and the reduction of documentation. Other control issues relate to the electronic nature of the activity. An organisation can address many of these issues by establishing procedures for ESDI use and training. Operational and management control considerations will vary by industry, organisational structure, existing facilities, and other resources. Some key issues include:

7.7.3.1. The Trading Partner Agreement.

This defines the trading relationship and requirements of trade within the ESDI environment between trading partners. Items that may be in the agreement include: transaction sets to be utilised; definitions for signature and "document"; the ESDI standards to be used; authorised verification procedures; responsibility for lost or stolen data; and, timing considerations. Generally, the agreement includes the terms and conditions stated on the back of paper documents. Appendix E (section 9.3.2.) includes sample Trading Partner Agreements and Network Agreements.

7.7.3.2. File Retention.

File retention issues result from the fact ESDI uses electronic source documents. Issues include how long ESDI transaction files and data should be retained for tax, audit, back up, and management purposes and the form the data should be stored in. Organisations should review their record management procedures to ensure that electronic documents are maintained properly and securely for an appropriate amount of time. Issues to consider when developing policy include: initial retention time; evidence format; hardware/software compatibility; and, system auditability and security. The file retention issues of paper reduction/elimination and audit/management trails are discussed in Appendix E (section 9.3.3.).

7.7.3.3. User Education.

This is an important consideration since ESDI may affect many functions and organisations within the business entity. Therefore, all affected personnel (management and staff) should be aware of their responsibilities in the ESDI environment. User education should stress the organisation's ESDI procedures and provide information on specific company policies.

7.7.3.4. System Availability and Reliability.

These issues become more critical as more ESDI relationships are established and ESDI technology is integrated into business operations. Since trading partners' relationships are generally close, poor planning in handling of system availability and reliability issues may alienate customers and suppliers, cause operational disruptions, and prove costly to both the sender and recipient. An integral aspect of system availability and reliability, contingency planning, is discussed in Appendix E (section 9.3.5.1.).

7.7.3.5. Message Authentication.

For intra-company transactions, the first component of information reliability is message authenticity.

Authenticity controls can be placed around electronic messages. Many of the controls are identical to those applying to paper documents (e.g. surrounding facts and circumstances), but some new controls are necessary, so that:

- secure, professionally maintained routes of communication, which are reasonably protected from unauthorised users, can be used;
- messages and acknowledgements can quickly be checked against one another, trading partner profiles, and other records for consistency;
- passwords can be used to identify the message sender to an intermediary network, and;
- secret passwords can be built into messages themselves.

Cryptographic schemes can substantially enhance the authentication of electronic messages. Yet cryptography today requires special equipment or programming, staff training, key management, and trading partner coordination.

The area of message authentication, including authentication, encryption, and electronic "signature" capabilities, is discussed in Appendix E (section 9.3.6.).

7.8. Management Recommendations For Handling Risk Aspects.

The benefits of ESDI are enormous. Ultimately, as companies integrate ESDI with their existing systems, they will be able to respond more quickly to trading partner requests, hold less stock, handle smaller deliveries, and demand improved responsiveness from suppliers and transport providers. Dates for automatically generated fund transfers can be clearly established, offering the potential to improve cash flow and interest cost management.

These benefits have a price. As transaction processing becomes more automated and more closely linked to operational decision making, the reliability and security of a company's computer systems and networks, and those of the parties with whom and via whom they interchange data, become increasingly critical.

To meet the challenge of faster response requirements, controls will be increasingly automated, with a bias towards preventive controls. Different types of controls will be required to replace visual review. The company's existing application controls may be inadequate to handle this new method of operation.

As a company's ESDI implementation evolves, the organisation will need to reassess the controls over their existing environment and applications. They should address additional risks that may be derived from the implementation of ESDI because of the resulting changes to their systems, procedures, and operations.

All organisations will be exposed to risk no matter how they do business. The exposures a business may face when using ESDI include fraud, disruptions to production and manipulation of company operations, impaired customer service, inappropriate decision making, inability to rely on financial statements, loss of market share or competitive advantage, and significant financial loss or embarrassment. These risks and the appropriate ESDI controls are further detailed in Appendix F (section 2 and section 3, respectively).

7.8.1. ESDI Strategic Planning and Management Concerns.

ESDI will affect the organisation and the industry within which it operates. Management will need to consider the difference ESDI will make to the business strategically, operationally, and in terms of accounting and control.

7.8.2. Application Controls.

In evaluating application controls companies should assess the adequacy of controls over existing systems that will be affected by the implementation of ESDI, as well as identifying how ESDI will generate a need for **different or additional** controls. Controls may differ depending on whether the transactions are inward or outward bound.

The whole application and any inter-related applications must be looked at as, once transactions are "in the system," there may be limited opportunity for further review and follow up. Segregation of duties (particularly in terms of input, generation, and authorisation of transactions) is an important consideration, particularly for outbound transactions.

A transaction can affect business functions far more quickly where ESDI is used. Thus the requirement for faster response will dictate the need for intervention earlier in the processing cycle to act on potential problems, particularly for inward bound transactions.

The key emphasis will be on timing and for this reason automated controls will be preferred. There is a greater need for preventive controls as detective controls may be applied too late. Automated controls will need to become more "intelligent" to compensate for the reduced human intervention in processing.

7.8.3. ESDI Authentication.

Authentication of ESDI messages is important to ensure the company is exchanging valid data with the **correct and authorised** trading partner. Once the company moves to a full implementation of ESDI they may no longer have the protection of paper documents with signatures. Authentication controls will need to cover both inbound and outbound transactions.

For outbound transactions, it is also important to consider the overall and specific applications security. Who can "give the OK" to send a message? Can invalid but apparently authorised transactions get through the system because of poor segregation of duties or review of initial documentation?

7.8.4. Security Inside The Computer.

These days, security is an important issue. White collar crime, espionage, viruses, and, more commonly, simple errors put computer systems at risk. As the reliance on computers increases so does the need for good security. Implementation of ESDI is another compelling reason to ensure that security is sound.

Some companies may consider that their internal security is already effective. They still need to consider whether the changes to their systems, as a result of ESDI, will raise their security requirement threshold.

7.8.5. Communications Security.

Communications security has been a hot topic for some time now as more organisations connect to local and wide area networks to meet their business needs. ESDI is yet another reason to address network security.

What communications standard (e.g., X.25, X.400) will be used and how will it be implemented? What features of the standard selected will you implement? What media and equipment will be utilised? What additional security hardware and software will you require (dial back modems, encryption hardware)?

7.9. Summary.

The challenge management and auditors face is how to implement ESDI technology to attain business objectives, and at the same time appropriately control its associated risks. While ESDI does pose some threats and risks to management these can be satisfactorily controlled if proper planning and consideration is given to the implementation process.

It's important that both application and general controls be in place over the ESDI environment (including the application systems) to ensure the integrity of electronic data. If management are using or propose to use ESDI, they should pay particular attention to the controls over their existing computer systems and their environment must be sufficiently sound to protect against any additional risks they may face via ESDI. If not, additional controls must be implemented to counteract the increased automation and speed of ESDI trading.

The review of appropriate controls will require sound knowledge and experience of data processing, data communications, and EDP audit principles and standards. Management should ensure that they have access to this expertise before commencing their ESDI implementation.

CHAPTER

8

A MFMD Example: The Financial Transaction Request Form

The aim of the second stage of the on-campus ESDI development, referred to as Multiple Forms / Multiple Destinations (MFMD), was to increase the functionality of the ESDI system implemented in stage one by making a facility available that would offer a choice of electronic forms and a choice of destinations. Each electronic form would provide the equivalent of an internal paper-based document and could have a predetermined destination.

The ESDI system would function as follows: if the forms do not have a predetermined destination (or if there is more than one destination provided), a menu of destinations is displayed and the user selects the appropriate destination; a menu of available forms is displayed and the user selects the appropriate form; the form is displayed, and the user fills in the form details; finally, when transmitted, the form details are delivered to the proper destination.

It was specified that, to demonstrate the increased functionality of the ESDI system, the Financial Transaction Request Form would be implemented. Once again, the Financial Transaction Request Form was essentially a request for information. Its purpose: to initiate the extraction of account status and financial report information residing on a database that is not a part of the Campus Network.

The University operates a central administrative structure, with all accounting functions being conducted through the Management Accounting section of Registry. As non-Registry staffs do not have access to the Registry computers, where accounting information is held, details of an account status, including financial transactions for an account, were printed monthly and distributed to the department operating the account. Because, from an administrative point of view, the University is a relatively complex system, the Departmental Administrators overseeing these accounts experienced significant delays between the time that a transaction was initiated and when the transaction was

processed by the central accounting system, and subsequently reported to the department. The Departmental Administrators possessed details of orders committed from an account, while the central accounting system reported details of payments made from an account, therefore, because of the delays inherent in the system, the reports detailing the account status would be significantly different from what they were known to be by the Departmental Administrators. Over a period of time many departments set up their own accounting systems to maintain the accounts that they were responsible for. Using these private systems, based on known funds committed, Departmental Administrators were able to provide Heads of Department with realistic budgeting information by keeping track of the "real" funds available in each account. The Departmental Administrators could not, however, account for the costs apportioned to them by the central accounting system (e.g., depreciation on the capital costs of equipment and machinery, etc.), hence, discrepancies developed between the private departmental system and the account status reports, once they'd been reconciled. Keeping these private systems in sync with the central accounting system generated an appreciable level of enquiries, which absorbed appreciable staff time.

The ESDI system then, with its increased functionality, enabled a Departmental Administrator to initiate the extraction of account status and financial report information from the central accounting system. The Departmental Administrator would select the destination, Management Information Services (processing the requests for information on behalf of the Management Accounting section), select the Financial Transaction Request Form, fill in the appropriate form details, and transmit the form.

The MFMD implementation consisted of two components, similar to the SFSD implementation: the *Server* software and the *Bridge* software. The *Server* software, being the actual ESDI software, controlled the processing and handling of the multiple electronic forms and was implemented as an integral part of the Computer Centre's electronic mail system. The *Bridge* software drove the dedicated personal computer, which we referred to as the "Bridge" machine, which provided the physical link between the Campus Network and the main Registry computer. The *Bridge* software would periodically interrogate a dedicated mailbox for electronic mail containing a Form Data File (created as a result of the successful processing of a form by the *Server* software). Any requests for information would result in the determining of the form type, the extraction of the required information from the Registry computer, and the return of the results to the originator.

The components of the MFMD implementation, the *Server* software (including the Financial Transaction Request Form) and the *Bridge* software, are described in more detail in the following sections. These components were obviously modified to provide for the increased functionality of the menu-based interface and the multiple forms, however, the MFMD implementation also provided an opportunity to improve the *Bridge* software,

rectifying a number of inadequacies that were designed into the original implementation. The relationship between these components is shown pictorially in Figure 15.

8.1. The Server Software.

The *Server* software, in conjunction with the Computer Centre's electronic mail system, NETMAIL, provided a complete ESDI system. The *Server* software controlled the processing and handling of the multiple electronic forms, while the electronic mail system provided both comprehensive coverage of the Campus, through the Campus Network, and a store-and-forward delivery system.

The *Server* software was implemented as an integral part of the Computer Centre's electronic mail system, effectively providing users of NETMAIL with a choice of two editors: one for the sending of unstructured mail, and one for the sending of structured form mail. To send unstructured mail, the user would select the standard electronic mail option, enter the address of the recipient, compose the message, and enter the subject, transmitting the mail when complete. To send structured form mail, the user would select the electronic form option, select the destination, select an electronic form, the electronic form would be displayed, and the user would fill in the form details, transmitting the form when complete.

To use the electronic form facility a user had simply to select the electronic form option, which would direct NETMAIL to run the *Server* software. The *Server* software was executed as a child process under NETMAIL, with a single parameter, that of a unique file name. NETMAIL expected two things from this process. Firstly, if the processing was successful then the results (a Form Data File) would be found in a flat file, the name of this file being that specified by NETMAIL by way of the parameter string. Secondly, if this file existed, that the first line of the file would contain the address that NETMAIL would send the results to. NETMAIL would encase the results in an electronic mail envelope, placing the destination address, sender address, and a standard subject line in the header, and would place the envelope in the electronic mail system, subsequently delivering it to the destination.

The *Server* software provided increased functionality over the SFSD implementation, which was very elementary, by providing a facility that offered a choice of electronic forms and a choice of destinations (the main program listing for the *Server* software has been included as Appendix G)[5]. The facility was provided by Blaise Computing's Menu Management Library [66]. The Blaise Menu Management Library provided tools that enabled the efficient implementation of moving highlight bar menus. The menus display a series of selectable items within a window and allow a user to select

[5]A complete program listing, including units, is provided on the accompanying diskette.

an item either by moving the highlight bar and pressing return or by pressing a predefined key. The highlight bar is typically moved with the cursor control keys, or by pressing a highlighted letter (often the first letter) of the item. These menus have titles, borders, and different display attributes, and can be virtual, so that more items can be provided than will fit in the window viewport (automatically scrolling in the viewport as the highlight bar is moved).

Using the Blaise Menu Management Library tools, the *Server* software first created the menus of available destinations and electronic forms. These menus were compiled each time the program was executed, providing flexibility for the adding and/or deleting of available destinations and electronic forms. These menus were compiled from a file containing the menu items (that is, the possible selections) for each menu (MENUS). As shown in Figure 22, each menu definition contained the internal menu name and the number of lines within the menu definition (lines 1 and 7), the displayed menu title (lines 2 and 8), and pairs of strings, on consecutive lines, containing the displayed menu string and the returned menu string (lines 3 and 4, and lines 9 and 10, for example). If a menu item is selected, as indicated by the displayed menu string, the following string is returned to the application program. This enabled the application program to determine the name of the Form Definition File associated with an electronic form menu item and the address associated with a destination menu item, while enabling more meaningful descriptions to be placed in the menus. The *Server* software searched for the Menu Definition File (MENUS) in the Network EDI directory, P:\EDI. This general Menu Definition File contained menu item definitions for the available Form Definition Files and the destinations thought appropriate by Management Information Services (the current implementation). Menus could be configured for individuals by tailoring this Menu Definition File to suit. For instance, if a user should only have access to a subset of Form Definition Files and/or destinations, a customised Menu Definition File can be placed in their personal network directory, by the System Administrator, with an environment variable directing the *Server*

```
1.   !FORMS#5
2.   " Available Forms "
3.   Class Roll Request Form
4.   CLASROLL.FDF
5.   Financial Report
6.   FINTRANS.FDF
7.   !DESTS#9
8.   " Destination "
9.   Management Information Services
10.  MISEDI
11.  Management Information Services (Guest Suite)
12.  MISGuest
13.  Paul Fraser (Production Technology)
14.  P.J.Fraser
15.  Stephen Quinn (MIS)
16.  S.K.Quinn
```

Figure 22. An exampleof a Menu Definition File (MENUS).

software to the appropriate directory. The *Server* software would still obtain the Form Definition Files from the Network EDI directory, however the user would only "see" a subset of these, based on the menus that were compiled from the Menu Definition File in their network directory. This implementation could be further customised by building a menu tree that accommodated instances of electronic forms, providing users with the choice of selecting a form containing default values for one or more fields in the form. These Form Data Files and/or Form Definition Files containing default values could be placed in the user's network directory, along with any customised Form Definition Files they may require for private use. The *Server* software would direct the Form Manipulation Record to search the user's network directory for these files and the Network EDI directory for any standard Form Definition Files (this feature has not yet been implemented).

Having obtained the Menu Definition File, the menu structures were compiled. Using the Blaise Menu Management Library tools, a menu is defined and controlled by a set of related structures allocated on the heap. The structures (or descriptors) determine the menu items (that is, the possible selections), the window and its viewport in which the items appear, the nature of the highlight bar if any, the relationship to the other menus, and the accepted keystrokes and their actions.

The Menu Descriptor is the basic structure used to define a menu and its features.

```
type
  _MenuPtr = ^_Menu;
  _Menu    = record                { Menu descriptor                }
               _MSig      : char;       { Signature byte ('M')      }
               _MWinPtr   : _WindowPtr; { Associated window         }
               _MId       : pointer;    { Menu identification       }
               _MItemNum  : byte;       { Number of menu items      }
               _MItemsPtr : _ItemPtr;   { Root of the item list     }
               _MITailPtr : _ItemPtr;   { Last item in the list     }
               _MLastPtr  : _ItemPtr;   { Last position of bar      }
               _MLastKey  : _KeyPtr;    { Last key pressed          }
               _MKeysPtr  : _KeyPtr;    { Root of menu keys list    }
               _MKeyProc  : pointer;    { Key control procedure     }
               _MParent   : _MenuPtr;   { Calling menu              }
               _BarFore   : byte;       { Highlight bar attribute   }
               _BarBack   : byte;
               _LDescFore : byte;       { Lotus-type description    }
               _LDescBack : byte;       { attributes                }
               _ProtFore  : byte;       { Attribute of protected    }
               _ProtBack  : byte;       { items                     }
             end;
```

Pointers in the `_Menu` structure are used to address other structures that define the menu. The window in which the menu is displayed is recorded in `_MWinPtr`. The head of the doubly-linked list of items is recorded in `_MItemsPtr`, and the last item is pointed to by `_MITialPtr`. Whenever the menu is exited, `_MLastPtr` records the pointer to the item at which the highlight bar rests. `_MLastKey` contains a pointer to the key descriptor corresponding to the last key pressed. `_MKeysPtr` points to the head of the list of

keystroke definitions for the menu. Finally, the fields _BarFore and _BarBack define the video attributes of the highlight bar. To enable easier manipulation of the menus by the *Server* software two Menu Descriptor fields were added to the _Menu structure, as defined by Blaise. _MId contains a pointer to a menu identification string, used to reference the menu, and _MItemNum records the number of menu items.

Menu items are defined by menu Item Descriptors declared as _Item. The list of menu items (its head is recorded in the Menu Descriptor field _MItemsPtr) determines the available menu selections.

```
type
  _ItemPtr = ^_Item;
  _Item    = record                { Menu item descriptor           }
               _ISig       : char; { Signature byte ('I')           }
               _Len        : byte; { Length of the menu item        }
               _LDescCol   : byte; { Coordinates of optional long   }
               _LDescRow   : byte; {   (Lotus-style) description    }
               _LDescLen   : byte; { Length of long description     }
               _HiChPos    : byte; { Position of highlighted char   }
               _HiChFore   : byte; { Attribute of highlighted       }
               _HiChBack   : byte; {   character in menu item.      }
               _Protect    : boolean;  { Is the item protected?     }
               _Active     : boolean;  { Can the item be accessed?  }
               _ItemStrPtr : pointer;  { Pointer to item string     }
               _ItemRetPtr : pointer;  { Pointer to return string   }
               _LDescPtr   : pointer;  { Pointer to Lotus string    }
               _ItemOption : _ChildOption;   { Child menu or proc   }
               _ChildPtr   : pointer;  { Pointer to child           }
               _RetItemNum : word;     { Return item number         }
               _PrevItem   : _ItemPtr; { Pointer to previous item   }
               _NextItem   : _ItemPtr; { Pointer to next item       }
               case integer of
                 0  : (_ColRow   : word); { Position of item        }
                 1  : (_Row      : byte;  { within the menu window  }
                       _Col      : byte)
             end;
```

The fields _Len and _ItemStrPtr record the length of the item text and a pointer to the text. Items are displayed using the video attributes of the menu window. Each item may have a highlighted character. One character in the displayed text string may have video attributes defined independently of the menu window attributes, and the values are recorded in _HiChFore and _HiChBack. The character is defined by its position in the item string (_HiChPos). If _HiChPos is zero no character is highlighted. This feature is used in conjunction with the keystroke definitions to enable the selection of items by pressing a key corresponding to the highlighted letter. When an item is selected, the _RetItemNum field can be inspected and used for reference. The Blaise Menu Management Library tools are very powerful in the flexibility they provide. The tools enable the "linking" of child procedures or child menus to the menu items. The field _ItemOption specifies which is linked, and _ChildPtr is a pointer to either a child menu that is read or a procedure that is invoked when the item is selected. Not wanting to read another menu or invoke a procedure, an Item Descriptor field was added to the _Item structure, as defined by Blaise. _ItemRetPtr contained a pointer to the return

string of the item, that is the string that will be used by the *Server* software if the item is selected.

The _Key structure defines the keystrokes that control the highlight bar movement and make item selections.

```
_KeyPtr = ^_Key;
_Key    = record
            _KSig     : char;        { Signature byte ('K')     }
            _Action   : byte;        { Menu action              }
            _Motion   : _MenuMotion; { Highlight bar movement   }
            _KeyItem  : _ItemPtr;    { Associated item          }
            _NextKey  : _KeyPtr;     { Next key in the list     }
            _KChSc    : _KeySeq;     { Character and scan code  }
          end;
```

The Menu Descriptor field _MKeysPtr contains the address of the first _Key record in the linked list. All menus use a common initial list of default _Key definitions. The effect of a key sequence depends on both its associated menu action and menu movement. A menu movement describes how the highlight bar is moved, or in the case of a virtual menu, how the menu is moved within the window viewport. The menu action determines if an item is selected and the state of the menu and its highlight bar after the menu is transmitted.

The *Server* software uses the Blaise Menu Management Library function __BuildMnu to create all related Menu Descriptors for a menu, using the variable _MenuDefaults to provide the default menu characteristics. The unit initialisation code (__InitMnu) assigns all elements of the _MenuDefaults record structure, which are defined through the InitMenuDefs procedure. Note that video attributes are assigned based on the display adapter and video mode at the time the program begins. For each menu defined within the Menu Definition File, the variables for the __BuildMnu function call are assigned and the function is called, returning a pointer to the created _Menu structure, or NIL if the menu cannot be constructed.

```
function  __BuildMnu(X1,Y1,X2,Y2 : byte;
                           Title : string;
                           MDefs : _MenuDefs;
                          MIdPtr,
                         TextPtr,
                       ActionPtr : pointer;
                       TextItems : byte) : _MenuPtr;
```

The variables X1, Y1, X2, Y2 define the column and row of the upper left and lower right hand corners of the viewport in which the menu is displayed. Title is used to specify the title of the menu, which is displayed on the border of the window viewport. MDefs is the record structure that defines the characteristics of the menu (this was defined by the variable _MenuDefaults in the InitMenuDefs procedure, the significance of which is described below). The MIdPtr variable is a pointer to a string containing the menu

identification. The variables TextPtr and ActionPtr are pointers to two separate arrays of strings, the TextPtr referenced array containing the menu item text strings, and the ActionPtr referenced array containing the strings that are returned if the corresponding TextPtr item is selected. Finally, TextItems contains the number of menu items to display. The __BuildMnu function installs the items in the order that they are found in the array referenced by TextPtr. Using the _MenuDefaults defined in the InitMenuDefs procedure, __BuildMnu creates menus that:

- have a vertical array of items;
- have a single line border;
- have a title centred at the top of the menu;
- have the first character of each menu item highlighted;
- are transmitted with Alt-highlight character, and;
- are virtual (that is, have more items than fit within the window viewport, and have the items scroll in the viewport).

The variables MIdPtr and ActionPtr have been added to the __BuildMnu function, as defined by Blaise, for the purposes of this implementation.

Once the menus have been created, the menu of destinations (_MId^ = "DESTS") is examined. Internally, the *Server* software refers to each menu by the name given in the Menu Definition File, its menu identification string. As each menu is created through __BuildMnu, a pointer to its Menu Descriptor is returned, which is stored in an array of _MenuPtr. To access a Menu Descriptor within this array, each Menu Descriptor is checked to determine whether or not the menu identification string matches that being sought. If the "DESTS" menu identification string cannot be found in any of the Menu Descriptors, the menu of destinations does not exist (there was no definition for this menu within the Menu Definition File) and, hence, the destination is regarded as being hardwired and any form will be sent to Management Information Services (MISEDI). If the menu of destinations exists, the number of menu items is examined (_MItemNum). If there is only one menu item, the return string of the menu item (_ItemRetPtr^) is obtained from the Item Descriptor, however, if there is more than one menu item, the menu of destinations is displayed and the user is prompted to select a destination.

The displaying of menus is handled through the ShowMenu procedure, which uses the Blaise Menu Management Library tools to display the menu (__DispMnu), read the menu (__ReadMnu), and remove the menu (__RemMnu), returning a boolean variable indicating whether or not the user aborted the menu and, if not aborted, the return string of the last menu item highlighted. The function __DispMnu takes a pointer to a Menu

Descriptor and displays the menu in the window viewport specified by the window pointer _MWinPtr. The function __ReadMnu operates the menu in response to user input:

```
function __ReadMnu(MenuPtr : _MenuPtr;
                   FirstItem : _ItemPtr;
                     Options : word;
                     var Key : _Keys) : _ItemPtr;
```

The keystrokes are interpreted and can move the highlight bar, scroll the menu (if it is virtual), transmit the menu with a selection, or return if no selection is made (that is, the menu is aborted). __ReadMnu returns a pointer to the menu item selected, or returns NIL if the selection is aborted. The last keystroke pressed is returned in the _MLastKey field of the Menu Descriptor. A pointer to the last item on which the highlight bar rested is returned in the _MLastPtr field of the Menu Descriptor. Then, finally, the function __RemMnu removes the menu pointed to by the Menu Descriptor pointer from the current display page.

Unless the menu of destinations was displayed (multiple destinations existed) and was aborted by the user, in which case the *Server* software returns control to NETMAIL, a destination string exists. This destination string is written to the first line of the open data file, whose name was specified, by way of a parameter string, by NETMAIL. The menu of electronic forms is then displayed and the user is prompted to select a form type. Once the user has selected a form, the form is processed by the forms engine, and if the user initiates an accept sequence after filling in the form details, the Form Data File, containing the field values from the electronic form, is written to the open data file. However, if the user initiates an escape sequence during the processing of the form, the data file is closed and erased.

8.1.1. The Financial Transaction Request Form.

The aim of this stage was to increase the functionality of the ESDI system, implemented in stage one, by making a facility available that would offer a choice of electronic forms and a choice of destinations. It was specified that, to demonstrate the increased functionality of the ESDI system, the Financial Transaction Request Form would be implemented, enabling a Departmental Administrator to request account status and financial report information from the main Registry computer. The Financial Transaction Request Form had to provide for four items of information: the account number, a date range (consisting of two dates), and the authorisation (the Form Definition File for the Financial Transaction Request Form is shown in Figure 23). These fields would enable a Departmental Administrator to obtain the account status and financial report information, as recorded in the central accounting system, for a given account number, over a specified period.

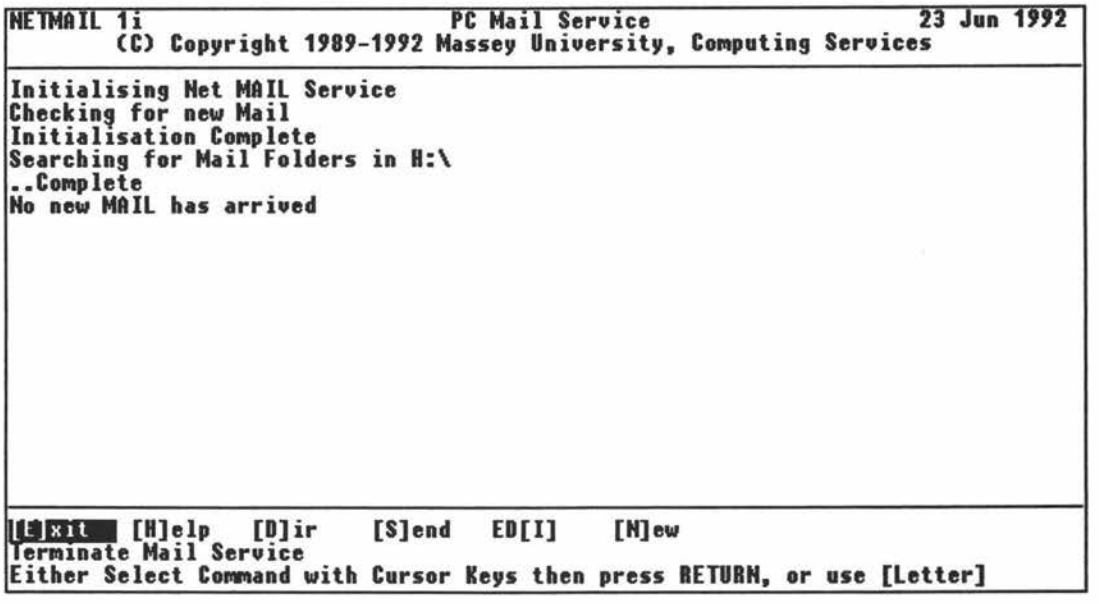
```
hFinancial Report
h
h SCREEN
*
*  Authorisation:     \________\    Account Number:  \^1_________\
=  Password [] req
=  Integer req
*
*  Transaction Date Range:
*
*       From :        \__^/__^/__\    To   :        \__^/__^/__\
=  Date req
=  Date req
*
```

Figure 23. The Financial Transaction Request Form Form Definition File.

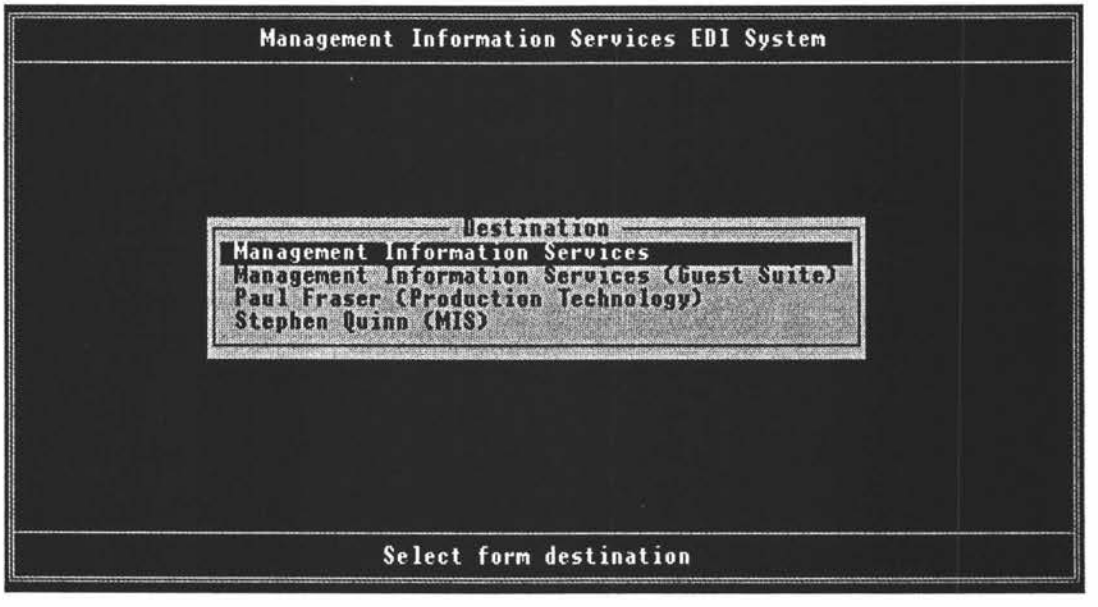
To use the Financial Transaction Request Form facility to obtain account status and financial report information any member of staff registered as a network user had simply to select the electronic form option from the Computer Centre's electronic mail system.

The following steps detail how account status and financial report information can be obtained for a given account number.

1. Once logged on to the Campus Network, invoke NETMAIL, the electronic mail service.



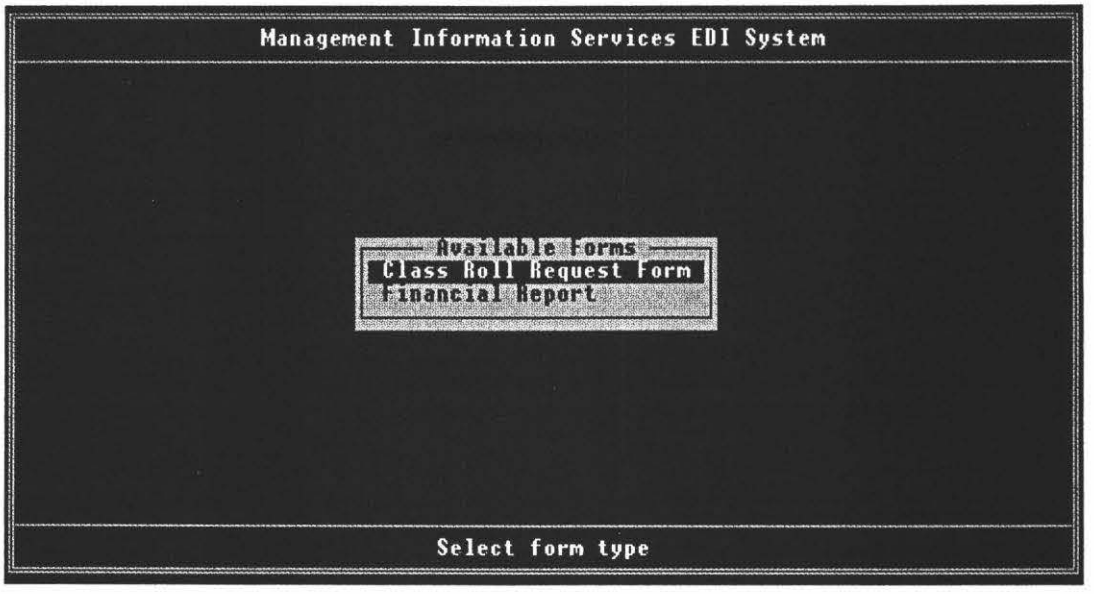
2. Select the ED[I] option to enter the ESDI system.



3. If a menu of destinations is displayed, select Management Information Services. Either highlight the selection by moving the highlight bar (using the up and down arrow keys, Home, or End) or by pressing the highlighted character of the selection, and press Enter or press Alt and the highlighted character of the selection.

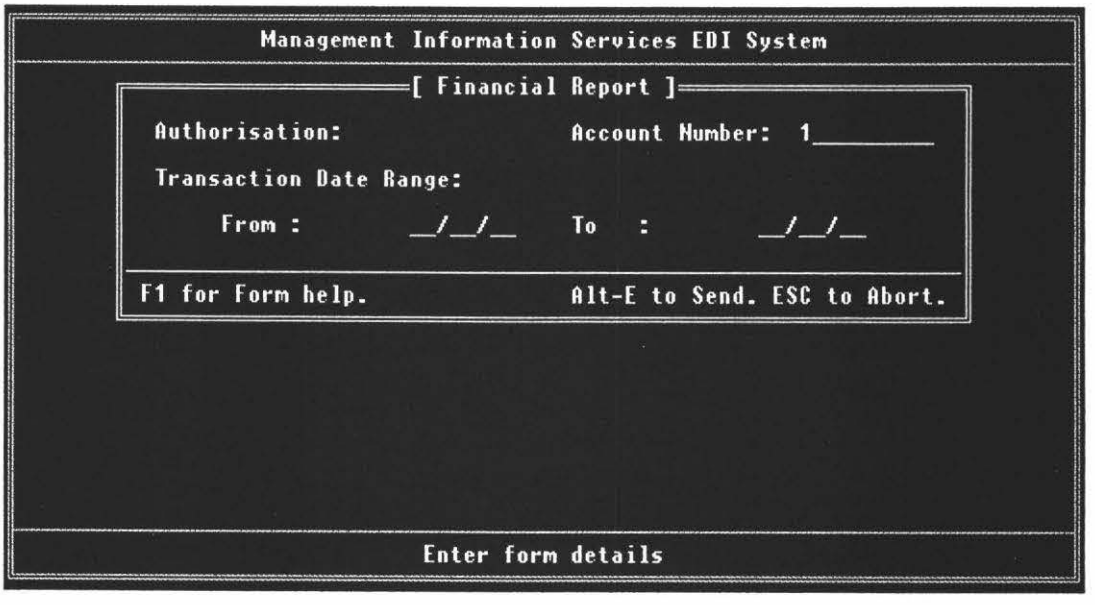
If there is only a single destination available, a menu of destinations will not be displayed, instead the destination is handled by the ESDI system.

A menu of available forms is displayed.



4. Select the Financial Report Form. Either highlight the selection by moving the highlight bar (using the up and down arrow keys, Home, or End) or by pressing the highlighted character of the selection, and press Enter or press Alt and the highlighted character of the selection.

The Financial Transaction Request Form facility is obtained.



5. Enter the authorisation. The authorisation will not be displayed as you enter it. Press Enter to confirm the authorisation.

To use the Financial Transaction Request Form facility a member of staff will need to obtain authorisation, in the form of a password, from the Head of Department responsible for the course. The authorisation confirms that a member of staff is entitled to the information being requested.

6. Enter the account number. Press `Enter`.

   An account number consists of the first 10 digits only, which precludes the specification of subaccounts. Only account numbers that begin with "1", i.e. company accounts, can be requested.

7. Enter the "From" date for the transaction range. Press `Enter`.

   The "From" date represents the start date for which transactions are to be reported. The date format is of the form DD/MM/YY.

8. Enter the "To" date for the transaction range. Press `Enter`.

   The "To" date represents the end date for which transactions are to be reported. The date format is of the form DD/MM/YY.

9. Any editing corrections can be made to the fields of the form by using the up and down arrow keys to locate the field to be corrected, and then re-entering the field contents.

10. Either press `Alt-E` to activate the request for the financial report or press `Esc` to cancel the request.

    After the form has been processed the user is returned to NETMAIL.

8.2. The Bridge Software.

The Massey University Campus Network, to which registered members of staff are connected, and the Registry Network, where the student records system and central accounting system are maintained, are operated as two separate networks for reasons of security. In order to be able to extract class roll information from the student records system, or account status and financial report information from the central accounting system, a dedicated personal computer, referred to as the "Bridge" machine, was used to provide a physical link between the two networks. This Bridge machine, which had a network connection to the Massey University Campus Network and a serial link to the Registry systems, was controlled by the *Bridge* software.

The *Bridge* software in the SFSD implementation was implemented as a recursive DOS batch file (BRIDGE.BAT), calling itself upon completion. A flow diagram for the *Bridge* batch file was shown in Figure 18. This batch file would periodically interrogate the ESDI system's dedicated mailbox for electronic mail containing a Form Data File (created as a result of the successful processing of Class Roll Request Forms by the *Server* software). Any requests for class roll information would result in the extraction of the information from the student records system and the return of the results to the originator of the request.

Since the requests for information from the ESDI system were being delivered through the Campus Network, the batch file had to somehow interrogate the mailbox, to establish whether any mail was waiting to be processed, and, if so, retrieve the items of mail from the electronic mail system. The batch file accomplished this by running NETMAIL in conjunction with a utility called KeyPlayer. KeyPlayer took keystrokes from a job file and placed them in the keyboard buffer when NETMAIL was executed, allowing NETMAIL to be effectively used from a DOS batch file, even though it was an interactive program.

An item of mail, having been saved to a file, then needed to be processed. Ideally, this would have involved a single program that verified that the mail contained a Form Data File, extracted the required field information from the Form Data File, verified that the request for information was authorised, processed the database script with the appropriate parameters (a script that extracted the required information from the remote student database), and prepared the results for the subsequent delivery to the originator of the request. However, because of difficulties experienced trying to process a database script as a child process from a parent application, the *Bridge* batch file evolved with some eight command lines where, ideally, there should have been only one. Not only did this

result in a larger batch file, but also one that was more complex, with conditional statements and process control flags being used to control the processing of the batch file.

The MFMD implementation provided an opportunity to improve the *Bridge* software, rectifying the inadequacies that were designed into the original implementation. The specifications for the new *Bridge* software were straightforward: it should exist as a single robust application program; it should interface directly with the electronic mail system, allowing waiting mail to be directly accessed from the mailbox and return mail to be placed directly into the system for subsequent delivery, thereby avoiding the use of NETMAIL and KeyPlayer (which was, at the worst of times, unpredictable); and, it should be flexible, allowing for the ease of integration of any future form processing.

With these specifications in mind, the *Bridge* software was designed as a network-dependant application program, interfacing directly with the Computer Centre's electronic mail system (the main program listing for the *Bridge* software has been included as Appendix H)[6]. The electronic mail system was provided by a dedicated host computer on the Campus Network, a DEC 3100 (cc-server4) running Ultrix, which also provided the electronic mailboxes for registered users. The services of this host Unix machine were provided to the Network through a restricted DOS file system mounted off cc-server4, therefore, by accessing this file system, the *Bridge* software was able to directly manipulate the electronic mail system.

In order for the *Bridge* software to be able to directly manipulate the electronic mail system, certain Network environment information was required. This information was obtained from the `$NETINFO` device driver during initialisation of the program. As part of the initialisation of the *Bridge* software, a procedure, `GetNetInfo`, was called to obtain the global network environment information from the device driver and write it to the global record `NetInfoRec`:

```
type
   NetInfoRec = record
                   Id        : array[1..8] of char; { Always NETINFO }
                   Version   : integer;             { Should be 12 }
                   TimeStamp : array[1..4] of byte; { Not in use}
                   DateStamp : array[1..4] of byte; { Not in use}
                   Status    : integer; { 0 = Virgin, 20 = NFS not available,
                                          21 = NFS available, 22 = User logged in }
                   NId       : array[1..32] of char;
                   Pad1      : array[1..32] of char;
                   UId, GId  : integer;
                   Host      : array[1..32] of char;
                   InetAddr  : array[1..4] of byte; { e.g. 130 123 3 1 }
                   Pad2      : array[1..4] of byte;
                   PrinterQ  : array[1..3,1..32] of char;
                   Auth_Map  : array[1..8] of byte;
                end;
```

[6]A complete program listing, including units, is provided on the accompanying diskette.

This procedure checked the version of DOS (DOS 3.0 or above was required to access the device driver), and then read the contents of the device driver. If the version of DOS was incorrect, the device driver could not be accessed, or if not enough bytes were read from the device driver the program was halted. Three items of information obtained from the device driver were important to the implementation: `NId` provided the Network user's identification (subsequently saved in the string variable `WhoAmI`); `InetAddr` provided the Network internet address for the particular machine (subsequently saved in the string variable `MyIPNum`); and `Status` indicated the network status of the machine.

Before looking at the MFMD *Bridge* implementation, it is first necessary to discuss the operation of the electronic mail delivery system. The Computer Centre's electronic mail delivery system was a conceptually simple system, which revolved around the restricted DOS file system mounted off the host mail system (cc-server4). This file system was accessible on the Campus Network as logical drive O:\. Any process that required the delivery services of the electronic mail system could, with sufficient privileges, place an item of mail in the mail server queue, the directory O:\PCMAILQ, which would be polled every minute by the mail server. Items of mail were placed in the mail server queue in a file unique to the process. The file name consisted of the Network internet address for the particular machine and a mail item number. Each process effectively had ten mail slots that could be utilised for the delivery of mail. The process would write an item of mail to a temporary file in the mail server queue:

```
O:\PCMAILQ\Q + MyIPNum
```

The process would then attempt to rename this file, effectively placing it in one of the mail slots. These mail slots were essentially the files:

```
O:\PCMAILQ\ + MyIPNum + "0"
                        ¦
O:\PCMAILQ\ + MyIPNum + "9"
```

The process would begin by attempting to place the item of mail in the first mail slot, and then, if unsuccessful, the second mail slot, etc. If the process was unable to rename the temporary file to one of the mail slot files it would delay a minute (in which time the mail server should have cleared the mail slots) and try again. The mail server, polling the mail server queue, would examine the header of each item of mail awaiting delivery and copy the item of mail to the recipient's electronic mailbox, subsequently clearing the mail slot. The recipient's mailbox consisted of a file in the mailbox directory, O:\NETMAIL, the file name being the recipient's Network user identification string. Each item of mail destined for a recipient would be placed in the recipient's mailbox file:

```
O:\NETMAIL\ + WhoAmI
```

Creating the mailbox file if it doesn't exist or appending to the file if it already exists (the recipient's mailbox file exists until such time as the recipient has completely processed the contents of their mailbox). In order to prevent a conflict between a recipient processing the contents of their mailbox at the same time the mail server writes or appends an item of mail to their mailbox file, or vice versa, lock files are used. The lock files, which reside in the lock file directory (O:\LOCKS), have a similar naming convention to the mailbox files:

```
O:\LOCKS\ + WhoAmI + ".MLK"
```

Any process that wishes to handle a mailbox file must first create the associated lock file. The lock file prevents the mail server from creating or appending to the mailbox file while it is in use by the process. If the process cannot create the lock file, that is, a file of the same name already exists, the mailbox file is being used by some other process (typically the mail server), in which case it will need to delay and try again.

The MFMD *Bridge* implementation, interfacing directly with the Computer Centre's electronic mail system, was implemented as a single application program. The *Bridge* program established that the user was logged in to the Campus Network, processed any items of mail on the *Bridge* machine that had been left unanswered, established the mailbox file and lock file names, and then cycled continuously, creating the lock file and checking for the existence of the mailbox file, until such time as a key is pressed. If the mailbox file existed, each item of mail found within the mailbox file was saved to a file and subsequently processed by establishing the nature of the item of mail and, if the mail was a recognised Form Data File, calling a child process that would process the request for information. The results of this request, if successful, were then passed to the mail delivery system for the subsequent delivery to the originator of the request.

The *Bridge* software could only interface with the electronic mail delivery system if the user was logged in to the Campus Network. The `Status` variable of the `NetInfoRec` record indicated whether or not the user was logged in. If the user was not logged in (`Status <> 22`), the program was halted.

Provided that the user was logged in to the Campus Network, and once any unanswered items of mail had been processed, the lock file was created and the existence of the mailbox file was checked. If the lock file (O:\LOCKS\MISEDI.MLK) could not be created (the mailbox file is being used by some other process), the *Bridge* program would delay a minute and try again. However, if the lock file could be created, the *Bridge* program opened the mailbox file (O:\NETMAIL\MISEDI), and, if the mailbox file was

opened without error, the items of mail within the mailbox were processed. If the mailbox file could not be opened (the mailbox file did not exist), the lock file would be erased, and the cycle would begin all over again.

If the mailbox file existed, each item of mail found within the mailbox file was saved to an individual file for subsequent processing. The procedure `CopyMail` took the mailbox file, which was in Unix format, and saved each item of mail to consecutively numbered files, beginning with NEWMAIL.000, in a temporary directory on the Bridge machine. Once each item of mail from the mailbox file had been saved to a separate file, the mailbox file (O:\NETMAIL\MISEDI) was closed and erased.

Each item of mail, having been saved to a separate file, was then processed, the processing being determined by the nature of the mail item and the mail item's contents. The procedure `ExamineMail`, processing each mail file in the temporary directory (TEMP\NEWMAIL.*) in turn, first determined the nature of the mail item. The mail item was searched for three consecutive form header lines, which, if found, indicated that the mail item was a Form Data File (Form Definition Files should not end up in the MISEDI mailbox). If it was established that the mail item was a Form Data File, and hence a request for information, the first header line would determine how it would be handled. The first header line, which contains the name of the Form Data File, was used to search a file containing, on consecutive lines, the name of the Form Data File and the name of the program, or handler, that processes the requests for information associated with that Form Data File. This file (FORMLIST), which effectively determines what requests for information can be processed by the Bridge machine, is shown below:

```
Class Roll Request Form
CLASROLL.EXE
Financial Report
FINTRANS.EXE
```

If the mail item does not contain a Form Data File, or it contains a Form Data File that is not recognised, i.e. the name of the Form Data File was not found in the list of handlers, it is processed by an exception handler (UNKNOWN.EXE). The handler that processes the mail item is then executed as a child process, by the *Bridge* software, with two parameters: the name of the file containing the item of mail and the name of the handler output file (TEMP\MAILOUT.DOC), which is subsequently passed to the electronic mail system for delivery.

The handlers for the Class Roll Request Form and Financial Report function much like the SFSD *Bridge* implementation. The handler obtains the electronic mail header information from the mail item, writes the response electronic mail header to the handler

output file, obtains the field data from the Form Data File contained in the mail item, checks the authorisation, and, if cleared, spawns REFLECTION with a script to obtain the requested information from the appropriate database. If the request for information was satisfied, an X-File line is added to the header and the requested information is copied to the handler output file, otherwise, if the request for information could not be cleared, the appropriate error message was written to the handler output file.

Any unknown mail that arrived at the Bridge machine was processed by an exception handler (UNKNOWN.EXE). This handler wrote an electronic mail header to the handler output file, specifying the address of the EDI System Administrator, and then copied the unknown mail item to the handler output file.

The handler output file, which contained either the requested information or an error message for the originator of the request, or unknown mail for the EDI System Administrator, was subsequently passed to the electronic mail system for delivery. The procedure `SendMail` copied the contents of the handler output file to a temporary file in the mail server queue, in Unix format. The procedure then attempted to rename the temporary file to one of the mail slot files, effectively placing it in one of the mail slots. If the procedure was unable to rename the temporary file to one of the mail slot files it would delay a minute (in which time the mail server should have cleared the mail slots) and try again. Once the mail item was placed into a mail slot, it would be delivered to its destination by the mail server. With the response mail from the Bridge machine delivered, the file containing the original mail item and the handler output file were closed and erased.

The MFMD *Bridge* implementation satisfied all the specifications for the new *Bridge* software. It existed as a single robust application program, which, depending on the nature of the mail item, called a child process that was designed to handle the processing of that type of mail. With sufficient privileges, it interfaced directly with the Computer Centre's electronic mail system, enabling waiting mail to be directly accessed from the mailbox and return mail to be placed directly into the system for subsequent delivery. Finally, it was flexible, allowing for the ease of integration of any future form processing by simply writing a handler that processes the request for information and by adding the name of the Form Data File and handler to the list of handlers (FORMLIST).

CHAPTER

9

Discussions and Conclusions

Electronic structured document interchange (ESDI) has been proposed as the last remaining technology in providing the complete infrastructure for the "paperless office." Complementing current electronic office system technology, including imaging technologies, electronic mail, and electronic data interchange, ESDI was designed to provide the electronic equivalent to structured internal documents.

In order for the paperless office to become a reality there must exist an electronic equivalent for every type of paper-based document within the office. Within a company paper-based documents can be categorised as documents having originated from external sources, internal documents, and documents destined externally.

Documents originating from sources external to the company are ideally handled by imaging technologies. The electronic equivalent of paper-based documents, produced by optical scanners, can be transmitted through networks, and can accommodate a mix of handwriting, word processing, and electronic mail. And since many systems use facsimile file formats for storing documents, they can accept facsimiles directly and send out a facsimile of any page on file.

Internal documents and documents destined externally can be further categorised as being of a structured type or of an unstructured type.

Unstructured documents, whether they are internal documents or documents destined externally, are typically sent through traditional mail systems. There are two reasons for this, firstly, any mail system must provide for universal delivery, and secondly, must provide the capability to deliver material in a variety of formats. Electronic mail, which is suited to the unstructured type of mail, can present problems in these areas. Electronic mail systems, through the process of standardisation, especially in the area of interconnection of mail systems, will eventually reach the level of maturity that it becomes a viable alternative to traditional mail systems.

Structured documents that are destined externally typically include standard business transactions such as purchase orders, purchase order acknowledgements, requests for quotations, quotes, invoices, bills of lading, and shipping notices. Such standard business transactions are increasingly being sent between trading partners through the use of electronic data interchange. Electronic data interchange, while reducing the amount of externally destined paper-based documents, also reduces the amount of paper-based documents that originate from external sources.

It was obvious that the electronic equivalent to structured internal paper-based documents remained the one stumbling block to the possible achievement of the paperless office.

Structured internal paper-based documents are often transitory and proprietary in nature and hence are unsuitable to be replaced by any previously discussed electronic office system. These documents, being transitory in nature, often being passed from person to person during processing, are not suited to image processing, which is used solely for static documents. Neither are these documents suited to electronic mail systems because of the documents structured nature. Electronic mail systems allow for the delivery of free format text, where the layout of such text is inconsequential as no further processing is required. Electronic data interchange technologies would provide the closest electronic equivalent to structured internal paper-based documents. However, electronic data interchange is a well-defined business practice, and because the documents are for internal business use and are not sent between trading partners, and because of the often proprietary nature of the documents (providing no standard format), electronic data interchange cannot be used.

Electronic structured document interchange, together with the current developments in electronic office systems, could provide the complete infrastructure for the paperless office, as well as provide many associated benefits. The benefits of ESDI include reduced costs, improved speed of delivery, improved reliability of delivery, improved security, improved internal communications, and improved intra-company flow of data. Indeed, ESDI systems have the potential to improve services by providing capabilities that are simply not possible with traditional paper-based systems.

Realising the potential benefits that could be gained from ESDI, Massey University, through initiatives from Management Information Services, proposed an ESDI implementation to facilitate the administrative functions of the University. The aim of the project was to set up at least one, and possibly more, fully functional systems utilising ESDI. The rationale behind the implementation of an administrative ESDI system was the realisation of the potential cost savings. Not only could cost savings be achieved because of the elimination of paper, printing, and postage costs, but also because of the reduced

labour costs brought about through reduced paper handling and data entry. Other rationales, including the improved speed of delivery and the improved reliability of delivery, were great incentives.

The fundamental objective of the project was to pass the electronic equivalents of paper-based documents around by means of some electronic delivery system, running on some underlying communications network. It was established that the Campus Network, which provided a comprehensive coverage of the campus, and the Computer Centre's electronic mail system (NETMAIL), which provided the necessary store-and-forward delivery system, would be used for the passing and delivery of the electronic equivalents of paper-based documents.

The electronic equivalent of internal paper-based documents had to provide an effective and efficient replacement in order to retain traditional business practices and to be preferred to the use of paper-based systems. A mechanism, available within the Department of Production Technology provided the desired electronic equivalent to paper-based documents. The mechanism was simple, yet powerful: simple in that an electronic representation of a paper-based document could easily be created (while providing the necessary content of its paper-based counterpart); and powerful; in that the mechanism can effectively provide the functions common to the processing of paper-based transactions.

The administrative ESDI system, implemented as an integral part of the Computer Centre's electronic mail system, provided a realisation of the potential benefits immediately.

Traditionally, lecturing staff and administrative staff would obtain printed class rolls from Management Information Services (MIS), which were printed and distributed to departments in bulk. These departments would photocopy the class rolls as required and staff would re-key the student records into their own private databases and spreadsheets. Departmental Administrators, on the other hand, would traditionally receive details of an account status, including financial transaction details for an account, from the Management Accounting section of Registry, which operates the central accounting function of the University. These account status reports were printed monthly and distributed to the department operating the account. The Departmental Administrators would then reconcile these account status reports with their own private departmental accounting systems. Significant problems were experienced with these traditional methods. Lecturing staff and administrative staff were faced with the near impossible task of trying to obtain timely and accurate class roll information, at a time when this information was highly variable, while dealing with the delays inherent in the system. Departmental Administrators experienced significant delays between the time that a transaction was initiated and when the transaction was processed by the central accounting system, and subsequently reported to

the department. The Departmental Administrators possessed details of orders committed from an account, while the central accounting system reported details of payments made from an account. Therefore, because of the delays inherent in the system, the reports detailing the account status would be significantly different from what they were known to be by the Departmental Administrators.

The administrative ESDI system implemented the Class Roll Request Form and the Financial Transaction Request Form, providing a facility to initiate requests for information from a system where security requirements precluded a more direct form of access. Lecturing staff and administrative staff were able to request class roll information from the Registry students records system, using the Class Roll Request Form, while Departmental Administrators were able to request account status and financial report information from the central accounting system, using the Financial Transaction Request Form. The system enabled users to receive requests for information electronically, often within a matter of minutes. Lecturing staff and administrative staff could obtain timely and accurate class roll information, at a time when they needed it most, without experiencing any administrative delays, and in a format that they could use directly with their own private databases and spreadsheets. Departmental Administrators could obtain up-to-the-minute account status and financial report information, whenever they needed it, allowing them to better reconcile their private departmental accounting systems and account for costs apportioned to them by the central accounting system, thus enabling them to keep their private departmental systems in sync with the central accounting system. No real changes in work habits were necessary, and the only visible effect was a lowered workload for some staff, often redeployed into more vital areas. The fact that these requests for information were processed automatically meant that the appreciable amounts of staff time spent servicing these requests and enquiries, estimated to be several person-months, could be better utilised, enabling staff to concentrate on more mission-critical tasks.

The administrative ESDI system provided a highly effective replacement to structured internal paper-based documents. From a MIS point of view, the use of electronic forms provided a degree of flexibility and robustness that could not be achieved with paper-based documents. The electronic forms, defined by Form Definition Files, could be easily constructed using nothing more than an ASCII text editor. The simplistic Form Definition Files enabled the necessary content of the paper-based counterpart to be provided for, while, at the same time, enabling any aspects of paper-based forms that were no longer required, but which were normally included for traditional reasons, to be omitted. The flexibility provided in using these electronic forms for business transactions enables the forms to be easily altered should the circumstances arise where the business practice changes, something that is not so easily or cheaply achieved with preprinted

forms. The electronic forms not only provide a great deal of flexibility over paper-based documents, but also a degree of robustness. One of the greatest problems with paper-based documents is the accuracy of data entry. The provision of formatted fields, set fields, and formulae and conditional fields in electronic forms go a long way in helping to alleviate this problem.

Complementing current office system technology, electronic structured document interchange may finally realise the paperless office. Any documents originating from sources external to the company would be handled by imaging technologies, while the coexistence of electronic mail, electronic data interchange, and electronic structured document interchange would provide for the remaining paper-based documents. All unstructured forms of document interchange, both internal to the company and external, could be realistically provided for by electronic mail. However developments in the standardisation of electronic mail have still to reach the maturity whereby universal delivery can be achieved. The remaining documents, structured documents, would be processed by electronic data interchange and electronic structured document interchange. Any inter-company transactions would be handled by electronic data interchange and, with the use of translation software, would be handled within the company, along with any proprietary forms, by electronic structured document interchange.

Ten years ago, it was quite fashionable to talk about the office as a paperless entity, but despite advances in computing technology and office automation and advances in office productivity we continue to process more and more paper. Why are we not enjoying the advantages of sharing information electronically? Perhaps the greatest resistance to the paperless entity is the fact that paper has been the main medium for storing information for hundreds of years. People feel secure getting hard copy. A piece of paper brings a feeling of comfort and security that is difficult to reproduce with a computer. People will always want to spread several sheets of paper across their desks for simultaneous viewing and want to make handwritten notes in the margins or scribble in a notepad while on the phone. Computers have not helped either. There is a general mistrust of electronic mail, word processing, and computer-based systems, and electronic documents, which do not contain signatures, pose questions of legality.

Will electronic structured document interchange make any difference in the quest for the paperless office? Probably not, at least not within the immediate future. As with any change, people either resist new ideas or at least take considerable time to adapt and implement them. The human element is vitally important in considering whether the paperless office is likely to become a reality. People who started using paper-based systems will probably go on using them for the rest of their working lives.

It has been heard said, "The paperless office is as likely as the paperless WC." Only time will tell if there is any truth to this statement, for, ten years ago, who would have heard of the bidet?

9.1. Future Work.

The administrative ESDI system, as it is currently implemented, has been a success, however, it still falls short on some of the functions common to the processing of paper-based transactions. The ESDI implementations in the first two stages of the project showed the use of "electronic forms," the electronic equivalents to paper-based documents, to be effective. The electronic forms could be easily constructed and could easily provide the content of its paper-based counterpart. However, the implementations did not show the electronic forms to be totally efficient. The electronic forms could be easily edited, but they could not be easily manipulated, unless you wanted to pass a form to a single destination. The examples used in the first two stages showed electronic forms being used as requests for information, that is, point-to-point manipulation of an electronic form. A more common function of paper-based document processing, if somewhat more difficult to implement electronically, is the passing of documents between multiple stations as a part of a business transaction cycle. This concept, which is referred to as transaction streams, was due to be implemented in the third and final stage of the ESDI development.

The final stage of the ESDI development was referred to as Accretive Forms. Accretion means to increase in size through external addition, which is precisely what was envisaged for the electronic forms in this stage. Consider, as an example, a department placing an order to their central stores, which requires some form of approval before the funds can be spent. Traditionally, the paper-based document may travel between several people, gathering signatures, before finally being sent to the central stores. This idea of a "travelling" form is what is referred to as a transaction stream. A transaction stream is effectively a predetermined path that a form must follow as a part of its inception, i.e., the stages of processing that a form must go through before it is regarded as a complete business transaction. In terms of an ESDI application, the central stores' order would have a predetermined path through the system where only certain fields are open at each step (the electronic equivalent to "office use only" fields on paper-based forms), the arrival of the electronic form in a particular mailbox implying signature and approval from the preceding step.

Transaction streams are a concept that allow businesses to utilise ESDI, while helping to retain as much of their traditional business practices as possible. For paper-based transactions that must travel between various people and functions to gather specific information as a part of its inception, transaction streams and electronic forms are ideally suited. Transaction streams allow the forms mechanism to automate the process of delivering a form along its predetermined path, passing it on to the next stage in the path

after the form has been processed sufficiently. Once at the end of the transaction stream, when complete, the form details can be delivered to its proper destination. The concept of transaction streams could also be taken a step further, so that a form at any predefined stage in the transaction stream may initiate one or more occurrences of another form, a form that follows a separate predefined transaction stream.

Future work would see the implementation of these transaction streams. The transaction streams would need to be implemented as an integral part of the Form Definition File. This allows the system considerable flexibility. The same form could have two instances, each with unique transaction streams, and so could be used for two totally different purposes. The implementation of these transaction streams would require the creation of a stream language and parser. The stream language would determine the path of the form and the processing that takes place at each stage, i.e., what fields are accessible, what fields can be viewed, what criterion determines the acceptance of the form, what action takes place if the form is aborted, etc. The parser, which interprets the stream language, would be implemented as a part of the forms engine, controlling the partial processing of the form at each stage.

With the implementation of transaction streams, ESDI would have the potential to provide the complete electronic equivalent to the handling and processing of internal paper-based documents, thus enabling businesses to take full advantage of the benefits of electronic processing and delivery, while retaining their traditional business practices.

Other issues that should be considered as a part of future work include the origin/occurrence of forms, electronic form carbon-copy, security issues, unstructured mail options, the electronic delivery system, and multi-platform availability.

The origin/occurrence of forms is an issue that will result in the almost certain need for a System Administrator. Some of the issues that the System Administrator will need to address include: who can create a form; who can modify existing forms; where forms reside; who can access what forms, etc. Inevitably, to make the System Administrator's task easier, an editor will need to be developed and made generally available for the creation of Form Definition Files.

A common feature of business forms is the use of carbon-copy forms to provide multiple copies. Some issues surrounding carbon-copy forms include: whether or not carbon-copies of electronic forms are required, it may be sufficient for the original form to be globally accessible; how carbon-copies will be administered if they are required; whether carbon-copies need to be updated if the original form changes; what happens to carbon-copies if the original form is trashed, etc.

As the use of ESDI evolves, the need to reassess the controls over the existing environment and application becomes more apparent, and the additional risks, which may

derive from the implementation of ESDI, because of the changes to the systems, procedures, and operations, should be addressed. Of particular importance to the on-campus ESDI implementation are the issues of audit trail and roll back. A comprehensive audit trail should be implemented, providing information such as a form's creation and accretion history, send date/time, receive date/time, etc., while roll back mechanisms are essential in case of a failure with the server or network, providing consideration to the aspects of recovery after failure.

In order for both electronic mail and electronic structured document interchange to co-exist in the paperless office, they should both be readily accessible, preferably via the same mechanism. It makes sense that electronic mail be a facility offered through an ESDI application, enabling users to send and receive electronic mail while performing their usual business function through the use of ESDI.

Issues in the use of the electronic delivery system also need to be addressed. The current ESDI implementation exists as an integral part of the Computer Centre's electronic mail system, limiting its availability to personal computers connected to the Campus Network. The ESDI system is too network-dependent and needs to be made available to other platforms (in particular the Apple Macintosh, which are used by most administrative staff). Different networks, providing different types of delivery systems and protocols, need to be addressed, with the expectation that only the delivery of a message to a specified address can be guaranteed, all other functions should be addressed by the applications software. In order to make the ESDI system more portable, in terms of availability on the networks and platforms, the use of the OSI model will inevitably need to be taken into consideration.

Appendices

APPENDIX

A

The FDEF Unit Interface

```
unit Unt_fdef;

interface

uses Unit_Sup,Unit_Str,Unit_Scn,Unit_Win,Unit_Key,Unit_Edt;

type
   KeyWrd  = string[7];
   DaysType = array[1..12] of byte;

const
   MAXFORMLEN = 400;
   DAYSTABLE : DaysType = (31,28,31,30,31,30,31,31,30,31,30,31);
   SIGN     = ['+','-'];
   NUMERIC  = ['0'..'9'];
   ALPHA    = ['A'..'Z',' ','a'..'z'];
   ALNUM    = ALPHA+NUMERIC;
   ASCII    = [chr(1)..chr(254)];
   REQUIRED = 1;
   DISPLAY  = 2;
   REEDIT   = 4;
   RIGHTJ   = 8;
   NOPRINT  = 16;
   UPPCASE  = 32;
   LOOKUP   = 64;
   CETFLD   = 128;
   FORMULA  = 256;
   CONDIT   = 512;
   INVSBLE  = 8192;
   VALID    = 16384;

{  types of Field }

   TEXTUAL  = 1;
   NUMBER   = 2;
   DATE     = 3;
   NTEGER   = 4;
   DOLLAR   = 5;
   REALNUM  = 6;
   TIME     = 7;
   MEMO     = 8;
   PAUSE    = 21;
   PASSWORD = 23;

{  where to find input/output }

   FDFILE     = 0;
   DATFILE    = 1;
   OPNFDFILE  = 2;
   OPNDATFILE = 3;
   FDFINMEM   = 4;
   DATINMEM   = 5;
   NONE       = 6;

   KEYWORDS : array[0..13] of KeyWrd = ('ABS','AVG','CASE','COUNT','IF',
                                        'INT','MIN','MAX','TODAY','DAY',
                                        'MONTH','YEAR','WEEKDAY','DAYS');
   GLOBALWRDS: array[0..9] of KeyWrd = ('SCREEN','DATA','QUOTES','NOFF',
                                        'FORMLEN','PAUSE','UNDERL','LEFT',
                                        'TOP','FDATA');
   MAXPRNTCMDS = 1;
   WINTOP : INTEGER = 2;        {Leave n blank lines at top of screen}
   WINBOT : INTEGER = 2;        {leave n blank lines at bottom}
   LITTORALS : set of ' '..'~'=['[',']','(',')','$','|','/','.',',',':','-'];
   _white = 15;

type
   ResultType = (Accept,Abort,Return);
   MssgeType = string[30];
   LineStrType = string[128];
   NameStr = string[10];
   PrntStrType = string[8];
   LabelStruct = record
      Name     : NameStr;
      Offset   : integer;
   end;
```

```
   FieldStruct = record
      FormLine : integer;          {Line # within Form of this field}
      Col      : byte;             {Column # "     "       "     "  }
      FldOffset: integer;          {Field Offset within Fd array}
      Lngth    : byte;             {Field length}
      DataType : byte;             {Field data type}
      Attrib   : integer;          {attribute mask for this field}
      ForOffset: integer;          {Offset within Fd array of formula}
      Name     : NameStr;          {Copy of field name}
      Value    : real;             {Field value if applicable}
      Known    : boolean;          {Flag for formula calc}
   end;
   FormDef = array[0..14000] of char;
   Fields = array[0..800] of FieldStruct;
   FormDefPtr = ^FormDef;
   FieldPtr = ^Fields;

   {Calling data structure definitions}

   FormOps=(Create,View,Print,Release,Save,XtData);
   FdfData = array[0..MAXFORMLEN] of LineStrType;  {fdf data strings}
   FdfDataPtr = ^FdfData;

   StrArray = record                                {fdf data structure}
      DataPtr : ^FdfData;
      FirstLine,LastLine : integer;
   end;
   Format   = record
      Where : byte;                        {in what media the info is stored}
      case byte of
      0   : (Name : ^_Path);               {data in file}
      1   : (Address : ^StrArray);         {data in memory}
      2   : (FilVar : ^text);              {data in an already opened file}
   end;

   IOSpecStruct = record
      Input,Output : Format;               {Where to find and put info}
   end;

   FormStruct   = record
      Window    : _WindowPtr;        {Ptr to Blaise window struct or nil}
      FormOp    : FormOps;           {operation to be done on form}
      FirstFld  : integer;           {field to start at}
      NumFlds   : integer;           {number of fields to enter}
      FldStr    : LineStrType;       {default override and entered string}
      LstKey    : _KeySeq;           {last key entered before returning}
      KeepScn   : boolean;           {keep image of form on screen}
      AcptKyStr : KeyWrd;            {string of form accept and exit keys}
      FdfDir    : ^_Path ;           {where the form definition files are kept}
      Account   : string[40];        {User account string (not implemented)}
      IOSpec    : IOSpecStruct;      {specifications of where and how data}
   end;                                 {may be found}

var
   InForm,OutForm : text;
   FieldDesc   : FieldPtr;
   Fd          : FormDefPtr;
   Line        : array[0..255] of integer;
   ColonArray  : array[0..31] of LabelStruct;
   NextLine,MaxFormLines,MaxOffset,MaxFlds : integer;
   NumPrntCmds,WinX,WinY,MaxFormCols,WinRows,WinCols : byte;
   PrntCmd : array[0..MAXPRNTCMDS] of PrntStrType;
   PrntStr : array[0..MAXPRNTCMDS] of PrntStrType;
   MssgWin,FormWin,ErrWin : _WindowPtr;
   Table,Trans,Mask,LineStr,ConstStr : LineStrType;
   FdfFile : _Path;
   FormOp : FormOps;
   WinT,WinB,WinR,WinL : byte;
   LeftFill,RightFill : char;
   IsHelp,OutFile : boolean;

procedure Translate(var Str : string; Decode : boolean);
procedure MakeTable;
function  FldToStr(FldIndx : integer) : LineStrType;
procedure Error(LineStr : LineStrType; Mssge : MssgeType; LineNo : integer);
function  WhatKey(ChScan : word) : _Keys;
procedure WriteStr(FormDesc : FormStruct);
```

APPENDIX

B

The FORM Unit Interface

```
unit Unt_form;
interface
uses DOS,CRT,Unit_Sup,Unit_Str,Unit_scn,Unit_Win,Unit_Fil,Unit_Edt,Unit_Key,
     Unit_Utl,Unit_Prt,Unit_Mem,Unt_Fdef,Unt_FEdt,Globals;
type
   KeyList = array[1..10] of _KeySeq;

function Process_Form(var FormDesc : FormStruct): boolean;
function LastFormKey : char;
```

APPENDIX

C

The SFSD *Bridge* Implementation: FILTER_1 Source Code

```
program FilterIn;

uses Crt, Dos;


const
    IdleTime = 600;       { Delay in seconds }
    MaxPassLength = 8;    { Maximum significant length of passwords }

type
    PasswordType = string[80];

var
    Ch : char;
    Count : integer;
    FormatReq,
    EnrolmentType : string[1];
    Course : string[5];
    CourseNumber : string[6];
    Authorisation : string[8];
    Line,
    Sender,
    Message,
    FormType : string[80];
    Found,
    Cleared,
    Finished,
    EndOfFile : boolean;
    SearchKey : PasswordType;
    Mail,
    LogFile,
    HaltFile,
    TempFile,
    BatchFile : text;



        procedure CheckAuthorisation(Key : PasswordType);

        var
            Password,
            SecureLine : PasswordType;
            Paper : string[7];
            CodeFile : text;


                function Upcase(p : PasswordType) : PasswordType;

                var
                    i : integer;


                begin
                    for i := 1 to Length(p) do
                        if p[i] in ['a'..'z'] then
                            p[i] := Chr(Ord(p[i]) - Ord('a') + Ord('A'));
                    Upcase := p;
                end;   { Upcase }


                function Trim(p : PasswordType) : PasswordType;

                var
                    i : integer;


                begin
                    while (Length(p) > 0) and (p[1] = ' ') do
                        Delete(p, 1, 1);
                    i := Length(p);
                    while (i > 0) and (p[i] = ' ') do
                    begin
                        p := Copy(p, 1, i - 1);
```

```
            i := i - 1;
        end;
        Trim := p;
    end;    { Trim }


begin
    Key := Upcase(Key);
    Found := FALSE;
    Cleared := FALSE;

    Assign(CodeFile, 'C:\CCNET\MISEDI\SECURITY.FIL');
    Reset(CodeFile);

    repeat
        Readln(Codefile, SecureLine);
        Paper := Upcase(Trim(Copy(SecureLine, 1, 6)));
        Password := Upcase(Trim(Copy(SecureLine, 8, Length(SecureLine))));
        if Length(Password) > MaxPassLength then
            Password := Copy(Password, 1, MaxPassLength);
        if Paper = Key then
            Found := TRUE;
    until Eof(CodeFile) or Found;

    if Found then
        if Password = Upcase(Trim(Authorisation)) then
            Cleared := TRUE;

    Close(CodeFile)
end;    { CheckAuthorisation }



procedure DateStamp;

var
    Year, Month, Date, DayofWeek,
    Hour, Minute, Second, Sec100 : word;
    HourString,
    MinuteString,
    SecondString : string[2];
    Day,
    MonthWord : string[4];


begin
    GetDate(Year, Month, Date, DayofWeek);
    GetTime(Hour, Minute, Second, Sec100);

    case DayofWeek of
        0 : Day := 'Sun ';
        1 : Day := 'Mon ';
        2 : Day := 'Tue ';
        3 : Day := 'Wed ';
        4 : Day := 'Thu ';
        5 : Day := 'Fri ';
        6 : Day := 'Sat '
    end;

    case Month of
        1 : MonthWord := 'Jan ';
        2 : MonthWord := 'Feb ';
        3 : MonthWord := 'Mar ';
        4 : MonthWord := 'Apr ';
        5 : MonthWord := 'May ';
        6 : MonthWord := 'Jun ';
        7 : MonthWord := 'Jul ';
        8 : MonthWord := 'Aug ';
        9 : MonthWord := 'Sep ';
       10 : MonthWord := 'Oct ';
       11 : MonthWord := 'Nov ';
       12 : MonthWord := 'Dec '
    end;
```

```
            Write(LogFile, Day, MonthWord, Date, ' ');
            if Hour <= 9 then
            begin
                Str(Hour, HourString);
                HourString := Concat('0', HourString);
                Write(LogFile, HourString, ':')
            end
            else
                Write(LogFile, Hour, ':');
            if Minute <= 9 then
            begin
                Str(Minute, MinuteString);
                MinuteString := Concat('0', MinuteString);
                Write(LogFile, MinuteString, ':')
            end
            else
                Write(LogFile, Minute, ':');
            if Second <= 9 then
            begin
                Str(Second, SecondString);
                SecondString := Concat('0', SecondString);
                Write(LogFile, SecondString, ' ')
            end
            else
                Write(LogFile, Second, ' ');
            Writeln(LogFile, Year)
        end;    { DateStamp }



begin
    Writeln('MISEDI Mail Processor');
    Writeln;

    Assign(Mail, 'C:\CCNET\MISEDI\LETTER.MW');
    {$I-}
    Reset(Mail);
    {$I+}

    if IOResult = 2 then
    begin
        Writeln('Waiting for mail ...');
        Writeln('Press any key to return to DOS');

        Finished := FALSE;
        Count := 0;
        while (not Finished) and (Count < IdleTime) do
        begin
            Delay(1000);    { Delay 1000ms }
            if KeyPressed then
            begin
                Assign(HaltFile, 'C:\CCNET\MISEDI\TEMP\HALTFLAG');
                Rewrite(HaltFile);
                Writeln(HaltFile, '0');
                Close(HaltFile);

                Ch := ReadKey;
                Finished := TRUE
            end
            else
                Inc(Count)
        end
    end
    else
    begin
        Assign(LogFile, 'C:\CCNET\MISEDI\FORMS.LOG');
        {$I-}
        Append(LogFile);
        {$I+}

        if IOResult = 2 then
            Rewrite(LogFile);
```

```
Assign(BatchFile, 'C:\CCNET\MISEDI\PROCESS.BAT');
Rewrite(BatchFile);

Assign(TempFile, 'C:\CCNET\MISEDI\TEMP\TEMP');
Rewrite(TempFile);

repeat
    Readln(Mail, Line);
    Message := Copy(Line, 1, 6);
until Message = 'From: ';

Sender := Copy(Line, 7, Length(Line) - 6);
Writeln(TempFile, Sender);

EndOfFile := FALSE;
repeat
    Readln(Mail, Line);
    if Eof(Mail) then
        EndOfFile := TRUE;
    Message := Copy(Line, 1, 1)
until (Message = 'h') or EndOfFile;

if EndOfFile then
begin
    Assign(HaltFile, 'C:\CCNET\MISEDI\TEMP\HALTFLAG');
    Rewrite(HaltFile);
    Writeln(HaltFile, '1');
    Close(HaltFile)
end
else
begin
    FormType := Copy(Line, 2, 79);

    Writeln(LogFile);
    DateStamp;
    Writeln(LogFile, Sender);

    Write(TempFile, FormType);
    if FormType = 'Class Roll Request Form' then
    begin
        for Count := 1 to 2 do
            Readln(Mail);
        Readln(Mail, CourseNumber);
        Course := Concat(Copy(CourseNumber, 1, 2), Copy(CourseNumber, 4, 3));

        Readln(Mail, EnrolmentType);
        Readln(Mail, FormatReq);
        Readln(Mail, Authorisation);

        Writeln(LogFile, 'Requested: ', FormType, ' for ', CourseNumber, ' (',
                         EnrolmentType, ')');

        SearchKey := Course + EnrolmentType;
        CheckAuthorisation(SearchKey);

        if Cleared then
        begin
            Writeln(TempFile);
            Writeln(TempFile, EnrolmentType, Course);

            Write(BatchFile, 'd:\reflectn\r1 , c:\ccnet\misedi\clasroll.cmd ');
            Writeln(BatchFile, Course, ' ', FormatReq, ' ', EnrolmentType);

            Write(LogFile, 'Script processed: ');
            DateStamp
        end
        else
            if Found then
            begin
                Writeln(LogFile, 'Error: Incorrect authorisation code.');
                Writeln(TempFile, ' for ', CourseNumber, ' (', EnrolmentType,')');
                Writeln(TempFile, 'Error: Incorrect authorisation code.');
            end
            else
            begin
                Write(LogFile, 'Error: No access to this paper. );
                Writeln(LogFile, 'Paper not set up in security file.');
```

```
                    Writeln(TempFile, ' for ', CourseNumber, ' (', EnrolmentType,')');
                    Write(TempFile, 'Error: No access to this paper. Please ');
                    Writeln(TempFile, 'contact Management Information Services.');
                end
            end
            else
            begin
                Writeln(LogFile, 'Requested: ', FormType);
                Writeln(LogFile, 'Error: Form type not identified.');
                Writeln(TempFile);
                Writeln(TempFile, 'Error: Form type not identified.')
            end
        end;

        Close(Mail);
        Close(LogFile);
        Close(TempFile);
        Close(BatchFile)
    end
end.
```

APPENDIX

D

The SFSD *Bridge* Implementation: FILTER_2 Source Code

```
program FilterOut;

uses Crt, Dos;


const
    IdleTime = 600;

var
    ErrorCode : integer;
    Sender,
    Message,
    FormType : string[80];
    ErrorMessage : boolean;
    Mail,
    OldFile,
    OldMail,
    HaltFile,
    TempFile : text;



        procedure RollRequest;

        var
            Ch : char;
            Count,
            Result : integer;
            Finished,
            Processed : boolean;
            HourString,
            MinuteString,
            SecondString : string[2];
            Day,
            MonthWord : string[4];
            Year, Month, Date, DayofWeek,
            Hour, Minute, Second, Sec100 : word;
            LogFile,
            ClassRoll,
            HaltFile,
            ResultFile : text;


        begin
            Assign(LogFile, 'C:\CCNET\MISEDI\FORMS.LOG');
            Append(LogFile);

            Assign(ResultFile, 'C:\CCNET\MISEDI\TEMP\MAILXFER.RES');
            Reset(ResultFile);

            Readln(ResultFile, Result);
            if Result = 0 then
                Processed := TRUE
            else
                Processed := FALSE;

            if Processed then
            begin
                Close(OldMail);
                Erase(OldMail);

                Assign(ClassRoll, 'C:\CCNET\MISEDI\CLASSR.DOC');
                Reset(ClassRoll);

                GetDate(Year, Month, Date, DayofWeek);
                GetTime(Hour, Minute, Second, Sec100);

                case DayofWeek of
                    0 : Day := 'Sun ';
                    1 : Day := 'Mon ';
                    2 : Day := 'Tue ';
                    3 : Day := 'Wed ';
                    4 : Day := 'Thu ';
                    5 : Day := 'Fri ';
                    6 : Day := 'Sat '
```

```
    end;

    case Month of
        1 : MonthWord := 'Jan ';
        2 : MonthWord := 'Feb ';
        3 : MonthWord := 'Mar ';
        4 : MonthWord := 'Apr ';
        5 : MonthWord := 'May ';
        6 : MonthWord := 'Jun ';
        7 : MonthWord := 'Jul ';
        8 : MonthWord := 'Aug ';
        9 : MonthWord := 'Sep ';
       10 : MonthWord := 'Oct ';
       11 : MonthWord := 'Nov ';
       12 : MonthWord := 'Dec '
    end;

    Write(LogFile, '       completed: ', Day, MonthWord, Date, ' ');
    if Hour <= 9 then
    begin
        Str(Hour, HourString);
        HourString := Concat('0', HourString);
        Write(LogFile, HourString, ':')
    end
    else
       Write(LogFile, Hour, ':');
    if Minute <= 9 then
    begin
        Str(Minute, MinuteString);
        MinuteString := Concat('0', MinuteString);
        Write(LogFile, MinuteString, ':')
    end
    else
        Write(LogFile, Minute, ':');
    if Second <= 9 then
    begin
        Str(Second, SecondString);
        SecondString := Concat('0', SecondString);
        Write(LogFile, SecondString, ' ')
    end
    else
        Write(LogFile, Second, ' ');
    Writeln(LogFile, Year);

    while not Eof(ClassRoll) do
    begin
        while not Eoln(ClassRoll) do
        begin
            Read(ClassRoll, Ch);
            Write(Mail, Ch)
        end;

        Readln(ClassRoll);
        Writeln(Mail)
    end;

    Close(ClassRoll);
    Erase(ClassRoll)
end
else
begin
    Writeln(LogFile, '   *** Error: Bridge can not be established.');

    Writeln('Bridge can not be established. Retrying ...');
    Writeln('Press any key to return to DOS');

    Finished := FALSE;
    Count := 0;
    while (not Finished) and (Count < IdleTime) do
    begin
        Delay(1000);
        if KeyPressed then
        begin
            ClrScr;
            Writeln('Returning to Network Environment.');
```

```
                    Assign(HaltFile, 'C:\CCNET\MISEDI\TEMP\HALTFLAG');
                    Rewrite(HaltFile);
                    Close(HaltFile);

                    Ch := ReadKey;
                    Finished := TRUE
                end
                else
                    Inc(Count)
            end;

            Close(OldMail)
        end;

        Close(LogFile);
        Close(ResultFile);
        Erase(ResultFile)
    end;    { RollRequest }



    procedure CreateJobFile;

    var
        Line : string[80];
        JobFile,
        MaskFile : text;

    begin
        Assign(MaskFile, 'C:\CCNET\MISEDI\TX_MASK');
        Reset(MaskFile);

        Assign(JobFile, 'C:\CCNET\MISEDI\TX_MAIL.JOB');
        Rewrite(JobFile);

        Readln(MaskFile, Line);
        while Copy(Line, 1, 1) <> 's' do
        begin
            Writeln(JobFile, Line);
            Readln(MaskFile, Line)
        end;
        Writeln(JobFile, 's', Sender, '<enter>');

        Readln(MaskFile, Line);
        while Copy(Line, 1, 9) <> 'P:\\EDI\\' do
        begin
            Writeln(JobFile, Line);
            Readln(MaskFile, Line)
        end;

        if ErrorMessage then
            Writeln(JobFile, '<enter>')
        else
            if FormType = 'Class Roll Request Form' then
                Writeln(JobFile, 'CLSROLL.MSG ', Message, '<enter>');

        while not Eof(MaskFile) do
        begin
            Readln(MaskFile, Line);
            Writeln(JobFile, Line)
        end;

        Close(JobFile);
        Close(MaskFile)
    end;    { CreateJobFile }



begin
    Writeln('MISEDI Mail Sender');
    Writeln;
```

```
    ErrorMessage := FALSE;

    Assign(HaltFile, 'C:\CCNET\MISEDI\TEMP\HALTFLAG');
    {$I-}
    Reset(HaltFile);
    {$I+}

    if IOResult = 2 then
    begin
        Assign(OldFile, 'C:\CCNET\MISEDI\PROCESS.BAT');
        Reset(OldFile);
        Close(OldFile);
        Erase(OldFile);

        Assign(Mail, 'C:\CCNET\LETTER.DOC');
        Rewrite(Mail);

        Assign(OldMail, 'C:\CCNET\MISEDI\LETTER.MW');
        Reset(OldMail);

        Assign(TempFile, 'C:\CCNET\MISEDI\TEMP\TEMP');
        Reset(TempFile);

        Readln(TempFile, Sender);
        Readln(TempFile, FormType);
        Readln(TempFile, Message);

        if Copy(Message, 1, 5) = 'Error' then
            ErrorMessage := TRUE;

        if ErrorMessage then
        begin
            Writeln(Mail, 'Requested: ', FormType);
            Writeln(Mail, Message);

            Close(OldMail);
            Erase(OldMail)
        end
        else
            if FormType = 'Class Roll Request Form' then
                RollRequest;

        CreateJobFile;

        Close(TempFile);
        Erase(TempFile);
        Close(Mail)
    end
    else
    begin
        Readln(HaltFile, ErrorCode);
        case ErrorCode of
            0 : Writeln('Returning to Network Environment.');
            1 : Writeln('Non-form mail has arrived.');
        end;

        Close(HaltFile);
    end
end.    { Main Program }
```

APPENDIX

E

Security Issues in Electronic Data Interchange (EDI)

Security Issues in Electronic Data Interchange (EDI)

Paul J. Fraser
Postgraduate Research Student

Department of Production Technology
Massey University
Palmerston North
New Zealand

1. Introduction.

Jamieson (1990) describes Electronic Data Interchange (EDI) as:

> "... the inter-company exchange of business data in a standard format via a telecommunications network between complementary computer applications."

It replaces the traditional physical exchange of paper documents such as purchase orders, invoices, or material release schedules.

EDI has been described as "a weapon for waging war on the paper mountain, the massive quantity of documents generated daily by business." However, the benefits of EDI go far beyond simply reducing the rate at which paper breeds. Further benefits of EDI include:

- reduced ordering costs due to a reduction in mail, phone, telex and facsimile costs;
- reduced ordering errors due to the elimination of transcription errors, the elimination of manual data re-entry into the recipient computer system and a reduction in paper handling;
- a reduced order cycle time, which in turn leads to a reduced investment in inventory (and possibly an implementation of a just-in-time (JIT) inventory system), a reduction in stock-outs and a reduction in interest charges on outstanding payments, and;
- increased sales productivity due to a reduction in necessary paperwork.

In general, EDI provides significant advantages for business, including increased productivity, improved customer service and a heightened ability to compete in the international marketplace. The reduction in paperwork and greater accuracy of data are obvious advantages. However, the elimination of human processing of documents changes the internal control environment so that existing internal controls may have to be modified.

Kemp (1990) states that:

> "Security is the protection of data from accidental or deliberate threats which might cause unauthorised modification, disclosure or destruction of the data; and the protection of the information system from degradation or non-availability of service.
>
> They all involve the firm in some kind of loss:
>
> 1. loss of data integrity
> 2. loss of availability of service
> 3. loss of confidentiality."

This paper, by reviewing data flow through a generalised EDI architecture, and by outlining the business risks associated with EDI, will cover the security issues of the technology by examining the associated control and auditing aspects, in terms of application and general controls.

2. The Conceptual EDI Architecture.

In general, moving a transaction through the EDI process generally involves three functions (steps) within each trading partners computer system. Powers and Carver (1990) describe these as:

- **The Communications Handler**, which transmits data between trading partners;
- **The Application System**, which processes the data to be sent to, or received from, the trading partner, and;
- **The EDI Interface**, which manipulates and routes data between the application system and the communications handler.

These functions comprise a generalised EDI architecture. The generalised architecture is generic in that it is not dependent on any specific hardware, software, communications protocol, or processing environment. Regardless of whether a transaction is sent or received by a microcomputer, minicomputer, or mainframe, transactions are processed through the EDI architecture. Consequently, the processing of those transactions through each step in the EDI process requires the appropriate control.

3. Communication Issues.

As indicated earlier, the communications handler transmits data between trading partners. There are several options available to transmit electronic documents between trading partners. The implementation generally involves adopting a network configuration, technical network standards, functional network standards and building the appropriate translation software.

3.1. Network Configurations.

There are four general configurations that can be adopted for an EDI services network.

3.1.1. Point to Point Configuration.

This configuration involves direct communication between business partner computing facilities by either leased lines or dial up services. This form of EDI services network requires that the trading partners:

- use the same standards and conventions for message formats;
- adopt compatible communications protocols;
- acquire their own translation software;
- develop a schedule for communication sessions, and/or have a dedicated line for incoming transactions, and;
- be individually and collectively responsible for network management and monitoring.

This approach is inherently inflexible. It is difficult to add another trading partner, especially if that trading partner cannot accommodate current message standards or communication protocols. Adding this partner can then result in a multiplicity of standards and protocols within one network, which can present a network management nightmare. Further, the communications session schedule must be revised and network management responsibility further distributed. Finally, the larger the network becomes, the greater the chances are that security will be breached.

3.1.2. Public Network Configuration.

This approach utilises a public telecommunications network for communication services. This permits some flexibility in that:

- the communications protocols and standards adopted by each trading partner may be different, provided that they are supported by the public network, and;
- EDI transactions may be carried out with any trading partner that has an appropriate connection to the network.

Utilising a public network also reduces the network management responsibilities of the trading partners. However, this approach still requires a schedule for communications sessions (and possibly a dedicated line for incoming transactions) and distributed network management.

3.1.3. Value Added Networks (VANs).

This method involves a public network that provides additional mailbox services. These services can be used for the sending or receiving of EDI transactions. All trading partners are still responsible for the development of message standards, but contract with a network service provider to act as the communications carrier. It is the communications carrier that provides the necessary mailbox facilities in addition to those facilities normally provided in a public network. Using this approach:

- it is unnecessary for each trading partner to provide a communications line for incoming calls;
- there is some flexibility as to when EDI transactions should be transmitted and received. For example, it could be agreed amongst trading partners that the electronic mailboxes are to be cleared at 5pm and 9am each day, and;
- network management responsibilities are reduced.

As a result, only message standards and some simple transaction scheduling need be agreed amongst the trading partners, hence providing each trading partner with a greater degree of flexibility in the way that EDI is implemented.

3.1.4. Third Party EDI Service Networks.

A third party network, often known as a clearing house system, serves as an EDI services bureau. In addition to the facilities of a value added network, trading partners need not be concerned with general details of message formats, communications protocols and

communications links. The EDI network will perform all necessary conversion between communication and message formats.

Third party EDI service networks also provide additional services, such as compliance checking on incoming data and transaction logging. These services can greatly simplify management of the EDI function.

3.2. Technical Network Standards.

Three sets of technical standards must be implemented within an EDI network: message standards, interchange standards, and communication standards.

3.2.1. Message Standards.[7]

These standards are responsible for establishing the format and content of each EDI transaction. Australia and New Zealand have adopted the ANSI X.12 standard (with minor modifications).

3.2.2. Interchange Standards.

These standards are responsible for the interchange of electronic messages across a network. The emerging standard for this operation is ANSI X.400.

ANSI X.400 is an electronic messaging standard. This standard fulfils three factors that are essential if electronic messaging systems are to be used for business transactions: they must be reliable (error free), they must be confirmable and they must be secure. The X.400 provides the following services to do this:

- **content confidentiality**: utilising asymmetric encryption to ensure that only the recipient can read the message;
- **origination authentication**: utilising a digital signature and encryption in order to verify the source of the message;
- **proof of delivery**: a user can request verification that a message has been delivered;
- **non-repudiation of origin**: a user can obtain proof that a message was delivered should the matter be disputed;

[7]Message standards are also discussed in section 3.3.2., under Functional Network Standards.

- **non-repudiation of delivery**: the transaction originator is notified when the message is read;
- **content integrity**: ensuring the message has not been tampered with in transit;
- **message sequence integrity**: ensuring that, if messages are required to be in a certain order, they remain in this order during transmission;
- **non-repudiation of submission**: a user can obtain proof that a message was submitted for another user;
- **message flow confidentiality**: utilising camouflaging data so that an observer cannot determine even the length of the message;
- **probe origin authentication**: a user can issue a probe to determine details of a messages origin, and;
- **report origin authentication**: provides a report on the message delivery and storage processes.

3.2.3. Communication Standards.

This involves adopting a set of standards related to communications speed and low level communications protocols (e.g. packet switched network using X.25).

3.3. Functional Network Standards.

Establishing the functional standards within an EDI network requires agreement between trading partners on transaction, message, security and data storage standards.

3.3.1. Transaction Standards.

This entails agreeing with trading partners what business transactions will be conducted using EDI, and any restrictions to be enforced. For example, trading partners may wish payment transactions of only up to $500 be involved within the EDI system.

3.3.2. Message Standards.

Establishing the message standards determines the form and content of each EDI transaction. These standards will dictate what data is to be transmitted as part of each EDI

transaction, and the format of the data within each transaction. In some industries in the U.S. and Europe, these standards already exist.

3.3.3. Security Standards.

Commercial transactions often involve some data that is confidential or sensitive (for example, discounts offered to specific customers). Accordingly, trading partners will have to determine how sensitive data is to be protected whilst in transit from trading partner to trading partner.

3.3.4. Data Storage Standards.

Trading partners will also have to agree upon the storage of sensitive data and the length of time that back-ups of transactions and transaction logs should be retained.

3.4. Translation Software.

Application programs never automatically generate or accept data files in standard format. This means that translation to/from standard format is a necessity in all EDI implementations. It can either be performed in-house by translation software that was either developed or purchased or it can be performed in-network (i.e. in the third party data centre). If translation is performed in-network, the usual scenario is for the third party to define a fixed field length file that it will translate to/from the standard format. If a company opts to purchase in-network translation services, it need only develop application link software to generate or accept that fixed field length format by mapping it to/from its internal application format. Not only does it take less expertise in the standard to develop this simpler mapping module, it also insulates the user from changes to the standard.

4. The EDI Architecture.

4.1. The Communications Handler.

The communications handler transmits and receives electronic documents between trading partners and/or VANs. Organisations frequently use their existing communications facilities as a communications handler for EDI transactions. The software used in the communications handler is usually designed to detect errors to protect against transmission error or interruption.

4.2. The EDI Interface.

The EDI interface usually consists of:

4.2.1. The EDI Translator.

The EDI translator translates between the standard format and a trading partners proprietary format (internal application formats). It may generate and send functional acknowledgements (standard EDI transactions that tell the trading partner that their electronic documents were received) and verify the identity of partners and check the validity of transactions by checking transmission information against a trading partner master file.

4.2.2. The Application Interface.

The application interface moves electronic transactions to, or from, the application systems and performs data mapping. Programs separate and deliver inbound transactions to the application systems and gather transactions from the application systems for outbound transmission. Controls need to be incorporated into this function to ensure that transactions are processed completely and accurately between the communications interface and the application systems.

4.3. The Application Systems.

Many companies have substantial investments in application systems that process transactions which originated as paper documents. Many companies are able to preserve

those investments by leaving the existing application systems unchanged. This is usually accomplished by designing "front end" procedures that enable EDI transactions to utilise existing applications systems. Such front end procedures become the EDI interface, and replace the manual procedures previously used to process paper transactions. Although new controls should be developed for the EDI interface, the controls for existing applications, if left unchanged, will usually remain unaffected.

5. EDI Risks.

This section briefly outlines some of the business risks associated with EDI. Management and auditors need to be aware of these risks as they provide a framework for placing the appropriate security and control mechanisms within the EDI application or the EDI environment.

Risks outlined by Wright (1989, 1990) include:

Loss of business continuity - "Going Concern problem"

Inadvertent or deliberate corruption of EDI-related applications could affect every EDI transaction entered into by an organisation, impacting customer satisfaction, supplier relations and perhaps ultimately business continuity.

Loss of confidentiality of sensitive information

Sensitive information may be accidentally or deliberately divulged on the network or in the mailbox storage system.

Increased exposure to fraud

Access to computer systems may provide an increased opportunity to change the computer records of both a single organisation and that of its trading partners.

Overpayment

Where amounts charged by or paid to suppliers are not manually reviewed before transmission, there is a risk that payments could be made for goods not received, payment amounts could be excessive or duplicate payments could occur.

Loss of audit trail

EDI eliminates the need for hard copy. There will be less paper for the auditors to check. The EDI user may not provide adequate or appropriate audit evidence, either on hard copy or on magnetic media.

Concentration of control

There may be increased reliance on computer controls where they replace manual controls, and they may not be sufficiently timely. The use of EDI with its greater reliance on computer systems concentrates control in the hands of fewer staff, increases reliance on people and increases risk.

Potential legal liability

Where liability is not clearly defined in trading partner agreements, legal liability may arise due to errors outside the control of an organisation or by its own employees.

Errors in the information system

Errors in the processing and communications systems can result in the transmission of incorrect trading information or inaccurate reporting to management.

Overcharging by third party service providers

Third party suppliers may accidentally or deliberately overcharge an organisation.

Manipulation of share values

The information available to the proprietors of third party networks may enable them to take unfair advantage of an organisation.

Not achieving anticipated cost savings

Where the anticipated cost savings from the investment in EDI is not realised by an organisation.

These risks and exposures will be addressed by examining the control and auditing issues related to EDI.[8]

[8]Although many of these risks and exposures are not directly related to security, these vulnerabilities that businesses face are diminished through the appropriate implementation of security controls.

6. Control and Auditing Issues.

Internal accounting controls are procedures that ensure accurate and reliable financial reporting as well as the safeguarding of the company assets. These controls are designed to prevent and detect financial statement inaccuracy that might affect the entity's financial position. Although internal and external auditors may have different objectives, both may find it necessary to evaluate internal accounting controls.

Internal audit is an appraisal function that examines and evaluates an organisation's activities, and encompasses the organisation's system of internal control to determine how well control objectives are met. External audit is an independent appraisal function which results in an opinion about an organisation's financial statements. The opinion is based on evidence regarding assertions made by management in the financial statements that has been evaluated by the auditor.

Generally, internal control procedures are divided into two major categories; application controls and general controls. Application controls are *specific controls over each separate computer application that ensure that only authorised data is processed completely and accurately by that system*. Application controls can be automated or manual in nature.

Since many applications controls in today's complex information processing systems are automated, it is critical to ensure that adequate controls exist over the functioning of the programs. This is the purpose of general controls, which relate to the development, implementation, maintenance, and operation of the applications systems and the security of data files and programs. They are designed to ensure that *programmed accounting and control procedures are developed properly, performed consistently, are modified only with proper authorisation, and that appropriate division of duties and responsibilities is enforced.*

7. EDI Controls.

Since electronic data interchange replaces the paper document environment with an electronic one, existing internal control procedures may need to be changed to accommodate the EDI processes. More control will now need to be exercised by the user's computer systems to replace manual controls such as sight reviews of documents. More emphasis will need to be placed on control over data transmission, with selection of document standards and appropriate protocols and use of acknowledgements becoming very important.

There will be reduced opportunities for review and authorisation. For example, organisations may pay on receipt of goods rather than on invoice. This will eliminate the current invoice authorisation review. As payment will probably be approved on the basis of sighting of a list of payments due, rather than on review of authorised documentation for each payment, more emphasis will be placed on the initial authorisation of orders and on evidence of receipt.

Transactions will be actioned more quickly, for example orders received will be processed and filled almost immediately.

This means that any controls will need to operate far more quickly to highlight errors and inconsistencies and correct these before they impact production or delivery.

Controls need to be in place to address:

- transmission errors (these should be addressed by the protocol and document standard selected and by the use of acknowledgements);
- errors in the generating applications of the message sender (the receiving organisation needs to assess whether the transmissions received are consistent with transactions normally received from that trading partner), and;
- manipulation of data, either within the sending or receiving application, on transfer to/from applications, while held in storage, or on transmission. This will include controls within the applications, procedures to address the risks associated with specific transaction types, consideration of message authentication and encryption techniques, key management procedures, etc.

Often the design of controls will require some imagination in order to implement a regime that will protect the organisation against itself and against other potential weak links in the EDI chain. This may mean increased use of expert systems as front end transaction processors and editors.

The nature of controls may not change significantly, but the way they are executed is likely to undergo major change and the computer is likely to be utilised more and more to provide this control.

All this means that organisations will place more reliance on the Electronic Data Processing (EDP) control environment. If an organisation cannot be sure that controls will be applied consistently, that its data is not adequately protected and that it cannot maintain operations, it is at risk.

At a minimum, key issues that must be considered include:

- EDI/systems access security;
- encryption;
- transaction and management audit trails;
- implementation of an information security policy;
- data communications policy, and;
- adequate user control procedures in end user areas.

The next sections address the application and general controls that are affected by EDI. Application controls will be discussed for each function in the generalised EDI architecture. General control issues unique to EDI systems will also be identified.

8. Application Controls.

Many controls used in paper document processing are not effective in the EDI environment. EDI application controls are usually automated and perform procedures that ensure transaction data is processed correctly throughout the EDI architecture. Those controls should be used throughout the EDI process, and address the completeness and accuracy of input as well as the authorisation of the input. Additional controls are required to monitor the generation of other EDI transactions such as functional acknowledgements. Note that once the data is in the application system it is subject to the usual controls applied to data processed by that application system.

8.1. Completeness of Input.

Completeness of input controls ensure that all transactions are input and accepted for processing *only* once. Techniques to accomplish this typically include batch totalling, sequence numbering of electronic documents, and one for checking against a control file (functional acknowledgements processing).

EDI completeness controls should verify that all transactions received from a trading partner are input properly and passed to the appropriate system once and only once. Control techniques may be applied to ensure completeness of input through each function of the EDI processing.

Controls should exist in the communications interface to ensure that all EDI documents are transmitted completely between trading partner systems. Standard communication software techniques, such as bit checking, often provide adequate control.

The EDI interface should have controls to ensure that all transactions are passed through the EDI translator to the application interface, and then to the appropriate application system. Both functions may involve techniques that use control total data included in the transmission by the sender. Control data can be included in the header and trailer records at various levels in the EDI data architecture. That data can be sequential control numbers or control totals of transactions, line items or quantities. If VANs handle the transmission, they may attach unique control numbers that can also be used for control purposes.

Once passed to the application system, EDI transactions are subject to the application's existing controls over completeness of input. Those controls may be modified to report on EDI transactions versus non-EDI transactions.

8.2. Accuracy of Input.

Accuracy of input controls ensure key transaction data is input accurately and accepted for processing. Techniques may include edit checking of key data, or matching transaction data to master files. Like completeness controls, EDI accuracy controls should prevent and detect errors throughout EDI processing. Communications interface controls should ensure that EDI document data is transmitted accurately and without damage from interference, which can usually be provided by standard communication software techniques.

As transactions are processed through the EDI interface, controls should exist to ensure that data passes through the EDI translator and the application interface accurately. The EDI translation software typically performs a number of compliance edits specific to the standard being used. In addition, various data editing procedures (similar to on-line input edits) may also be performed when data undergoes EDI interface processing.

Errors that occur during EDI processing can result in the rejection of selected transactions or complete transmissions. However, there are situations where the data will be corrected and processing continued. Records of these corrections should be maintained for audit purposes.

Resolving errors usually involves users and Management Information Systems (MIS), and may entail contacting the trading partner. Once passed to the application system, EDI transactions are subject to the application's existing controls over accuracy of input.

8.3. Authorisation of Transactions.

Authorisation of transaction controls are designed to ensure that only valid and properly authorised transactions are processed. Generally, controls over authorisation performed by the communications handler involve typical sign-on procedures, including ID and password verification of the trading partner or VAN. During processing by the EDI interface, identification codes and the type of transaction being received may be checked against approved codes in a trading partner master file. Discrepancies should result in suspension of processing for that transaction until the transaction is authorised properly. After passing to the application system, EDI transactions should be subject to the application's existing controls over authorisation of transactions.

The following discussion details controls which are typically part of an application system and are not greatly affected by EDI processing. Despite this, all of the controls should be reevaluated when an application system is modified to reflect the demands of EDI processing.

8.3.1. Generation of Transactions.

Generation of transaction controls are over computerised generation of data and are designed to ensure that all computer-generated transaction data is complete, accurate, and authorised. Since EDI processing does not automatically generate financially significant transactions, control concerns are minimal. However, functional acknowledgements generated during the EDI interface process should be controlled to ensure completeness, accuracy and authorisation.

8.3.2. File Continuity.

File continuity controls are over the continuity of master files and are designed to ensure that those files are updated as necessary and that the data remains correct and current. These controls are generally not affected during EDI since processing involves only input to or output from the application system - not storing or updating data. However, when master files are updated using EDI data (through a data mapping process performed by the EDI interface), these controls should be carefully reviewed.

8.3.3. Asset Protection.

Asset protection controls ensure that physical movement of assets is approved and recorded appropriately. EDI processing by itself does not effect the movement of physical assets. Since information received through EDI may ultimately result in physical movement of assets (payments, shipments) the information must be protected by appropriate information security measures which will be discussed later.

8.3.4. Update of Data.

Update of data controls ensure the completeness and accuracy of data file updates. Since EDI entails input or output processes, and update occurs in the application system after EDI transactions have been merged with regular transactions, controls already in place are not affected. Note that if corporate data bases are updated directly by the EDI interface function (through a data mapping process), these controls should be carefully reviewed.

9. General Controls.

General controls ensure data integrity by controlling development, maintenance, and operation of computer systems, as well as data and program security. Although existing general controls are applicable for EDI systems, new issues need to be addressed as a result of EDI processing.

The implementation of EDI systems should be controlled during the development process and should include controls over design, programming, testing, implementation, and modification. Just as with an application system, the computer operation controls are also important in EDI processing. In addition, it is strongly recommended that an internal and/or external auditor knowledgeable about EDI participate during the entire system development process.

Another issue to be considered with EDI systems is that the definition of "the system" may be expanded. The processing of a transaction by the "EDI system" may include VANs or multiple trading partners. Therefore, additional data security, confidentiality, and integrity considerations need to be addressed when the external environment is included in the system definition. The following discussion details general control objectives and related EDI control issues.

9.1. Implementation and Maintenance.

Implementation controls are designed to ensure the appropriate development, testing and implementation of a system. Once the system is operational, maintenance controls ensure that changes to the system are designed and implemented correctly. Frequently, EDI systems are implemented or modified according to the same procedures used for application systems.

An important maintenance task specific to EDI systems is ensuring that the multiple standards that may be required for different trading partners are kept current. This can be a very complex issue, and many companies use translation software provided by vendors who supply regular standards revisions and updates. A related consideration is the development of procedures necessary to add and modify authorised trading partners and new EDI transactions.

EDI processing may introduce new issues for computer operations control, such as scheduling communications with trading partners or VANs. In addition, issues addressing the resumption of processing in the event of failure or interruption should be considered during all phases of EDI processing. Backup, recovery, and contingency planning are very important since loss of electronic records can have a significant impact on the organisation's relationship with customers and suppliers. Since electronic documents are

the only forms of audit and legal evidence, data file retention policies and procedures may need to be re-evaluated.

9.2. Data File and Program Security.

Security controls are designed to help prevent or detect deliberate, unauthorised changes to data files and programs. Security controls may include physically securing hardware, data, and programs, and restricting logical access to system data and programs.

When communicating directly or through a VAN, access controls should be in place that require a user-id and password to gain admittance into the computer system. EDI environments involving a VAN should have appropriate security controls over the data residing at the value added network. Those controls can be specified in a contract and verified by on-site visits or through an independent third party review and evaluation of the VAN's control environment. When necessary, transmitted data may be encrypted as an additional security measure.[9]

9.2.1. Audit of Third Party Service Providers.

It is necessary to gain some assurance as to the effectiveness of controls and security provided by third party service providers. This is normally best achieved by an annual review, by a qualified audit firm, of these controls, with a report provided to all users.

Example areas for a third party audit include:

General:

- organisation structure;
- policies, procedures versus risk;
- contracts;
- confidentiality of algorithms and procedures;
- physical security;
- information security;
- complaints monitoring;
- audit/arbitration;
- user security/performance review;
- reconciliation of messages/charges;
- storage utilisation/charges and protection;
- compliance with legislation/convention, and;
- maintenance of history details.

[9]Encryption techniques are discussed in section 9.3.6.2., under Message Authentication.

Access/Confidentiality:

- access procedures and controls;
- key management;
- access monitoring;
- follow-up of authentication failure, and;
- confidentiality (user profiles and data).

Development and Maintenance:

- standards;
- maintenance procedures;
- message formats and their availability, and;
- software issue/maintenance.

Contingency Plans:

- alternative network/hardware;
- UPS (Uninterrupted Power Supply);
- backup software/data/documentation;
- completeness of recovery, and;
- testing.

9.3. Operational and Management Control Issues.

EDI raises operational and management control issues that relate to an organisation's efficient use of resources, achievement of management's goals and objectives, and adherence to legal and regulatory policies and procedures. Failure to recognise the importance of EDI on those controls may reduce the strategic advantages of EDI even if application and general controls are effective.

Some operational and management control issues result from elimination of paper and reduction of documentation. Other control issues relate to the electronic nature of the activity. An organisation can address many of those issues by establishing procedures for EDI use and training. Operational and management control considerations will vary by industry, organisational structure, existing facilities, and other resources. Some key issues include:

9.3.1. Legal Issues.

While many implementation and security and control issues surrounding EDI have been addressed, the widening use of this technology reveals legal questions with which few are familiar. In general, legal issues and EDI is a subject that many EDI users find easy to

ignore. As a result, it is not uncommon for trading partners using EDI to act without signed agreements specifying the responsibilities of each party involved.

This attitude has worked well in the past because EDI systems have generally been developed between trading partners with long established relationships. For such companies, good trading relationships are paramount. Thus mishaps such as wayward transactions or a security breach are unlikely to result in litigious consequences. However, as EDI proliferates as a technology, other companies will begin to employ it as a means of doing business. Trading partners may not then have the benefit of an established relationship.

At this time, there is no known litigation related to EDI. It is a legal no man's land, and is likely to remain so in the foreseeable future. Even so, several legal issues relate to the use of EDI and users should be aware of these. The relevant areas of law involved include:

9.3.1.1. Contract Law.

Questions such as "Where does legal liability fall?" and "When is a transaction affected?" are to be found under contract law. With EDI the contract should be specific with all parties understanding that they are not dealing with paper-based systems.

9.3.1.2. Confidentiality.

Non-disclosure agreements and obligations.

9.3.1.3. Copyright.

Jamieson (1990) states that in Australia the law of copyright provides an automatic form of protection, as one of the amendments to the Act in 1984 saw the term "literary works" defined to include computer programs.

9.3.1.4. Negligence.

Even though a service provider may state "all care and no responsibility" liability may exist for both direct and consequential damages arising through the use of the network or value added service.

International EDI Regulations.

The International Chamber of Commerce has developed the Uniform Rules of Conduct for Interchange of Trade Data by Tele-transmission, (UNCID) to assist in the resolution of disputes that may arise due to EDI. The first draft was based on the idea of creating a standard for communication agreements, but this was found to be impractical. UNCID is now intended only as a bridging operation until appropriate bodies of EDI law have been formulated.

The aim of UNCID is to set the basic standard of technical requirements and procedures for EDI. The standards cover such issues as:

- message structures;
- interchange standards;
- acknowledgements;
- confirmations;
- authentication;
- transaction logging, and;
- data storage.

UNCID also provides guidelines for developing agreements between trading partners. However, it appears to hold little legal authority, except in the indirect sense that non-conforming behaviour may be regarded by the courts as professional negligence.

9.3.1.5. Trade Practices.

Penalties for false or misleading conduct may be applicable for misuse.

9.3.2. The Trading Partner Agreement.

This defines the trading relationship and requirements of trade within the EDI environment between trading partners. Items that may be included in the agreement include: transaction sets to be utilised, definitions for signature and "document"; the EDI standards to be used; authorised verification procedures; responsibility for lost or stolen data; timing considerations; and, identification of VANs and their related responsibilities. Generally, the agreement includes the terms and conditions stated on the back of paper documents.

9.3.2.1. Sample Trading Partner Agreements.

Issues that should be dealt with in a trading partner agreement include:

- the signing requirement;
- EDI standards;
- timing;
- accountability;
- standard of care;
- force majeure;
- message validation and error checking;
- security and control;
- trade terms and conditions;
- entire agreement;
- confidentiality;
- arbitration, and;
- governing law.

9.3.2.2. Network Agreements.

Network or third-party value added services are used to make communications between EDI users easier. Networks are generally unregulated, and general contract law normally governs the legal relationships between network parties and EDI users. Issues that should be dealt with in a trading partner agreement include:

- entire agreement;
- data control and ownership;
- confidentiality and security;
- backup;
- statistical information;
- audit and auditor's access;
- access to documentation;
- custom programming;
- liability for errors;
- standard of performance or warranty;
- remedies;
- force majeure;
- termination;
- internetwork connections, and;
- testing of systems and interfaces with EDI, and internal interfaces.

9.3.3. File Retention.

File retention issues result from the fact EDI uses electronic source documents. Issues include how long EDI transaction files and data should be retained for tax, audit, backup, and management purposes and the form the data should be stored in. Organisations should review their record management procedures to ensure that electronic documents are maintained properly and securely for an appropriate amount of time. Issues to consider when developing policy include: initial retention time; evidence format; hardware/software compatibility; and, system auditability and security.

9.3.3.1. Paper Reduction/Elimination.

Paper will be reduced and eventually eliminated. This is one of the major driving forces for EDI's introduction throughout the world. In reducing its hard copy information, management must ensure that sufficient data is retained in the appropriate format and for the required length of time to:

- satisfy legal and audit requirements;
- provide accountability;
- enable follow-up of errors or arbitrate in the event of a dispute, and;
- facilitate reconciliation of charges.

Initially, whilst the level of paper will be reduced, it is unlikely that paper will be done away with. Some users of EDI still print out documents as they do not trust the computer systems of their trading partners. It may take some time to resolve these problems and to gain legal and general acceptance that paper can be replaced by magnetic media.

There is also a genuine concern that existing magnetic media, apart from WORM (write once, read many) disk, do not provide a secure method of storage nor do they provide the capability to prevent changes being made to original "documents". More widely accepted use of WORM disk or similar read only after initial write media, more effective and accepted security requirements, certified physical storage, a "notarised audit trail" (suggested in Europe) or checksums on files are all possible alternatives which should be addressed. This area is currently being researched.

The difference in the records will include, in the short term:

- less paper to tick, requiring the auditor to use a different approach;
- large volumes of data representing smaller value, more frequent transactions, and;
- the need to facilitate reconciliation of charges.

In the longer term, if businesses have their way, paper will be eliminated. Paper elimination projects are already being mooted overseas. Auditors will then need to grapple with the following issues:

- how do you ensure recorded data has not been changed;
- what techniques will be used to ensure that the auditor has the whole population, and;
- what techniques will be used to retrieve data and to confirm original documents?

9.3.3.2. Audit/Management Trails.

Hard copy and magnetic media management and audit trails should be reviewed to ensure there is adequate information for:

- settlement of disputes;
- reconciliation of transactions;
- reconciliation of charges;
- establishing accountability for transactions, and;
- conducting the audit.

This information should be appropriately controlled or in such a form that it cannot be altered or to ensure that any alterations are authorised, controlled and recorded. Frequency of archiving, retention of current online data, etc., should be considered. This will have an impact on transaction selection and audit timing.

9.3.4. User Education.

This is an important consideration since EDI may affect many functions and organisations within the business entity. Therefore, all affected personnel (management and staff) should be aware of their responsibilities in the EDI environment. User education should stress the organisation's EDI procedures and provide information on specific company policies.

9.3.5. System Availability and Reliability.

These issues become more critical as more EDI relationships are established and EDI technology is integrated into business operations. Since trading partners relationships are generally close ones, poor planning in handling of system availability and reliability issues may alienate customers and suppliers, cause operational disruptions, and prove costly to both the sender and recipient.

9.3.5.1. Contingency Planning.

Provision should be made for recoverability in the event of failure. In addition to the need to ensure no duplication/omission arises on recovery, consideration should be given to contingency planning for reinstatement of processing and temporary premises for short term outages. These should be agreed with the trading partner. For example, if orders are to be transmitted by phone when the computer is down, how do you confirm these are authentic?

Other items in the contingency plan should include alternative network connections and hardware. EDI network software at both the third party provider and the EDI user interface should be backed up and securely held offsite. Testing of the provider's contingency plan should take place on a regular basis. Use of an uninterrupted power supply should also be considered as part of the contingency plan.

9.3.6. Message Authentication.

For inter-company transactions, the first component of information reliability is message authenticity.

Authenticity controls can be placed around electronic messages. Many of the controls are identical to those applying to paper documents (e.g. surrounding facts and circumstances), but some new controls are necessary, so that:

- secure, professionally maintained routes of communication, which are reasonably protected from unauthorised users, can be used;
- messages and acknowledgements can quickly be checked against one another, trading partner profiles and other records for consistency;

- passwords can be used to identify the message sender to an intermediary network, and;

- secret passwords can be built into messages themselves.

Cryptographic schemes can substantially enhance the authentication of electronic messages. Yet cryptography today requires special equipment or programming, staff training, key management and trading partner coordination.

9.3.6.1. Authentication.

When data security is of prime concern, as in financial transactions, there is an additional level of security that can be used to ensure that any tampering with transmitted data can be detected by the receiver. The method that provides this security is called authentication.

Immediately prior to transmission, there is a Message Authentication Code (MAC) computed by performing certain multiple computations on the original data, initiated by a secured authentication key. The MAC obtained is appended to the original data and is transmitted along with it.

Upon receiving the data, the receiver performs the same computations using the same authentication key. If the data received is identical to that sent, the same MAC is computed at the receiving site. Any difference in MAC shows that the data has been tampered with.

The security of this method rests on the security of the authentication key at either end of the transmission. Ideally, the responsibility for maintaining each key is divided, with at least two people knowing only a portion of the key, each not knowing the other parts. Additionally, the key must be updated regularly.

As each key is only valid between a specified trading partnership, the business of key management, i.e., the generation and transmission of authentication key parts to the necessary parties, will be a very important component of this security system.

9.3.6.2. Encryption.

While authentication provides a means to detect tampering, it does not camouflage the data in any way.

In order to make transmitted data indecipherable to the casual inquiry it must be scrambled. The scrambling technique is called encryption. Again, the sender employs a key to encrypt the data and the receiver must employ the same key to decrypt the data.

To ensure total security, i.e., encryption and authentication, the data must be authenticated first and then the full stream excluding the outside envelope, which identifies the sender and receivers must be encrypted.

9.3.6.3. Electronic "Signature" Capability.

An electronic signature password-type code (agreed to by both partners) identifies an authorised trading partner transmission and its associated documents. Issues to consider involve how the electronic signature is applied, generated, and protected from unauthorised use.

10. Summary.

The challenge management and auditors face is how to implement EDI technology to attain business objectives, and at the same time appropriately control its associated risks.

While EDI does pose some threats and risks to management, and therefore the auditor, these can be satisfactorily controlled if proper planning and consideration is given to the implementation process. In the EDI systems development environment it is important that the auditor be involved in the prime developments, and especially in the choice of network providers.

It's important that both application and general controls be in place over the EDI environment (including the application systems) to ensure the integrity of electronic data. In addition, auditors should address storage, retrieval, and review of documentation issues when performing audit tests.

As Powers and Carver (1990) state:

> "The key to a well controlled EDI environment is a timely, joint effort among management, users, and auditors. Such an environment will allow the organisation to realise and attain the maximum benefits from EDI technology."

11. References.

1. Harker, S. (1989, January). Audit trails: Todays high-tech/high-touch audit challenge. EDPACS. The EDP Audit, Control and Security Newsletter, pp 1-10.

2. Jamieson, R. (1990). EDI: An audit approach. Proceedings EDPAC '90, Region 8 Conference, EDP Auditors Association, Melbourne, Australia, May.

3. Kemp, E. (1990). 58.467 Information systems security. 1990 Internal Lecture Series, School of Information Sciences, Massey University.

4. Kerr, S. (1989, April). Legal laissez-faire. Datamation, pp 54-55.

5. Kimberley, P. (1988). The A to Z of EDI: An introduction to electronic newspeak. Wellington: Messaging Communications and Trading Services.

6. Meehan, B.R. (1989). Audit implications of electronic data interchange (EDI). Price Waterhouse, Australia.

7. Messmer, E. (1989, July 3-10). Is EDI legal? InformationWEEK, pp 39-43.

8. Norris, D.M., and Waples, E. (1989). Control of electronic data interchange systems. Journal of Systems Management, March, 21-25.

9. Palmer, R. (1990, May 28). Network security issues tackled. The Dominion, p.37.

10. Powers, W.J., and Carver, T. (1990). EDI: Control and audit issues. The EDP Auditor Journal, 1, 25-32.

11. Rochester, J.B. (1989). The strategic value of EDI. I/S Analyser, 27 (8), 1-13.

12. Sokol, P.K. (1989). EDI: The competitive edge. New York: McGraw-Hill.

13. Wright, B. (1990). Auditors should be aware of EDI's legal issues. The EDP Auditor Journal, 1, 53-58.

14. Wright, M. (1989). EDI and the auditor. KPMG Peat Marwick, Sydney, Australia.

15. Wright, M. (1990). Electronic data interchange (EDI) control considerations. KPMG Peat Marwick, Sydney, Australia.

16. Wright, M. (1990). The legal agreement with trading partners: Control and audit issues. KPMG Peat Marwick, Sydney, Australia.

APPENDIX

F

Management Recommendations For Handling Electronic Data Interchange (EDI) Risk Aspects

Management Recommendations For Handling Electronic Data Interchange (EDI) Risk Aspects

Paul J. Fraser
Postgraduate Research Student

Department of Production Technology
Massey University
Palmerston North
New Zealand

This paper serves to assist management in addressing the risks and key control issues associated with Electronic Data Interchange (EDI). Details of the key control issues are addressed in the paper entitled ***Security Issues in Electronic Data Interchange (EDI)****, therefore, they are only mentioned in this paper for completeness.*

1. Introduction.

1.1. What is EDI?

Electronic Data Interchange (EDI) is the process of transferring information from one company's computer to another company's computer, **in a standard format**, so the receiving company understands the sending company's requirements and is able to act on them. Simply stated it gives companies the ability to transact business **without using paper**. This discussion addresses the issues of auditing and controlling paperless environments.

EDI is not new. What is new is the increasing acceptance of EDI as a business tool throughout the commercial world.

1.2. EDI as a Business Tool.

EDI will ultimately change the way companies do business. As a result of the extensive introduction of EDI:

- companies will be able to respond more quickly to their trading partner's requests and vice versa;
- companies will increasingly rely on their computer to support quick response trading and deliver accurate and up to the minute information;
- transcription and input errors will be reduced. However trading partners may be more vulnerable to errors in each other's systems because of the increase in automatic processing;
- companies will rely on their computer and network(s) and on those of any third party service providers via whom their messages are sent for quick, accurate and secure message transmission and processing;
- as EDI usage evolves further, paper usage will be reduced and should ultimately be eliminated; and
- new legal agreements will need to be adopted between trading partners to set the ground rules for electronic interchanges and protect each party in the event of litigation. Similarly, agreements will be required with third party service providers.

EDI will result in a win-win situation for trading partners. The benefits of EDI trading will accrue to all those using it. These benefits can only be fully exploited if EDI systems are adequately and efficiently controlled and secured.

2. The Need For Controls.

The benefits of EDI are enormous. Ultimately, as companies integrate EDI with their existing systems, they will be able to respond more quickly to trading partner requests, hold less stock, handle smaller deliveries and demand improved responsiveness from suppliers and transport providers. Dates for automatically generated fund transfers can be clearly established, offering the potential to improve cash flow and interest cost management.

These benefits have a price. As transaction processing becomes more automated and more closely linked to operational decision making, the reliability and security of a company's computer systems and networks and those of the parties with whom and via whom they interchange data become increasingly critical.

To meet the challenge of faster response requirements, controls will be increasingly automated, with a bias towards preventive controls. Different types of controls will be required to replace visual review. The company's existing application controls may be inadequate to handle this new method of operation.

As a company's EDI implementation evolves, the organisation will need to re-assess the controls over their existing environment and applications. They should address additional risks which may derive from implementation of EDI because of the resulting changes to their systems, procedures and operations.

All organisations will be exposed to risk no matter how they do business. The exposures a business may face when using EDI include fraud, disruptions to production and manipulation of company operations, impaired customer service, inappropriate decision making, inability to rely on financial statements, legal liability, loss of market share or competitive advantage and significant financial loss or embarrassment. These risks are detailed below.

2.1. Business Continuity - "Going Concern".

Inadvertent or deliberate corruption of EDI-related applications could affect every EDI transaction entered into by an organisation, impacting customer satisfaction, supplier relations and perhaps ultimately business continuity.

2.2. Loss of Confidentiality of Sensitive Information.

Sensitive information may be inadvertently or deliberately disclosed on the network or in the mailbox storage system. This could jeopardise sensitive negotiations or otherwise impact the organisation.

2.3. Increased Exposure to Fraud/Blackmail.

Access to computer systems may provide an increased opportunity to change the computer records of both a single organisation and that of it's trading partners, enabling significant fraud to be perpetrated.

2.4. Manipulation of Payments.

Where payment transactions are automatically generated by the system these could be manipulated or diverted or could be generated in error or at the wrong time.

2.5. Concentration of Control.

There may be increased reliance on computer controls where they replace manual controls, and they may not be sufficiently timely. EDI **places greater reliance on computer systems**, concentrating control in the hands of fewer individuals, with the potential to increase risk.

2.6. Reliance on Third Parties.

Companies will become more reliant on third party service providers and even trading partners to ensure security over transactions and continuity of processing. This may expose the company to weak links or unscrupulous employees within those organisations.

2.7. Data Processing and Communications Errors.

Errors in computer processing and communications systems may result in the transmission of incorrect trading information or the reporting of inaccurate information to management.

2.8. Potential Loss of Audit Trail.

There may be less paper available for verifying and reconciling transactions. A company may not be able to provide adequate or appropriate audit evidence, in hard copy or on magnetic media, for the audit to be completed without qualification.

2.9. Potential Legal Liability.

Where liability is not clearly defined in trading partner agreements, legal liability may arise due to errors outside the control of an organisation or it's own employees.

3. EDI Controls.

3.1. EDI Strategic Planning and Management Concerns.

EDI will affect the organisation and the industry within which it operates. Management will need to consider the difference EDI will make to the business strategically, operationally and in terms of accounting and control. As many of the decisions in relation to interchange will be made at the industry level, management should ensure that they are in a position to influence decisions made by the relevant EDI industry group.

Controls related to "EDI Strategic Planning and Management Concerns" include:

- *Risk Analysis,*
- *Business Planning,*
- *EDI Specialist,*
- *Systems Development Standards,*
- *Data Security Policies and Procedures,*
- *EDI Implementation Management,*
- *Accounting Controls,*
- *Operational Controls, and*
- *Legal/Contractual Requirements.*

3.2. Application Controls.

In evaluating application controls companies should assess the adequacy of controls over existing systems which will be affected by the implementation of EDI, as well as identifying how EDI will generate a need for **different or additional** controls. Controls may differ depending on whether the transactions are inward or outward bound.

The whole application and any inter-related applications must be looked at as, once transactions are "in the system", there may be limited opportunity for further review and follow up. Segregation of duties (particularly in terms of input, generation and authorisation of transactions) is an important consideration, particularly for outbound transactions.

A transaction can affect business functions far more quickly where EDI is used. Thus the requirement for faster response will dictate the need for intervention earlier in the processing cycle to act on potential problems, particularly for inward bound transactions.

The key emphasis will be on timing and for this reason automated controls will be preferred. There is a greater need for preventive controls as detective controls may be

applied too late. Automated controls will need to become more "intelligent" to compensate for the reduced human intervention in processing.

Controls related to "Application Controls" include:

- *Completeness and Accuracy,*
- *Authorisation, and*
- *Reporting, Logging and Audit Trails.*

3.3. EDI Authentication.

Authentication of EDI messages is important to ensure the company is exchanging valid data with the **correct and authorised** trading partner. Once the company moves to a full implementation of EDI they may no longer have the protection of paper documents with signatures. Authentication controls will need to cover both inbound and outbound transactions.

For outbound transactions, it is also important to consider the overall and specific applications security. Who can "give the OK" to send a message? Can invalid but apparently authorised transactions get through the system because of poor segregation of duties or review of initial documentation?

Controls related to "EDI Authentication" include:

- *Authentication of Trading Partner,*
- *Authorisation of Transaction Initiation,*
- *Authentication of Transaction Content, and*
- *Authentication For When EDI System is Unavailable.*

3.4. Security Inside The Computer.

These days security is an important issue. White collar crime, espionage, viruses and, more commonly, simple errors put computer systems at risk. As the reliance on computers increases so does the need for good security. Implementation of EDI is another compelling reason to ensure that security is sound.

Some companies may consider that their internal security is already effective. They still need to consider whether the changes to their systems, as a result of EDI, will raise their security requirement threshold.

Controls related to "Security Inside The Computer" include:

- *Logical Security,*
- *Physical Security, and*
- *Reporting, Logging and Audit Trails.*

3.5. Communications Security.

Communications security has been a hot topic for some time now as more organisations connect to local and wide area networks to meet their business needs. EDI is yet another reason to address network security. Some issues to be considered are as follows.

What communications standard (e.g., X.25, X.400) will be used and how will it be implemented? What features of the standard selected will you implement? What media and equipment will be utilised? What additional security hardware and software will you require (dial back modems, encryption hardware)?

Companies may have already addressed these issues, but if they have not or wish to review their current status, the checklist following provides a useful starting point. It is also important to note however that any aspects of data communications which directly impact the company's trading partner will need to be addressed at an industry level (will acknowledgements be used?). Companies should satisfy themselves that they have an "end-to-end" assurance of interchange integrity.

Controls related to "Communications Security" include:

- *Security,*
- *Encryption,*
- *Verification,*
- *Authentication,*
- *Audit Trails,*
- *Third Party Reporting, and*
- *Operational.*

3.6. Security - Third Party Networks & Mailbox Storage.

When a company implements EDI, it is probable that they will be relying on one or more third party service providers to ensure that interchange between the company and their trading partners consists of valid, complete and authorised transactions. The company will also be dependent on their trading partner's internal security to ensure that their data is protected from external parties.

A company's best means of protection is the contractual agreement with their third party supplier and assurance that the third party is subject to regular independent audit.

They should also ensure that the level of security and control (including reconciliations) within their organisation complements the known level of security and control provided by the third party.

Controls related to "Security - Third Party Networks & Mailbox Storage" include:

- *Logical and Operational Security,*
- *Third Party Agreements,*
- *Audit Trails,*
- *Customer/Supplier Software, and*
- *Audit of Third Party Provider.*

4. Summary.

If management are using or propose to use EDI, they should pay particular attention to the following:

- controls over their existing computer systems and environment must be sufficiently sound to protect against any additional risks they may face via EDI. If not, additional controls must be implemented to counteract the increased automation and speed of EDI trading;

- the organisation must be sufficiently involved with their Industry group for EDI to ensure that their security, control and management needs in relation to EDI are met, within the framework of any standards and procedures defined by the Group for use in intra-industry EDI communications; and

- contractual arrangements with trading partners and third party service providers, together with supporting manuals, should provide adequate protection for the organisation against direct or consequential loss in the event of failure, error or corruption of any part of the EDI network or systems of trading partners or third parties linked to that network.

The review of appropriate controls will require sound knowledge and experience of data processing, data communications and EDP audit principles and standards. Management should ensure that they have access to this expertise before commencing their EDI implementation.

5. References.

1. Fraser, P.J. (1990). Security Issues in Electronic Data Interchange (EDI). Department of Production Technology, Massey University, Palmerston North, New Zealand.

2. Harker, S. (1989, January). Audit trails: Todays high-tech/high-touch audit challenge. EDPACS. The EDP Audit, Control and Security Newsletter, pp 1-10.

3. Jamieson, R. (1990). EDI: An audit approach. Proceedings EDPAC '90, Region 8 Conference, EDP Auditors Association, Melbourne, Australia, May.

4. Kemp, E. (1990). 58.467 Information systems security. 1990 Internal Lecture Series, School of Information Sciences, Massey University.

5. Kerr, S. (1989, April). Legal laissez-faire. Datamation, pp 54-55.

6. Kimberley, P. (1988). The A to Z of EDI: An introduction to electronic newspeak. Wellington: Messaging Communications and Trading Services.

7. Meehan, B.R. (1989). Audit implications of electronic data interchange (EDI). Price Waterhouse, Australia.

8. Messmer, E. (1989, July 3-10). Is EDI legal? InformationWEEK, pp 39-43.

9. Norris, D.M., and Waples, E. (1989). Control of electronic data interchange systems. Journal of Systems Management, March, 21-25.

10. Palmer, R. (1990, May 28). Network security issues tackled. The Dominion, p.37.

11. Powers, W.J., and Carver, T. (1990). EDI: Control and audit issues. The EDP Auditor Journal, 1, 25-32.

12. Rochester, J.B. (1989). The strategic value of EDI. I/S Analyser, 27 (8), 1-13.

13. Sokol, P.K. (1989). EDI: The competitive edge. New York: McGraw-Hill.

14. Wright, B. (1990). Auditors should be aware of EDI's legal issues. The EDP Auditor Journal, 1, 53-58.

15. Wright, M., Jones, R., Preuss, L., Smith, G., Burns, S., and Page, T. (1990) The EDI Control Guide. EDI Council of Australia and the EDP Auditors Association, Sydney, Australia.

16. Wright, M. (1989). EDI and the auditor. KPMG Peat Marwick, Sydney, Australia.

17. Wright, M. (1990). Electronic data interchange (EDI) control considerations. KPMG Peat Marwick, Sydney, Australia.

18. Wright, M. (1990). The legal agreement with trading partners: Control and audit issues. KPMG Peat Marwick, Sydney, Australia.

APPENDIX

G

The MFMD *Server* Implementation: Source Code

```
Program EDIShell;

{$M 16384, 0, 65536}

{$A+}   { All data is word aligned                              }
{$B-}   { Short circuit boolean evaluation                      }
{$D+}   { Full debug information is output   - DEBUGGING ONLY   }
{$E-}   { Emulation library not included                        }
{$F-}   { Do not force all routines to be far                   }
{$I-}   { Enable I/O checking                                   }
{$L+}   { Enable generation of local symbols - DEBUGGING ONLY   }
{$N-}   { All real operations are in software                   }
{$O-}   { Disable overlay code generation                       }
{$R+}   { Include range checking             - DEBUGGING ONLY   }
{$S+}   { Stack checking enabled             - DEBUGGING ONLY   }
{$V-}   { Relaxed string type checking                          }


uses Crt, Dos, Globals, Unt_Tree, Unt_Menu,
     Unit_Sup, Unit_Str, Unt_Fdef, Unt_Form, Unit_Win, Unit_Scn;


var
    Test,
    Aborted : boolean;
    InFile,
    OutFile,
    FormName,
    Selection :PathStr;
    ItemPtr : _ItemPtr;
    OutMail : text;
    FormDesc : FormStruct;
    OutputWin : _WindowPtr;




    {$F+}
    procedure MyExit(var Regs : Registers);


    begin
        if (Regs.AX <> 0) then
        begin
            Writeln('Unexpected run time error.');
            ErrorAddr := NIL;
        end;
    end;    { MyExit }
    {$F-}



    procedure DrawScreen;

    var
        count : integer;
        Check : boolean;


    begin
        if (__IsCurScn) then
            __COffScn(TRUE);

        __ClearScn(_LIGHTGRAY, _BLUE);

        __BoxScn(1, 1, MaxX, MaxY, 15, _LIGHTGRAY, _BLUE);

        __QuikScn(1, 3, _LIGHTGRAY, _BLUE, '╠');
        __QuikScn(1, (MaxY - 2), _LIGHTGRAY, _BLUE, '╠');

        __QuikScn(MaxX, 3, _LIGHTGRAY, _BLUE, '╣');
        __QuikScn(MaxX, (MaxY - 2), _LIGHTGRAY, _BLUE, '╣');
```

```
        for count:= 2 to (MaxX - 1) do
        begin
            __QuikScn(count, 3, _LIGHTGRAY, _BLUE, '-');
            __QuikScn(count, (MaxY - 2), _LIGHTGRAY, _BLUE, '-')
        end;

        __QuikScn(2, 2, _LIGHTGRAY + _INTENSITY, _BLUE,
                  __JustStr('Management Information Services EDI System', ' ', 78,
                            _CENTER_STR));

        __QuikScn(2, (MaxY - 1), _LIGHTGRAY + _INTENSITY, _BLUE,
                  __JustStr('Press any key to return to DOS ...', ' ', 78, _CENTER_STR));

        OutputWin := __MakeWin(4, 4, (MaxX - 3), (MaxY - 3), 12, 13, _LIGHTGRAY, _BLUE,
                               _NOBORD_WIN, _LIGHTGRAY, _BLUE);

        Check := __DispWin(OutputWin);
    end;    { DrawScreen }



    procedure PrintMsg(Message : string);


    begin
        __QuikScn(2, (MaxY - 1), _LIGHTGRAY + _INTENSITY, _BLUE,
                  __JustStr(Message, ' ', 78, _CENTER_STR));
    end;    { PrintMsg }



{ Main Program }

begin
    { Set up error handling routine }

    __InsErSup(@MyExit, TRUE);

    DrawScreen;

    InitMenuDefs;
    BuildMenuTree;

    Assign(OutMail, ParamStr(1));

    repeat
        Rewrite(OutMail);

        if not DestHardwired then
        begin
            if Menu[MenuNo('DESTS')]^._MItemNum = 1 then
                Selection := _MenuString(Menu[MenuNo('DESTS')]^._MItemsPtr^._ItemRetPtr^)
            else
            begin
                PrintMsg('Select form destination');
                ShowMenu('DESTS', Aborted, Selection);
            end
        end
        else
            Selection := Destination;

        if Aborted then
        begin
            Close(OutMail);
            Erase(OutMail);

            ClrScr;
            __CSetScn(_COn);
            __CSetScn(_CBottom);

            Exit;
        end
        else
            Writeln(OutMail, Selection);
```

```
        PrintMsg('Select form type');
        ShowMenu('FORMS', Aborted, Selection);
    until not Aborted;

    FormName:= EDI_Directory + 'FDF\' + __CvtStr(Selection, 33);

    InFile := FormName;
    OutFile := EDI_Dir + 'FDF\';

    with FormDesc do
    begin
        Window := OutputWin;
        FormOp := create;
        FirstFld := 0;
        NumFlds := 255;
        FldStr := '';
        KeepScn := true;
        AcptKyStr := '';
        FdfDir := @OutFile;
        with IOSpec do
        begin
            Input.Where := FDFILE;
            Input.Name := @InFile;
            Output.Where := OPNDATFILE;
            Output.FilVar := @OutMail
        end;

        PrintMsg('Enter form details');

        Test:=Process_Form(FormDesc);

        if Test then
            PrintMsg('Form queued to send')
        else
            PrintMsg('Returning to Netmail');

        Delay(2000);
    end;

    Close(OutMail);
    if Test = FALSE then
        Erase(OutMail);
    ClrScr;
    __CSetScn(_COn);
    __CSetScn(_CBottom);
end.    { EDIShell }
```

APPENDIX

H

The MFMD *Bridge* Implementation: Source Code

```
program Bridge;

{$M 16384, 0, 65536}

{$A+}   { All data is word aligned                          }
{$B-}   { Short circuit boolean evaluation                  }
{$D+}   { Full debug information is output   - DEBUGGING ONLY }
{$E-}   { Emulation library not included                    }
{$F-}   { Do not force all routines to be far               }
{$I+}   { Enable I/O checking                               }
{$L+}   { Enable generation of local symbols - DEBUGGING ONLY }
{$N-}   { All real operations are in software               }
{$O-}   { Disable overlay code generation                   }
{$R+}   { Include range checking             - DEBUGGING ONLY }
{$S+}   { Stack checking enabled             - DEBUGGING ONLY }
{$V+}   { Strict string type checking        - DEBUGGING ONLY }


uses Scn_Util, Crt, Dos, Globals, Network, EDI_Util,
     Unit_Sup, Unit_Key, Unit_Scn, Unit_Str, Unit_Win, Unit_Pgm;


var
    LockFile,
    IncomingFile : file;
    LockFilename,
    IncomingFilename : string;
    KeyStruck,
    LockCreated : boolean;




    {$F+}
    procedure MyExit(var Regs : Registers);


    begin
        if (Regs.AX <> 0) then
        begin
            WriteError('Unexpected run time error.');
            ErrorAddr := NIL;
        end;

        ChDir(InitialDir);
        __FlushKey;
        __ResetScn(ExternalVideo, ErrorCode);
    end;   { MyExit }
    {$F-}



    procedure DrawScreen;

    var
        count : integer;
        Check : boolean;
        OutputWin : _WindowPtr;


    begin
        if (__IsCurScn) then
            __COffScn(TRUE);

        __ClearScn(_LIGHTGRAY, _BLUE);

        __BoxScn(1, 1, MaxX, MaxY, 15, _LIGHTGRAY, _BLUE);

        __QuikScn(1, 3, _LIGHTGRAY, _BLUE, '╟');
        __QuikScn(1, (MaxY - 2), _LIGHTGRAY, _BLUE, '╟');

        __QuikScn(MaxX, 3, _LIGHTGRAY, _BLUE, '╢');
        __QuikScn(MaxX, (MaxY - 2), _LIGHTGRAY, _BLUE, '╢');
```

```
    for count:= 2 to (MaxX - 1) do
    begin
        __QuikScn(count, 3, _LIGHTGRAY, _BLUE, '-');
        __QuikScn(count, (MaxY - 2), _LIGHTGRAY, _BLUE, '-')
    end;

    __QuikScn(2, 2, _LIGHTGRAY + _INTENSITY, _BLUE,
              __JustStr('MISEDI Bridge Software', ' ',78, _CENTER_STR));

    __QuikScn(2, (MaxY - 1), _LIGHTGRAY + _INTENSITY, _BLUE,
              __JustStr('Press any key to return to DOS ...', ' ', 78, _CENTER_STR));

    OutputWin := __MakeWin(4, 4, (MaxX - 3), (MaxY - 3), 12, 13, _LIGHTGRAY, _BLUE,
                           _NOBORD_WIN, _LIGHTGRAY, _BLUE);

    Check := __DispWin(OutputWin);
end;    { DrawScreen }



procedure ExamineMail;

{ This procedure examines the temporary mail directory for }
{ mail requests that have not been answered. It examines   }
{ each peice of mail and decides which handler to invoke   }

var
    Wildcard,
    Filename,
    Handler,
    MailToSendName : string;
    Args : string[255];
    DirInfo : SearchRec;
    RqstFile,
    MailToSend : text;


        procedure SendMail(var MailToSend : text);

        type
            CharFile = file of char;

        var
            OutMail : CharFile;
            OutMailName,
            NewOutMailName,
            LineToSend,
            AttemptStr : string;
            MailSent : boolean;
            Attempt  : integer;


                procedure WriteToUnix(var OutMail : CharFile;
                                          s : string);

                var
                    i : integer;
                    LineFeed : char;


                begin
                    LineFeed := Chr(10);
                    for i := 1 to Length(s) do
                        Write(OutMail, s[i]);
                    Write(OutMail, LineFeed);
                end;    { WriteToUnix }



        begin
            MailSent := FALSE;
            OutMailName := OutgoingDir + 'Q' + MyIPNum;
            Assign(OutMail, OutMailName);
```

```
    Rewrite(OutMail);

    while not Eof(MailToSend) do
    begin
        Readln(MailToSend, LineToSend);
        WriteToUnix(OutMail, LineToSend);
    end;

    Close(OutMail);

    repeat
        Attempt := 0;
        repeat
            Str(Attempt :1, AttemptStr);
            NewOutMailName := OutgoingDir + MyIPNum + AttemptStr;
            Inc(Attempt);

            Assign(OutMail, OutMailName);
            {$I-}
            Rename(OutMail, NewOutMailName);
            {$I+}
            if IOResult = 0 then
                MailSent := TRUE
            else
            begin
                Writeln('Can not use filename. (' + NewOutMailName + ').');
            end;
        until (MailSent) or (Attempt > 9);

        if MailSent then
            Writeln(OutMailName + ' renamed to ' + NewOutMailName + '.')
        else
        begin
            Writeln('All mail slots used. Delaying for server to clean up.');
            Delay(60000);   { 1 minute delay }
        end;
    until MailSent;
end;   { SendMail }



procedure LookupHandler(var RqstFile : text;
                        var Handler : string);

{ This procedure examines the rqst_file and returns the handler }
{ which knows how to process the file                           }

const
    Unknown = 'unknown.exe';

var
    Line,
    Header1,
    Header2,
    Header3 : string;
    Found,
    HeaderFound : boolean;
    FormFile : text;


begin
    Header1 := '';
    Header2 := '';
    Header3 := '';

    HeaderFound := FALSE;
    while not Eof(RqstFile) and not HeaderFound do
    begin
        Readln(RqstFile, Line);

        if (Copy(Line, 1, 1) = 'h') and (HeaderFound = FALSE) then
        begin
            Header1 := Copy(Line, 2, Length(Line) - 1);
```

```
                Readln(RqstFile, Line);
                if Copy(Line, 1, 1) = 'h' then
                begin
                    Header2 := Copy(Line, 2, Length(Line) - 1);

                    Readln(RqstFile, Line);
                    if Copy(Line, 1, 1) = 'h' then
                    begin
                        Header3 := Copy(Line, 2, Length(Line) - 1);
                        HeaderFound := TRUE
                    end
                    else
                    begin
                        Header1 := '';
                        Header2 := ''
                    end
                end
                else
                    Header1 := ''
            end;
        end;

        Found := FALSE;
        if HeaderFound then
        begin
            Assign(FormFile, 'FORMLIST');
            {$I-}
            Reset(FormFile);
            {$I+}
            if IOResult <> 0 then
            begin
                WriteError('Cannot find file. (FORMLIST)');
                repeat
                until KeyPressed;
                Halt;
            end;

            while not Eof(FormFile) and not Found do
            begin
                Readln(FormFile, Line);

                if Line = Header1 then
                begin
                    Readln(FormFile, Line);
                    Handler := Line;
                    Found := TRUE
                end
                else
                    Readln(FormFile)
            end;

            Close(FormFile);

            if not Found then
                Handler := Unknown;
        end
        else
            Handler := Unknown;
    end;    { LookupHandler }



begin
    Wildcard := 'NEWMAIL.*';
    FindFirst('TEMP\' + Wildcard, AnyFile, DirInfo);
    if DosError = 3 then
    begin
        {$I-}
        MkDir('TEMP');
        {$I+}
        if IOResult <> 0 then
        begin
            WriteError('Cannot create directory. (' + WorkingDir + 'TEMP)');
            repeat
            until KeyPressed;
            Halt;
```

```
        end
        else
        begin
            Writeln('Created directory.  (' + WorkingDir + 'TEMP)');
            FindFirst('TEMP\' + Wildcard, AnyFile, DirInfo);
        end;
    end;

    while DosError = 0 do
    begin
        Filename := 'TEMP\' + DirInfo.Name;
        Assign(RqstFile, Filename);
        Reset(RqstFile);
        LookupHandler(RqstFile, Handler);
        Writeln('Received mail to be handled by ' + Handler + '.');
        Handler := __CvtStr(Handler, _TO_UPCASE_STR);
        Close(RqstFile);

        MailToSendName := 'TEMP\MAILOUT.DOC';
        Args := Filename + ' ' + MailToSendName;

        __WinExPgm(Handler, Args, 4, 4, (MaxX - 3), (MaxY - 3),
                   _LIGHTGRAY, _BLUE, ErrorCode);

        if ErrorCode <> 0 then
        begin
            case ErrorCode of
                2: WriteError('File not found. (' + Handler + ').');
                3: WriteError('Path not found. (' + Handler + ').');
                8: WriteError('Insufficient memory to load ' + Handler + '.');
            end;

            repeat
            until KeyPressed;
            Halt;
        end;

        if DosExitCode = 1 then
            Halt;

        Assign(MailToSend, MailToSendName);
        {$I-}
        Reset(MailToSend);
        {$I+}
        if IOResult = 0 then
        begin
            SendMail(MailToSend);
            Close(MailToSend);
            Erase(MailToSend);
            Erase(RqstFile);
        end;

        Writeln;

        FindNext(DirInfo);
    end;
end;   { ExamineMail }



procedure PauseForKey(var KeyStruck : boolean);

{ This procedure delays 1 minute or until a key is pressed }

var
    WaitCount : integer;


begin
    WaitCount := 0;
    while (WaitCount < NumWaits) and not KeyPressed do
    begin
        Delay(WaitDelay);
        Inc(WaitCount);
        if (WaitCount mod 12) = 0 then
            Write('.');
```

```
    end;
    KeyStruck := KeyPressed;
end;    { PauseForKey }



procedure OpenLockFile(var LockCreated : boolean);

var
    FileStatus : word;


        function CreateNewFile(Filename : string ) : word;

        { Attempts to create a NEW file FILENAME - returns 0 if successful; }
        { 1 is returned if a file of that name already exists; 2 is          }
        { returned for any other error condition (write protect, etc ).      }

        var
            Regs : Registers;
            SRec : SearchRec;


        begin
            Filename := Filename + #0;
            with Regs do
            begin
                AH := $5B;    { Create a new file }
                DS := Seg(Filename[1]);
                DX := Ofs(Filename[1]);
                CX := 0;      { Attributes for a normal file }
            end;
            MsDos(Regs);

            if (Regs.Flags and 1) = 0 then
            begin
                with Regs do
                begin
                    BX := AX;      { Copy file handle from AX to BX }
                    AH := $3E;
                end;
                MsDos(Regs);

                if (Regs.Flags and 1) <> 0 then
                    Writeln('Error closing lock file.')
                else
                begin
                    CreateNewFile := 0;

                    { Attempt to verify file has been created }

                    FindFirst(Filename, ReadOnly + Hidden + SysFile, SRec);
                    if DosError <>0 then
                    begin
                        Writeln('Could not create the lock file.');
                        CreateNewFile := 2;    { Unable to create file }
                    end;
                end;
            end
            else
                if Regs.AX = 80 then
                    CreateNewFile := 1     { File already exists }
                else
                    CreateNewFile := 2;    { Unable to create file }
        end;    { CreateNewFile }



{ Attempt to create the lock file }

begin
    LockCreated := FALSE;
```

```
    repeat
        FileStatus := CreateNewFile(LockFilename);
        if FileStatus = 0 then
            LockCreated := TRUE
        else
            if FileStatus = 1 then
                LockCreated := FALSE;
    until LockCreated;
end;    { OpenLockFile }



procedure CopyMail;

{ This procedure reads the incoming mail file and splits it into }
{ individual mail requests. These are filed in separate files in }
{ the temporary directory                                         }

type
    Str255 = string[255];

var
    CharsRead,
    CurrentChar,
    CurrentFile : integer;
    Filename : string;
    MailEOF,
    MailLoaded : boolean;
    TextFile : text;
    LongString : Str255;



        procedure GetChar(var Character : char;
                          var EOFFLag : boolean);

        { This procedure returns the next character from the incoming  }
        { mail file. It buffers the input into Line. If EOFFlag = TRUE }
        { then EOF has been reached                                    }

        const
            BufferSize = 1024;

        var
            Line : array[0..BufferSize - 1] of char;


        begin
            EOFFlag := FALSE;

            if CurrentChar > CharsRead then
            begin
                BlockRead(IncomingFile, Line, BufferSize, CharsRead);
                if (CharsRead = 0) then
                    EOFFlag := TRUE
                else
                begin
                    CurrentChar := 0;
                    Character := Line[CurrentChar];
                    Inc(CurrentChar);
                end;
            end
            else
            begin
                Character := Line[CurrentChar];
                Inc(CurrentChar);
            end;
        end;    { GetChar }
```

```
        procedure GetLine(var LongString : Str255;
                          var EOFFlag : boolean );

        { This procedure returns the next line from the incoming }
        { mail file.                                             }

        var
            Character : char;


        begin
            LongString := '';
            GetChar(Character, EOFFlag);
            if not EOFFlag then
            begin
                while not EOFFlag and (Character <> Chr(10)) do
                begin
                    LongString := LongString + Character;
                    GetChar(Character, EOFFlag);
                end;
                EOFFlag := FALSE;
            end;
        end;    { GetLine }



    begin

        { Initialise the pointers to force filling the buffer }

        CharsRead := 0;
        CurrentChar := 1;
        CurrentFile := 0;

        GetLine(LongString, MailEOF);
        while not MailEOF do
        begin
            Str(CurrentFile :0 , Filename);
            Inc(CurrentFile);
            while Length(Filename) < 3 do
                Filename := '0' + Filename;

            Filename := 'TEMP\NEWMAIL.' + Filename;
            Assign(TextFile, Filename);
            {$I-}
            Rewrite(TextFile);
            {$I+}

            repeat
                Writeln(TextFile, LongString);
                GetLine(LongString, MailEOF);
            until MailEOF or (Copy(LongString, 1, 5) = 'From ');

            Close(TextFile);
        end;

        MailLoaded := TRUE;
    end;    { CopyMail }



{ Main Program }

begin
    { Set up error handling routine }

    __InsErSup(@MyExit, TRUE);

    DrawScreen;

    if not NetStatus then
    begin
        WriteError('You are not logged on to the network.');
```

```
        repeat
        until KeyPressed;
        Halt;
    end;

    ExamineMail;

    IncomingFilename := IncomingDir + WhoAmI;
    LockFilename := LocksDir + WhoAmI + '.MLK';

    KeyStruck := FALSE;
    while not KeyStruck do
    begin
        Write('Waiting for mail server');
        PauseForKey(KeyStruck);
        if not KeyStruck then
        begin
            OpenLockFile(LockCreated);
            if LockCreated then
            begin
                Writeln;
                Writeln('Checking for mail.');

                Assign(IncomingFile, IncomingFilename);
                {$I-}
                Reset(IncomingFile, 1);
                {$I+}
                if IOResult = 0 then
                begin
                    CopyMail;
                    Close(IncomingFile);
                    Erase(IncomingFile);

                    Writeln('Answering mail.');
                    ExamineMail;
                end;

                Assign(LockFile, LockFilename);
                {$I-}
                Erase(LockFile);
                {$I+}
            end;
        end;
    end;
end.   { Bridge }
```

References

1. Luber, A. (1987, November). Paperwork? No, communication. Purchasing World, 31 (11), p 88.

2. Buchheit, F. (1989, February). Paper to paperless - managing cultural change. Purchasing World, 33 (2), pp 40-41.

3. Burgetz, B. (1990, September). Proliferation of office technology. CMA Magazine (Canada), 64 (7), p 23.

4. Korzeniowski, P. (1990, February). "Paperless office" Deluged with paper. Software Magazine, 10 (2), pp 66-69.

5. Mueller, N. (1990, September 24). Beating the paper glut. Computerworld, 24 (39), p 117.

6. Ritter, L.S. (1988, July). Trend in copiers is one of great expectations. Office, 108 (1), pp 60-62.

7. Buch, J.R. (1986, April). High technology: Has the office become a paperless society? Journal of Information & Image Management, 19 (4), pp 14-16.

8. ibid.

9. Moore, L. (1990, August). Electronic file cabinets for the white collar factory. Management Accounting, 72 (2), pp 49-51.

10. Anonymous. (1987, June). Communications reigns as top priority. Modern Office Technology, 32 (6), pp 66-70.

11. Campbell, R.L. (1987, Autumn). Making paperwork count. Insurance Software Review, 12 (3), pp 46,48.

12. Cole, M. (1988, October). Network your way to the automated office. Accountancy (UK), 102 (1142), pp 92-93.

13. Dawson, G., and Eager, N. (1988, January). The paperless credit office. Business Credit, 90 (1), pp 18-20.

14. Finnie, S. (1988, April 25). Corporate electronic publishing: Piecing together the big picture. Computerworld, 22 (17), pp S1-S3,S8-S12.

15. Jones, C. (1989, June). The automated agent: Yesterday's dream, today's reality. Life Association News, 84 (6), pp 20-25.

16. Lukanen, P.C. (1988, November). Where does the future of electronic publishing lie? Office, 108 (5), pp 104,108.

17. Martin, C.R. (1989, March 9). Organizing the paper trail. Machine Design, 61 (5), pp 125-129.

18. ibid.

19. Graham, S.D.A. (1987, November). Would you settle for a "less paper" office? Office, 106 (5), pp 86-87.

20. Morley, J. (1988, November). Mobile storage systems glide toward greater efficiency. Today's Office, 23 (6), pp 16-20.

21. Rowh, M.C. (1987, October). Records management systems are better than ever before. Office, 106 (4), pp 92-93.

22. Cody, A. (1990, November). The changing forms of business paper. Today's Office, 25 (6), pp 36-40.

23. Radding, A. (1988, May 30). Users can't escape paper. Computerworld, 22 (22), p 57.

24. Anonymous. (1988, November). Shredders confound myth of the paperless office. Modern Office (Australia), 10, pp 14-15.

25. Anonymous. (1988, August). Optical disk filing systems ready to take off. OEP Office Equipment & Products (Japan), 17 (119), pp 36-41.

26. Alter, A.E. (1988, October). Image meets reality. CIO, 2 (1), pp 28-33.

27. Carroll, P. (1989, October). The paperless office comes true. Working Woman, 14 (10), pp 73-76.

28. Dorris, V.K. (1989, December 7). Treading the paperwork trail. ENR, 223 (23), pp 36-40.

29. Fisher, M.J. (1989, April 15). Digging out with image technology. Datamation, 35 (8), pp 18-27.

30. Plesums, C.A. (1989, May). An image worth a thousand files. Best's Review (Prop/Casualty), 90 (1), pp 54-58.

31. Waddell, J. (1989, April). Scanning the future. Insurance Review, 50 (4), pp 43-45.

32. Graham, G. (1990, May). End of the paper chase? Canadian Business (Canada), 63 (5), pp 73-74.

33. Anonymous. (1987, November). Diversification is progressing in the optical filing system field. OEP Office Equipment & Products (Japan), 16 (110), pp 30-32.

34. Faff, T. (1986, November). Optical imaging systems: Emerging technology finding its niche. Journal of Information & Image Management, 19 (11), pp 22-25.

35. ibid.

36. ibid.

37. Bate, J. St. J. (1987). Management guide to office automation. London: Collins.

38. Vervest, P. (1985). Electronic mail and message handling. London: Frances Pinter.

39. Hirschheim, R.A. (1985). Office automation: Concepts, technologies, and issues. Wokingham, England: Addison-Wesley.

40. Trudell, L. (1984). Options for electronic mail. White Plains, N.Y.: Knowledge Industry Publications.

41. Wilson, P.A. (1983). Introducing the electronic mailbox. Manchester, England: N.C.C. Publications.

42. ibid.

43. Chorafas, D.N. (1982). Office automation: The productivity challenge. Englewood Cliffs, N.J.: Prentice-Hall.

44. Miller, R. (1990, August). Electronic mail cuts through the paper bottleneck. Today's Office, 25 (3), pp 28-31.

45. Cole, M. (1988, October). Less paper: The first step to no paper. Accountancy (UK), 102 (1142), pp 88-92.

46. Adhikari, R. (1990, May). Using EDI technology in banking. ComputerData (Canada), 15 (5), pp 14-15.

47. Senn, J.A. (1989, July-September). EDI to the rescue - more productivity, less paperwork. Business, 39 (3), pp 51-57.

48. Baker, C. (1988, October). EDI: Shaping the way we trade. Accountancy (UK), 102 (1142), pp 86-88.

49. Anonymous. Electronic Data Interchange: A Definition. NZEDIA, Wellington.

50. Anonymous. (1990, October). Electronic Data Interchange: Security and Control. Solutions for Business. Coopers & Lybrand Deloitte, London.

51. Kimberley, P. (1991). Electronic Data Interchange. New York: McGraw-Hill.

52. Sokol, P.K. (1989). EDI: The competitive edge. New York: McGraw-Hill.

53. Anonymous. Electronic Data Interchange: A Service Provider Profile. NZEDIA, Wellington.

54. Payne, R.A. (1991, November). EDI gains popularity for transmitting intercompany business data. Computer Magazine, 24 (11), pp 68-70.

55. Anonymous. (1988, December/1989, January). EDI: Using technology to gain a competitive edge. C & L Executive Briefing. Coopers & Lybrand, New York.

56. Austin, J. (1988, December). Key to the value added. British Telecom World (UK), pp 24-26.

57. Boiling, M. (1988, October). Accountants lead in EDI pilot survey. Accountancy (UK), 102 (1142), p 85.

58. Jordahl, G. (1989, November/December). How EDI helps drive down freight costs. Financial Manager, 2 (6), pp 42-45.

59. Anonymous. EDI in Australia. EDI Council of Australia, Hawthorn, Victoria, Australia.

60. ibid.

61. Selwyn, M. (1989, September). Moving to end the paper chase. Asian Business (Hong Kong), 25 (9), pp 58,60.

62. Massey University. (1992). Massey University Administration Handbook. Palmerston North: Massey University.

63. Staff. (1991, December 2-16). Massey solves the paper chase. Massey University Campus News, 35.

64. Borland International Inc. (1988). Turbo Pascal: User's Guide Version 5.0 Scotts Valley, CA: Borland International Inc.

65. Borland International Inc. (1989). Turbo Pascal: Reference Guide Version 5.0 Scotts Valley, CA: Borland International Inc.

66. Blaise Computing Inc. (1989). Power Tools Plus/5.0: User Reference Manual. Berkeley, CA: Blaise Computing Inc.

67. Kemp, E. (1990). 58.467 Information Systems Security. 1990 Internal Lecture Series, School of Information Sciences, Massey University.