

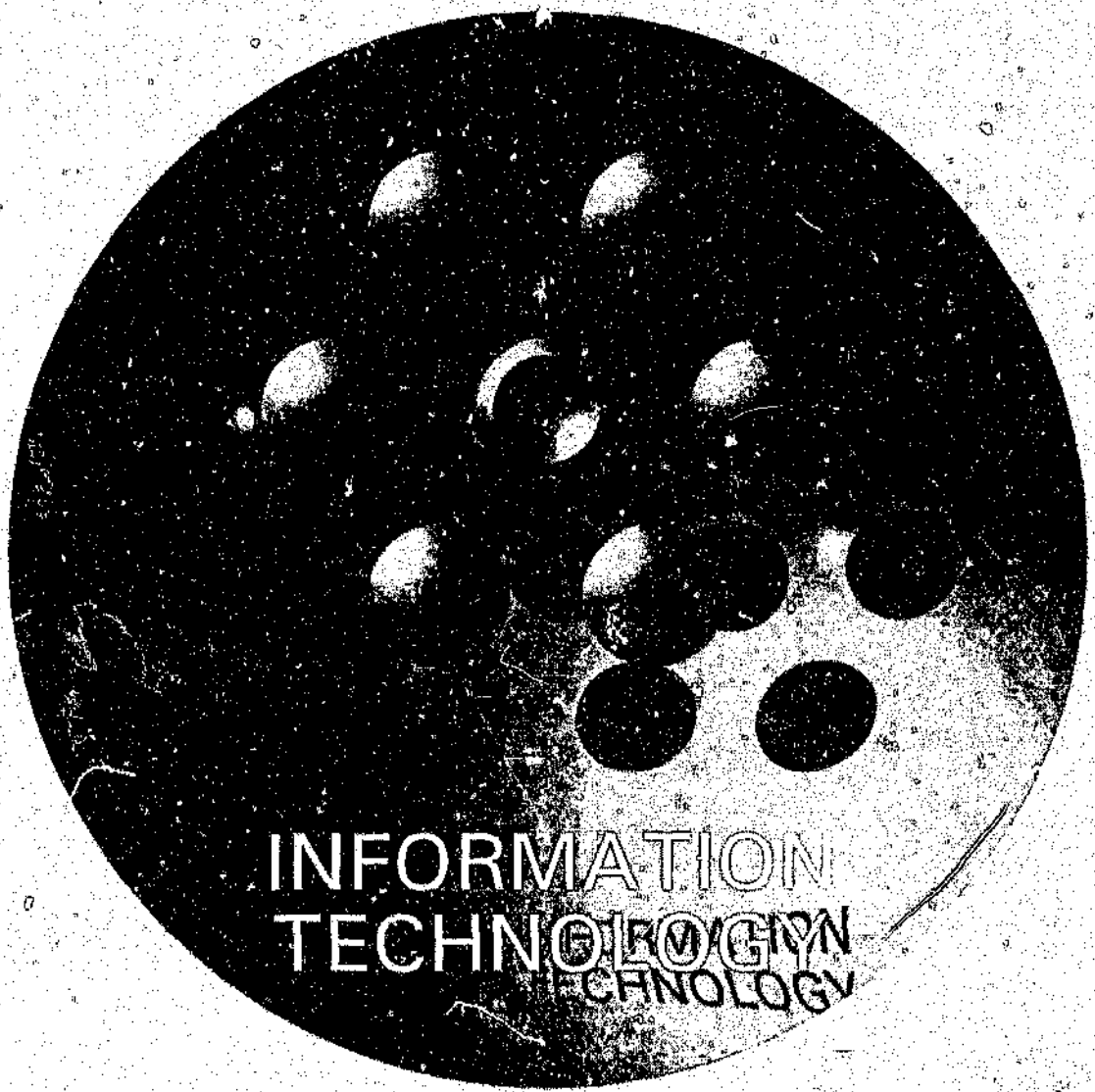
https://dspace.org/

Information technology : an information survey into the practical implementation of informal technology in a real world environment

Barker, Christopher Damian

2016-07-14
http://hdl.handle.net/10539/20581

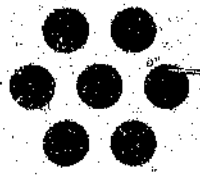
Downloaded from a DSpace repository. https://dspace.org/



An Information Survey into the Practical Implementation of Information Technology in a Real World Environment

A dissertation submitted to the Faculty of Engineering,
University of the Witwatersrand, Johannesburg
in fulfilment of the requirements for the degree of
Master of Science in Engineering

By Christopher Damian Barker, South Africa, 1993



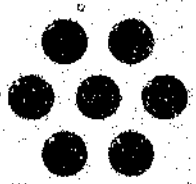
DECLARATION

I declare that this dissertation is my own, unaided work. It is being submitted for the Degree of Master of Science in Engineering at the University of the Witwatersrand, Johannesburg. It has not been submitted before for any degree or examination in any other University.

_____

(C D Barker)

16 day of October 1993



ACKNOWLEDGMENTS

The following sources of information are gratefully acknowledged for allowing the reproduction of certain extracts from their presentations or information systems:

- ZiffDavis publishing is an on-line information organisation accessed through several database forums on Compuserve (Computer and Magazine Database Plus were used primarily). This provided a plethora of articles (thousands are available) from leading-edge magazines, most of which are not available in South Africa. Searching facilities allow one to cross-reference articles, and to ensure that a good cross-section of public, and expert, opinion has been gleaned.

> **Computer Database Plus** (Go CompDB) is a comprehensive collection of computer-related article summaries and/or full text from leading computer industry publications covering hardware, software, electronics, engineering, communications and technology applications. Computer Database Plus includes nearly 200 magazine, newspaper and journal titles including PC Magazine, Byte, Digital Review, MacUser, PC Week, PC World and Communications of the ACM. Coverage for most titles starts with 1987. Search the over 250,000 articles by subject, company name, product name, featured people, publication name and publication date. Updated weekly.
>
> Find full-text magazine articles in more than 90 publications in **Magazine Database Plus** (Go MDP). This valuable research tool offers many advantages over your local library because the Database gives you six different methods of finding an article, magazines cannot be vandalized, and you don't have to leave the comfort of your own home or office.
>
> (Compuserve CIM Almanac)

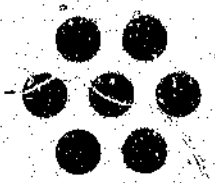
  All articles shown in the references of this report were gathered from this service.

- KIS (QData) South Africa hold commendable presentations in promoting their Information System products.

  Quotes and concepts have been utilised from their Oracle launch seminar.

- Professor Martin Healey is internationally known as a leading industry commentator, researcher, consultant, writer and seminar leader. He is a foremost authority in the fields of communication systems, PC's, mid-range systems and the integration of said. His seminars, whilst exorbitantly expensive, are both in-depth and personal. Undoutedly, his are the seminars of choice with attendance being hotly contended on his South African visits.

  Several graphics and summations in the report were adapted from his seminar content.



ABSTRACT

This report exposes the corporate manager to a new field that, to the uninitiated, amounts to nothing short of pure 'vision'.

It outlines the simplistic power of IT, and how it can be applied to solve and simplify many information management problems, moves on to explain how this new technology has only recently become available to the average concern on a single site scale, and how it is well within the reach of the non-technical manager.

The more technical aspects deal with the low-level implementation complexities that are often glossed over by system vendors, and attempts to expose the forward thinking manager to the practical realities, whilst simultaneously maintaining a positive stance on the feasiblity of a working installation.

Separate sections cover the management of change that presupposes such advanced thinking, as well as general management applicable to the wide range of skill levels traversed by business systems.

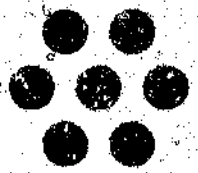


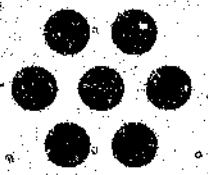
TABLE OF CONTENTS

TITLE PAGE I
DECLARATION II
ACKNOWLEDGMENTS III
ABSTRACT IV
TABLE OF CONTENTS V
TABLE OF FIGURES IX

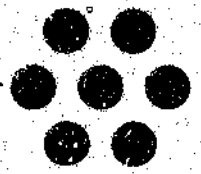
INTRODUCTION 1
REPORT LAYOUT 2
PROJECT BRIEF 3
Objective 3
Problem Statement 3
The Underlying Cause 3
The Real-World Problem 4
Mass Balance 4
IT Scope 5
Information Survey 5

BACKGROUND 7
INFORMATION TECHNOLOGY (IT) 8
Database Power and Simplicity 8
The Vision 8
Obstacles 8
Information Technology vs Systems 9
Enabling Technologies 11
System Architectures 11
LAN Servers 12
DBMSs 12
High-speed networks 13
Merits 13
Demerits 15
VANDENBERGH FOODS AND IT 16

IT SYSTEMS 18
NETWORKS 19
Topologies/Geometries 19
Time Sharing 19
Distributed Data Processing 19
Local Area Networks (LANs) 21
Communication Protocols 23
Background 23
Problem 25
Current Situation 26
Stop-Gap 30
The Right Track 31
Solution 31
Management Protocols 34
Problem 34
Current Situation 35
Solution--DMI 37
The Drawbacks of DMI 38
Conclusion 39
Network Operating System (NOS) 39
Background 39

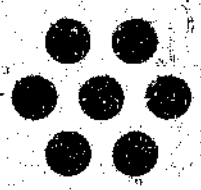


The NOS vs the PC OS 39
NOSs for Database Servers 40
Criteria 41
Reviews 42
Conclusion 46
Network Installation 49
Stage 1 49
Stage 2 49
Stage 3 49
Stage 4 49
OPERATING SYSTEMS (OS) 50
Background 50
Criteria 51
Compatibility 51
Graphical User Interface (GUI) 51
Client OS Reviews 52
Apple 52
DOS and Windows 53
Windows NT 53
OS/2 54
Unix 55
Conclusion 56
Server OS 56
MODULARITY 58
Information System (IS) Building Blocks 58
Information Flow 58
Time Frames 58
Users 58
Advantages 59
ARCHITECTURES 60
File server Database Management Systems (DBMSs) 60
Client/server DBMSs 60
Origin 60
Definition 60
Advantages 62
Disadvantages 65
Distributed (Social) Systems 66
NetWare Loadable Modules (NLM's) 68
Standalone DBMSs 69
DATA ACQUISITION SYSTEM (DAS) 71
Requirements 71
GIGO 71
NINO 71
Functions 72
Criteria 72
Graphical User Interface (GUI) 72
Programmability 73
Reviews 73
INFORMATION STORAGE (IS)--DATABASE MANAGEMENT SYSTEMS (DBMS) 74
Structure 74
Data Description Language Processor 74
Performance Statistics Processor 74
Backup/ Recovery Module 74
Database Manager 74
Structural Type 75
Origination 75
Organisation 75
Explicit Structures 77
Implicit Structures 78



Non-Relational Structures 80
Overview 81
Functions 82
Criteria 82
Clients 83
Network 83
Features 83
Reviews 87
MANAGEMENT INFORMATION SYSTEM (MIS)--DATABASE CLIENTS 90
Options 90
Fourth Generation Languages (4GLs) 90
Third Generation Languages (3GLs) 90
SQL Hooks 90
Functions 91
Criteria 91
Server Support 91
Query Tools 91
Analysis Tools 92
Report Tools 92
DBMS Abilities 92
Reviews 93
TRANSPARENCY 97
Servers 97
Overview 97
Structured Query Language (SQL) 97
The Standards Plethora 97
The Importance of Standards 99
Enhancing the Power of SQL 99
Clients 99
Problem 99
Current Situation 100
Solution 101
Networks 104

IMPLEMENTATION 105
METHODOLOGY 106
Nolan's IT Implementation Life Cycle 106
Initiation 107
Contagion 107
Control 107
Integration 107
Data administration 107
Maturity 107
The Modified Life Cycle 107
Responsibilities 108
The MIS Committee 108
Project Teams 109
Planning Fundamentals 109
Benefits From Planning the IS Project 109
Alternate Planning Approaches 111
Planning Phase 112
Recognise the Problem 112
Define the Problem 112
Set System Objectives 112
Identify System Constraints 112
Conduct a Feasibility Study 113
Prepare a Study Project Proposal 113
Approve or Disapprove the Study Project 113
Establish a Control Mechanism 113



Monitoring Project Progress 114
Implementation Phase 114
Plan the Implementation 114
Announce the Implementation 114
Organise the Information Services Staff 115
Select the Computer 115
Prepare the Software Library 115
Prepare the Database 115
Educate Participants and Users 115
Prepare Physical Facilities 116
Cutover to the New System 116
Operations Phase 117
Influences on the Life Cycle 117
Prototyping 117
CASE 118
Downsizing Specifics 119
Management Support 119
IS Department Restructuring 119
Skill Diversities 120
Outsource Implementation Skills 120
CHANGE MANAGEMENT 121
Cost of Failure 121
Short Term 121
Long Term 121
Managing Barriers 121
Success Factors 122
Transition Management 122
Managing Resistance Management 123
Reasons for resistance 123
Resistance Augmentation Factors 123
Methods of Resistance 123
Change Reaction Pattern 124
Reaction Management 124
Management Qualities 125
The Approach Policy 125
GENERAL MANAGEMENT 127
The Ohio Behavioural Management Model 127
Background 127
The Behavior Relationship 128
The Behavioral Management Model 129
Management Styles 129
Links to Other Theories 130
Style Selection 130
Subordinate Readiness 130
Regression & Progression 131
Implementation 132
Conclusion 132

THE FUTURE 133
TECHNOLOGIES 134
ARCHITECTURES 136

GLOSSARY 138
COMPANIES AND PRODUCTS 139
ACRONYMS AND WORDS 140

REFERENCES 153
DISCLAIMER 154
REFERENCES AND BIBLIOGRAPHY 155

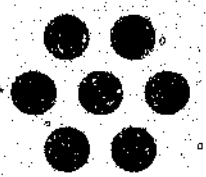
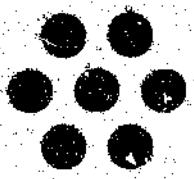
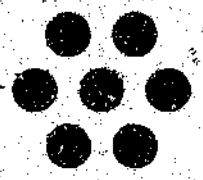


TABLE OF FIGURES

Information Systems in Information Technology 10
Information Systems in Computer Systems 11
The Progression of Information Systems 14
Network Topologies 20
The LAN Domain 21
The LAN Generations 22
Army Communication Conundrum 24
Protocol--Traffic Analogy 26
The OSI Layered Architecture 27
Protocol Layered Architecture Comparison 29
Protocol Component Examples 29
Network Operating System Control 39
The Information System Modules 58
Client/Server Traffic Reduction 63
Client/Server Architecture Efficiency 65
Typical Database Engine Schematic 75
Inverted File 76
Internally Linked List 76
Explicitly Related Tables 77
The DBMS Decision Chart 82
The SQL Family Tree 98
The Role of APIs 102
The Cooperative System Development Process 108
Planning Orientation Decision Chart 111
System Cutovers 116
Change Resistance Curves 124
The Ohio State University Subordinate Behavioral Path 128
The Ohio State University Augmented Behavioral Management Model 129



INTRODUCTION



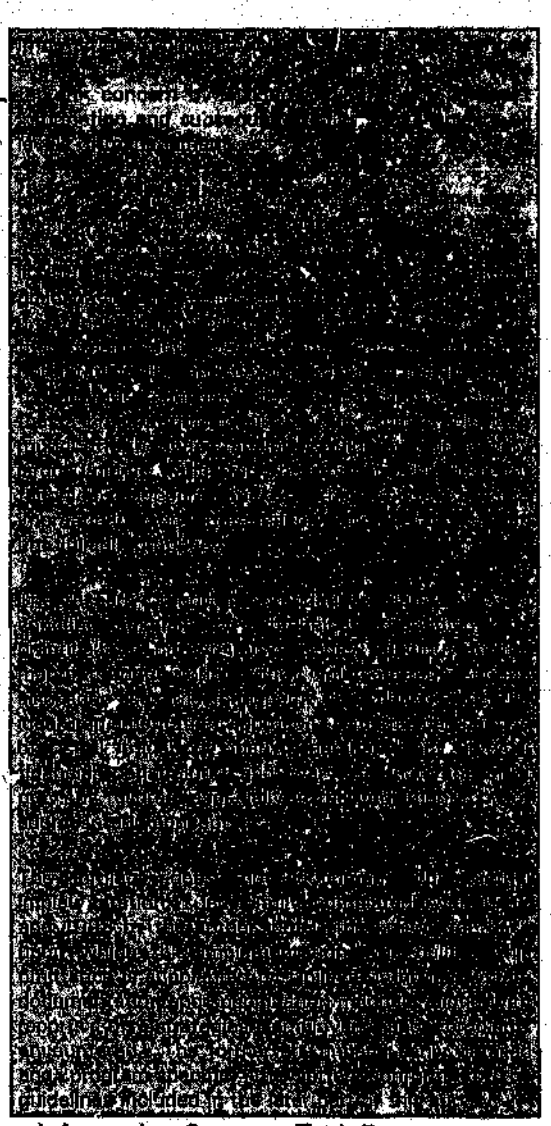
REPORT LAYOUT

Each main section of the report contains a summary block (shaded) that provides a non-technical overview of the section. It is possible to gain an overview of the report by reading these sections alone, but they do not divulge any technical details which appear in the text. This system pre-empts the need for an executive summary, which would otherwise be a simple compilation of these blocks.

The disadvantage of reading the summaries, and digressing into the text only when required, is that the text may contain reference to details divulged only in previous areas. Another complication is that the highly technical nature of the subject means that industry terms and acronyms are used right from the word "go", yet may only be dealt with, in detail, in a later section. In an attempt to overcome this, a glossary of terms is included at the end of the report, but the reader may still find that things only come together on the second read.

PROJECT BRIEF

OBJECTIVE



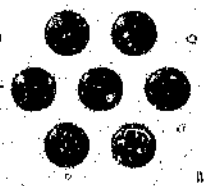
This study intends to preempt the problems associated with implementing a full blown IT system in a truly practical environment encumbered with all the problems of complex transactions, numerous events, a largely unskilled labour force, irrefutable loss of control, and severe financial restrictions. In short, your typical South African manufacturing environment ravaged by the repression of 1992/3.

The company within which the exercise is to be conducted is a subsidiary of what can be considered to be one of the largest multinationals world-wide, Unilever. Yet, despite the large number of operating companies encompassed, mostly of a manufacturing nature, no work in this direction has yet been addressed and this project is considered as being the pioneer to factory level IT implementation within the group, receiving focus from the European based Unilever Information centre (Technical Information Services).

Due to the magnitude of IT implementation within a business, coupled with the size of the Unilever group, the project is merely to address factory level transaction recording with the aim of attaining a daily mass balance (described later) across site. IT strategies for the business as a whole, commonly referred to as levels one and two, are being addressed by an Information Systems Task Force at group level, the current project being considered an essential factory interface component, but one that had only been planed for three years hence.

PROBLEM STATEMENT

THE UNDERLYING CAUSE A lack of any forward-thinking Information Systems department within the Unilever South Africa group has resulted in a complete lack of appreciation of the power and simplicity of Information Technology in a manufacturing environment. Its potential to address a serious problem within one of the group's factories has been overlooked for the last couple of years, resulting in wasted time attempting other approaches. The problem is ideally suited to IT implementation, a fact that could only be illustrated by such a precarious venture into the unknown. Such a venture had to be preceded by some serious forethought—the purpose of this Information Survey.



THE REAL-WORLD PROBLEM

Vandenbergh Foods is a margarine and salad oil manufacturing and packaging concern, and thus it deals primarily with fat *en* mass, an extremely costly commodity once extracted from its raw seed state. Over the last year, the site mass balance (a reconciliation of all the raw materials received to the product despatched) has been showing increasingly alarming disparities from its zero datum. The magnitude of these has peaked at 240 tons (R 550 000) in a single month, with an average of R 100 000 becoming the norm. With Vandenberghs producing a characteristically economy dependent product, at annual profits in the region of twenty million Rands during the 1992/3 price war, these are losses that can be ill afforded.

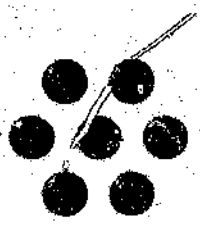
It should be noted that even when one can be sure that a loss is not actually a physical privation, the financial repercussions remain the same as the calculations of costs of production are obviously based upon the more agreeable figures. Thus, ultimately, prices are set below direct costs ex works (DCEW), an all too easy occurrence on low margin commodity products, and huge losses are incurred--in fact to the magnitude of the originally perceived misplacement. Naturally cost of capital calculations are also skewed, and thus all business decisions (primarily the capital expenditure related ones) based on this factor are erroneous.

MASS BALANCE

To digress, a mass balance involves calculating the difference between the inputs and outputs, and adjusting for opening and closing stocks within a specified time frame, which should ideally equate to zero. A negative variance (loss) indicates that losses have not been correctly accounted, or that they are occurring outside of accounting controls (such as shrinkage). Once one has eliminated the possibility of large scale theft, and once one is sure that the local lakes are not immersed in a three foot layer of oil having incredulously escaped the airtight process systems, one has no choice but to query the efficacy of the accounting control systems. Mass balances can be performed over an entire factory, else over individual departments, and even over single vessels. However, for every area one has to have a complete handle on every input (date and size), every output (including in-process losses), and an accurate measurement of stocks at the beginning and end of each time frame.

Vandenberghs stock takes span an entire non-working day (the last Sunday of the month), and cover in excess of 133 tanks and vessels spread over four plants and totaling over 4 700 tons of oil (5100 litres). Naturally packaging materials and chemicals are also stocked but these have been excluded as they do not affect the mass balance. Raw materials take the form of oil in over 100 25 ton tankers per month, or seed in over 650 30 ton rail and road trucks per month. Naturally each has to be accounted for, with a total of 27 facts for each transaction being recorded. A similar plethora of information is required for truck despatches. As product moves through the factory in semi-continuous batches ranging from 45 tons to 2.2 tons, each transfer has to be recorded and generates at least 18 pieces of information (sometimes up to 31), all of it being essential to controls. With 11 000 fluid tons per month moving through as many as 11 accounted interfaces (excluding rework), this amounts to a vast deluge of information.

When a mass balance over site, requiring the bare minimum of data for circumferential inputs and outputs, yields the aforementioned losses, the immediate approach to a solution is the targeting of smaller areas in order to try and isolate the source of loss. Naturally, each sub-area requires a similar amount of data as the site, and immediately the scope of work doubles for a single split. When it is discovered that each area gives inequitable balances, further splits are required. A Vandenberghs site mass balance is usually only available two to three weeks after a month-end stock take, the aforementioned scenario has digressed to a stage where three full-time Engineers, and the cooperation of an entire management team, cannot come up with a balance within a month, even with full-time cumulative monitoring during the month in question.



The conclusion is that manual control systems, despite recent and complete revamping, are completely inadequate at identifying potential loss of control and erroneous inputs, but more importantly are not capable of providing management information and queries with less than a significant amount of manual data processing work--all of which is subject to further error. In short, a full blown data capture and manipulation system is required to overcome these and many more problems.

IT Scope The scope of this data is significant enough to qualify its capture as a complete handle on factory level IT. Other, non-mass related, transactions may be added to the system as confidence escalates, but these typically represent far more manageable areas than does mass control, and the proven implementation of the latter would be assurance enough of the ability to incorporate the remaining controls.

The converse also applys; an IT system is capable of taking complete control of the data capture and handling.

INFORMATION SURVEY

One of the biggest obstacles in the path of such a new philosophy is not resistance to change, but rather an inability to comprehend the practical simplicity of IT, on the part of management, and thus a strong resistance arising from disbelief that time and effort invested can be made to pay for itself.

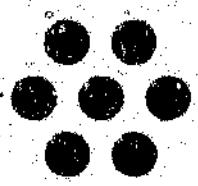
The information survey attempts to address the conflict between the need to state things simply so that it can be comprehended by non-technical persons, and the need to divulge details so that the more learned reader can be persuaded of the practical realities. It is imperative that the report conveys the vision of IT, and how the simple building blocks can be used to provide very powerful information controls in a fraction of the time that was possible in the days of mainframe computing.

The research, in itself, was necessitated by the fact that the implementation of IT, without a long term plan, can only result in rescinded work at a later point. The philosophy of "Start small, think big," abounds, and as in such cases, the small steps have to be made in the right direction. There is a very definite possibility of studying a small portion of the underlying principles, and thereafter feeling fairly confident that one knows all there is to know, other than that which industry vendors will be able to cater for. In other words, the principle of "A little knowledge is dangerous," also features.

This arises from vendor assurances that their product can address all the firm's concerns, using such phrases as "integration", "open systems", "communication protocol drivers", "distributed" and other esoteric adaptations of real definitions to reassure the uninitiated. A closer examination will start to expose the differences between integration and interfacing, the extent of 'openness', the complexity and downsides of protocols, and the, as yet, unattainability of truly distributed systems.

Whilst the report concentrates on exposing these complications, the bottom line is that the technologies are moving in the right direction, the IT philosophy has been enabled, and implementation really is fairly simple provided one avoids the pitfalls, and starts off on the right platform. The weighting towards the contrary arises from the vast array of critical information that abounds, and from the fact that "It can be done" can only be said in so many ways before repetition kicks in.

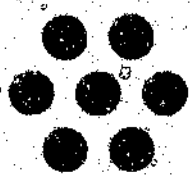
The summaries throughout the survey attempt to convey the more positive message in a format that the manager can understand, the details digress into the technical acronyms that abound in an industry where descriptions such as "Transmission Control Protocol/Internet Protocol" abound, and the bit size nature of information means that manipulation must involve a vast array of permutations, all of which need aligning.



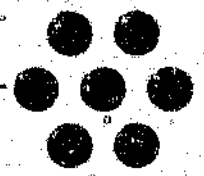
The survey has been approached from the point of view expressed by a manufacturing concern, as described later, implementing such a philosophy and attempting to reconcile all the disparities. It is not intended to address the levels that a programmer would have to be familiar with, yet it is not so shallow that the concern would have to look further for a complete understanding. This middle road has not been attained through the bulk text, which wanders from one extreme to the other, but hopefully through the combination of the summaries and the text.

The report does not attempt to lay out the specifications for an IT system design--these would be specific to each site, nor does it attempt to document the actual implementation path--this falls within the scope of company reporting. These two phases have, however, been completed subsequent to the successful implementation of the project.

This report was compiled in mid 1993, and hence all absolute references to dates throughout must be indexed from this period.

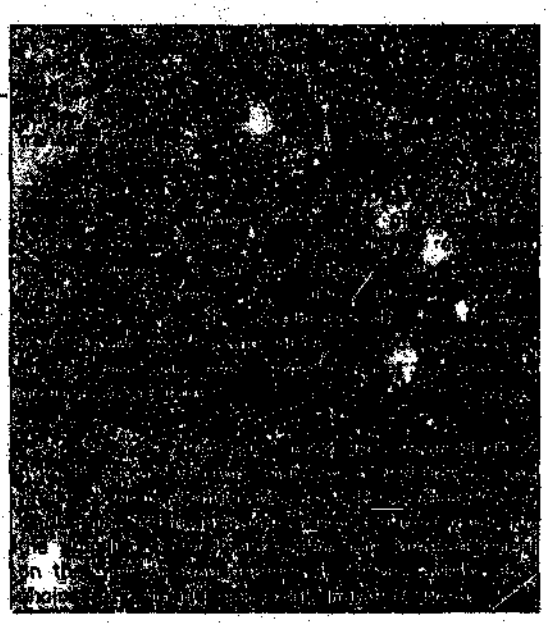


BACKGROUND



INFORMATION TECHNOLOGY (IT)

DATABASE POWER AND SIMPLICITY



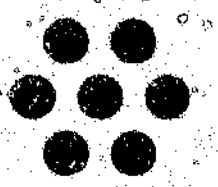
THE VISION Acceptance of IT requires an understanding of the 'vision'--the simplicity of the concept, and the power of its establishment. It is important to realise that *all* business control systems are based on a database platform--even if it is some proprietary model not conforming to the database definition, but storing records and fields just the same. The database components include a file (table) containing records (rows) and fields (columns), each cell (item) containing data in a pre-specified format (date, integer, text, etc.).

In a bank, a table may represent all their current accounts. Each record could be a transaction, with fields for account number, date of transaction, place, amount involved, etc. Any report/query could quickly extract all records for a given account number over a given period. Another table may contain all account details, with fields for account number, owner's name, date opened, credit limit, etc. A relational reference between the two tables, on the account number, would allow one to combine the account detail information with the account report, using running totals and some calculations for each record, producing a statement that showed when the account balance dropped below the credit limit, and what the interest charges were for each period in the red. The simplicity of this lies in the table analogy, its practicality arises from the ability to link computers and their database tables through a network/modem. Suddenly ATMs become nothing more than dumb terminals sending and receiving requests for reports. In fact, the limited range of reports the bank is capable of producing illustrates how simple their setup is.

Seemingly complex systems that draw money from one account to pay another are merely triggered procedures that occur on a given day (actually when all the procedures are running that night in a batch job--the reason banks close so early), that add a negative record to the one table, and a positive to another, both equal to the negative balance in the second table on the specified prior date. All the information for this procedure is stored in a third table that specifies the date the procedure should run, what account tables it should look at, what dates to use, and any other required information. The procedural batch job just runs through this third table every night looking for something to do. All extremely simple jet stunningly powerful and beneficial--all those reduced man-hours, and the wonder facilities now available to the man in the street--statements in a flash, daily interest calculations, account payments, the list goes on.

OBSTACLES This example also illustrates the downside--security. With lines (usually messages are passed through the telephone services rather than dedicated communication lines) to ATMs and across banks throughout the country, what stops criminals tapping in and sending credit messages to the database? The answer is security at the database. The database engine (controlling tables) must request a password for every instruction, yet this password must not be readable off the communication lines else it may be duplicated.

So, here we see the power of IT, yet the underlying awareness one must have of its complications. Many more lie in the communication protocols, and even more in the



hardware requirements, but the philosophy is sound, proven, and can be applied to any business--few are as complicated as banks and even they are childishly simplistic.

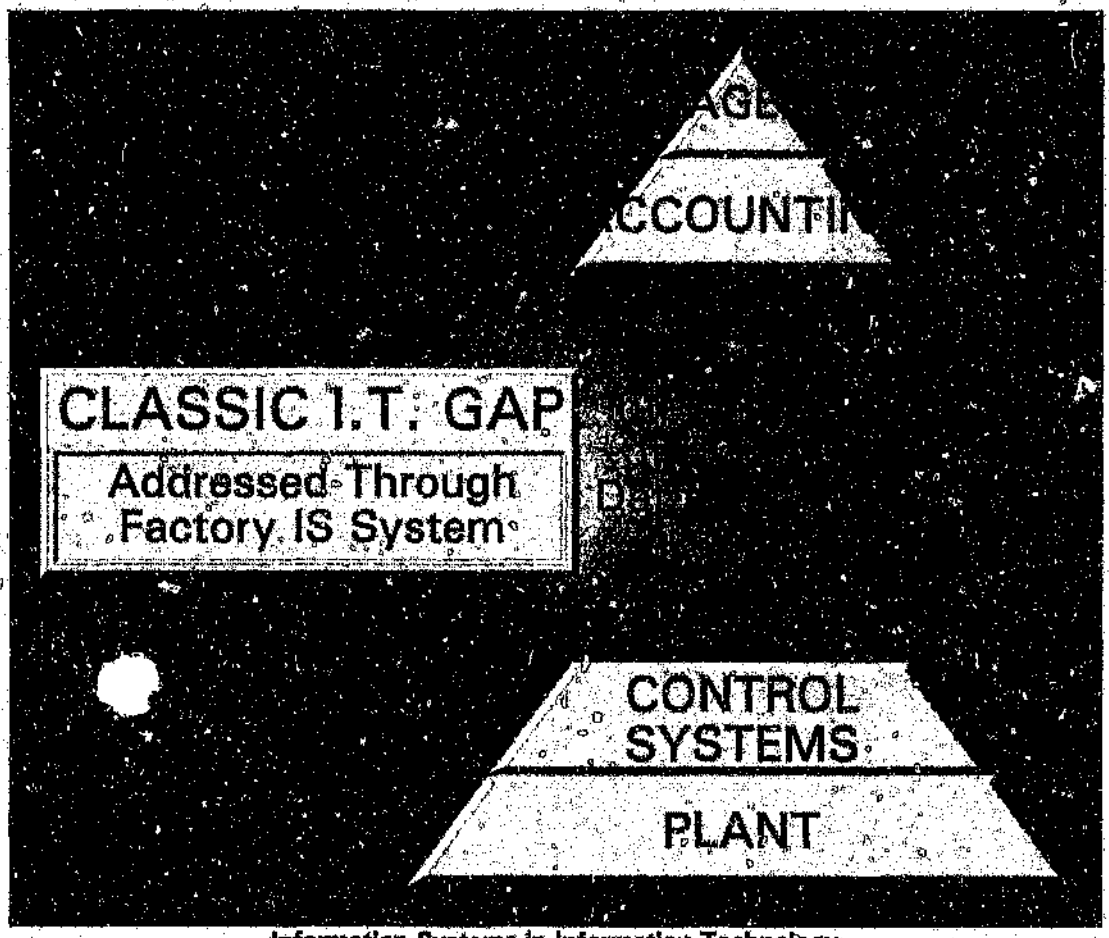
INFORMATION TECHNOLOGY VS SYSTEMS

The distinction between IT and IS (information systems) seems to lie in the software and hardware. IT is a combination of the two, whilst IS is the software aspect only. Other views point out that distinction of the two is not possible, and that they are too closely interwoven to treat separately. In this report the latter view has been adopted, but at the risk of infringing on the company's long term IT strategy (a 5 year plan), another distinction has been adopted that allows the factory level implementation to proceed unhampered by company objectives, whilst still conforming to this ultimate plan.

IT is considered to be a control and information system governing the whole Unilever group. Threads are extended into the factories to glean relevant data, but beyond that the system operates independently. IS, on the other hand, operates on a factory level and culminates in the data that the IT system pulls out. It also spans all other transaction systems, where most may not be applicable to the IT system but are essential to factory reporting and hence control.

The link lies in the threads applicable to the IT system. An IS allows all data on site to be checked and balanced, provide a high level of credibility to the IT data. Without IS the IT data is compiled through unchecked systems and may be flawed, allowing inaccurate data to extend into the group results. An IS system links the IT bottom end to the plant Supervisory Control and Data Acquisition (SCADA) systems allowing data to be derived from the source reducing the number of manual error entry points. In addition, the IS checks and streamlines data through algorithms that help reduce any errors arising in the control systems, that may otherwise have passed unnoticed through the manual assimilations and into the IT system.

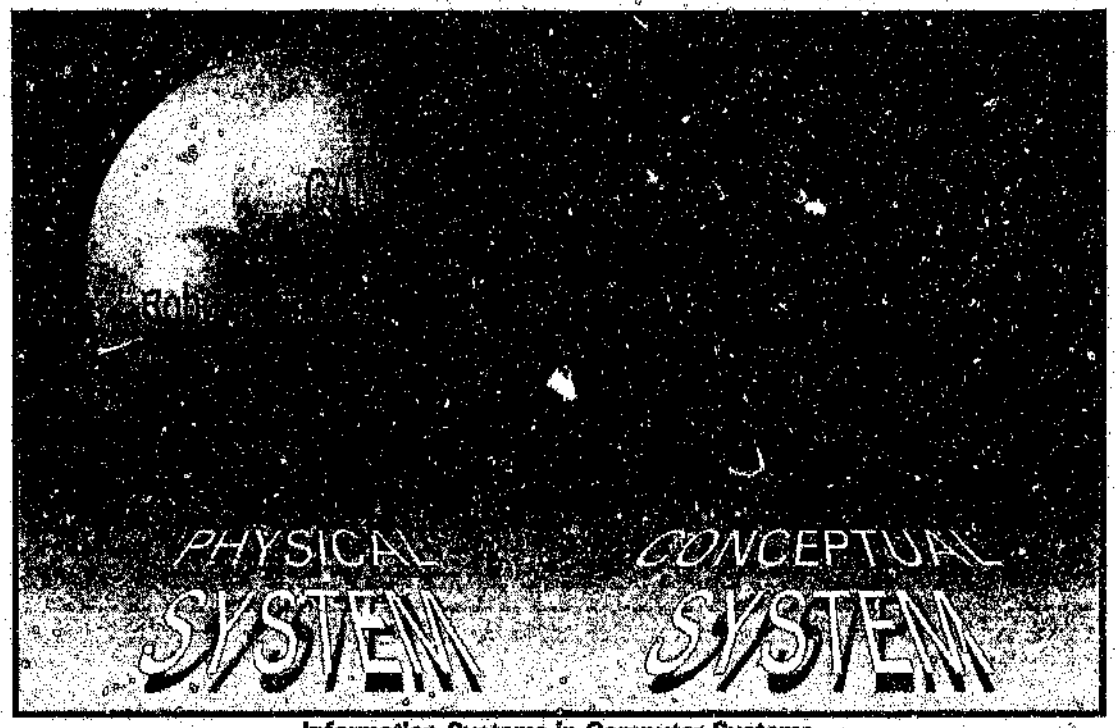
This analogy conforms to the various schematics on IT and CIM (Computer Integrated Manufacture) that show the IT/CIM 'gap' between plant control systems and management information.[1] The many variations on this schematic can all be aligned with that shown below.



Information Systems in Information Technology

Despite the comparison between IT and CIM gaps, the two should not be confused. The gap refers to the flow of information through an organisation, the areas covered by the two philosophies are quite different.[2] The following diagram illustrates the scope of each, where IT spans the conceptual system, and CIM the physical systems. The acronyms refer to the following:

- CAM: Computer aided manufacture
- CAD: Computer aided design
- MIS: Management information system
- EIS: Executive information system



Information Systems in Computer Systems

Despite the distinction between IS and IT, the rest of this report will refer to IS as IT, as another acronym for Information Storage may result in complications. However, the report does address the lesser scope of IS rather than corporation-wide IT, yet is an IT implementation in the true sense of the definition.

ENABLING TECHNOLOGIES

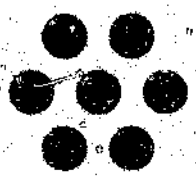
IT has become a practical reality in the manufacturing environment as recently as 1992. Evidence of earlier installations (particularly in banks) arose from proprietary mainframe systems—not a practical reality in the manufacturing environment where an entire department would have to be dedicated to control and maintenance of said computing power, and transaction control does not merit such outlays. Only recently, with increasing (international) competition, shortening time frames, more complex plants, and a general inability to cope on gut feel and manual systems, has the need for IS really arisen.

Actual installations are facilitated by reduced costs resulting from the client/server architecture which provides a very high power to cost ratio. This, in turn, has arisen from increasing desktop computing power and decreasing costs, and server power that allows a similar desktop machine, with a bit more memory, to deal with the network management and database control that a mainframe would otherwise have to do.

System Architectures

File server architectures are standard network configurations where data is stored on a centrally accessible place (the file server), and the server control routing of data from and back to each workstation. Security is limited to ensuring that a file is not opened for change by more than one workstation at a time.

When a database is used across the file server Local Area Network (LAN), the whole database must be pulled to the workstation, used, and released after results have been obtained and changes made. Apart from network loading and several other disadvantages that will be discussed later, this system severely restricts multiple use by different workstations, who have to queue for access and use. From here a philosophy arose of leaving the database engine on a single machine called the database server, which would



then control access from multiple workstations (now called database clients), and would also process queries as they came in returning a minimum of data across the LAN. This concept had been widely exploited in mainframe environments, where clients were typically 'dumb' terminals (little power), and the server was an all powerful, high maintenance, and very unfriendly COBAL proprietary package. New movements have facilitated the mimicking of this architecture within a mini (PC) environment gleaning numerous advantages over and above the typical client/server benefits.

Client/server computing has evolved from concept to reality at a surprising number of companies, mainly because of the convergence of three key technologies.

LAN servers, Database-Management Systems (DBMSs) and high-speed networks--the client/server building blocks--have matured enough in the past year to give companies the confidence they need to erect new computing architectures, according to industry analysts.[3]

These technologies have converged so quickly that even some analysts have been surprised by the client/server wave. In a recent study, market researcher Forrester Research (an organisation dedicated to gathering and marketing information on business systems and strategies) found that 69 percent of Fortune 1000 companies now use server databases, compared with only 26 percent a year ago.[4]

It's as if corporations are rushing to embrace the technologies that will at last free them from costly, proprietary computing structures. In the client/server world, the only proprietary notion is that of rightsizing--all of the enabling technologies fitting together in a scalable, interoperable platform that allows each company to manage growth at its own pace.

Before jumping headlong into the client/server market, however, one should note the continuing evolution of enabling technologies. LAN servers are relatively stable products, but database-management systems and high-speed networking technologies are expected to change dramatically in the coming year.

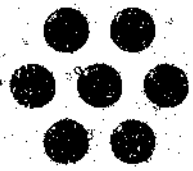
LAN SERVERS After years of marginal growth, LAN servers have become a lucrative market. Forrester Research predicts that US LAN server sales will almost double in the next few years, from US$ 700 000 in 1992 to US$ 1.2 million in 1996.[5]

A couple of forces are driving this trend. First, companies are putting more PCs on networks and may need additional servers to increase performance. Organisations are also putting more network applications on-line. Rather than installing E-mail, fax and database applications on a single server, companies are installing dedicated servers for these types of applications.

DBMSs As the Forrester report indicates, 1992 was the year that database-management systems caught on. However, most client/server database applications are still in the pilot stage, causing some reluctance among managers of information systems to rely on the new paradigm for mission-critical applications.

Part of the problem is the lack of administration tools available for client/server database systems. While some of the more established database engine vendors work on adding tools similar to those available on mainframes, many corporate customers are relying on a two-pronged strategy of running mission-critical applications on mainframes and decision-support applications on client/server databases.[6]

In addition to the lack of administration tools, database vendors have yet to perfect true distributed-database computing, in which databases on a LAN are constantly updated to reflect the latest changes. For now, administrators are skirting the problem by adding redundancy to LAN servers in the form of Redundant Arrays of Inexpensive Disks (RAID)



devices (these provide on-line and constantly updated backups/mirrors of critical information) or uninterruptible power supplies.

Most analysts say the tools now lacking in client/server database systems will be added in the next couple of years, and the Holy Grail of distributed databases will be uncovered by 1995.[7]

HIGH-SPEED NETWORKS

Systems administrators in the throes of rightsizing must constantly be aware of technology beyond the horizon. Nowhere is this watch more promising--and confusing--than in high-speed networking.

Clearly, the momentum in the market is toward 100Mbps (Mega bits per second) networking. The question is, how can corporations tap the power of emerging standards such as Fiber Distributed Data Interface (FDDI) without sacrificing their existing installations? One answer is CDDI, for Copper Distributed Data Interface--an alternative to fiber-optic networking that allows organisations to keep the copper wire they have installed in their buildings.[8]

Even more promising is Fast Ethernet--a proposal for increasing the speeds of Ethernet networks from the current 10Mbps to 100Mbps. At the moment, however, Fast Ethernet proponents are split into two camps. One calls for extending Ethernet so that corporations can switch between 10Mbps and 100Mbps speeds, and the other favors replacement of a basic Ethernet layer to achieve the higher speed.

Many of the issues surrounding 100Mbps networking should be worked out within the year. Meanwhile, the existing state of the art--10Mbps Ethernet and 16Mbps Token-Ring networking--is more than adequate for pilot client/server applications.

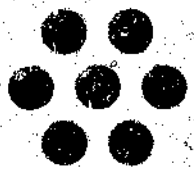
MERITS

An excellent example for the need for the type of automation exemplified by IT was the realisation by the Bank of America in the 1950s that manual handling of checks would require, in the foreseeable future, a work force equivalent to the total adult population of California.[9] Other recent implementations include systems to automate corporate operations (purchasing, inventory control, requirements planning, etc.), automated checkout systems, corporate information networks (marketing, products, payroll, employees, etc.), reservation systems (airlines, car rental, theaters, hotels, sports functions, etc.), electronic mailing, electronic fund transfers in banks (about US$ 9×10^9 in electronic fund transfers were completed in 1988 in the US, two thirds of the global GNP)[10], stock brokerage transactions, credit card verifications, and inter corporate networking to name a few.

The following merits are not those listed in a standard text book, but those that have been realised, or their definite potential identified, through the implementation of IT within Vandenberghs. They may thus be lesser than those promised by vendors, but all are realistic.

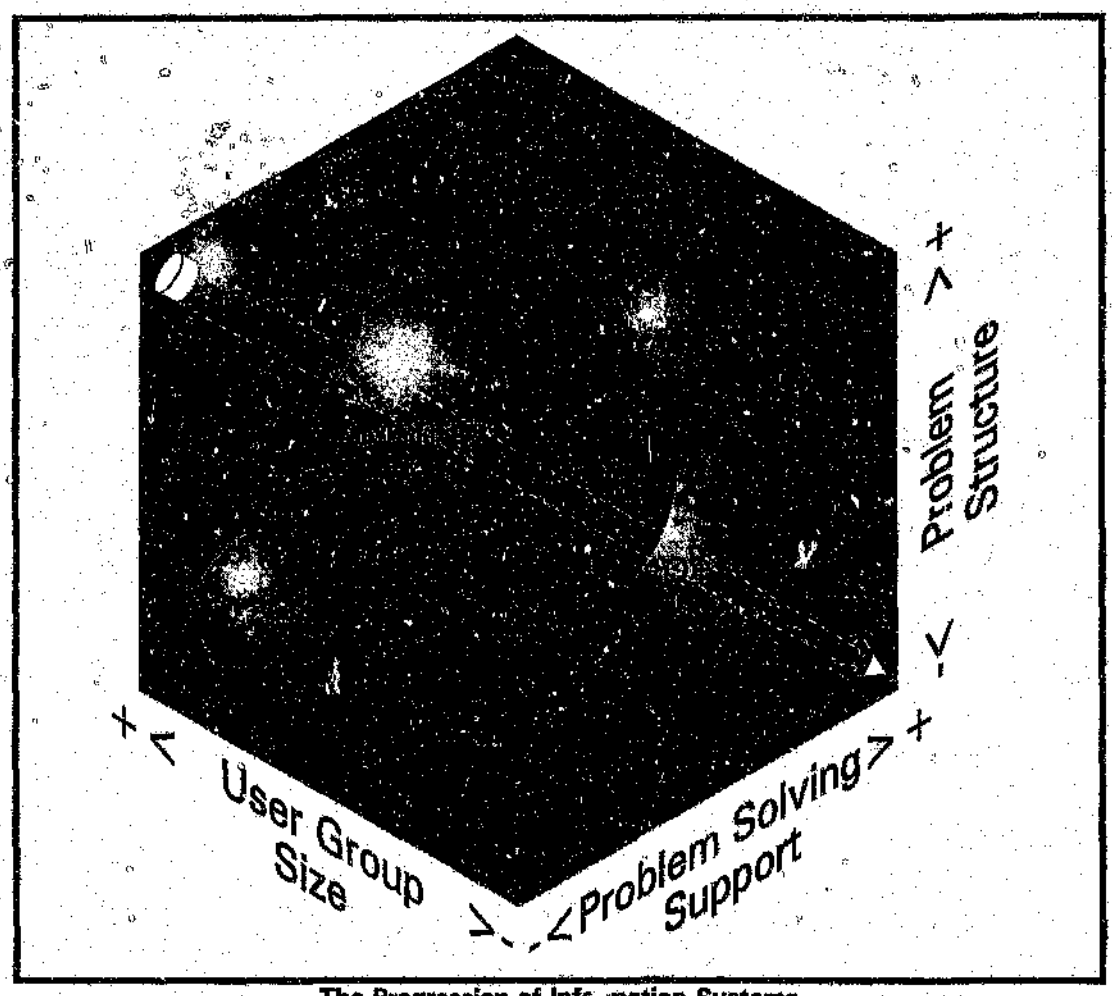
The first advantage of IT is undoubtedly reduced manning in maintaining outdated manual systems. Reconciliation of said systems, adding tables in varying manner, and compiling reports in different formats for various functions is resource intensive, slow, unreliable, and uninspiring work.

Another advantage, though not the one yielding the fastest returns, is the availability of information. Information 'glut' should not be too much of a concern as reports and queries can be easily configured to yield only the desired data. This data, being sourced from the lowest possible level, is bound to be more reliable than reports previously configured through manual systems.



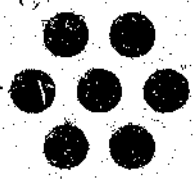
Problem solving capabilities are improved when a large, historical, information base has been compiled. Rapid reports and nested queries allow one to delve to a level not possible through manual systems, and triggered reports can highlight new problems (outside of specified parameters). A competitive advantage can be gleaned through information assisting decision making, but more realistically, rapid problem solving and identification targets the reduction of losses that may otherwise have gone unnoticed--leading to reduced efficiencies.

Information stored in a 'open' IS is available to numerous desktop applications including spreadsheets, dedicated information extractors (Management Information Systems--MISs), and information diagnosis applications (Executive Information Systems--EISs). Finally, expert systems and decision support systems can be brought into play to make use of this information. This natural progression is illustrated below, showing the decreasing level of information specialist support required, the increasing problem solving abilities, and the decreasing user group size (i.e. the most advanced level is usable by a single user rather than being requested by a departmental manager).



The Progression of Information Systems

The final culmination is 'groupware' where via a common data repository, managers get to share information, and pass it amongst themselves in the form of graphs, pictures and text documents/reports for input from each individual. The first major vendor application to make its mark in this arena is Lotus' Notes for Windows. A simple database capable of storing any type of information in a field (from a number to picture, or even multimedia video), the data can be made available to other users for their augmentation or simply distribution.



Once IT has been embedded in the company, its expansion into other areas is very easy to justify, further augmenting its capabilities, and hence its power on an exponential scale.

Finally, extending the philosophy, the linking of the IS to plant control SCADA systems is not at all impractical. SCADA systems typically hold a maximum of two day's worth of data in their RAM. If this were dumped (through a gateway--more later) to a database, not only would all this information be immediately available to the IS, but it could be fed back to the control system, through an algorithm, to keep the controls in line with desirable, historical, performances.

Numerous other benefits of IT are gleaned from the client/server architecture, but this is covered under the section on architectures later.

DEMERITS

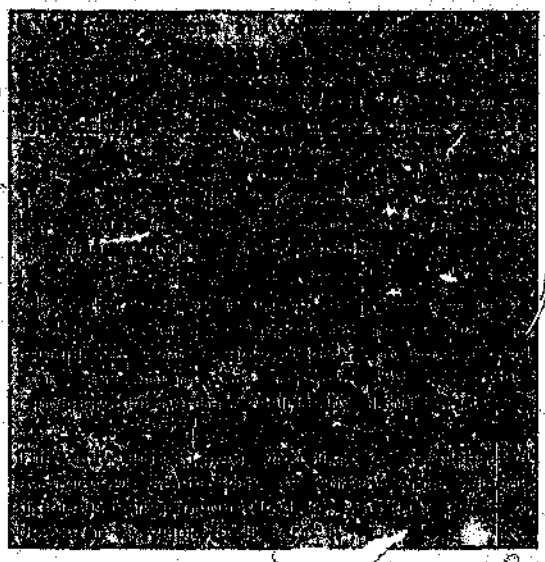
The demerits of IT implementation are historically cost and security. The security aspect arises from the new plethora of information relating to the business, that may become available to competitors. This concern can be negated by the fact that whatever form the information is in, if it is available to management, then it is available to competitors through whatever means. Naturally, if it is unavailable, the entire benefit of IT is negated.

The former concern of cost is virtually eliminated by client/server architecture, doing away with large mainframes and supporting departments. Further, the client/server components are mainly PCs which have to be procured for the company's day-to-day business irrespective. On the other hand, client/server introduces some new complications of its own, but these are discussed in the later section on architectures.

Other disadvantages relate not to the philosophy of IT, but to its failed implementation when attempting to incorporate the advantages of advancing technology, and concentrating on scaling down to existing resources.[11] These are issues that the report will address, and run as follows:

- The IBM PC, as utilised in many IT installation attempts, is inadequate for both hardware and software requirements below the 386 level;
- Industry progression form Intel 286 machines to more advanced chip architectures has still been tied to PC DOS, which is inadequate without an overlaying operating system such as Windows;
- The commercial failure of IBM's OS/2 has meant that compatibility and support was overestimated, and cross-platform integration requirements severely underestimated;
- The ethos of network users sharing a common filing system is very well entrenched. Only recently has 'groupware' (user group software) started to hype and illustrate the advantages of sharing a database rather than a file server, and thus client/server architecture acceptance has been hard to instill;
- Third generation client/server installations are badly served by DOS clients, which further hampers their acceptance and development;
- PC to mainframe connectivity is predominantly limited to terminal emulation software, and file transfer procedures rather than 'open' and transparent integration;
- Software gateways that attempt to address connectivity issues operate on an application rather than system level, thus bringing them into the realm of user computing where they can confuse and thwart users.

VANDENBERGH FOODS AND IT



An overview of the company's problem was discussed in the above problem statement. Following is some background into the concern with respect to its IT strategy, its current systems, and the cultural opposition to IT.

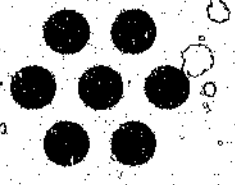
Vandenberghs has no factory level Management Information System (MIS), despite having an 'ivory tower' like Business Systems department responsible for maintaining on-site systems such as MRPII, Maintenance Scheduling, and line scheduling. Non-of these systems are 'open' and non provide information for general access and use. In addition, as 'off-site' controlled applications, there is no benefit to be claimed through 'down-sizing', a common justification for introducing client/server architectures with their reduced costs. Thus the vision of IT is not evident to any (with one exception) senior managers, and thus its potential is not appreciated, as was mentioned earlier.

'Islands of automation' abound in Vandenberghs, and with a historically rapid management turnover rate, these have been repeatedly rebuilt. That is, until recently when the plant complexity increased to a level that disallowed the luxury of time spent in constructing systems, gut feel no longer worked in a new plant with new managers, and high losses meant lots of time spent on fighting 'fires' rather than addressing the above limitations. Management computer literacy, while high compared to other organisations and any company a a couple of years ago, does not extend beyond spreadsheets (except for control engineers). A high level of automation has been attained, but this data is not retained in a database, is not readily available to management through information systems, and is thus used only by the control processes themselves, the proverbial black box. Factory level data is used for group results, potential losses are exorbitant, and a Works Accounts department spends a great deal of time compiling data that should be under the control of production management.

As with most companies, Vandenberghs assesses project viability on distributed cash flow (DCF) and payback period calculations. The budgetary constraints outlined above mean that any proposals have to meet more stringent requirements to overcome the company's increasing cost of capital hurdle.

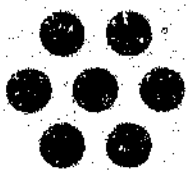
The problem that arises with IT, is that being totally unproved within Vandenberghs, there are no cost benefits that one can be assured of achieving. IT is usually justified on the grounds of decreased costs in migrating from a mainframe to a client/server environment--no mainframe exists within Vandenberghs factories. The benefits of improved competitiveness are hard to quantify, and as such cannot yield a DCF--beside which, competition in such an environment is currently being dealt with through a vicious price war curtailing any further capital resources. Even the simpler benefits of improved control, reduced workload, and reduced manning requirements are unusable as no history, within Vandenberghs, of such achievements exists. Even the team involved with implementation had little confidence in the system's abilities to yield such benefits, until implementation began to prove that they did arise.

In short, Vandenberghs is ripe for IT implementation, yet is encumbered by numerous opposing forces.



To overcome the budgetary constraints, the entire project had to be based on the addressing of a critical issue, in this case mass balance. Without this 'fire', focus would not have been given to the project, and even with this focus it never went under the guise of esoteric IT, but rather the more established label of 'control systems'. The difficulty here was keeping the project running. Once the fire seemed to have subsided, focus and resources were lost, only to be re-introduced at the next month-end crisis. The length of time required for IT implementation (at least a year--in the hands of skilled experts), meant that this focus had to be retained through clandestine means--under different guises and as 'overtime' work aside from other projects during lulls. In addition, the IT 'solution' was only comprehended a full year into the problem, and thus a year of credit time was lost attempting other solutions. As it happened, these approaches were in line with the ideal and classical IT implementation path, as outlined in a later section.

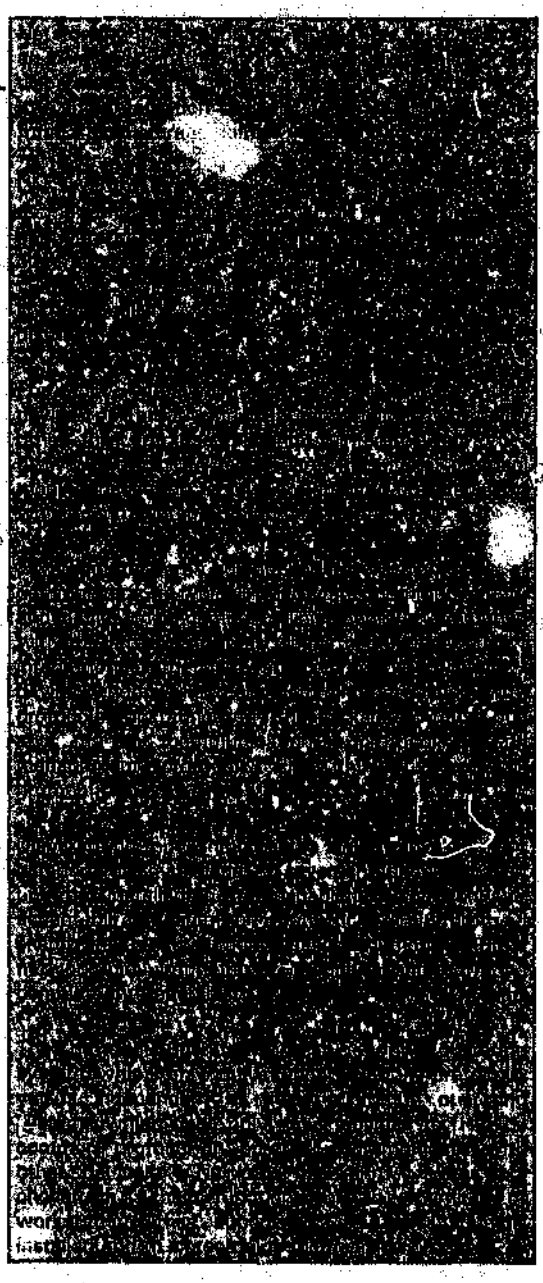
For a mass balance (as discussed earlier) to be completed, every single piece of information must be assimilated for every transaction. Any ommitances will result in a discrepancy. In addition, mass balance spans virtually all site transaction controls, and in itself has wide enough scope to constitute IT.



IT SYSTEMS

NETWORKS

TOPOLOGIES/GEOMETRIES



The following is a brief overview of network topologies, and is not intended as technical detail designed to assist in clarifying decision making as is the rest of the report, but is provided simply to facilitate a rough understanding of designs so that the rest of the text may be readily understood.

Although there is practically no limit to the number of combinations of datacom hardware and software that can be assembled to form a network there are only two basic approaches to computer processing in the network. These are time sharing and distributed processing.[12]

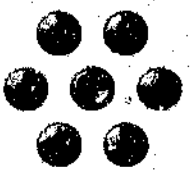
TIME SHARING A time-sharing network is one that consists of a single computer shared by multiple users who gain access by means of terminals. Dumb terminals will suffice since all of the processing is performed by the single computer. This approach enjoyed its peak popularity during the late 1960s and early 1970s when firms sought to maximise their computing power by installing large computers at their headquarters. The first users were scientists and mathematicians who used the central computers to perform computations that required large systems.

This configuration receives vast support today, but by most it is considered a technology more suited to the future (5 years hence) as current networking speeds do not readily support the more pressing requirement for intensive graphical displays.

DISTRIBUTED DATA PROCESSING When small computers became popular, the firms changed their strategy and began distributing the minis and micros throughout the organisation. When these systems are interconnected, the technique is known as distributed processing or distributed data processing (DDP)--not to be confused with a distributed server architecture. Even though small computers stimulated the DDP approach, such networks can include computers of all sizes.

When the network is used for distributed processing, the processors can be interconnected in one of six typical patterns. The term topology is used to describe a particular network pattern:

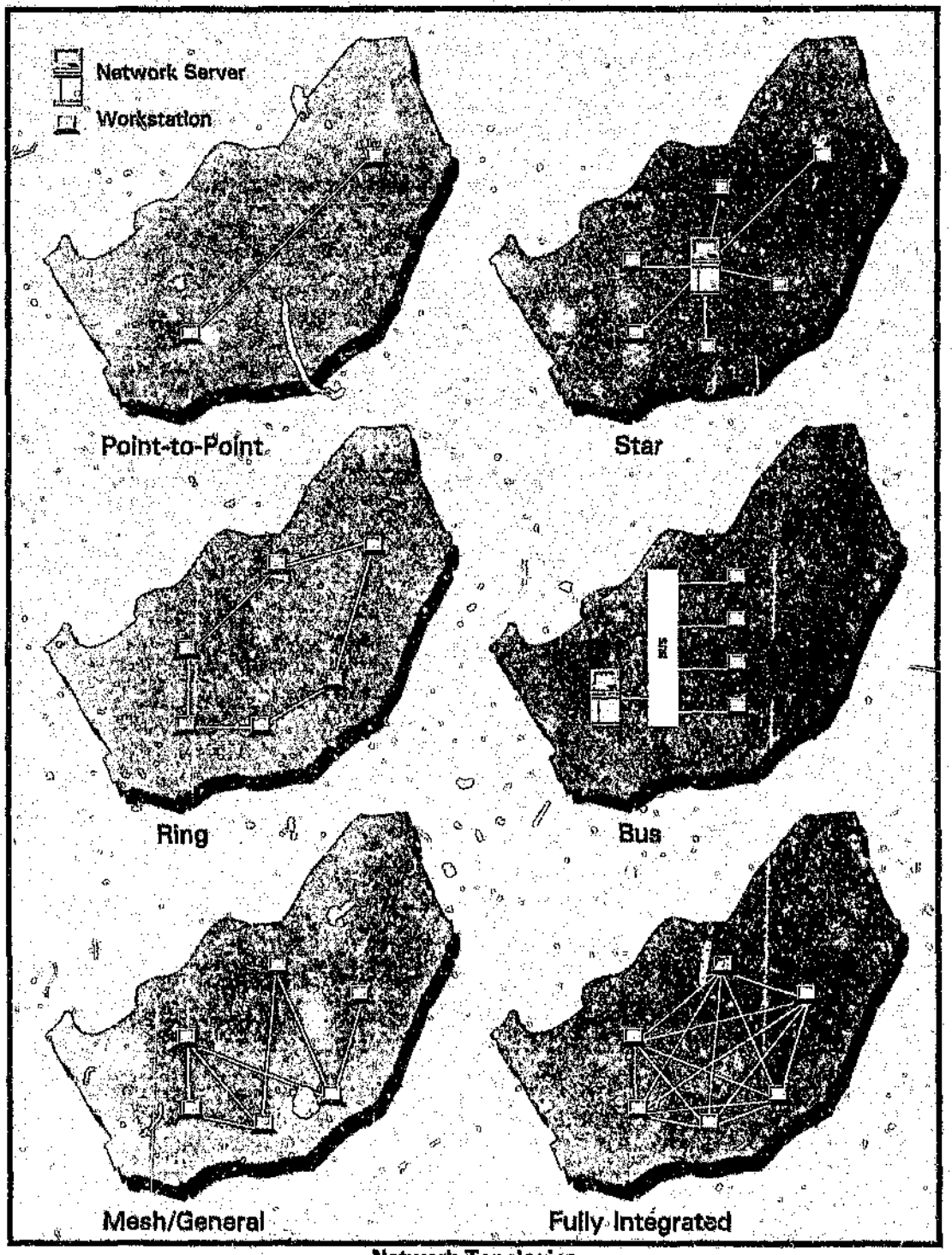
STAR A star network includes a central computer as shown in the figure. The central computer is called the central node. A node is a point in the network where the circuits



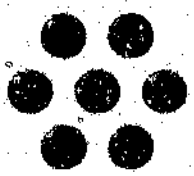
are interconnected by one or more units. The other nodes of the network can be computers of any size. Some nodes can be terminals.

RING When the network does not include a central node, the name ring network is used.

BUS In addition, the workstations and network server can be attached to a bus--a single circuit of limited length.



Network Topologies



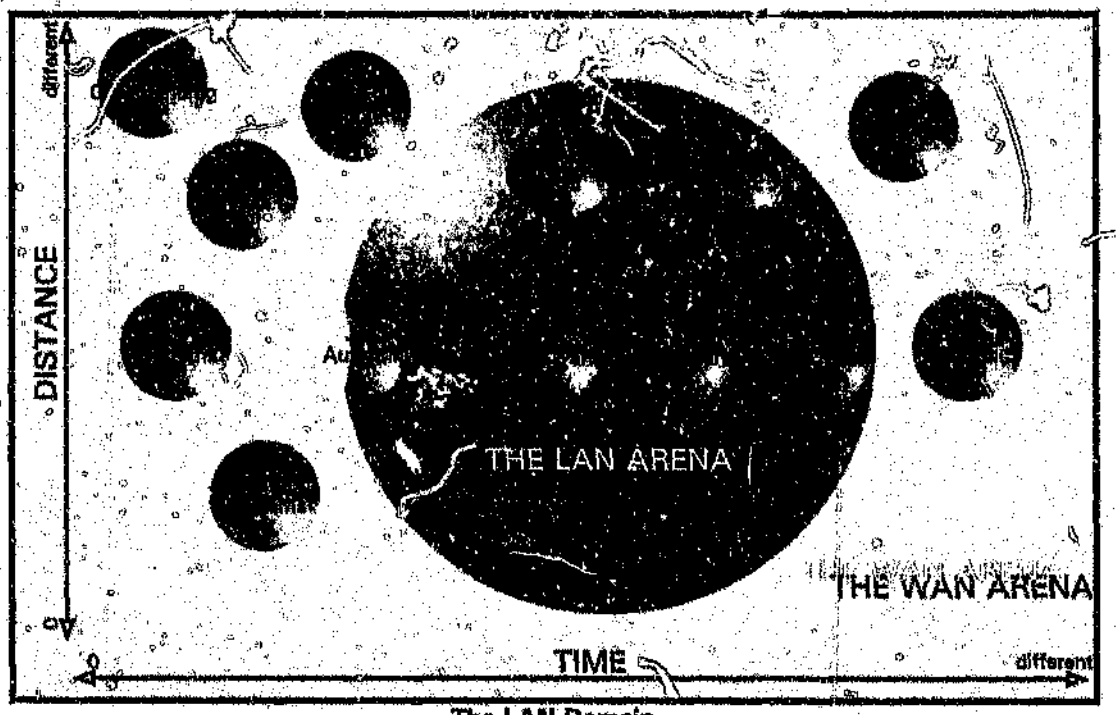
POINT-TO-POINT Sometimes referred to as peer-to-peer networking, this hardly constitutes a network until more than two computers are introduced, yet it does raise the subject of communication difficulties.

GENERAL/MESH Where several possible communication paths exist, networks have to have the facility to determine the optimal routing taking distance, link capacity and load, connection speed, and workstation speed into account.

FULLY CONNECTED The most fault tolerant of configurations, it is impractical for large networks as there are $N \times \frac{(N-1)}{2}$ links for N nodes. Each node requires N-1 ports for connection to other machines, and routing algorithms can take longer to run than the least optimal route would have taken.

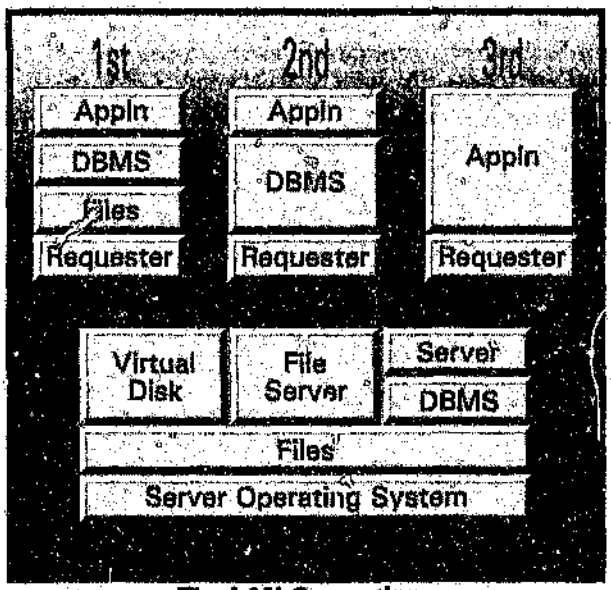
Networks such as the star, and ring are commonly called WANs, for wide area network. Smaller-scale networks can exist within each of these three WAN topologies, or they can exist alone. Two of the smaller network designs are local area networks (LANs) and private branch exchanges (PBXs). Although PBXs have achieved wide acclaim for their ability to handle both data and voices simultaneously, they are not a favoured system of the future due to several limitations and the ability of LAN topologies to cater for these facilities and a great deal more. Both LANs and [illegible] hence considerable compatibility problems when it comes to a common communication language, as is discussed later.

LOCAL AREA NETWORKS (LANs) LANs typically cater for the groups of software interconnectivity shown below.



The LAN Domain

While there is no official classification of LAN systems the nomenclature in the following figure is a good reference.[13]



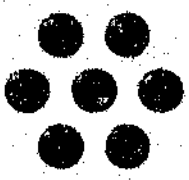
The LAN Generations

First generation LANs provided an image of a local disc on a common server. Today this concept is only used for bootstrapping diskless workstations. The second generation networks, dominated by MS Net, Novel Netware and more recent products such as Lantastic, map the basic PC DOS file and print facilities onto a communal, shared server. This concept is fine for multiple individual PC users since the shared file server can be made more reliable and more manageable than multiple independent discs. Access control can be imposed and archiving and mirroring of discs provides a more robust service. Sharing a 'logical' printer via a print spooler is also invaluable to multiple PC users as is access to communication gateways.

The flaw with second generation networks is that they only attack the multiple, single-user concept and not a multi-user architecture as is nowadays common in minis and mainframes. Indeed it should be realised that early minicomputer operating systems such as DEC RSTS/E also failed to provide shared data resources before version 6. The dramatic improvement in quality that ensued should have been a lesson for today but the PC industry is too arrogant by far and never learns from past experience.

The requirements for a multi-user system is to share the database, not the lower-level file system. It must be recognised that a DOS (or Unix, concurrent DOS, etc. for that matter) file is simply a byte string. It is sub-divided into records and fields, with appropriate indexing schemes by a data management system. The database management system can range from a simple indexec access method (ISAM) such ac C-ISAM or B-Trieve, through a multi-keyed access method such as dBase or IDMS to a full relational database system such as Sybase, Oracle, Ingres, etc. The requirement for a multi-user system is to share the DBMS, as in the third generation network configuration. With a shared file system there is no security (or access control); once a DOS program is granted access to a file it can corrupt it. There is no 'roll-back'--so essential to transaction processing, and there is no access control at the record level, i.e. one user being able to update a filed where others can only read it or not even see it. Thus client/server is essential to bring security back to PC networks as is found in mini and mainframe syst. ns.

This concept of a workstation executing the application with a graphical interface connected to a shared database server in the most common (but not by any means the only) example of client/server computing. It is also common to think of a Sun or H-P/Apollo as a workstation and an Intel x86 ISA machine as a PC. This is not true. For a workstation, connection to the LAN is a fundamental facility and must be part of the system software, in contrast to a personal computer where communications is an application option.



PC DOS is a killer in client/server systems. Since the client needs robust, flexible and transparent connection to the server, a protected mode multi-tasking workstation operating system is needed. Thus OS/2.2, Win32 or possibly Unix (but most unlikely on PC platforms) are the candidates. Windows 3. 1 is an enigma; it is a DOS extension and technically inadequate so that its inherent limitations will be inevitably and painfully exposed by client/server computing. Windows 3. 1 is not really a product, it is creating a market for a replacement product which Microsoft hope will be Win32 (the PC version of Windows-NT) but, some say, is more likely to be OS/2.2 going on Microsoft's track record for missing delivery dates by wide margins. It doesn't matter though whether its OS/2-2 or Win32 as long as DOS and Windows can be replaced. The general recommendation is that for users starting with GUI interface and client/server computing, to bypass Windows and go straight to OS/2.2 as it will be far cheaper in the long run.

COMMUNICATION PROTOCOLS

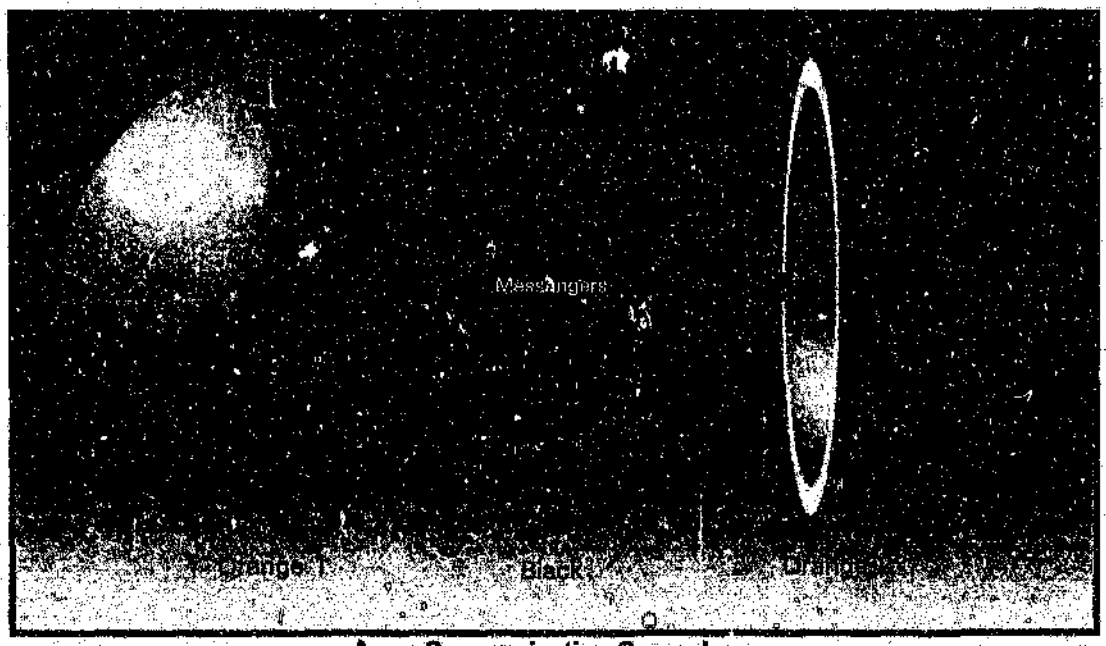
BACKGROUND If two devices are to successfully communicate with each other, their communication must be coordinated so that one knows (or can learn) when to talk (transmit) and the other knows (or can learn) when to listen (receive). No useful communication takes place if both are only listening or only talking. (Data processors may usefully transmit and receive at the same time, but techniques for doing so require coordination).[14]

There must be adequate synchronisation of basic signal elements--for example, sinusoidal-like wave forms for some types of communication and pulse strings for others--so the receiver knows when to look for data. This involves clocking at the transmitter and clock recovery at the receiver. Signal elements must be processed to yield bit patterns, with the transmitter knowing when to transmit and the receiver knowing when to look for them. Next, bits must be assembled into characters, with the receiver able to determine when complete characters are available. Characters must be assembled into frames or messages, with techniques to determine beginning and end of frames. Furthermore, it is common for different types of information, such as control, address, data, and error protection overhead, to be included in one frame so methods for locating different types are needed.

Determining which device has access to, or control over, a communications medium at a given time is another type of coordination. Initiation, use and termination of a connection, or of a dialogue using a connection, must be synchronised. Connection establishment and termination can be required at virtually any level of an architecture; it may include establishment and termination of connections between data sources/sinks and communication facilities, of connections over physical links, and of connections at the data transfer level or at higher levels.

Another category of related problems occurs in error recovery, which often involves putting both ends of a communication link back into a known state. Synchronisation points established in some of the higher layers of protocol architectures are points at which the states of both ends of a link are known (at least with high probability), so the system can be restarted from these points if problems occur. Major problems for coordinating sender and receiver arise from the fact that they must communicate over communication links that are not perfectly reliable.

A classic problem in distributed communications illustrates the difficulties. Analogous situations occur in computer networks. The problem is known as the three army problem.[15]



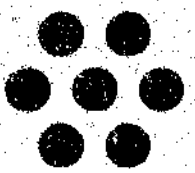
Army Communication Conundrum

There are three armies, two orange and one black, positioned as shown in the figure. The black army is larger than either orange army but not as large as both combined. In this simplistic model of a battlefield, the larger army always wins a battle. Hence, the orange armies can be assured of victory if both attack the black army simultaneously, but either orange army is assured of defeat if only one attacks the black army. The only method of communication between the orange armies available to them is via messengers who must go across black army lines to reach the other army. This type of communication is far from being perfectly reliable, since the black army will try to capture any messengers passing through.

Assume that the commander of the two orange armies, with headquarters in the area occupied by Orange Army 1, wants both orange armies to attack at 2:00pm on Saturday. No attack will take place, however, unless each orange army is certain that the other will attack at the same time. To ensure this, Orange Army 1 sends a messenger through the lines to Orange Army 2. There is no guarantee this messenger will get through, however, so neither army will attack until receipt is verified. Hence, after receiving the message, Orange Army 2 sends back a messenger to notify Orange Army 1 of receipt. If the confirmation message gets through, both orange armies will know the schedule for an attack, but neither attacks until it is absolutely sure the other will attack simultaneously.

The exchange of messages so far is not adequate, since Orange Army 2 is not sure its confirmation message got through, and knows Orange Army 1 will not attack if it did not. Orange Army 1 can, of course, return still another message to indicate the confirmation got through, but it as no way of guaranteeing the confirmation of the confirmation will get through. Without this, Orange Army 2 will not attack on schedule.

Although the two armies can continue indefinitely to send messengers back and forth, there is no way to absolutely guarantee a simultaneous attack with this approach. This can be deduced from the following reasoning. Assume a finite number of messages is adequate and that the most recently sent (or received) message is the last essential one--that is, a simultaneous attack is guaranteed if and only if it has been received. If it is known to have been received, no more messages need to be exchanged. However, the army sending this last message will not attack unless it knows the message got through. This implies another (acknowledgment) message is also essential. This gives a contradiction, so the assumption that a finite sequence of messages is adequate must be false.



There are techniques for guaranteeing a very high probability of a simultaneous attack, but no way of gaining an absolute guarantee. Similarly, there is often no way to absolutely guarantee coordination in a network. However, this is not a life-or-death situation in networks, and a high probability is satisfactory, a goal facilitated by protocols dictating the simultaneous coordination of both machines. The datacom devices follow the protocols in communicating with one another. This has been called shaking hands.

A protocol is a set of mutually agreed upon rules of procedure stating how two or more parties are to interact to exchange information. Relatively informal protocols may be satisfactory for some situations, such as the exchange of information between people, but they need to be carefully defined for communication among machines. What is communicated, how and when it is communicated must all be specified.[16]

The key elements of a protocol are as follows:

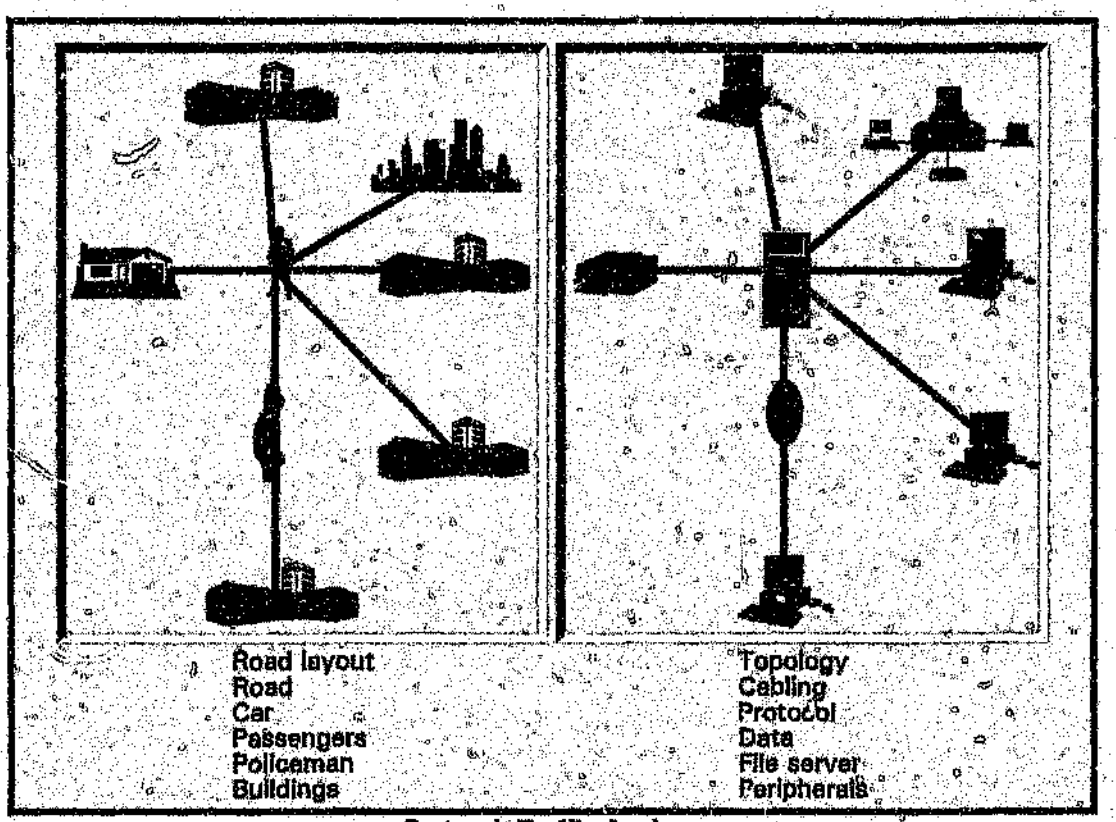


Problem One of the characteristics of the datacom field since its inception, has been the wide variety of hardware and software products on the market. This variety has been a blessing for datacom users since it has stimulated competition among the suppliers and made available a wide selection of models. However, the variety has been a burden in that it has made it difficult to interconnect the different suppliers' products.

With all the new possible client and server operating systems, companies will soon gain an unprecedented array of options. If the networking world were perfect, any combination of these would work together and mesh smoothly with installed networks.

An analogy of data communication to a traffic pattern help in an understanding of protocols and where they fit in. If the road network (grid pattern, ring roads as in UK, completely random as in SA) can be thought of as the network topology, then the cabling means can be thought of as the road. The communication protocol can be thought of as the cars, and the destinations as peripherals running in certain operating environments/systems. The passengers would be the data, and a traffic policeman the network server. Finally, the management protocol would be the format of the address book for all destinations.

To identify a peripheral (destination) and it's running state, the network management protocol (address book) would provide the necessary information. Passengers (data) would be loaded into a car (protocol) at the source (by the source peripheral), and either directed (routed) by the police-man (server) to the destination (destination peripheral), else would just drive around the roads in a fixed route until the destination was reached (as in a token-ring topology). Provided the destination had a road and accepted cars, or a railway and accepted trains etc. (had a protocol driver), the car would be unloaded and the passengers (data) accepted.



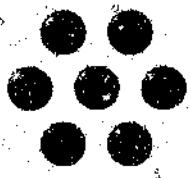
Protocol--Traffic Analogy

CURRENT SITUATION Luckily, some of the suppliers saw the problems coming before the situation got out of hand. IBM was among the first. In 1970 IBM was marketing 200 different datacom products that could be interconnected fifteen different ways. IBM management decided that a set of standards should be defined that could serve as a guide for future developments.[17]

IBM named its system of protocols Systems Network Architecture, or SNA. In designing SNA, IBM considered all of the activities necessary in transmitting data through a network with a user at one end (called the user node), the host computer at the other (called the host node), and several intermediate nodes that consist of such other devices as the front-end processor and cluster control units. IBM separated the physical activities that transmit the data from the logical activities that control the physical transmission. SNA was aimed at the logical activities. SNA classified the logical activities into layers. The purpose of the layers was to insulate the user from changes in the datacom hardware and software. For example, the firm could convert from a network of dial-up lines to a LAN and the change would not affect the user or the user's software.

SNA was received so well that other computer manufacturers developed their own standards. For example, Burroughs announced its Burroughs Network Architecture (BNA), and Honeywell developed its Distributed System Environment (DSE). However, users did not regard the large number of manufacturers 'standards' as the solution to their problem. For example, IBM's SNA made it easy to interface IBM hardware and software, but it did not help the user who wanted to mix IBM products with those of other datacom suppliers.

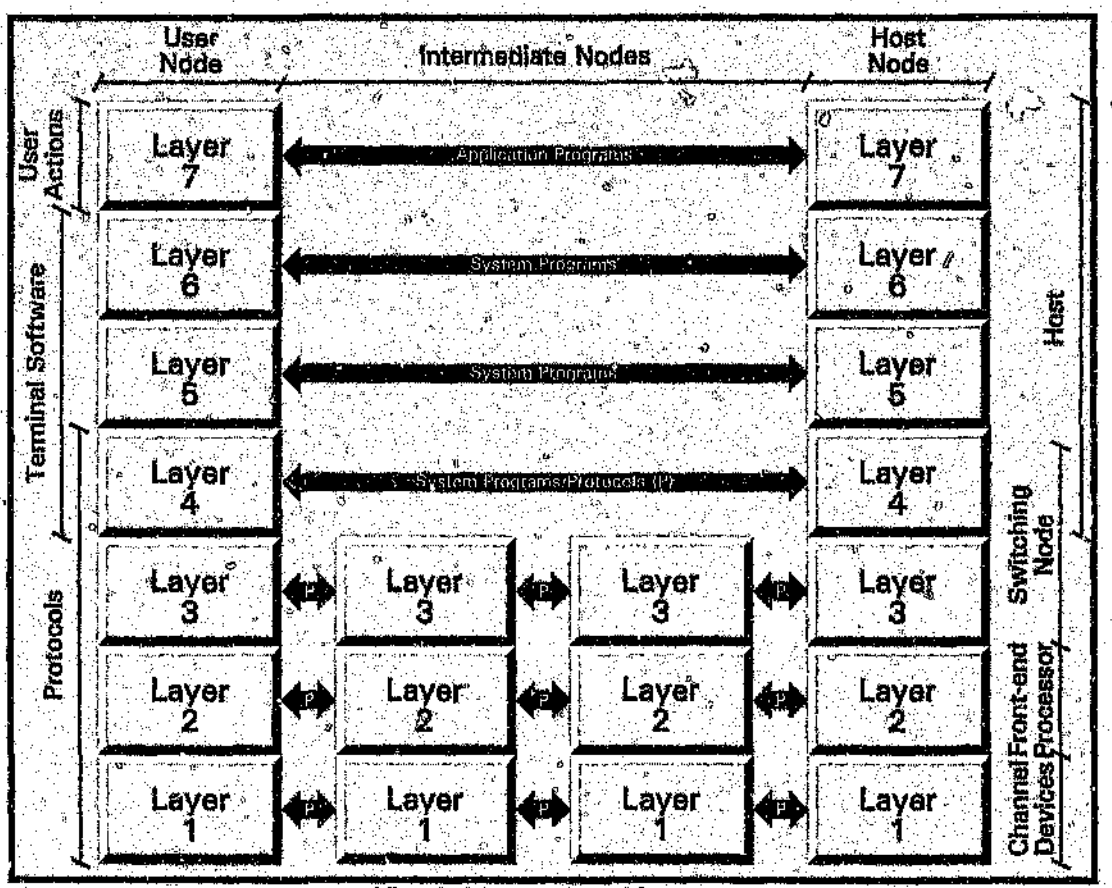
THE OSI MODEL The problem of incompatibility among datacom products affected users on a worldwide basis, so in 1978 the International Standards Organisation (ISO) announced a system of network protocols and named it the OSI model. OSI stands for Open Systems Interconnection. like SNA, the OSI model uses layers to define the logical and physical activities. Because of IBM's large customer base, SNA quickly became an unofficial standard, especially among users of IBM mainframes. However, DEC's network architecture, called DECnet, also has achieved a large following, primarily among users of



smaller scale systems. The OSI model has picked up support rather slowly. One reason is that protocols for all of the layers have evolved gradually--level by level. Protocols for the first three layers were announced in 1978 but did not receive widespread use until the mid-1980s. By all rights, the OSI model should become the true datacom architecture standard. In the meantime, many users will continue to support the computer manufacturers' architectures, which may incorporate gateways to make them compatible with the OSI model.

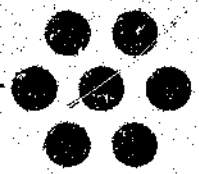
The basic idea of a layered architecture is to divide the architecture into small pieces. Each layer adds to the services provided by lower layers in such a manner that the highest layer is provided a full set of services to manage communications and run distributed applications. A basic principle is to ensure independence of layers by defining services provided by each layer to the next higher layer without defining how services are performed. This permits changes in a layer without affecting other layers. Prior to the use of layered protocol architectures, simple changes such as adding one terminal type to the list of those supported by an architecture often required changes to essentially all communications software at site.

The figure illustrates the OSI model.[18]



The OSI Layered Architecture

The host node is at the right, the user node is at the left, and the intermediate nodes, which include the front-end processor and the cluster control unit, are in between. The bottom three layers appear in all nodes, but the upper four layers appear in only the host and user nodes. The brackets at the right of the figure indicate which hardware units are used in performing the functions of the various layers. The brackets at the left indicate whether the functions are performed by user actions, terminal software, routines in the terminal ROM, or by protocols. The arrows linking the nodes at each layer are labeled to indicate the form of the linkage--application programs, system programs, and protocols



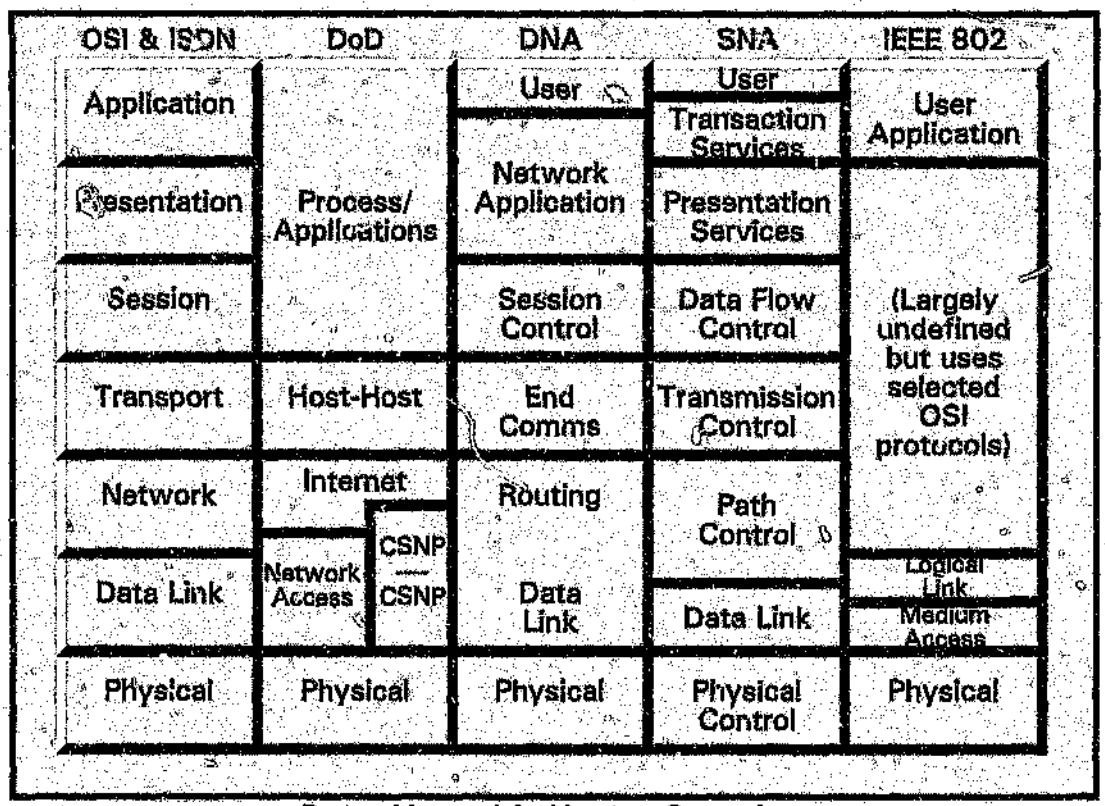
(P). A layer in one node 'talks' to the corresponding layer in another node. Each of the layers is described briefly below. The numbers below match the layer numbers in the figure.

1. Physical layer:	Transmits the data from one node to another.
2. Data link layer:	Formats the data into a record called a frame and performs error detection.
3. Network layer:	Causes the physical layer to transfer the frames from node to node.
4. Transport layer:	Enables the user and host nodes to communicate with each other. It also synchronises fast and slow speed equipment as well as overburdened and idle units.
5. Session layer:	Initiates, maintains, and terminates each session. A session consists of all the frames that comprise a particular activity, plus signals that identify the beginning and the end. A session is like a telephone call that begins with 'hello' and ends with 'good bye.' Standard log-on and user identification routines are used to initiate datacom sessions.
6. Presentation layer:	Formats the data so that it can be presented to the user or the host. For example, information to be displayed on the user's screen is formatted into the proper number of lines and number of characters per line.
7. Application layer:	Controls user input from the terminal and executes the user's application program in the host.

Let's assume that the user wants to use a mathematical model located in the host to select the best pricing strategy. The user indicates that the pricing model is to be used by entering instructions in the terminal, under control of layer seven (the application layer). Layer six (the presentation layer) changes the input data into the format used for transmission, and layer five (the session layer) initiates the session. Layer four (the transport layer) selects the route the message will follow in traveling from the user to the host node, and layers three and two (the network layer and data link layer respectively) cause the data to be transmitted through layer one (the physical layer). When the message reaches the host node, control moves up the layers to the application program (the model) in the host. Communication from the host back to the user follows the same pattern--down the layers in the host node, across to the user node, and up the layers in the user node.

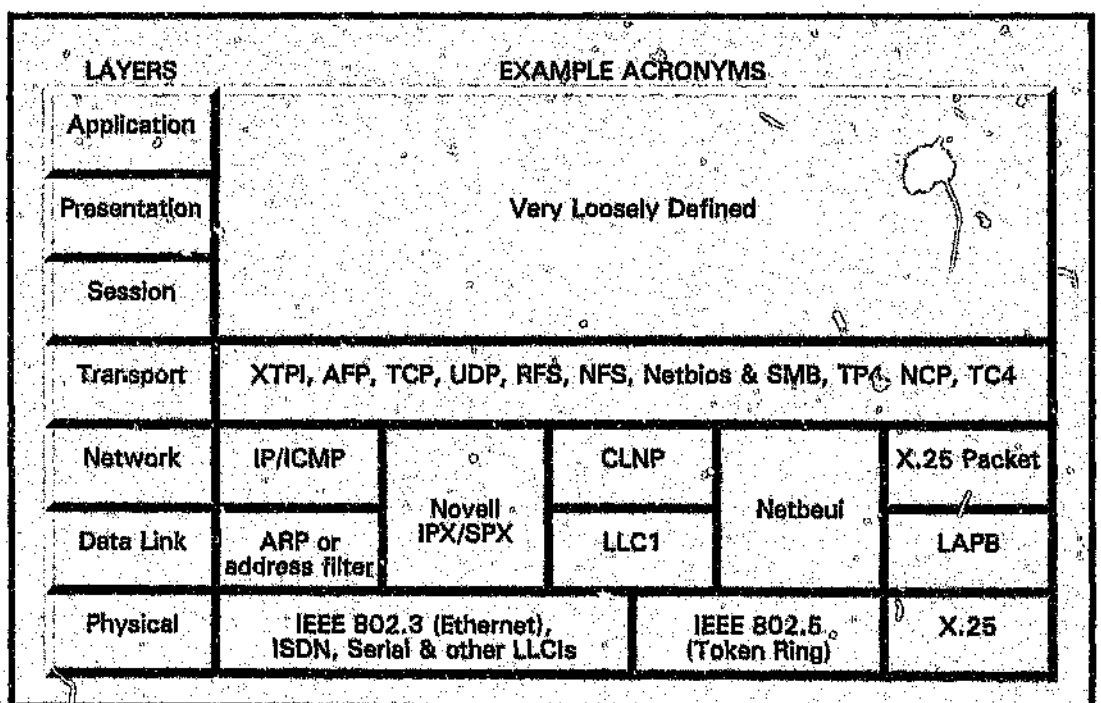
OTHER MODELS Several other layered standards exist, as mentioned above. The primary ones are OSI (covered), ISDN (Integrated Services Digital Network), DoD (US Department of Defense), DNA (Digital Network Architecture), SNA (IBM Network Architecture), and IEEE 802. These are compared to the OSI model in the following figure, but it must be emphasised that the figure should be interpreted with caution as the architecture differ considerably enough that it is impossible to give a completely valid comparison in this manner, and any more detail is beyond the scope of this report.

There are significant differences in the general orientation of the architectures, in flexibility for interfacing with different layers, in bypassing layers, in approaches to network control, and so forth.[19]

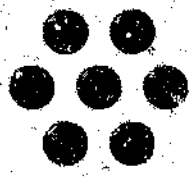


Protocol Layered Architecture Comparison

For the purposes of the following discussion, an illustration of some of the components typically found in each layer follows. Whilst a certain amount of component interchangeability is supposed to exist by virtue of the layered architecture, the diagram shows how there are some limitations with protocol suites being limited to the vertical stacks. These acronyms abound in the text and it helps to be able to place a few.



Protocol Component Examples



STOP-GAP TUNNELING Vendors now provide the necessary hooks for linking today's network operating systems with other environments (though often at extra cost). The most common approach is to enable a client or server to communicate over multiple network protocols simultaneously. A NetWare server, for example, can speak SPX/IPX or TCP/IP (middle layer communication protocols) with equal fluency to other servers. Someone sitting at a PC logged on to the network, for example, can see a NetWare server as the F: drive and a LAN Manager server as the G: drive.

The key to this capability lies in the device driver specifications espoused by Microsoft and Novell: Microsoft and 3Com's Network Driver Interface Specification (NDIS) and Novell's Open Datalink Interface (ODI).[20]

Device driver specifications also free hardware developers from the need to write separate drivers for each protocol stack. They can write one driver to a given specification, and the driver will work, at least theoretically, with every protocol stack supported by that specification.

Some systems use IP tunneling to communicate with others. Networking software in the client takes packets with the native proprietary protocol of the operating system, either Novell's SPX/IPX or Banyan's Vines Internet Protocol (VIP), and wraps the packets inside IP packets. The networking software then moves the encapsulated packets across a TCP/IP backbone for delivery to a node that can strip away the outer envelope and use the IPX or VIP packet. Alternatively, some use another type of tunnelling to move TCP/IP traffic across networks by encapsulating IP Packets within native packets. Tunneling allows TCP/IP applications and devices to use the standard IPX or native routers and other network services.[21]

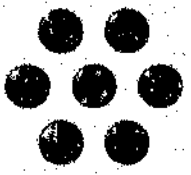
Unfortunately, not all protocols have standard headers, IPX being particularly troublesome, and this form of WAN interconnectivity falls over.

MULTI-PROTOCOL ROUTERS LANs were never designed to be interconnected; there is no mechanism for flow control. They can be bridged together (a join where all traffic passes between both networks) but the lack of flow control causes unpredictable responses and often total block-out. Bridging is also inflexible. Thus routers are becoming the norm, providing store and forward facilities to all individual packets to be directed to various destinations (like bridging but with a intelligent filtering and routing allowing it to pass only the desired packets across the gap).[22]

Routing is relatively straight forward if only one protocol is in use but with multiple protocols they have either different network layer protocols or, with some LANs, no routing protocol at all.

Multiple independent networks, one for each protocol, is prohibitively expensive, so that organisations need to lay in a single, common, backbone network and to use this to carry the multiple protocols. Encapsulating protocols into the protocol of the backbone e.g. XI routes X.25 over an SNA backbone or QLC routes SNA over a X.25 backbone, is practical but inefficient, particularly in handling polling and end-to-end flow control and pacing. This is very much a lowest common denominator approach with protocol-specific enhancements being lost in most cases.

The current answer is to install multi-protocol routers. These dominantly use an SNMP managed IP network and translate various inbound protocols, fully enabling the routing of the originating protocols (there are problems with protocols with a limited network layer such as IPX--thus eliminating the possibility of connecting Novell LANs in this manner). Unfortunately multi-protocol routing does not offer any inter-operation between the different systems and thus merely perpetuates the basic interconnection problems. By replacing all the different protocols with TCP/IP then only one protocol needs routing. In



practice SNA is the most viable suite, though DECnet will have to continue for current users so that two protocols will have to be supported, but not numerous options as at present.

THE RIGHT TRACK Multi-protocol routers aim to create a common wide area backbone network which can handle the connection of SNA to SNA, IPX to IPX, etc. They do not provide any interconnection between the conflicting systems. This is not a solution, it is another short term stop gap, the Open Systems concept is the only way forward. Fortunately the solution is at hand due to the unprecedented universal acceptance of the TCP/IP protocols. This older protocol has come to the fore on the back of Unix systems, but is now available on all systems from PC to mainframe. It has achieved the interconnection and inter-operation of heterogeneous systems promised by OSI.

The reasons that TCP/IP has succeeded were OSI failed are twofold:[23]
- It is common and not plagued by endless options;
- It provides a common set of user related services, telnet (a virtual terminal), ftp (file transfer) and smtp (a mail service). To this the Sun NFS (file server) and RPC/XDR (remote procedure call) have now been added and so at last we are addressing the real issues of interconnectivity as seen from a users viewpoint.

The basic theme then is that TCP/IP should become the dominant system, replacing all others wherever possible. It is not satisfactory that TCP/IP should be added alongside other protocols. Clearly this cannot be achieved over-night, if ever. It is unrealistic to expect large SNA users to abandon SNA in favour of TCP/IP, even though this option is available. DECnet and possibly APPN/AS400 users are more likely to change, but still slowly. Unix systems are no problem and so the practical focus must centre on LANs. Here also there is a huge installed base but these are not very satisfactory systems (despite the 'religious' fervour that they attract as with any immature technology) and the time is ripe for replacement. The magic difference between the LAN and mini/mainframe systems is that the latter can run multiple concurrent protocols but this is a major headache with LANs because of PC DOS limitations.

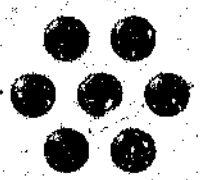
Since PCs have faster screen-fill rates than ASCII VDUs they have a better user interface. However they also encourage 'playing' rather than working and present a high cost of ownership due to the complexities of networks compared to minicomputers. Thus LANs are the better solution for 'work groups', all doing similar activities, in a corporate environment with MIS support, while time-shared systems with VDUs are better for general purpose business computing. It is almost immoral to inflict a PC LAN on small business system user when multi-user PCs are readily available.

The problem with replacing existing LAN software, still using the same hardware, is quite simply the scale of the installed base but it can and probably should be done. The method adopted must be to build pilot systems first in order to establish preference for applications and development tools and to filter out those DOS programs which are so hardware dependent that they must be eliminated. It is a matter of balance.

The opportunity to replace LAN software is ripe now, although some would prefer the focus to be on pilots until the end of the year with major replacement programs the following year. The winds of change are fanned by two major events: one is the eventual arrival of OS/2.2 and possible competitors to replace PC DOS while the second is the general realisation that client/server computing is the way forward.

SOLUTION The solution seems to lie with IBM's new outlook on life--the realisation that they are no longer the only suppliers and that trying to force their proprietary systems on the market place is no longer acceptable.

A combination of a new protocol suite, as well as a networking sturcture combine in what seems to be a very likely working combination. Both are discussed separately.



THE IBM NETWORKING BLUEPRINT[24] It is difficult for specific divisions of IBM to create a personality of their own. PCs and peripherals such as printers and disc drives are relatively easy, but the other divisions are more tightly coupled. And they are strongly centered on mainframe computing. This of course is the key problem IBM have to face up to, that the mainframe is no longer the major centre of the computer world, but one element (albeit a very significant one still) Distributed systems, both midrange systems and PC networks are equally common. Thus there is a general trend from 'mainframe-centric' to 'network-centric' concepts. Properly controlled, this new focus encourages 'rightsizing' as opposed to 'downsizing', with systems being built around the most appropriate technology. This is a far cry from the current 'Islands a Technology' which now exist, with little to no integration.

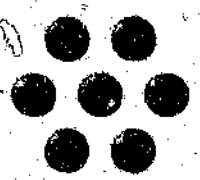
It follows then that IBM Network Systems division must play a ever increasing role in IBM's future. Further they must establish an individual identity since they must continue to support the huge existing IBM customer base but they must also establish themselves as 'open connectivity solution' vendors, without detriment to their SNA users. Establishing an identity not tied to the old IBM will be more difficult than supplying open products. Indeed IBM Network Systems already provide as wide, if not a wider range, of connectivity products than any other supplier. The range of TCP/IP products is amazingly comprehensive and support for Ethernet (as well as token-ring) is good, which is more than can be said for Ethernet specialists in supporting token-ring. The 6611 for example is a general purpose multi-protocol router aimed at nonIBM as well as IBM customers, although the latter are naturally favoured. IBM Network systems even support OSI on mainframe systems since some users still exist; this is a thankless task with all the various options and profiles to support.

The key to creating a unique identity for the Network System division lies in the Network Blueprint. This model, produced about a year ago, is in fact quite revolutionary, because while it starts from an existing SNA model, it is directly aimed at providing a flexible method for providing all existing services such as CICS, across the dominant protocols of SNA and TCP/IP. It is also clearly defined to encompass APPN as an upgrade for SNA users and provides support, where possible, for the troublesome LAN protocols Netbeui and IPX.

It is very easy to miss the significance of the Blueprint, particularly for the mass of SNA users, in that it integrates the communication models in one architecture. Thus both SNA and TCP/IP protocols are embedded in new releases of VTAM in contrast to the separate SNA (VTAM) and TCP/IP protocol stacks in current use. The concept of separate stacks means that extra software products are needed to provide access to applications and services which normally reside with an alternative communications system. As an example, an interface product is required to provide a telnet session from the TCP/IP address space into VTAM in order to run CICS applications; access to enhanced CICS functions is not normally available. However with TCP/IP now implemented in VTAM alongside SNA protocols, a common API is presented (CPI-C) to all the normal VTAM supported systems.

The open concept does not stop here, however, because normal TCP/IP applications are now supported on the SNA protocols because the Berkeley Sockets API (the common one in TCP/IP networks) is implemented alongside CPI-C. This is referred to as Multi-protocol Trancport Network (MPTN). Clearly when APPN is established for VTAM it too will be integrated alongside , TCP/IP with the same consistent APIs.

Integrating existing systems with open standards has been a goal of many suppliers for years with little success, now IBM have a much more satisfactory product for their customers. They key to this rather startling success by IBM (apart from the new commitment by Network Systems) is the three tier architecture of the Blueprint. The seven layer model of the ISO Open System Interconnect (OSI) concept has been too complex to integrate with existing, incompatible systems. IBM now conceive a model with applications interfacing with communications via the common API of CPI-C, Sockets



and MQI as the top layer (Application Support) alternative protocol stacks (roughly the network and some transport layering from the OSI model) at the middle layer (Transport Network), with a variety of datalink services such as Token-ring, Ethernet, Frame Relay, SDLC/HDLC, etc. (Subnetworks). This model is achievable in practical environments and both support existing systems and is open to adding new technologies.

IBM have often made claim to 'openness' which has equally often been challenged in the past because they did not integrate systems. Now it will be very difficult to provide a more open communications environment than that offered to mainframe users. It now remains to see what is implemented for other systems along the same lines.

ADVANCED PEER TO PEER NETWORKING PROTOCOL[25] Advanced Peer to Peer Networking (APPN) is IBM's strategic communication protocol for the future. It will eventually replace the current hierarchical form of SNA but it remains to be seen whether it can ever gain the market dominance achieved by the older protocols. This is not related to quality but to IBM's less than dominant position today.

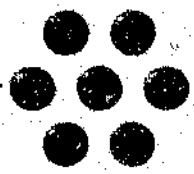
Superficially (and in most peoples minds) APPN is the AS/400 communication protocol; the fact that it is strategic and far more all-embracing is not yet accepted. Indeed some current SNA users may well be thinking in terms of staying with their current products and switching to alternatives other than APPN via the downsizing route.

Initially APPN was introduced on System/36 machines (in a much reduced form to the current products), as a reaction to DECnet. Historically IBM made mainframes and DEC minicomputers so that it followed naturally that SNA was hierarchical (focusing on the central mainframe) while DECnet was peer-to-peer (creating a big system from a set of smaller ones). Thus as first minis and then PCs became more attractive, so the demand for peer-to-peer communications grew.

The first major APPN products were embedded in OS/400 to enable networks of AS/400s to compete with networked VAX/VMS systems. All these mid-range machines could still connect into an SNA network by emulating end-nodes such as 3270s (PU2/LU2) and the key conversation protocol LU6. 2 via a PU2. 1 node. Here however APPN has a big advantage over DECnet for SNA users since while the SNA and APPN networking protocols are different, the same LU6. 2 user protocol is employed in both network types (not so however for terminals.

While the battle raged between AS/400 and APPN versus VAX/VMS and DECnet, PC networks grew apace, with much simpler protocols inspired by the old Xerox XNS, dominantly Netware IPX/SPX and IBM/Microsoft Netbeui. Neither IPX nor Netbeui are suitable for corporate networks, which would in theory be much better served by APPN or DECnet. DEC do use DECnet in their Digital Patchworks PC LAN products but APPN is yet to make an appearance outside the AS/400 for IBM; neither has won any market share, despite the serious current problems with inter-networking PC LANs. But probably the most significant impact on communications has come from the Unix world. They adopted the old Arpanet TCP/IP protocols--older than either SNA or DECnet--which then lead to inter working of equipment from various Unix vendors including IBM and DEC. This then had the knock-on effect of support for TCP/IP on most other products, including the existing mainframe and midrange systems, but probably of more long term significance in the PC LAN arena. Thus it is TCP/IP that has become the challenge to SNA and not DECnet. Indeed both SNA and DECnet must be usurped by TCP/IP which is more of a problem for DEC than IBM. The relative failure of OSI is a further blow to DEC since with DECnet phase V they backed the loser (OSI) and missed the winner (TCP/IP). Today IBM are well ahead with a very comprehensive TCP/IP product set. With the latest release of VTAM TCP/IP now available as a direct alternative to SNA for all major S/390 services.

TCP/IP is thus IBM's ally. Now they can continue to enhance the traditional SNA systems by replacing the old hierarchical networking with APPN. In other words making future releases of SNA based on APPN, making mainframes a peer member of distributed



networked. APPN could even have a significant impact on PC networks yet, which need the peer-to-peer networking for simplicity and stand-alone subsystems. IBM made some early mistakes in positioning APPN by refusing to publish the protocols but now they are licensing them out, encouraging other vendors to fully implement APPN. Note then that whole networks could be running on APPN without any IBM kit in the network since the central host is no longer mandatory, in direct competition to TCP/IP. IBM can exploit this concept to keep their existing S/390 customers happy. Full APPN on a mainframe, in VTAM, will appear sometime in 1994, integrated with the TCP/IP networking; interim products may well appear before the fully multi-protocol integrated version is released.

Currently the IP routing protocol, RIP, has been adopted for the backbone of multi-protocol routers, including the IBM 6611. However APPN is also used as a backbone by the 6611 and rumour has it that IBM have High Performance Routing (HPR) integrated with next releases of APPN (APPN+) which significantly outperforms RIP as a backbone.

MANAGEMENT PROTOCOLS

PROBLEM Even when the clients and servers can communicate, share data and work together, one challenge remains: network management. The need to manage a variety of client and server systems introduces complexity at every level.

In the traditional network management approach, a network comprises many instances of many types of managed objects. There are physically managed objects, such as modems, multiplexes, bridges, and routers, and logically managed objects, such as the IP code that runs in the routers or in the host nodes.

Administrators of such heterogeneous networks also face the problem of managing an assortment of user directories. If all servers cannot, for example, share the same set of user names and passwords and automatically propagate user changes across all servers, then every new user or even new password will send administrators scurrying to change directory databases.

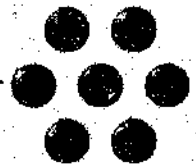
Users also will pay if the different types of servers cannot share directory information. A user who wants to work both with files on a NetWare server and with data in a database server on a Unix system might have to remember and maintain different log-in IDs and passwords for each environment.

Finally, administrators who have to deal with a variety of servers face the usual range of server management problems. Such fundamental procedures as backup and periodic maintenance will be necessary on every server. If each server provides different tools and procedures for those tasks, administrators will soon yearn for simpler days when options were fewer.

For most network managers, the PC remains a tantalising but unreachable source of crucial information that would help solve many networking problems. Without this information, the network data gathered from a sundry of devices -bridges, routers, UPSs (uninterruptible power supplies), and wiring hubs, to name a few--remains decidedly incomplete.

Proprietary methods are available for managing the desktop, but no proprietary method adequately integrates desktop information with data collected from other network devices. Without the ability to correlate information from all network devices, a manager's view of the network is imperfect at best.

At the simplest level, all the systems should support some basic network-management protocol (the upper layers of the layered protocol architecture), such as Simple Network Management Protocol (SNMP). This is necessary to allow administrators to spot and



handle such basic problems as which servers are down, which clients or servers are placing unreasonably large loads on the network, and so on.

CURRENT SITUATION

The management problem is solved by a combination of managers and agents. An object is managed by associating agent code with it. Agent code consists of management instrumentation (such as status variables, counters, timers, and test routines) that lets the managed object be monitored and controlled, plus an implementation of some management protocol (such as SNMP and CMIP Common Management Information Protocol). The management instrumentation typically conforms to a particular Structure of Management Information (SMI). An SMI specifies the rules used to define the information in a managed object so it can be monitored and manipulated through the management protocol. The SMI is analogous to a programming language that a programmer uses to declare data structures and to permit operations that can be performed on those data structures (such as the functions and procedures).

The combination of a particular protocol with an SMI forms what is called a network management framework. The Internet community has defined the Internet Standard Network Management Framework (ISNMF), which is used by nearly everyone in the SNMP community. The Network Management Forum (NMF) has also defined a management framework with the NMF Release 1 specifications. This framework uses CMIP, and it has been implemented by several vendors (AT&T, BT, HP, IBM, DEC, Timeplex, Unisys, France Telecom, and STET). This broad support allows interoperability among the different vendors' network management systems.[26]

SNMP and CMIP define three components; an agent, a console, and a protocol. The agent is code developed by the vendor that resides in a particular device. It gathers network information from the product and stores that data in a database. The format for this database, called the Management Information Base (MIB) in SNMP parlance, is specified by the SNMP specification. The console is where network managers request and display information from the agent. Connecting the console and the agent is the protocol, or the language for requesting networking information from and returning it back to the console.

Although SNMP and CMIP let network managers gather information across the enterprise from a range of devices, neither have been sufficiently implemented at the desktop, partly because they consume too much space.

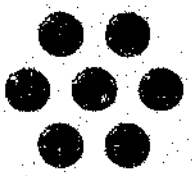
These competing standards are each designed to manage network devices from a single application or console. Although the most widely used, SNMP lacks many of the advanced features offered by the US government-endorsed CMIP. However, new SNMP standards under development should help it catch up with CMIP.

Coming to terms with network management systems means understanding these acronyms. Manufacturers tout support for a standard as a major feature of their products, but the question of which standards will dominate the market remains to be answered. If one plans to base buying decisions solely on standards support, one runs the risk of not getting the best tool for the job.

SNMP SNMP dominates today's market. In its survey of 400 network managers at Fortune 1000 and government sites, the Business Research Group (BRG), found that 34 percent of the sites were using SNMP.[27] Only 14 percent were using CMIP.

Developed in 1987 to manage routers on the Internet, SNMP was used in commercial products before CMIP hit the US market. SNMP was built on TCP/IP, a transport protocol that is still widely used in heterogeneous networks.

Jeff Case, president of SNMP Research, and one of the original authors of SNMP, says its continued popularity is no accident--it's easier to implement than CMIP.
"Right now, there are at least nine different profiles for implementing CMIP. With SNMP there's one way to do things--no options, no alternatives." As a result, buyers of SNMP-



based consoles, agents, and management information bases (MIBs) can be certain these components will be capable of exchanging information.[28]

Simplicity can also be a drawback, because it invites elaboration. Even when standards are created, they are violated as soon as a new feature is added by a vendor.[29] In SNMP, the violation takes the form of private extensions to MIBs and incomplete standard MIBs. A MIB is an object-specific database that defines what can be monitored or controlled on a device such as a router or a server. SNMP defines a standard MIB that, in theory, lets any network management console work with any agent and MIB. But vendors routinely omit parts of the standard MIB or use extensions that other consoles can't support.[30]

Perhaps the biggest strike against SNMP is its lack of security. Passwords for configuring routers, hubs, and servers are passed across the network in a single unprotected packet. That means anyone with a mischievous or malicious bent could gain access to those passwords and wreak havoc with the network devices. Another limitation of SNMP is that it currently includes support only for TCP/IP networks, although vendors have tweaked SNMP to run on other network transports, notably Novell's Internetwork Packet Exchange (IPX) and Apple's AppleTalk.

Analysts, vendors, and users agree, therefore, that it is unrealistic to dismiss CMIP. "I think the reality of the market is that both of these protocols will exist," says Sanjiv Ahuja, manager of network-management services for IBM. "It's not an either/or question."[31]

CMIP Established by the International Standards Organisation, US government approved CMIP is part of that group's Open Systems Interconnection (OSI) suite of communications protocols.

CMIP offers many features that SNMP lacks. One version called CMOT, for CMIP over TCP/IP, runs on TCP/IP, and a transport-independent version called CMOL, for CMIP over Logical Link Control, runs at the Logical Link Control layer, freeing the protocol to run on any network, regardless of transport.[32]

A security feature is also included in CMIP, but it is tied to the OSI general security model, which is still incomplete.

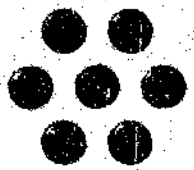
A CMIP console also can establish a session with a device and download many different pieces of information--all error statistics, for example--with one command. In contrast, SNMP must query for each piece of information.

Despite the appeal of its features and the backing of the federal government--which specifies that all contractors must eventually support OSI--CMIP has found little acceptance among vendors and users. One reason is that the specification is very broad and leaves lots of room for interpretation when it comes time to implement the protocol.

Still, CMIP is likely to become popular in at least one area: the integration of network and telecommunications management. Telecommunications equipment makers will probably standardise on CMIP because of the federal government's involvement in telephone regulation.

Part of SNMP's continued dominance over CMIP and other protocols is its flexibility. Missing features can be added to SNMP through a standards process that typically takes a year or two. Changes to the core CMIP specifications, on the other hand, must wait on OSI's approval cycles, which take a minimum of seven years.

In fact, the Internet Engineering Task Force (IETF), which oversees SNMP, is currently formulating two additions to SNMP: the Remote Monitoring (RMON) MIB and SNMP version 2 (SNMPII).[33]



RMON defines a MIB that lets active network monitors continuously feed information about network health and traffic into an SNMP-based console without waiting for a request.

SNMPII, now pending before IETF, adds security through encryption and authentication, can run on multiple network-transport protocols, and adds bulk data retrieval and features for hierarchical management, but isn't expected before late 1994.

SOLUTION--DMI SNMP now manages mostly network devices, such as routers and hubs. But there is increasing interest in managing PCs and the applications running on them. As a result several vendors, headed by Digital Equipment, IBM, Intel, Microsoft, Novell, Sun Microsystems, Synoptic Communications, and nearly 200 other vendors, a consortium which may just have the clout to turn out an industry standard, formed the Desktop Management Task Force (DMTF) last year. The charter of DMTF is to define a standard way for PC hardware, operating systems, and applications to share information with LAN management programs, whether those programs are proprietary or based on SNMP.[34]

Dubbed the Desktop Management Interface (DMI), the specification aims at creating a standard management agent that can gather from data any DOS-based PC. At a minimum, DMI will identify the manufacturer, name, version, serial number (if appropriate), and installation date and time of a component. More extensive implementations will extend these entries and provide a host of information such as the amount of memory installed and problems with the device.

With this information, network managers can be smarter about the way they troubleshoot and maintain PC-centric LANs. By combining this information with data gathered from bridges, routers, and wiring hubs, network managers can for the first time begin to diagnose enterprise-wide problems without leaving their desks. They will, for example, be able to determine why two users on separate networks can't communicate.

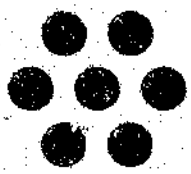
The current draft of DMI (released in March '93) defines a management agent for DOS and DOS/Windows workstations. Drafts for Macintosh, OS/2, Unix, and other environments will follow. The final specification is expected later this year.

The potential benefits are enormous but not without cost. Though managers won't have to pay to gain DMI functionality, they will need to upgrade their existing clients, which will be a substantial investment. What's more, buyers must be careful about the level of DMI support they purchase. Some DMI implementations will gather more data than others.

Vendors will also find that even proficient assembler programmers may have difficulty writing the code necessary for building the agent. This could further delay DMI's acceptance, though with DMI's potential benefits, most vendors expect to support DMI by year's end.

DMI hopes to change that. Like SNMP and CMIP, DMI presupposes three elements: an agent collecting network management information inside the user's PC, a management console where the data is displayed and requests are issued, and a communications protocol that allows the management station to query the agent.

Unlike SNMP and CMIP, however, DMI only defines the architecture of the agent. And, different from SNMP or CMIP agents, the DMI agent consists of a main component, a TSR, and overlays. The main component resides permanently in memory and consumes about 6Kb. Overlays are individual modules that are called by the main agent for managing the different components in the PC. These components may be hardware (such as a hard disk) or even software (such as a specific application). When a LAN administrator issues a DMI command, the main agent in the PC automatically loads the overlay for the specific component.[35]



Although software, hardware, and other elements of a PC are important, network adapters remain the prime candidates for LAN management software. This is primarily because most client problem-solving begins with determining whether the PC is connected to the LAN. With DMI, LAN administrators will be able to query the general health of all network adapters on the LAN, as well as query their traffic statistics.

There are three portions of a main agent. First are the files storing information about each component. The specification for the way in which this information is formatted is called the Management Information Format (MIF). This is roughly equivalent to SNMP's MIB definition. Below the MIF files is the second component, the Component Interface (CI), which defines how the main agent can issue commands to query, reset, or control components. And above the MIF files is the Management Interface (MI), which lets a LAN product vendor query, control access, and be notified when a threshold is exceeded (this is called an Event Notification in DMI parlance and an Alert in SNMP lingo).

A management console can use three commands to query an agent: List, Get, and Set. A management application will use one or more of the List commands to query a PC's MIF files to determine what components are installed in that machine. Other List commands let the management application discover the functions of a particular component. Get allows the management station to request individual statistics from a component, and Set allows the management station to alter that statistic.

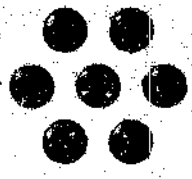
While DMI deals with MIF files and component-management commands in great detail, it doesn't specify how the information and the commands cross the LAN. Information and commands will cross the LAN in any number of ways: via SNMP or CMIP, or even by writing directly to a network layer protocol such as IP or NetWare's IPX. Many users will want some kind of mapping between SNMP and DMI so they can query DMI agents with their existing SNMP-based management stations. This will then allow network managers to integrate data collected from the PC with data collected from other network devices. To date, this layer has not been defined, though it is expected to be completed by year's end.[36]

THE DRAWBACKS OF DMI

As one would expect, LAN management software will need additional RAM over and above the 6K used by the TSR in order to send and receive management requests and responses. Even then, frugal designers and skilled assembler programmers will be needed to keep the size of the MIF transport software to a minimum.

Even a skilled assembler programmer, though, will find the TSR overlay management scheme a tricky environment in which to implement the processing of MIF language commands, although the DMTF will supply program templates and sample source code. DMTF's TSR specification may present problems. The TSR, for example, will have to serialise access to the program's overlay area, which is where the agent loads and unloads modules for managing individual components. This is important to prevent two requests for component module information from colliding with each other. Programmers aren't perfect and neither are the software modules they write. After LAN vendors iron out the initial wrinkles and fine-tune their DMI software, however, one'll most likely find DMI-aware LAN products the rule rather than the exception in most organisations.[37]

What's more, not all the information in the MIF files will be generated completely by the computer. And the amount of work it will take for one to enter manually the reams of information about the networked PCs is daunting, to say the least. LAN vendors will be able to supply component-specific MIF-file data when one buys new products, but the PCs will have to be upgraded, as will network adapters, hard disks, and other components the company already owns. Moreover, the company will need to design and implement a plan for gathering and entering MIF data if it wants to take full advantage of DMI. A remote-control text editor would be helpful, but such a product has not been defined by the specification.



The success of the DMTF's efforts will depend on two factors. Suppliers of LAN products will need to update LAN drivers and perhaps redesign hardware so that they are DMI-aware. LAN management software vendors will need to add MIF language processing to their products.

Both criteria will be met if the DMTF vendors support the specification.

CONCLUSION DMI fills a large, gaping hole in network management: the administration and control of PC workstations. All LAN vendors will eventually have to support DMI if they expect to sell hardware and software to companies with PC-based LANs.

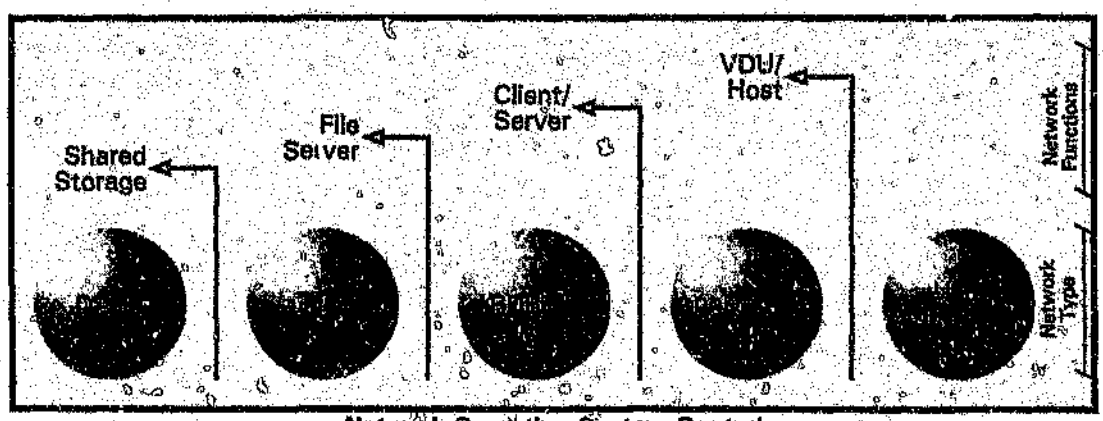
With those sentiments and the backing of the leading networking vendors, DMI will almost certainly catch on as a standard. A better alternative doesn't exist.

In order for DMI to succeed fully, however, NT, OS/2, Unix, and Windows clients must be supported. With these environments, as with DOS, one can expect to find the DMI TSR functions included in the operating system.

Not every device and component will be instantly DMI-compliant and DMI-aware, of course. At first, the PC will load the DMTF-supplied TSR and only a single component module, perhaps one for the PC itself. Later, one will install a new driver for the network adapter that is DMI-compliant. Eventually, if DMI catches on, one will have component modules that DMI management stations can use to gather information on every device in the PC. When that day arrives, the LAN administrator will spend less time walking around the office and more time actually solving problems.

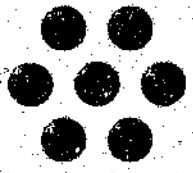
NETWORK OPERATING SYSTEM (NOS)

BACKGROUND The extent of the network operating system's control over the network has progressed over time. Initially starting as a central file storage point accessed by workstations (typified by the Novell NetWare LAN), it has now progressed to facilitate client/server computing as well as terminal/host computing as initially typified by mainframes. These capabilities are a function of the NOS and are illustrated below.



Network Operating System Control

THE NOS VS THE PC OS Network operating systems, as we know them today, are in the midst of a fundamental change. Instead of two distinct products--the network operating system and the operating system--the next-generation operating systems will offer built-in networking hooks. The network operating system may become a server solution only, while one will get the client PC hooks from the PC OS. Microsoft reportedly will include intrinsic networking capabilities in a future version of Windows, taking advantage of a Windows feature-called Service Provided Interfaces to create dynamic link libraries (DLLs) that support the directory services of various network operating systems.



Today, fully integrating the LAN Manager, NetWare, or Vines client PC software into a PC running Windows involves loading patches, using third-party products, and many other work-arounds. The idea of a universal Windows client, perhaps integrated into the server software with a special software option, as in the Macintosh today, certainly appeals to Microsoft, and it presents no problems for Banyan Systems with its unique enterprise architecture. To Novell, however, this approach means losing any presence in the client PC.

NOSs FOR DATABASE SERVERS

While the client may undergo a radical face-lift, server software will likely undergo its own changes. Most radical of these will probably be the introduction of network operating systems that can take advantage of multiple processors.

Today a network operating system's file and print server functions typically do not stress the server's CPU. Instead, servers are limited by the time needed to access the hard disk and move the data to and from the adapter cards. But when one adds network management functions and database management software to the server, the CPU can become the site of the bottleneck.

One approach to the problem is to develop faster CPUs. Intel's 586 (their latest advance on the x86 chip-set) is surely a step in this direction. Another approach is to move the network operating system onto a computer with multiple CPUs, such as one of Compaq's Systempro line. But the developers of LAN Manager and NetWare have almost ignored this hardware capability; only Vines can balance the processing load between two CPUs on specific hardware platforms. Microsoft states that its planned Windows NT operating system is designed for symmetrical multiprocessing.

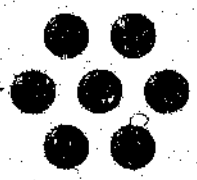
Another area of performance improvement consideration is that of 32-bit operating systems versus older 16-bit. The former provides more power, speed, process control and data accessibility.

While both NetWare and OS/2 2.0 support 32-bit operations, most OS/2-hosted SQL databases are currently based on the 16-bit OS/2 1.x API (Application Programming Interface). Most NetWare-hosted SQL databases are true 32-bit applications. There are other fundamental differences between the two platforms. SQL databases designed as NetWare Loadable Modules (NLMs) tend to be faster than those that run under OS/2. One reason is that they are dynamically linked to and become part of the operating system; there is no layer of software between the application and the hardware. But a misbehaving NLM (NetWare Loadable Module) can crash the entire system. On the OS/2 platform, the operating system sits between the hardware and applications, servicing requests for resources. The mediation of the operating system adds overhead and so exacts a speed penalty, but it does protect against ill-behaved applications.[38]

Another reason NLMs are faster is that NetWare is non-preemptive; each process is allowed to run to completion, uninterrupted. It is the responsibility of the NLM to release control of system resources at convenient times to give other processes access to those resources. In fact, this is a formal requirement for Novell certification of NLMs. OS/2, on the other hand, is a preemptive operating system; OS/2 has responsibility for managing system resources. There is nothing to prevent OS/2 from interrupting a process at an inconvenient time, thereby slowing the completion of the process. The difference between the two approaches boils down to speed versus safety.

Unix is a 32-bit operating system that's been available on PCs for nearly a decade. Highly scalable, there is a version of Unix available for almost any processor one can imagine, from a desktop computer to a room-filling mainframe. In addition to maturity and scalability, Unix also boasts the unique advantages of multiple-user and multiple-processor support.

The ubiquitousness of Unix on computers of all sizes makes the move to a bigger box relatively painless, since virtually all the same tools and procedures can be retained. Just



about every major SQL database vendor, including Gupta Technologies, Informix, Ingres, Oracle, and Sybase, has a version of its product that runs on Unix. In many cases, Unix was the SQL database's first platform; versions for NetWare and OS/2 were later additions. In fact, vendors often update the Unix versions of their products before other versions, with the result that the Unix product supports a more advanced feature set.

Multiple-processor support is the main key advantage offered by many versions of Unix. For a well-tuned SQL database, the processing power of the server is often the last remaining constraint. On OS/2 and NetWare systems, one can add more disk drives and more RAM, but one can't add more CPU power; Unix doesn't suffer from this limitation. Some Unix boxes can support as many as 30 processors. Microsoft's NT operating system, released in the second half of 199_, also supports multiple processors and multiple platforms. But until NT proves its viability, Unix is the only PC operating system with these capabilities.

Like OS/2, Unix is a preemptive, multitasking operating system. (NetWare is not preemptive; the NLMs themselves are responsible for relinquishing system resources so other applications can have access). But unlike either OS/2 or NetWare, Unix is also a multi-user operating system. With Unix, one can store both the 'back end' and the 'front end' on the same machine, and access both via a dumb terminal. This can be a very efficient configuration when a user must connect remotely and the client application is passing a lot of data back and forth to the server. When multiple processors are used on the server, the increased load can be handled easily.

While there are advantages to Unix, they come at a price. Unix hardware and software is usually more expensive than systems destined for either OS/2 or NetWare. Because of the complexity of Unix and the arcane (by today's standards) administrative facilities, the same can be said for the administrative and training resources necessary to run a Unix system at peak performance. However, for state-of-the-art SQL database systems, Unix warrants a serious look.

CRITERIA The major criteria are listed below. Several lesser features are illustrated in the 'Reviews' section.[39]

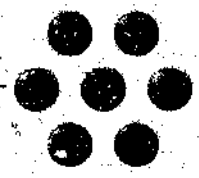
FAULT TOLERANCE Going global means that resources must be available when a user needs them. It also entails ensuring that the network can function with the reliability of a phone service. At the heart of that service is server reliability. Fault tolerance is an emerging feature for all three network operating systems.

Safety features range from monitoring the operational status of an uninterruptible power supply attached to the server, to mirroring disk drives or drive controllers to ensure continued operation if one fails, to server duplexing, where one runs a separate computer as a constant hot standby for fail-safe operation.

NETWORK MANAGEMENT With networks growing in size and complexity, network managers need a way to control and monitor an enterprise-wide network better. One of the developments during the past year in network management was Novell's Hub Management Interface (HMI). HMI is a specification that integrates the wiring hub, the center of many Ethernet and Token-Ring cabling schemes, and the file server into a common device.

Putting the wiring hub inside the server reduces costs because there is no cabinet or power supply and the server's CPU runs the hub management software. Wiring hubs also make it easier for vendors to ship ready-to-run networks, complete with pre configured file servers.

Depending on the configuration, however, there may be drawbacks to integrating the hub and server: The hub/server can create a central point of failure. Nevertheless, wiring hubs enable the correlation of network operating system information with information about the



physical infrastructure. If the network is failing, a management application will be able to determine whether it is the server or the network itself that is causing the error.

RELIABILITY & RESILIENCY OSI (Open Systems Interconnection) is a set of standards for communication of data. It is often better if one can standardise on one protocol such as TCP/IP (Transmission COntrol Protocol/Internet Protocol) and IPX (Internex Packet Exchange), but there is often a need for DECnet and SNA (Simple Network Architecture) protocols. In connecting any data station with any other, it is vital that it is secured against failures or problems. This could be caused by line congestion and it is necessary to remute without any interference by users.

NETWORK DESIGN It is vital to adopt a well thought out strategy for the network. This should involve decisions at four levels. The first is the access level, the level at which users gain access to the system. The second level is that of the work group, how users who work together share data and resources, and how they are linked for this purpose. Thirdly the distribution level, which is how the various work groups, perhaps downsized divisions of the new organisation, connect with each other, possibly across remote geographic areas.

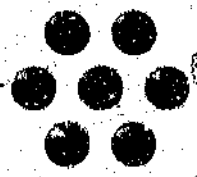
Finally, the core systems, that support the basis on which all the networking takes place. It is like the vision of a totally automated country in which any individual at home can get into a local suburban system, then into a city-wide system, then into a national communications grid, and finally into the world grid. At each level there are different communication needs, in terms of speed of transfer and the amount of traffic that can be supported.

ROUTING ALGORITHMS With the growth of networks and their interconnections at the distribution level, there are often many alternative paths for a message to move from one user to another. The mechanism for choosing which path is best is called the routing algorithm and should be based upon the choice of the basic carrier, the number of alternative paths, the congestion on the lines, and the speed of the different paths available.

MIGRATION To move from main frame communications to the networked environment means planning carefully. Often it is required to integrate the SNA mainframe network into the LAN through multi-protocol converters and routers. And it is generally advisable to plan for parallel operations as far as is possible.

REVIEWS 85% of the LAN (Local Area Network) market is dominated by Novell's NetWare, Microsoft's LAN manager, and Banyan's Vines. Digital's Pathworks is fast becoming a force capable of uniting NOS disparities, and is also a strong contender for future market dominance.[40] This limited range, the import of the NOS to a system's functioning, maintenance, and power, and the fact that NOSs are widely divergent, makes it easier to provide a comprehensive review of each product, than to compile a feature chart that would be able to convey the meanings of the complex terminologies. Hence the following is a list of the more widely reviewed features found in each system, followed by a chart that is possibly over-simplistic when read alone, but summarizes what is stated in the main text.[41]

NETWARE V4.0 Introduced by Novell in 1983, the first version of NetWare offered basic file and print services for small departments. Several years and revisions later, NetWare v2.x and v3.11 dominated the market with improved security; high reliability; support for many workstations, including DOS, OS/2, Macintosh and Unix machines; and support for many protocols, including SPX/IPX, TCP/IP, OSI and AppleTalk. NetWare v3.11 provided the highest performance of any comparable network, but it lacked server domains, directory services, efficient WAN support and protection against server applications. Novell also faced strong competition from such vendors as Digital, Microsoft and Banyan.



With NetWare v4.0, Novell is attempting to address the concerns of corporate users and ensure its market dominance.

Server Architecture: At the heart of NetWare v4.0 is the NetWare Operating System, which provides a dedicated server environment. The NetWare OS performs basic operating system functions such as allocating memory for processes, scheduling tasks and providing access to the file system. One of the highlights of the NetWare OS is its modular design. This architecture allows the dynamic loading of software modules called NLMs (discussed earlier).

File Service Protocols: NetWare v4.0 supports DOS, Windows and OS/2 clients by using the Internet Packet Exchange (IPX) protocol and the NetWare Core Protocol (NCP). NC functions at the upper layers of the OSI model and is used to establish, maintain and terminate network sessions. NC messages are routed by IPX, which uses a data gram delivery system. IPX does not contain error-checking functions. Applications can use the Sequenced Packet Exchange (SPX) protocol, which provides virtual session and error-checking functions.

NetWare servers use the Service Advertising Protocol (SAP) and the Routing Information Protocol (RIP) for service announcements and server-to-server routing updates. NetWare also supports the Open Datalink Interface (ODI), which enables multiple protocols to function over any network card that supports ODI.

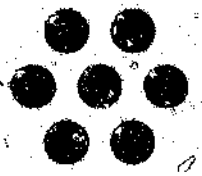
Novell provides NLMs for supporting Macintosh clients over AppleTalk, Unix workstations over NFS and TCP/IP, and OSI systems using FTAM. NetWare v4.0 provides an integrated environment for all these clients, enabling them to share files and print services.

NetWare Directory Services (NDS): NetWare v4.0 can treat multi-server networks as a single entity. This enables users to log onto the network once rather than onto individual servers and lets supervisors manage one logical network rather than separate file servers.

WAN Support: Novell has improved WAN support in NetWare v4.0. The packet burst feature of NetWare enables server and clients to exchange several data packets before sending verification. Also implemented in this release is the Large Internet Protocol (LIP), which enables servers and clients communicating over a WAN to negotiate the packet size rather than use the default size of 512 bytes. Both features improve NetWare performance over a WAN. However, IPX still is not as efficient in a WAN environment as some of the other protocols, such as DECnet and TCP/IP. Many customers are asking Novell to incorporate file and print services using TCP/IP rather than tunneling IPX packets through TCP/IP.

VINES v5.5 One of NetWare's major competitors is Banyan's virtual Network Solution (Vines). Vines is a NOS that runs as a dedicated file server on an x86 platform. The Symmetric Multi-processing (SMP) version of Vines can provide superior performance by taking advantage of SMP machines such as the Compaq Systempro. However, Vines' real strength is its ability to simplify the task of using and managing a network of multiple servers by integrating them into a single logical network.

Vines v5.5 Architecture: The Vines server runs on to of a Unix operating system shipped with the product. Users, however, needn't be familiar with Unix to set up and manage a Vines network. All network services on a Vines server run as processes performing various tasks. The Vines server service executes on every server and handles internal communications, monitors the status of services and synchronises time with other servers. The Vines Vanguard service implements network security. Vanguard authenticates user names and passwords before allowing access to the network.



Protocol Support: Vines uses the LAN Manager protocol for file and print services. Messages are exchanged between clients and servers using the Vines Sequenced Packet Protocol (SPP), which provides virtual connection services. Vines also provides the Network Remote Procedure Call (NetRPC) interface for developing network applications. StreetTalk and other Vines services, such as mail, use NetRPC for communications. Like NetWare, a limitation of Vines is the use of proprietary protocols for file and print services. However, Vines supports such protocols as TCP/IP, Net BIOS and X.25 for other functions. Vines can also tunnel the Vines protocols in TCP/IP packets.

StreetTalk Directory Services: Vines provides basic file and print services similar to those of other NOSs. What separates it from the rest is that integrated into the Vines OS is StreetTalk, Banyan's highly acclaimed distributed global directory service. StreetTalk allows resources to be uniquely identified on the network, making them easier to access and manage. All resources, including file services, users and printers, are defined as objects. Each object has a StreetTalk name associated with it.

WINDOWS NT LAN MANAGER V3.0 Microsoft's LAN Manager server is available on the OS/2 platform, providing file and print services to DOS, OS/2 and Windows clients. The OS/2-based server received limited acceptance because of lower performance and just because it was OS/2. However, LAN Manager will soon be available in the base Windows NT product and in the Windows NT Advanced Server, which will make it a viable option.

Networking Capabilities: Windows NT features built-in peer-to-peer networking, which allows file and printer sharing among work groups. Windows NT uses LAN Manager technology for file and print services and can function as a server and client. Windows NT offers LAN Manager services, using the proprietary NetBEUI protocol for LAN access or TCP/IP for WAN access. Windows NT features built-in support for DOS, OS/2, Windows and Windows NT clients. Applications support is provided using a variety of standards.

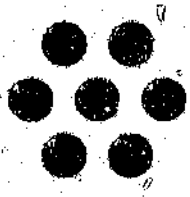
Many vendors are working to integrate their NOSs with Windows NT. Digital has developed Pathworks for Windows NT, which enables a Windows NT system to function as a Pathworks client or server over DECnet. Novell and Banyan are also developing client shells for Windows NT, which will make Windows NT one of the most connected NOSs in the industry.

One feature lacking in LAN Manager is a global directory service such as StreetTalk or NDS.

Windows NT Advanced Server: The basic networking features in Windows NT are sufficient for small work-groups. For larger networks, Microsoft recommends the Windows NT Advanced Server, which offers DOS, Windows, Macintosh and NetWare connectivity, SMP support, advanced security, and server domains for enterprise networks. Domains can be used to group a logical collection of servers. Users can then log onto the domain once and have access to all the authorised resources. Trust relationships allow domains to share user and group lists to enable users to access resources across all domains in the network. Domains also provide network administrators with centralised management capability.

One can access the Windows NT Advanced Server from a remote office via modem, using the Remote Access Service (RAS) module. Macintosh support is expected to be included in the final release, with the Windows NT Services for Macintosh (SFM). The Advanced Server also offers extensive fault-tolerance features, providing enhanced data reliability using the NT File System (NTFS), disk mirroring, disk duplexing and disk striping (RAID5). The user account database is automatically replicated, thereby eliminating any single point of failure.

PATHWORKS While Microsoft and Novell battle it out with NetWare and LAN Manager, Digital has chosen a different philosophy for desktop dominance. Digital's strategy is that



whether users choose NetWare or LAN Manager, Digital will provide the software tools necessary to integrate and manage their networks on one of the fastest hardware platforms in the industry.

This philosophy is evident in the availability of Windows NT on Alpha AXP (a hardware platform for whose acronym the author could not find a translation). Digital also has announced that native NetWare v4.0 will be available on the AXP platform. Further, the upcoming Pathworks product line for OpenVMS will offer one of the best choices for integrating and managing the various network technologies. Future Pathworks releases will enable a single OpenVMS system to support NetWare, LAN Manager and Macintosh clients. And Pathworks servers are also available on the ULTR, IX, SCO Unix (various flavours of Unix) and OS/2 platforms.

Client Support: Digital offers software to enable DOS, Windows and OS/2 systems to function as LAN Manager-based clients and access the services of the Pathworks servers. In addition to standard file and print services, the software includes applications for terminal emulation, mail and the X Windows System. For file and print services, no additional software is required on the Macintosh clients, since they have built-in AppleShare. The Pathworks for Macintosh client software includes software that provides Macintosh clients with DECnet, terminal emulation, mail and X support.

OpenVMS Server Architecture: Unlike NetWare and Vines, which run in a dedicated server environment, Pathworks servers run as OpenVMS detached processes. An OpenVMS system running Pathworks can simultaneously support networked clients and interactive OpenVMS users. This design provides a rich environment in which PC, Macintosh and OpenVMS users can share data, mail, printers and other resources.

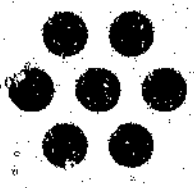
Servers and Protocols: The Pathworks for OpenVMS LAN Manager server offers file and print services over DECnet and TCP/IP to DOS, Windows, Windows NT and OS/2 clients. The TCP/IP option allows the server to support not only Pathworks clients but also any LAN Manager client that can use TCP/IP. Both the DECnet and TCP/IP transports provide efficient LAN and WAN capability.

The Pathworks for Macintosh server is compatible with the AppleTalk Filing Protocol (AFP) v2.0 and supports Macintosh clients over AppleTalk. Included with the product is an AppleTalk-to-DECnet gateway that enables AppleTalk-only Macintosh systems to communicate with DECnet systems.

Future Enhancements: One of the most impressive Pathworks v5.0 components is unofficially called FrameWorks. This multi-vendor integrated management console system enables one to simultaneously manage all the LAN Manager and NetWare servers on the network from any Pathworks PC. FrameWorks uses a slick Windows interface to perform all the management tasks, such as adding users, creating services, setting up printing and managing security for both network environments. Future enhancements will support management of OpenVMS and Windows NT servers. Hooks are provided to allow developers to add management interfaces. A key design focus of FrameWorks is to keep the terminology intact when one is managing a particular environment. For example, when one adds users, the same screen will show the appropriate fields, depending on the server--NetWare or LAN Manager--being managed.

The Pathworks v5.0 client software is also undergoing a face-lift. The software will be available as separate products for the DOS and Windows platforms. The DOS platform will support a command-line and a character-based Windows interface. The Windows product offers all utilities and applications with a Windows interface.

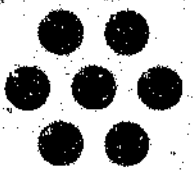
Pathworks v5.0 clients will be able to use NDIS or ODI drivers for communicating with the network cards. The clients will also be able to use DECnet, TCP/IP, NetBEUI, IPX, LAT and LAST simultaneously, and load and unload the protocols dynamically, all without having to reboot. Digital is rewriting the TCP/IP stack, which is expected to be smaller



and faster than in the previous versions. The combination will help relieve the RAM cram in DOS workstations as mentioned earlier.

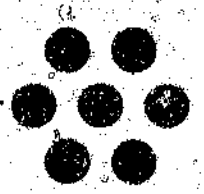
CONCLUSION Outside of PC based applications, the LAN operating system becomes a choice of three.[42] Novell Netware is the current PC LAN leader, but the Netware server operating system only runs on PC hardware and since it is not a general OS relies on Netware Loadable Modules (NLMs) for servers. This is obsolete technology. Microsoft Lan Manager (MS/LM) uses OS/2 for the server, which is general purpose and offers more scope, particularly for transaction processing with stored procedures. Further MS/LM includes multiple protocol options, including TCP/IP, which should be the primary choice since it is supported on all systems. Unix can also be used in MS/LM networks with NCR's LM/X version 2 server product. Novell may yet produce an acceptable Unix server product.

The third choice is Sun ONC, effectively taking PCs to the established engineering LAN world. Here there is only one protocol, TCP/IP, and it is supported by all systems from mainframe through mini to PC. The NFS file server is not as efficient as Netware (nothing is when it comes to pure file serving) but the range of choice of server operating systems makes it way ahead for client/server systems.

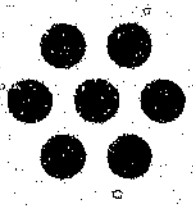


TABULAR COMPARISON

Maximum number of simultaneous users	1000	100	#
Maximum number of server volumes	500	64	12
Maximum number of currently open files	8E3	1E5	2E5
Drivers supported:			
NDIS	☑	☑	☑
ODI	☒	☑	☒
Distributed database	☑	☒	☑
Global naming abilities	☑	☒	☑
Physical names mapped to logical names	☑	☒	☑
X.500 compliant naming conventions	☒	☒	☑
Directory of resources	☑	☑	☑
Synchronised changes to directory	☑	☒	☑
Directory of users available across WAN	☑	☒	☑
Passwords can be assigned to resources	☑	☒	☑
Blocks simultaneous log-ons	☒	☑	☑
Can create a system wide log-on script	☑	☒	☑
Allows hex fixes	☑	☑	☒
Automatically updates workstation shell at log-on	☒	☒	☑
Supports TCP/IP encapsulation	☒	☒	☑
TCP/IP native	☑	☑	☑
Supports OSI Encapsulation	☒	☒	☒
OSI native	☒	☒	☑
Supports SNA gateway	☒	☑	☑
Administrator can perform remote boot	☑	☑	☑
Maximum number of adaptors per server	4	16	4
HDLC/synchronous bridge	☒	☑	☑
T1 bridge/router	☒	☒	☒
X.25 point-to-point bridge/router	☒	☒	☒
X.25 multiport packet-switched router	☒	☒	☒



Automatic clearing	☑	☑	☑
Defered printing	☑	☑	☑
Administrator can view queue	☑	☑	☑
User can view queue	☑	☑	☑
Administrator can reorder queue	☑	☑	☑
User can reorder queue	☒	☒	☒
Administrator can delete job	☑	☑	☑
User can delete job	☒	☒	☒
Multiple queues per printer	☑	☑	☑
Multiple printers per queue	☑	☑	☑
Print preprocessor	☑	☒	☑
Split seeks	☒	☑	☑
Read verifications	☑	☑	☑
Hard disk mirroring	☑	☑	☑
Hard disk duplexing	☑	☑	☒
Split seeks on mirrored drives	☒	☑	☑
Sector recovery on mirrored drives	☑	☑	☑
Server duplexing	☒	☒	☒
OSI		☒	☒
FTAM	☒	☒	☒
Unix	☒	☑	☑
NFS	☒	☑	☒
OS/2/HPFS	☑	☑	☑
Salvages deleted files	☒	☑	☒
Purges deleted files	☑	☑	☑



NETWORK INSTALLATION

In conclusion the following strategy should be adopted today for totally new installations or for pilots readying to upgrade current LANs.

STAGE 1 Install a Unix server with TCP/IP and NFS. This can be a 486 PC with SCO or Interactive Unix, an RS/6000 with AIX, a VAX with ULTRIX, a Sun SparcServer, etc. Its a flexible choice; all support the NFS, as standard. Just as skills have been acquired to install and setup NetWare, etc. then new skills are needed for TCP/IP, etc. Install, in a PC DOS workstation a TCP/IP plus NFS requester package e.g. Sun PC/NFS, ftp Software, Woolongong, etc. Install standard PC packages e.g. Lotus 1-2-3, Word Perfect (DOS versions) on the Unix server and test them. All standard single-user DOS programs will work. However many PC programs are extensions to DOS and thus won't work. The current developments in PC TCP/IP products are to add Netbios compatibility (similar to the Netbios extensions in Netware); this will be essential cover further DOS programs. Any 'DOS' programs, no mind how popular, that won't run must be banned.

STAGE 2 Choose a database supplier and install the server on the Unix machine. Then choose a development tool to write programs in the PC to work with the SQL requester. Note that all SQL tools don't work with all SQL databases; make sure favourite 4GL applications work with the chosen database.

STAGE 3 Implement one or more Unix programs on the server which can be run using the telnet feature in the PC or a multi-port comms card and ASCII VDUs can be added.

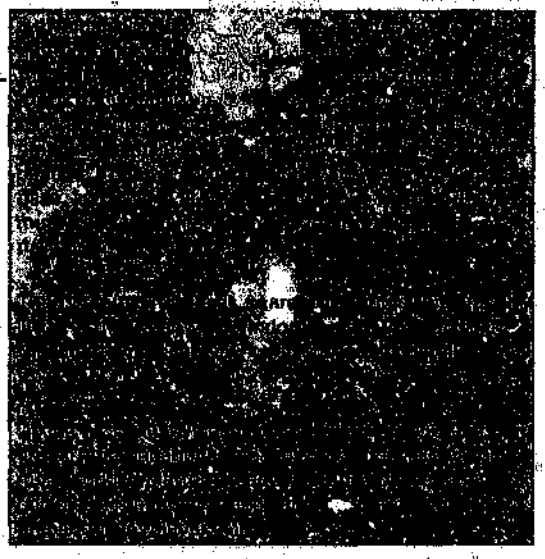
STAGE 4 Implement a gateway to any host computer or wide area network. If at the same time as this development TCP/IP has been added to the host computer then a gateway isn't needed.

At any stage look for packages as well as developing programs and whenever they are available introduce OS/2. 2, Windows-NT, etc. workstations. At any stage existing Unix workstations can be simply connected to the same network. The choice of Ethernet or Token-ring is relatively unimportant although historically there is more support for Ethernet.

Networking is never simple and TCP/IP is no easier than Netware. But the vast increase in facilities enabled e.g. RISC hardware and the interoperability without gateways is well worth the effort.

OPERATING SYSTEMS (OS)

BACKGROUND



An operating system provides a basic set of instructions and functions that make the process easier for the application programmer and the user. The operating system also serves as an interface to the underlying hardware, providing a 'level playing field' for everyone who wants to use the computer system.[43]

Most operating systems (OSs) are divided into two basic layers, the 'kernel,' which interacts directly with, and is specific to, the underlying hardware, and the 'services,' the functions that the operating system performs, which may or may not be hardware-specific.

Strictly speaking, computers do not need operating systems to work. If an application is coded directly for the underlying hardware and then loaded into memory in some fashion, the computer can be directed to 'jump' to the first memory address of the application and execute from there. Obviously, this is not a terribly efficient way of running software.

There are several models around which an operating system might be based.[44]

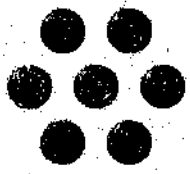
MONOLITHIC MODEL The monolithic operating system model is a structure wherein all the functions of the operating system are allowed to communicate directly with one another, with no particular hierarchy or sequence. DOS uses this model, and it is a part of what makes DOS so hard to modify significantly.

Because the individual elements are interrelated, modifying one, like the file structure, means modification of others, and maintaining compatibility with earlier versions is that much more difficult.

LAYERED MODEL In this structure, each function may only communicate with the function directly beneath it in the hierarchy. With regard to adaptability, this model has an advantage over the monolithic model in that each layer communicates only with a single other layer, which essentially isolates the individual layers. Modification of any layer, then, can be accomplished without changing the entire structure of the operating system.

CLIENT-SERVER MODEL The most modern approach to the design of an operating system is the client-server model. In this model, each non-kernel OS function is treated as a server to the application 'behind' it. These servers communicate with the client and the kernel independently. The client-server model also combines the best features of the other models, in that each function is independent, for ease of modification, yet each is directly linked to the kernel for speed.

The debate that now arises is what OS to choose for each workstation.



CRITERIA

Unlike an application, an operating system is not chosen for the features it offers. Operating systems are chosen, or designed, for factors like portability, adaptability, robustness, speed, security, and their development environment. All of these elements are in one way or another affected by the OS design model, which changes from system to system.

The criteria operating system designers deem most important determines, to a large extent, how the OS is coded.[45] Assembly language is the fastest, but the least portable. Portability is enabled by using a high-level language, like C. Adaptability is the ability of an operating system to accommodate new developments in hardware, and is enhanced by the coding of the OS in 'modules' that are relatively independent of one another.

Robustness is also affected by modularity, and is the ability of the operating system to survive hardware and application failures. Security and development environments are also a matter of design, though not so related to the actual structure of the OS.

COMPATIBILITY Cooperative work requires either that the same applications be available on all the systems or that the different applications on those systems read a common document format.

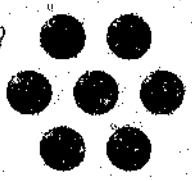
Even true client/server applications face similar problems. If a Windows NT-based database server is to serve Unix and OS/2 clients, for example, front ends for that server must be available on both Unix and OS/2. Without such front ends, those clients would be cut off from the server.

The OS that the server runs on can be quite independent of the operating systems each of the workstations runs on--especially when non-dumb desktops are used. In fact, it is quite desirable to run on different OSs as this is often a function of the computer chip set, and hence its power, and one rarely requires the same power on the desktops as on the server.

Protocols, as discussed earlier, can be made available from any OS, and are the factors enabling disparate OSs to communicate. Thus, the server OS receives calls from client stations on other OSs through a common protocol, and returns information in a similar manner.

GRAPHICAL USER INTERFACE (GUI) A lot of the advantages associated with Windows (DOS, Unix, Macintosh or whatever) based applications, are those associated with Windows itself; namely a common look and feel across applications, quick access to other applications, Dynamic Data Exchange (DDE), Object Linking and Embedding (OLE), etc. Many companies are now standardising on Windows, as this appears to be the direction that the ship is headed. Apart from these general advantages, Windows databases offer two other distinct advantages over their DOS counterparts. The first is their ease of use--they make excellent use of GUIs and various tools and utilities to provide a highly visible and easy method of developing sophisticated applications. The second advantage is their ability to store graphic images directly in tables. Moving to a Windows database product is not without disadvantages. These products are generally slower than their DOS counterparts. Also the fact that staff may have to be retrained to use the new application on a Windows platform also presents its own special breed of problems.

These pros and cons are all good and well, but what about those users with existing applications developed under DOS based databases who choose to migrate? Porting existing data to Windows presents no problem as most databases support a data import command (or in fact may be able to access the existing data directly). The problem lies in the underlying application. Here users will have to face a hard fact. If one wants to move to Windows, existing DOS applications will generally have to be re written. This may not be as daunting as it first appears. Developing applications on a Windows based database



can prove a surprisingly easy task. If the existing application is essentially written on a data entry/query basis, then the move should not prove too troublesome, and in fact a number of applications can be ported without any programming what-so-ever. Also a number of Xbase applications can be easily ported to FoxPro which has powerful cross platform application facilities. There is no simple solution to the problem that is essentially a trade off between the advantages of Windows and the difficulties involved in porting the application. For simpler applications real benefits can be derived in moving to Windows. Highly complex applications may have to be completely re-engineered and it may be best to let working applications lie.

The development of DAS (Data Acquisition Systems) and MIS (Management Information Systems) under Windows is discussed in more detail under those sections.

CLIENT OS REVIEWS

There are only five main operating system worth considering for client applications, Apple, DOS (and Widows), Windows NT, OS/2, and Unix.[48] These options are discussed below.[47]

APPLE In the micro world, Apple was the first to introduce a computer operating system that was really for the non-technical user--an OS that incorporated a graphical screen which enabled users to perform many system functions without reverting to keyboard commands. Significantly, Apple also required application developers to maintain the conventions, with regard to function, that Apple had established. The result was computer hardware that was easy to use, consistent in appearance and function throughout its suite of applications, and relatively bug-free and seamless in operation.

Apple has maintained this high standard throughout the development of its computer line, to the point that even the most powerful Macintosh available today has an ease of use that is generally unparalleled in the technical marketplace.

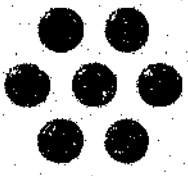
Every operating system manages system memory, the sum total of which is called the heap. This is a fundamental reason for the existence of an operating system. How that memory is made available to applications, though, varies. Most systems use pointers to assign a given block to an application. In the Macintosh, when an application makes a system call for a memory block, the OS allocates the required amount (if it's available) and assigns a handle to the block.

Essentially, a handle is a pointer to a pointer. This allows the Mac OS to construct blocks of sufficient size where no contiguous block existed. Separate memory pointers are all linked to a single handle, and the application never knows the difference. In practice, this allows more efficient use of available memory, because individual pointers can be allocated and deallocated from a handle as needs dictate.

Resources are another concept that originated with the Macintosh. What they do is allow software to manipulate many different types of code and data structures in a consistent fashion. Resources are data identified by types--types the operating system recognizes and uses in template fashion.

Although resources can be created by the system programmer, care must be taken not to duplicate any existing standard resource types. Unfortunately, resources may make for efficiency in operating system handling of data structures, but they don't translate well to other types of operating systems' classifications. The potential for rough translation makes portability to DOS, Windows, or Unix extremely difficult.

Since the Macintosh is a graphically oriented system, all screen and printer output (with the exception of direct PostScript code) is handled by QuickDraw routines--the official Macintosh method for handling graphics output. Under System 7 (Apple's latest offering),



all QuickDraw functions are reduced to a set of 13 routines (which vary by device) known as the QuickDraw bottleneck procedures. This feature allows an application programmer to write code that is essentially non-cognisant of the display or printer type, greatly simplifying their task. At the same time, peripheral manufacturers only have to make their devices recognize the QuickDraw routines.

Finally, events are trigger functions in the Macintosh environment. An event is caused by a mouse click, a key being depressed, or a disk being inserted in a drive. Events tend to change the entire feeling of the Macintosh interface, by transferring apparent control to the user. Applications are event-based, so that the user guides the application in its processes.

The main obstacle to System 7's dominance of the desktop is its lack of portability to other platform types. It isn't so much that System 7 isn't portable (anything is, with enough effort) but that it seems to be specific not only to the Motorola 68000 family of processors, but to the Macintosh as well, and Apple shows no signs of changing that stance. In other words, Apple isn't trying to be all things to all people, but to be everything that some people want.

DOS AND WINDOWS

When IBM introduced the PC in 1981, with Microsoft DOS as the operating system, application consistency was not the standard. Because DOS was a more 'open' system than Apple's proprietary OS, applications were developed in a free-for-all fashion that resulted in a tremendous number of programs to choose from, each with their own standards of operation and methods of handling memory, display, and file functions.

While this was ultimately of benefit to IBM and its competitors, and to those who chose the PC or its 'compatibles' as their system of choice, it also had the result of producing very little consistency. There was no guarantee of how an application might 'behave' with respect to the underlying operating system.

Originally, Windows was an attempt to bring some of the features that the Macintosh had made so popular into the PC environment. But because of the way software had been developed for the Intel/IBM and DOS architecture, there were numerous difficulties, not the least of which was an inability of the underlying hardware to provide the kind of memory addressing and processing power needed for a multitasking graphical environment. However, the 386 and 486 has changed that, and now DOS and Windows can be seriously considered as a suitable environment for future migration to Windows NT.

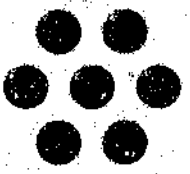
WINDOWS NT

Today's micros have sufficient processor power to handle the demands of a complex and powerful operating system, and Microsoft is rushing to fill the gap. Windows NT (New Technology) is Microsoft's first stab at providing a user interface as a part of the operating system, not as an add-on layer, and as such it brings several new features to the environment.

Windows NT will bring together support for several different processors and hardware platforms, and provide an operating system environment for application programs that nominally run under DOS, Windows, OS/2, and certain Unix implementations. In addition to this, it offers security, built-in network and distributed processing support, hardware and software failure protection, and symmetric multiprocessor capabilities.

The client-server architecture of NT is divided into an NT Executive (the kernel) and several subsystem modules (the 'user' modes) which handle the various compatibility components of the operating system. The executive is further divided into system services (memory manager, I/O manager and file systems, process manager, etc.), the kernel, and the Hardware Abstraction Layer (HAL), which interfaces with the underlying hardware.

Because NT is written in a high-level language (ANSI C and some C++), with only some hardware-specific modules written in assembly for speed and efficiency, it is portable to



practically any type of 32-bit processor with a minimum of modification. Microsoft's target processors include Intel's 386 and 486 processors, along with the Pentium, the MIPS R4000 RISC CPU, and DEC's Alpha.

Another important element of Windows NT is the POSIX subsystem. POSIX (the Portable Operating System Interface extensions) is a group of standards that define how an operating system interacts with its hardware and applications. Key attractions of standard, open systems, from either the developer or system purchaser side, include vendor independence, protection from technological obsolescence, preservation of software investment, availability of off-the-shelf applications software from independent suppliers, availability of software engineers familiar with standard environments, and connectivity between various dissimilar computers within the user organization. All of these are in line with the design goals of Windows NT.

OS/2 When IBM moved away from DOS and brought out OS/2, it represented a chance for IBM to predominate the PC industry, and to coordinate system activities *vis--vis* the application software. Unfortunately, by that time, over 50 million IBM PCs and compatibles were merrily motoring along under DOS, and OS/2 didn't offer enough new functionality for most people to switch over, so they didn't.

But IBM has been on a bit of a comeback ever since it rolled out OS/2 2.0, and now, 2.1. The very latest release of this operating system offers many of the functions we have defined as important to the next generation. Version 2.1 is the 32-bit version of the original OS/2 design goals, and in this capacity OS/2 has achieved its intent.

OS/2 is composed of a kernel, device drivers (similar to DOS), and dynamic link libraries. Version 2.1 provides APIs (Application Program Interfaces) for DOS applications, OS/2 1.x programs, and 16-bit Windows software. There is also an OS/2 2.1 32-bit API, and many programs have already been developed for this environment.

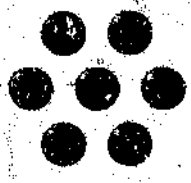
Most of the APIs in OS/2 are located in the kernel, but some derive support from the dynamic link libraries, which are shared libraries of module-based functions that allow for extensibility of the operating system.

In creating the compatibility standards between 32-bit and 16-bit applications, the OS/2 design team developed two unique processes, LDT (Local Descriptor Table) Tiling, and 'Thunks'. These procedures allow an application based on 16-bit memory addresses to run in a 32-bit environment, though they handle different portions of the process. Sixteen-bit memory addressing in DOS extenders limits applications to 64K of address space in each block, or page, of extended memory, which means that an application running in either real or virtual memory needs to swap as each 64K boundary is crossed.

OS/2 contains both 32-bit and 16-bit code, enabling compatibility of coexisting flat and segmented memory models. LDT Tiling enables an application to have up to 512Mb of virtual address space, and physical swapping only occurs as the application passes whatever limits exist in real RAM. Every process, whether 16-bit or 32-bit, has a local descriptor table to map the virtual address space. If an application is using 16-bit code, it maps using the LDT selectors; if the program is using 32-bit addresses, it uses the flat model.

Thunks are the technology that allows the 16- or 32-bit API functions to map one to another. LDT Tiling enables Thunks to make the architecture conversion efficient. As far as the application knows, the underlying service structure is transparent, regardless of the direction of translation.

OS/2 is also an integral part of IBM's SNA (Systems Application Architecture), IBM's plan for the integration and migration of applications across their entire line of hardware systems.



Currently, OS/2 runs on 80386 and 80486 processors, but is slated to be ported to other architectures. One problem that exists in this area, though, is that the RISC engines of most workstation-class systems are unable to efficiently emulate 8088 compatibility, the basis for every DOS application developed to date. Like System 7 for the Macintosh, then, this makes OS/2 a less-than-ideal choice for any corporate environment with a wide variety of platform types, except as one element in a network.

UNIX All the operating systems discussed thus far have originated with personal computer systems. On the other end of the hardware spectrum, that is, in the non-micro world, some variation of Unix had been the 'open-standard' operating system for quite some time. AT&T's Unix was developed as a time-sharing, multitasking, multi-user OS with the idea of giving as many people as possible access to expensive processing time and system resources.

AT&T licensed Unix to a wide variety of vendors and universities, and because the hardware platforms on which this system ran were expensive, it was viable to customize the operating system code to the underlying hardware. The result of this individualism is that today there are more flavours of Unix than there are Aylesbury Yoghurts.

Yet Unix, in one variation or another, does hold the promise of providing all the features in the wish list. The problem is, which version. Today, two major variants of the original Unix structure seem to have dominance in the microcomputing world (they receive the most reviews and press coverage): SVR4 and OSF/1.

System V Release 4, or SVR4, is AT&T's version of the complete operating system. SVR4 brings together most of the functions and systems of AT&T, Berkeley BSD, and Sun OS releases into a single package standardized under the SVID (System V Interface Definition) suite of benchmarks. The development of this operating system is now the responsibility of a non-profit organization, Unix Software Laboratories, or USL, for commercialization and standards development. This 'standard' has been widely touted as Unix's saviour in that it will help eliminate the weakness of this platform arising from its non-standard array of standards.

The second major flavor of Unix is OSF/1. OSF, the Open Software Foundation, is a consortium of vendors led by IBM, DEC, and HP. OSF was founded in protest to AT&T's agreements with Sun Microsystems concerning the direction and development of Unix. OSF members banded together to push OSF/1, a version of Unix based on the Mach kernel developed at Carnegie-Mellon University.

The driving forces behind SVR4--AT&T, Microsoft, and Sun Micro--have been largely responsible for the expansion in popularity of Unix from the restricted worlds of high-end technology and university environments to commercial high-volume use in the business world. On the other hand, IBM, HP, and DEC are the traditional high-volume vendors of commercial business computing solutions. Market share, then, will be determined by which combination of operating system and hardware platform offers the most flexible, functional, and friendly environment.

Compatibility is a serious issue, and is probably the primary reason that the Unix market has stayed so fragmented for so many years. The SVID and SVVS (System V Verification Suites) are one means of determining compatibility, though they are limited to compatibility within System V standards and guidelines. Another set of standards that are experiencing increased impact are the POSIX standards (discussed earlier), though these are not universally accepted, or even, in many cases, firmly established.

POSIX compliance does not specify a certain type of Unix. It specifies an API, so that any operating system which complies with a standardised test suite should be able to run any application with only a recompile. If applications are produced to an ABI (Application Binary Interface), as they are in the DOS world, then shrink-wrapped software for disparate system types becomes possible. Unfortunately, there is no way for a standard



ABI to be established for Unix, or any other operating system, until consumers agree on the underlying hardware that they will use.

One benefit of Unix, whatever flavour one chooses, is that it can be (and has been) ported to just about any platform. All of the major vendors have some version of Unix (or a workalike) running on their hardware platform. And one of the persistent complaints about Unix--its difficult command-line interface--has been largely eliminated by graphical front ends that can make Unix as friendly as the Macintosh.

Unix uses a combination of a layered and monolithic approach to operating system management, and considers the entire computer, from applications to hardware and network connections, part of the Unix environment. The highest layer (of five) is the Application Environment layer, where the graphical user interface was made popular by Sun.

The next layer down, Layer Four, is the System Software layer, which includes the shell, the file system, editors and utilities, E-mail and communications functions, and software development tools and libraries. Layer Three is the kernel, which supports the system call interface, kernel functions, network and other extensions, and the hardware interface. Layers Two and One are both hardware. Layer Two is the individual computer system that is running Unix, and Layer One is the network connections that enable it to communicate and process with other computer systems.

The suitability of Unix as a corporate-wide operating system comes down to a choice, as with other operating systems, of the types of applications that are available. This used to be a stumbling block for Unix vendors, because there were so many different versions of the operating system that creating an application that would work consistently across multiple releases was a software vendor's nightmare. However, the consolidation of features and functions that has allowed programmers to create code for an API has had a ballooning effect on the proliferation of cross-platform applications.

CONCLUSION In short, there is very little one need concern oneself with as far as interoperability goes, Oracle's new Cooperative Development Environment (CDE) is a platform in which an application is designed, and then generated for either the DOS Windows, Windows Motif, or the Apple Macintosh environment--an augmentation on 4GLs (4th Generation Languages). The decision as to what OS to use rests purely with the power required (high enough to cater for future applications, yet low enough to be economically viable), the necessity for a Graphical User Interface (GUI), and the desired front-end applications which may well be platform specific.[48]

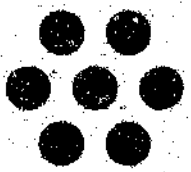
SERVER OS

The server is an altogether different story to workstation operating systems. The PCs have coped with simple file serving, but they won't cope with running a RDBMS except for small networks. Thus the options for the server operating system must be more open. Netware-386 is the first possibility with the RDBMS implemented as a Netware Loadable Module (NLM). This, however, is totally inadequate for two reasons:

- transaction processing applications need 'stored procedures' on the DBMS server, which are loadable modules created with the application;
- a much faster, wider range of hardware must be supported.

The industry will not accept Netware as a standard OS in competition with Unix, etc.

Thus the choice of server operating systems is OS/2 (not acceptable because of the PC only limitation), Unix or an existing mini (VAX/VMS, AS/400) or mainframe. Unix is the clear winner on three grounds:



- Virtually all hardware platforms are available. A RISC machine is much better value for money than a PC, except for small systems;
- All the RDBMS vendors have a server version implemented and highly tested for the common Unix systems;
- Unlike Netware or OS/2, Unix can run local applications accessed by dumb ASCII terminals (or telnet in workstations).

The latter point is the clinching factor since now the application system designer can mix dumb terminal applications with simple DOS workstations or fancy workstations with OS/2.2, Windows-NT, Unix or the Mac OS. This breaks the 'terminals versus PC' argument of older mini versus DOS LANs.

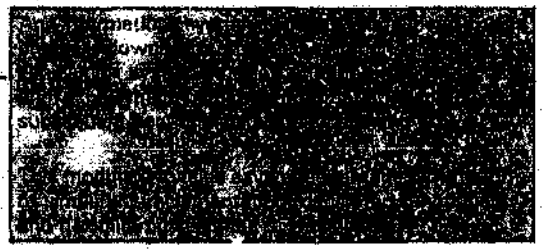
Unix can be introduced into PC networks by exploiting the H-P developed LM/X server within Microsoft Lan Manager. This, however, would be a foolish move. Why take Unix to the immature PC networking, why not take the PCs to the robust and established TCP/IP networks? By using the Sun Open Network Computing (ONC) architecture, with the NFS server replacing the MS-Net SMB or Netware servers, the LAN is now fully compatible with all other systems. ONC is supported by IBM VMS, DEC VMS, etc. as well as Unix.

It now remains to provide the TCP/IP protocol stack, NFS requester and SQL requesters for the workstations. These must be used instead of PC network protocols, not alongside. There are a number of suppliers of such workstation soft-ware for PC DOS, OS/2, and Mac. Windows-3 can also be used but an intelligent network card is needed to achieve reliability as a hardware level to substitute the precarious low level calls in such hybrid systems.

With the TCP/IP system the PC will also get telnet, which will enable access to the Unix programs on the 'server', and ftp transfer files for private PC applications.

MODULARITY

INFORMATION SYSTEM (IS) BUILDING BLOCKS



The IS system can be broken down into three modules, each residing on a common network backbone.

- The Data Acquisition System (DAS) collects data, screens in a posts it to the central data repository;
- The data repository, or Information Storage (IS) module, structures and organises the data, accepting and answering requests from the information presentation module;
- The Management Information System (MIS) module draws the data from the IS, and presents it in a format acceptable to the user--a graph, report, or summed figure.



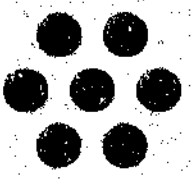
The Information System Modules

The simplicity of the IT architecture is illustrated through these three modules.

INFORMATION FLOW The DAS may gather information from the plant through a computerised system linked directly to the plant control systems, else it may accept information through a terminal punched by an operator who copies data from manual transaction reporting systems, else from the plant control system reports. The MIS feeds information into the company-wide IT system for group results. Feedback to plant control and operations is generated by the reporting system and passed through management personnel.

TIME FRAMES Time frames for each area help to identify their position in the IS. Plant control is characterised by milli second to second long time frames for information cycling. The DAS system accepts information with an indication of its occurrence to the minute, the IS stores information referring to hours or days, the MIS presents summaries spanning days to months, and the IT system is usually only concerned with weekly to annual results.

USERS The lower echelons of the IS are used primarily by the plant operators, whilst the higher modules are designed to cater for plant management through to company management. The environments in which each module reside and operate can vary, except for the DAS and MIS which are usually the same system/package.

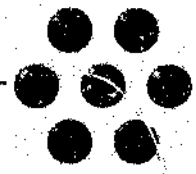


ADVANTAGES

This modularity, which typifies client/server architecture, as discussed later, has several advantages.

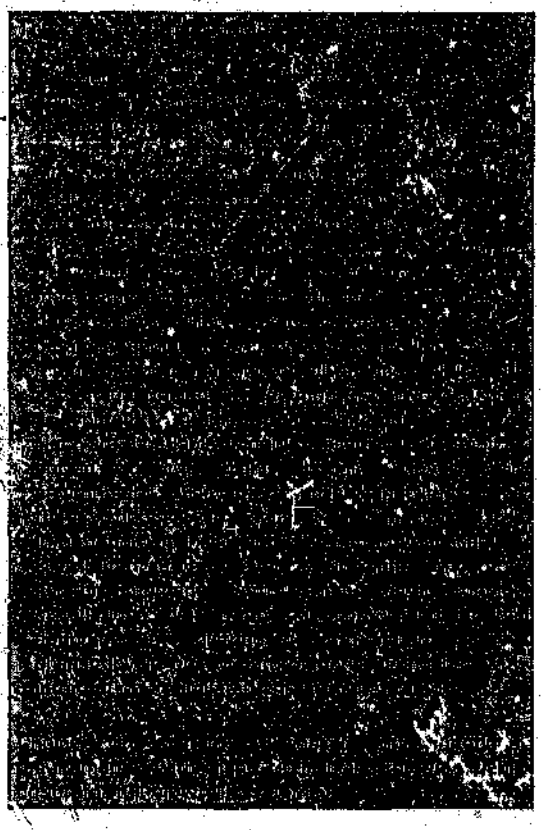
Firstly, there is a high level of support for each specific function found in vendor packages. A complete system may well have limitations in certain areas in which the vendor did not specialize. The issue of compatibility is discussed later. Modularity allows any single area to be upgraded without having to affect any other. Further, several instances of each module can occur, catering for divergent needs and cultural requirements across the concern. Finally, modularity allows for the concurrent testing of any package for any module, and the discarding of antiquated modules with few (if any) repercussions on the system as a whole. With a single dedicated platform, it's replace everything or nothing.

All these issues are dealt with in more detail throughout the following text, which delves into each module separately.



ARCHITECTURES

FILE SERVER DATABASE MANAGEMENT SYSTEMS (DBMSs)



This is the traditional PC architecture, pioneered by Borland's dBase and initially intended as a single user flat file storage environment. Progressive expansion of this package has allowed it to migrate to a full blown multi-user and somewhat relational package, yet still only designed to operate on a file server.

The database management software resides on the workstation (once loaded), with data files residing on the server. Utilising the networking principle of storing files on a file server, where they may be accessed by any workstation, the file server database program attempts to implement locking on the file so that multiple users can access it. The crux, however, is that management is covered by the instance of the DBMS running on the workstation, and there may be several instances attempting to access a single file.

This system reduces the efficiency of locking, offers security to hacking (a data file may be accessed by any other instance unless the file is encrypted, but this key may be surpassable), and also places all the computing power requirements on the desktop. The server merely passes the whole file backwards and forwards, which also results in network loading.

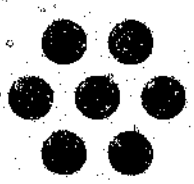
It is important to note that this architecture is an extension of the stand-alone database system which resided entirely on a single PC, and whose data has been transferred to a centrally accessible storage unit. Thus the philosophy is encumbered by the disadvantages of layered modifications being built upon an unsuitable foundation.

File server DBMSs do, however, serve a large user base interested only in stand-alone applications, unconcerned with corporate-wide information access, and not looking for mission critical multi-user access.

CLIENT/SERVER DBMSS

ORIGIN As was covered earlier, client/server computing arose from a background of mainframe computing, and migrated through several enabling technologies, to the PC environment. Its structure in this environment varies somewhat from mainframe computing in that one of the enabling factors was desktop PC power, and this is put to its best possible advantage.[49]

DEFINITION Client/server computing means a computing environment where the presentation, application processing, and data storage and retrieval are (potentially) performed on different computers, possibly in different locations.



The potentiality exists because some of these activities can be combined and performed on the same computer, while still preserving the client-server idea. For instance, in LANs, it is common to let the workstation perform the data entry, application, and display processing, while a server manages the file or database system (including security aspects) and print queues.

Previously, the desktop terminals were 'dumb' and did no work, with applications residing and running on the server along with the database engine, now the facility exists to spread the load a little and pass some work off onto the station clients, thus allowing the server to cope with more tasks. The distinction between the server and clients can become fuzzy, but the philosophies remain sound until one starts to migrate to distributed computing (discussed later).

If the right distributed computing infrastructure is used, the client can run on one machine and the server can run on a different machine that may be geographically distant from where the client program is running. Furthermore, the server can provide services to multiple clients. A classic example is where multiple applications need the services of a database management system. Another client-server candidate has applications use the services of multiple servers, and for more complex applications, the application itself may be divided into multiple server components--each with the ability to run on different machines--true distributed computing.

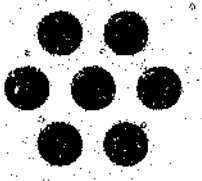
Often distributed or client-server computing is associated with an object-oriented programming style that partitions the software into modules. Each module presents boundaries or well-defined interfaces to the other modules. In the object-oriented approach, each module implements an abstraction, or a model of some aspect of the problem domain. A module's user only needs to understand the semantics of the interfaces and the module's essential characteristics, and not how the module is constructed. In essence, the object-oriented module hides the details of how the module is implemented from the users of the module. Intelligent encapsulation, therefore, attempts to include within a module those design decisions that are likely to change. If change can be kept inside a module without affecting the well-defined interfaces to the module, then application reliability can be reasonably assured.

A relational database management system (RDBMS) is an example of a module that presents a well-defined interface, such as the Structured Query Language (SQL), to its clients, and where the internals of how the RDBMS is implemented are not relevant to the rest of the application.

In object-oriented terminology, and terminology that is used by the Open Systems Foundation (OSF), an instantiation (a running instance in a computer) of the program code and data structures that implements an object-oriented module is sometimes generically called an 'object server'. Note that a client is any module that uses the resources or services of an object server. So it's possible, even desirable, for one object server to be a client to another object server. For example, the RDBMS may be a client to the operating system, which provides basic file services to the RDBMS.

The client/server concept grew out of the competitive world of open systems. "When people say client/server now, one can almost substitute the phrase open systems," says Roger Sippl, a founder of database vendor Informix.
"Open systems gave entrepreneurs the opening to solve particular problems, which benefits the users. Client/server's openness means that one can choose from various vendors' products."[60]

If an enterprise moves to client/server computing, it will change forever the way it accesses, distributes, and uses data. With this approach data is no longer under the tutelage of the MIS (management Information Systems) department; it is readily available to users through the network. Consequently, users don't have to wait for the MIS staff to run their report. They can grab the data themselves and run it through a spreadsheet from



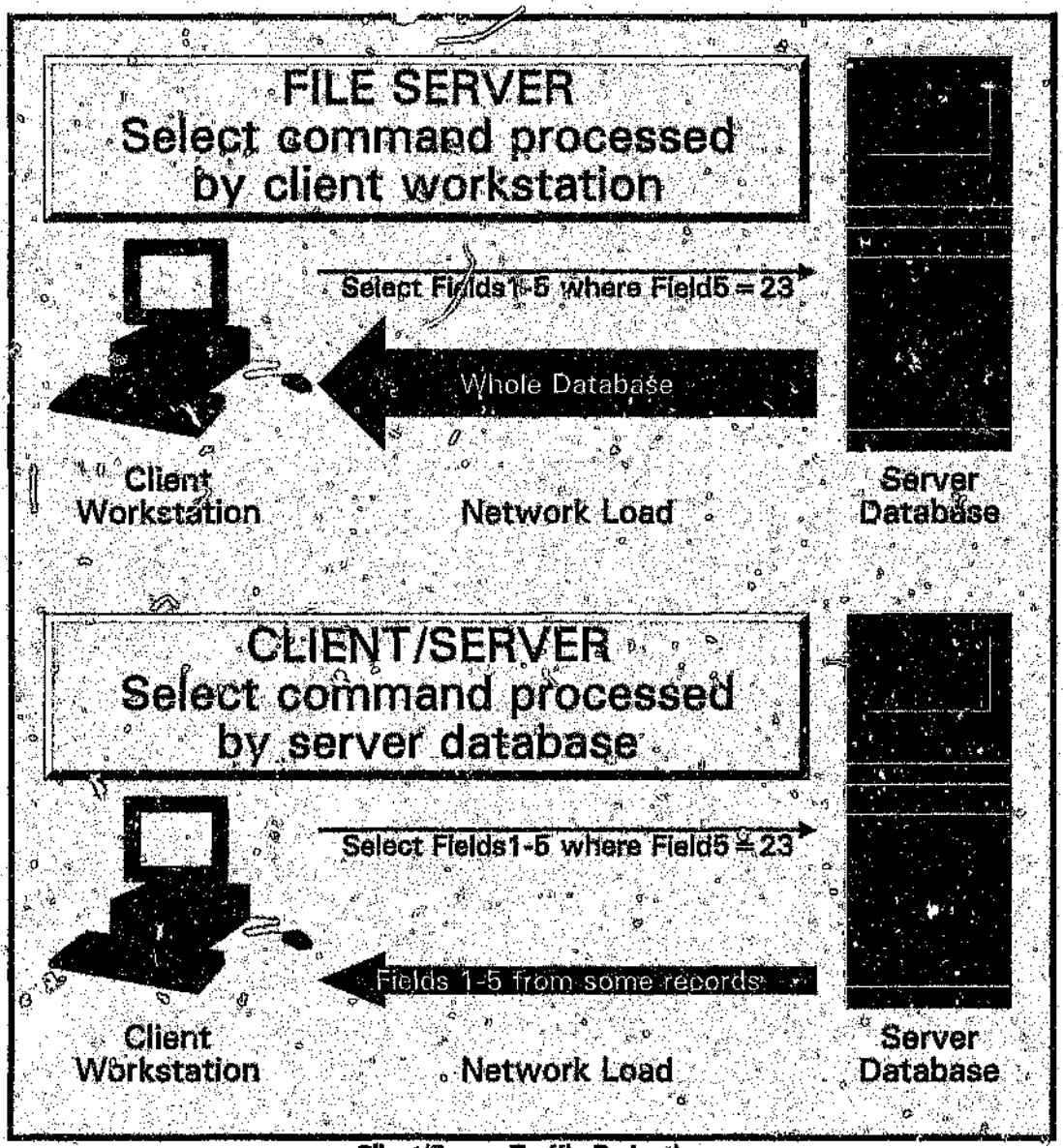
their desktop PC using the brand of computer and the software tools that they're most comfortable with and that get the job done their way.

ADVANTAGES A recent survey of 200 companies conducted by Work Group Technologies, a market-research firm, showed that the three points respondents considered most important in a client/server environment are better access to information, cost savings and overall price/performance benefits.[51]

NETWORK TRAFFIC. From a hardware standpoint, many of the benefits of client/server technology come from the processing power of desktop PCs. When most of the processing power resides on a host mainframe, there is a great deal of communication between the mainframe and users' terminals. In a client/server model, much of the mainframe traffic is unloaded by transferring appropriate tasks to client PCs. Managers originally bought PCs to give themselves some power on the desktop, independent of a mainframe, client/server PC LANs give one the best of both worlds as users can get at the processing power of the server, and if something goes wrong they still have the autonomy of the PC.[52]

The smarter the database, the less the front-end programmer has to worry about. Intelligence on the database server, the so-called 'back end', is what distinguishes the client/server model from the file server model. In the file server model, the network file server makes the database files available to each workstation, but its participation ends there. For example, if one used a file server database to retrieve the records of all people living in Cape Town, the file server would transmit the entire database table across the network and it would be left to the workstation to determine which records satisfied the query. This can burden the network tremendously.

In the client/server model, the server plays a more active role; there is intelligence in the back-end software. Network traffic is reduced because the server processes queries and then sends the client what it needs and nothing more. The server can also control access to sensitive data with a high degree of granularity (discussed later), which provides better security than the file server architecture can. Moreover, SQL databases provide myriad features for protecting the integrity of data.

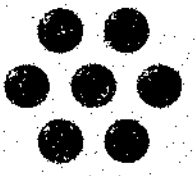


Client/Server Traffic Reduction

MODULARITY What may best define client/server computing is that it lets one pick and choose among platforms. Anyone looking for *the* client or server is missing the point of client/server, which it the ability to easily and rapidly customise relatively low cost platforms for specific applications.[53] Because of the ability to specialise ones platforms, one can get exactly the type of software one needs to satisfy ones req irements.

This modularity allows the system to expand to meet growing needs. Growth can be facilitated by adding capacity to the servers or by adding more servers to the network, one can add more clients, or segment the network if it starts to become a bottleneck. There's no upward limit to the scalability.[54] As new computing platforms emerge, new environments and system components can be evaluated in a modular fashion. For example, when debating future server platforms for Intel architectures, one may look at Windows NT, SunSoft's Solaris, and Next Computer's Next Step. Some people believe in Unix, which will probably remain their choice for large database servers. Some consultants who focus on LANs, however, find difficulties with Unix databases.

CENTRALIZATION Many of the benefits of client/server technology are achieved through centralised network resources, allowing for better management and deployment of those



resources. All users on a LAN can access the software that resides on the server, for example, and LAN managers can update the server software rather than having to update each PC.

Placing facilities such as fax gateways on the server can also prove more efficient than placing a fax machine in each department of a corporation. Users save time by avoiding long waits at the fax machine, and benefit from a log of transmissions and receipts that help them track lost or incomplete faxes. A direct result of shorter application-development cycles is a more streamlined and responsive organisation that can deliver information to customers quickly and effectively.

ROBUST INFORMATION ACCESS Previously, management computing was characterised by 'islands of automation' where each manager had some form of spreadsheet control system that he was familiar with, but that on-one else could understand. This would fall over when the situation changed and the spreadsheet couldn't, or when the manager moved on leaving a new person to create a new system. All this was extremely counter productive.

Integrating database applications with desktop applications increases productivity, a major benefit of incorporating client/server technology.[55] One can do a database query and paste the results into a spreadsheet, and continue to work with it. Being able to take the data one step further has not always been so easy.

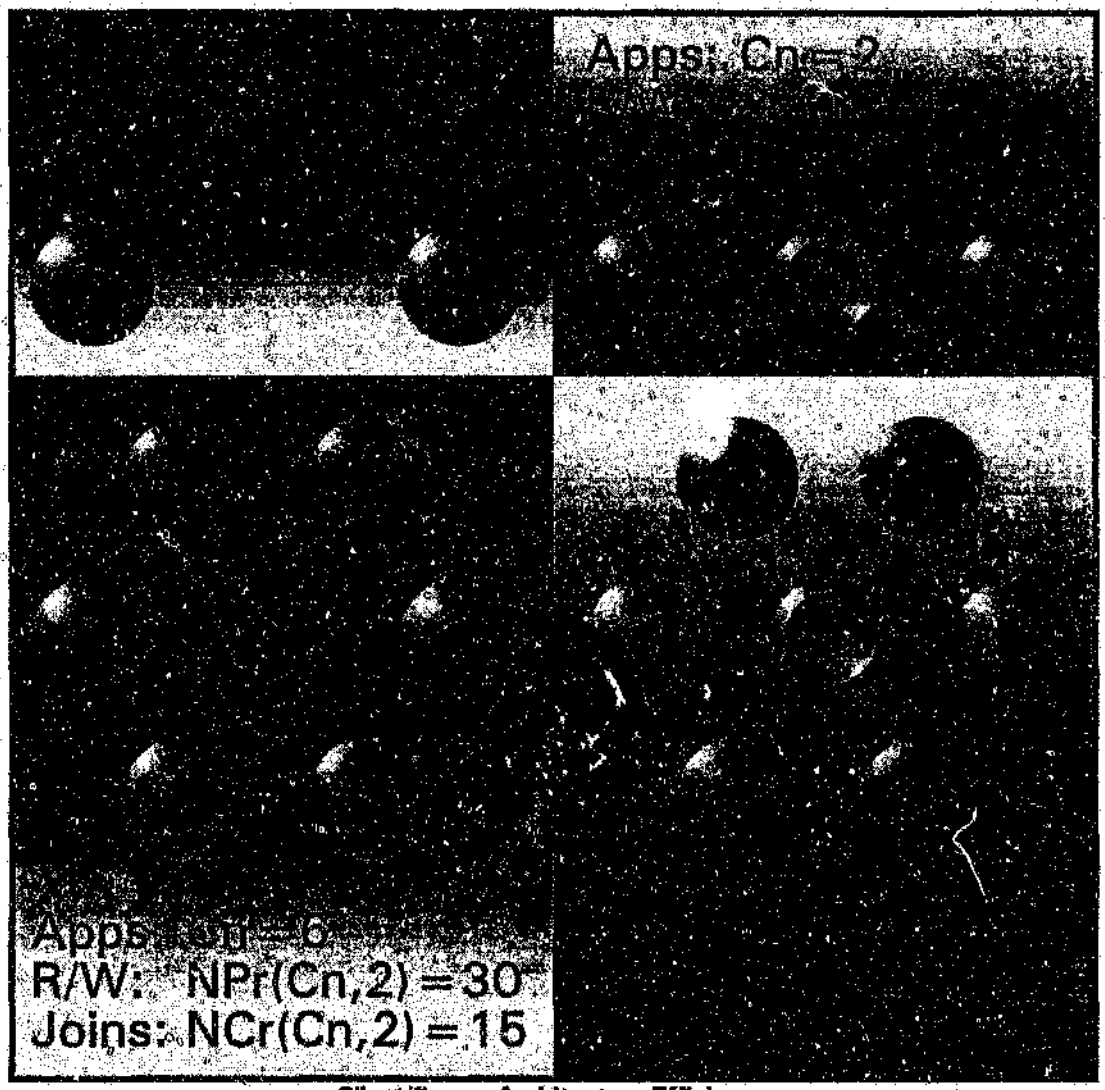
Client/server architecture provides as many access points to data as their are workstations on the network. It also brings to the enterprise more tools to manipulate data. Not only is data more available to users, but it's available through a system that can grow and change with the company as user needs evolve. User generated reports could not previously be generated on the mainframe, as this limited access to computing resources.

Further, because client/server systems are open and modular one can tailor the system for ones own particular applications, picking and choosing the most cost-effective hardware and software components for users tasks.

Forrester Research analyst Weintraut, sees client/server technology becoming more versatile in the future. He envisions a new breed of 'tactical' client/server applications, which he describes as applications that one can dynamically modify to meet rapidly changing business situations.
"Previously, changes would queue up for months or years in the MIS backlog," he says, "precluding any consideration of modifying applications in a pedestrian manner."[56]

CORPORATE-WIDE SIMPLIFIED INTEGRATION Finally, client/server architecture provides true integration to the business systems. Often vendors cite interfacing as integration--without the promised results. Interfacing involves transferring information from one system to another, on a regular basis. This may be every night, or more frequently if possible. The process will usually tie up both systems, may take some time, and will load the transmission media (network/modem) for that period. Information is not available in 'real-time' (up-to-date) in this method. Integration is where both systems access (read and write to) the same data pool, and save for queued instructions to the database server, information is available in real-time. Client/server is inherently integrated in that all systems use a common data repository, thus allowing mission critical and real-time applications to run together, and to share information.

The greatest advantage, is that as a system is added to the company environment, it can be readily integrated into the whole architecture with only as much work as would otherwise have been required to integrate it to a single other system. This concept is illustrated below where the left column shows how in a non-client/server setup, two applications require the same amount of work to integrate as two in a client/server environment on the right. But as one increases the number of clients (say to six), the number of potential links, and hence work, increases dramatically, whilst the client/server yields an increasing integration to work ratio as the system expands.

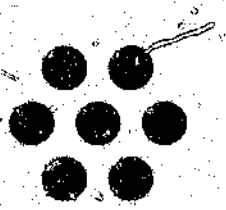


Client/Server Architecture Efficiency

In the diagram, 'Apps' refers to the number of clients ('C') on the system (integratable applications), 'R/W' to the number of read and write gateways that would have to be written, and 'Joins' to the number of links created. Naturally, the industry standardisation on client/server communications means that, hopefully, in a client/server architecture, no gateways would have to be written as the client would come with the facility to talk to and read from and industry standard database.

DISADVANTAGES A dilemma many corporations face when considering a transition to client/server systems is the fate of 'legacy' applications (applications built on proprietary platforms in a 3GL (3rd Generation Language) such as COBOL, and that form a significant part of the company's business), which must be re-engineered to function in a client/server environment. Re-engineering is not a complicated process in a client/server environment, however, and managers, application developers and users alike glean the advantages of the shorter development cycles. In contrast, mainframe application-development lead time can be two years or more. Additionally, re-engineering can be superseded by re-building, often desirable in situations where a system has been in place for so long that it must surely need bringing up to speed with advancing company operations and new modes of competitive operation.

Other problems are illustrated by the following anecdote:



Jim Daley at the House of Representatives considers the IBM ES/9000 he runs to be an excellent machine, and he knows exactly how to keep it running. But when there's a problem in his client/server system, there is a long list of things he has to check. "If you're sitting at a Macintosh, a database query goes from there to the Sybase database on Unix. From there, it could go to an IBM RS/6000, which then runs it up to the mainframe. The mainframe turns around and sends it all back. It's gone through several different products that haven't been mentioned. They all come from several different vendors. Then again, it may be a hardware problem."[57]

This typifies what might be the biggest challenge to successful client/server computing: One has to put together a complex system of hardware and software from multiple vendors. And when snarls develop, one has to determine which piece caused the problem. For many, this can be a rude awakening. One no longer has one dominant vendor holding ones hand, IBM is not there saying, "Don't worry." Just as personal computer users learned to get out their screw drivers and plug in components, so too, must client/server network managers learn how all the pieces fit together rather than relying on installation support experts. This gives rise to another problem in that client/server systems require about one maintenance person per 50 workstations.[58]

But the biggest obstacle in bringing up a client/server system is that one must relearn and retrain. Client/server computing is relatively new, and its development tools are different from those that large enterprises are accustomed to. Many corporate developers have been writing big, batch-oriented COBOL programs or interactive, character-based C programs.

Such concerns may be an indication that client/server computing has left its pioneering phase and is in an early stage of maturity. Bob Epstein (Sybase's CEO) notes that prospective customers now evaluate products with questions about system management, not with a feature list in hand. Thus one can expect vendors to respond to market demands by offering maintenance and development tools, and reducing this problem in the immediate future.[59]

DISTRIBUTED (SOCIAL) SYSTEMS

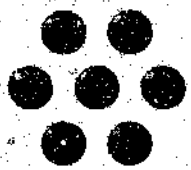
Vendors love to solve problems by declaring them solved. Declarations don't require much research and development, don't take much time to produce and don't cost much to make. Unfortunately, simply saying something is so doesn't make it so. Client/server database systems, for example, are not necessarily distributed database systems. No matter what their vendors might say.

Spotting a client/server database system is pretty easy. A server machine runs database server software. Some machines networked to that server run client software that uses the data-management services of the server database software. End of story.

Now the logic of some vendors goes: Put a few such database servers on a network, possibly a WAN, and the database servers are distributed. The mere presence of two or more geographically distributed database servers does not create a distributed database system. First, a true DDBMS must be able to work with logically related data that is spread across multiple databases on multiple machines. The 'logically related' part of this definition is vital.

If one database server on a network, for example, holds employee information and another database server on that network contains unrelated data on items for sale, one doesn't have a DDBMS. One just has two database servers on the same network.

But, if the database server software lets one work with related data on both servers--say, employee data on one machine and insured dependent data on the other--then this is a



DDBMS. Slightly more technically, if one can run single database transactions that manipulate data on multiple machines, and if the database servers give those transactions the same consistency and integrity support as single-server transactions, this is a DDBMS.

This architecture means, that a company may have data stored at each of its source sites, yet accessed from a central head-office. Each site, in turn, may be given access to the other sources for their own use. The advantages include assimilation of data where it counts, reduced unnecessary movement of data to a central repository and thus minimised network traffic and increased access times, shared loading on workstations, and the facility to heave varying servers for varying situations. This would include mission-critical real-time servers in manufacturing environments, and slower queued servers for accounting control.

A more sophisticated distributed environment would include the ability to spread processing tasks across multiple stations. Symmetrical multiprocessing evenly splits the load between multiple processors on a first-come, first-served basis. Asymmetrical processing sends specific tasks, such as printing, to each processor. But so far network planners are not clamoring for multiprocessing capability.

As client PCs load multiple protocol stacks and we see the rise of more universal clients such as Macintoshes and PCs running Windows, all servers will look and act alike to the people who are using the resources. The developers of server software will focus on comprehensive management capabilities, the ability to add database and communications services inter networking between network environments, and the system reliability features that are necessary for mission-critical applications.

Distributed database systems face some problems that plain database servers don't. Two such problems are particularly tough.[60]

The first is dealing with storing data on multiple servers. In the example above, all the employee information was on one machine, and all the dependent data was on the other. No piece of data appeared in more than one place. That's a 'partitioned' database, and it's conceptually easy: for any one piece of data one goes go to only one place to get it.

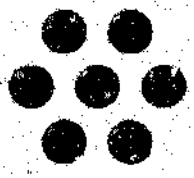
Partitioned distributed databases, however, can be expensive to use precisely because one always has to send requests to where the data is located. That process can take a lot of time over a network, and it can also turn any server that holds frequently used data into a bottleneck.

To avoid those problems, one can duplicate parts of the database on different servers and create a distributed database system with 'replicated' data. The cost here is that every time a data item changes anywhere, the database system must make sure every copy of that item changes.

The other especially difficult problem for distributed database systems is a related one: how to support multi-server updates and maintain database integrity.

To remove an employee entirely in the example, one must delete both the record of the employee and the records of all that employee's dependents. Deleting only one or the other makes the overall database corrupt. (The right answer here is to support two-phase commit--discussed later).

Truly distributed systems are in their pioneering phase, and are encumbered by the increased sophistication of required security controls and data flow control. Further, joining multiple relational tables on a single machine can be a very slow process, joining multiple tables over a distributed network, even when working with fast lines, can be impossibly slow.



Protocols, allowing communication between disparate operating systems and networks, becomes increasingly complex as the range of variations increases. As is discussed further on, standards are being worked towards, but over WANs (Wide Area Networks) as opposed to LANs, the problems augment significantly.

Plain client/server database systems don't have answers. No matter what the vendors say.

NETWARE LOADABLE MODULES (NLM'S)

One of the most important developments for LAN users is the recent appearance of database server engines which load and run on a NetWare 386 file server. The new products are called Netware Loadable Modules (NLMs).

With surveys indicating that 75% of all installed LANs are Novell based, this opens up many possibilities.[61]

Because most file servers are far too large--especially in new LANs where vendors over specify or customers buy for anticipated future needs--using this power for database engines is ideal.

Historically there have been a variety of database server products on Unix and OS/2 platforms, which need substantial RAM and a protected mode operating system--requirements not met by DOS.

Novell users face several problems with database engines, such as the foregoing, which demand expensive hardware including a second large server and disk storage.

Such database systems also introduce a foreign operating system which requires skilled staff and a communications protocol to allow the database server to communicate with the workstations.

Unfortunately these communication interfaces are bulky, unreliable and slow. In addition, database server vendors often lack the expertise to handle other operating systems, leading to all the problems associated with multi-vendor installations.

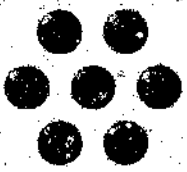
Against this background there is an impending flood of database server products being ported to the NLM platform.

NLMs offer a pragmatic solution which protects investment. They run in protected mode in a full 32-bit environment, making the most of NetWare. This strong operating system is largely unrecognised and often neglected, but offers users a proper multitasking operating system which is optimised for particular tasks. Its track record is impressivewith thousands of installed sites, it is stable and mature.[62]

Peripheral benefits are legion. Most users have fast 386 or 486 machines installed, but the file server spends most of the time waiting for LAN traffic. In some busy sites with 70 to 100 users, real-world cop processing loads typically use 10%-20% of CPU, leaving plenty of surplus power for the database engine.

The file server is usually isolated with a UPS (Uninterruptible Power Supply)--an ideal situation for a vital database engine. In addition, NLMs offer all the client-server benefits. For users who already have important file based applications on LANs, the integrity provided by the new product is a boon.

In traditional database applications, file servers are treated as mere boxes with no intelligence, and data integrity is spread very thinly across the entire LAN. If one



workstation is faulty, the result--almost without exception--is corruption. So to provide a reliable file based database operation, users have had to stabilise the entire LAN, a major management exercise consuming money and resources.[63]

The server maintains the database integrity centrally so workstation faults are immaterial, with the database rejecting incomplete or corrupt actions. Relational integrity is another benefit, because the server understands rules and relational data--in line with true client/server RDBMS (Relational Database Management System) architecture.

NLMs, beyond the database architecture concept, provide virtually all network services, including file, print, communications, and messaging in addition to database services. A key feature is the ability for the system manager to load and unload NLMs while the OS is running. Some NLMs, such as file, print and database services, are provided with the base product. Others, such as Macintosh, NFS and IBM connectivity, can be purchased as options from Novell and other third-party vendors. NetWare v4.0 allows one to run NLMs in ring 0 (operating system mode) for better performance or ring 1, 2 or 3 (protected environment) for reliability and protection.[64]

STANDALONE DBMSS

A hybrid of client/server and file server architectures is the standalone DBMS. Being suitable for single workstation implementation, or on a standard network server, it is more readily justified for development work with the intention of migrating databases to a client/server platform. More recent application (MIS) packages, however, include a standalone engine for such development or full utilisation, and thus this architecture may be justified in aiming for the advantages accruable from these recent releases.[65]

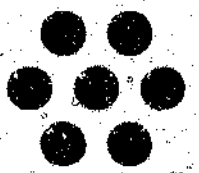
SQL database systems built on the client/server model are generally used on networks, but while a network is the perfect environment for a system that's up and running, it can be less than ideal for development work. A standalone engine can spell relief during the development process on a number of fronts.

Many things can go wrong during the development process. At the very least, as tables are being normalised and optimised there is heavy access to the server, which can compete with other client applications. Under the worst circumstances, bad queries generated during development could tie up the server and cause work disruptions elsewhere on the network.

Another consideration is the mobility of the development process. It is inconvenient for programmers to be tied to a network for development. With a standalone engine, development can also take place at home or on the road.

Standalone engines are not strictly for development; they can provide a powerful vehicle for local data access. Even when the data is not shared by multiple users, there are data management and integrity benefits to be gained by accessing data via an engine rather than accessing the files directly. With the advent of multitasking on single-user machines, it is possible to have more than one application accessing the same data at once. Concurrency (discussed later) is handled better by a central processing engine than by competing applications.[66]

There is also a scalability benefit to standalone SQL engines. In some situations, it is useful to be able to run a networked database application on a local PC; standalone engines can provide this scalability. For example, a salesperson might need to download contact data to a notebook computer and access the data on the road. The ability to run the same application on the network and on a notebook can be a major time convenience.



Finally, although they are called 'standalone engines', these products can usually support multiple users over a network using the file server rather than the client/server architecture. Although it is inefficient, it can be an inexpensive way to get a networked database up and running before investing in more expensive client/server hardware. More important is their ability to be migrated from the file server platform to the client/server. Thus a company restricted by inexperience and limited budgets, can operate in a file server environment, until an application has proved itself and merits migration to a more robust platform. Naturally the justification would fall away if the application failed, and would only be necessitated once its use started to overload the file server or multi-user security became a critical issue.

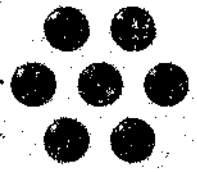
Standalone engines can take three different forms.[67]

- One common form is as a set of static link libraries of which versions for both DOS and Windows exist. This method has the advantage of limiting the number of files that must be distributed with the application. Also, because the linker extracts only those modules needed by the program, the total memory required is minimised.
- Another possibility is to use a DOS-hosted engine that runs as a TSR (Terminate but Stay Resident application) on the client workstation and is accessible via interrupts. This method more closely mimics the networked architecture found in 'legacy' mainframe environments. In these products, the developer is shielded from direct interrupt calls via a library of alias functions that match those found in the client/server version of the engine. The primary benefit of this approach is smaller executables.
- Finally, under Windows, standalone engines are shipped in the form of DLLs (dynamic link libraries). This offers several advantages, including simultaneous multi-application support and asynchronous queries.

The standalone SQL engine chosen depends largely on whether one needs to develop an application that will ultimately run on a network in a client/server configuration, or one which will be used to access local data on a PC.

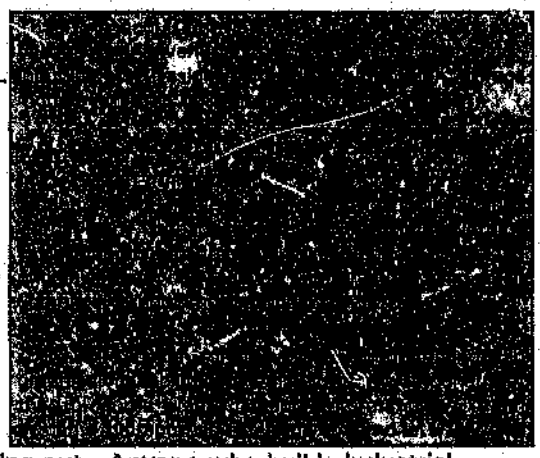
If one plans to move an application to a network, then upward scalability will be the prime consideration; the API (Application Programers Interface) for the standalone engine should be as similar as possible to the API for the networked version. To this end, a growing number of SQL back-end vendors now sell standalone versions of their database engines for DOS and Windows development.

Third-party products are a good choice in development for multiple database formats and back-end engines, or if one doesn't need the power of the client/server architecture.



DATA ACQUISITION SYSTEM (DAS)

REQUIREMENTS



As the front end to the entire IT system, the DAS is extremely important. It is often perceived as a very small module of the IT schematic, but it can represent the most work during the system compilation, and any errors introduced at this stage can impact throughout and be very cumbersome to amend at a later stage.

GIGO, garbage in, garbage out, was already a watchword of systems analysis in the '60s. The word of the '90s is NINO: nothing in, nothing out. Anyone who builds industrial-strength applications on microcomputers in the 1990s should know both terms.[88]

Those who attempt to build an information system without knowing GIGO and NINO are bound to fall prey to what is often termed the 'shrink-wrap fallacy': the assumption that serious, large-scale applications can be developed without requisite IS experience.

When applications are straightforward, the amount of data is small, and the data structure is uncomplicated, GIGO and NINO are seldom a problem. But they become dangerous when integration gets complex, the collection of data grows large, and the data structure is convoluted.

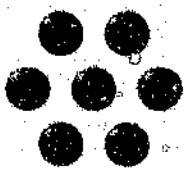
GIGO GIGO seems simple enough. Even a novice user of a basic spreadsheet program knows that if the data he enters isn't accurate and reliable, the information he gets out won't be accurate and reliable. Each data item needs to be scrutinised on input to be sure it is accurate and consistent, it conforms to established code tables or descriptions, it matches valid account numbers, and so forth. Audit trails, rollback and recovery procedures, account sum validations, cross-footed totals, and similar measures must be built into the system to ensure data integrity. Less obvious, are the small inaccuracies that, alone, seem inconsequential, yet when compiled into report totals result in inexplicable variations.

NINO NINO is an even simpler principle. One cannot get information based on certain data if one doesn't put that data into the system in the first place. However, NINO poses more difficult problems, because the 'nothings' in NINO involve both the application's data content and structure. If that structure does not provide a place for data--or provides a place in which that data doesn't fit adequately--nothing will get in.

All applications require some sort of structured collection of data. Whether it is as simple as an array of cells in a spreadsheet or as complex as a distributed, fully normalised, relational database spread over multiple servers, some data structure must exist for an application to be possible.

In the extremely simple case, the database specification may not include a field that is essential for all forms of recording, but simply did not come to the attention of the IS person due to all the other data he was aligning.

More subtly, consider a plethora of information assimilated about a series of cash flow transactions. Should the date field be left out of the database specification, seemingly



justifiably because in this case date was not seen to be of any import, then a later attempt to reconcile a report over a small time slice, against another figure, will be impossible. Less obviously, if a field for the time a record was inputted by the user is not included, it will never be possible to determine what the lag between data generation and input is. Also, if a critical field that may not represent a part of the transaction is not verified to ensure that it has been completed, this fact may not become evident until such a time as someone attempts to generate a report grouped or sorted on this field. Such grouping or sorting can be the be-all and end-all of a result, if one is attempting to identify the source of any problems, and can render the entire system, as well as all the data punching effort, wasted when needed.

Specification of a database revolves around the DAS input screen and algorithms, and the time and effort that should be put into this task should be given as much emphasis as the other modules.

FUNCTIONS

The functions of the DAS can be clearly itemised:

- Collect data, either though man-machine interface (MMI), else gateway to control systems;
- Assist user in inputting data accurately through prompts, familiar/logical layouts, clear requirements, and possibly on-line check calculations/totals for guidance and confirmation;
- Check integrity of data against predefined parameters;
- Process/manipulate data a bit to conform to storage data type in IS module;
- Pass data to IS module, possibly confirming its receipt.

CRITERIA

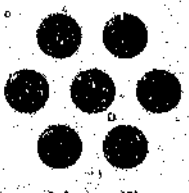
Graphical User Interface (GUI)

A data-entry screen, being the information gateway into a program, can make or break an application. A data entry screen should encourage quick, accurate input of information, with the option of reducing errors by verifying the data--not letting a user leave an important field blank, or insisting that the contents of a particular field must be unique (such as a batch number) or must fall within a certain range. Creating data-entry forms can be painless or painstaking, and a GUI can usually make this all a lot easier.

Freed from the limits of a text interface, the design tools in Windows databases encourage creation of better screens. Standard equipment for screen designs include radio buttons, check boxes, and scrollable pick lists that drop down from a field as one moves into it. List boxes, toggle buttons, text zoom facilities, help messages in status bars, on-screen graphics, titles and context-sensitive help all make input far more accurate and reliable. With colours, sizable fonts, customised input formats to force desired inputs, gone are the days where the limited number of characters that could fit on a screen meant that messages were usually ambiguous, and users couldn't see where to place their input, and why it wasn't being accepted.

Most packages let one create customised menus to help automate tasks, but an increasingly popular option is to rely on the mouse and assign a macro or programming module to an on screen push button. The mouse itself helps users to jump to input boxes without having to tab through numerous screens as was prevalent in the days of HP systems.

Sophisticated packages read a value as soon as it has been entered, rather than when the entire form is complete, allowing the value to be used in other inputs (to grey out an area if it no longer applies under the chosen selection), allowing the value to be verified



immediately so that the user knows exactly where he went wrong, displaying things like running totals and calculated figures based on the input, and even incorporating sub-forms linked to a value on the main form to restrict displayed data. These packages also have event triggers for when a user enters a field, changes it, exits it, or even double-clicks it with a mouse--facilitating all sort of assistance options.

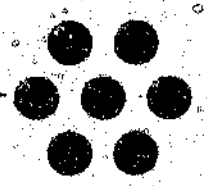
Multiple forms on the screen at the same time help eliminate errors, help screens that can sit beside the main form, cue-cards (guide through help), and various layouts such as spreadsheet like rows, or page like forms all assist in clarifying layout.

In all, few applications are more suited to the GUI environment than DAS interfaces.

PROGRAMMABILITY For sophisticated form screens that provide all the features listed above, one requires either a high level 3GL (3rd Generation Language) with an excellent command list, else a 4GL that takes all the work out of screen generation. There are probably very few areas in which the productivity increases of a GUI feature more than in screen building.

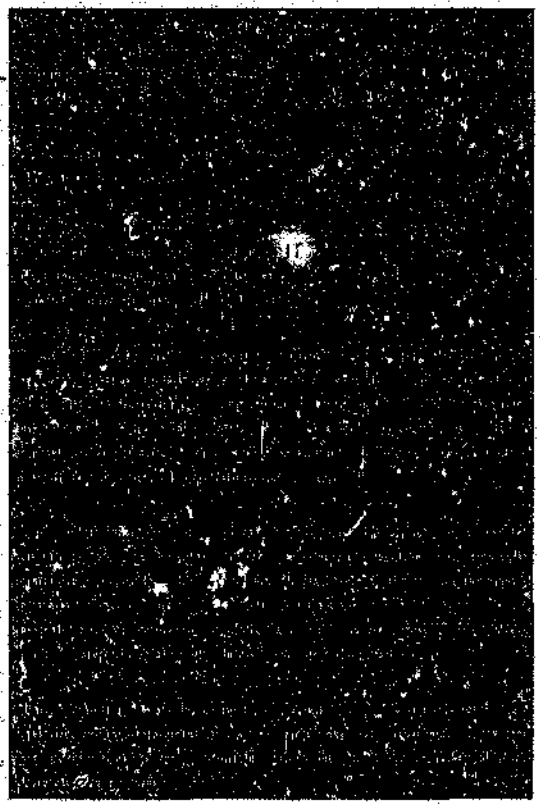
Most other required features are listed under the GUI discussion. It is fair to say that most GUI packages feature these abilities, and most character based ones don't. However, a package should be evaluated for the more subtle items discussed.

REVIEWS DAS packages usually come with the MIS (Management Information System) front end, and thus the review in this forthcoming section would apply. One should note that whilst DASs are nearly always subsets of the MIS, an MIS will not always have DAS facilities, typically an EIS (Executive Information System) wouldn't.



INFORMATION STORAGE (IS)-- DATABASE MANAGEMENT SYSTEMS (DBMS)

STRUCTURE



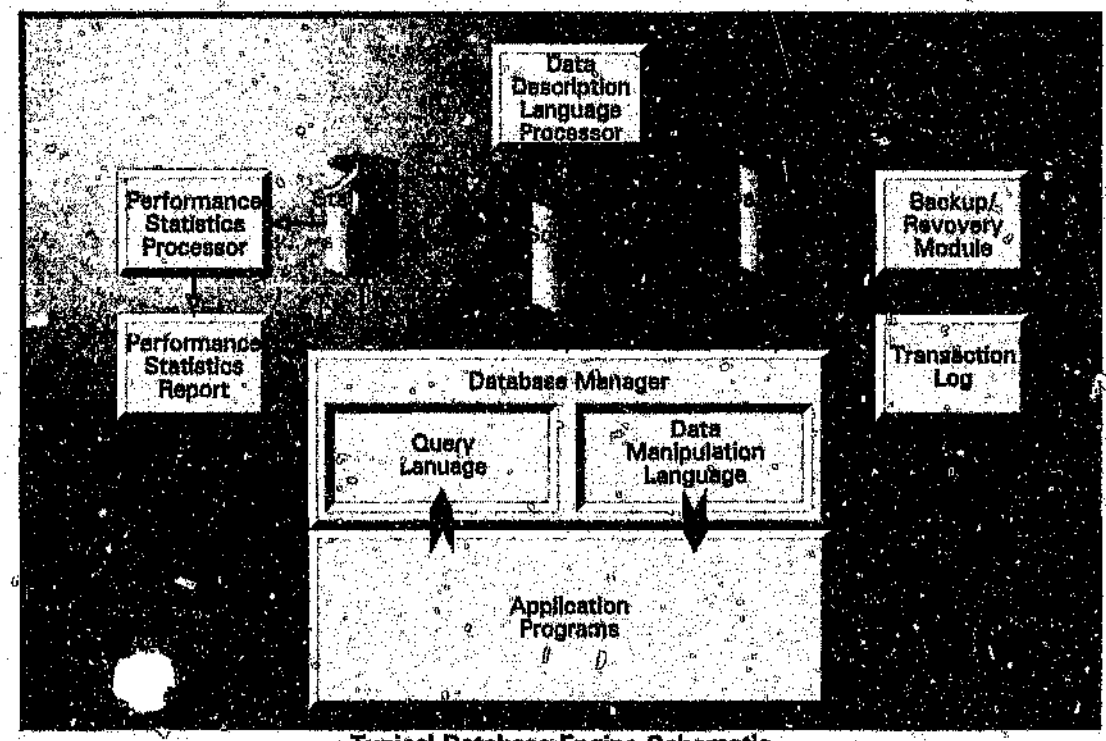
A typical database would contain the following components.[89]

DATA DESCRIPTION LANGUAGE PROCESSOR The data description language (DDL) processor transforms the data dictionary into the database schema. All DBMSs have a DDL processor which defines the data type in each field. This could typically consist of the field name, the type of data (text, number, etc.), the size of the data (text length, numeral byte size), and the number of decimal places for numeric data. It is here that Binary Data (BLOBs--discussed later) specifications start to cause complications in RDBMSs (Relational Database Management Systems).

PERFORMANCE STATISTICS PROCESSOR The performance statistics processor maintains statistics that identify what data is being used, who is using it, when it is being used, and so forth. The manager in the information services unit named the database administrator use the statistics to monitor database use. Microcomputer based DBMSs typically do not include the performance statistics processor.

BACKUP/ RECOVERY MODULE The backup/recovery module is used to recover from a disaster that makes the database unusable. The transaction log enables the reconstruction of the database from said disaster, and is essential in fragmenting architectures where any PC may go down unexpectedly for any reason.

DATABASE MANAGER All DBMSs include a database manager. The most important of the DBMS software, it handles the users data requests. The query language and the DMI (Desktop Management Interface) are a part of the database manager. The database manager also produces the performance statistics processed by the performance statistics processor, and the transaction log processed by the backup/recovery module. The database manager is the only DBMS component that is main memory resident. The others are transient routines.



Typical Database Engine Schematic

STRUCTURAL TYPE

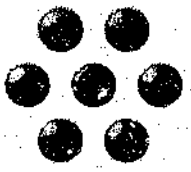
ORIGINATION At the dawn of computing, data was stored in a sequential manner on punch cards, ticker tape, and later magnetic tapes media. Access to this data was cumbersome in that every record had to be read, from the beginning, until the required point was reached. Later the magnetic disk, a direct access storage device (DASD), allowed the read/write mechanism to be directed to a particular record without a sequential search.

ORGANISATION This opened the way to information retrieval as characterised by databases today, the problem that remained was one a relating records in computers that people would do naturally.

The constraint was imposed by the way the data was physically arranged in secondary storage. Information specialists sought ways to solve this physical organisation problem, and their efforts led to something that became known as logical organisation. Logical organisation integrates data from several different physical locations. It is how the user sees the data. For example, the credit manager sees the report information as fitting together logically--all of the fields relate to a past-due-receivable. The physical organisation, on the other hand, is how the computer sees the data--as separate files. Techniques were developed to achieve a logical integration of data within a single file, and also a logical integration between multiple files.[70]

LOGICAL INTEGRATION WITHIN A SINGLE FILE Two approaches were developed that enabled the selection of certain records from a single file without the need to search the entire file. The two approaches were called inverted files and linked lists.

An inverted file is a file maintained in a certain sequence, but an index has been prepared that enables records from the file to be selected in some other sequence. The type of problem that the inverted file was designed to solve was the request by a manager for a report listing only certain records in a file. For example, a sales manager might want to see a listing of the sales made by Salesperson 23. Each record in the table had to be examined by the computer to determine if it has a Salesperson 23 record. If so, it was



selected and used to print the report. A file containing thousands of records might include only a dozen or so records for Salesperson 23, but every record in the file had to be scanned. It was very inefficient. The information specialists realised they could create an index for the Salesperson master file that listed all of the records for each salesperson. When the index is needed, it is read into primary storage and the program scans the salesperson number column looking for Salesperson 23. When that row is found the program can scan across to the right and pick up the needed record numbers. If the Salesperson master file is DASD based, the needed records can be selected without searching the entire file.

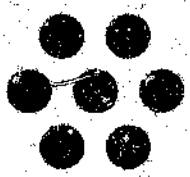
Salesperson Number	Name	Customer	Order Item	Order Quantity	Order Price	
16	#####	#####	#####	#####	#####	
20	#####	#####	#####	#####	#####	
23	#####	#####	#####	#####	#####	
25	#####	#####	#####	#####	#####	
110	#####	#####	#####	#####	#####	
2	#####	#####	#####	#####	#####	
23	#####	#####	#####	#####	#####	
94	#####	#####	#####	#####	#####	
1	#####	#####	#####	#####	#####	
23	#####	#####	#####	#####	#####	
16	#####	#####	#####	#####	#####	

Inverted File

In linked lists, another technique was used to achieve the same results. Assume that the same manager wants the same report. A separate field, the salesperson link, is added to each record in the file. The field contains a link, or pointer, that serves to tie together all of the records for each salesperson. The program selects the records by scanning each record in the file until the first record for Salesperson 23 is found. The link field in the first record is called the head, and it identifies the next record for Salesperson 23. The next record is retrieved (again assuming a DASD capability), and its link field identifies the third record. This process is continued until the last record is retrieved. The link field in the last record contains a special code that identifies it as the tail.

Index	Salesperson Number	Customer	Order Item	Order Quantity	Index Link	
1	16	#####	#####	#####		
2	20	#####	#####	#####		
3	23	#####	#####	#####	7	
4	25	#####	#####	#####		
5	110	#####	#####	#####		
6	2	#####	#####	#####		
7	23	#####	#####	#####	10	
8	94	#####	#####	#####		
9	1	#####	#####	#####		
10	23	#####	#####	#####	9999	
11	16	#####	#####	#####		

Internally Linked List

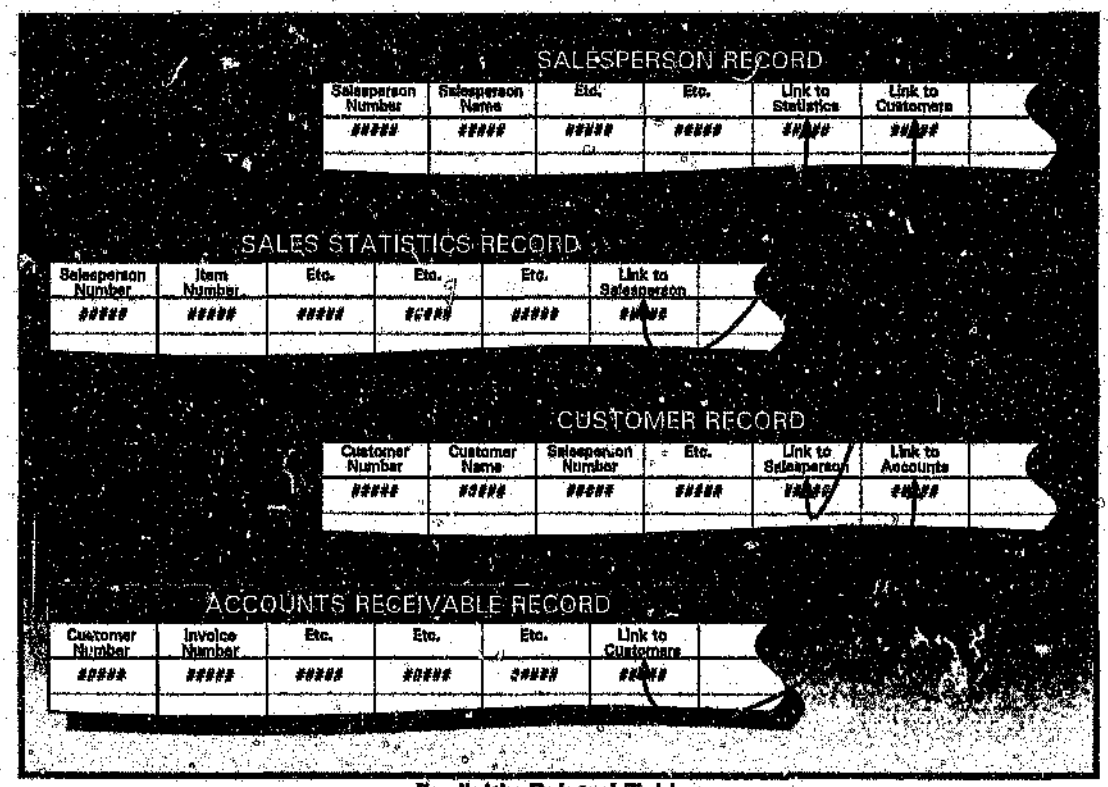


LOGICAL INTEGRATION BETWEEN FILES Inverted files and linked lists provide a way to logically integrate records that are physically scattered throughout a single file, but there is also a need to achieve the same results between multiple files.

In the mid-1960s General Electric modified COBOL to allow data to be retrieved from multiple files. Links were used to interconnect the records in one file to the logically related records in other files. GE's system was named IDS (for Integrated Data Store), and it was the first step toward an integrated database of multiple files.

EXPLICIT STRUCTURES The logical integration of files can be established explicitly or implicitly. The inverted indexes and link fields establish explicit relationships between logically integrated data. The indexes and fields exist physically and must be incorporated into the files when they are created. If they are not, the manager's request for logically integrated information cannot easily be met.

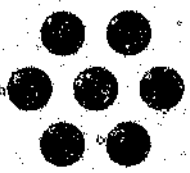
HIERARCHICAL STRUCTURE For several files using explicit relationships, relationships can be defined across the files. In the figure, each salesperson has a single salesperson record that contains data such as that listed in the figure.



Explicitly Related Tables

Each salesperson has multiple sales statistics records--one record for each sale. Each salesperson also has multiple customer records that provide information about each customer in the salesperson's territory. Finally, each of the salesperson's customers can have multiple accounts receivable records--one for each unpaid purchase.

The records in the figure are arranged in a hierarchy. Each record on one level can be related to multiple records on the next-lower level. When such a logical relationship exists between the records in different files it is called a hierarchical structure. A record on one level that has subsidiary records is called the parent. The records on the next lower level are called children. In a hierarchical structure, a child can have only a single parent. Another important feature is the link fields establishing the explicit relationships. Once



one retrieves a salesperson record (say for Salesperson 23), the links can lead one to the other records in other files that are logically related to that salesperson.

NETWORK STRUCTURE The other popular database structure that uses explicit relationships is the network structure. The main difference between the network and hierarchical structures is that in the network structure a child can have more than one parent.

THE LIMITATION OF EXPLICIT RELATIONSHIPS While the hierarchical and network structures are a giant step toward eliminating physical constraints, the approach of using explicit relationships does not offer a complete solution. It is necessary to identify the groups of files that must be logically integrated before the database is created. This restricts the manager from making special ad hoc requests for combinations of information not previously specified.

IMPLICIT STRUCTURES

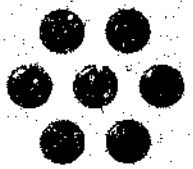
In the early 1970s Edgar F.Codd and C.J.Date, working separately, developed an approach to establishing relationships between records that do not have to be stated explicitly. In other words, special link fields do not have to be included in the records. Their approach has been named the relational structure, and it uses implicit relationships--relationships that can be implied from the existing record data.

THE RELATIONAL DATABASE MANAGEMENT SYSTEM (RDBMS) Assume we want to use two data tables to prepare a report. The data in a relational database is in the form of tables called flat files. A flat file is a two-dimensional arrangement of data in columns and rows. We want the report to list the salespersons in Territory 1, showing their salesperson numbers and their names. Both tables are needed; Table A provides the means of identifying the records for territory 1, and the salesperson names come from Table B. If a person were told to prepare such a report, he would need no special instruction, and would logically integrate the data, even though no explicit relationship exists.

The implicit relationship is established by the salesperson number field in both tables. The salesperson number field ties together the territory numbers in Table A and the salesperson names in Table B. This is the way the contents of multiple files are logically integrated in a relational database.

A relational database is characterised by conformance to 12 basic rules as postulated by E F Codd, the founder of database theory.[71] This test base has since been expanded into hundreds of rules, but for the context of this report, the 12 suffice.

- All information is represented explicitly at the logical level and in exactly the same way--by values in tables;
- Each and every atomic value in a database is guaranteed to be logically accessible by resorting to a combination of table name, primary key value, and column name;
- Null values are supported for representing missing and inapplicable information in a systematic way, independent of data type;
- The database description is represented at the logical level in the same way as ordinary data, so that authorised users can apply the same relational language to its interrogation as they apply to the regular data;
- Whilst several languages and modes of terminal use may be supported, there is one language that can express all of the following items:
  - Data definitions;
  - View definitions;
  - Data manipulation (interactive and through programs);
  - Integrity constraints;
  - Authorisation;
  - Transaction boundaries (begin, commit and rollback);
- All views that are theoretically updateable are also be updateable by the system;
- The capability of handling a relation as a single operand applies not only to the retrieval of data, but also to the insertion, update and deletion of data;



- Application programs remain logically unimpaired whenever any changes are made in either storage representations or access methods;
- Application programs remain logically unimpaired when information-preserving changes of any kind that theoretically permit unimpairment are made to the base tables (on-line database/table repair and modification is facilitated);
- Integrity constraints specific to a particular relational database are definable in the relational sub language and storable in the engine, not in the application programs;
- The data manipulation sub language enables applications and inquiries to remain logically the same whether data is physically centralised or distributed;
- If a low-level (single record at a time) sub-sub language is used, that level cannot be used to subvert or bypass the integrity rules or constraints expressed in the higher-level relational language (multiple records at a time).

As can be seen, all these rules are geared towards supporting a client/server environment with complete control being in the engine, and queries and answers simply being accepted and passed back.

THE OBJECT-ORIENTED MANAGEMENT SYSTEM (OODBMS) The big argument in today's database market is not which is the best database management system (DBMS), but DBMS gurus are busy with another question: Does the future of databases lie with the relational model or the object-oriented DBMS (OODBMS) model?

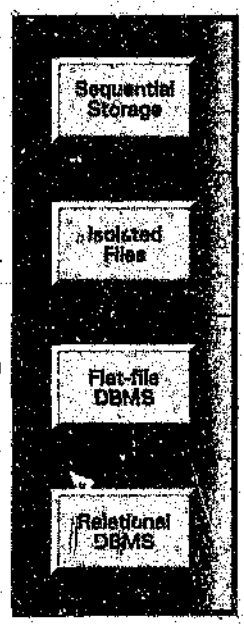
While the debate may have little meaning for the average database user, it will eventually lead to fundamental changes in how database programmers work with information. Most of today's DBMSs use a tabular range based on the relational view of data. The relational model works well with the common ingredient of databases: text strings, numbers, and Boolean logic operators. As we move through the '90s, however, DBMSs are being required to pack CAD/CAM files, scanned and bit mapped graphic images, sound and video clips, and other pieces of multimedia. Working with this kind of data, often called binary large objects (BLOBs), is difficult in the relational model, because a RDBMS must include a special set of rules that govern each kind of new data. It doesn't take many of these before the RDBMS's application programming interface (API) departs from the essential simplicity of the RDBMS design. That's where object-oriented evangelists see their window of opportunity.

In an OODBMS, data presentation is encapsulated within an object. An object is a variable that contains both the data and the code that tells how that data is to be handled. The object is then used as a discrete item. Instead of struggling to develop links between the data and the user interface, the object contains the hooks for direct user viewing and interaction. One can't beat the OODBMS approach for displaying and working with difficult data types such as graphics, but there may be serious problems when it comes to searching an OODBMS and working with alphanumeric data.

In the RDBMS world, Structured Query Language (SQL) has established itself as a powerful and easy-to-use language for data query and manipulation. There's no such standard in the OODBMS world. Some DBMS designers have suggested that object-oriented languages might fit the bill. Other DBMS designers believe that this misses the boat, since the object-oriented languages are far more difficult to use than SQL for database work. Also, since SQL is designed from the ground up for database queries, it's highly questionable that a generic object-oriented programming language could handle database work as smoothly as today's SQL-based solutions.[72]

Today, there are only a few companies with true OODBMSs, and none of them are shipping products that work on the Intel x86 platform. When and if PC versions of OODBMSs appear, they may find their path to the market blocked by RDBMS-oriented products with strong user bases, that include object-oriented extensions. The most likely evolution of the database model is toward a DBMS that, just as today's programs mix flat-file and relational concepts, combines influences from both the RDBMS and OODBMS models.

An illustration of the progression from sequential storage through to an object-oriented database is shown.



NON-RELATIONAL STRUCTURES Not all firms have achieved the degree of logical integration of their computer data described here. Such an ability is not necessary when either of two conditions exist. First, if the firm uses the computer solely for data processing, a pre database hierarchy with files as the highest level can produce the desired results. Second, if there is no requirement to integrate the contents of multiple files, there is no need to adopt the database concept.

PROPRIETARY DATA STORAGE There are many firms using the computer in a very sophisticated way without a logically integrated database. In some cases these firms have purchased prewritten software systems to process their daily transactions. This is common in the life insurance industry. The prewritten systems perform all of the necessary file maintenance, reducing the need for an integrated database and DBMS. While the integrated database and the DBMS are not an absolute necessity, there is no doubt that the trend is in that direction. This is most evident at the microcomputer level where a DBMS, an electronic spreadsheet, and a word processor are considered the foundation for end-user computing.

FLAT FILE DATABASES Another alternative is flat file databases. These are a subset of relational models in that they represent a single table with no relationships. They are simple and they assume that each set of information can be represented on one line. The most common example is the telephone book. Each person has one address, one name and one telephone number, which can all be included in one record for that person. In database terminology, each element of data in an instance (record) has a one to one relationship with each other element of data in that instance.

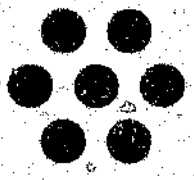
Relational database systems are often large and complex while flat file systems can be lean and efficient for simple applications. Thus, one should consider the necessity of a true relational structure before moving into this field.

SPREADSHEETS Not to be laughed off, the spreadsheet is probably the most familiar computing tool to management today. It is far easier to get managers to use a spreadsheet than to move towards a database system, especially if the latter restricts them from doing things that a spreadsheet was otherwise designed for.

Many people who need to keep track of data do so with a spreadsheet program rather than a database. One reason is Lotus 1-2-3, which for years dominated the business software market as no single program ever will again. Another reason is that, at least at first glance, a spreadsheet--especially a three-dimensional spreadsheet--looks like a great database. The rows and columns correspond to the fields and records of a flat-file database, and a 3-D spreadsheet's pages resemble the tables of a relational database.

Still, the real value of a database is in data manipulation and retrieval, and it's here--in sorting and organising information and formulating elaborate queries for retrieving selected parts of the data--that most spreadsheets, at least without help from third-party query applications, fall behind dedicated databases. Even so, its possible that a spreadsheet with enough built in database power can satisfy many record-keeping needs.

Almost all spreadsheets can sort data by row or column, letting one specify key fields. Some allow the design of custom screens for data entry; and some Windows worksheets provide for joining two or more database tables, giving at least a modicum of relational functionality. In short, if ones database needs don't go much beyond keeping records of clients or customers, almost any spreadsheet can do the job.



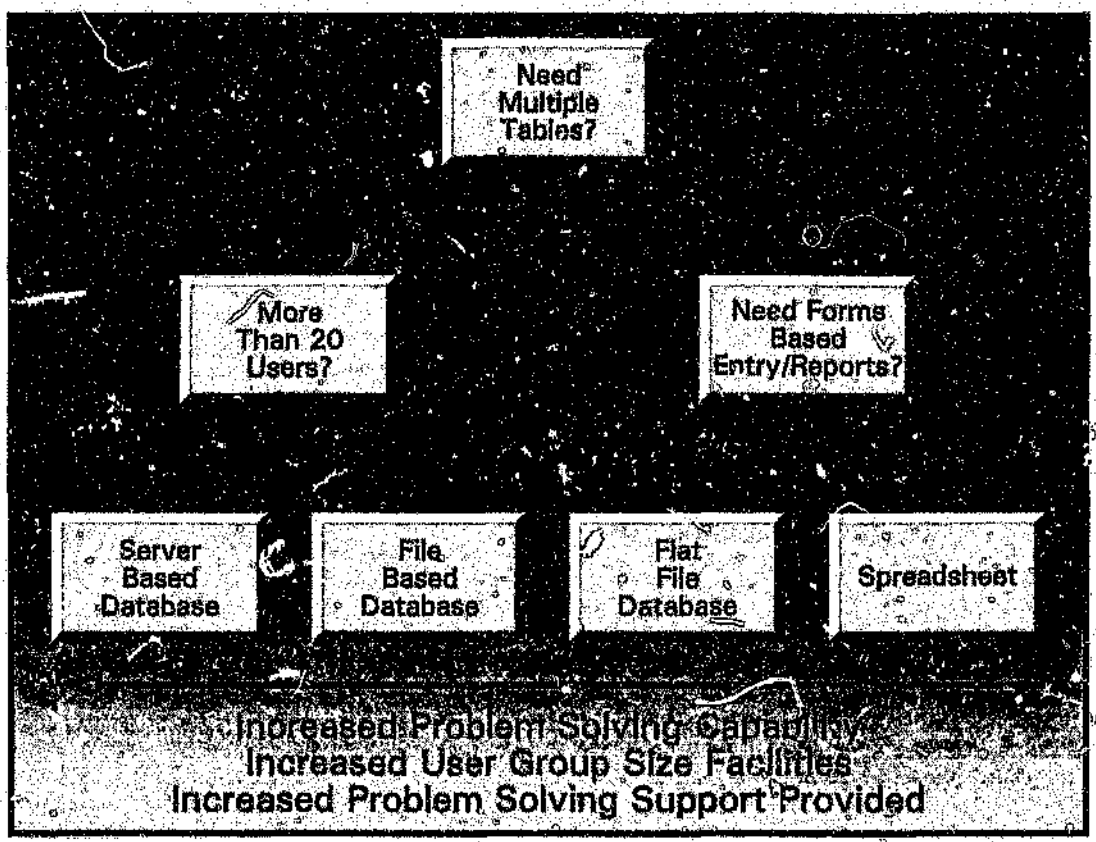
Whilst a database query and report cannot readily handle amortizations, financial return calculations, and more importantly, rapid graph compilation and amendment, there are very definite reasons for using them over spreadsheets.

For any level of sophistication and where relationships have any bearing, the above starts to fall down. More relevant are the limitations on spreadsheet size. A database can be virtually unlimited in size as it is stored on the DASD media, and read from there. Spreadsheets have to be loaded into PC memory--and even when this is possible, the system will slow down at high loads (high being considerably less that a databases limits). Single user limitations to access become desirable when one considers how easily a spreadsheet can be irrevocably destroyed by the removal of a single column by an ignorant user, hidden formulas also mean that only the original designer will ever understand how it works--reading someone else's complex sheet is a nightmare, unlike structured language programs. Moreover, spreadsheets, while exportable, are not accessible through other applications and remain islands of automation bucking the ethos of open systems and data distribution. In all, they are defiantly a step backwards unless dealing with a situation that cannot warrant a true DBMS.

Spreadsheet vendors don't surrender that easily; the current trend is to bundle a spreadsheet with a fairly complex data query facility. Quattro Pro for Windows has its DataTable DeskTop, Excel has ODBC, and QA-SuperCalc has Silverado, to name a few. Allowing read access to some databases, these do not address the storage aspects listed above.

In sum, while there are several ways to use a good spreadsheet as a database, when it comes to serious data handling, a powerful relational database will always win the day.

OVERVIEW A decision tree for determining the type of database required is illustrated below. Seemingly simplistic, this is all that is really required to make this decision, whereafter the actual vendor application will have to be given considerably more thought. The spreadsheet application leg does not refer to SQL hooking as will be discussed later, but simply indicates that, as prevails in desktop systems, a spreadsheet can be used to store limited data--as discussed above.



The DBMS Decision Chart

FUNCTIONS

The main functions of a DBMS are listed below. These are not to be confused with the criteria one would apply in assessing a DBMS as are listed in the following section.

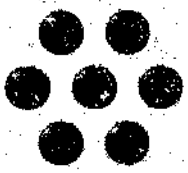
- Multi-user access control security
- Multi-user integrity control through file/record/field locking
- The organisation of data which facilitates a reduction in data redundancy (duplications)
- The integration of data from multiple sources
- The retrieval of data, often in a specified format, and the processing of queries (in a client/server environment)

CRITERIA

Choosing the right database server, can be a painful process. Database servers are complex and bristle with high-tech features, and evaluating them can become a full-time job. Selecting a database server is a process whose results will affect all those who need access to the data the server will manage.

Fortunately, by recognising the different criteria a database server should meet, buyers can simplify and speed the database-server selection process.

Database servers are part of an overall environment that includes a network, a server system, and one or more types of clients. Most organisations evaluating database servers will already have at least part of this environment in place. Thus, how well each database



server meshes with the existing hardware server and network environment is a key consideration.

CLIENTS Because the data in a database server is usable only when it is available to client applications, the compatibility of each database server with the client front ends being considered is another key issue. In addition, as with any product set, a feature-based comparison is an important part of the selection process.

Buyers should also consider the type of front-end database software the clients will run. The client software must be compatible with the server's database system. This can lead to a chicken-and-egg dilemma: Should buyers choose the database server or the front-end packages first?

For many buyers, it makes sense to choose the front-end products first. Users might already be familiar with a particular PC stand-alone database system, so using that system as a front end to the new database server--assuming that is possible, of course--could save user training costs. Choosing the front end first also makes sense because most users will be more concerned with the software they see and use than with the database server, which operates, to them, largely behind the scenes.

Another sensible option is to pick a small number of front ends so that different users can use the software they like best. Hard-core developers, for example, may want powerful front ends such as 3 & 4GLs (3rd & 4th Generation Languages), other users may want the simplicity and ease of use of such tools such as hooks from spreadsheets to the database.

NETWORK A typical network environment will contain one or more servers and one or more types of networks. Any database server that blends seamlessly with the existing setup will thus have a cost advantage over those that do not.

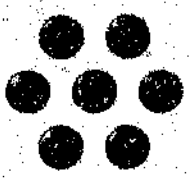
Of course, if a database server's performance or features on a particular platform are so compelling that they make the cost of adopting the new platform worth bearing, then the type of the existing servers need not matter. For most buyers, however, the first step in the database-server selection process will be to identify which products can run on existing servers.

Some database servers may also be able to run under different operating-system software, but on the same hardware. A database server that runs on X86-based Unix or OS/2 servers and supports NetWare's SPX/IPX (Sequenced Packet Exchange/Internet Packet Exchnage) network protocol, for example, may work just as well as a NetWare Loadable Module server in a NetWare environment. The more a database server can 'speak' the same network protocols as its clients, the less work it will take to get that server running.

The reverse is also true: If a database server cannot use the same network protocols as the existing clients, adopting that server means changing all the clients. If, for example, a database server works only with TCP/IP (Transmission Control Protocol/Internet Protocol), then either that server will not work in a NetWare environment or it will require the clients to be changed so that they, too, can work with TCP/IP.

Such changes can be expensive because they affect multiple clients and can create a great deal of setup and configuration work for network administrators.

FEATURES The third area to consider when selecting a database server is the most obvious one: the features of the different candidates. With database servers, features that may sound like mere technical widgets can prove to be critical for certain applications. For this reason, the more prominently featured ones (featured in vendor blurbs/seminars, and featured in industry reviews) are listed below, though several more can be seen in the review table in the reviews section.[79]



BACKUP & RECOVERY This can often sound high-tech but can translate into bottom-line benefits when something goes wrong and a database gets corrupted by a system failure. When such disasters strike, the ability of the database server to roll back, or undo, all incomplete transactions is vital. Incremental backups are also vital for databases too large to back up fully on a regular basis.

Many SQL databases provide rollback capability so that if both transactions are not committed, neither will be--the partial changes will be undone. For example, when moving money between accounts, data integrity may demand that a debit in one place be balanced by a credit in another. Both transactions must be committed to the database or the data will no longer be consistent.

Recovery should not have to be facilitated through the network operating system providing mirrored disks or copied databases--facilities designed to cover a hardware breakdown.

TWO-PHASE COMMIT This ensures data integrity when updates are being made in a distributed database environment. With two-phase commit, the update to the database takes place in two steps: a prepare-to-commit step, and then--if all the relevant engines respond with a 'ready' signal--the actual commitment step. An inability to serve this function can lead to incompatibility with other databases if more that one server is combined in the system. The result is a necessitated API to bridge this gap--an unnecessary complication.

CONCURRENCY Any organisation that has multiple users simultaneously updating a database will want to examine each database server's concurrency controls. These controls manage the locks needed to let many users work with database data at once. The levels of locking the server supports are important because they determine at what level users can share a database. If a server locks entire tables, then only one user can be updating any given table at a time. The lower level at which a table is locked during updates--whether at the page level or the record level--can greatly influence access speed. Ideally, only the particular record being updated is locked, since that gives other users access to as much of the data as possible.

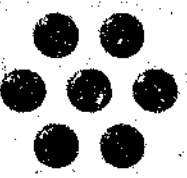
When updates are being processed for many records, it's more efficient to coarsen the level of granularity and lock the entire page, rather than locking multiple records. This conserves the number of locks available to the system as a whole. Most SQL databases provide automatic lock escalation, which changes the level of granularity of the lock depending on the amount of data being updated.

DEADLOCK CONTROL As applications become heavily used, the weakness of a simplistic record-locking scheme becomes apparent: If the first user doesn't release the record in short order, other users are made to wait. When two or more users have their requests in, the threat of a deadlock exists, and everyone waits for one of the others to release a lock.

QUERY LANGUAGE Each database server provides one or more languages for manipulating its data. Structured Query Language (SQL) is the modern standard for servers, but there are many varieties of SQL. The completeness of a server's SQL implementation and its level of adherence to the ANSI SQL standard can be an issue for organisations that already use mini or mainframe SQL-based database systems.

As covered later, SQL has taken on many enhancements through various vendors. The compatibility of the final outcome, with other applications, and the advantages of each enhancement need to be considered.

SQL OPTIMISATION SCHEMES Aside from the DBMSs ability to use SQL queries, is the more subtle, yet more important feature of how well this works. The goal of optimisation is to ensure that an SQL query returns results as quickly as possible with the fewest number of



hits against the database. There are two types of SQL optimisation: syntax-based optimisation and the newer, more effective cost-based optimisation.

Syntax-based optimisation exploits the fact that in SQL, the way one specifies a query can drastically alter its efficiency. With syntax-based optimisation (also called rule-based optimisation) a set of rules based on the physical position of expressions in the WHERE clause lets the programmer know exactly how a query will be implemented. It takes skill on the part of the programmer to derive benefit from this knowledge. The advantage of placing the onus on the programmer rather than the database software is low overhead on the server. However, if an SQL code generator is being utilised, or if the programmer is not experienced, there will be no g in.

With cost-based optimisation, statistics are gathered about the database--number of tables, number of rows, types of data in each row, availability of indexing for a particular column, and so on—and the optimiser makes use of this knowledge to determine the best plan for query execution. If the optimiser determines that a full table scan would be more efficient than using an index to process a particular SQL statement, then the index would not be used no matter how the SQL statement was worded. The advantage of the cost-based method is obvious: The expertise for determining the best way to implement a query is transferred from the user to the database engine.

The disadvantage is that finding an optimal solution can take a long time. In fact, there can be so many ways to implement a query that a complete evaluation of all of them could take longer than the longest-executing solution would take. The search can be aborted by setting a time limit or the optimiser can be programmed to accept a 'good enough' solution.

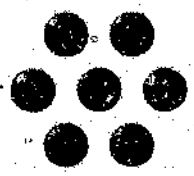
SIZE CONSTRAINTS While most modern database servers have more than enough capacity for most needs, it's still wise to verify that each potential database server can hold all the tables and records an organisation is likely to need.

DATA TYPES One particular type of data is worth noting separately: binary unformatted data. Most database servers are designed expressly to manage record-oriented data. More and more organisations, however, need to store and manage such large chunks of data as text files, graphic images, sound files and even full-motion video. Many database servers attempt to meet this need by storing these files as unformatted objects commonly known as BLOBs (Binary Large Objects).

PRICE No matter how technically correct a product may be, for most organisations that product's price will remain a factor. Different database vendors use different pricing schemes, so accurate comparisons demand care and attention. One server's pricing, for example, might be based on the maximum number of active users possible, while another's might be based on system type. Another unexpected suprise may come in the additional charges imposed for gateways/protocols required to access existing systems, as some systems don't automatically incorporate the variations.

CONSISTENCY Consistency refers to data integrity. Databases organise information by breaking it down into related data elements. A database with integrity upholds rules that govern the relationships between its data elements: for example, "all employees must have a social security number"; "invoice amounts must be greater than 0"; or "whenever an amount is posted to a journal, a corresponding entry must be posted to the general ledger."

Databases are consistent when all of the data conform to the rules. Unless the DBMS has a way to enforce the rules, a database may become inconsistent after it is updated. Sometimes a complex change forces the database to be inconsistent temporarily, but in the end, order must prevail. The unit of work that takes the database from one consistant state to another is called a transaction.



REFERENTIAL INTEGRITY (RI) Relational databases are made up of separate tables that can be joined on the basis of common information. If a parent record is deleted but the related child records are not, the child records are orphaned. RI simply means that no orphans are permitted in any table. There are three ways that a record can become orphaned: the parent record is deleted, the parent record is somehow changed so the relationship is lost, or a child record is entered without a parent record.

Without RI built into the DBMS, the front end has to maintain control. This means that simple front ends are not allowed, any changes have to be carefully vetted, and future migration to new applications is compounded. This is the way old mainframe applications work, and is the largest disadvantage of legacy applications--it does not conform to the ethos of client/server.

Most SQL databases today include RI features as part of the database engine but implement them in different ways. Declarative RI, is implemented through keys that are stored within the database tables themselves. Parent tables contain primary keys that are composed of foreign keys found within each of the child tables. This is the preferred method, following the ANSI Level 2 Integrity Enhancement addendum.

Another way to implement RI is through the use of tr[illegible]lled procedural RI. Triggers ensure RI by automatically executing rule-s[illegible]ments whenever an UPDATE, DELETE, or INSERT command is encounte[illegible] exa[illegible] whenever a parent record is deleted, either all the associated child record[illegible] also deleted (cascading deletes) or the deletion is disallowed.

Procedural RI, does not fully comply with the ideal of the referential model for data storage--storing all information about the tables within the tables themselves--but does give the database developer a front-end-independent method of safeguarding RI.

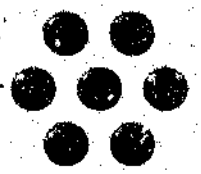
CONNECTIVITY AND PORTABILITY This determines the product's suitability for distributed database environments. For connectivity, the breadth of support for gateways to minicomputer and mainframe systems is considered, along with the presence of features such as two-phase commit, distributed recovery and distributed deadlock detection. Portability is measured by the number of platforms supported by the package and the consistency of features across platforms.

STORAGE ARCHITECTURE The basic storage architecture of the system includes features such as disk spanning and striping, hashed index support, and asynchronous I/O capability. The number of import and export formats supported should also be considered, along with image data support and distributed processing capabilities such as cross-device table joins and the ability of the server to send queries to another server. Naturally the server architecture that the DBMS runs on also features here.

SPEED Reading, writing and indexing speeds will affect the organisation, and become more important as the number of users is increased. This is aside of query speeds.

SECURITY All client/server based DBMSs provide log in security to control user access. On file server based applications a limited amount of control is facilitated through database encryption and locally stored password tables. These, however, are often simplistic and can be bypassed through separate instances of front end tools, they also present some risk in that encrypted data cannot be recovered in the event of a hardware disaster, and hooks from other front ends cannot readily deal with the encrypted formats. In short, a client/server is necessitated when data security is required.

For these secured databases, the engine should minimise the inconvenience of logging in, allowing the passwords and user names to be built into queries and passed with the query. In addition, any convergence with the network operating system security, and that of any other databases on the system, is highly desirable where this does not compromise security.



MIGRATABILITY This is covered in more detail under the management information system (MIS) module section. It refers to the ability of the database to be transported onto a client/server architecture server without front-end applications and the database engine and built-in procedures having to be re-written.

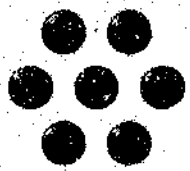
OS PLATFORM The OS that the database is required to run in must meet with the criteria specified in the NOS (Network Operating System) section of this report. As was mentioned, should the benefits of the particular DBMS outweigh the complications of integrating a new OS into the network system, then this factor is overridden.

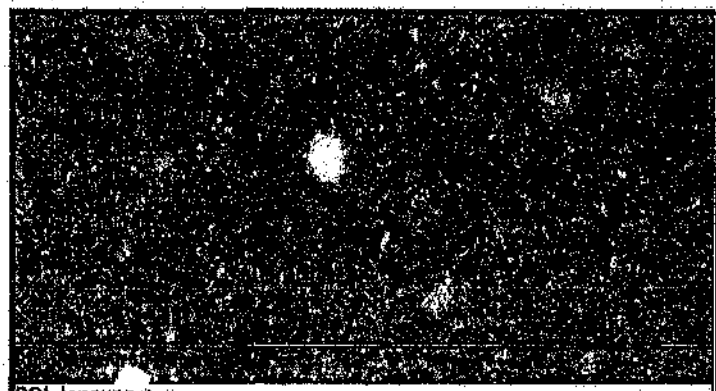
REVIEWS

The following tabular comparison of the current leading IS servers is oriented towards the scope of this report in that it primarily addresses server architectures suitable for client/server implementation. Many PC based multi-user packages with adequate database engines exist, most of the MIS packages featured later support these capabilities, but only dedicated servers conforming to the RDBMS rules have been reviewed.

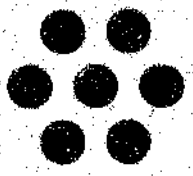
These packages represent the industry standards in SQL server installations in that they dominate the market.

Stored procedures:							
Compiled SQL stored on server	☒	☑	☑	☑	☒	☑	☑
Compiled 3GL code stored on server	☑	☑	☒	☒	☒	☒	☒
User can execute stored procedures from command line	☒	☑	☑	☑	☒	☑	☑
Stored SQL procedures can use control logic	☒	☑	☑	☒	☒	☒	☑
Triggers:							
Number of triggers allowed per SQL statement	0	0	3	0	0	0	3
Number of times trigger can cascade	0	0	16	0	0	0	16
Trigger rules can be applied by column	☒	☒	☑	☒	☒	☒	☑
Referencial integrity:							
Declarative RI via keys	☑	☒	☒	☑	☒	☑	☒
Procedural RI via triggers	☒	☑	☑	☒	☒	☒	☑
Domain integrity:							
Can define new data types	☒	☒	☑	☒	☒	☒	☑
Domain integrity enforcement on server	☑	☒	☑	☒	☒	☒	☑
Backward and forward scrolling:							
Implemented on server	☒	☒	☒	☑	☑	☑	☒
Implemented on client	☒	☒	☑	☒	☒	☒	☑
Hashed indexes	☒	☒	☒	☒	☒	☒	☒
Hashed clustered indexes	☒	☒	☑	☒	☒	☑	☑

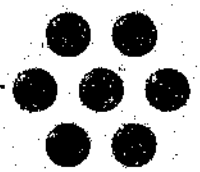




SQL language:							
Full ANSI compliance level	L2+	L1	L1	L1	L2	L2+	L1
DB2 compatibility	☒	☒	☒	☑	☑	☒	☒
SQL command optimisation:							
Cost based	☑	☑	☑	☒	☒	☑	☑
Syntax based	☒	☒	☑	☑	☑	☒	☑
3GL support:							
C API	☑	☑	☑	☒	☑	☑	☑
Fortran API	☑	☒	☑	☒	☑	☑	☑
Rexx API	☑	☒	☑	☒	☒	☑	☑
Basic API	☒	☒	☑	☑	☒	☑	☑
COBOL API	☑	☒	☑	☑	☑	☑	☑
Pascal API	☒	☒	☑	☑	☑	☑	☑
Assembler API	☒	☒	☑	☑	☒	☑	☑
C compiler	☑	☑	☒	☑	☑	☒	☒
COBOL compiler	☑	☒	☑	☒	☑	☑	☑
Fortran compiler	☑	☒	☒	☒	☑	☒	☒
Database backup and recovery:							
On-line backup	☒	☒	☑	☑	☑	☑	☑
Roll forward	☑	☑	☑	☑	☑	☑	☑
Roll back	☑	☑	☑	☒	☑	☑	☑
Separate mirroring of transaction log and database	☒	☒	☑	☑	☒	☒	☑
Transaction logs can be stored separately	☑	☑	☑	☑	☑	☑	☑
Security:							
Assign access by group	☑	☒	☑	☑	☒	☒	☑
Assign access by user	☑	☑	☑	☑	☑	☑	☑
Disk file management:							
Supports data types larger than 2GB	☒	☒	☒	☒	☒	☑	☒
Spanning of database	☒	☑	☑	☑	☑	☑	☑
Spanning of transaciton log	☒	☒	☑	☑	☑	☒	☑
Striping of database	☒	☑	☑	☑	☒	☑	☑
Stripping of transaction log	☒	☒	☑	☑	☒	☒	☑
Asynchronous I/O	☑	☑	☑	☑	☒	☒	☑

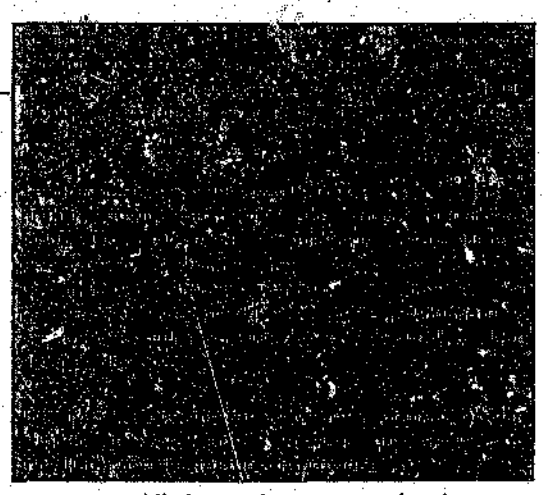


Locking granularity:							
Record level	☑	☒	☒	☒	☑	☒	☒
Page level	☒	☑	☑	☑	☑	☑	☑
Table level	☑	☑	☑	☑	☑	☒	☑
Automatic lock escalation	☑	☑	☑	☒	☑	☑	☑
Minimum page size	4k	2k	2k	.5k	2k	1k	2k
Read consistency levels:							
Cursor stability	☑	☑	☑	☑	☑	☑	☑
Repeatable read	☑	☑	☑	☑	☑	☑	☑
Multi-versioning	☒	☒	☒	☒	☒	☑	☒
Release locks after reading	☑	☑	☑	☑	☑	☑	☑
Uncommitted (dirty) reads	☑	☑	☒	☒	☒	☒	☒
Deadlock detection:							
Cost based	☑	☒	☑	☒	☒	☒	☑
Aborts transaction	☑	☑	☑	☑	☑	☑	☑
Aborts via timeout	☑	☑	☒	☑	☑	☒	☒
Can connect to other servers	☑	☒	☑	☑	☑	☑	☑
Can connect to other platforms	☑	☑	☑	☒	☑	☑	☑
Can join tables stored on different servers	☒	☒	☒	☑	☑	☒	☒
Can join tables stored on different platforms	☒	☒	☒	☒	☑	☒	☒
Two-phase commit	☒	☒	☑	☒	☒	☒	☑



MANAGEMENT INFORMATION SYSTEM (MIS)--DATABASE CLIENTS

OPTIONS



There are three main types of SQL front-end development tools: the new crop of dedicated SQL front-end building 4GLs; SQL connectivity libraries for use with traditional programming languages such as BASIC, C, or COBOL; and SQL hooks from other desktop applications to the database.[74]

FOURTH GENERATION LANGUAGES (4GLS) Most dedicated SQL front-end development tools employ an event-driven procedural language. Forms are designed graphically using visual objects and non-object-oriented code attached to these objects to handle various events. All the tools support simple events like entering a field or clicking a button, but the tools differ in the quantity of events they support and the degree to which they can be expanded.

4GLs compile these objects into a code that operates completely transparently to the developer, with the developers augmented input being through a 3GL module that gets incorporated into the code. This type of programming is very object oriented, and gleans all the advantages of such a philosophy including reusable modules, simplified upgrading, expedited programming, and simplified logic.

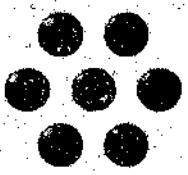
The largest advantage of 4GL is their useability to non-programmers. This may be very important in any industry dominated by Accountants and Engineers, and where the resources of a full-time programmer are not always available. With a limited knowledge of databases and their operation, any intelligent person can make full use of a 4GL to create sophisticated applications.

THIRD GENERATION LANGUAGES (3GLS) The chief strength of dedicated SQL development environments is their high-level support for database operations; this can also be their most serious drawback. For example, if part of the application involved non-database activity such as communications support, a high-level tool might not provide the functionality required. A 3GL such as BASIC, C, or Pascal is sometimes necessary to accomplish the task.

All of the SQL engine vendors provide access libraries for use by 3GL programmers, although support may not be available for both DOS and Windows. These libraries usually include functions that enable one to connect to a database over a network and functions for accessing data.

Most SQL access libraries are written to be callable from the C programming language. Developers working in other languages or in a high-level development environment can use wrapper libraries, which are provided either by back-end vendors or by a third party. Wrapper libraries provide translation of unsupported data types and supply high-level alternatives to functions that cannot be used directly.

SQL HOOKS In addition to dedicated SQL access libraries, there are many products designed to connect local databases to SQL databases. SQL add-ons are available from the vendors of many database systems. These products provide a means for translating record-oriented



operations into SQL operations so that a 4GL (fourth-generation language) application can be ported to a client/server configuration with ease. They also provide pass-through functionality for access to engine features without parallels in 4GL.

For some products, there is also a healthy third-party add-on market. These products attempt some translation of record-oriented functions, but they are not nearly as transparent as access tools provided by the vendors; they are primarily used for in-line SQL, similar to 3GL SQL access.

As was mentioned under the server architecture discussion above, the many libraries and add-ons let one connect to SQL databases from almost any application. A surprising number of products provide a way of hooking into SQL.

FUNCTIONS

The MISs main functions include:

- Generation of the query in a format that the IS engine will understand. This generation can be facilitated through Query By Example (QBE) and Graphical QBE (GQBE), through tabular assistance, else just through an error checking text editor;
- The processing of the query in a non-client/server environment;
- Data manipulation to convert the data into information (summations, averages, groupings, etc.);
- The presentation of results in a usable format (tables/graphs).

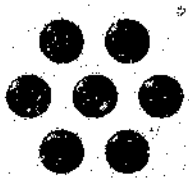
CRITERIA

The following is a list of the more important criteria used when assessing an MIS product, several less important ones appear in the review table in the review section below.[75]

SERVER SUPPORT The single most important consideration when choosing a front-end tool is which back-end engines it supports. There has been a race among vendors to have their tools support as many SQL databases as possible. At the same time, there has been a movement within the industry to adopt a standard so as to make the form of the data irrelevant (see the section on transparency). This approach, however, risks falling prey to the lowest-common-denominator pitfall. Back-end vendors are constantly striving to distinguish their products by adding features that go beyond existing standards. The real test of the emerging API (application programming interface) standards will be their ability to take advantage of the unique features of specific back-end engines.

Even today, with no standards in place, an important consideration when selecting an SQL front end is its 'awareness' of the back-end engine--the depth of its support for an SQL engine's unique features. For example, Sybase's SQL database engine includes a COMPUTE BY syntax. Many front-end tools cannot generate this statement or provide syntax checking for it. Also, some engines have unique data types, such as the BLOB data type for image data. When the front-end tool presents one with a list of data types, are special data types such as this on the list?

QUERY TOOLS Query and report tools provide a quick and easy way to generate queries and formatted reports. A key criterion when evaluating query and report-writing tools is their accessibility to the end user. How difficult is it for the average user (without any knowledge of SQL) to construct a query or report? Is the interface intuitive? Are the output options flexible? If certain SQL statements cannot be generated by the menu system (such as nested queries), does the user have the option of going in and actually writing SQL statements? Can the generated SQL code be tweaked to improve performance?



Desirable tools include Query by Example (QBE), an intuitive way of compiling SQL code graphically. This type of interface allows applications to be written that can be passed onto managers with no programming experience for them to extract their own data--part of the whole IT ethos.

ANALYSIS TOOLS Analysis tools go beyond the bare minimum of retrieving raw data for viewing on-screen or on a printout. They massage raw data into useful information. The typical manifestation of such tools is an Executive Information System (EIS). Data is retrieved and then represented in an easy-to-understand format such as a graph or summary table. Very often there is some way to 'drill down' from a more general display to a more specific one.

For example, if a pie chart is displayed with the elements Products, Services, and Investments, the user might be able to click on Products to see a graphical representation of the distribution of sales by product. This hierarchical analysis of data is the hallmark of most EIS systems.

Another strength of some analysis tools is their ability to update the currently displayed data periodically via timed or event-driven queries. Some can set 'alarms' that are triggered when certain conditions are met in the data. For example, if one needs to know when a stock price drops below a certain cutoff, one could set an alarm that would color-code the data in red.

Most analysis tools have the ability to access and integrate information from a variety of sources--various SQL engines and different file formats--for analysis and presentation as a single unit.

REPORT TOOLS Although database output has evolved over the years, most packages follow the same basic outline for creating a report. The most popular format is the columnar report, with fields listed side by side and information printed in columns.

When more than one line is required to display a record, a free-form report format is necessitated. To produce mailings, reports that fit data into predefined formats for labels, envelopes, or other media are desirable. Some packages can create cross tabs, a variable report option that spreads field information into a spreadsheet format for easier analysis. Using one field in the table for row values, and one for column values, the package summarises or counts a third numeric field, adding the values where the rows and column values intersect. For example, one can list products as rows, months as columns, and generate a list of sales by product by month.

Most packages save some time creating a basic report format, and the more sophisticated ones use cue cards and 'wizards' to guide one through the process asking pertinent questions in familiar terminology--all moving towards assisting non-programmers in report generation.

Another desirable feature is 'double pass' report generation where totals can be used before the end of the report is reached. Thus, in any record, a value can be shown as a percentage of the total for that field. Older packages do not know the denominator until the report has finished printing.

Reports are one area in which Windows databases surpass those on the DOS platform by far. Instead of menus, Windows packages use one or more dialog boxes to help define the report. Using a mouse, one can quickly move fields and column headings around. One can add graphics the headers or even for each field, and some packages allow one to add a graphic if a certain condition is met (e.g. sales exceed target for that record). Fonts can be applied easily, and the report can be easily previewed before printing.

DBMS ABILITIES For a product that may be developed in a file server environment, and for purely file server databases which come in the same package as the MIS, certain DBMS (Database

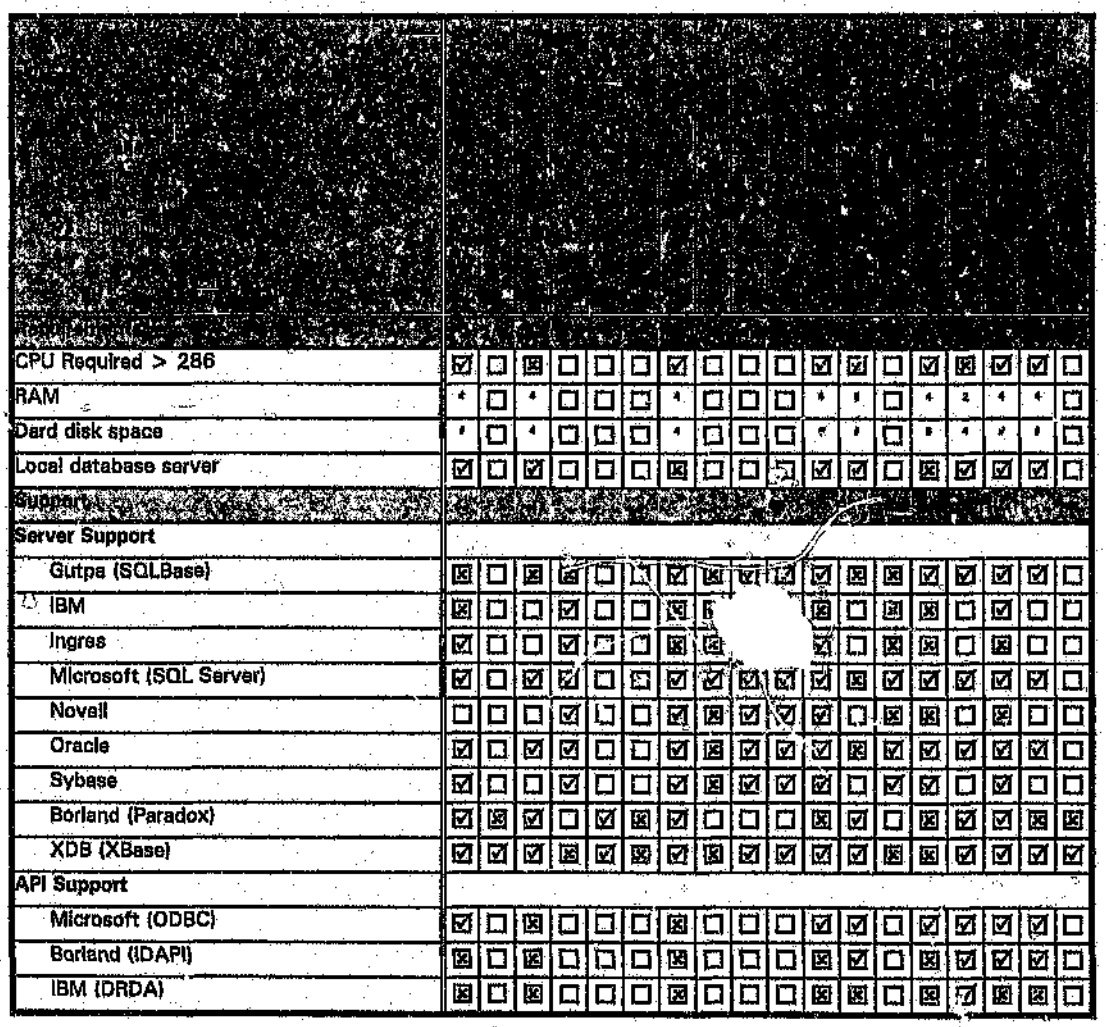
Management System) abilities are important. A comprehensive list of these can be seen under the IS section, but two stand out.

The first is the ability to cater for multiple users of a single database. Whilst conditions may exist such as the database must be accessed from the same MIS, or one with compliance with the database format, the package should be able to control file, record and field locking, though perhaps without any varying levels of granularity.

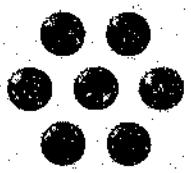
Secondly, a certain amount of security should be established to prevent the casual user from entering restricted areas and wrecking havoc. Whilst the degree of controlled access is severely limited, facilities such as data encryption should help to restrict access.

REVIEWS

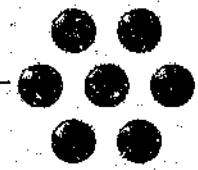
The following tabular comparison of the current leading MIS packages is oriented towards the scope of this report in that it primarily addresses PC based packages, and favours those providing a graphical interface--desirable for management usage. The favoured choice is not given here, but will be elaborated upon in the forthcoming section on implementation of the IT system.



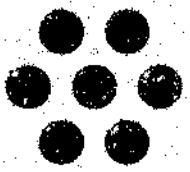
CPU Required > 286	
RAM	
Hard disk space	
Local database server	
Support	
Server Support	
Gupta (SQLBase)	
IBM	
Ingres	
Microsoft (SQL Server)	
Novell	
Oracle	
Sybase	
Borland (Paradox)	
XDB (XBase)	
API Support	
Microsoft (ODBC)	
Borland (IDAPI)	
IBM (DRDA)	



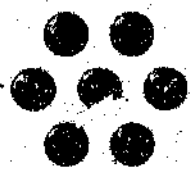
Export/Import Facilities																		
dBase	☑	☑	☑	☐	☑	☑	☐	☐	☐	☐	☐	☑	☐	☐	☐	☐	☐	☑
Lotus 1-2-3	☑	☑	☑	☐	☑	☑	☐	☐	☐	☐	☐	☒	☐	☐	☐	☐	☐	☑
Excel	☑	☒	☑	☐	☒	☑	☐	☐	☐	☐	☐	☑	☐	☐	☐	☐	☐	☑
Paradox	☒	☒	☑	☐	☑	☒	☐	☐	☐	☐	☐	☑	☐	☐	☐	☐	☐	☑
Quattro Pro	☒	☒	☒	☐	☒	☒	☐	☐	☐	☐	☐	☑	☐	☐	☐	☐	☐	☒
Text	☑	☑	☑	☐	☑	☑	☐	☐	☐	☐	☐	☑	☐	☐	☐	☐	☐	☑
OS Platforms																		
DOS	☒	☐	☐	☒	☐	☐	☑	☒	☑	☒	☒	☐	☒	☒	☐	☒	☐	☐
OS/2	☒	☐	☐	☑	☐	☐	☒	☑	☑	☒	☑	☐	☒	☒	☐	☒	☐	☐
OS/2 migratibility	☒	☐	☒	☐	☐	☐	☑	☐	☐	☐	☒	☑	☐	☒	☑	☒	☒	☐
Windows	☑	☐	☐	☑	☐	☐	☑	☑	☒	☑	☑	☐	☑	☑	☐	☑	☐	☐
DDE client ability	☑	☐	☒	☒	☐	☐	☑	☒	☒	☑	☑	☑	☑	☑	☑	☑	☑	☐
DDE server ability	☑	☐	☒	☑	☐	☐	☑	☑	☒	☒	☒	☑	☑	☑	☑	☑	☑	☐
OLE	☑	☐	☑	☐	☐	☐	☒	☐	☐	☐	☑	☑	☐	☑	☒	☑	☑	☐
MDI	☑	☐	☑	☐	☐	☐	☒	☐	☐	☐	☑	☑	☐	☑	☒	☑	☑	☐
Drag and drop	☑	☐	☑	☐	☐	☐	☒	☐	☐	☐	☑	☑	☐	☑	☑	☑	☑	☐
Quick report generator	☑	☐	☑	☐	☐	☐	☒	☐	☐	☐	☑	☑	☐	☑	☑	☑	☑	☐
Two-pass report writer	☑	☐	☑	☐	☐	☐	☑	☐	☐	☐	☑	☑	☐	☒	☒	☑	☑	☐
Multitable forms & pop-up lists	☑	☐	☑	☐	☐	☐	☒	☐	☐	☐	☑	☑	☐	☑	☒	☑	☑	☐
Query by example (QBE)	☑	☐	☒	☐	☐	☐	☒	☐	☐	☐	☑	☑	☐	☒	☑	☒	☒	☐
Query by form (QBF)	☑	☐	☑	☐	☐	☐	☒	☐	☐	☐	☑	☒	☐	☒	☑	☑	☑	☐
Interactive SQL	☑	☒	☑	☐	☑	☒	☑	☐	☐	☐	☑	☑	☐	☑	☑	☑	☑	☑
Charting & graphing	☑	☐	☒	☐	☐	☐	☑	☐	☐	☐	☑	☒	☐	☒	☒	☒	☒	☐
Data Sources																		
Accesses multiple sources in one query	☑	☐	☐	☒	☐	☐	☒	☑	☑	☒	☑	☐	☑	☑	☐	☐	☐	☐
Integrates & displays data from multiple sources	☑	☐	☐	☑	☐	☐	☑	☑	☑	☑	☑	☐	☑	☑	☐	☑	☐	☐
Joins tables from multiple sources	☑	☐	☐	☒	☐	☐	☒	☑	☑	☒	☑	☐	☑	☒	☐	☒	☐	☐
Schedules queries for automatic updates	☑	☐	☐	☑	☐	☐	☑	☒	☑	☑	☑	☐	☑	☑	☐	☒	☐	☐
Updates server database	☑	☐	☐	☑	☐	☐	☒	☑	☑	☑	☑	☐	☑	☑	☐	☑	☐	☐
Updates multiple databases at once	☑	☐	☐	☑	☐	☐	☒	☒	☑	☑	☑	☐	☑	☑	☐	☒	☐	☐
Programming																		
Constructs queries from menus	☑	☐	☐	☑	☐	☐	☑	☑	☒	☑	☑	☐	☒	☑	☐	☑	☐	☐
Constructs nested queries from menus	☒	☐	☐	☑	☐	☐	☑	☒	☒	☑	☒	☐	☒	☑	☐	☒	☐	☐
User can view constructed SQL code	☑	☐	☐	☑	☐	☐	☑	☒	☑	☒	☑	☐	☒	☑	☐	☑	☐	☐
User can edit constructed SQL code	☑	☐	☐	☑	☐	☐	☑	☒	☑	☒	☑	☐	☑	☑	☐	☑	☐	☐
SQL syntax on-line help	☑	☐	☐	☑	☐	☐	☑	☒	☒	☒	☒	☐	☒	☒	☐	☑	☐	☐



Native programming/scripting language	☑	☐	☐	☑	☐	☐	☒	☒	☑	☑	☑	☐	☑	☑	☐	☒	☐	☐
Allows testing of application with existing package	☑	☐	☐	☑	☐	☐	☒	☒	☑	☑	☑	☐	☒	☑	☐	☒	☐	☐
Traces through program	☑	☐	☐	☒	☐	☐	☒	☒	☒	☑	☒	☐	☒	☑	☐	☒	☐	☐
SQL low level abilities	☑	☐	☒	☐	☐	☐	☒	☐	☐	☐	☑	☑	☐	☑	☒	☒	☑	☐
On-the-fly syntax checking	☑	☐	☒	☒	☐	☐	☒	☒	☑	☒	☑	☑	☑	☑	☒	☑	☑	☐
Embedded SQL	☑	☐	☒	☐	☐	☐	☒	☐	☐	☐	☑	☑	☐	☑	☒	☒	☑	☐
Extensible language via DLL	☑	☐	☒	☐	☐	☐	☒	☐	☐	☐	☑	☑	☐	☑	☑	☒	☑	☐
Object-orientated inheritance	☒	☐	☒	☐	☐	☐	☒	☐	☐	☐	☑	☑	☐	☑	☒	☒	☑	☐
Debugger	☑	☐	☒	☐	☐	☐	☒	☐	☐	☐	☑	☑	☐	☑	☒	☑	☑	☐
Code check-in/check-out	☒	☐	☒	☐	☐	☐	☒	☐	☐	☐	☑	☒	☐	☑	☒	☒	☑	☐
Code/object reusability	☑	☐	☒	☐	☐	☐	☒	☐	☐	☐	☒	☑	☐	☑	☒	☒	☑	☐
Application Distribution																		
Distributable	☑	☐	☒	☒	☐	☐	☒	☑	☒	☑	☑	☒	☑	☑	☒	☒	☑	☐
Royalty free distribution	☑	☐	☒	☒	☐	☐	☒	☑	☒	☒	☑	☒	☒	☒	☒	☒	☑	☐
Relational Capabilities																		
Number of tables that can be linked	+	#	#	☐	#	☒	☐	☐	☐	☐	☐	+	☐	☐	☐	☐	☐	+
Linked tables in forms & reports	☑	☒	☑	☐	☑	☒	☐	☐	☐	☐	☐	☑	☐	☐	☐	☐	☐	☑
Database-level referential integrity	☑	☒	☑	☐	☒	☒	☐	☐	☐	☐	☐	☑	☐	☐	☐	☐	☐	☑
Form level referential integrity	☑	☒	☑	☐	☑	☒	☐	☐	☐	☐	☐	☑	☐	☐	☐	☐	☐	☑
Point-and-click data objects																		
Lookups and pick lists	☒	☑	☑	☐	☑	☑	☐	☐	☐	☐	☐	☑	☐	☐	☐	☐	☐	☑
Incrementing fields	☑	☑	☑	☐	☑	☑	☐	☐	☐	☐	☐	☒	☐	☐	☐	☐	☐	☑
Data presentation																		
Style sheets/template preferences	☑	☒	☑	☐	☑	☒	☐	☐	☐	☐	☐	☑	☐	☐	☐	☐	☐	☑
Avery label layouts included	☑	☑	☑	☐	☑	☑	☐	☐	☐	☐	☐	☒	☐	☐	☐	☐	☐	☒
Cross tabs	☑	☑	☒	☐	☒	☒	☐	☐	☐	☐	☐	☑	☐	☐	☐	☐	☐	☒
Graphs	☑	☑	☒	☐	☒	☒	☐	☐	☐	☐	☐	☑	☐	☐	☐	☐	☐	☒
Automatic linking of data files	☑	☒	☒	☐	☑	☒	☐	☐	☐	☐	☐	☑	☐	☐	☐	☐	☐	☑
Point-and-click multitable layout design	☑	☒	☑	☐	☑	☒	☐	☐	☐	☐	☐	☑	☐	☐	☐	☐	☐	☑
Experts, wizards, cue-cards or other guides	☑	☒	☒	☐	☑	☒	☐	☐	☐	☐	☐	☑	☐	☐	☐	☐	☐	☒
Attach files to documents	☑	☑	☑	☐	☑	☒	☐	☐	☐	☐	☐	☑	☐	☐	☐	☐	☐	☑
Queries																		
QBE, QBF	☑	☒	☑	☐	☑	☑	☐	☐	☐	☐	☐	☑	☐	☐	☐	☐	☐	☑
Experts, wizards, cue-cards or other guides	☑	☒	☒	☐	☑	☒	☐	☐	☐	☐	☐	☑	☐	☐	☐	☐	☐	☒
Embeded SQL requests	☑	☒	☑	☐	☒	☒	☐	☐	☐	☐	☐	☑	☐	☐	☐	☐	☐	☑
Edit query results directly & update data	☑	☑	☑	☐	☑	☑	☐	☐	☐	☐	☐	☒	☐	☐	☐	☐	☐	☒
Create from form or report	☑	☒	☒	☐	☑	☑	☐	☐	☐	☐	☐	☑	☐	☐	☐	☐	☐	☒
Security																		
Group or application level	☑	☒	☒	☐	☑	☑	☐	☐	☐	☐	☐	☑	☐	☐	☐	☐	☐	☒
Table level	☑	☒	☑	☐	☑	☑	☐	☐	☐	☐	☐	☑	☐	☐	☐	☐	☐	☑
Field level	☑	☒	☑	☐	☑	☑	☐	☐	☐	☐	☐	☑	☐	☐	☐	☐	☐	☑

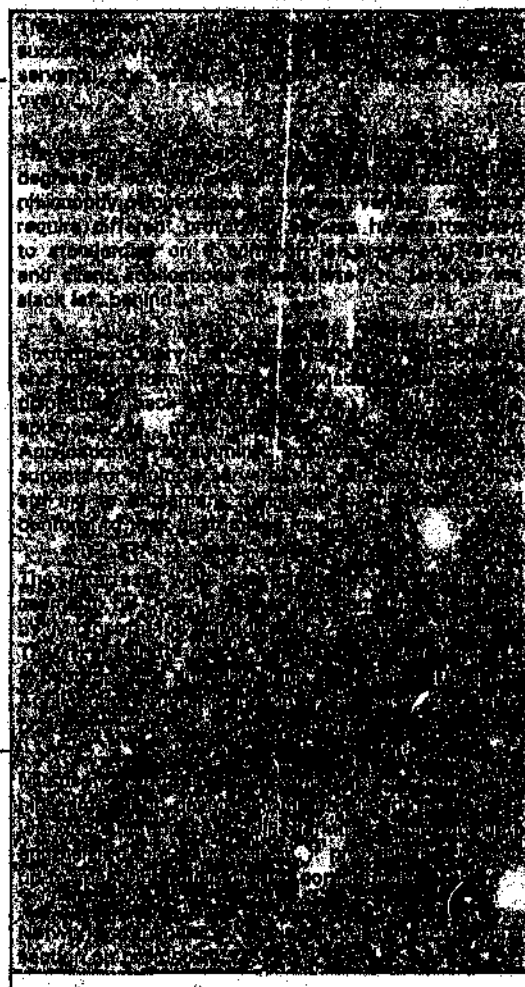


Text, numeric, date, binary	☑	☑	☑	☐	☑	☑	☐	☐	☐	☐	☐	☑	☐	☐	☐	☐	☐	☑
Time	☑	☒	☑	☐	☑	☑	☐	☐	☐	☐	☐	☒	☐	☐	☐	☐	☐	☑
Memos	☑	☑	☑	☐	☑	☑	☐	☐	☐	☐	☐	☑	☐	☐	☐	☐	☐	☒
Currency	☑	☑	☑	☐	☑	☑	☐	☐	☐	☐	☐	☑	☐	☐	☐	☐	☐	☑
Counter	☑	☒	☒	☐	☑	☒	☐	☐	☐	☐	☐	☒	☐	☐	☐	☐	☐	☑
Logical	☑	☑	☑	☐	☑	☑	☐	☐	☐	☐	☐	☒	☐	☐	☐	☐	☐	☑
Find on record key	☒	☒	☒	☐	☒	☑	☐	☐	☐	☐	☐	☒	☐	☐	☐	☐	☐	☒
Find on name	☑	5	☒	☐	☒	☑	☐	☐	☐	☐	☐	☒	☐	☐	☐	☐	☐	☑
Simple Query	☒	2	☒	☐	☒	5	☐	☐	☐	☐	☐	☒	☐	☐	☐	☐	☐	3
Joined Query	2	#	#	☐	#	+	☐	☐	☐	☐	☐	3	☐	☐	☐	☐	☐	#
Customer report	#	#	#	☐	#	+	☐	☐	☐	☐	☐	#	☐	☐	☐	☐	☐	#



TRANSPARENCY

SERVERS



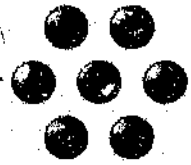
OVERVIEW SQL is a terse, specialised language. With just a few commands, it lets one accomplish what would take dozens of lines of code in Xbase (the programming language designed to address the first PC based--and industry accepted--database engine, dBase). This alone makes SQL a good choice for a networked environment, since it means less network traffic. But SQL is also especially well suited for extracting data from a relational database, since that is what it was designed to do; it was developed in conjunction with the relational database paradigm. One key feature of SQL is that one asks the database for what one wants; how to get it is left to the database software. This insulates users from changes in the database design which may occur over time.

A typical SQL statement, easily understandable, might run as follows:

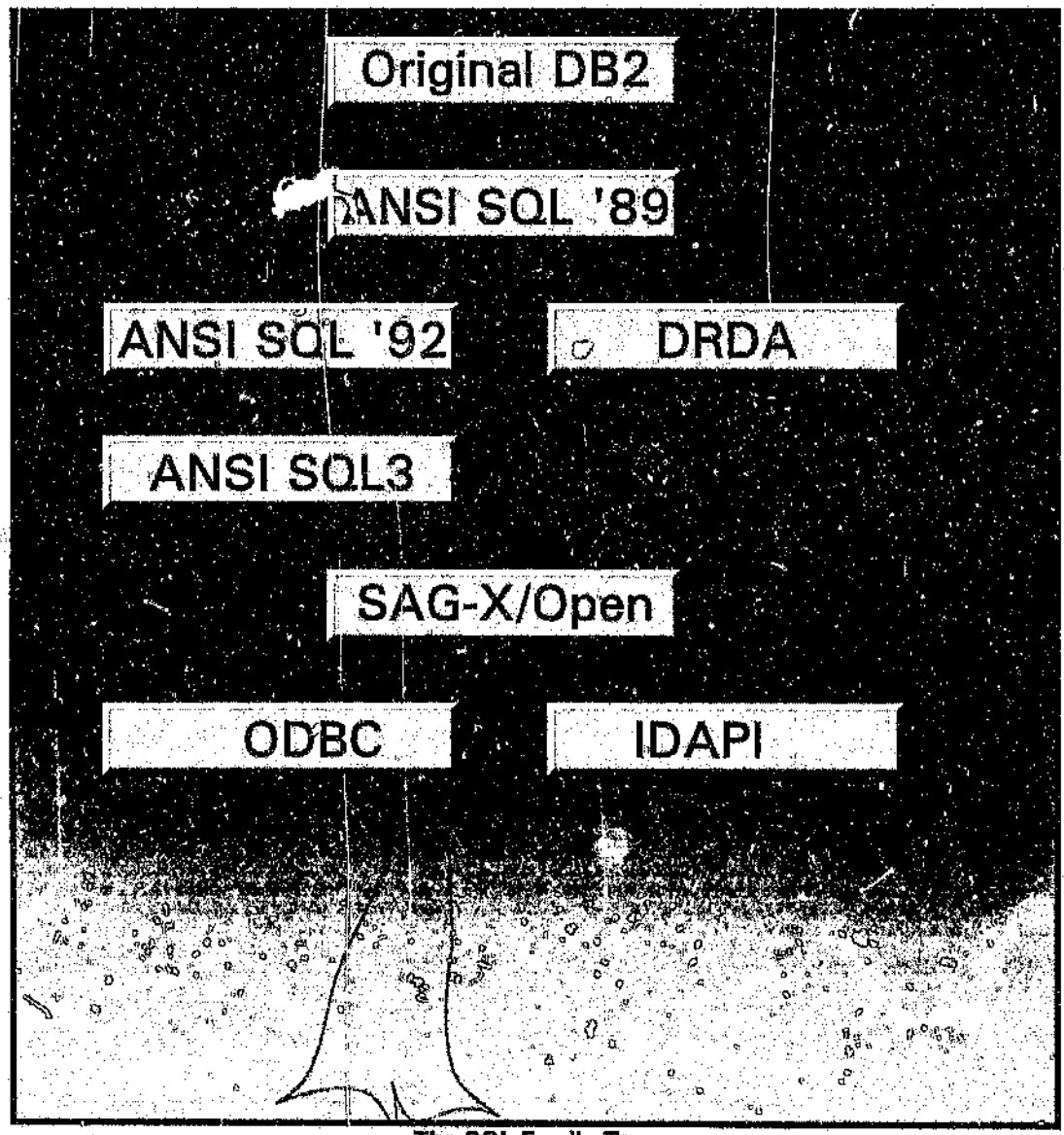
```
SELECT DISTINCTROW table1.field1,
table1.field2, table2.field1
FROM table1, table2 INNER JOIN ON
table1.field1, table2.field2
WHERE table1.field1 BETWEEN value1 AND
value2
GROUP BY table1.field1/table2.field2,
table1.field3
```

STRUCTURED QUERY LANGUAGE (SQL) SQL (Structured Query Language--Pronounced Sequel) has been the standard data description and access language for relational databases for almost a decade, making it a linchpin technology for client/server computing. The only problem with the SQL standard is that there are too many of them. No fewer than eight efforts are under way to create a standard based on SQL. ANSI alone is responsible for three standards, either published or in progress. Other efforts involve industry consortia, such as SAG (the SQL Access Group) and X/Open, and companies such as IBM, Microsoft, and Borland. Considering that every database speaks its own SQL dialect and has its own set of unique extensions to the language, it's not surprising that client/server installations were, until recently, the exception rather than the rule. The lack of one widely accepted standard drives up the cost of databases and related tools and makes maintaining a client/server environment complex and difficult. There is little difference between too many standards and no standards at all.[78]

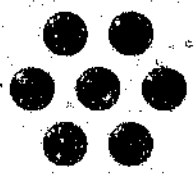
THE STANDARDS PLETHORA The American National Standards Institute (ANSI) took the first crack at creating an SQL standard in 1989 (SQL '89) and followed up with a revision in October of last year (SQL '92). But because ANSI standards are vague about the details of their implementation, products that claim to be SQL compliant often fail to work with one another. The proliferation of different SQL implementations by vendors led to the formation of The SQL Access Group (SAG). A consortium of hundreds of companies in the database field, SAG



was formed to generate SQL standards based on a subset of ANSI SQL '89 and to deal with issues of connectivity and programming interfaces for SQL clients and servers. In conjunction with the industry group X/open, SAG is constructing a standard based on the current state of the art in client/server implementations. Many major companies are not waiting for the efforts of standards-setting bodies to bear fruit but are launching their own attempts to create de facto standards. Last year, SAG member Microsoft introduced ODBC (Open Database Connectivity), an SQL database API (Application Programming Interface) standard based in part on a draft of the SAG standard. Borland, also a member of SAG, is readying its own SQL API standard, called IDAPI (Integrated Database Application Programming Interface), which is based on the SAG version. Both companies are lining up support for their standards. In addition, ANSI is working on its next-generation SQL standard, called SQL3. Not expected until the latter part of the decade, SQL3 will add structure, branching, flow control, error handling, and object-oriented capabilities to an already groaning SQL standard. Finally, IBM is developing its own standard for database access, DRDA (Distributed Relational Database Access), which includes yet another version of SQL.[77]



The SQL Family Tree



THE IMPORTANCE OF STANDARDS

The reasons for the proliferations of SQL standards are many, one is that standards are designed to serve only as guides, this leaves a lot of room for interpretation as a vendor takes a language specification and implements it in software.

This assumes a vendor is trying to conform to a standard. More often than not, vendors are trying to do more. The problem is that each vendor has probably made excellent, but different, decisions about how to extend the SQL standard.

Writing standards-compliant code means that developers are using 3-5 year old technology. If one wants the benefits of the latest technology, one can't conform to standards. For example, Sybase has had flow-control statements, stored procedures, and triggers in Sybase's SQL dialect since 1987. SQL '92 doesn't have those features which are the type of thing found in the SQL3 standard. In the absence of an acceptable standard, database vendors have invented their own.

ENHANCING THE POWER OF SQL

The current ANSI standard for SQL specifies two levels of compatibility: ANSI SQL Level 1 and ANSI SQL Level 2, which is a superset of Level 1. Additionally, the standard specifies a separate integrity enhancement addendum. The most robust ANSI compatibility claim a vendor can make for its product is with ANSI SQL Level 2 with the Integrity Enhancement addendum.[78]

Vendors are starting to provide enhanced versions of standard SQL that support extensions such as stored procedures, triggers, and execution control through branching and looping. Transact SQL (the dialect provided with Microsoft SQL Server and Sybase SQL Server) and PL/SQL (provided with Oracle Server) are examples of enhanced SQL dialects.

Stored procedures are precompiled SQL statements that are stored on the SQL database server. A client executes a stored procedure by issuing an EXECUTE PROCEDURE_NAME statement. This way, two words cross the network rather than the hundreds that could be in the procedure. And since the procedure is already optimised and compiled, the server does not need to spend additional time on these tasks. Frequently executed queries are prime candidates for stored procedures.

Triggers are similar to stored procedures and execute in response to a database event. When a trigger is associated with an SQL statement, a set of commands is executed every time the SQL statement is executed.

As SQL grows larger and more complex, the standards' bodies are taking longer to ratify them. In fact, the glacial pace of ANSI is largely responsible for the involvement of industry consortia like SAG and X/Open in the SQL standards business.

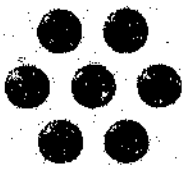
CLIENTS

PROBLEM

In most firms, important data is scattered among multiple computers, from PCs to mainframes, and among more than one application or DBMS.

With the growing popularity of SQL databases on virtually every platform imaginable, the use of proprietary APIs (application programming interfaces) for data access and manipulation presents more and more of a problem. A front-end program developed for use with one vendor's back-end engine cannot be used with another's, and application developers must spend costly time learning many variants of the same thing.

Users, however, are more interested in the ends (acquiring the data they need) than the means (navigating network directories to locate the data). That's because end users need



true connectivity--transparent access to data so that one can find what one needs as quickly and easily as if it were stored on the desktop computer.

CURRENT SITUATION Even if the market's competitive pressures forever deter the universal adoption of a single SQL standard, database and tool vendors are looking at other ways of making life easier in the client/server environment.
"I do not see the SQL standards as too important," says Ron Wolf, product marketing director for SQL database vendor Gutpa. "The next step down-the programming interface-is where significant things are happening. Those standards are quite a bit more useful."

The programming interface-ref ed to as the API or the CLI (call-level interface) lets programs communicate with a database using function calls instead of SQL statements. Where SQL standards address data access and manipulation, API standards deal largely with connectivity issues: connecting the client to the database and exchanging SQL messages and data. A common API for SQL would make it less expensive for developers to support several different databases, making their tools more affordable. However, a common API may prove elusive, given market realities.

While individual vendors have always provided these kinds of connectivity solutions for their own products, these 'solutions' have typically been proprietary and not 'open', so they could not be used with products from other vendors. One could not, for example, seamlessly build a report based on data stored in databases from two different vendors.

To provide that seamless interoperability, one program must support another's APIs (application programming interfaces), routines that can be 'called' to perform some task in a standard, predictable fashion.

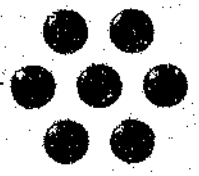
THE COMPONENTS OF AN API An API is nothing more than a syntax (a standard set of functions) that is used by developers for a particular task. An API used to access SQL engines is a set of functions that standardise communication with SQL back ends. With the exception of precompiling, the API has little or nothing to do with a particular SQL dialect, but rather the manner in which SQL is passed to the engine and results are retrieved.

For example, one engine might have a three-parameter function called dbEXEC for sending SQL commands to the engine. Another engine might have a function with a similar purpose called SQLRequest that takes two parameters. In the case of the function with the third parameter, results (perhaps only partial) will be retrieved as part of the call to send the SQL statements. With the other function, though, a separate call is required to retrieve results. These seemingly minor differences can cause a great deal of frustration for a developer attempting to build a back end-independent application.[80]

The simplest road to transparent data access is to connect to the same vendor's SQL engine on different platforms. This is relatively simple because the API remains the same and only the platform changes. Vendors that provide versions of their products for multiple platforms (and most do) generally provide communications software for accessing data stored on other platforms. A company that uses one vendor's SQL engine on all of its computers will have an easier time integrating its databases than will a company which uses products from multiple back-end vendors.

The reality, however, is that most companies use software from more than one vendor. A company's database software is usually purchased piecemeal over time rather than with an overarching grand plan, and most companies do not have the luxury of throwing out their existing software to adopt a new, standardised system on all platforms. In general, transitions to new database software are made incrementally, if they are made at all.

SERVER APIS--GATEWAYS Gateways, software that provides transparent access to the database engines of many vendors, are sometimes developed by database vendors themselves in an effort to bring more developers into their fold. While such libraries are



not portable, they do guarantee access to all the features the SQL engine supports. Gupta Technologies builds gateway features into its engines that enable developers to write to Gupta's SQLBase API but specify that these calls should be 'passed through' to another SQL engine.

Most gateways, though, are provided by third-party vendors. Using a DB2/Gateway, it is possible to move an application to DB2 or another format without even recompiling the code it operates by intercepting calls to dbLIB and translating them to calls to the host system.

Like users, vendors like transparency. The fact is, no matter what DBMS being accessed, there are certain common tasks--such as opening database files and retrieving or updating data--that every client program needs to do. Standardising on an existing API is a good approach because then at least one engine's features will be fully accessible. But there are cross-format advantages to using a non dedicated API.

SOLUTION APIs, just like their underlying programs, are evolving--and Windows perhaps is the best example of this. Its API is so complex that Microsoft has divided it and named each part: there's ODBC (Open Database Connectivity) and MAPI (Microsoft's Mail API), for example. Microsoft has even coined a new term, WOSA (Windows Open Systems Architecture), which refers to all of the individual Windows APIs.[81]

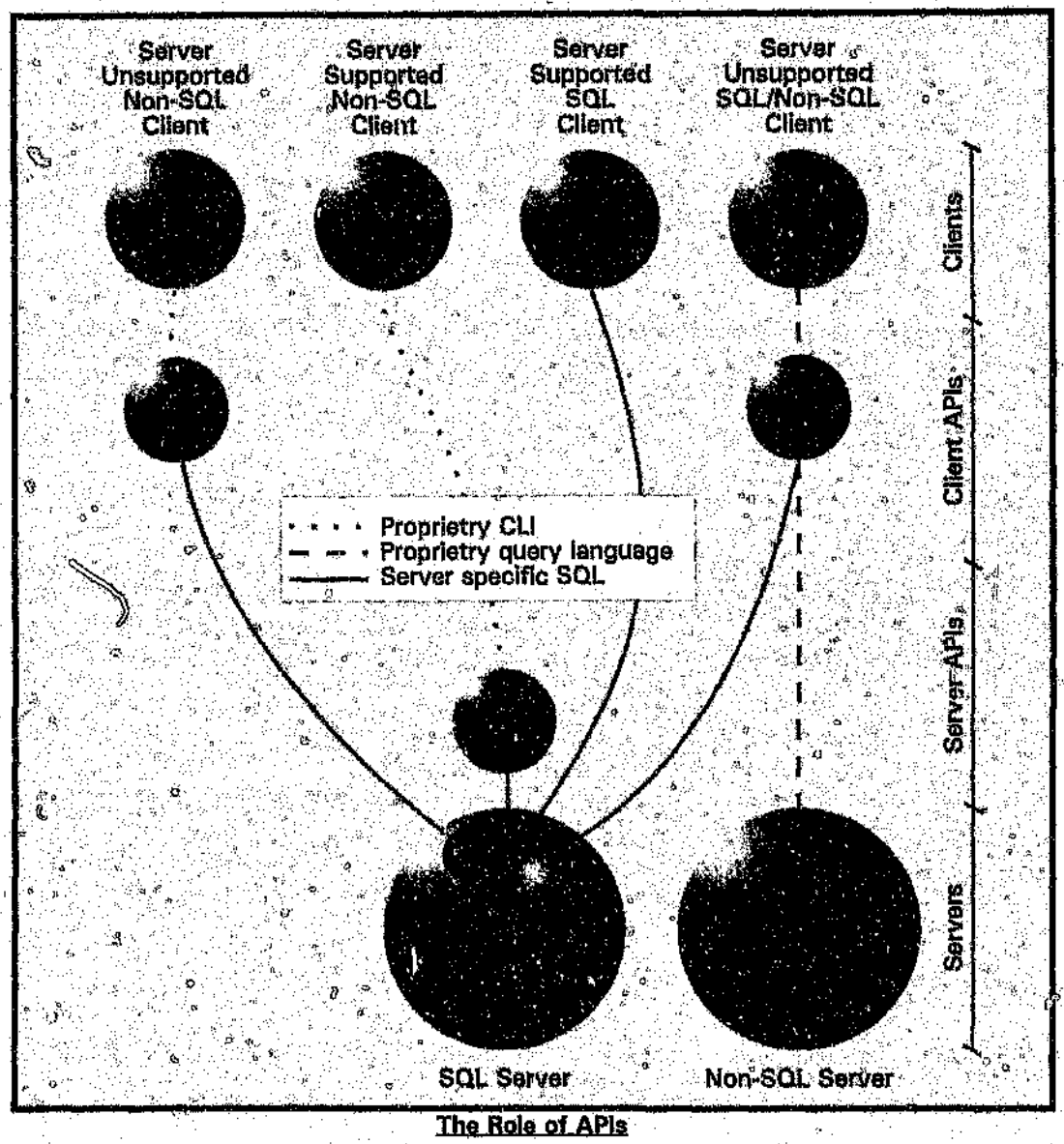
The solution? A common API for database access. And this is where Microsoft and Borland's open database APIs enter the picture. Based on a common subset of the ANSI standard, these have been criticised for taking the lowest-common-denominator approach, which doesn't exploit the features of a database, other than perhaps the developer's own.

Rather than evolving from an actual product, these committee-designed standards originated from paper. It will be the responsibility of each vendor to create compliant software. Microsoft's new standard is called ODBC; the Borland version is called ODAPI (Open Database Application Programming Interface). ODBC has been completed whilst the more ambitious ODAPI standard is further off but intended to support non-relational data access and object-oriented database technology. Both standards provide the following:[82]

- A set of standard function calls (an API) to allow applications to connect to a database management system, execute SQL statements, and retrieve results. It would be the responsibility of each engine vendor to release a driver that complies with the standard. The drivers will take the form of 'wrapper' libraries that translate the vendor-specific calls into the standard API. This will preserve the utility of existing applications that call the old API.
- A specification of SQL syntax and supported data types. One of the major requirements for successful cross-engine interoperability is the standardisation of SQL language components. This will probably be the greatest stumbling block to transparent data access because of the need to preserve existing functionality.
- A standardised set of error codes and standardised error-handling procedures and behavior.

If the computer industry as a whole accepts one of the standards, the number of applications supporting access to SQL databases will mushroom. Providing access to so many different standards is a developer's nightmare; a single API would greatly ease the task. In mid '93 Oracle announced its support for ODBC, providing a good indication that this may well be the race winner.

The differences between client APIs and server APIs is illustrated below. The common API allows for the replacement of the server API, as well as gateways and third party connectivity packages where the client supports this commonality.

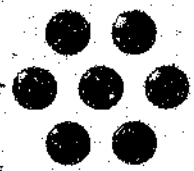


The Role of APIs

ODBC The core component of ODBC is a generic 'driver manager' that rides herd over one or more drivers for the specific data sources to which ODBC gives access. A Windows program called the ODBC Administrator registers and configures data sources.

SAG Specifications: ODBC follows closely the recommendations of the SAG (SQL Access Group). That consortium has specified both a standard SQL grammar and an API (or call level interface) that clients use to connect to and interact with servers. ODBC drivers can and do differ in their degree of support for the SAG specifications. In terms of SQL grammar, for example, the ODBC specification defines three conformity levels: minimum, core, and extended. With respect to the API, the specification defines core, level 1, and level 2 conformity. In both cases, the term core denotes SAG compliance.

While the drivers fall tall short of the SAG core in terms of SQL grammar, they exceed it with respect to the API--both conform to level 1. The core API provides only basic connection, query, and commit/rollback semantics. Level 1 adds data dictionary APIs, options for controlling connections and statements, and driver specific connection dialog boxes.[83]



Driver Classes: ODBC drivers belong to two general classes. Single-tier drivers (e.g., the dBase driver) process API calls and implement SQL directly. Multi-tier drivers (e.g., the SQL Server driver) process API calls but pass SQL statements to a server. To programs and users, though, the data sources these drivers present look and act just the same.

The dBase driver is, literally, a dBase engine driven by an SQL syntax. An instance of the driver points to a local or network directory that contains one or more dBase files. These files appear as SQL tables that can be joined and updated. That gives ODBC programs more control over dBase data than some full-blown Xbase products offer.

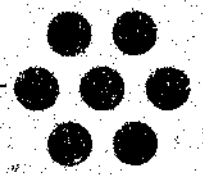
While ODBC is being billed primarily as a means of connecting to enterprise data, its flat-file support will also greatly benefit applications that create and manage private data. It's a trivial matter to create an Xbase table, insert values into it, and run queries against it. A communications program that might otherwise implement its own proprietary database of modems or phone numbers can now use what will be, in effect, a Windows API for persistent storage. If the program later needs to migrate that storage to a different flat-file format or to a server, it's no problem. Multi-tier drivers connect ODBC clients to servers directly or indirectly through gateways. The driver might involve itself in low-level client-to-server communication, but it's not required to do so. The SQL Server driver, for example, leans on the same named-pipes driver and under lying NetBEUI or IPX transports that other Windows-to-SQL Server agents use. The proprietary SQL Server API can be discarded since the ODBC API replaces it. That's irrelevant if only using SQL Server, but a huge advantage if there are plans to migrate to Oracle, NetWare SQL, or some other server.

How does ODBC handle features that are specific to individual servers? First, there's a notation for standard SQL extensions such as date/time literals and outer joins. Most servers offer these features using a proprietary syntax; ODBC's 'escape calluses' enable programs to use them portably. Second, ODBC will pass statements that don't conform to its notion of SQL grammar straight through to the server. Use of such grammar is then no longer portable, but applications can exploit vendor-specific extensions through ODBC.

Key ODBC Objects: From a programmer's perspective, the key ODBC handle-based objectives are environments, connections, and statements. There is one environment per application instance. The environment maintains state for one or more connections, so a program can talk to dBase on connection handle 1 while it talks to SQL Server on connection handle 2. (Connections can't be mixed, though; there's no way to join a dBase table to an SQL Server table.) With a level 1 driver, one can control options such as the auto-commit mode and the transaction isolation level on a per-connection basis. Given a valid connection, one can allocate a statement handle that is then used to submit SQL text and retrieve results. Options one can control at the statement level include asynchronous processing, query time-out, and size (in rows) of the maximum result set.

As connections and submit statements are negotiated, the driver manager tracks state transitions. It will report an appropriate error if, for example a program tries to call SQLFetch before executing a query. (Because the driver manager handles this chore, specific ODBC drivers don't have to--a benefit of the layered driver architecture). One can execute a one-off query directly, or prepare a query for deferred (possibly repetitive, possibly parameterised) use. One can also choose to bind application storage to result columns in advance of a query or to transfer data after the fact. With a level 2 driver, one can use a scrollable cursor and can control its sensitivity to change in the underlying tables. Most flexible is a keyset-driven cursor. In this case, the driver stores keys for an entire result set and uses the cached keys to detect deleted or changed rows. A mixed cursor extends this treatment to huge result sets by maintaining a virtual keyset.

COMPLICATIONS As expected upon ODBC's inception, other major vendors, notably Borland with IDAPI (Integrated Database API) and IBM with DRDA (Distributed Relational Database Architecture), proposed alternate APIs. Then software publishers started choosing sides.



Only time will tell which database application supports which API. While a handful of major APIs is better than having different ones for every DBMS, it takes time to write code and test the driver that links a product to all the supported DBMSs through a given API. ODBC drivers have appeared in products such as Access, Visual Basic 2.0, and DataEase Express for Windows 1.1.

One of the emerging problems with ODBC is that it's confusing: There are just too many levels at which a program must be complaint with ODBC. Not only are there two separate areas of compliance--Basic ODBC SQL Grammar and the ODBC API--but each has its own additional levels as well. The other problem with ODBC is more straightforward: The initial drivers just don't deliver acceptable performance.

While Borland's IDAPI specification has attracted key vendors like IBM and Novell, neither the specification nor any IDAPI drivers were generally available by mid '93. Borland claims IDAPI will be more 'robust' than ODBC and that it won't be limited to Windows front ends. Microsoft counters that ODBC is being extended to include Macintosh and other platforms. Borland also claims that IDAPI will offer a better interface to non-SQL data than ODBC does.[64]

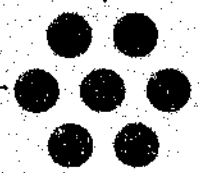
In the long run, perhaps all the APIs will coalesce and we'll benefit from truly open systems with perfect, transparent interoperability. But, in the short term, companies are being asked to vote for an API with purchasing Rands.

It's too early to tell which API will prevail, or how the two will coexist. Market momentum seems to be on Microsoft's side. ODBC got off to an earlier start than IDAPI, and more vendors--including Apple, Lotus Development, Gupta, and Pioneer--are committed to creating ODBC drivers. But early ODBC drivers are beginning to make people wonder if ODBC can ever fully deliver.

Microsoft's products--Access, Visual Basic for Windows, FoxPro for Windows, and SQL Server--either already have or will soon have ODBC drivers, as does Oracle--arguably the industry server leader. Borland's products--Paradox for Windows, dBase for Windows, InterBase, and Quattro Pro for Windows--will all inter-operate with IDAPI drivers. That's about all that is known know for sure, but the success and capabilities of all Microsoft products over all others, and its impressive marketing muscle, provide a certain degree of confidence in ODBC--it'll survive even if it's not optimal.

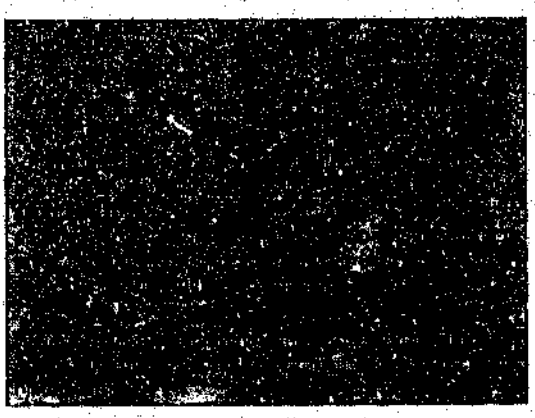
NETWORKS

Network transparency is facilitated through protocols as were discussed extensively above.



IMPLEMENTATION

METHODOLOGY



The following section is not an overview of how the implementation of IT within Vandenberghs progressed, nor how it should have progressed, but rather what the industry has to say about how IT should be implemented in a typical environment. How this method was adapted, what guidelines were followed, and what political stances were adopted is covered in the forthcoming section on the implementation--the current section simply being a literature survey.

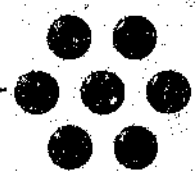
In hindsight, the author does not necessarily support the following guidelines, but rather feels that in politically volatile environments like international concerns, a 'play-it-by-ear' or more intuitive approach is necessitated, and that the flexibility must exist to be able to adapt ones strategies as required.

However, some underlying principles do hold true, and an attempt has been made to concentrate on these below, whilst also highlighting those that would apply in a large, somewhat bureaucratic, concern like Vandenberghs.

NOLAN'S IT IMPLEMENTATION LIFE CYCLE

One of the most widely known theoretical constructs in the IS field is Nolan's stage theory. In 1974 Richard L.Nolan and Cyrus F.Gibson, both Harvard professors, described how a firm's computing activity evolves through four stages -- initiation, expansion, formalisation, and maturity. In 1979 Nolan elaborated on the model, expanding it to six stages.[86] The table shows these stages in terms of the applications portfolio, the IT department organisational structure, planning and control, and user awareness. A movement from the bottom left hand corner, to the top right hand corner represents how the firm spends an increasing amount of money for the computer installation as use evolves through the following phases

Growth Process	Initiation	Contagion	Control	Integration	Data Administration	Maturity
Applications Portfolio	Functional cost reduction applications	Proliferation	Upgrade documentation and restructuring of existing applications	Retrofitting existing applications using database technology	Organisation & integration of applications	Application integration 'mirroring' information flows
Organisation	Specialisation for technological learning	User-oriented programmers	Middle management	Establish computer utility and user account teams	Data administration	Data resource management
Planning & Control	Lax	More lax	Formalised planning and control	Tailored planning and control systems	Shared data and information systems	Data resource strategic planning
User Awareness	Hands off	Superficially enthusiastic	Arbitrarily held accountable	Accountability learning	Effectively accountable	Acceptance of joint user and data processing accountability



INITIATION The computer is usually installed for the purpose of reducing costs. It is frequently located in the accounting department since data processing applications dominate. In some cases a separate computing unit is established.

CONTAGION Word quickly spreads throughout the organisation of the advantages of using the computer. New applications are added without any overall plan, and the size of the computer staff grows to meet the demand.

CONTROL Management becomes alarmed at increasing computer costs and institutes controls. In addition, management decides that some of the investment should go into decision support.

INTEGRATION Systems that were implemented separately during the second stage are integrated so that data flows from one to another.

DATA ADMINISTRATION Management becomes aware of the importance of the database to both data processing and decision support. DBMS software is implemented to manage the data resource.

MATURITY At this point, all of the major IS components are in place. An executive committee exercises overall control, information services is a major functional area of the firm, user areas exhibit both a computer and an information literacy, terminals and micros are installed throughout the company, and end-user computing is a reality.

In addition to describing the stages in this manner, Nolan also provided management with some benchmarks so that it could determine the stage that the firm has reached.

THE MODIFIED LIFE CYCLE

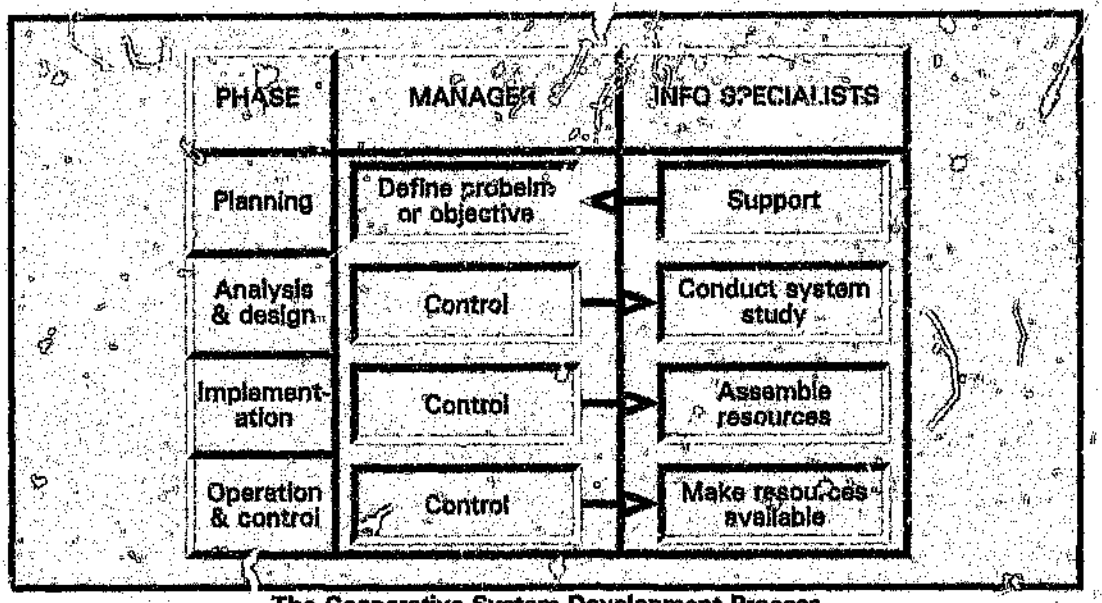
While stage theory is intuitively appealing, not everyone bought the concept. In 1983, D H Drury, a professor at McGill University, subjected Nolan's model to a field study of 144 companies.[96] A mathematical analysis of the data failed to support the collective use of the benchmarks as useful indicators. However, Drury recognised that "there is evidence that the benchmarks are of interest individually in spite of their collective difficulties." Other criticism came from professors John L Kinb and Kenneth L Kraemer of the University of California.[97] They addressed Nolan's basic assumptions and concluded that the theory failed to explain why computer installations evolve as they do. However, they did credit the theory for making two theoretical contributions:

- It recognises that growth of computing is influenced by forces both inside and outside the organisation.
- It recognises that management alternately follows policies of tight and loose control in an effort to maintain equilibrium. When allowed to grow, the computing resource expands rapidly and is brought back into line by tight control.

Nolan's model most likely is not a precise instrument that a firm's managers can use to gauge the progress of their particular IS. However, it recognises two simple facts, that computer use becomes more sophisticated over time and that the firm must attain a certain level of sophistication in the areas of control, integration, and data administration before it can consider its computer use as mature. With this understanding of the long-term evolution, the adapted life cycle model emerges.

We can define the system life cycle as the evolutionary process that is followed in implementing a computer-based information system or subsystem. The letters SLC are often used, as are the letters SDLC, for system development life cycle. There is general agreement among computer systems experts that the system life cycle is an accurate description of the implementation tasks that must be accomplished. The only disagreement comes in the number of steps and their names. Each authority tends to

describe the process in a slightly different manner. A close inspection reveals, however, that all of the descriptions follow the same general pattern. The pattern is based on the systems approach—understanding that it has to be done, considering alternate solutions, selecting the best, implementing it, and following up.[88] The more favoured interpretation of the system life cycle is summarised in the following figure.



The Cooperative System Development Process

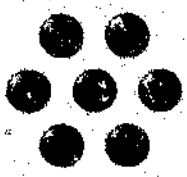
It is the responsibility of the manager in the planning phase to define the problem to be solved or the objectives to be achieved, and information specialists provide support. During the remaining three phases, the manager provides control over the processes that are performed by the information specialists. Analysis and design consists of a system study that is conducted by the systems analyst. Implementation involves all of the information specialists who assemble the necessary resources. Finally, the information specialists, primarily the operators, make the resources available to the users. This is a traditional interpretation of the system life cycle, which is characteristic of a mainframe-based installation where most of the work is done by the information specialists. The trend to end-user computing blurs, and in some cases completely erases, the dwindling line between the manager and the specialists. The manager performs all of the activity when end-user computing is implemented to its fullest. The sequential life-cycle process is also presently undergoing change as the result of other influences. First, prototyping dictates that certain of the steps be repeated in an interactive fashion for each prototype. Second, CASE tools are shifting some of the workload from the implementation phase to the planning and the analysis and design phases. These trends are covered later.

RESPONSIBILITIES

Some person or persons within the organisation must have overall responsibility for the IS project. This should not be the CIO (Chief Information Officer) who is responsible for the firm's computing resources, but rather one or more executives who represent the potential users of the new system.[89] When the new system is broad in scope, such as a company-wide commitment to MIS, OA (Office Automation), or EIS (Executive Information System), the president may decide to lead the project. As the project scope narrows, the likelihood increases that lower level executives and managers will provide the leadership.[90]

THE MIS COMMITTEE

Most firms like to use committees as a means of pooling knowledge for solving problems or carrying out projects, and for improving communications. When the purpose of a committee is to provide ongoing guidance, direction, and control, it is called a steering



committee. When a firm establishes a steering committee for the purpose of directing the use of the firm's computing resources, the name MIS committee is frequently used. The MIS committee is ongoing -- its membership is relatively stable, with members serving until the president decides that a change is in order. Permanent members usually include the president and the vice-presidents, including the C.O. Temporary members include lower level managers and consultants who are involved with current systems projects.

MIS COMMITTEE FUNCTIONS The MIS committee establishes policies that ensure computer support for the objectives of the firm. The committee also provides fiscal control by serving as the approval authority for all requests for computer related funds. In addition, the committee resolves conflicts that arise within the firm concerning priorities for computer use.[91]

STEERING COMMITTEE ADVANTAGES A number of advantages accrue from the steering committee approach. It centralises authority over computer use where it should be--at the top management level. Top management is actively involved in the computer effort. This involvement increases the likelihood that the computer will be used to support users throughout the firm and that computer projects will be characterised by good planning and control.

The steering committee serves as an information conduit connecting the executives, the users, and the information specialists. The committee enables each group to better understand the needs of the others.

PROJECT TEAMS The MIS committee seldom gets directly involved with the details of the systems being implemented. Instead, that responsibility is given to project teams that implement specific systems to meet the needs of individuals or groups within the organisation. Project team members include both managers and non-managers who will use the system, as well as information specialists. The team leader provides the leadership throughout the life of the project. Unlike the MIS committee, the project team is temporary, it is disbanded when implementation is completed.

The following discussion assumes that the cycle is controlled by an MIS committee working in conjunction with project teams. Whilst recognising that such an arrangement does not exist in all firms, this practice is prevalent in the company subject to this report's investigation.

PLANNING FUNDAMENTALS

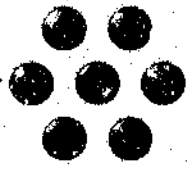
Implementation of a IS or a major subsystem is a large-scale activity involving many people and facilities, much money and equipment, and considerable time. The IS project can require as many resources as the development of a new product or the construction of a new plant, and it should receive the same degree of planning.[92]

BENEFITS FROM PLANNING THE IS PROJECT The MIS committee and the project teams anticipate that planning will yield the following benefits:[93]

DEFINE THE SCOPE OF THE PROJECT Which organisational units, activities or systems are involved? Which are not? This information provides an initial estimate of the scale of resources required.

RECOGNISE POTENTIAL PROBLEM AREAS Planning will point out things that might go wrong so that they might be prevented.

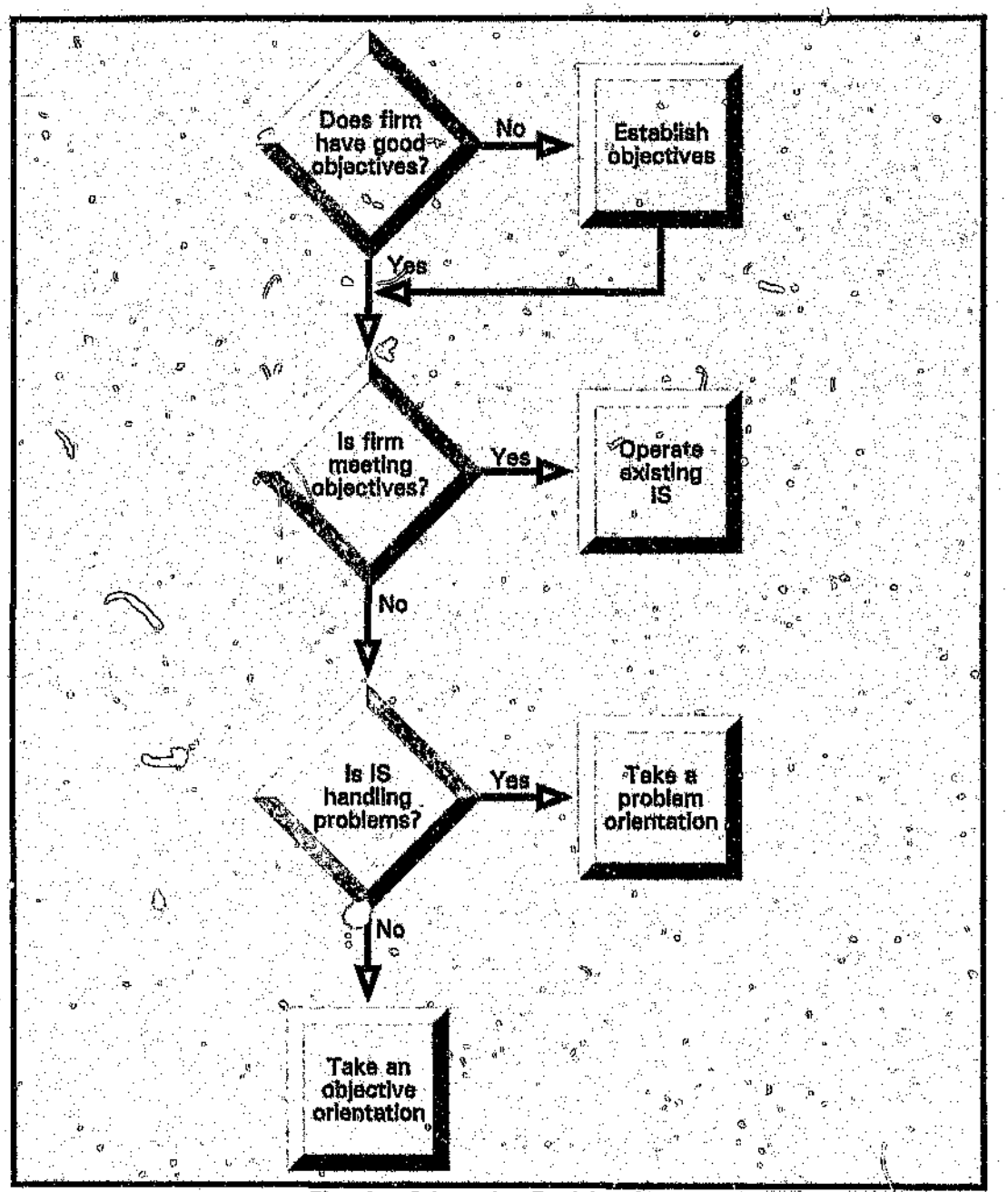
ARRANGE A SEQUENCE OF TASKS Many separate tasks will be necessary to implement the IS. These tasks are arranged in a logical sequence based on information priorities and the need for efficiency.



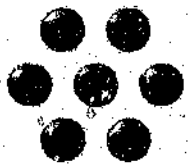
PROVIDE A BASIS FOR CONTROL Certain levels of performance and methods of measurement should be specified in advance. Each project team should define what needs to be done, who will do it, and when it will be accomplished. The team should make these specifics available to the MIS committee so that the committee can exercise control throughout the project.

Management invests time in planning with the anticipation that it will pay dividends later in the life cycle.

ALTERNATE PLANNING APPROACHES There are two basic ways to approach a IS project. One is an objective orientation where the firm uses its objectives as the starting point. The other is a problem orientation where the firm designs an IS to address specific problems.[94]



Planning Orientation Decision Chart



If the firm does not have good objectives, they must be defined before proceeding, then the firm must determine whether it is meeting its objectives. If so, there is no need to alter the IS. If the firm is not meeting its objectives, it must choose between the two planning approaches. The choice depends on how well the IS is handling the major problems. If the IS is doing an overall good job, the existing system need only be modified to better address specific problems. If, on the other hand, the IS has serious faults, the firm needs a fresh approach and should follow an objective orientation.

PLANNING PHASE

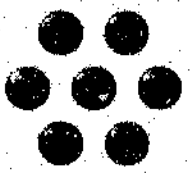
RECOGNISE THE PROBLEM The need for a IS project can be recognised by the firm's environment, the firm's executives, lower level managers, non-managers, and the firm's information services staff. Individuals and groups in the environment will often be the first to recognise a need to improve the IS. Customers, suppliers, the government, the financial community, and stockholders call weaknesses, and often strengths, to the attention of persons within the firm--often top managers. IS projects initiated by the executive's have two characteristics. First, they tend to be broad in scope, affecting the entire organisation. Second, they result in systems that influence the long-term performance of the firm.[95] Most requests for IS projects probably originate with lower line managers. Being in daily contact with their systems, these managers are quick to notice both difficulties and opportunities.

Non-managers, such as secretaries and factory workers, know the details of their systems better than anyone. When these employees do not take their systems as given and look for improvement opportunities, they can be a strong force in initiating system change. In only rare cases will information specialists initiate a IS project. The main reason is that information specialists are seldom on the scene to notice the problem symptoms. Someone else recognises a problem or potential problem and asks for help.

DEFINE THE PROBLEM Once the manager realises that a problem exists, he or she must understand it well enough to pursue a solution. However, the manager does not attempt to gather all of the information at this point. That would require a full-scale system study. Instead, the manager seeks only to identify where the problem exists and what the general difficulty is. If the manager does not wish to engage in end-user computing, then she or he solicits the assistance of the information specialists.

SET SYSTEM OBJECTIVES A logical relationship exists between the company objectives and the expected performance of the IS. The company's objectives determine the objectives that each functional area, and the managers in those functional areas, should attain. The manager's objectives, in turn, determine their information needs. Finally, the manager's information needs determine the IS performance criteria--its standards of performance. For example, a firm might have an objective of realising 12% return on investment during the coming year. To help achieve this objective, the marketing function might have an objective of R25 million in sales revenue with operating expenses under R1.2 million. In order to meet these marketing objectives, the vice-president of marketing might require periodic sales and expense reports. Later in the life cycle, the report requirements will be made more specific. At this point, however, the criteria are stated in only general terms, such as "Provide a monthly expense report that compares actual expenses with budgeted expenses for each operating unit." Similar statements establish the general performance criteria for each IS subsystem.

IDENTIFY SYSTEM CONSTRAINTS The new IS will operate under many constraints. Some are imposed by the environment, such as the government's demand for tax reports and the customers' demand for billing information. Other constraints are imposed by the firm's management, such as limits on computer cost or response times. It is important that these constraints be identified before work on the IS actually begins. In this way, the IS design will fall within the constraints.[96]



CONDUCT A FEASIBILITY STUDY The systems analyst takes the information from the first four steps and uses it as a guideline for conducting a feasibility study. A feasibility study is a brief look at the major factors influencing a system, which will enable the manager to solve the defined problem or achieve the desired objectives. It is not a full-scale study effort; this study will come later, after the project's feasibility has been demonstrated.

There are five dimensions of project feasibility study:[97]

TECHNICAL Are hardware and software available to perform the necessary processing?

ECONOMIC Can the proposed system be justified on an economic basis?

LEGAL Will the proposed system operate within legal and ethical boundaries?

OPERATIONAL Is the system design such that it can and will receive the support of the people who must make it work?

SCHEDULE Will it be possible to implement the system within the imposed time constraints?

The systems analyst gathers the information necessary to answer these questions. Personal interviews of key employees provide most of the information. In some cases additional information can be obtained by searching through records. The intent is to gather the needed information in the shortest time and at the least cost.

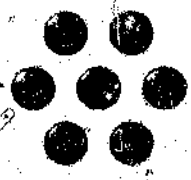
PREPARE A STUDY PROJECT PROPOSAL The systems analyst summarises the findings up to this point in a study project proposal. If the feasibility study indicates that the IS project should continue, it will be necessary to conduct a full-blown system study. The system study will provide the detailed basis for the design of the new system in terms of what it should do and how it should do it. The system study will be conducted by the analyst or a team of analysts and will span several weeks or months. The study will therefore represent a considerable expense. The systems analyst prepares a study project proposal to provide the manager with a basis for the decision to incur this expense.

APPROVE OR DISAPPROVE THE STUDY PROJECT The MIS committee weighs the pros and cons of the proposed project and system design, and takes one of three actions as shown. First the committee asks, "Is there enough information upon which to base a decision?" If the answer is "no", the committee specifies that the analyst gather additional information. Once the committee feels that sufficient information is available, the go/no go decision is made. If the decision is go, the project continues to the study phase. If the decision is no go, both the committee and the systems analyst turn their attention to other matters.

As the committee considers approving the project, two key questions are asked:[98]
- Will the proposed system solve the identified problems or enable the firm to meet its objectives?
- Is the proposed study project the best way to determine the design of the proposed IS?

It is usually easier for the committee to answer the first question than the second. Even though committee members may be both computer and information literate, they may not have a good understanding of the system life cycle. In that case, the committee must have confidence in the abilities of the information specialists to recommend the best study approach. It is here that the political credibility of the manager selling the IS proposal must be sound, and all the politics of selling come into play.

ESTABLISH A CONTROL MECHANISM Before the study begins, the MIS committee must take steps to establish control over the project. Whereas the objective of the IS is to satisfy the information needs of management, the objective of a IS project is to produce the desired IS within both cost and time constraints. Project control assures that the cost and time objectives are met.



Project control involves the specification of what needs to be done, who will do it, and when will it be done.

WHAT NEEDS TO BE DONE The MIS committee uses the feasibility study to identify the jobs that the IS will perform. For example, in the marketing information system the jobs could include:

- Product subsystem
- New product evaluation model
- Product deletion model
- Place subsystem warehouse location model
- Promotion subsystem territory
- assignment model
- Pricing subsystem
- Pricing model

WHO WILL DO IT The CIO, representing the MIS committee, next decides who will perform each of the subsystem tasks. The general design specifications identify the types of employees needed. For example, if the warehouse model will use linear programming, the persons given that responsibility must possess LP skills.

WHEN WILL IT BE DONE The knowledge of the tasks to be performed and who will do them enables the CIO to estimate how long each task will take.

The specification of the project in terms of what, who, and when a reincorporated into a project schedule. The amount of time required for each task is listed in person days. A person day is the time required for one person to do the job. By assigning multiple persons to a task, it is possible to reduce the number of calendar days, although not necessarily in a linear fashion. At this stage of the project, these figures are only approximations. They are based on the combined experience of the CIO and other information services managers in developing similar systems.

MONITORING PROJECT PROGRESS Once the project schedule has been established, it is next necessary to document it in a form that will facilitate control. There are a number of documentation techniques that range from simple graphs to network diagrams.

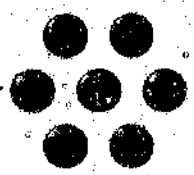
A network diagram uses arrows to represent the tasks and line-arrows to show the interrelationships. Sometimes the terms CPM diagram and PERT chart are used to describe the network diagram. Prewritten project management software is available to help the manager control the project--not a subject of this report

IMPLEMENTATION PHASE

At this point the design exists only on paper; it is a model of the planned system. It is now necessary to convert the model into a physical system. Implementation is the acquisition and integration of the physical resources that produce a working system. Many separate tasks, involving practically everyone in the firm, make up the implementation effort. [98]

PLAN THE IMPLEMENTATION A control mechanism established at the end of the planning phase in the form of a chart or network diagram should be updated and made more detailed. The managers and information specialists have a specific knowledge of the system design, and they can use this knowledge to develop a very detailed implementation plan.

ANNOUNCE THE IMPLEMENTATION The implementation project is announced to the employees in much the same way as the system study. The main objective of both announcements is to alleviate employee fears. The implementation announcement has an additional objective of appealing to the



employees for support. Many employees will be involved with the implementation, and their cooperation is is needed.

ORGANISE THE INFORMATION SERVICES STAFF The systems analysis group has completed the bulk of its work in designing the new system. Specialists in data communications and database administration also participated when needed. As the design evolved, recruiting and training efforts were launched bring the programming staff up to the needed level in terms of numbers and skill. Now the programmers are added to the project teams.

SELECT THE COMPUTER Over the years a well-detailed process has evolved for requesting bids from suppliers of computing equipment and then evaluating the proposals. In large companies this process is very clearly documented and the penalties for non-conformance very severe. Thus, once a decision has been made as to what should be installed (the details of which were covered in the bulk of the literature survey), tenders should be handed over to the designated functions.

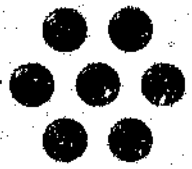
PREPARE THE SOFTWARE LIBRARY When a firm decides to create its own application software, the programmer uses the documentation prepared by the systems analyst as the starting point. The programmer might prepare more detailed documentation such as data dictionary descriptions. The coding is performed and the programs are tested. The end product is the software library of application programs. This sequence of first selecting the hardware and then preparing the software library fits the situation where the firm prepares its own software. When prewritten application software is purchased, the software selection comes first, followed by the hardware.

PREPARE THE DATABASE Preparation of the database can be difficult when (1) the firm is converting from a system of manual files to computer media, (2) the files are large, (3) the files contain very old data, and (4) some of the data has not been maintained in the past. The degree to which any of these conditions exist determines the difficulty of the task. The database administrator (DBA) should interface with the users during the analysis and design phase, gaining an understanding of data needs. When the database schema has been defined, and a decision has been made concerning which computer to use, the DBA can guide the selection of the DBMS. The DBA advises the MIS committee which DBMS appears to be the best, and the committee makes the decision. Once the decision has been made, the DBA directs the database preparation and the user training.

EDUCATE PARTICIPANTS AND USERS The IS will affect many people. Some will make the system work and others will use its output. All must be educated concerning (1) their role in the system, and (2) how the system will benefit them. The education program is aimed not only at members of the firm but also at elements of the environment. The education can be provided by systems analysts, personnel in other departments, or outsiders such as consultants.

INTERNAL EDUCATION Employees on the operational level, such as clerical personnel, factory workers, and salespersons, must learn how to do specific tasks. These tasks include filling out forms, operating terminals, and using output. Managers also must know how to use the system interpret the output, and perhaps operate terminals. But, more importantly, the managers must understand the role of their departments in the new system. They must understand the flow of data and information that links the departments.

ENVIRONMENTAL EDUCATION Suppliers and customers generally need more information on the new system than do other members of the environment. Representatives of the firm's purchasing and sales departments often communicate this information in person, stressing the need for the companies to work together. If any of the firm's employees belong to labor unions, union officials should be included in the education effort. The sessions can be conducted by members of the firm's industrial relations department, assisted by the information services staff. The people who need education and the types of education that are needed should be identified early in the life cycle. Then, the educational

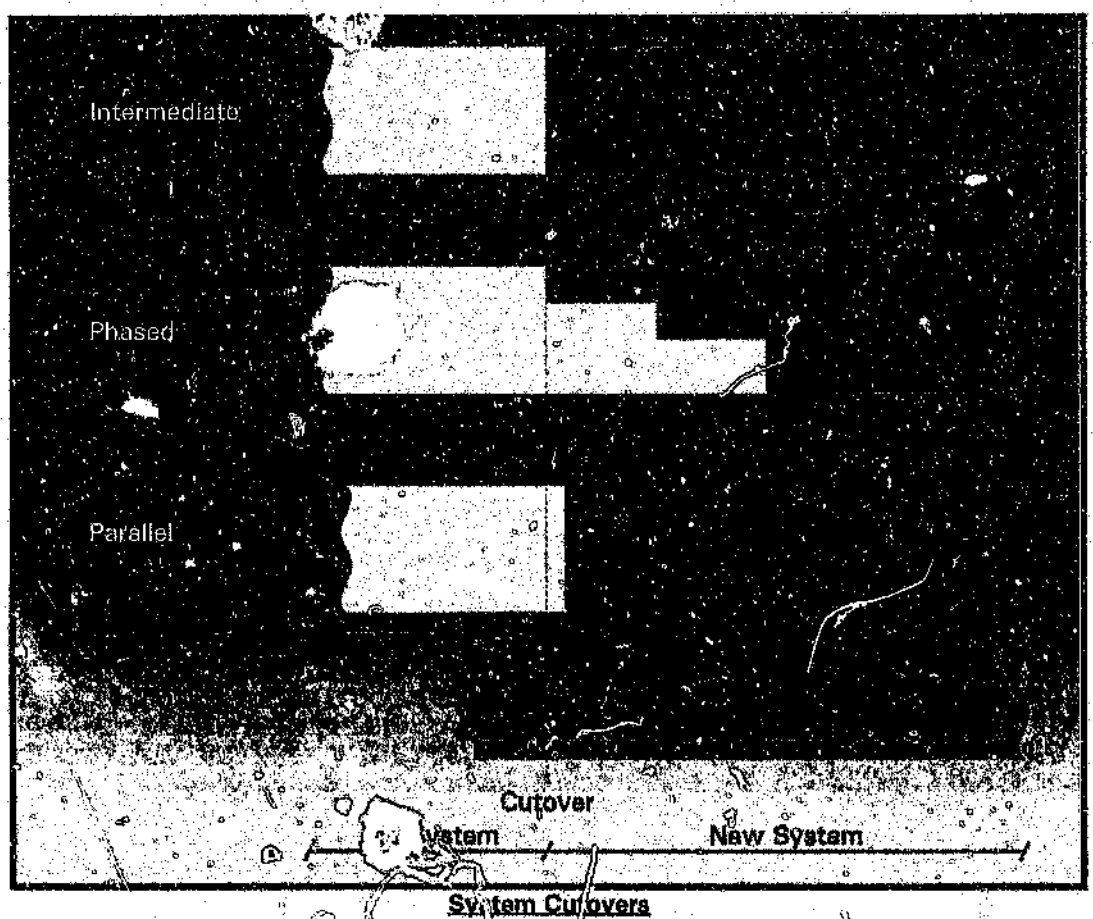


programs can be scheduled at the appropriate time-just before the learned material is applied.

PREPARE PHYSICAL FACILITIES The work required to prepare the physical facilities to house the computer depends on the amount and type of hardware needed. If only a few additional units are to be installed, they likely can be housed in existing areas. Microcomputers and small minicomputers also pose no real challenge. If, on the other hand, a new mainframe or large system is needed, a complete construction project may be necessary.

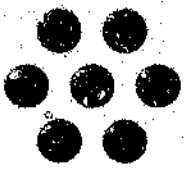
A project to construct computer facilities requires the regular construction skills plus special technical knowledge. Representatives of architectural and contractor firms, along with those of the computer manufacturer, can provide the needed expertise. The manager of computer operations usually has the responsibility for representing the firm to these outside groups.

CUTOVER TO THE NEW SYSTEM The process of halting use of the old system and starting use of the new one is called cutover. There are three basic approaches: immediate, phased, and parallel.[100]



IMMEDIATE The simplest approach is to convert from the old system to the new one on a given day. This is the least expensive and time consuming but is feasible only for small firms or small systems. As the scale of the operation increases, the timing problems become too great.

PHASED If the entire system cannot be converted at once, it can be divided into subsystems and converted a subsystem at a time. A phased cutover can be followed within a single geographic location or among several locations. For example, at the firm's headquarters the order entry subsystem can be implemented first, followed by the



inventory subsystem, and so on. Or if the firm has sub-offices located across the country, each can convert to the new system in succession.

PARALLEL. A parallel cutover requires that the old system be maintained until the new one is fully checked out. This approach offers the greatest security against failure but is the most expensive since two sets of resources must be maintained. A big advantage is the ability to fully debug the new system, using live data, before the old system is scrapped.

Once the cutover has been successfully completed, the operations phase begins.

OPERATIONS PHASE

Shortly after cutover, when the new system has had a chance to settle down, a post implementation review is conducted to evaluate how well the system is satisfying the performance criteria. This review is repeated, perhaps on a yearly basis, throughout the operational life of the system. Ideally, the reviews are made by an unbiased third party such as an auditor or consultant.

During the time a manager uses the system, minor modifications are made to keep the system current. For example, a tax rate in a payroll system is updated to reflect a change in a tax law. Such activity is called system maintenance and can account for a large portion of the information services staff time. At some point, the system will need major repair or even replacement. When that happens, the life cycle is repeated to produce the improved or new system.

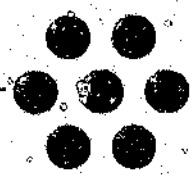
INFLUENCES ON THE LIFE CYCLE

Two innovations in system development are currently exerting influences on the traditional life cycle approach. These influences are prototyping and CASE.

PROTOTYPING This is useful when implementing expert systems and executive information systems. A prototype provides the developers and the potential users with an idea of how the system in its completed form will function. The process of producing a prototype is called prototyping, and it is best suited for those situations where the user is not completely certain what he or she wants. By providing a prototype, the user is better able to see what the possibilities are, and this better understanding can trigger more exact specifications.

Prototyping steps run as follows:

- Identify user needs: This is accomplished by the systems analyst, primarily by means of personal interviews.
- Develop a prototype: The systems analyst and the programmer use prototyping tools such as 4GLs (4th Generation Languages), DBMSs, electronic spreadsheets, and modeling languages. The strength of prototyping lies in ensuring that the initial attempts are on a much smaller scale (less work to be redone/expanded upon), yet must cover all of the system's potential.
- Evaluate the prototype: The analyst and programmer educate the user in prototype use and give the user an opportunity to become familiar with the system.
- Determine if the prototype is acceptable: The user advises the analyst and programmer whether the prototype is satisfactory. If so, the next step is skipped.
- Revise the prototype: The analyst and programmer make changes to the prototype as recommended by the user. The revised prototype is made available to the user, and the previous two steps repeated.
- Use the prototype and replace with operational system: In those cases where the prototype contains all of the desired elements, the prototype becomes the operational system. In those cases where the prototype is only a shell of the needed system,



lacking certain elements, the prototype will serve as the blueprint of the operational system.

THE ATTRACTION OF PROTOTYPING In 1989 consultants J M Careyand and J D Currey surveyed over ninety firms of all types to learn their experiences with prototyping. They identified six attractions, listed below in order of their frequency of mention:[101]

- Better user involvement
- Define requirements better
- Get it working quickly
- Design on-line processes
- Determine project feasibility
- Test new technologies and tools

In explaining the difference between prototyping and the traditional life-cycle approach, most respondents mentioned the user, along with words like see, feel, and touch.

THE PITFALLS OF PROTOTYPING Carey and Currey also identified the difficulties in taking a prototyping approach are listed below, in rank order.

- Keeping the scope of the project under control
- Managing alterations to the system
- Lack of established guidelines
- Inadequate development controls
- Inadequate application controls
- Lack of documentation
- Inability to use as an operational system
- No standard task list
- End-user drift

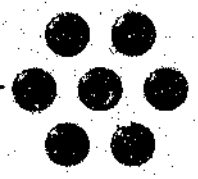
The term quick and dirty is often used in conjunction with prototyping. It appears that such use is justified in those cases when a system is produced that includes inadequate controls and documentation. However, those deficiencies are due to lack of discipline rather than to weaknesses in prototyping. The persons doing the prototyping--the information specialists and perhaps the manager working alone--can be disciplined to include controls and provide documentation before the system becomes operational. Such discipline is established through the firm's policies.

PROTOTYPING AND THE SYSTEM LIFE CYCLE Some disagreement exists concerning whether rototyping can coexist with the system life cycle. Some authorities believe that the life cycle is completely replaced. One position is that prototyping can replace the shorter term life cycles of individual subsystems within the IS, such as DSSs (Decision Support Systems) and expert systems but that it does not replace the life cycle for the IS as a whole. The prototyping activity comes during analysis and design, and, to a lesser extent, during planning and implementation.[102]

CASE

Computer Aided Software Engineering (CASE) has grown from a tentative exploration of assisting programs, to full blown packages that interface with 4GLs to the point where the distinction becomes blurred.

A prevalent example of this is Oracle's new Cooperative Development Environment (CDE), released in August 1993, where a case tool is used to define the application through management oriented schematics, and the application is then generated through the 4GL. The 4GL level can be used as the starting point for more specific tasks and if the CASE module is considered too expensive for the concern. Other object oriented programming



4GLs also take on the appearance of a CASE tool when modules are laid out as flow diagrams, with conditional modules being placed on each branch option.[103]

CASE tools are a subject in themselves, and it suffices to say that they have a fairly significant effect on the life cycle in that they make many steps redundant, and render other superfluous due to the ease of being able to jump forwards with new repercussions should any mistakes be made. CASE tools are becoming a practical reality, and further bring closer the holy grail of cross-platform applications that can be designed once, and compiled for several different operating environments (as with CDE).

DOWNSIZING SPECIFICS

IT implementations are often involved with the migration of mainframe systems to PC based client/server architectures--downsizing. Whilst this was not the approach applicable to the subject of this implementation project in Vandenberghs, it does represent a significant portion of the industry literature into IT and its implementation, and downsizing savings is the single largest contributor to IT justification in industry. For completeness, some guidelines/considerations are discussed below.

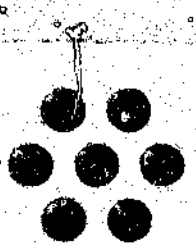
Reviews of downsizing scenarios, both locally and internationally, reveal certain common threads, some positive and some negative. Perhaps the most significant of these involve management issues rather than technical rules. Four general rules crop up repeatedly in the literature related specifically to downsizing.[104]

MANAGEMENT SUPPORT Make sure to have top management support. Downsizing essentially entails downsizing the IS culture--this is a management issue, not a technical one. Therefore the need to secure the co-operation of both technical and management staff is crucial. Gaining top management support where 'top' is defined as company president should be no problem. After all, what top executive is going to reject a plan that promises budgetary savings in addition to a data processing movement toward commodity and away from cult status?

IS DEPARTMENT RESTRUCTURING Reorganise, distribute, refocus and downsize the IS department. The nature of an organisation is likely to change substantially as distributed network based computing takes hold. Changes will occur in the business units using new systems as well as in the IS department that is developing (at least partially) the new environment. The concept of business process re engineering concerns what happens in the business units as new communications and computer systems take hold, typically, the end result is an organisation that has fewer management layers in the organisation hierarchy.

Data processing techniques are now coming full circle and changing the culture, organisation and approaches used by IS professionals. Over time, the size of the typical corporate IS department will shrink as users become more intimately involved with computer-based systems. In addition, the focus of the IS department has to change--away from applications and towards networks and databases. In other words, the successful data processing department of the future will focus on databases, connectivity and standards. The development of most of the client side work will be left to the consumers of these new systems.

In the past, the IS professional was concerned with the main frame. That concern will be replaced by the management of shared data and networks. Previously, the typical user department was concerned with a dedicated minicomputer that ran the department applications. Now the user department will be dealing with a series of applications that run on the client workstations and are, in turn, supported by data accessed locally over a LAN server or remotely through wide area net works, distributed data bases, information warehouses or other appropriate technologies.



Lastly, and of most importance, the consumer of these technologies and data is now dealing with something very different. Depending on the user's background, he/she will be accustomed to working with mainframe (slave terminals or independent PCs). That user will now have non-personal computers that sport graphical interfaces and run client side applications against shared databases or sources, very different to the clerical-like controlled environment of dumb terminals.

SKILL DIVERSITIES Staff carefully, include both PC and mainframe people on the implementation team. A downsized system includes important elements from both main frame and PC technologies. Downsizing means the use of networked workstation and PC technology to build enterprise-oriented, transaction processing types of applications. Such applications must be supported by robust systems. Robust means that data integrity and security are built in features. A robust system has recovery techniques to insure that no transactions are lost or half completed; transactions are to be either fully completed or all interim changes should be backed out. The essence of main frame systems has been robustness. PC systems have focused more on useability and immediate responsiveness. And now with Windows and Macintosh GUIs (Graphical User Interfaces) predominating, PC systems provide ease of learning and adaptability as the user interface remains similar across applications.

In many IS departments there has been a wall separating the PC support staff and the mainframe systems people. However, successful downsizing requires skill sets from both camps. The mainframe group can bring expertise in large scale systems; they understand the requirements for applications that must support large numbers of people or must operate in a 7 day 24 hour mode. PC people bring experience with GUIs, DOS, Windows, NetWare etc. Those organisations that build development teams out of both groups will do better when downsizing. It is important to do the necessary to make sure there are communications going between the PC and mainframe people.

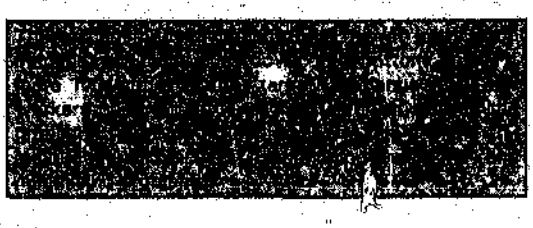
OUTSOURCE IMPLEMENTATION SKILLS Use outside help, including conferences, consultants and integrators. It should be obvious that any move to a new technology requires major investments in learning and training, and yet it is something that is frequently underestimated by even the most experienced people.

Much of the technology in this field changes so rapidly that it should all be considered current events. Conferences and industry literature are a good way to keep current, and it's essential to keep current if downsizing investments are to make any sense. This approach is tempered by other organisational philosophies of investment followed by a period of stagnation until reinvestment is again warranted by a sufficiently large progression in technology.

Probably the most frequently cited regret following downsizing implementation is that consultants or systems integrators were not used more frequently than they were. In dealing with new technologies, and bringing in knowledgeable advice is usually cheaper in time and money than trial and error discovery approaches. The general advice is to identify the technologically tough areas and have consultants available who can help when they are needed.

CHANGE MANAGEMENT

The implementation of IT is encumbered, primarily, by the change it wrecks throughout any organisation. Thus, any difficulties and considerations are largely those associated with managing change, the differences being the practical aspects outlined above.



Managing change is a massive subject of its own. It is also a highly debated topic as implementations vary considerably between cultures (company and race), group sizes, intellectual levels, and a myriad of other factors. An attempt will not be made here to lay out any strict rules on to how to manage this, but rather the more global considerations will be laid out as things would vary for each environment. A discussion on actual problems encountered falls under the scope of in-house company reports and rathes than the this information survey.

COST OF FAILURE

In order to understand the import of carefully managing change, the repercussions of not doing so need to be clarified.[105] Some are as follows:

SHORT TERM

Direct
- Wasted resources (money, time and people);
- Business objective not achieved (hence all associated repercussions)

Indirect
- Morale suffers;
- Job security is threatened.

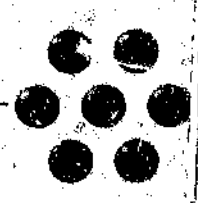
LONG TERM

Direct
- Strategies are not completed.

Indirect
- Lowered confidence in leadership;
- Resistance to change increases;
- Consequently next change project is likely to fail;
- Organisation survival is threatened.

MANAGING BARRIERS

Some typical human barriers to technological change (the more commonly documented ones), and the more recent ideas as to how to manage them are outlined below:



BARRIER	CONTRIBUTING FACTORS	STRATEGY
Excessive expectations	• Overselling of results • Bad understanding of how people change	• Manage transition period (covered later)
Excessive action orientation	• Crises/fires • Historical record of crisis management • Short term frame of reference	• Establish long term objectives • Define frame of reference
High personal risk for sponsors	• Short term reward systems in company	• Modify reward/appraisal systems
Lack of collaboration	• Territorial imperative • Insecurity of people	• Generate common goals in leadership and reward systems
Inevitable resistance to change	• Loss of status and power for persons affected • Inertia required to implement change hard to generate	• Anticipate and minimise resistancereassure people • appoint project champion and steering committee system
Uncertainty	• Lack of skills in people to be using system	• Communication/training and involvement • Manage implementation (covered later)

These strategies vary somewhat between various literature sources, but the basic approach to each remains sound--each requires expansion for each new problem.

SUCCESS FACTORS

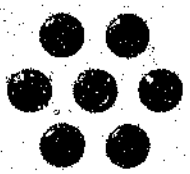
With an understanding of where one's strength and weaknesses lie in the following categories, one is better equipped to deal with the inevitable resistance to change:

- Implementation history: Success and failure of past implementations;
- Sponsors: Level and type of commitment of key stakeholders;
- Change agent: Skills and motivation of transition managers;
- Change impetus: The profile of main motivator for change;
- Target resistance: Inevitable resistance from all groups affected;
- Culture: Values, behaviours and 'unwritten rules' that support or oppose change.

TRANSITION MANAGEMENT

Most change models presented in the literature have three distinct phases. Typically these are unfreezing the organisation, transition where one is working and moving towards change, and finally refreezing the organisation.[106] The motivators, strategies and main players involved for each are shown below:

	PRIMARY MOTIVATORS	PRIMARY STRATEGY	PRIMARY ROLE PLAYERS
UNFREEZING	Need/discomfort/crisis	Questioning why	Sponsor
TRANSITION	Support	Implement structure	Agent/target people
REFREEZING	Solution	Questioning how	Target people/agent



Some tactics that can be adopted in each of these phases are as follows:

	COMMUNICATION	REINFORCEMENT
UNFREEZING	• Explain objectives and rationale • Emphasise cost of not changing • Focus on external drives for change rather than internal deficiencies • Describe personal belief in change • Be specific about what is and is not changing for each group • Utilise the language and frame of reference of the target group	• Demonstrate sponsorship rationale • Illustrate arguments practically ('Walk the talk') • Emphasise reinforcement management (rewards, punishment, level of effort)
TRANSITION	• Repeat change messages often • Change medium and words to impact various frames of reference • Focus attention ahead toward achieving change rather than on returning to the past • Provide as much information as possible • Encourage overt resistance • Provide training in change • Communicate things that won't change	• Create an atmosphere in which it is safe to surface resistance • Involve targets in implementing the change • Provide resources--time, energy and money to achieve change • Make symbolic actions to demonstrate commitment • Reword supporters and motivate resisters with reinforcement • Celebrate small victories
REFREEZING	• Communicate progress • Acknowledge the costs that targets have paid • Be honest about problems that still remain or new problems that were not anticipated	• Demonstrate sponsor commitment • Maintain reinforcement management • Reward champions • Celebrate victories

MANAGING RESISTANCE MANAGEMENT

REASONS FOR RESISTANCE

- Maintenance of the status *quo* and avoidance of the transition period;
- Protection of individual and organisational values, emotions and *modus operandi.*

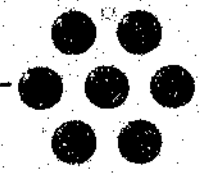
RESISTANCE AUGMENTATION FACTORS

Resistance to change increases when there is:

- A low perceived need for change;
- An implication of poor past performance;
- A high level of disruption;
- A low reward structure for high costs/input;
- Resulting negative consequences;
- An irreversible outcome;
- Doubt about the project success;
- Fear of the unknown;
- Unclear expectations;
- Low target involvement.

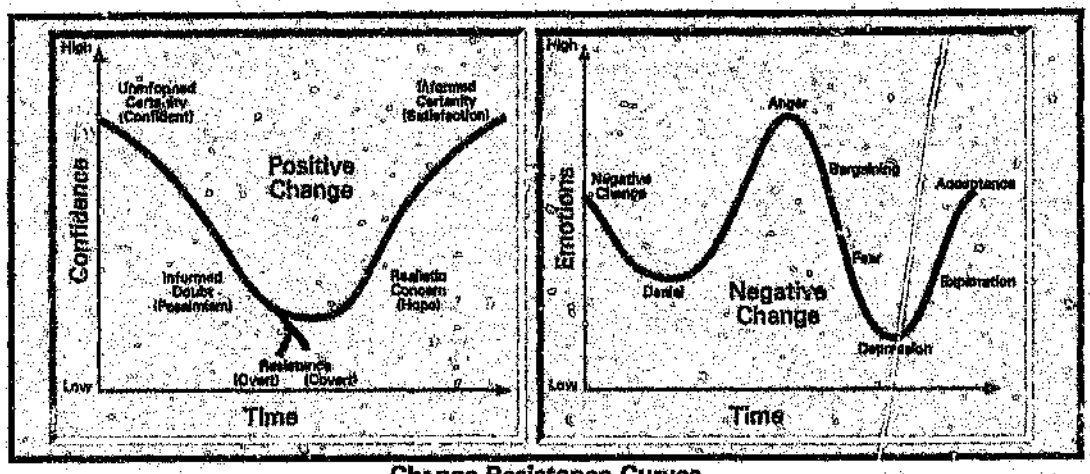
METHODS OF RESISTANCE

- Can't do it: Skills inadequate;
- Won't do it: Motivation lacking;
- Overt: Voiced objections--often founded;
- Covert: Clandestine attempts to thwart project.



CHANGE REACTION PATTERN

Two resistance curves are followed by the targets when change is being implemented. The first is desirable and expected for positive change, the second is adverse and accompanies negative change[107].

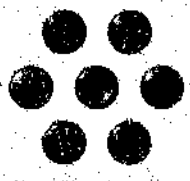


Change Resistance Curves

REACTION MANAGEMENT

The first reaction pattern above is very similar to that followed by the Situational Management model proposed by the Ohio State University (covered later).[108] Management of each stage follows the guidelines proposed in this model, but modified to cater for change. The second conforms more closely to adult-child transactional analysis and should be dealt with in only the first two quadrants/sectors outlined in the aforementioned model.

REACTION	MANAGEMENT RESPONSE
Uninformed Certainty	• Surface all potential unintended consequences of the change--recognise what could possibly go wrong while at the same time maintaining enthusiasm; • Communicate that this is panacea--there were problems before the implementation--there will be problems after it; • Gather the frames of reference for targets for the implementation.
Informed Doubt	• Acknowledge both the negative and positive data; • Mobilise resources to anticipate resistance.
Overt Resistance	• Develop a problem solving climate--surface issues and utilise targets to help with implementation; • Demonstrate that attention has been paid to feedback; • Allocate specific resources to deal with resistance.
Covert Resistance	• Continuously create as safe an atmosphere as possible--resistance will not surface initially; • Provide different forums for resistance (surveys, focus groups and suggestion schemes); • Separate the content of the content of the resistance from the process--provide recognition and reward for bringing resistance forward, even if the content is unfounded.
Realistic Concern	• Build target confidence by acknowledging progress; • Identify barriers that still exist and continue to deal with both the negative and positive data; • Increase focus on 'follow-through' and the nearness of completion.



Informed Certainty	• Reward achievement; • Identify what was learned for use on following implementations; • Identify new tasks to be accomplished; • Prepare for next change.
Denial	• Strengthen relationships; • Divide change into small steps and focus on first steps; • Avoid confrontation.
Anger	• Legitimise the anger--targets have a right to be angry as they do not have control; • Do not take things personally.
Bargaining	• Do not bargain as this changes the definition of the change and will generate a new response; • Confrontation--and 'no-deal' stance.
Fear	• Provide support; • Inform targets that resources will be provided to assist them; • Increase two-way communication tactics.
Depression	• Encourage responsibility; • Reform the change to test for opportunities; • Provide support.
Exploration	• Test new options in the new situation; • Acknowledge progress; • Build confidence.
Acceptance	• Reward and acknowledge progress; • Identify what was learned to use on following implementations; • Prepare for next change.

MANAGEMENT QUALITIES

As is outlined in the discussion on situational management, the role of the manager and change agent is important. This implies that the right choice of person is critical to success. Some characteristics of effective change agents, above those of normal managers, are listed below:

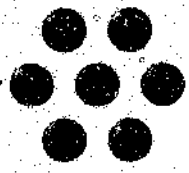
- Successful personal and organisational history;
- Credibility with key sponsors;
- Trust with key targets;
- Awareness of cultures (company and racial);
- Project champion responsibilities;
- Ability to develop teamwork among sponsors, agents and targets through common goals and interdependence for success;
- Communication skills;
- Value differences;
- The ability to integrate diversity;
- Implementation/action orientation.

THE APPROACH POLICY

All the aforementioned qualities need to be aligned to help in the adoption of a transition management approach rather than a hammer policy, the differences being outlined below:

	HAMMER	TRANSITION MANAGEMENT
Investment	High afterwards	High early
Maintenance	High	Low
Time Frame	Fast	Slow
Motivation	'Do it!'	'Do it right'
Commitment	Organisational: 'Theirs'	Individual: 'Ours'

The method adopted is crucial if the change implemented does not meet up to expectations. It is here that the repercussions of bad management manifest themselves and what achievements were obtained start to crumble--failure.



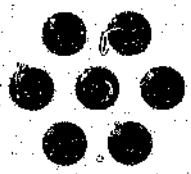
What most people don't realise is that few implementations do meet up to expectations, and that a certain degree of failure is inevitable. It is this deficiency that needs to be carefully managed and anticipated through the above approach. The reasons for failure, in this lesser degree, are numerous.

If one considers why change is so difficult for people to cope with, one will conclude that it is invariably the magnitude and severity of the change. Even with the most brilliant change management program some implementations will be doomed as they attempt to achieve too much, too soon and too radically. Often these extremities are adopted deliberately in the belief that the magnitude of change is required to justify the implementation, and that a project would not obtain approval with any lesser scope.

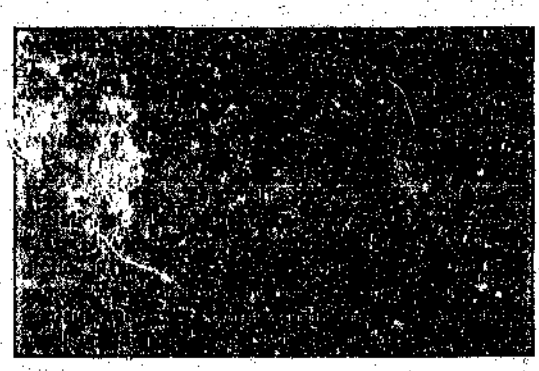
Other failings arise from the setting of unrealistic goals. Managers, being of a somewhat higher order, tend to set goals that they themselves are capable of achieving, invariably more than that which the targets are capable of.[108] Thus, very few systems are implemented within the specified time, span the originally perceived scope, and fall with projected budgets. This problem is further exasperated by the widening skills gap in South Africa, with white collar skills decreasing and the blue collar work force size increasing.

Another reason for 'failure', is the desire to implement the latest, often untried, technologies. In more philosophical fields (JIT, OPT, FMS, etc.) this often amounts to attempting an implementation that cannot fit into the company/national environment, rather than a modified system utilising some desirable aspects of the philosophy. Few managers seek credit for implementing tried and tested systems when the opportunity exists to install an entirely new concept along with all its academic recognition.

In short, complete success is highly unlikely in any change implementation, and careful management of the shortfall requires due consideration.



GENERAL MANAGEMENT



A little more attention is dedicated to this aspect of management than to that of managing change, as the author has adopted an approach that is somewhat fixed yet quite substantiated. The system is one taught through the Unilever Management Development Program, by external experts, and as the text will show is extremely robust, aligns several other management theories taught through typical institutions, and has been fairly extensively researched and tested by the author before its acceptance.

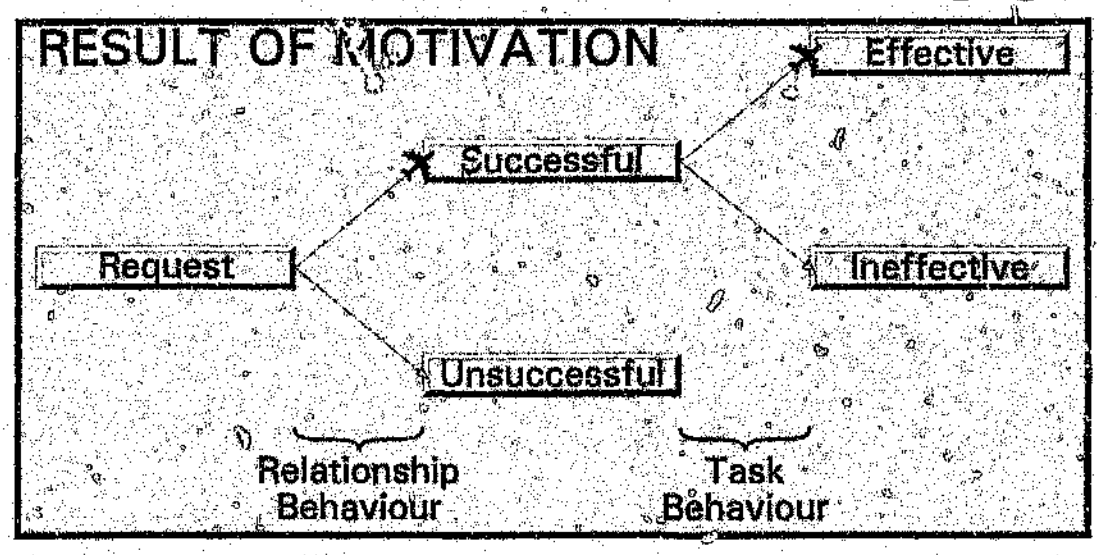
Modern approaches to motivation tend to incorporate a certain amount of Vroom's expectancy model, a strong aversion to the classical approach to performance appraisal systems, and a leaning towards reward systems for due performance. The latter is gaining momentum based on the world-wide recession of the 90's, wherein it is not economically viable to support employees who are not performing, and performance is the essence of survival, and the rest does refer strongly to all the motivational concepts discussed thus far.

The whole approach can be summed up under a model developed by the Ohio State University in the USA, and augmented to encompass Vroom's thinking as recently as 1988.[10] The model not only encompasses motivation and its inexorable link to management styles, but also addresses the complex issue of management versus leadership and employee development, as well as being applicable to all levels of management from unskilled labor, through to personal/social transactions (these being considered more complex than the management of highly skilled labor). In short, managing through this model, in the author's humble opinion, can address all the relevant issues of management today in one fell swoop.

Little literature is available on this model, save for a couple of articles and references in recent Engineering publications and Unilever course work notes, and thus what follows is a brief outline of its working as gleaned from the aforementioned training program. It suffices to say that of all the theory taught in University undergraduate and postgraduate Engineering courses, this seems to be the only theory that sufficiently compounds all the current theories and presents a truly systematic approach that can realistically be implemented rather than being a mere observation of human behavior.

THE OHIO BEHAVIOURAL MANAGEMENT MODEL

BACKGROUND The model is built up from the belief that any attempt to obtain an action from a subordinate (motivate him) is a two stage process, and that the ultimate task may only be obtained if both stages are managed correctly.



The Ohio State University Subordinate Behavioral Path

As the above picture depicts, any instruction/request from the manager has a resultant behavior in the subordinate. Th outcome of this behavior (successful means he moved to attempt the task presented) is dependent upon the first management input stage, called relationship behavior. This is the extent to which the manager engages in providing support by two-way or multi-way communication. This includes listening, encouraging, facilitating, providing clarification, praising, and is often termed 'stroking' the subordinate. Thus, whether a subordinate will respond to a request is dependent upon the nature of this behavior--largely related to Maslow's social need (and also addressing safety--job-wise that is--and esteem to a lesser extent). Relationship behavior has its foundation in transactional analysis (covered above) which is also the basis for human behavior. Thus human behavior is linked to the model.

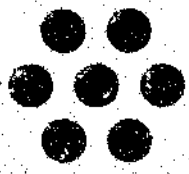
The second step is the result of the subordinate's attempt. If he is competent he may well succeed, if not he will fail. This 'competency' is derived from the manager's second input called task behavior, defined as the extent to which the leader engages in spelling out the duties and responsibilities of an individual. This includes telling people what to do, how to do it, when to do it, where to do it, and who's to do it.

RELATIONSHIP BEHAVIOR	TASK BEHAVIOR
• Employee centered • Showing consideration • People concern • 'Stroking' subordinate	• Job centered • Initiating structure • Production concern • What, where, when, why, who, how

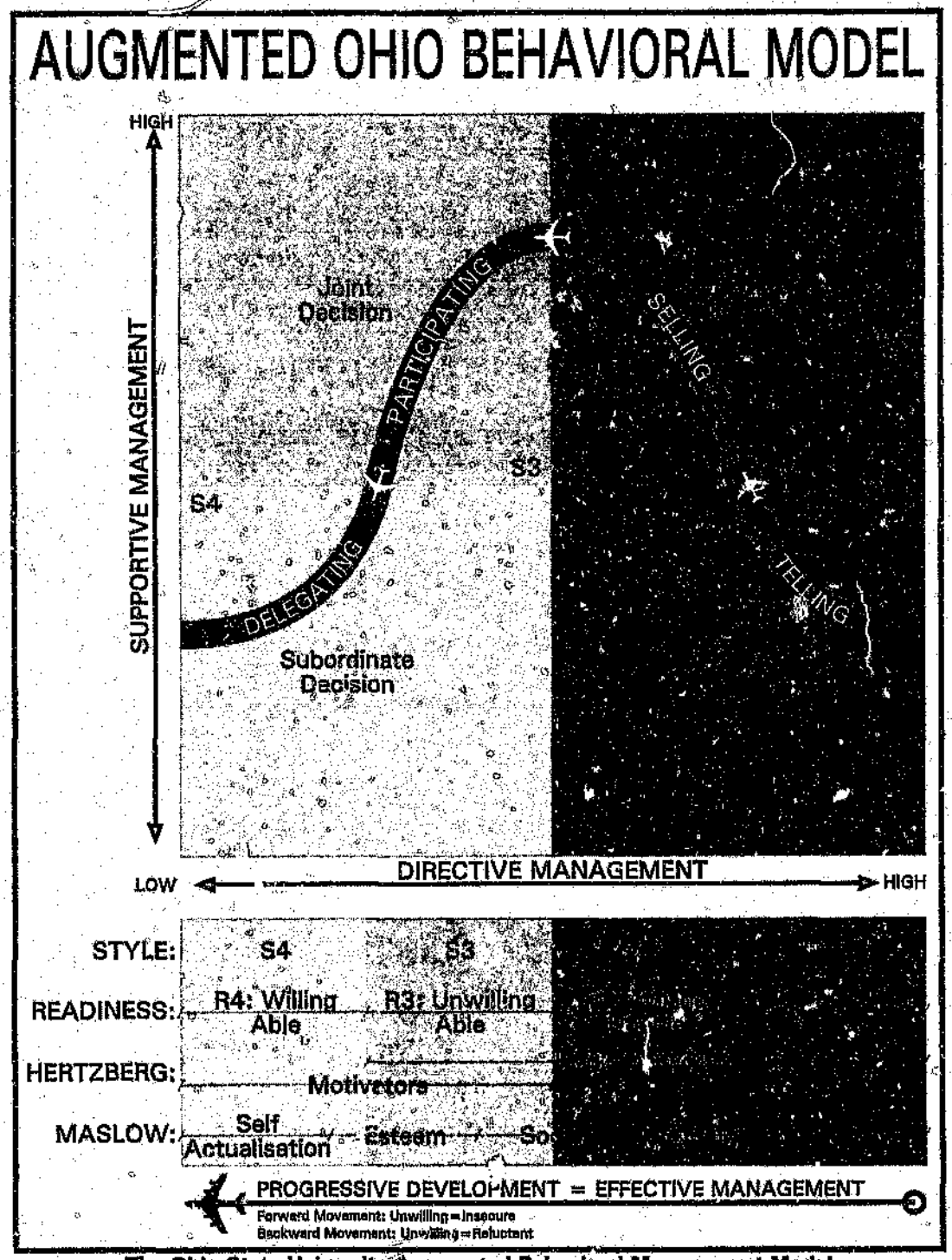
THE BEHAVIOR RELATIONSHIP

Naturally, the amount of each type of behavior that is exercised must vary for each individual, and, in fact, it must vary for a single individual as he progresses through his career. A highly competent employee will not take well to being told how to do a job, yet the same employee required explicit direction when he started. Another employee may get 'too friendly' and thereafter disrespectful if too much social interaction is entered into, whilst the same employee will later require lots of praise and encouragement whilst tackling new areas of responsibility.

This all sounds extremely complex if one pauses to consider some more implications, but the beauty of the model is that the indulgence into each behavior can be graphed to indicate one of four classic styles of management. At this point the graph in the figure below may be referred to, the areas not explained thereafter will be dealt with when the subordinate side is addressed.



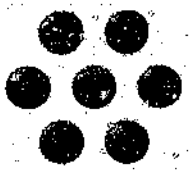
The Behavioral Management Model



The Ohio State University Augmented Behavioral Management Model

Management Styles The graph of support (relationship behavior) and direction (task behavior) as imposed by the manager upon the subordinate gives rise to four quadrants each depicting a management style. The quadrants are not distinct with hard boundaries, and merely represent the area in which a manager does/should operate.

Starting with style 1 (S1--bottom right hand quadrant), the management style is one of telling in which the manager makes the decision and provides a lot of direction with little support. This is considered as autocratic management and is particularly suitable for



incompetent (new) workers. It is typified by one-way communication in which the manager gives specific instructions and supervises closely.

Style 2 (Quadrant S2) is often referred to as 'selling', in which the manager explains his decision thus providing the subordinate with more impetus to perform. The process of explanation requires an increase in relationship behavior, and it is desirable to curtail direction somewhat–the reasoning for this follows later

Style 3 (S3) involves reaching a joint decision in which the subordinate and manager both have impetus, and is termed participating. Relationship behavior remains high but direction is dramatically cut down as the subordinates ideas are indulged. This style is desirable where the subordinate may have valuable experience that would aid the manager. The relationship input is necessitated by the high level of communication and trust required to prevent classical conflict situations (an entirely separate paper altogether).

Style 4 (S4) is the classic delegation in which a job is handed to the subordinate with no recommendation on how to go about it, and little need for encouragement. Responsibility and the decision lie entirely with the subordinate, and the manager merely appraises thereafter. This is a style which managers have great difficulty dealing with and is often the reason they do not ever perform at their optimum.

Links to Other Theories

All of these styles are addressed from different angles in Harris' One Minute Management, Monkey Management, Arnold Mohl's Productivity Management and many other highly recognised motivation works. This model simply puts them in a model that can be easily remembered and readily referred to unlike the other lists of do's and don'ts. The importance of selecting the correct style will become more apparent as we move onto the lower half of the model (the subordinate readiness), and the fuzziness of the boundaries will become more apparent.

Style Selection

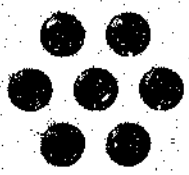
To give a rough idea of the consequences of an incorrectly applied style, let's look at a couple of examples. Bear in mind that most managers implement one style most of the time, and are completely unaware of the need to adapt and change.

If a new subordinate is incompetent, and we delegate (S4 instead of S1), he will turn out a result that may cripple the business. If, on the other hand, the subordinate is highly competent and we use a telling style (S1 instead of S4) we get a very demotivated employee with no self-esteem. Here we can start to see the link to Maslow, the previously mentioned task outcome resulting from two types of management behavior, and the link between management style and motivation. Both the above could have fairly severe consequences and S1&4 are considered to be the high-risk high-reward styles of management. Styles S2&3 are considered lower risk due to the manager's presence in the decision making, yet lower return due, again, to the manager having to include his time in the task and the possibility of lowered subordinate motivation.

Finally, we tie leadership into the model when we consider the fact that moving from S1 to S4 must include development of the subordinate (the definition of leadership) in that h' reliance on both direction and support are tempered. In addition, the process of forward movement has been shown to instill a high level of trust leading to enhanced future efficacy. A movement from S4 to S3 is considered a regression, is highly undesirable, and is usually a function of the subordinate's 'readiness'. It is this readiness that also allows one to determine which style of management would be most suitable.

Subordinate Readiness

The Ohio model augmentation of 1988 stemmed from extensive research into the subordinate side of the model. This led to the bar below the graph that depicts the state of the subordinate's mind and abilities, and relates it directly to the management style. The researchers found that all subordinate states could be categorised as conforming to one of the four combinations of willing, able, unwilling and unable. Readiness varies from person to person, and for a single person from task to task.



Ability is defined as the knowledge, experience and skill that the individual has towards the task. Willingness is his confidence, commitment, and motivation to accomplish the task. It is important to realise that willingness is actually a measure of security and the word unwilling can be replaced with insecure (or unconfident). The four states are as follows.

Level 1 (R1) Unwilling and unable: The individual is neither able to do the task, nor willing--he is insecure and requires encouragement (relationship behavior) if he is to progress to another state.

Level 2 (R2) Willing and unable: The individual have received enough encouragement to continue, but is actually incompetent for the task. This is an extremely dangerous stage as he may be liable to rush into something and mess it up. Subordinate confidence if often reliant upon the manager's involvement and hence there is a dual requirement for the manager's relationship behavior.

Level 3 (R3) Unwilling and able: The individual has the necessary skill but is not confident about the new task presented to him. This is classic in development when a person is promoted to a position in which he feels out of his depth and lacks the confidence to go it alone. Management relationship behavior is necessary to provide this impetus, though direction is not very necessary.

Level 4 (R4) Willing and able: The subordinate is both competent and confident--he may be delegated to, in fact, he must be delegated to else he will start to lose faith in his ability (if his manager doesn't seem to trust him enough to leave him alone), and will regress to R3.

The strong link between Styles and Readiness is quite apparent, and the import of matching the two is self-evident. These readiness levels are quite comparable to the 'B' states expounded by Vroom in his expectancy model.

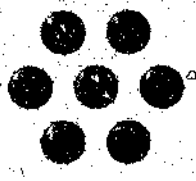
REGRESSION & PROGRESSION

As mentioned earlier, the aim of the leader is to ensure progression along the path as indicated by the '→' in the graph. This is considered as developmental and is highly desirable as staff then become more capable in their competence, are suited to promotion to positions that may be vacated, and in general are capable of taking work away from the manager.

In forward progression along the curve, unwillingness is characterised by insecurity. But the word 'secure' was not used as in backward regression (where a subordinate's readiness decreases, and so too must the applied management style) unwillingness is considered more as reluctance. Thus, a subordinate may move from R4 (willing and able) to R3 (unwilling but able) if he make a severe mistake, and though perfectly confident that he won't make it again, may be extremely reluctant to take the risk. A regression from R2 (willing but unable) to R1 (unwilling and unable) usually occurs when the task content is increased beyond the employees capabilities. The employee needs more S1 management to bring him up to a readiness level of R2 for S2. The R3 (unwilling but able) to R2 (willing but unable) regression is somewhat unusual in that is seems unlikely to occur. This can happen when a mistake shatters the subordinate's confidence to the extent that he is actually unable to perform. His willingness will stem simply from an inability to comprehend the possibility of a recurrence.

IMPLEMENTATION

The implementation of the model is somewhat unorthodox. Rather than keeping the theories 'in-house' as with most strategies, the developers advocate teaching subordinates about the model being used on them. The idea is that they will then be able to recognise



how they are being managed, and request a change in style if it can be warranted. This helps prevent the usual management traits that make it so hard for some managers to adapt.

This kind of approach is often referred to in topics on 'feedback' in which the subordinate is advised to request appraisals of performance where these are not readily given. Further, in understanding the importance of progression, the system can be used as an unbiased behavior modification technique when the subordinate fails to perform. Here, a manager may have to regress his style and the subordinate would appreciate the subtle, yet highly effective, reprimand.

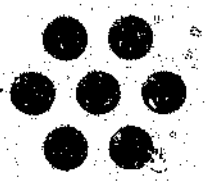
Naturally the converse is also true in that progression will feed the employee's esteem resulting in increased motivation.

CONCLUSION

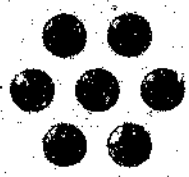
The topic could bear considerable more elaboration at the risk of changing the emphasis of this report, but this is left for individual contemplation. It suffices to say that the model encompasses most of the standard motivational theories such as:

- Skinner's Behavior Modification (through a change in styles);
- Thomas Harris' Transactional Analysis and defense mechanisms (the foundation of the model's research);
- Vroom's Valence Model (adapted to readiness levels);
- McGregor's Theory X/Y Management (covered in S1/S2-4 with both systems being allowed their merits);
- Maslow's Hierarchy of Needs (social, esteem and self-actualisation fed, whilst safety and physiological needs are supported);
- Hertzberg's Hygiene Factors and McClelland's Achievement Motivation (as an extension to Maslow);
- Participation Motivation (in S3&4 management, but only applied where applicable--not indiscriminately);
- Taylor's Authoritarian versus Human Resource management styles (as for Theory X/Y).

Hence the authors selection as the model for general implementation in an environment where management style is all imperative, yet the time to contemplate each new decision is not available and an easily referenced model is necessitated.



THE FUTURE



TECHNOLOGIES

According to the gurus, the PC industry is:[111]

- About to experience the biggest battle in its history--one that will decide which of the growing number of 32-bit operating systems will dominate the market for the next 10 years.
- About to enter a new, more fragmented era in which no single operating system will be dominant in more than a few segments of the market.
- About to enter Phase II of the Windows era, because the operating systems war is over, Windows has won, and all this talk about Unix, OS/2 and so forth is just hot air.

There will be a struggle for dominance among network operating systems, as new kinds of PC operating systems and applications place new demands on local and wide area networks. There will be a skirmish in the application-server arena, as the migration to client/server architecture and downsizing of mission-critical applications turns the spotlight on the database server locked in the back room.

Systems will vie for the mind share of corporate developers, who are looking for answers to their client/server needs. And finally, there will be a vicious fight for dominance on the desktop--the final criterion by which success has long been measured in the PC industry.

No one knows yet whether a single 32-bit operating system will capture all three server, development, and desktop client areas or whether several operating systems will co-exist.

The latter outcome holds considerable merit, because it would allow organisations to apply the best technical solution, or pick the best price/performance option, for a particular situation. But it also raises substantial questions about interoperability.

Moreover, despite all the hoopla, no one knows for certain which operating systems will engage in these battles.

Microsoft, for instance, is entering the fray armed with Windows NT and the venerable Windows 3.1/DOS combination, but claims to have significant reserves in Chicago, a scaled-down 32-bit Windows-based operating system for desktop use, and the Cairo project, a fully object-oriented Windows-based operating system.

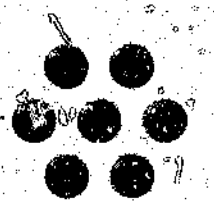
IBM, meanwhile, is betting on OS/2 as its PC operating system for all purposes--but that could change when Taligent, IBM's joint venture with Apple Computer, delivers the object-oriented operating system that it has in development.

Apple, not wanting to be the only one with all its eggs in one basket, is working with Novell to develop an Intel-based PC version of the Macintosh operating system running on top of Novell's DR-DOS.

Finally, while SunSoft's Solaris and Next's NextStep have the spotlight at the moment, it's anybody's guess which, if any, of the Unix-based operating systems will achieve enough prominence to avoid getting lumped in e ail phrases such as "and other Unix-based systems."

The requirements of application servers, development environments and desktops vary considerably, as do the abilities of the 32-bit contenders.

On the desktop, for example, many factors come into play: hardware requirements, application availability, compatibility with current applications and user-interface considerations. A different set of factors emerges in the realm of the application server,

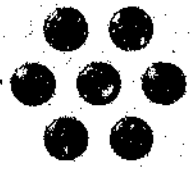


where performance and stability are the keys to success. The development environment requires dependable tools and off-the-shelf software.

Despite their different requirements, the divisions between these niches are anything but clear. Security and performance are important to end users, just as application availability and interoperability are important on application servers. By the time one gets halfway down anyone's list one will have hit nearly all of the factors discussed above. In the end, one thing is clear: 32-bit operating systems are about to present PC buyers with the toughest decisions that they've ever had to make. On the one hand, it is possible to make a good case for applying nearly any of the available or forthcoming 32-bit operating systems to nearly any task. If one's guesses are correct, it is also possible to achieve great successes.

On the other hand, corporate buyers who make the wrong leap of faith could find that they've made critical, career-threatening mistakes.

It is a time of enormous opportunity and enormous peril. Wise buyers will weigh these decisions carefully, allowing neither promises nor partisanship to any company or product to sway their decisions.



ARCHITECTURES

Client/server computing as most organisations use it is a temporary stopping point, a rest area on the road to the real goal: distributed computing. Today, a relatively few specialised machines--servers--provide a relatively few services--database, electronic mail--to networked clients.

That's fine for the moment, but much more is possible. True distributed computing, in which each piece of computing work can run on a system well-suited to perform that work, is not just a dream. Achieving it, however, will be a nightmare without standards. Chief among the necessary standards are those that govern the management and interaction of distributed objects.

Imagine the following: You need to mail to your boss a numerical analysis of the last decade of financial data for each division of your company. Your desktop system is a relatively under powered machine, but you're part of a distributed computing network, so you're in luck. You tell the analysis program to run the job, and it handles the rest.

First, it casts about for the data by shouting onto the network the equivalent of, "Hey, who can get me the complete financials for the last decade? A database server responds and ships the data to your system. Your system knows it's not a fast number cruncher, so it shouts a request, this time for a serious compute engine. A system answers, your system ships the data to that system and gets back the answer. Your system adds a note you wrote and hands the whole shebang to an E-mail server, which delivers the mail to your boss.

This scenario is obviously extremely complicated to implement, because it assumes a lot of underlying abilities. Each system must know its capabilities and be able to talk to other systems about those capabilities. The network must contend with many simultaneous interactions just like the ones the client system needed.

The answer lies in distributed objects and distributed object management. These technologies will give systems on a network the ability to treat different services and forms of data all simply as objects available for use.

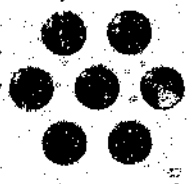
If this technology seems too far fetched, think again: One can see precursors to it on PCs today. The most common one is the Object Linking and Embedding (OLE) standard of Windows. OLE admittedly runs just on a single PC and is nowhere near as flexible as the above scenario, but it still has the idea right. It lets any OLE-compatible application work with data from any other OLE-compatible application.

Still, the leap from OLE to true distributed computing is obviously a big one. To make that leap, the industry needs standards.

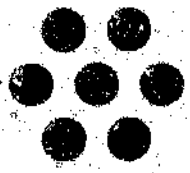
Unfortunately, a lot of different groups want to own those standards. The Object Management Group (OMG) exists just to define such standards. All the big micro players, including Microsoft, Apple, IBM and Novell, have their own ideas about these standards.

It's time for these groups to play together. The best starting place is Microsoft's OLE, which may not run on any platforms other than Windows, may have its share of technical flaws, but a plethora of PCs support it.

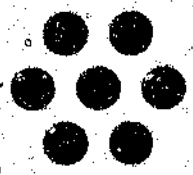
To move the industry forward using OLE, the OMG and Microsoft should work out a plan for evolving OLE into a standard basis for distributed computing. As an industry consortium with a wide variety of members, the OMG makes a great repository for these plans. For its part, Microsoft should yield control of OLE to the OMG.



Other vendors then would just have to rally around the OMG, and we could all move a little faster into tomorrow. The odds of this happening, of course, are pretty low, but the sooner we encourage Microsoft and other vendors to adopt industry wide object-management standards, the sooner we'll finally reach the stage of true distributed computing.



GLOSSARY



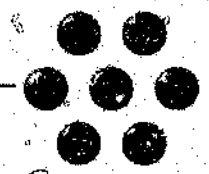
COMPANIES AND PRODUCTS

Several prominently featured companies and their database server products are listed. This list represents the industry standards in server engines.

COMPANY	PRODUCT	OS
Gupta Technologies	SQLBase Server	NLM
Ingres Products	Ingres Server	OS/2
International Business Machines	Extended Services with Database Server	OS/2
Microsoft	Microsoft SQL Server	DOS
Novell	NetWare SQL	NetWare
Oracle	Oracle Server	Netware
Sybase	Sybase SQL Server	Netware

Several other companies feature a lot, the following list shows their main product in the context of this report.

COMPANY	PRODUCTS
Apple	Macintosh OS
Banyan	Vines NOS
DEC	OpenVMS
IBM	OS/2 OS
Intel	x86 chips
Microsoft	DOS, Windows, Windows NT OS
Next	NextStep OS
Novell	NetWare NOS
Solaris	SunSoft OS
Various	Flavours of Unix such as UnixWare, Univel, ...



ACRONYMS AND WORDS

3GL: See Third Generation Language

4GL: See Fourth Generation Language

A

OFFICE AUTOMATION (OA): Workstation application programs that simplify otherwise normal duties such as word-processors, spreadsheets, etc.

AGENT SEQUENCED PACKET EXCHANGE: Software or a combination of software and hardware that gathers information about a network device and executes commands in response to a management console's requests. In SNMP, the agent's capabilities are determined by the MIB.

AMERICAN NATIONAL STANDARDS INSTITUTE (ANSI): The standards group responsible for a number of telecommunications standards.

AMERICAN STANDARD CODE FOR INFORMATION INTERCHANGE (ASCII): Standard alpha-numeric code and character set common in the US and thus most PCs.

ANSI: See American National Standards Institute

API: See Application Programming Interface

APPLICATION PROGRAMMING INTERFACE (API): An application (usually background) that lets programs communicate with a database using function calls instead of SQL statements.

ASCII: See American Standard Code for Information Interchange

AVAILABILITY: This is a quality of an information system which specifies the extent to which the organisation is dependent upon having the information when it needs it, or having the system in place when it is required. The extent of reliance upon availability can be determined by estimating the cost of the situation in which the system is unavailable.

B

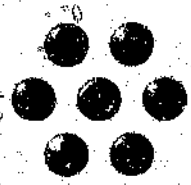
B: Symbol for a byte, a measure of data size, often prefixed by M or k to indicate Mega and kilo respectively.

BINARY LARGE OBJECT (BLOB): The acronym given to any data that doesn't conform to the simple text, numeric, date, etc. types commonly used in databases. Usually refers to picture files and multimedia objects. In modern implementation of database systems, BLOBs can not only be stored, but can be identified in terms of the procedures used to edit, display or execute them.

BLOB: See Binary Large Object

BNA: See Burroughs Network Architecture

BPS: Bits per second refer to transfer speeds through communication media.



BURROUGHS NETWORK ARCHITECTURE (BNA): Burroughs' answer to IBM's SNA protocol suite.

C

CAD/CAM: Computer Aided Design/Computer Aided Manufacturing. The usage of computer graphics in engineering and design. This is a very significant part of computer graphics today and yet is being overshadowed by general business graphics and a variety of other graphics applications.

CALL-LEVEL INTERFACE (CLI): An application that lets programs communicate with a database using function calls instead of SQL statements operates on the CLI level of the OSI model.

CDDI: See Copper Distributed Data Interface

CENTRAL PROCESSING UNIT (CPU): The processing heart of a PC, its power base. One or more CPU chips can exist in more advanced configurations. The CPU restricts the OS that can run on the PC as OSs have to be written around particular configurations.

CHECKPOINT: The point at which recent updates stored within cache memory are written to disk is called a checkpoint. Most engines can be configured to perform checkpoints at specific time intervals.

CLI: See Call-Level Interface

CLIENT: A process running on a user workstation, that may request services from another process, which could be located on the same workstation, on a server in a LAN environment, or on a mainframe half way across the world.

CLIENT/SERVER: A form of distributed processing in which a client can request a server to perform tasks for it, and hence relieve the client of unnecessary processing. For example in a database environment the client could be a workstation requesting information about a part database. The server could be a mid range computer housing the parts database, which receives the request for information, processes it, and then sends the results back to the client. The alternative is the situation in which the client has to do all the work itself, by reading in the entire parts database and performing the search itself.

CMIP: See Common Management Information Protocol

CMIP OVER LOGICAL LINK CONTROL (CMOL): A network-independent version of CMIP co-developed by IBM and 3Com. While CMOL can run on any network regardless of transport, it can't pass through routers, limiting its usefulness to LANs.

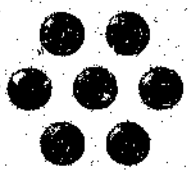
CMIP OVER TCP/IP (CMOT): A version of CMIP that runs on TCP/IP networks.

CMOL: See CMIP over Logical Link Control

CMOT: See CMIP over TCP/IP

COLLISION: Result of multiple attempts to transmit at the same time on a multiple access medium. Usually all colliding transmissions wipe each other out and require re transmission.

CO-OPERATIVE PROCESSING: A form of distributed processing in which multiple different architectures, operating systems, networks, all combine into a single large computer.



COMMIT: A database statement that is used to indicate that the current transaction is accepted and can be placed into the database in its entirety. The opposite of this is rollback.

COMMON MANAGEMENT INFORMATION PROTOCOL (CMIP): A hierarchical, secure management protocol; an alternative to SNMP.

CONCURRENCY: The ability to do more than one thing at once. Within the database context this refers to multiple users accessing the same database, tables, and records at the same time. Controlled through locks.

COPPER DISTRIBUTED DATA INTERFACE (CDDI): A standard for co-axial cabling configuration.

COST-BASED OPTIMISATION: This query optimisation technique utilises statistics gathered on the database to estimate how 'costly' (in terms of time) possible solutions to a query would be, and then chooses the least costly method for execution.

CPU: See Central Processing Unit

D

DAS: See Data Acquisition System

DASD: See Direct Access Storage Device

DATA ACQUISITION SYSTEM (DAS): A front-end to a database server used to port information into the database. This may be a software and hardware gateway, else an MMI

DATA DEFINITION LANGUAGE (DDL): The language used to define the data models used within the database itself--not necessarily the models that the user sees (prettified to give understandable names and layouts).

DATA DICTIONARY: A description of all of the data elements that are used by all of the firm's computer programs. The description includes the data element name, the type of data, the size, how each element is used, to name a few parts.

DATA MANIPULATION LANGUAGE (DML): The syntax that is incorporated into a program to process data retrieved from a database.

DATABASE: A direct access file used to store information in a file analogous to a table. Data is managed as a unit independently of processes that access it.

DATABASE MANAGEMENT SYSTEM (DBMS): The compound software engine that handles the storage, maintenance and retrieval of, and is responsible for controlling access to, and security and integrity of data in a set of databases.

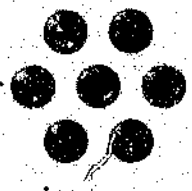
DBMS: See Database Management System

DCEW: See Direct Cost Ex Works

DDBMS: See Distributed Database Management System

DDL: See Data Definition Language

DEAD PARROT (DP): See Parrot, dead



DEADLOCK: A system lock that may arise from two users simultaneously querying a database, resulting in a repetitive cycle of clashes and simultaneous retries.

DECLARATIVE INTEGRITY: With declarative RI, referential integrity is enforced by the use of primary and foreign keys, which are stored within the database tables themselves and describe the relationships within the database.

DECNET: DEC's answer to IBM's SNA protocol suite. This layered architecture is sometimes called DNA.

DESKTOP MANAGEMENT INTERFACE (DMI): A specification, from the DMTF, and aimed at creating a standard management agent that can gather from data any DOS-based PC. At a minimum, DMI will identify the manufacturer, name, version, serial number (if appropriate), and installation date and size of a component.

DESKTOP MANAGEMENT TASK FORCE (DMTF): A coalition of companies led by Intel, and including Novell and Microsoft, with the charter of creating a standard method to manage networked desktop computers and applications.

DIRECT ACCESS STORAGE DEVICE (DASD): A secondary storage unit that has the capability of sending the access mechanism directly to the location where data may be written or read.

DIRECT COST EX WORKS (DCEW): The price of a commodity as it leaves the factory, based on raw materials, direct costs (labour, machinery, comsumptions, etc.) but before business costs such as Sales and Marketing are factored in.

DISASTER RECOVERY: The process of protecting the hardware, software, data, processes and communications capabilities of the information systems and of thereby ensuring business continuity in the event of a major disaster.

DISTRIBUTED DATABASE MANAGEMENT SYSTEM (DDBMS): A DBMS capable of handling files distributed across several PCs, possibly even across several LANs.

DISTRIBUTED DATABASES: Within a distributed computing environment there is still a potential need to store enterprise-wide data, so that users see the data as a single united whole even though it is physically stored across multiple different distributed computers, file servers or database servers. In general there are two ways to split a large database across multiple sites, firstly by deciding which records are to be stored at which location (horizontal distribution) or alternatively deciding to store parts of each record at one location and other details elsewhere (vertical distribution).

DISTRIBUTED MANAGEMENT ENVIRONMENT (DME): A network management protocol that will allow both SNMP and CMIP to exist within one framework. Proposed by OSF in 1991, it is still only partially defined.

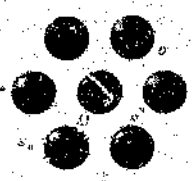
DISTRIBUTED PROCESSING: This is the concept of allowing multiple computers to execute a single process or computation. This could be achieved by many different computers working together to solve a particular computation or by one computer directing another to do a part of its work. This assumes that the process can actually be split into smaller parts. See also co-operative processing.

DME: See Distributed Management Environment

DMI: See Desktop Management Interface

DML: See Data Manipulation Language

DMTF: See Desktop Management Task Force



DNA: Digital Network Architecture. See DECnet.

DoD: Layered networking architecture developed by the US Department of Defense--largely communications control oriented.

DOS: Disk Operating System pioneered by Microsoft for IBM, now forms that largest OS base in desktop PCs.

DOWNCOSTING: The process of moving from one IT infrastructure to a functionally equivalent one at a lower overall cost.

DOWNSIZING: The process of moving an information system from a larger computer to a smaller one or a set of smaller ones. See rightsizing.

DP: See Dead Parrot

E

EIS: See Executive Information System

ENTERPRISE DIRECTORY: A global naming structure in which every user can be assigned a name that is known throughout the organisation and beyond, so that, no matter what the computer structure, it is possible to find each user, and never to get confused between two users.

ENVIRONMENT: Also called software environments, this means the range of software that underlies the users operation and includes the operating system, the network operating system and perhaps even the database management system and the user interface.

ETHERNET: A laboratory project originated by Xerox in 1974, this has become the best known LAN. As of late 1988, about 75% of all installed LANs used the Ethernet protocol--a system of multiple access in which any station transmits when ready, determines whether collision has occurred, then re transmits if necessary.

EXECUTIVE INFORMATION SYSTEM (EIS): A more sophisticated MIS, usually without DAS nor IS capabilities, but with extended reporting and analysis facilities used to 'drill down' into data.

EXPLICIT RELATIONSHIP: A logical link between records and files of a database accomplished by incorporating link fields into the records.

F

FDDI: See Fiber Distributed Data Interface

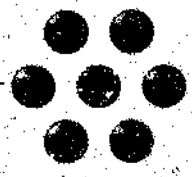
FFDBMS: See Flat File Database Management System

FIBER DISTRIBUTED DATA INTERFACE (FDDI): A standard for fibre optical cabling configuration.

FIELD: A portion of a record reserved for a single data element.

FILE: A group of records relating to a particular subject.

FILE SERVER: The PC responsible for common storage on a network, and upon which the NOS usually resides.



FLAT FILE DATABASE MANAGEMENT SYSTEM (FFDBMS): A DBMS that does not support links between files.

FOURTH GENERATION LANGUAGE (4GL): A relatively new breed of software intended to facilitate end-user computing. The names natural and non-procedural are used because of the user-friendly syntax and because it is not necessary that the processes be performed in a particular order as with a programming language. New 4GLs are very graphically oriented to expand the philosophy further.

G, H

GATEWAY: Relay operating at a layer higher than network layer.

GRANULARITY: The level of locking on a database. This is typically on the database, file, record or field level.

HMI: See Hub Management Interface

HUB MANAGEMENT INTERFACE (HMI): A specification that integrates the wiring hub, the center of many Ethernet and Token-Ring cabling schemes, and the file server into a common device.

I

IETF: See Internet Engineering Task Force

IMPLICIT RELATIONSHIP: The linking of records and files in a database without the addition of special fields. The relationships are implied using the contents of fields already in the records, accomplished through the use of a relational calculus.

INFORMATION STORAGE (IS): The module of an IT/IS system that is responsible for data storage--the DBMS

INFORMATION SYSTEMS (IS): For the purposes of this report, a factory level implementation of IT, but sometimes used to refer to the software side of IT.

INFORMATION TECHNOLOGY (IT): The philosophy of recording, implementing and reporting transactions with the aid of computing systems.

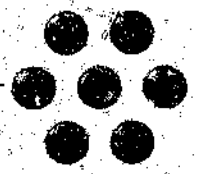
INSTANTIATION: A running instance of a program, or its code, on a computer platform.

INTEGRATE: The process of linking two information systems, so that they both have access to the same information, possibly in separate data pools, but more realistically in a common data pool. Information is read and written directly to the pool(s) by each system.

INTEGRATED SERVICES DIGITAL NETWORK (ISDN): A data communications circuit that has the ability to transmit voice, data, and image signals.

INTEGRITY: A system characteristic that enables it to accomplish what was intended. In the database context this refers to extent to which data is correct and accurate, and to which it is protected against any attempt to introduce incorrect and inaccurate data. Validation rules support integrity, often referred to as consistency.

INTERFACE: The process of linking two information systems, so that they both have access to the same information, but in different data pools. Each application reads and writes to



its own pool, with transfer of information between the pools (updating) taking place at regular intervals.

INTERNATIONAL STANDARD'S ORGANISATION (ISO): The standards-making body responsible for OSI, a set of communications standards aimed at global interoperability consisting of 75 member countries.

INTERNET: A large network community in the US who share mailing and information resource facilities through a common communications protocol, IP.

INTERNET ENGINEERING TASK FORCE (IETF): The standards-making body for the Internet; the arbiter of TCP/IP standards, including SNMP.

INTERNET PACKET EXCHANGE (IPX): A communication protocol used, primarily, by Novell's NetWare NOS.

INTERNET PROTOCOL (IP): The communications protocol used primarily by the Internet community (see TCP/IP).

IP: See Internet Protocol

IPX: See Internet Packet Exchange

IS: See Information Systems or Information Storage

ISDN: See Integrated Services Digital Network

ISO: See International Standards Organisation

IT: See Information Technology

J, K, L

JOIN: An operation in which two tables are combined on common attributes. If there are no common attributes, this reduces to a Cartesian product--typically very load intensive. Joins are not typically available in fully compliant SQL, but many vendor specific adaptations cater for them.

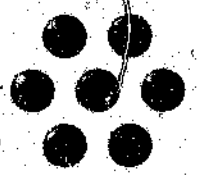
LAN: See Local Area Network

LOCAL AREA NETWORK (LAN): A network of computers that are connected using circuitry owned by the firm. A common target platform for a downsized solution to a business information system.

LOCKS: Locks are placed onto database items in use by one user or process to ensure that other users or processes do not create inconsistent data. Locks may be placed on the database, table, or record, with a trade-off between performance of operations and the accessibility of data (database lock means high performance but inaccessibility for other users). Some systems can vary the level of locking (granularity) as required.

M

MAINFRAME: A large to very large computer, often consisting of many central processors, and many support processors, which is able to handle very large amounts of data, transactions and users in a single environment



MAN MACHINE INTERFACE (MMI): A terminal of input device used to 'interface' computers to personnel.

MANAGEMENT INFORMATION BASE (MIB): A database that defines what information can be gathered and what aspects can be controlled for a network device.

MANAGEMENT INFORMATION FORMAT (MIF): The DMTFs equivalent of SNMP's MIB.

MANAGEMENT INFORMATION SYSTEM (MIS): The front-end client to an information system used for accessing the database (converting data to information) and sometimes used for controlling input into the database (DAS). Standalone MISs often have their own IS module.

MANAGEMENT INFORMATION SYSTEM (MIS): A system that provides the manager with information for decision making. It is usually the front-end to an DBMS and converts data into information.

MIB: See Management Information Base

MID-RANGE: A generic term for a group of computing systems, by many manufacturers, which are too small to be called mainframes, and generally far too large to be called PCs.

MIF: See Management Information Format

MIGRATION: This is the process of moving an application from one platform to another. This typically involves re-design and re-programming to some extent as there are few languages which support multi-platform portability, and most systems on mainframes were built before the advent of these languages. See porting.

MIS: See Management Information System

MIS: See Management Information System

MISSION CRITICAL: A system that is accessed by multiple users and must be capable of producing up-to-date real-time results.

MMI: See Man Machine Interface

MULTI-VERSIONING: Multi-versioning is a record locking solution that uses prior snapshots of the database for read-only access, so users are not locked out during updates.

N

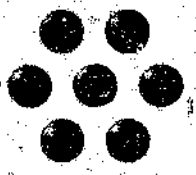
NETWARE LOADABLE MODULE (NLM): A program that can run under Novell's multi-tasking NetWare environment.

NETWORK MANAGEMENT CONSOLE: The administrator's window into the network. The console is the control panel through which network devices, such as routers, are monitored, configured, and reset.

NETWORK MANAGEMENT PLATFORM: A set of management services upon which specific management applications, such as consoles, can be built.

NETWORK OPERATING SYSTEM (NOS): The system that resides on the network file server and controls communication, routing and device contention across the network.

NLM: See NetWare Loadable Module



NOS: See Network Operating System

O

OBJECT: A distinct type of management information or variable for a network component. For example, the on/off status of a router is an object. In SNMP, MIB objects can be read (Get objects) or read and written (Get and Set objects). Operators can manipulate these objects to gather information and change configurations.

OBJECT ORIENTED DATABASE MANAGEMENT SYSTEM (OODBMS): A DBMS with the capacity to store BLOBs as a data type.

ODAPI: See Open Database Application Programming Interface

ODBC: See Open Database Connectivity

OLTP: See On Line Transaction Processing

ON LINE TRANSACTION PROCESSING (OLTP): The ability to handle large to very large volumes of data transactions coming into a system, to process them and deal with them at every required level. This is one area in which mainframes still vastly out perform any computers.

OODBMS: See Object Oriented Database Management System

OPEN DATABASE APPLICATION PROGRAMMING INTERFACE (ODBC): Borland's standardised API for database access, proposed in answer to Microsoft's ODBC and touting object oriented capabilities.

OPEN DATABASE CONNECTIVITY (ODBC): Microsoft's standardised API for database access.

OPEN SYSTEM: An operating system which can run across a number of different platforms. By far the most well-known open system is Unix. The term open system is also used to mean an, application or environment in which the user is freed from any knowledge of the platform.

OPEN SYSTEMS FOUNDATION (OSF): A Cambridge, Massachusetts-based nonprofit organization responsible for assembling the DME specification.

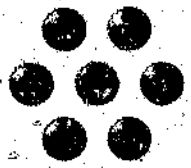
OPEN SYSTEMS INTERCONNECTION (OSI): The suite of communications standards mandated by ISO and the U.S. government. CMIP is part of this suite.

OPERATING SYSTEM (OS): An operating system is the base upon which PC programs run and provides a basic set of instructions and functions that make the process easier for the application programmer and the user. The operating system also serves as an interface to the underlying hardware, providing a "level playing field" for everyone who wants to use the computer system.

OS: See Operating System

OSF: See Open Systems Foundation

OSI: See Open Systems Interconnection



P, Q

PC: See Personal Computer

PERSONAL COMPUTER (PC): A desktop computer, smaller than a mini yet powerful enough to perform server functions in today's environment.

PLATFORM: This term generally means the hardware on which the computing systems reside, but it can be extended to mean the operating systems as well.

PORTING: The movement of an application from one platform to another without any significant changes. Often used to mean transferring the application system to and from Unix, or between its many versions.

PRIVATE MIB: Also called extended MIB. A standard MIB to which proprietary extensions have been added in order to monitor or control network device functions not included in the standard MIB specification.

PROCEDURAL INTEGRITY: With procedural RI, referential integrity is enforced by the use of predefined procedures that are automatically executed when certain types of changes are made. Since the integrity rules are stored separately from the database tables, procedural integrity is not the best approach, but it is preferable to relying on the front-end application to enforce RI.

PROPRIETARY SYSTEM: An operating system which is vendor-specific, typically optimsed for the architecture of the underlying hardware, rather than being open, in the sense of being standardised across a number of platforms.

PROTOCOL: The standards that are used in interfacing datacom equipment an circuitry, a 'common language'.

R

RAID: See Redundant Arrays of Inexpensive Disks

RAM: Random Access Memory is the volatile memory that a within which computer may store data and the process it is currently running. There is a very real 640kb limit in DOS based PCs that makes the loading of numerous driver applications problematic.

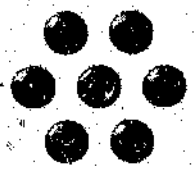
RDBMS: See Relational Database Management System

RECORD LOCKING: In a multi-user environment where the database is constantly being updated, it is difficult to ensure that a user's view of data is consistent--that is, that all related transactions have been processed. Record locking is one solution, but the price is that other users must wait to gain access to 'locked' data (data that another user is currently using).

RECURSION: See Recursion

REDUNDANT ARRAYS OF INEXPENSIVE DISKS (RAID): A system of providing redundancy to the storage media in such a way that reliable hardware backup is provided. RAID5 refers to the latest specification of this setup.

REFERENTIAL INTEGRITY (RI): RI means internal consistency within the database; that is, no records that refer to missing records (termed 'orphans'). An example of a violation of referential integrity would be a database consisting of both a customer table and an order table, where the customer record for a particular order is missing. RI is maintained



through a set of rules, which may be located in the client-side application (least reliable), in a stored procedure on the database server (more reliable), or via a coding scheme within the tables themselves (most reliable).

Relational Database Management System (RDBMS): A DBMS with implicit relationships between files.

Reliability: The ability of a system to provide a service in terms of up time.

Remote Monitoring MIB: An extended MIB that delivers statistics about data packets and network traffic to an SNMP console.

Resilience: The stability of systems or network. Its ability to defend itself against faults.

Resizing: The process of changing the size of the hardware or software platforms in order to optimise the benefits of the information systems of the organisation.

RI: See Referential Integrity

Rightsizing: The process of determining, for a particular business application or information system, the correct platform and environment for the application to use, in order to optimise the overall business goals.

Roll forward: Roll forward is a feature used for recovery from system crashes. An instance backup (a snapshot of the database) is restored, and then all subsequent changes to the database are re-applied utilising the transaction log (a description of all changes to the database).

Rollback: Rollback allows transactions that were only partially completed because of a hardware failure to be discarded, thus restoring the database to its previous (consistent) state. Partial rollback, through the use of save points, can be useful in complex transactions where it is not desirable to discard the entire transaction.

S

Samesizing: The decision to keep the same sized computer to manage the business operations.

SCADA: See Supervisory Control and Data Acquisition

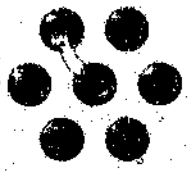
Security: The ability of a system to withstand threats from unauthorised users and processes.

Sequenced Packet Exchange (SPX): A communication protocol used, primarily, by Novell's NetWare NOS.

Server: A computing device that is able to perform operations on request, typically on behalf of a community of users and processes. For example a file server, database server, printer server or communications server

Simple Network Management Protocol (SNMP): An IETF-defined protocol that runs natively on TCP/IP networks. Considered the de facto standard for network management, it is used to monitor the status of devices, but because of its lack of security, is rarely used to control network devices.

SMI: See Structure of Management Information



SMP: See Symmetric Multiprocessing

SNA: See Systems Network Architecture

SNMP: See Simple Network Management Protocol

SNMP VERSION 2 (SNMPII): The next generation of SNMP. It will add security (encryption and authentication), support a hierarchical management scheme, and run on network transports other than TCP/IP, including AppleTalk, IPX, and OSI.

SNMPII: See SNMP version 2

SPX: See Sequenced Packet Exchange

SQL--PRONOUNCED 'SEQUEL': See Structured Query Language

STORED PROCEDURES: Precompiled SQL statements that are stored on the SQL database server. A client executes a stored procedure by issuing an EXECUTE PROCEDURE_NAME statement.

STRUCTURE OF MANAGEMENT INFORMATION (SMI): An SMI specifies the rules used to define the information in a managed object so it can be monitored and manipulated through the management protocol. The SMI is analogous to a programming language that a programmer uses to declare data structures and to permit operations that can be performed on those data structures.

STRUCTURED QUERY LANGUAGE (SQL): Originally developed as a way of making queries on a database easy, it is now used as a complete database language, incorporating a data dictionary and DDL. Since the programmer specified what is wanted and not how, this is considered to be a very high level language as in a 4GL.

SUPERVISORY CONTROL AND DATA ACQUISITION (SCADA): A plant control interface used to report the status of the plant to operators, used by the operators to effect automatic controls, and used to control the plant through internal procedures.

SYMMETRIC MULTIPROCESSING (SMP): A form of multiprocessing, using either standard or custom processing components which share the load by behaving in a similar manner.

SYNTAX-BASED OPTIMISATION: With this query optimisation technique, generic rules are used to reword a query so it will execute more quickly. Although, the optimised solution requires programmers' skill and may not be as fast as which cost-based optimisation would provide, its advantage is low overhead.

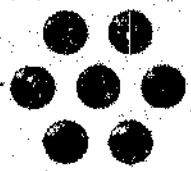
SYSTEMS INTEGRATION: The ability to get systems (hardware) connected together so that the end-user does not see them as separate entities.

SYSTEMS NETWORK ARCHITECTURE (SNA): IBM's protocol suite that pioneered the way to standardisation of protocols.

T

TCP/IP: See Transmission Control Protocol/Internet Protocol

TELNET: A part of the TCP/IP protocol suite that allows managers to control network devices, such as routers, remotely.



THIRD GENERATION LANGUAGE (3GL): A language with low-level capabilities for accessing various database formats. Often a high-level language such as C or COBOL.

TOKEN-RING: A network configuration (topology) in which the LAN forms a ring onto which each workstation is attached. Communication is controlled through a 'token' that is passed between machines and 'allows' them to 'speak' when they have control.

TRANSMISSION CONTROL PROTOCOL/INTERNET PROTOCOL (TCP/IP): Developed by the U.S. Department of Defense to inter network disparate computers, TCP/IP is the most popular protocol for multi-platform networks. Protocol operates on equivalent of OSI model transport layer.

TRIGGERS: Similar to stored procedures but execute in response to a database event.

TWO-PHASE COMMIT: Used in distributed database scenarios, this protocol first checks to see if all of the systems involved in the current transaction are ready either to 'commit' or to 'roll back.' The protocol then passes the appropriate operation to finalise the transaction.

U, V, W

UNINTERRUPTIBLE POWER SUPPLY (UPS): A device which provides continuos power in the event of a power failure.

UPS: See Uninterruptible Power Supply

UPSIZING: The process of combining many smaller computer systems into one larger one, for example combining many individual PCs into a LAN with a central server, or replacing many LAN servers by a mid-range or mainframe server.

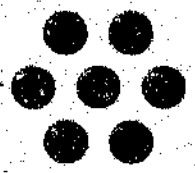
USER EMPOWERMENT: The process of giving more capabilities to the users to create and maintain their own independent information systems, rather than being dictated by a centralised committee or division.

VINES INTERNET PROTOCOL (VIP): Vines' adaptation of IP

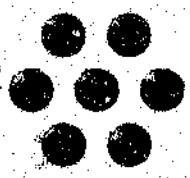
VIP: See Vines Internet Protocol

X, Y, Z

X86: The generic term for Intel's family of 8086 chips found in most DOS based PCs. The x replaces the 2, 3, 4 or 5 designating each progressively more powerful upgrade to the chip set.



REFERENCES



DISCLAIMER

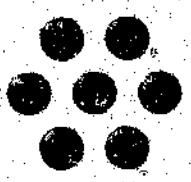
Most references were sourced through Compuserve Africa (the CSIR link through to the US), and from there onto the Internet for extraneous articles. Compuserve provides information through forums where industry experts may be contacted through electronic mail (used for clarification and understanding throughout this report), but more relevantly through an extensive database of periodicals.

The periodicals maintained by Ziff-Davis Publishing Company include almost every magazine and journal published. An emphasis is placed on computing journals, and it is from here that most leading edge information in this report was sourced. Articles uploaded by Ziff-Davis usually precede South African receipt by up to four months, with numerous publications being unavailable here, else industry specific research that is not provided for commercial distribution.

Compuserve facilitates the searching of text on date, publication name, key words, subjects, and several other filters. These were used to ensure the trapping of a wide range of opinions on each subject delved into.

The down-side of such extensive research is that many references are somewhat biased, and whilst an attempt has been made to stick to well recognised sources, the inevitable misinformation will creep in in the form of a failed prediction or two. The author, however, is fairly confident that all recommendations made are based on sound support and numerous positive factors.

Some references below have not been linked to the text as they were used indirectly, or formed part of a summation. Others were used more than once in the text, but have only been linked where most applicable.



DISCLAIMER

Most references were sourced through Compuserve Africa (the CSIR link through to the US), and from there onto the Internet for extraneous articles. Compuserve provides information through forums where industry experts may be contacted through electronic mail (used for clarification and understanding throughout this report), but more relevantly through an extensive database of periodicals.

The periodicals maintained by Ziff-Davis Publishing Company include almost every magazine and journal published. An emphasis is placed on computing journals, and it is from here that most leading edge information in this report was sourced. Articles uploaded by Ziff-Davis usually precede South African receipt by up to four months, with numerous publications being unavailable here, else industry specific research that is not provided for commercial distribution.

Compuserve facilitates the searching of text on date, publication name, key words, subjects, and several other filters. These were used to ensure the trapping of a wide range of opinions on each subject delved into.

The down-side of such extensive research is that many references are somewhat biased, and whilst an attempt has been made to stick to well recognised sources, the inevitable misinformation will creep in in the form of a failed prediction or two. The author, however, is fairly confident that all recommendations made are based on sound support and numerous positive factors.

Some references below have not been linked to the text as they were used indirectly, or formed part of a summation. Others were used more than once in the text, but have only been linked where most applicable.

Author: Barker Christopher Damian.
Name of thesis: Information technology- an information survey into the practical implementation of information technology in a real world environment.

PUBLISHER:
University of the Witwatersrand, Johannesburg
©2015

LEGALNOTICES:

Copyright Notice: All materials on the University of the Witwatersrand, Johannesburg Library website are protected by South African copyright law and may not be distributed, transmitted, displayed or otherwise published in any format, without the prior written permission of the copyright owner.

Disclaimer and Terms of Use: Provided that you maintain all copyright and other notices contained therein, you may download material (one machine readable copy and one print copy per page)for your personal and/or educational non-commercial use only.

The University of the Witwatersrand, Johannesburg, is not responsible for any errors or omissions and excludes any and all liability for any errors in or omissions from the information on the Library website.