# *A Technology Reference Model for Client/Server Software Development.*

by

**RITA CHARLOTTE NIENABER**

Submitted in part fulfilment of the requirements
for the degree of

**MASTER OF SCIENCE**

in the subject

**INFORMATION SYSTEMS**

at the

UNIVERSITY OF SOUTH AFRICA

**SUPERVISOR: PROF A L STEENKAMP**

15 JUNE 1996

## ABSTRACT

In today's highly competitive global economy, information resources representing enterprise-wide information are essential to the survival of an organization. The development of and increase in the use of personal computers and data communication networks are supporting or, in many cases, replacing the traditional computer mainstay of corporations. The client/server model incorporates mainframe programming with desktop applications on personal computers.

The aim of the research is to compile a technology model for the development of client/server software. A comprehensive overview of the individual components of the client/server system is given. The different methodologies, tools and techniques that can be used are reviewed, as well as client/server-specific design issues. The research is intended to create a road map in the form of a Technology Reference Model for Client/Server Software Development.

## KEYWORDS

*Client/Server, Open Systems, Software Development, Software Standards, Interoperability, Object-orientation, Middleware, Groupware, Reference Model, Distributed Systems.*

## ACKNOWLEDGEMENTS

I would like to thank the Foundation for Research Development for their financial assistance in providing the equipment  to realize this dissertation.

A special word of thanks to Professor Lerine Steenkamp, my supervisor, for her guidance, inspiration and advice, for which I am grateful.

My sincere gratitute and appreciation to my husband, Sarel, for his encouragement and assistance throughout the project.

I would also like to thank my daughters, Marie-Louise, Karen and Tanya for their support.

# TABLE OF CONTENTS

# PREFACE

This dissertation has been done in part fulfilment of the requirements of the M.Sc. degree in Information Systems at the University of South Africa. The other part of the degree requirements was the completion of five study modules, which were:

*Object-Orientation:* The module covered object-oriented software development. An object-oriented approach, comprising a life-cycle methodology with modelling and design techniques and notations was studied and implemented as a prototype.

*Software Specification Techniques:* The module covered formal specification techniques proposed in recent years. General principles of software specification and specific techniques were studied. Actual experience was obtained in using these specifications.

*Human-Computer Interaction:* The module covered the study of human cognitive and physical capabilities to the purpose of incorporating this knowledge into the design of technology components. Models to support interface design, including task analysis, dialogue design, on-line help, cognitive models and software engineering notations were studied. Effective implementation and evaluation of interactive systems were considered.

*Open Systems:* A special-topic module which comprised a study of the planning and implementation of open systems. Key issues were identified, reviewed, interpreted and evaluated. The study included the rationale for open systems, concepts and principles, the role of standards as well as a strategy for migrating to open systems.

*Client/Server Application Development:* A special-topic module which provided practical experience in using application development environments such as Microsoft Visual Basic and Access for developing a client/server event-driven application in LANs. The opportunity to define, design, and implement tools for developing and maintaining client/server applications was given.

The research presented in the dissertation is concerned with the area of software development, with special emphasis on a growing new paradigm, namely client/server application development. Client/server application development has emerged as a research field which focuses on this development environment. The study explores the rationale for the

development of client/server applications. Various trends in the industry are identified as drivers for client/server systems.

The purpose of the research is to compile a Technology Reference Model for the development of client/server software. Thus the meta-primitives of client/server systems are identified, as well as the basic components of a typical client/server application. A comprehensive overview of the individual components of the client/server system is given, including key aspects and design issues in the development of a client/server application. The methodologies that can be used are reviewed, and tools and techniques are considered.

The study was executed in accordance with the Object-Oriented Information Systems Engineering Environment (OOISEE) project undertaken by the Department of Computer Science and Information Systems at UNISA, which is aimed at establishing a foundation within which research projects may be undertaken at postgraduate level.

# LIST OF FIGURES

# LIST OF TABLES

# LIST OF ACRONYMS

| | |
|---|---|
| AD/EXPO | Application Development Expo |
| AD/TECH | Application Development Technology |
| ANSI | American National Standards Institute |
| ASC | Accredited Standards Committees |
| API | Application Programming Interfaces |
| ATM | Asynchronous Transfer Mode |
| BIOS | Binary Input Output System |
| BLOB | Binary Large Objects |
| BSD | Berkeley System Distribution |
| CBT | Computer-Based Training |
| CCITT | International Telegraph and Telephone Consultative Committee |
| CDE | Common Desktop Environment |
| CD-ROM | Compact Disc Read Only Memory |
| CICS | Customer Information Control System |
| CMIP | Management Information Protocol |
| COM | Component Object Model |
| CORBA | Common Object Request Broker |
| COSE | Common Open Software Environment |
| DCE/RPC | Distributed Computing Environment/Remote Procedures Call |
| DAL | Data Access Language |
| DARPA | Defense Advanced Research Projects Agency. |
| DBMS | Database Management System |
| DCE | Distributed Computing Environment |
| DDE | Dynamic Data Exchange |

| | |
|---|---|
| DML | Database Manipulation Language |
| DRAM | Dynamic Read Only Memory |
| DRDA | Distributed Relational Database Architecture |
| DSS | Decision Support System |
| DNA | Digital Network Architecture |
| EDA/SQL | Enterprise Data Access/SQL |
| EIS | Executive Information System |
| EPSS | Electronic performance support systems |
| FDDI | Fibre distributed data interface |
| DSOM | Distributed System Object Model |
| GUI | Graphical User Interface |
| I-CASE | Integrated CASE |
| IDAPI | Integrated Database API |
| IDE | Integrated Development Environment |
| IEC | International Electrotechnical Commission |
| IEEE | Computer Society of the Instituted of Electrical and Electronic |
| IPC | Inter Process Communication |
| IPX | Internet Packet Exchange |
| ISO | International Standards Organization |
| IT | Information Technology |
| JAD | Joint Application Design |
| LAN | Local Area Network |
| MAN | Metropolitan Area Network |
| MB | Megabyte |
| MIS | Management Information System |
| MOM | Message-Oriented Middleware |

| | |
|---|---|
| MSDOS | Microsoft Disk Operating System |
| MHz | Mega Hertz |
| NFS | Network File System |
| NGPM | Next Generation Process Model |
| ODBC | Open Database Connectivity |
| OLE | Object Linking and Embedding |
| OMA | Object Management Architecture |
| OMG | Object Management Group |
| OOP | Object-Oriented Programming |
| OOISEE | Object-Oriented Information Systems Software Engineering Environment |
| ORB | Object Request Broker |
| OSF | Open Systems Foundation |
| OSI | Open Systems Interconnection |
| PC | Personal Computer |
| RAD | Rapid Application Development |
| RAID | Redundant Array of Inexpensive Disks |
| RAM | Read Only Memory |
| RISC | Reduced Instruction Set Computer |
| RPC | Remote Procedure Calls |
| SAG | SQL Access Group |
| SASQ | South African Software Quality Group |
| SCSI | Small Computer Systems Interface |
| SNMP | Simple Network Management Protocol |
| SMP | Symmetric Multiprocessing |
| SOM | Systems Object Model |
| SONET | Synchronous Optical Network |

| | |
|---|---|
| SPX | Sequenced Packet Exchange |
| SQL | Standard Query Language |
| TCP/IP | Transmission Control Protocol/Internet Protocol |
| UPS | Uninterruptable Power Supply |
| USL | Unix Systems Laboratories |
| VUE | Visual User Environment. |
| WAN | Wide Area Network |
| WIMP | Windows, Icons, Menu's and Pointers |
| WWW | World Wide Web |

# GLOSSARY OF TERMS

**Application program** Software designed to form tasks such as word processing, billing or inventory control.

**Application program interface (API)** Standard interface for interworking applications and utilities.

**Client** An information or service requester, usually on a network and often an application on a PC or workstation..

**Groupware** Groupware is the term for applications written to support the collaboration of several users.

**Interoperability** The ability to interconnect computers that use different operating systems to be able to exchange information.

**Information systems** An information system can be seen as an arrangement of interdependent human and machine components that interact to support the operational, managerial, and decision-making information needs of a business.

**Information technology** The technological components comprising information systems.

**Information technology strategic planning** is the process of describing how to use information technology to contribute to the achievement of the corporate strategy and gain competitive advantage.

**Methodology** incorporates the various processes, methods and techniques used in executing a task.

**Method** incorporates a specific way of executing a task.

**Middleware** is software with API interfaces, protocols and call libraries that enables application fragments on different platforms to communicate over a network, as well as to interface with databases.

**Multimedia** Sophisticated software that combines electronic media, such as basic text and graphics with animation, video, music and voice.

**Multiprocessor** A computer made up of several processors in parallel.

**Open systems** are based on a set of standard relationships that enable different computers, subsystems, applications and system software to operate together.

**Object-orientation** is a multilayered architecture of reusable, interchangeable software components analogous to the multilayered architecture of hardware engineering.

**Parallel processing** A form of processing in which several programs or tasks are processed in parallel, on physically different processors.

**Protocol** A set of rules for the exchange of data between a terminal and a computer or between computers.

**Portability** The capability of moving software applications from one operating environment to another, at minimal cost.

**Standards** A point of reference.

**Strategic planning** An orderly means of assessing the information needs of an organization and defining the systems and databases that will best satisfy those needs.

**Scalability** the need to change the overall capacity of an application.

**Technology plan** The technological realization of the application plan, corporate strategy and the information requirements in the organization.

**Uninterruptable power supply (UPS)** A continually charging battery that powers your computer should the main source of power fail.

# TRADEMARKS

All trademarks are those of their respective owners, including:

Adobe and PostScript are registered trademarks of Adobe Systems Inc.

Apple and Macintosh are registered trademarks of Apple Computers.

Banyard Vines is a trademark of Banyan Systems, Inc.

cc:Mail is a trademark of cc:Mail, Inc, a subsidiary of Lotus Development Corporation.

Crossroads is a trademark of Crossroad Systems Inc.

dBASE, dBASEIII+, dBASEIV, dBASEV, Delphi, Interbase, Client/Server DBMS, Pascal, QuattroPro, ObjectVision and Paradox are registered trademarks of Borland International Inc.

DEC, DECNet, DECWindows, DNA, LinkWorks, ULTRIX, VAX, Alpha and VMS are trademarks of Digital Equipment Corporation.

DBGateway is a trademark of Mercury Interactive Corporation.

Data Ease is a registered trademark of Data Ease Corporation.

Easel and Easel/2 are registered trademarks of Easel Corporation.

Ellipse is a trademark of Cooperative Solutions.

HP, HP-UX, NewWave, PRISM, Allbase, OpenView and VUE Visual User Environment is registered trademarks of Hewlett-Packard Company.

Hyper STAR is a trademark of Constellation

IBM is a registered trademark and AIX, AS/400, APPN, CICS, DB/2, DB2/2, DisplayWrite, DRDA, DB2/6000, Delivery Manager, DSOM, ESA, IMS/VS, IMS/DC, LAN Server, LANFocus Management/2, Micro Channel Architecture, MCA, MVS, NetView, OpenDoc, OS/2, OS/2 Communication Manager, RS/6000, SOM, SQL/DS, System Network Architecture, SystemView, SAA, VM, VSAM, VTAM and Workplace Shell, are trademarks of International Business Machines Corporation.

Intel is a registered trademark and Pentium, 80286, 80386 and 80486 are trademarks of Intel Corporation.

Ingres, Ingres 4GL, Ingres-Star are trademarks of Ingres Corporation.

Informix, Informix OnLine, InformixStar are trademarks of Informix, Inc.

APS, PVCS are a trademarks of Intersolv.

Lotus, 1-2-3, Freelance, Lotus Manuscript, Lotus Notes, Samna, Symphony, and DIF are registered trademarks of Lotus Development Corporation.

Lifespan is a trademark of SEMA Corporation.

MIPS is a trademark of MIPS Corporation.

MS-DOS and SQL Server and are registereds trademarks of Microsoft Corporation. Chicago, Cairo, OLE, ODBC, Windows, Windows NT, Windows for Workgroups, LAN Manager, Visual Basic, Access, MSWord, Excel and Flow Mark, Multimedia Extensions are trademarks of Microsoft Corporation.

Mozart is a registered trademark of Mozart Systems Inc.

NCR and Top End are trademarks of NCR Corporation.

NeXTSTEP is a trademark of NeXT Corporation.

Novell, Netware, Unixware, NMS and NetWare 3.x and 4.x are registered trademarks of Novell, Inc.

OSF, OSF/1, DCE, DME and Motif are trademarks of Open Software Foundation.

Oracle, OracleCASE, Oracle SQL Forms and Oracle Toolkit are a registered trademarks of Oracle Corporation.

OMW, Kappa, CommManager, Forte are registered trademarks of Intellicorp.

OpenV OPSS is a trademark of OpenVision.

POSIX is a trademark of Institute of Electrical and Electronics Engineers Inc.

PowerBuider is a trademark of PowerSoft Inc.

SCO is a trademark of Santa Cruz Operations.

Sun is a registered trademark and NCA, Open Look, Network File System, NFS, NeWS, NeWS Development Environment, NDE, SunOS, SUN SPARC, SOLARIS, SunView, and Xview are trademarks of Sun Microsystem, Inc.

Smalltalk is a trademark of Xerox Corporation.

Sybase is a registered trademark and SQL/Server, Open Server, Net/Gateway are trademarks of Sybase Incorporation.

SQLBase, Gupta, SQLWindows are trademarks of Gupta Technologies.

Timbuktu is a trademark of Farralon Computing.

Tuxedo, Rhapsody, StarServer E, and STARLAN are trademarks of AT&T.

UNIX, COSE are registered trademarks of UNIX Systems Laboratories.

X/Open is a trademark of X/Open Corporation.

X Window System, X11, Kerberos are trademarks of the Massachusetts Institute of Technology.

Workflow is a trademark of Action Tech Corporation.
WordPerfect is a registered trademark of WordPerfect

# CHAPTER 1

# Introduction To The Research Topic

## 1.1 Introduction

In recent years, information technology and information systems have been changing rapidly. Information, as a major resource of an organization, must be readily accessible and available to all employees at all times. At present, the focus is not only on the information usage of one end user, but on information accessibility for enterprise-wide use. In today's highly competitive global economy, well-structured information resources are essential to the survival of an organization and can be used to give the organization a competitive advantage over competitors in the market.

Technological development plays a major role in the changing nature of Information Systems. The computer industry is continuously experimenting and hardware components are replaced with smaller, faster and cheaper components. The traditional computer mainstay of corporations for transaction processing used to be the central mainframe, managed by a professional MIS department. However, with the development of the personal computer and data communications, computer systems were interconnected and remote processing operations became a reality (Cashin, 1993). During the 1980s networking gave rise to distributed computing.

The capabilities of personal and microcomputers provided the real impetus for widespread distributed computing in the 1980s. The personal computer brought independent processing power to the desktop of every employee in an organization and increased the number of computing systems by orders of magnitude. At the outset, personal computer computing was simply an extension of large-scale computing and was used in stand-alone mode. Advances in communications technology and networking resulted in the use of various forms of personal computers instead of simple terminals. However, the availability of the personal computer to all end users led to the growing sophistication of end users and the redistribution of computer users in the enterprise. Improvement in user interfaces, i.e. graphic user interfaces (GUIs), the advantages of the distribution of resources, combined with the power of the personal computer, and the wide variety and availability of desktop applications gave rise to an increase in the use of personal computers, connected in a local area network configuration. Increasing attention has also been focused on the applications used in these environments and how they may be redistributed in the environment to achieve optimal use of user expertise, basic hardware technology and resources.

## 1.2 The Problem and its Relevance

Information technology comprises the technological components of an information system. Existing information systems seldom satisfy users' requirements, as they are often delivered late, exceed the budget, or do not meet performance and functionality requirements. The information system is used as a resource in the business strategy of the organization. In order to accommodate changes in business strategy, the structure of the information system must also be flexible and able to accommodate changes in the enterprise.

Currently, four factors are changing the business environment namely globalisation, computer technology and data communication advances, management and information economy (Bergen, 1994). Each of these factors is explained briefly.

**Globalisation:** Boundaries to commerce are opening up and capital spending is no longer driven by business cycle spending but by global considerations. Technology supplements this by the growing utilization of information. The "information highway" that is currently under construction has the potential of materially enhancing our lives. It is predicted that the next few years will see a massive growth in the network resources available to business and home. The Internet with services such as the World Wide Web (WWW) provides a protocol with attractive multimedia interface to allow relatively unskilled people to access information on servers all over the world. Information must be restructured to support this new environment. The information highway requires an information architecture that can handle all forms of information including text, tables, images, high-fidelity audio and full-motion video.

**Computer technology and data communication advances:** It was stated that in 1993 30% of all homes in the US had a PC, 40% of computers in the US were connected to a network, and data traffic over phone lines were growing by 30% per annum. Computers and networks are essential means of improving productivity in businesses and of the individual. The changing role of the information system brings changes to the structuring of information. New technologies are also placing computer functionality on the desk top of every employee in the enterprise. In a typical organization of 500 users there are probably 300 or more user-built applications running. Information is accessible to more end users and control, security measures, and data management over these applications are becoming more important. Legacy systems that perform a central function in organizations cannot just be discarded, such as old mainframe systems. This also implies new security measures, network protocol and management measures.

**Management:** Changes in management are essential. The profile of the organization seems to be continually changing, as is information technology. Management has to recognize and utilize these changes to maintain a competitive advantage over its competitors. Distributed computing will also redistribute power and usage.

**Information economy**: Business investments in computers and telecommunications equipment are growing at an astounding rate. It was reported in 1993 that in 1991 the investment in computers and telecommunications equipment for the first time exceeded capital spending for industrial, construction and other equipment. Businesses have to invest in the most appropriate technology, systems and telecommunications to give them a competitive advantage. Overall, the rapidly changing business environment of today requires the information system to provide adaptability, flexibility and dependability.

## 1.3 State of the Art

Isolated desktop solutions used in stand-alone personal computers are not sufficient to meet business requirements any more. The user and the organization need an integrated view of the information in the organization. A personal computer network may facilitate the communication but still does not provide sufficient performance and flexibility in database environments.

There is currently a need for interoperable, distributed tools and standards upon which architectures can be built, based on the power of the PC and the network. Distributed systems comprising a mainframe and workstations do not comply with the data accessing and performance demands of large organizations.

There is also a tendency for document-based instead of application-based development. Integrated environments will provide the solution to this need.

Using only mainframes or only stand-alone microcomputers both have their disadvantages:

- The mainframe can store and handle large volumes of information, but traditionally data was stored as separate files and did not represent an integrated view of enterprise-wide information.
- Each organization has a plethora of legacy systems that will have to be converted or archived. These old systems cannot be discarded as they contain valuable information.
- Using stand-alone PCs in an organization does not provide the organization with interoperability of data, data sharing, it results in data redundancy, duplicate systems and data residing throughout the organization. Inconsistencies may also prevail, as updating of the same records may be needed in more than one department.
- Programmer productivity is often low as duplicate development, coding, implementation and maintenance take place throughout the organization.

## 1.4 Why Client/Server?

Robertson (1994) states that the issue today is: "The ability to comprehend the full potential of the technology and exploit these capabilities for long-term competitive advantages! "

Every decade has its rising technologies. The 1970s had management information systems (MIS) and terminals for data input, the 1980s decision support systems (DSS), executive information systems [EIS] and the microcomputer. The 1990s have given us more powerful mainframes, micros, laptop and notebook micros, GUIs and client/server technology. These technologies, although not the only ones reflecting progress in IT, have had a profound effect on how we develop information systems.

The client/server configuration combines new hardware and software technologies for maximum productivity. New technologies include PCs and workstations with CPU power approaching that of a mainframe, local area networks and user-friendly GUIs.

It has also been predicted that software is headed towards distributed environments in the form of client/server models, enabling technologies, for example object-oriented components, document-centric software architectures, data warehouse technology, and end user programming.

The client/server model seems to be the solution to the task of incorporating the performance of mainframe computing with the versatility of desktop applications for personal computers in a cost-effective way.

Client/server is defined by Smith (1995) as a form of cooperative, distributed processing in which tasks are divided among user workstations (the clients) and the server (roughly analogous to the host.) Martin (1994) describes it as splitting the application into tasks by putting each task on the platform where it can be handled most effectively. Generally this is achieved by placing the processing for the presentation on the end user's machine (the client) and the data management and storage on the server. Data processing, however, may be split between the client and the server or may be executed at the client workstation. Watterson (1995) accurately describes it as an architecture that is based on a modular design, splitting the workload between client computers that request services, such as printing, information retrieval, updating or deletion, and server computers that process the request. It is important to remember that processing is performed at both the client and the server.

A client/server system infers an approach not just to computing but to business processes in general. Client/server is an approach in which users, managers and decision makers have access to the corporate data and applications. Information management is improved with centralized validation and access rules, easy access to data improves the user's productivity and management's ability to make decisions. Information systems professionals can develop the application on their desktops which frees the mainframe resources and reduces development time. As is the case with new technology, the first step requires care, patience, and money. The environment, the technological concepts and the implementation are not as simplistic and straightforward as it seems. With the advent of open systems, standards in the industry are contributing to compatibility and interconnectivity between applications. However, support and compatibility are required from many vendors at the hardware, software and network level. Hardware reliability becomes a major issue, as do data and hardware security. With the spread of application software across the enterprise, maintaining and updating software require new methodologies and controls. Information systems professionals need to be trained in this new technology. Client/server computing changes the organization's structure. Users become responsible for maintaining their systems and must also be trained to perform duties formerly done by the IS professionals.

Up to now, client/server was defined as an architecture that splits processing between the client, the desktop PCs and the server, namely the server computer. The basic components interacting in a client/server system comprise more than the client and the server, both of which are based on the use of hardware and software. A client/server system is a complex one relying heavily on the network hardware, application development software, the different role players and the application development process. A preliminary conceptualization of the basic client/server model is illustrated in figure 1.1.

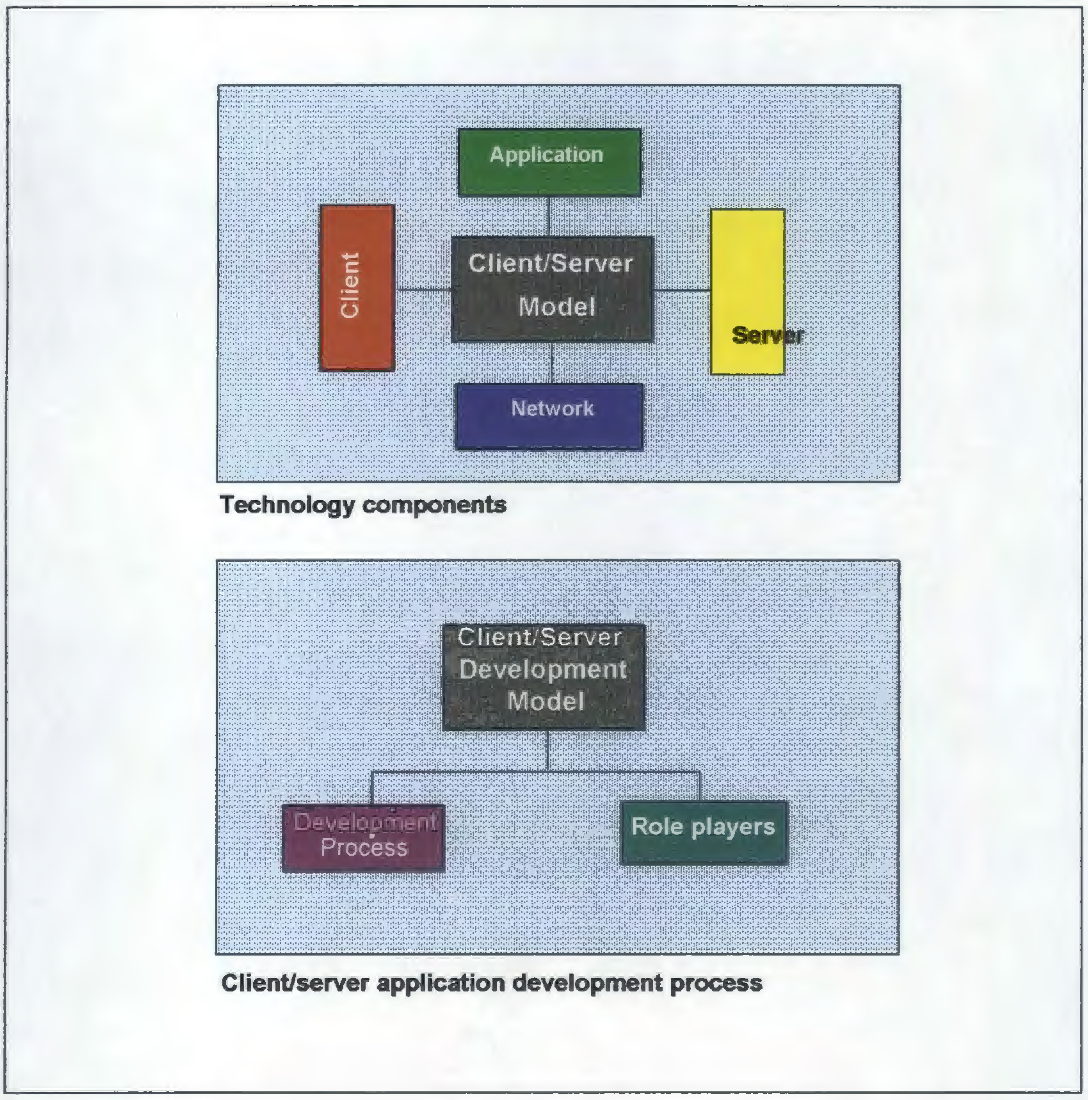


**Figure 1.1** *Preliminary conceptualization of client/server model*

## 1.5 Proposed Solution

### 1.5.1 Scope of this Investigation

This study forms part of the OOISEE project. The dissertation is of limited scope and is in part fulfilment of the requirements for the M.Sc. degree. The study focuses on the various aspects of client/server systems. A few essential issues guided the research. The most important issue is that, in this case, software development will involve a substantial number of integrated elements from different hardware platforms and versatile software applications, all of which communicate over distributed or networked systems. The development process of a client/server system differs inherently from traditional systems as it is of larger scope and involves different role players. This research project is intended to create a road map in the form of a technology reference model for client/server software development. The reference model will contain all the relevant aspects and components needed in the development process for a client/server target system.

### 1.5.2 Constraints of this Investigation

The following considerations constrained the investigation:

- The study of the hardware and network architecture is not the major objective and will not be studied in detail, but only in terms of the basic architecture and development process.
- The object-oriented paradigm which is fundamental to the OOISEE project.

### 1.5.3 Method of Investigation

- The state of the art of client/server systems was examined by means of a literature study.
- An attempt was made to conceptualize the problem, and establish a technological reference model for the development of client/server systems.

The survey included the use of CD-ROM technology, text books and relevant literature. The references were analytically reviewed, interpreted and classified in terms of the constraints of the investigation. The following keywords were used:

- client/server
- software engineering
- object-orientation
- open systems
- technology reference model.

## 1.6 Format of the Dissertation

The dissertation consists of six chapters. **Chapter 1** presents an introduction to the research topic. A motivation for investigating the area of research is given. The shortcomings in the industry are identified contextualizing the relevance of this study. The method of investigation is described and a preliminary conceptual model proposed.

**Chapter 2** explains the rationale for developing client/server applications. Various trends in the industry can be seen as drivers for client/server systems. An overview is given of trends in the industry, technological software development advances, open systems and systems integration, and the role of standards in the drive to client/server systems. The evolution to client/servers is addressed. The changing role of information systems and business process reengineering is reviewed. Software engineering concepts are also inherently part of the move to client/server systems.

In **Chapter 3** the basic components of a typical client/server application namely the client, the server, the network, the application and the role players are discussed. Different client/server classifications and application topologies are identified.

**Chapter 4** gives a comprehensive overview of the individual components of the client/server system. The client, the server, the network and the application are discussed with reference to the hardware and software components. The preliminary conceptual model is extended to include these components.

**Chapter 5** discusses the design of client/server applications, identifying key aspects in the development of a client/server application. Various aspects are overviewed, for instance strategic planning, business needs, key design issues and management of the development process. Process models, methods and techniques that can be used are reviewed, as well as development tools. The different role players are identified. The conceptual reference model is extended to support client/server application development.

The dissertation concludes with a summary and conclusion of the research in **Chapter 6**.

# CHAPTER 2

# Rationale for Change

## 2.1 Introduction

Redeveloping information systems to incorporate and utilize new technologies is one of the main challenges facing computer professionals and business people alike (Kavanagh, 1995). New technologies, for instance client/server systems, reengineering, outsourcing, downsizing and rightsizing are current challenges faced by the computer industry. Businesses are also facing major changes. The 1980s was characterized by rapid growth, expansion and an overinvestment in technology and infrastructure. In the 1990s standard practices were no longer sufficient and solutions were sought in strategies and management. In the future, businesses that will survive will be those that are able to transform business, supported by appropriate technology. Technological advances, such as personal computer technology, the rapid evolution of graphic user interfaces, networking and communications are changing the way today's computing systems are used to meet the evermore demanding business needs.

In addition, there has been substantial improvements in software development methodologies and supporting tools which have contributed to the impetus towards client/server technology. The role of information systems in the enterprise, integrated environments, open systems, software engineering concepts and standards have also contributed. This chapter reviews the historical context of the pertinent technologies and discusses the various drivers for client/server computing.

## 2.2 A Brief History of Computing Systems.

The first generation computers (1944-1958) was developed and used as general-purpose computers. Computers of this generation were very expensive devices using vacuum tubes, with a limited market due to the high cost. In second-generation computers (1959-1963), transistors replaced vacuum tubes and computers tended to become smaller and faster. In the third generation (1964-1970) the integrated circuit replaced transistorized circuitry. Magnetic disks were being used widely and computers supported capabilities such as multiprogramming and time-sharing (Hutchinson et al, 1993). The traditional environment was that of the mainframe where one system controlled all the transactions in the organization. The IBM System/360 and System/370, for instance, pioneered a scaleable, standard architecture (Kavanagh, 1995). These computer systems provided office support functions, such as payroll and accounts. The fourth generation used large-scale integration circuits, and in 1971 the microprocessor was introduced which combined all of the circuitry for the central processing unit on a single chip. The microcomputer was built from standard off-the-shelf parts and found new classes of users, not only in home use but also in personal business applications. The personal computer continued to develop and today's personal computers are capable of processing speeds and a storage capacity that rival that of the mainframe and the mini. While the performance increased, the costs decreased dramatically. Reduced instruction set computing (RISC) further improved processing speed. Thus the personal computer developed to a level where not only better performance is offered for the price, but also increased flexibility for upgrading (Marion, 1994). Computer systems during the 1980s provided core business support, office automation and decision support systems. The 1990s was characterized by a radical transition to business reengineering. Now the computer system supports the core business processes and most tactical and strategic decisions. An electronic document infrastructure is fundamental to businesses. Distributed processing places applications and databases where needed, often integrated with PC packages. Initially, each of these types of computer system addressed a fundamentally different market in terms of volume and price, but with the advances in the personal computer field, the differences have almost disappeared.

Furthermore, different computer systems may be combined in a cooperative system to meet business needs. Corporate restructuring, such as mergers, acquisitions and consolidations makes it necessary to connect or replace stand-alone applications. In many organizations, personal computers that were previously used in the stand-alone mode are being connected to form networks that support workgroup computing in cooperative systems. This is called upsizing. At the same time, some organizations are downsizing mainframe applications to take advantage of the greater cost-effectiveness of networks of personal computers and work stations. Corporate downsizing has given individual managers a broader span of control, thus requiring access to a wider range of data. Applications downsized from expensive mainframes to networked microcomputers give the user access to a much more user-friendly environment and cost-effective execution. Resource sharing is also possible. Alternatively, a set of stand-alone microcomputers may be used in the organization. Each of the end users may have access to his own data and application packages, resulting in islands of information throughout the organization. An organization could also have incorporated a combination of mainframe systems and stand-alone microcomputers, with no interconnection. These different configurations will be discussed briefly.

### 2.2.1 Centralized multi-user systems.

In this architectural approach, multiple users are connected to one centralized computer. All functional and data components reside and execute their tasks on one centralized computer platform. Multiple users can access this platform simultaneously. This approach is based on mainframe and minicomputer technology and it provides a large number of users (200 to 10,000) with costly hardware and software (Shafe, 1994). Centralized storage for large databases are provided, as well as a stable reliable environment. Centralized multi-user systems are driven by the following aspects:

- the need to spread costly investment in hardware, software and technical support staff across as many applications as possible
- the need to provide large numbers of simultaneous users (200 to 10,000) with access to applications
- the need to provide centralized storage for large databases used by multiple users
- the need to minimize data flow across networks.

Figure 2.1 illustrates this approach.

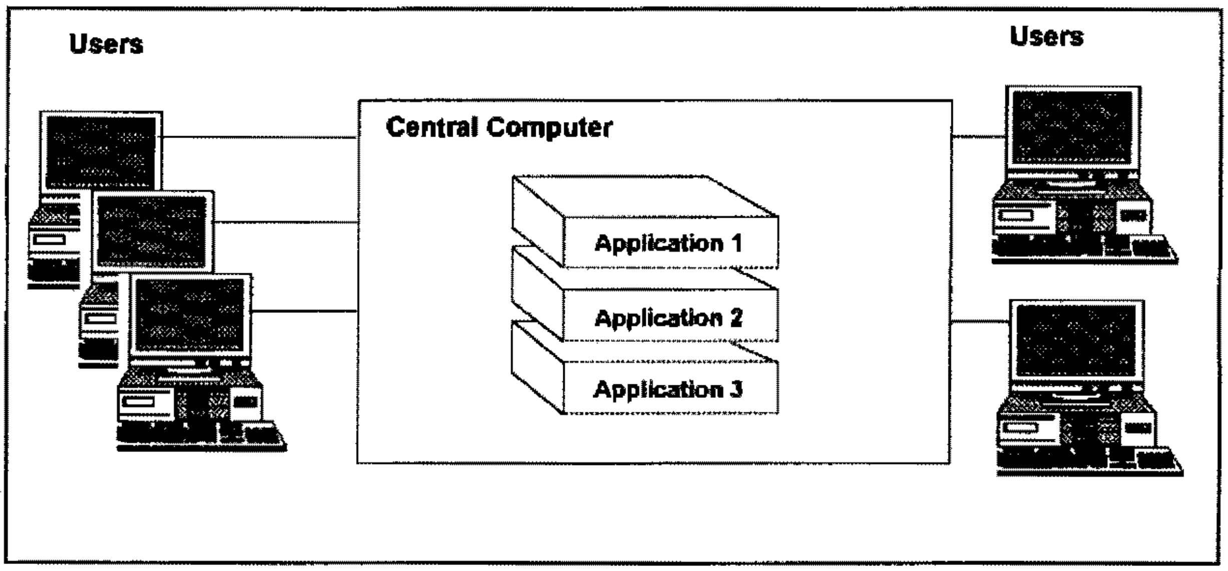


**Figure 2.1** *Centralized multi-user architecture (Vaughn, 1994)*

This technology is proprietary, however, and generally not compatible with other systems. It is also very expensive and requires a controlled environment with raised flooring, air-cooling plants, sophisticated power distribution and a large support staff. It is also very stable, reliable and well-supported although it is proprietary and expensive.

## 2.2.2 Decentralized systems.

The centralized facilities of the 1960s and 1970s became decentralized facilities without network links. A system comprising several geographically dispersed computers, each with its own functions and processes, is called a decentralized system. However, the computer systems are not connected by a network and data are transmitted via a channel. Such a system could not easily share data, applications or resources. A characteristic of such an environment is the distribution of applications of an organization on the most appropriate computer. With decentralization it is inevitable to create incompatible "islands of technology" (Marion, 1994). Individual units and systems are developed at different locations in an organization, so workload, data, and even applications may overlap. It is also possible that although the work on these systems may initially have been unrelated, more interaction, data sharing and resource sharing might be needed as the systems develop or the organization expands.

### 2.2.3 Distributed systems

A distributed system is a system in which computing functions of applications are performed across multiple sites. An application designed for this approach aims at providing the user with the required functionality and data resources on that user's platform. Distributed processing usually involves the implementation of related software across two or more data processing centers. According to Crepeau and Weitzel distributed processing (Cerutti et al. 1993):

"can be defined as a set of geographically distributed data processing resources and activities that operate in a coordinated fashion to support one or more organizational activities."

In a distributed environment, all the computing tasks that were once accomplished by a single, centralized system are distributed to a number of smaller self-contained systems.

In addition, such a system has a distributed database viewed as a single logical database that is physically spread across computers in multiple locations connected by a data communications network. This type of database allows multiple users to share the data resources. Distributed computing arose through:

- a paradigm shift driven by the need for enterprise-wide, cooperative applications
- the needs of the organization to form a deliberate policy
- the strategy to devolve computing from the traditional mainframe environment to a distributed one
- the aim to integrate existing and future heterogeneous systems within one organization.

Advantages of distributed databases are increased reliability and availability, local control, modular growth, lower communication costs, and faster response. There are, however, also disadvantages, such as higher software costs, higher complexity and processing overhead, and additional management aspects such as data integrity, data distribution and security.

### 2.2.4 The local area network

A local area network supports a network of personal computers, each with its own storage device, that are able to share common devices (such as a hard disk) or software (such as a DBMS) attached to the LAN. One PC is designated as a file server where the shared database is stored. A file server is a device that manages file operations and is shared by each of the client PCs that are attached to the LAN. In the basic LAN environment all data manipulation occurs at the workstation where the data is requested.

Each personal computer is authorized to use the DBMS, therefore there is one database but many concurrent copies of the DBMS, one on each of the active personal computers. The primary characteristic is that all data manipulations are performed at the personal computer, not at the file server. The file server simply acts as a shared data storage device. Figure 2.2 illustrates this.

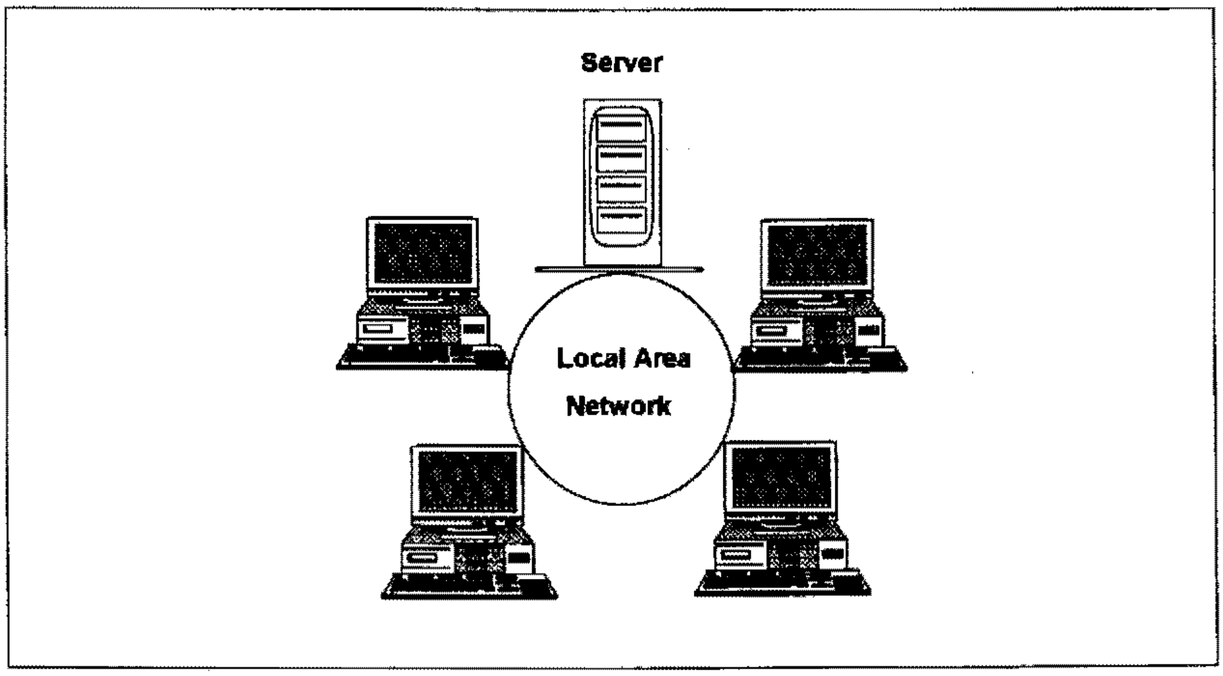


**Figure 2.2** *Local Area Network (Marion, 1994)*

There are three limitations when using LANs (Critchley & Batty, 1993):

- First, considerable data movement is generated across the network, placing a burden on the PC to do extensive data manipulation. This creates a high network traffic load while functions are performed and possibly duplicated on the PCs.

- Second, a full version of the DBMS is loaded on each workstation. This uses a considerable amount of memory which means there is less room for other application programs on the PC workstation. As each workstation executes tasks on its own, each client must be powerful enough to provide suitable response time.
- Third, the DBMS copy on each workstation must manage the shared database integrity and security, e.g. locks. Programming is more complex, as each application must handle proper concurrency, recovery and security controls.

However, local area networks are typically within the 'local' range, with a total network cable length of under 2 kilometres. When an organization is geographically dispersed, it may be preferable to implement a distributed system.

### 2.2.5 Client/server systems

Client/server systems have developed as a result of the disadvantages and shortcomings of distributed and local area network systems, in an effort to combine the advantages of new technologies. The client/server architecture has been described as a form of LAN in which a central database server or engine client/server configuration performs all database commands sent to it from client workstations, and application programs on each client concentrate on user interface functions (Critchley and Batty, 1993). A more accurate description is given by Vaughn (1994), stating that the functional components of an application are partitioned in a manner that allows them to be spread across, and executed on, multiple different computing platforms, sharing access to one or more common repositories of data. Figure 2.3 illustrates the idea:

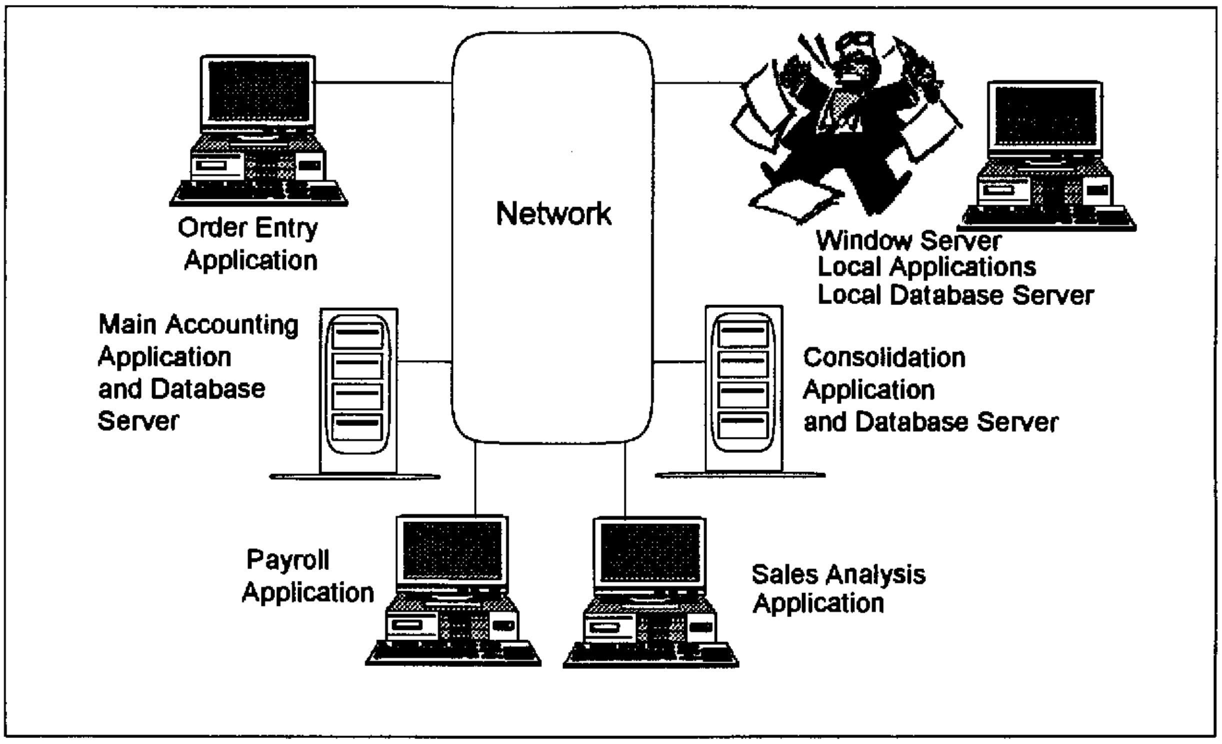

Network
Order Entry Application
Window Server
Local Applications
Local Database Server
Main Accounting Application and Database Server
Consolidation Application and Database Server
Payroll Application
Sales Analysis Application

**Figure 2.3** *Client/server application (Gagliardi, 1994)*

It was motivated by the trend of price reductions and performance improvements achieved by the microprocessor-based components of the computer industry (Kavanagh, 1995).

## 2.3 Software Development Advances

Several software development innovations influenced the development of client/server applications. The object-oriented paradigm, rapid application development (RAD) environments, GUI, cooperative work, the development of distributed relational databases and middleware may be cited.

### 2.3.1 Object-Oriented paradigm

The object-oriented paradigm allows one to interpret an application as a collection of objects that interact with one another, change as a result of that interaction and yet maintain their identity through change. This allows the creation of a library of standard objects that can be reused to build many different types of applications. Wessels (1993) states that NCR's object-orientated programming (OOP) has a number of benefits:

- Lower development cost will result in a 70% shorter development cycle.
- Higher programmer productivity was implied by 75% fewer lines of code written, while 50 - 80% of the code was reusable.
- Higher end user productivity.
- Improved developer productivity.
- The maintenance cost can be reduced.
- Flexibility and scalability. Scalability of a system implies the ability to run the application on machines of varying sizes. This allows the application to support growth of hardware.
- Better resource utilization.
- The OOP experiment demonstrated that the time spent and the expenses were less because of the reduction in the total number of lines of code as well as the ability to reuse proven and tested codes.

Changes in business require rapid application development and the object-oriented paradigm supports this. In 1990 Cox (1994) identified object-oriented methods as part of the solution, as object-oriented analysis represents a multilayered architecture of reusable, interchangeable software components analogous to the multilayered architecture of hardware engineering.

- There is a marked compatibility between the objectives of object-oriented methods and client/server systems.
- The modularity of object-oriented analysis is an inherent factor in a client/server system which consists of different separate modules.
- Abstraction is an advantage of object-oriented systems and is also present in client /server systems. The modular components of client/server systems are seen in levels of abstraction.

- Both object-oriented systems and client/server systems are extensible as they contain components that may be added or removed as needed.

The Object Management Group (OMG) has been encouraging the development of cross-vendor compatibility standards for all object-based systems for several years. MICROSOFT'S release of the OLE Version 2.0 specifications has generated a heightened interest in the role that object technology is likely to play in the future of computing. OpenDoc from IBM is its strongest contender and also based on object-orientation. Intellicorp's SA representative specializes in object technology and has a range of products including OMW, Kappa, Kappa CommManager and Forté.

## 2.3.2 Graphic user interfaces (GUI).

Many graphic user interfaces are based on the WIMP (Windows, Icons, Menu's and Pointers) environment and provide a user-friendly presentation layer to applications. The development of Microsoft Windows 3.0, 3.1 and 95, IBM's Presentation Manager, Open Software Foundations' Motif, and USL's OpenLook, has had a major impact on the acceptance of client/server computing.

## 2.3.3 Cooperative software

Cooperative software refers to applications that are split across more than two machines with transactions running across them (Forge, 1995). Cooperative systems are often built around different pools of data, data entry points, and data access or usage areas. To support existing systems or legacy applications, cooperative software is often concerned with expanding the existing mainframe or minicomputer with new equipment and software. Cooperative software allow more users to communicate and interact with each other in the client/server environment.

Cooperative software permits the development of cooperative systems which have often grown out of a calculated or unconscious reengineering of the business processes. As such, they are tailored to specific business situations to provide a platform for everyday core operations with commercial advantages that could not be obtained through traditional information systems. Examples are a European bank, an automotive import/export company, savings and loan company and an airline (Forge, 1995).

### 2.3.4 Development of distributed relational databases

Since the 1960s, database management systems have handled all such functions as human interface support, transactions and database support in one program on one machine. Although fairly easy to implement, these systems do not permit easy distribution of the database across several machines on the network, or distribution of functions such as user interfaces. The client/server model is particularly useful for database access in a distributed environment. New software has been developed to support distributed data and SQL, the standard language to access client/server data. Vendors of database products, such as Sybase, Oracle, Ingres, and Informix have developed products to support and access multiple data structures pushed by the specific advantages of distributed databases, which include:

- Cost attractiveness of the modular increase in capacity of client/server databases
- Reduction in telecommunication charges and network traffic problems by reducing access to remote data centres and placing data closer to access points
- Linking disparate databases to build a single composite information database for the user
- Use of low-cost platforms, such as PC LAN servers where possible
- Placing databases near to users and owners of data, increasing their control over the data. Flexibility to cope with new business demands.

### 2.3.5 Multi-threaded or concurrent processing

Whereas MS-DOS could only support one process at a time, new operating systems have been developed that support concurrency and communication, such as OS/2 and UNIX. These aspects support distributed client/server systems and allow cooperative software.

### 2.3.6 Middleware

Middleware refers to various high-level services that mediate between PCs and the remote resources they are trying to access. Watterson (1995) refers to it as a sort of a railroad switch linking the front-end to remote data. The function of middleware is to make physical and logical connections to the target data source. Middleware can also be seen as a type of protocol used to link front-ends to back-ends. The development of middleware facilitated communication between client applications and the server, using a variety of methods. Middleware will be discussed in more detail in Chapter 4, section 4.4.2.2.

## 2.4 The Changing Role of Information Systems

The fundamental role of information systems in organizations is changing. As technology moves processing power and applications development tools to the user department level, traditional IS functions are being decentralized and delegated to the user level (Marion, 1994). Organizations are changing to smaller business units that are more adaptable to changes in the business environment and correspondingly more competitive. Three major contributing factors are defined as: moves to departmental computing, the development of user-centred environments, and increased end user sophistication. The client/server application architecture provides support for all of these aspects.

### 2.4.1 Moves to departmental computing

Change is the driving imperative of today's highly competitive, global business environment (Lewis, 1995). In order to support business strategies and objectives, companies are streamlining processes, shortening production cycles and 'time-to-market', trying to reduce costs and improve quality and customer support. This process is also known as business process reengineering (BPR). To help their businesses succeed today, IS managers need to implement systems that can adapt to change. Therefore, the move to departmental computing gives the manager more power and versatility. The trend to decentralize authority and responsibility in organizations gives departments their own technical resources to devote to automation projects that best fit their needs. Many organizations are taking steps to distribute information system expertise to departments or develop departmental information technology staff (Marion, 1994).

Client/server systems require less development time and ongoing support than their mainframe equals. Client/server systems can adapt to change. They are flexible, scaleable and open, allowing the organization to grow and move forward without creating barriers.

There is, however, a limit to decentralization. Regardless of how much autonomy a business unit has, it still needs to share information with other business units in the entire organization. Some experts believe that the role of the central IT department of the future will be to formulate and define standards and provide consulting services to the business units in the organization.

### 2.4.2 End user environments

There is a shift towards distributed processing systems with increased end user programming. Factors that influenced this move is the accessibility of PCs, availability of laptops and notebook computers and availability of on-line services.

The advantages of telecommunications being available to the PC end user will also bring a marked change. Lunn (1992) predicted in 1987 that major growth in 1990 to 1995 would be downsizing from mainframe to client/server. He subsequently predicts that transaction processing is approaching economic saturation and industry growth will come from areas such as mail-enabled applications. The use of Bulletin Board Systems, Internet and CompuServe is soaring due to the acceptance of e-mail and fax services, awareness of the value of interoperability, and the evolution of cooperative computing and increased congestion in urban areas.

Traditionally the computer user was limited to professional software developers working on a mainframe or minicomputer. The personal computer made the computer environment and application packages accessible to a wider range of end users, with the definition of end user here expanding broadly to any person with sufficient knowledge to use the computer and its applications. A combination of technologies has led to the development of a "user-centred" approach to developing information systems.

### 2.4.3 Increased end user sophistication

As the use of personal computers increased, the availability of enabling technologies in the form of 4GLs, application packages and programming languages increased. Distributed systems allowed the end user to access enterprise-wide data that was previously unattainable. All of these factors are enhancing end user sophistication.

The growing trend of user-friendly 4GLs supports end user computing, as it is easy to learn and use and it gives the end user control over his own functions and his own data. Basic computer skills are becoming more common and with the support of state-of-the-art GUIs the amount of end user training is reduced. GUI-based application development tools and 4GLs are providing more development power to the end user.

## 2.5 Open Systems and Systems Integration.

The need for integrated systems is often the key to distributed systems moving to an open system environment (OSI). Systems integration holds the promise of aligning technology with business needs. Systems integration is defined by Bergen (1994) as the responsibility and accountability for coordinating all activities relating to the planning, developing, purchasing, implementing and managing of complex information technology systems for the business.

Systems integration has the primary goal of bringing together disparate systems. The openness of the system is determined by the extent to which it supports the available standards. We can regard an open system as a set of standard relationships that allow different computers, subsystems, applications and system software to operate together. An open system is characterized by all the interfaces that connect with the outside world. It includes software interfaces, such as the user, applications and information, and communication interfaces, such as buses and peripheral interfaces.

A practical example of open systems is the 386/486 PC market (Kavanagh,1995). We can assemble a system from component hardware and software and run it with reasonable confidence. As opposed to proprietary systems, an open system gives its owner the freedom to change vendors and architectures if necessary. History has shown that it is advisable to be prepared to move systems to new platforms and architectures. Jones (1994) differentiates between a vertical (vendor-driven) model and a horizontal (user-driven) model. The vertical model is vendor-driven where the vendor supplies the user with a specific configuration, for instance IBM, DEC, NCR or UNISYS. Installing one of the above, the user will have no choice in selecting the operating system, application packages or database, as the specific configuration only supports specific packages. The horizontal model gives the user a choice of combining applications, hardware and operating system. This more open system will supply the user with a more powerful environment to support his/her own needs.

A goal of current systems is the ability to comprehend the full potential of the technology and exploit these capabilities for long-term competitive advantages.

The need for open systems is not only driven by business needs but also by cost and user demands:

- The cost of migrating to new systems from a vendor-driven model is too high. In an open system different applications can be used without a high cost.
- User demands extend beyond the resources of a single vendor. The vendor will not be able to constrain the user as to specific applications or packages. A growing demand for sophisticated solutions has exceeded any single vendors capability. If the company wants to remain competitive, it will have to use all the resources available and not be constrained by a single vendor.

Specific advantages of open systems (Gagliardi,1994) are:

- Vendor independence. To qualify as open, a standard must be sold, supported, and controlled by a large number of independent companies. For example, the source code of MS-DOS is owned by Microsoft, and all additions must be added by Microsoft. Innumerable companies own UNIX code, and major contributions have been made to the UNIX standard have been made by institutions such as the University of California at Berkeley.
- Portability allows applications and their data to be moved from one hardware platform to another. This is a critical component of reducing maintenance costs, as the life span of hardware is longer than that of software (McFadden and Hoffer, 1994).
- Interoperability implies the ability of applications to share data. This will also reduce maintenance by reducing work.
- Scalability of a system implies the ability to run the application on machines of varying sizes. This allows the application to support growth of hardware.

A client/server system based on an open architecture will allow organizations to integrate emerging technologies into current environments giving them a competitive advantage. Its systems will give the organization the competitive advantage of being able to utilize all resources available, and supply the employees with all the utilities needed. The user will no longer be constrained by vendor support.

Overall, the rapidly changing business environment of today requires the information system to provide adaptability, flexibility, dependability and interaction with other systems. The client/server system promises to be the answer to these goals.

There is a paradigm shift in thinking with the system becoming a more strategic resource transforming all the existing systems into one cohesive management information tool. The focus will be on ensuring smooth internal relationships between departments and external relationships with suppliers and buyers. 'Islands of information' will be integrated into a single cohesive network. Client/server systems will be a solution to the problems surrounding the realisation of these goals.

## 2.6 Standards

Over the past years, a dominant trend in computing has been the move to software standards (Gagliardi,1994). Standards encompass many different aspects of computer software, such as user interface, operating system, data access, communication, language and development methodology. The User Alliance for Open Systems defines standards as those which allow unimpeded access to the information required to do one's job. A standard can also be seen as an industry-accepted way of doing things that are created by consensus.

Standards are important to keep the computer system open to new products and flexible to change with changing needs. Standards will also:

- allow a free choice of applications developed to the market standards thus ensuring the user of a safe investment
- create the possibility of using standard parts, thereby decreasing costs and increasing the speed of improvement

- facilitate interconnectivity of systems, promoting portability and flexibility
- make existing systems compatible with future systems
- introduce a homogeneous client/server computing environment across different operating system platforms
- ensure open, free market competition and variety control
- ensure lower costs
- ensure the presence of interchangeable or reusable parts in a system.

Standards have become central to the direction and future of open systems and the IT industry. There is, however, considerable confusion about standards and their implementation. Providers also try to exert influence to have their products standardized.

De facto and de jure standards exist. A de facto standard is the term applied to a product or system from a provider that has captured a large share of the market and which other suppliers tend to copy or use in order to obtain a share of the market. A de jure standard is created by an organization developing formally recognized standards. It is created in an open forum in which everyone has a chance to participate, and under rules of consensus. De jure standards cannot be changed by an individual but must go through the same procedure by the standards developing organization. The open systems interconnection (OSI) standard, Ethernet, POSIX, SQL and other language standards are examples of these standards.

Standards are essential for the implementation and incorporation of client/server systems. The industry, vendors, researchers and users are instrumental to the development of standards for client/server systems that meet the needs of the customers.

The definition of standard software components lags behind the definition of standard hardware components. However, the importance of software standards is gaining in momentum in organizations.

Gagliardi (1994) defines eight major levels of software standards when looking at the areas financial reporting, managerial analysis and process control:

- user interface standards to insulate operations
- customization standards to insulate transactions
- application standards to insulate processes
- development standards to insulate source code
- data format standards to insulate data
- query language standards to insulate the database engine
- operating systems standards to insulate hardware
- networking standards to insulate the system.

Dewire (1993) defines four areas where standards influence the development of client /server systems:

- Platforms: Hardware and software vendors develop theses standards following de facto standards, such as Intel chip, UNIX, and DOS with Windows.
- Networks: Industry-standard networking protocols such as OSI and TCP/IP are being used instead of vendor-specific protocols.
- Middleware: This type of software that is situated between the application and the operating system includes GUIs, databases, e-mail systems, software development tools and IS management tools.
- Applications: Organizations decide on standard applications to facilitate workgroup interaction and work-product compatibility at enterprise-wide level.

Standards support the realisation of open systems.

## 2.7 Business Process Reengineering

In section 2.4.1 the process of restructuring the business processes was discussed. Business process reengineering (BPR) restructures the business prosesses in an organization, identifying areas that must change according to the business strategy, using a methodology for implementing the changes.

Client-server computing supports the restructuring of business processes, as it is based on a modular, flexible architecture. The flexibility to expand and change is achieved with the versatile architecture of the client/server system. New technologies comprising hardware and software may be implemented according to user and organizational needs. New interfaces can be created for old applications to accommodate business changes. Thus client/server applications can be used as a tool for business process reengineering. Business benefits derived from client/server systems will be productivity, accuracy, portability and flexibility to support change.

## 2.8 Software Engineering Concepts

Software engineering is a dissipline that tries to enhance the quality of the product. During the phases of the product life-cycle, elements are identified to better the product, testing is streamlined, methods of verification are implemented. Computer professionals have seen many changes in systems, techniques and methods used. CASE tools, fourth generation languages, AI databases and even COBOL at one time, were thought to provide the answer to programmer productivity problems. Client/server systems differ from the previous changes as it is grounded on a trend of price reductions and performance improvements achieved by microprocessor-based components of the computer industry. It addresses the end user demands for systems that are manageable and understandable. It is also used world and industry-wide. Client/server technology promises the benefits of flexibility, scalability, technological potential and reduced costs (Gagliardi, 1994).

*Flexibility:* Multi-user mainframe performance, as measured by response time versus workload, tends to remain consistent until the system's design capacity is approached. With a workload beyond 70 to 80 percent performance decreases rapidly. This limits the flexibility of the system. Client/server architectures exhibit significantly different performance characteristics (Marion, 1994). The processing is distributed across different platforms with only partial and limited server sharing. This results in a more predictable performance curve. The cost per unit is more predictable and increases in a linear fashion. The architecture is inherently scaleable and capacity can therefore be added in smaller and less costly increments.

*Scalability:* Client/server architectures based on open-industry standard technologies are superior to the proprietary technologies of a multi-user architecture. In the proprietary multi-user environment the customer's upgrade alternatives are limited to what the manufacturer provides (Dewire, 1993). The client/server systems are based on open technologies. Therefore, the customer has many alternatives at his disposal for adding processing capacity in incremental steps that are suited to the user's need.

*Technological potential:* Because of the proprietary nature of multi-user systems, technological advances have been controlled and directed by a small number of original equipment manufacturers. The competition for these manufacturers in their specific market segments was little, if any. The high cost of research, development, and production of proprietary new technologies has historically resulted in relatively long product life cycles. These factors, combined with the financial need to protect customer technology investment supported the move to client/server systems. However, client/server enabling technologies have been driven by intense competition across the entire range of supporting technologies, from central processing units to memory to disk drives! Emerging and existing industry-wide standards are protecting customer concerns.

*Complexity*: Client/server environments are uniformly multivendor, requiring the application developer to integrate often incompatible hardware, software and network components. Therefore, client/server systems are rendered more complex. However, the modularity of components simplifies the design.

Driving forces for client/server are compiled and illustrated in Figure 2.4

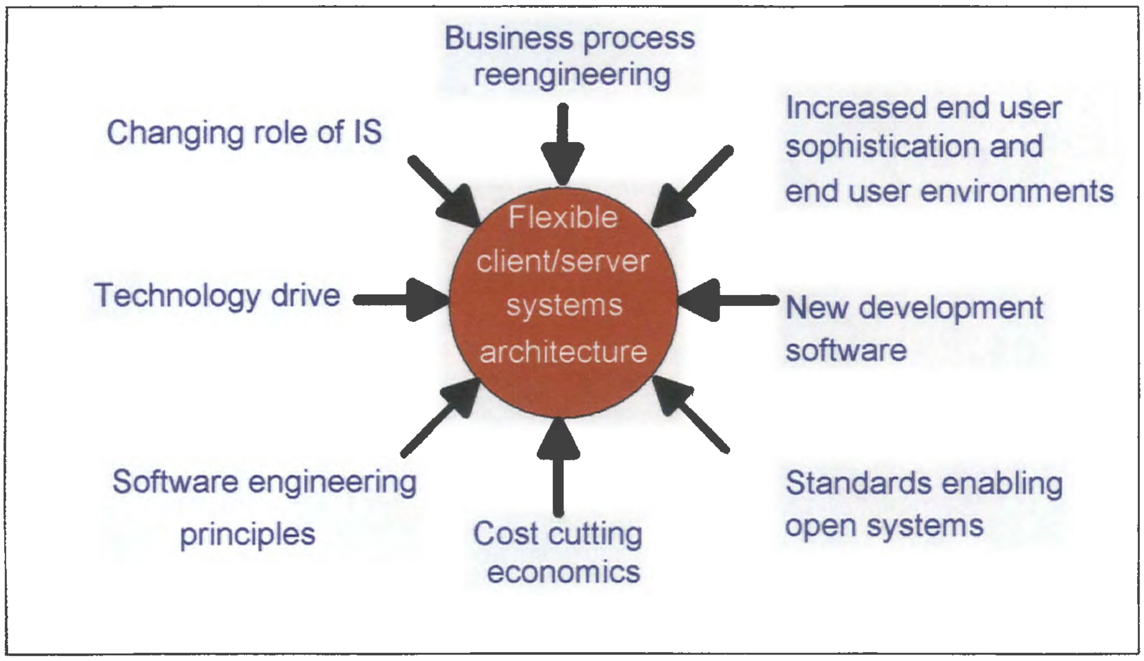

Business process reengineering
Changing role of IS
Increased end user sophistication and end user environments
Flexible client/server systems architecture
Technology drive
New development software
Software engineering principles
Cost cutting economics
Standards enabling open systems

**Figure 2.4** *Driving forces for client/server systems.*

## 2.9 Summary

The move to client/server systems is the result of different factors. Technological advances reduced the price of personal computers to such an extent that their availability greatly increased. Performance and storage capacity of personal computers also increased, leading to the implementation of personal computers throughout organizations. With increased availability and advances in technology, end user computing grew and led to a demand for enterprise-wide data that are accessible to the end user. Client/server systems address real end user demands for systems that are manageable and understandable. Client/server configuration solves these demands. Local area networks permitted connection between personal computers, mainframes, minis and other local area networks, thus allowing the end user access to a large variety of software and data. Client/server systems thus provides a flexible, modular architecture that makes it easy to build new applications and maintain old ones. Costs will be reduced by replacing expensive hardware with less expensive hardware configurations.

Standards allowed the use of open systems. As the client/server configuration consists of a large variety of systems, standards and open systems greatly enhance and support the implementation of client/server systems.

The changing role of information systems supports the move to client/server applications. The move to departmental management of systems, increasing end user sophistication, programming, increased availability of computers, user-friendly software and graphic user interfaces all add impetus to the move to client/server systems.

To support change, a flexible computer system is needed. The traditional computer systems, such as mainframe systems, did not support business reengineering at all. Client/server systems are flexible and versatile and will be able to support change. It is also believed that the use of client/server applications will enhance the quality of the product.

In the next chapter the components of client/server systems will be discussed.

# CHAPTER 3

# Client/Server Technology

## 3.1 Introduction

Client/server technology is made possible by a combination of different technologies. Centralized multi-user systems, distributed computing, cooperative computing, distributed transaction processing and multi-user environments, as discussed in the previous chapter, all contributed to the concept.

What is a client/server architecture? Webster's dictionary defines architecture as the art and science of designing and erecting a building. Within the context of information systems client/server architecture can be seen as an approach to the design of a software application. Client/server systems (Forge, 1995) have been defined in various ways:

"Client/server is where an application is split into parts that are executed in different places".

Vaughn (1994) provides a more precise definition:

"Client/server architecture is an application design approach that results in the decomposition of an information system into a small number of server functions, executing on one or more hardware platforms, that provide commonly used services to a larger number of client functions, executing on one or more different but interconnected hardware platforms, that perform more narrowly defined work in reliance on the common services provided by the server functions."

An overview of the five basic components of the client/server model are presented in this chapter, followed by a taxonomy of client/server systems.

## 3.2 The Client/Server System

A client/server system comprises a front-end client, with full functionality, that can request that work be done by the server system, performing data management and other routine tasks. The application itself deals with the customized procedural logic and user interaction. Connections are established according to a communication protocol. A simplified illustration of the components of the client and the server, connected by a network, can be seen in figure 3.1 (Dewire, 1993).

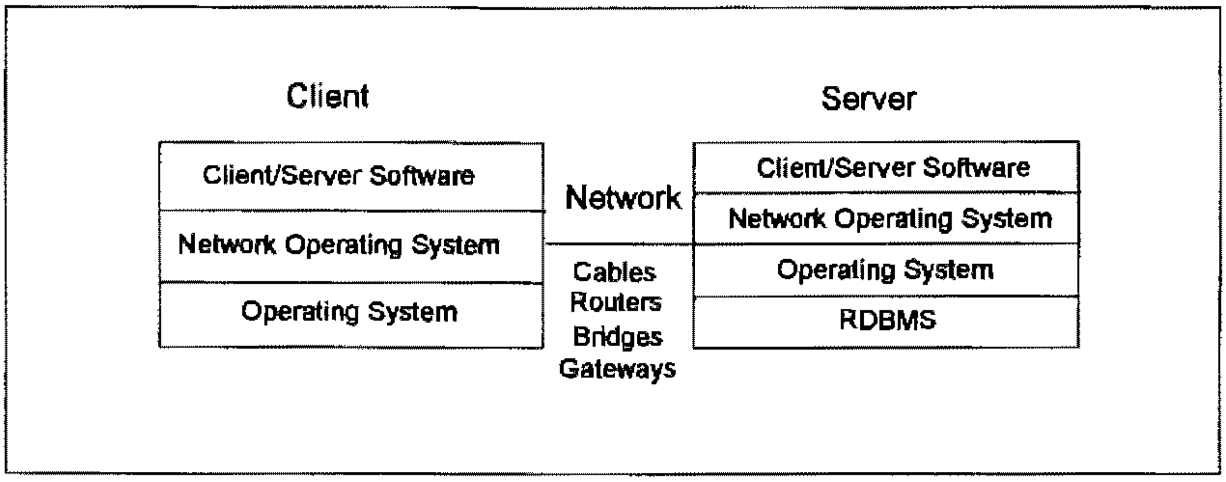


**Figure 3.1** *Components of client/server computing.*

Figure 1.1 in Chapter 1 presents a basic client/server model, consisting of the following components: the client, the server, the network, and the application. The functional components of an application is designed and executed across multiple processing platforms. It is important to remember that the participants are inherently functions and therefore the client of one function is able to act as a server to another function. Client/server applications are still in a developing phase. These applications are most commonly database applications but not necessarily so (Vaughn, 1994). Workgroup applications that provide shared communications, scheduling, workflow routing and other functions are also built on a client/server architecture. The basic characteristics of the components will be discussed in the next section and explored in more detail in Chapter 4.

### 3.2.1 The Client.

The client hardware is the desktop machine that runs client software and is operated by the end user. The client could be a personal computer or workstation but also a mini or midrange computer. The client software is used to formulate the data requests of the user and send it to the server, and also to receive the data and results of requests from the server. Client software may perform some application logic on the results before passing it back to the presentation layer. The client workstation will comprise the presentation layer of the software, usually a graphic user interface (GUI), operating system, desktop applications and the application program interface (API). The presentation layer constitutes the layer that the user interacts with. It is often, but not always a graphic user interface, providing the user with a graphic-oriented, user-friendly, front-end. The client also uses an operating system, robust enough to support the presentation processing and application logic processing required for the applications. The windowed environments are proprietal and each is designed for a specific operating system, for example Windows 3.1 for DOS. Communication software handles transmissions of requests and receives the results.

In some cases a client can also be a server, acting as a client by requesting data from another server. Processes are assigned to the area that can best support the functions. For example, a PC is much more suited to support a GUI than a mainframe terminal.

The basic client and server services are listed in table 3.1.

| Client Functions | Server functions |
|---|---|
| GUI | File, print, database server |
| Distributed application processing | Distributed application processing |
| E-mail | E-mail |
| Terminal emulation | Communication |
| Application processing | Network, configuration, and resource management |

**Table 3.1** *Basic client/server functions (Dewire, 1993)*

The client will be discussed in detail in section 4.2 of Chapter 4.

### 3.2.2 The Server

The central file server manages the connections in a network configuration. The server contains the data management software that has been designed for server functionality. Compared to the desktop micro, the server has increased memory and storage capabilities, increased processing power and improved reliability. The server contains data management software, a server and network operating system, network software and application software.

The data management software responds to the requests from the client and provides data retrieval functions, as well as updating and storing of data (Marion, 1994). Relational databases have become the de facto standard structure and SQL the de facto data access language. Servers also provide repositories for data. DBMS server software provides gateways to non-relational data. It also incorporates management functions, such as backup and recovery routines, and testing and diagnostic tools.

Before database servers were available, a local machine required that the database reside physically on the same machine for data-accessing functions. It was almost impossible to access any data on any machine other than that where the data resided. Database servers facilitated central and distributed storage. The application runs on the workstation, accessing the database with requests that execute database services in the server. The database software on the client system intercepts requests for data access and uses the network software to route the request to the database software running on the server. The software executes the request on the server and returns the result to the client. Enhanced server features include:

- Application programs are never aware of the locations of devices invoked.
- Multiple database types can be used in the system with a minimum of programming effort.
- New types of databases may be added with little or no impact on application programs.
- The load is automatically balanced among several servers performing the same service.

Different types of servers, i. e. file server, data server, database server, computation server, will be discussed in Chapter 4 in section 4.3.1. Key factors in the development of successful client/server applications are the separation of presentation management from the other application services, and the distribution of application logic between the client and the server.

### 3.2.3 The Network

The client and server are linked by a network or other communication system. The network component moves requests to and from the client and server. The network hardware comprises the cabling, the communication cards and the devices that link the server to the clients. Communications allow the server to access other servers and clients in the network, and may consist of more than one hardware platform. Network design involves selecting a particular network architecture, e.g. Token Ring, Ethernet, ARCnet, a transport protocol, e.g. TCP/IP, NetBios or APPC, and hardware networking equipment. Network capacity will influence the adequacy of performance. Key assets are reliability, speed and bandwidth.

Depending on the geographical range of the organization, the network may be a LAN, WAN or MAN (local, wide, or metropolitan area network). In a multinational configuration, communication may be facilitated by either a WAN, or modems and traditional phone lines. Additional hardware is required for interconnecting LANs, or to link the LAN to a WAN or MAN.

Networks require a network interface card (NIC) or adapter for connecting PCs to the network, the architecture of the network determining the type of adapter.

Network software resides on the network, permitting communication and data flow between the different components. The network operating system manages the network-related input/output processes of the server. Each network operating system has its own protocol which is a set of rules defining formats, order of data exchange, and actions concerning transmission or receipt of data (Watterson, 1995). Peer-to-peer LANs are a specialized category, suitable for departmental workgroups. Windows for Workgroups (Microsoft) and LANtastic (Artisoft) are examples of these. Messages and data are transmitted based on several protocols. The network is usually managed by a network specialist and not by the IT professional designing the system. The network will be discussed in more detail in chapter 4, section 4.4.

Middleware spans the client, server and the network, and can be seen as software that connects applications, databases, user interfaces, and shared services. Middleware are discussed in detail in Chapter 4, section 4.4.2.2

### 3.2.4 The Application.

To study the relationship between client and server it is necessary to examine the basic elements that are used in the computing process of the application.

The application is a program that will be executed partly on the client workstation and partly on the server. An application will use the client's user interface for the presentation to the user, and the server for data services and processing. Communication software will provide the physical link and the underlying protocols between the various parts. A basic client/server application comprises an operating system, database management system, data storage, application software, user interface, and display device.

Middleware is a crucial part of the client/server system providing key functions for connecting applications, databases, user interfaces and shared devices, grouped together. Middleware is addressed in greater detail in Chapter 4, section 4.4.2.2.

### 3.2.5 The Role Players

Due to the fact that client/server applications frequently involve multiple platforms, multiple databases and multiple application programs, they are more complex than a single system application. It may be necessary to form a team, consisting of persons with the necessary skills. The project team must be managed by a project manager, one or more programmer analysts, one or more LAN specialists, a data communications specialist, and an end user or end user representative. In Chapter 5 the role-players and the necessary skills will be discussed in detail.

## 3.3 Categories of Client/Server Applications

Client/server computing has various dimensions and there are several variations and ways of implementing it. Client/server systems may be categorized based on their functions, their architecture or their implementation, for example Dewire (1995) identifies three classes based on the area where most of the processing is done. He also categorizes applications according to their support function. Marion (1994) and Hall (1994) classify client/server computing according to the implementation approach. An important classification is based on the architecture of the system, and two-tiered, three-tiered and multi-tiered architectures may be identified. The categories of client/server applications are reviewed in this section.

### 3.3.1 Classes of Client/Server Applications

Dewire (1993) categorizes client/server applications, based on the area where most of the processing is done, namely host-based, client-based and cooperative prosessing, as illustrated in figure 3.2.

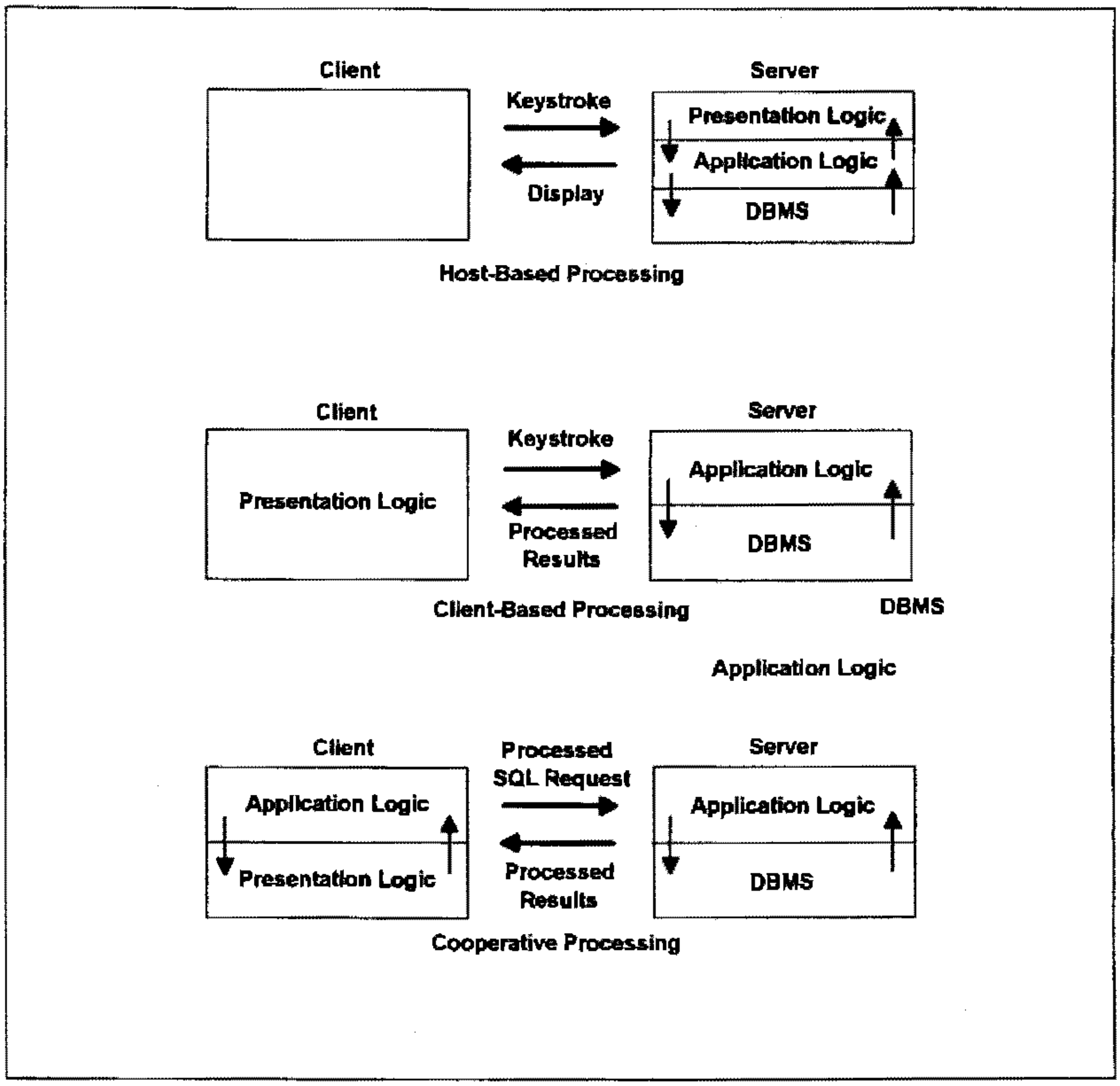


**Figure 3.2** *Classes of client/server applications*

### 3.3.1.1 Host-based processing

The basic class of client/server application has a presentation layer running on the desktop machine while all application processing runs on the server/host. This configuration is based on the rationale that users are more productive working with an easy-to-use graphic front-end. The application requires less functionality on the client.

### 3.3.1.2 Client-based processing

This configuration places all the application logic on the client machine, with the exception of data validation routines. Coordination is required between platforms and between the software, while the use of the network is more sophisticated. Users can access data on any node, but the use of the server is still constrained.

#### 3.3.1.3 Cooperative processing.

This configuration can be described as a full client/server approach, using a fully cooperative peer-to-peer approach. In this approach all components are equal and can request services from and provide services to one another. Processing is performed at the most appropriate component. Data manipulation may be performed by either the client or the server. Application data may exist on both the client and the server. Cooperative processing requires coordination and a great amount of integrity and control issues.

### 3.3.2 Support functions in the enterprise

Group interaction in the client/server environment ranges from electronic messaging and mail, shared data, to shared applications.

#### 3.3.2.1 Office systems

Client/server systems provide a framework for electronic communication. Many linked LAN systems are being used for enterprise-wide mail systems and workgroup applications. These could incorporate electronic mail (e-mail), access to bulletin boards or groupware software, such as Notes from Lotus Development Corp. Mail products include Microsoft's Mail 3.0, and Lotus's cc:Mail.

#### 3.3.2.2 Database Access

Some client/server applications are written to access corporate data, as illustrated in Figure 3.3.

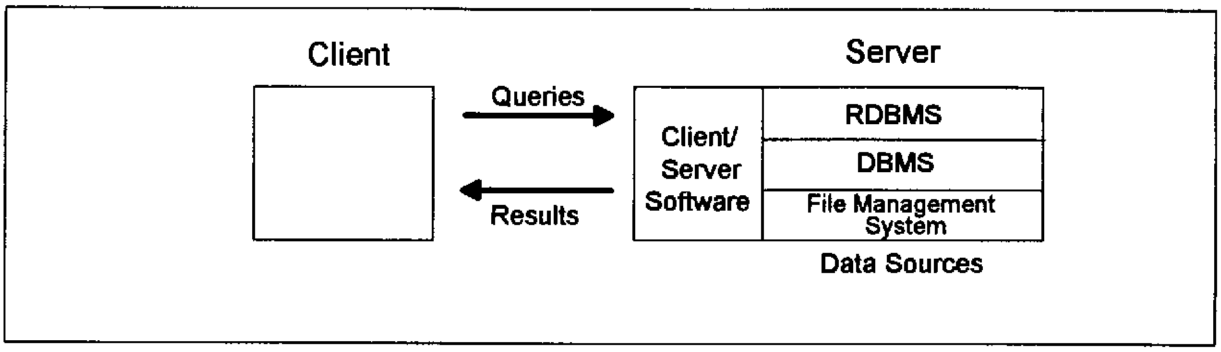


**Figure 3.3** *Database access (Dewire, 1993)*

In these systems various users access the data which are stored on a centralized server. Applications may be read-only or read-write and the main objective is to improve user productivity.

### 3.3.2.3 Transaction processing applications

Typical transaction processing applications include on-line systems such as order entry, inventory, and point-of-sale systems, others include mission-critical applications, such as air traffic control systems. Important aspects of these systems are security, recovery procedures for system failure, and commit and rollback facilities.

### 3.3.2.4 Investigative applications

These applications are designed to support decision makers by supplying them with the necessary data. Investigative systems can also be called Decision Support Systems (DSS) and Executive Information Systems (EIS). Although these systems are supported by the data extracting and managing capabilities of query languages, tools to develop such systems are not readily available.

## 3.3.3 Classification Based on Functions

Hall (1994) identifies four basic types of client/server functions, namely data file services, remote procedure call services, database services and enhanced c/s capabilities. The conversational and peer-to-peer configurations are also identified.

*Remote procedure calls (RPC)* : During RPC the application client program issues a call to a subroutine. The RPC software intercepts the call and routes it to the location of the subroutine to be executed. RPC software at the location of the subroutine is used to cause execution of the subroutine and then route the results back to the application client program. Simple RPC systems only locate a subroutine and transfer it, while more complicated RPC systems cause the subroutine to be executed where it resides and route only the results back to the application client.

*The conversational paradigm* allows a process to initiate a conversation with another process. Both can send messages and replies, or terminate the session. The initiating process and the server process are in a client/server relationship. Conversational capabilities are provided by enhanced client/server products, like TUXEDO.

*Enhanced C/S processing* adds the following features to the basic structure: application servers may be placed on any system in the network and the load is automatically balanced among the servers, performing the same service. Multiple application servers may be active on the same or different machines. Requests may be routed to a particular server, but the application program is never aware of the location of the server. Multiple database systems may be used in the system while new types of databases can be added or existing ones may be changed.

### 3.3.4 Implementation Approach

Client/server computing covers a broad spectrum of implementation approaches, varying from simple file transfer, applications programming interface, GUI-based systems to peer-to-peer applications integration (Marion, 1994).

*Simple file transfer* is the least complex approach. It consists of basic file transfer from a server to a client on request. The client and server may be independent applications running on different platforms.

*The application programming interface (API)* is a more complex approach. The client/server relationship in this situation is based on an application-to-application interface between the host application and a PC client.

### 3.3.5 Client/Server Architectures

Client/server systems can be designed to use any combination of distribution. Two-tier, three-tier or an enhanced configuration can be implemented (Eckerson 1995).

#### 3.3.5.1 Two-tier model

Initially client/server applications were built on a two-tier architecture that was designed to support 15 to 20 users and to run non-mission-critical functions. Most of these have been built with GUI development tools that packs all the code for user interface, application logic, and services at the Windows-based PC. The client then issues SQL calls to the server across a local area network. In a two-tier deployment architecture the client portion of the application runs on a desktop PC or workstation, and the server portion runs on a server machine across the network. Encouraged by the success of these systems, developers increased the number of users, functions and data sources supported by these applications.

#### 3.3.5.2 Three-tier model

This model extends the previous model by adding a middle tier of intermediate servers to support application logic and distributed computing services. The middle tier is critical for providing location and migration transparency. The three basic elements of an application are presentation, functional logic, and data. The presentation refers to the user interface, functional logic to tasks and rules and data refers to the information the business accumulates and that have to be accessed and manipulated. In a three-tier deployment architecture, the desktop PC handles the application's presentation component, the intermediate server supports functional logic and services, and the back-end server handles data processing. Now the physical environment mirrors the logical application architecture. This creates an inherent synergy that holds many advantages. Figure 3.4 illustrates.

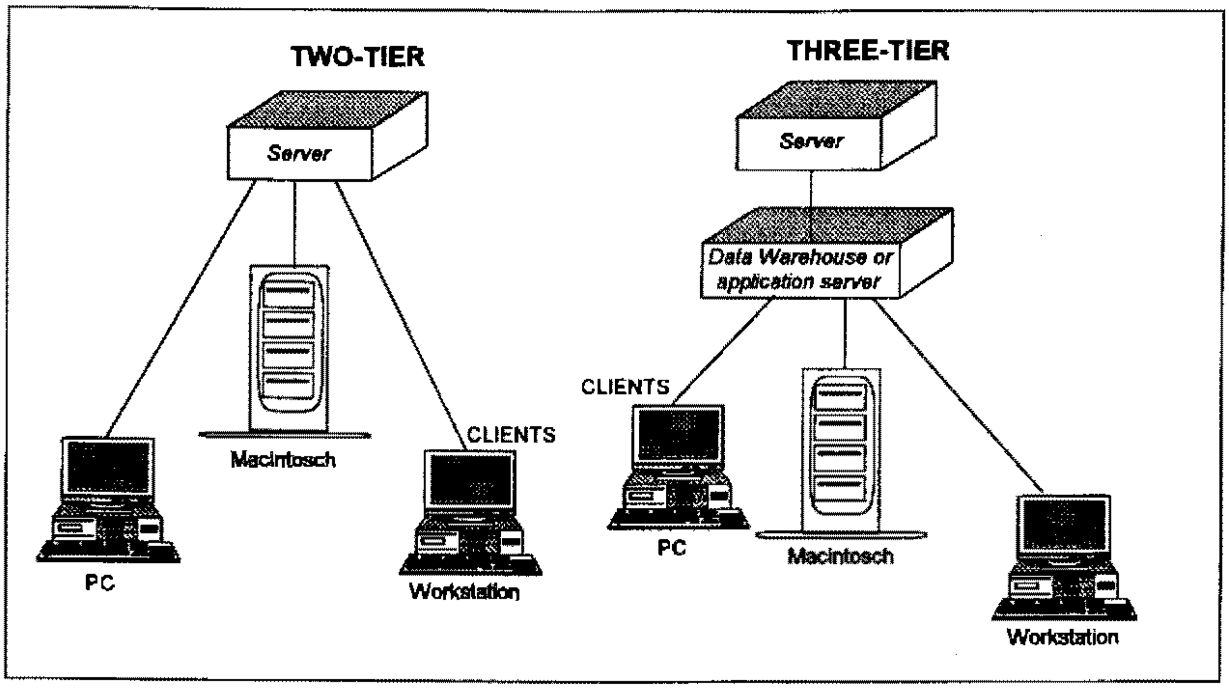


**Figure 3.4** *Two-tier and three-tier approaches to client/server (Watterson, 1995.)*

The three-tier architecture strives for tier-independancy. A key issue of the architecture is to ensure that each of these application elements is a separate, independent component. In other words it must be possible to change any of the three components without the others being affected.

## 3.4 Summary

In this chapter, the basic components of client/server systems were identified. These components namely the client, the server, and the network will be discussed in more detail in Chapter 4. Various dimensions of client/server computing were discussed as well as various ways of implementing it. First, it was classified according to the area where most of the processing is done, secondly, according to its support functions in the organization. A third classification is based on the implementation approach. The architectural topologies of these systems may also differ.

# CHAPTER 4

# Technological Components of Client/Server Systems

## 4.1 Introduction

Developing client/server applications that are compliant to open systems requires a thorough knowledge of the environment as a whole. The different technology components of the client, the server, the network and the application will be discussed in this chapter. Having presented an overview of the client/server components in the previous chapter, this chapter considers each of these in terms of representative technologies.

## 4.2 The Client

In Chapter 3, section 3.2.1, the client was introduced. Basic components of the client workstation are the basic hardware and the application software, such as the operating system, the database connectivity software, various applications, tools and a GUI as summarized in figure 4.1.

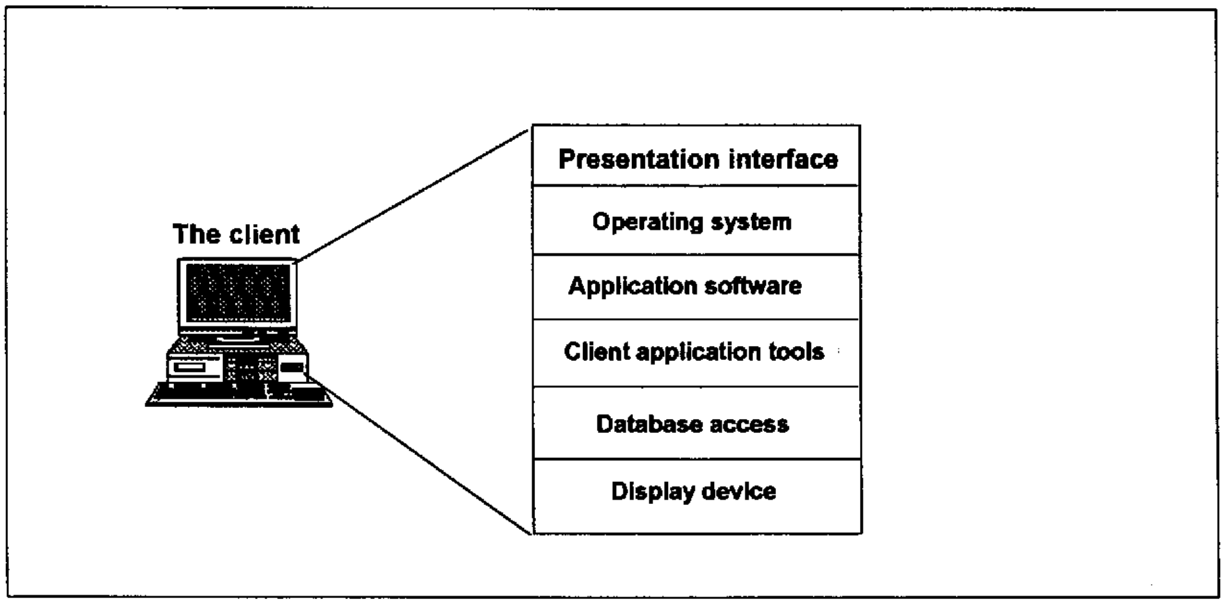


**Figure 4.1** *The client (Marion, 1994)*

The client functions usually make up the major part of the application. Special consideration should be focused on mapping end user functionality to client workstations. The client consists of hardware and software components.

### 4.2.1 The Client Hardware Platform

The front-end client machine will be responsible for the presentation and manipulation of data, and communication with the server. The client hardware must be powerful enough to run the presentation software. This will imply either a microcomputer or a powerful workstation, such as a UNIX-based workstation. The basic hardware components of the client will be the central processing unit (CPU), random access memory, direct access storage devices, one or more input device, and output devices in the form of printers or colour video monitors. A wide variety of hardware is available in proprietary, semiproprietary and open systems-compliant format to satisfy the needs of the customer.

As stated earlier, the accessibility of the PC resulted in its increased use, to such an extent that it is fast becoming the standard for the end user. The PC was first built and implemented for non-commercial use with a memory of only 640 Kilobytes, using MS-DOS as an operating system. However, technological advances increased the power and capabilities of the PC while decreasing the price, with the result that the PC has become accessible for commercial use.

From the old 286 AT CPU with a clockspeed of 10 to 16 MHz, the central processing units have evolved into the 386 (between 20 - 40 MHz), 486 (between 25 - 66 MHz) and the 166 MHz Pentium processor.

High-speed ports using the 16550 UART chip takes advantage of high-speed file transfers and downloading. The double and quad-speed CD-ROMs are currently priced competitively and offer twice the throughput of single-speed drives. Hard disks can be installed according to specific use, with a 1GB SCSI hard disk.

Notebooks with Pentium processor is frequently and conveniently used by businessmen. Although reduced set vs. complex instruction set hardware need to be considered, other aspects such as the following are more important in the selection of appropriate hardware for the client station: These include:

- The processing speed of the platform must be fast enough to support the user needs.
- Sufficient memory must be available to load and execute the GUI and the applications.

- Local storage must be sufficient for the specific client requirements.
- Does the platform support one or more operating system and which does it support?
- Proprietary platforms will support a single OEM while open platforms are based on industry standard technologies and are therefore considered superior.
- Will the platform cost-effectively meet the functionality needs of the client workstation?
- Is the PC equipped with a network or terminal emulation card for network connection?

The main options of hardware for the client is shown in table 4.1.

| Platform | Comments |
|---|---|
| IBM PC Compatible<br>Intel 286 (AT class) or compatible<br>Intel 386 or compatible<br><br>Intel 486 or compatible<br>Intel Pentium or compatible | Uses CISC microprocessors<br>Does not support the current version of Windows<br>25-33MHz systems are considered old technology compared with 100 MHz 486s and Pentiums<br>Standard in late 1994<br>Relatively expensive |
| IBM Power PC | Uses PowerPc RISC processor, designed to run AIX, OS/2, WindowsNT, Workplace OS, and Sunsoft Solaris software. |
| Apple MacIntosh | Based on Motorola's 68000 microprocessors |
| Apple Power Macintosh | Uses PowerPC RISC processor<br>Designed to run MacIntosh system 7, Windows, and DOS software. |
| UNIX workstation, for example Sun SPARC, HP 9000, IBM RS/6000 | More expensive than PCs.<br>Uses Motif as GUI |

**Table 4.1** *Hardware options for the client platform (Watterson, 1995)*

### 4.2.2 The Client Software Platform

Selecting client software is a major challenge, as there are literally hundreds of packages and options to choose from in this rapid-growing, competitive market. The minimum software will consist of the presentation interface (also called interface environments), operating system, application packages, application tools, application programming interface and database access tools, as illustrated in table 4.2.

| Component | Example |
|---|---|
| Presentation interface | Windows 3.1, Presentation manager |
| Operating system | DOS, OS/2, Windows95 or NT |
| Client application tools | 3GL, 4GL, Visual programming, object-oriented tools |
| Application packages | Spreadsheets, word processors, etc |
| Database access | ODBC |
| Application programming interface | |

**Table 4.2** *Client software components*

The software components will be discussed in the following section.

### 4.2.2.1 Presentation interface

The end user communicates and works with the system using the presentation layer. The presentation interface should conform to the general goals of the client/server architecture in an open distributed environment. Presentation management should be based on standards to allow systems to interact with users in a consistent style (Berson,1992). Presentation interfaces should also be portable across a wide variety of hardware platforms and able to interoperate with other open systems applications, although this goal cannot always be achieved.

The presentation interface could take the form of a character-based menu or a graphic user interface GUI. Windows and OS/2 environments support a user-friendly environment consisting of a WIMP interface. The WIMP interface presents information in windows, with icons representing tasks and application packages. Tasks displayed are selected by clicking the mouse button when the pointer is on an icon. De facto standard icons include a file cabinet for storage and a trash can for discard.

Earlier GUI interfaces followed guidelines for IBM's Common User Access (CUA) methodology. IBM has developed a new set of guidelines, CUA '91, that specifically addresses GUI standards called WorkPlace shell, designed for the Office version. These standards include drag and drop, control features such as container, notebook, slide and value set as well as standard dialogues for such operations as Open and Save.

The X Model is the architectural model of the X Window System, developed jointly by Massachusets Institute of Technology, IBM, and DEC in an effort known as Project Athena.

It is based on the client/server model and provides a network transparent communication protocol between the application and its presentation logic, high-performance, device-independent graphics, and a hierarchy of windows. Two GUIs, namely Motif and OpenLook, are based on the X Windows System.

As many GUIs are developed for a specific operating system, they are incompatible with other configurations. A standard GUI with a common application programming interface will be extremely beneficial to application development. Table 4.3 identifies GUIs that run on specific operating systems.

| OPERATING SYSTEM | GUI |
|---|---|
| MS-DOS | Windows 3.X |
| Windows 95 / NT | Windows 95 / NT |
| OS/2 | Presentation Manager |
| UNIX | Motif |
| DECs UNIX and VMS | DECwindows |
| MacIntosh | Mac |
| Sun Microsystems | OpenLook |

**Table 4.3** *Operating systems with Graphic User Interfaces*

Industry acceptance and hardware and software vendor support are two of the general requirements for the GUI. Others are (Berson, 1992):

*Portability:* Applications should be portable across various open systems platforms. A standard GUI should maintain a stable API (application programming interface) for every platform for portability.

*Standards compliance:* A de facto standard for an open system's GUI is the MIT X Window System. X Window serves as a model for the National Institute of Technology, ANSI and IEEE standards, and X/Open specifications.

*Development tools:* The GUI should be accompanied by a comprehensive set of development tools.

*Flexibility*: It should be able to accommodate new types of displays and other input/output devices.

*Internationalization:* This is essential in today's global markets, and will include the languages, numbers, monetary units of other countries.

*Platform independence:* To be truly open, the GUI must be designed to operate independently of the platform it runs on.

OSF/MOTIF is based on the industry standard X Window system, and runs on IBM's RS/6000, DEC's VAX, SUN's SPARC, MIPS 2000 and R3000, INTEL 80286, 80386, 80486, and Motorola 68020, 68030, 68040 and 88000 architectures.

The GUI use an applications programming interface (API) to link a GUI with an existing application. The GUI interactions are handled by the API.

### 4.2.2.2 Client operating systems

The operating system controls the operation of the computing system, performs file handling and memory management. Various operating systems are currently available for client workstations or microprocessors. The most popular operating systems used on client machines are:

- Microsoft's MS-DOS, IBM's PC-DOS or a DOS clone such as DR DOS from Novell (formerly from Digital Research)
- IBM's OS/2
- A UNIX-based operating system, such as USL's UNIX system, UNIX System V Release 4, IBM's AIX, and HP's HP-UX.

Most network operating systems can access these operating systems. It is important to remember that the operating system for the client machine determines the GUI. Windows 3.X is DOS-based, Presentation Manager is OS/2-based, and Motif and OpenLook are

UNIX-based. Windows 95 is an operating system in itself, although compatible with DOS. In the next section various operating systems will be discussed.

## MS-DOS

Microsoft operating system, MS-DOS, and IBM's PC-DOS are used as standard operating systems on PCs. MS-DOS 5.0 has improved memory management, data protection and on-line help. Apart from improvements like step-by-step debugging of AUTOEXEC.BAT CONFIG.SYS, upgraded DEFRAG 6.2 provides features such as disk compression, automatic memory optimization, virus protection, CD-ROM caching and tighter integration with WINDOWS. One of the features of DOS 6.2 is to allow Windows for Workgroups 3.11 users to operate in the Fast File Access mode (full 32-bit I/O). Recent MS-DOS versions have added UNIX-like attributes. Windows 3.x augments the capabilities of DOS with its own memory management routines, simulates multitasking operations and provides queued input messages that permit event-driven interaction.

## OS/2

IBM's 32-bit operating system, OS/2, provides multitasking and recognizes and uses the whole memory (there is no 640 Kilobyte limitation). It does, however, need substantial memory as the recommended minimum configuration is 6 Mbytes of memory and an 80-Mbyte hard drive. It uses the icon-driven interface WorkPlace Shell (WPS), and can run MS-DOS and OS/2 applications on the screen simultaneously. It is a multitasking operating system, but has been criticized as slow, difficult to use and unattractive by Windows supporters.

IBM plans to split the operating system into a client version and a server version (Forge, 1995). The client version will support enhanced multimedia and pen and voice-based applications, and the server version will provide multiprocessing and distributed computing capabilities across platforms running OS/2 and UNIX.

## WINDOWS environment

Fierce competition is instigated between operating systems by vendors adding extra utilities to their operating system for the standard price. The first version of Microsoft's Windows appeared in 1985 as an add-on to MS-DOS. In 1991 DR DOS pioneered value-added DOS by tossing in memory management, the PC-Kwik disk cache, and SuperStor disk compression. Novell continued the tradition with its DOS 7 which includes 386 multitasking, security and networking software.

One of the advantages of Windows-based operating systems is the large volume of application software that has become available for this environment. Spreadsheet users, for example, can accomplish more with Windows using Excel, combined with the word-processing features of MSWord than with a spreadsheet in an MS-DOS environment.

Windows 3.1 provided users with improved memory protection, an efficient file manager and a multitasking environment that can efficiently navigate among spreadsheets, e-mail, word processing and database applications. Three technologies provided by the Windows environment and supporting client/server systems are:

- Object Linking and Embedding: OLE can create compound documents which are a collection of objects with links to the software tool that created them and can be shared with other users. OLE allows the user to exchange data among Windows applications and transfer multimedia applications
- Dynamic Link Libraries: DLLs allow routines to be coded as modules and linked by applications
- Dynamic Data Exchange: DDE is used to exchange data between Windows-supported applications.

These technologies are now reviewed:

**Object linking and embedding (OLE)**

The interapplication protocol, OLE, is contained in the file MSOLE2.VBX. OLE 2.0 is a Microsoft published standard for allowing interprocess communications of application objects containing data to be exchanged. Compound documents consist of various documents in one compound document allowing the user interaction between these documents. A data entry form in a Visual Basic or Access application, containing an Excel spreadsheet and a Word document, in addition to Visual Basic or Access controls can be seen as examples of compound documents. Linking and embedding differs with respect to the storage of the linked/embedded data. The concepts linking, embedding, OLE automation and dynamic data linking will be reviewed.

*Object linking* contains a reference to an object (Menninger, 1994). When a range of spreadsheet cells are linked, the data are stored in another file. Only a link to the data and an image of the data are stored in the OLE control. When this document is changed from within Access (or Visual Basic), the original file is changed. When the linked object is changed outside the application, for example in the Excel spreadsheet, the changes will be reflected in the application. Linking is useful to maintain only one set of data that is accessed from several applications.

*Object embedding* places a copy of the object in the Visual Basic or Access form, in the OLE embedded object. When an object is embedded in an application, no other application has access to the data in the embedded object. Embedding will enlarge the size of the application, but will maintain integrity better than linking.

*OLE automation* is the ability to control one application from within another (Menninger, 1994). An example of this is a Visual Basic application that allows the user a selection out of various MSWord documents. At the click of the command button, a mail-merge is initiated between the MSWord document and Access table from the Visual Basic application. The Visual Basic application instructs MSWord to send completed envelopes to the printer or through electronic mail to recipients. The difference between this capability and linking and embedding is that, in this case, the application is directly controlling the foreign application rather than containing its objects. This aspect will be discussed in section 4.4.2.3.

*Dynamic data embedding* is a process by which keystrokes or strings can be sent from one application to another. However, no information concerning the communication could be sent back, such as error messages from the recipient to the sender.

DDE and OLE allows the user to perform similar functions, but there are also some fundamental differences (Cerutti & Pierson, 1993):

- The OLE software control transfers control to the other application, i. e. for editing, therefore the data appears in the same view mode that it would in the application. The image appears automatically in that view mode.

- Transfer of control with OLE allows the user to access the application using its native software.

Microsoft's operating systems are evolving to an object model supporting an integrated environment. Applications will now become components of the operating system, simplifying the construction of compound documents.

Windows-based environments provide the user with an easy user-friendly interface. Windows95 eliminates MS-DOS, as it is an operating system in itself. However, Windows 3.X is compatible with its newer counterparts.

Windows NT offers a consistent user interface and standard application programming interface (API) no matter what platform the software runs on. Windows NT is not an operating support environment as Windows 3.1, but a fully-fledged operating system. It was developed to run on both Intel and RISC processors. NT provides multiprocessing, and can be ported to various platforms. It contains a C language kernel and can support DEC's Alpha RISC platform, as well as HP's RISC machines and Sun's SPARC stations. OS/2 applications can run under NT, as well as Windows applications. Other advantages are (Cashin, 1993):

- preemptive multitasking
- virtual memory
- program size can reach 4 Gigabytes
- memory protection
- symmetric multiprocessor support
- security.

The built-in LAN Manager capabilities enable most communication tasks to be executed on the server. GUI interfaces remains on the client. NT also supports TCP/IP , DEC Pathworks, Netware, and it is compatible with POSIX.

**UNIX**

Unix may function as a client device, but it is more typically associated with server systems. UNIX supports multitasking, and as a multi-user environment it would rather be installed on a workstation than a micro. UNIX will be discussed in more detail in section 4.3.2.1 as a server operating system.

#### 4.2.2.3 The network operating system

In addition to its own operating system, a client in a LAN-based system or any other network system, must also run a portion of the network operating system (NOS). The NOS provides the software connection between the operating system and the network. The client NOS allows application programs to access the network through the client operating system and is also responsible for the management of the client's connection to the network. Network devices and storage are addressed as if they were physically attached to the client. NOS will be discussed further under network operating systems in section 4.4.2.1.

#### 4.2.2.4 Client application packages and development tools

The application software and development tools include all software used at the client workstation by the developer and the end user. One of the results of the growing importance of personal computers is the shift of emphasis from the mainframe to the desktop processor, and accordingly to software used on the desktop processor. According to Lewis (1995), industrial software is being driven by document-centric design, reusable component implementation based on industry standards, and end user programming. He identifies a number of trends which are of importance in today's information systems:

- The use of software will change in accordance with the technological shift. At the conservative end, changes may only include a shift to GUI-centric, desktop, stand-alone and productivity software.

- At the other extreme, changes may include more than isolated desktop software solutions and may include portable, interoperable, distributed software solutions.
- Interoperable, distributed, easy-to-use end user programming tools and standards are needed to build enterprise-wide architectures and applications.
- Business process reengineering, networked distributed hardware, together with numerous smaller complications are increasing the complexity of software. Object technology may be an answer to this.

Distributed software may be divided into various categories of software as shown in figure 4.2. The broad category of distributed software is first divided into the two categories, namely Decision Support Systems, and On-line Transaction Processing.

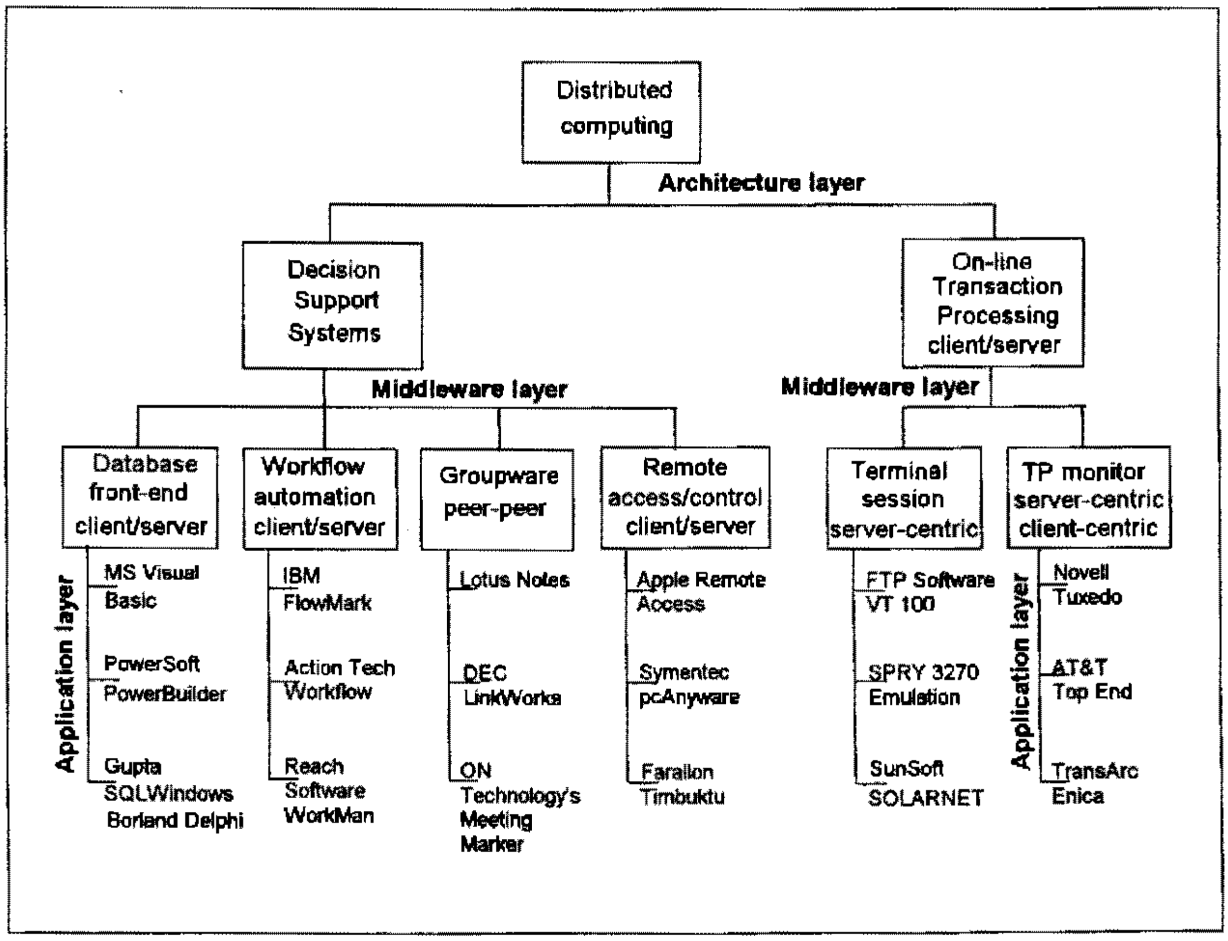


**Figure 4.2** *A partial taxonomy of distributed computing software (Lewis, 1995)*

Watterson (1995) categorizes traditional software by form, function and heritage. Tools classified by form are 3GL, 4GL, visual programming tools, or object-oriented programming tools.

The 3GL tools are standard programming languages, such as COBOL, C, FORTRAN and Pascal. These languages support the structured programming paradigm.

The 4GL tools are proprietary high-level languages which reduce the amount of coding necessary. Examples are Oracle SQL Forms, dBase V and Clarion.

Visual programming tools have emerged to facilitate rapid application development (RAD) and prototyping in the current GUI environments. They support WIMP interfaces and functions and options for quickly creating forms, reports and menus. Programs like PowerBuilder, Paradox for Windows, and Microsoft Visual Basic are examples.

Object-oriented tools support component-based software development. Tools are provided to support the developer to customize and reuse objects. Objects include items such as buttons, list boxes, pop-up windows and events. Microsoft's Visual C++, Borland's Delphi, and Microsoft Visual Basic are examples. The last-mentioned two appear to overlap and are converging. Client/server front-end tools are categorised in figure 4.3

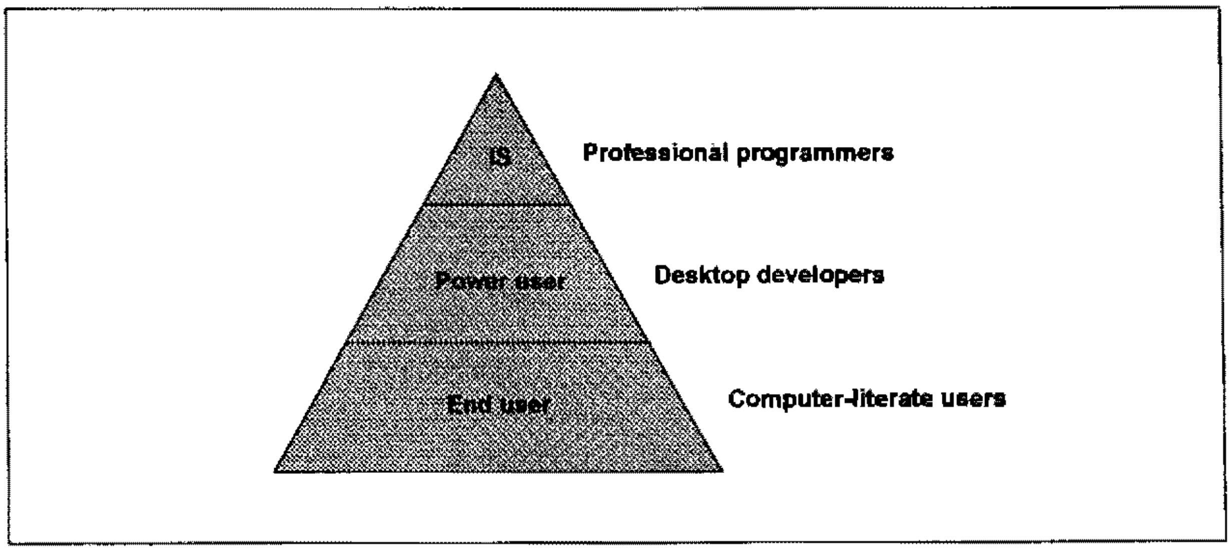


**Figure 4.3** *The market for client/server front-end tools (Watterson, 1995)*

Various packages started as desktop applications but have been rearchitected as front-ends to larger enterprise databases. Examples are Sybase, Informix 4GL for Windows, Oracle Forms. Some applications have been converted to workgroup and cross-platform applications. Mixed environments have been accommodated by software vendors by creating several versions of one package. For example, Microsoft Word runs under both DOS and Windows, WordPerfect runs under DOS and Windows as well as on UNIX workstations.

*End-user tools*

Traditionally end users would use standard applications such as word processing, spreadsheets and presentation tools for their basic administrative and office functions. Easy-to-use data query and reporting facilities are also available to the end user, as are workgroup applications, for example electronic mail. At the rate that end user programming is growing, the distinction between professional development and end user development is shrinking. Database applications are handled with equal efficiency by some end users than by the professional database developer. Applications and application tools are discussed in detail in Chapter 5 in section 5.5.

*Database Access*

Relational database structures have become a de facto standard for client/server computing. The DBMS organizes the data for accessibility via application programs. The physical organization of data is managed by the data storage system, and the logical storage is managed by the DBMS. The DBMS organizes, stores, retrieves and relates data components. Various commercial packages are currently available.

Proprietary software which do not comply with database open standards cannot be accessed by any tool other than that proprietary product line. Up to the move to open systems, database vendors sold front-end client tools that worked only with their databases. Fortunately client tools are becoming more flexible and interoperable, and middleware facilitates the connections. Middleware will be discussed as intermediate software. However, not all stand-alone personal computer databases are suitable for client/server systems, because such a database may not necessarily incorporate a distributed database system. Database system repositories will reside largely on the server side, although it may be played on the client if necessary. ODBC does, however, support cross-platform interaction.

Standard Query Languages (SQL) are used to access the relational database. ODBC support and larger database management systems will be discussed as intermediate software and middleware in Section 4.4.2.2 as they do not reside on the client, but as an intermediate interface between the client and the server.

The client application development environment comprises the presentation layer software, the graphic user interface, operating system and application development tools. One important component must, however, be added to provide the client/server system with full functionality. That component is called middleware. It does not reside only at the client station, but spans the entire environment, namely the client, network, and server and it accesses the application. Forge (1995) describes it as sitting between the application halves and on top of the communication protocols. Middleware was introduced in section 3.2.4, but will be discussed in more detail in section 4.4.2.2 as part of network software.

## 4.3 The Server Platform

The platform upon which the server system operates will determine much of the functionality, power and support it will give the end user. The back-end server manages the data resource, it stores, retrieves, and protects data. It also handles data management tasks, such as file and record locking, rollback and audit trail. The database server may be a powerful PC LAN database, a mainframe file or an Oracle or Sybase database running on UNIX. The server could also be a device server, superserver, a micro/server or a database server machine. A 486DX2/66 processor would be adequate for a file server with about 40 clients on a NetWare-based LAN; larger networks may require the extra power of a Pentium chip. 4 GB Fast SCSI-2 hard disk will give enough disk capacity and a UPS, for instance the 800 VA UPS, gives protection from power failure. Management software is available to optimize and troubleshoot the network. The perfect application server must have a state-of-the-art fast-storing processor in a system that supports multiprocessing, has significant processing power and can consistently run uninterrupted while constantly directing and managing the system's data and hardware. However, as technology continues to evolve, in today's terms a effective processor is a 166MHz Pentium or a RISC processor such as Hewlett-Packard's PA-RISC line. In a client/server approach, database management, manipulation, security, plus some logic are executed on the server.

The database server is simply a large application running on top of an operating system (Forge, 1995). As not all servers runs on all operating systems, and the operating system affects the performance and portability of the entire system, the choice of an operating system is crucial to the database server. The choice of the operating system is influenced by the hardware. The components of the back-end server are closely integrated and selection of the components should be made in tandem. However, the important facilities that it will provide to the client/server system are data storage, a database management system, and an operating system for control and communication software. Figure 4.4 illustrates these components.

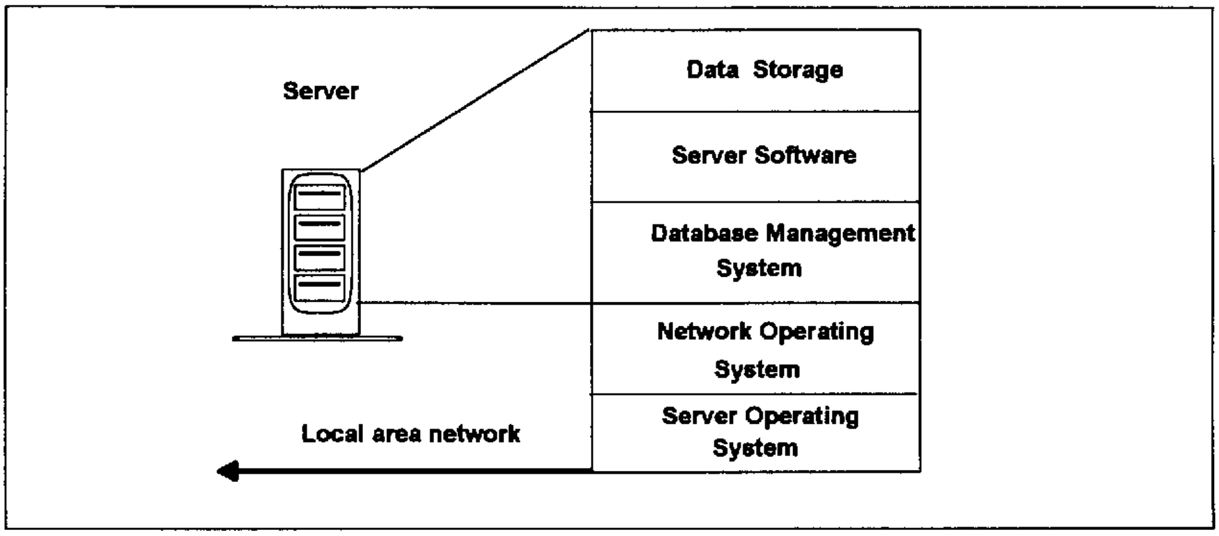


**Figure 4.4** *The server components*

The server will require special software to handle the requests of the client. The server renders the following types of services:

- data file services
- remote procedure call services
- database services
- enhanced C/S capabilities.

Today networks use traditional file servers, supported by a combination of operating system and network manager software. Two types of file services are available, namely file access and remote procedure call (RPC). File access services make files available to the network, and while the user may have to sign on as a specific user, knowledge of the physical location of the files may not always be necessary. Novell Netware file server is a popular example. File servers are implemented by software in the network or operating system so that file access requests are routed to the server system.. The server system accesses the requested file on behalf of the client system and returns a reply which includes the result of the access.

### 4.3.1 The Server Hardware Platform

The concept of a server serving several workstations or personal computers which originated as organizations needed to share expensive peripherals, such as laser printers, CD-ROM readers, fax machines, as well as software. The basic server hardware can be in the form of a microcomputer, superserver, database machine or fault-tolerant machine. Dewire (1995) identifies 6 types of servers, namely

- file server
- application server
- data server
- computing server
- database server
- communication server

each of which will be reviewed.

#### 4.3.1.1 The file server

File servers manage a workgroup's applications and files to be shared by the group. They retrieve and transport large amounts of data across the network. Database functions, such as security, and locking concepts are executed at the client, as there is no single engine at the server end to support it. All data must be moved across the network before filtering, sort or merge functions can be activated.

#### 4.3.1.2 The application server

An application server serves as a host replacement or as the actual host. Applications are downsized from the host and used by workstations.

#### 4.3.1.3 The data server

The data server is used only for data storage. It will be used in conjunction with a computing server and may be used by more than one computing server. It actually functions as a data warehouse. It does not perform any application logic, but only data management functions, such as data validation and data searches. This kind of server requires a large amount of memory for storage as well as substantial hard disk capacity.

#### 4.3.1.4 The computing server

The computing server executes the computing part of data extractions from data servers. Some of its functions are to transport requests for data from the client to the server, and to forward the extracted data to the client. High-performance capabilities and large amounts of memory are required.

#### 4.3.1.5 The database server

This type of server is typically used in a client/server configuration. The application may be run on the client, partly by the client and partly by the server, or entirely by a server. The database server accepts requests for data, retrieves data from its database and passes it to the client. The data management function resides on the server and the application logic and presentation logic resides on the client. Database servers may contain a lock manager, multi-user cache management and scheduling. Queries are answered by processing a SQL statement and sending the results to the client.

### 4.3.1.6 Communication server

Communication servers provide gateways to other LANs, networks, mainframes and computers. Requirements are multiple slots and fast processors to translate networking protocols.

### 4.3.1.7 Features of server machines

Unique features and facilities provided by servers are multiprocessing, multithreading, disk arrays, memory subsystems and handling redundant components. Each one will be explained only briefly.

*Multiprocessing:* When a computer system contains more than one processor, multiprocessing is possible. Vendors are including multiprocessing in the hardware to support symmetric processing or multiprocessing. In symmetric processing a task is dynamically assigned to any processor, and resource usage is maximized. This capability must, however, be supported by either the network or the server operating system. Operating systems claimed to support it are Banyard VINES SMP, Santa Cruz Operations SCO MPX and Windows NT. Figure 4.5 illustrates this.

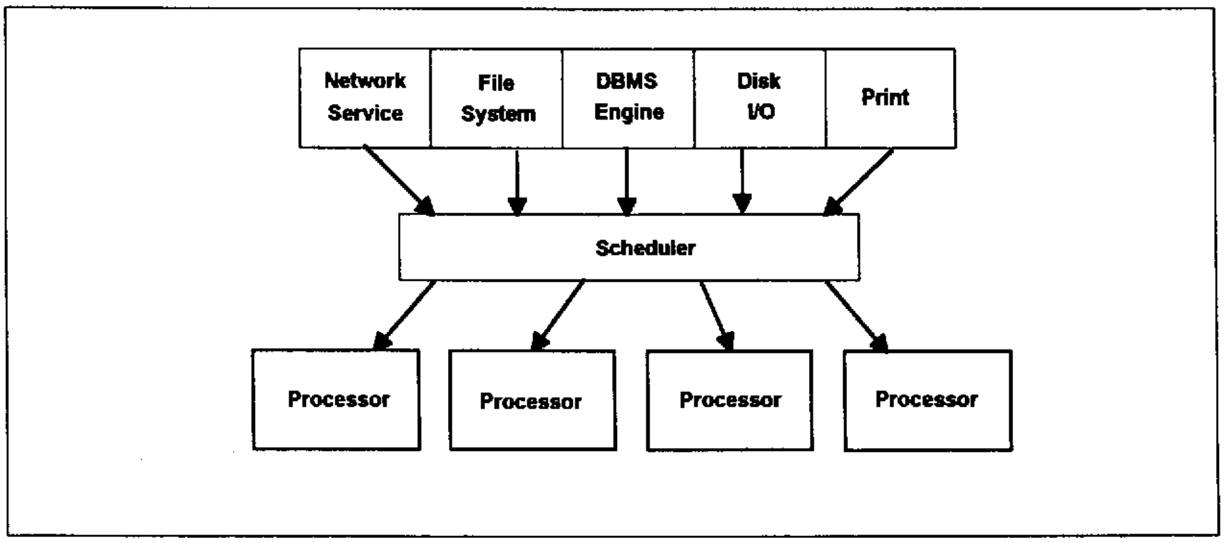


**Figure 4.5** *Symmetric multiprocessing*

*Multithreading:* Multiprocessing supports multithreading which can be explained as the concurrent execution of multiple tasks. Older operating systems achieved multithreading by dividing the basic process into multiple processes. The multithreaded environment breaks it into independent executable tasks or threads, which can then be executed concurrently.

*Disk arrays:* RAID or redundant arrays of inexpensive disks are standard on superservers and optional on other platforms. They usually include a 486 or equivalent file server and software to control access to different drivers. RAID can transparently recover from the failure of any one of the disks, and allow one of the drivers to be replaced while the system is on-line. Various methods of implementation are possible, but fall outside the scope of this investigation.

*Memory subsystems*: ECC (Error correction code) memory is a memory subsystem design that automatically corrects single-bit errors and corrects multiple-bit errors. Data corruption of data travelling within the server is prevented by parity checking and error-correcting code.

*Redundant components*: Disk drives, power supplies, fans etc. are options on micro/servers and built into superservers.

This short discussion does not cover all server features, but only those that are of importance in this environment. This study does not cover the server and network architecture in detail but only with reference to the construction of a client/server system. Consult Berson (1994) for detailled information on server hardware.

### 4.3.2 The Server Software Platform

The server stores large amounts of data that are to be accessed, retrieved, manipulated and stored again. The essential software components are the operating system, database management system and network software. As stated in section 4.3, operating system and DBMS decisions should be made in tandem as the operating system determines which DBMS can be used and, furthermore, it influences the entire client/server system's performance and portability. Benchmarks are formal suites of tests which run under strict basic rules to test the performance of the DBMS. The original transaction processing benchmark, TP1 has been supplemented by TPC-A, TPC-B and TPC-C (TPC=Transaction Processing Council). Server software categories are listed in figure 4.6.

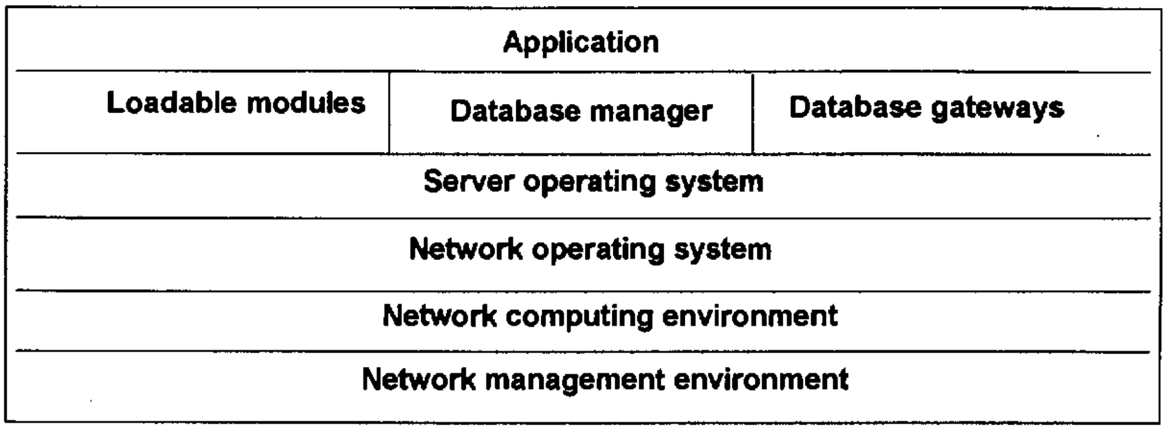


**Figure 4.6** *Server software (Dewire, 1993)*

Network software will be discussed in Section 4.4.2.

### 4.3.2.1 Server operating system

The operating system used at the server is of crucial importance. Some of the basic issues in the choice of the operating system are:

- Capacity and performance: Does the server need to support a hundred, thousand or more users concurrently?
- Interoperability: The platform and operating system must be able to connect to existing and maybe parallel systems in the enterprise.
- Scalability: As the business may expand and reorganize, the system must be able to support the expansions.

Servers can either run a mainframe operating system or any other proprietary operating system. The various operating systems will be reviewed briefly. The midrange or minicomputer market, is dominated by three vendors: IBM, Digital and HP. The IBM/400 has its own operating system called OS/400. Midrange systems from Digital use VAX/VMS (Virtual Address Extension/Virtual Memory System), Alpha/VMS or Open VMS. HP3000 uses the MPE operating system. Only a few representative server operating systems will be discussed. These are either 32-bit or 64-bit operating systems. Microsoft's Windows NT/AS, Novell NetWare 3.X and 4.X, IBM's OS/2 and Taligent (developed by Apple, IBM and HP) are also suitable as server operating systems.

**UNIX**

Although UNIX is considered more 'open', there is no UNIX standard. As a server-based operating system, Unix exists in many forms. Care must be taken to ensure that UNIX will work with individual hardware platforms and with the client and server software used. There is IBM's Advanced Interactive Executive (AIX), Sun's Solaris, OSF's OSF/1, DEC's Ultrix, HP's UX, USL's System V Release 4 (SVR4), to name but a few. Unix Sun Microsystems, Novell, IBM and HP are jointly developing a common UNIX operating system environment for all their hardware platforms. The joint UNIX effort intends to allow UNIX applications also to also appear and work the same on PCs and workstations from multiple vendors. Programmers will benefit most from consistent UNIX APIs, as it will be much easier to port applications among different hardware platforms.

Different versions of UNIX may use the same system calls but implement them very differently. Nearly all UNIX implementations adhere to industry standards, but they have proprietary extensions that add enough value to justify a custom version for a specific hardware/software platform. Incompatibilities between UNIX versions are also caused by variations in the original source code, but it is still easier to port a UNIX application from one version to another than to port an application from one proprietary operating environment to another. As UNIX can also be used as a client operating system, it was introduced in section 4.2.2.2.

**OS/400**

This proprietary operating system runs on IBM's AS/400 midrange computer and is often involved in 'rightsizing' schemes that present a transitional approach to downsizing and client/server. The mainframe becomes the repository, connected to the PC client station.

Interoperability capabilities on AS/400 machines include TCP/IP, Ethernet, X.400 messaging, X.25 interface, plus distributed support via DCE.

**OS/2**

OS/2 is introduced in section 4.2.2.2 as a 32-bit, multitasking operating system with strong support in the developer environment. It functions primarily in a server role, although nothing precludes its use as client. Some of its advantages are:

- performance improvements of 32-bit versus 16-bit environments
- demand paging provides more efficient memory use
- the Workplace shell provides enhanced user interface
- file support for object-oriented features has been incorporated via extended attributes
- supports MS-DOS, Windows and DCE.

**VMS**

DEC's multitasking operating system is a mature system offering robust services and solid vendor support. Impressive functionality and stability are offered, but it may be too pricy for the standard client/server system. The Advantage-Server family provides a full set of hardware, software and service options ranging from a desktop MicroVax to a mainframe. DEC's older PCSA (Personal Computing System Architecture) is based on the client/server model. The server can be any VAX, MicroVax, PCLAN/Server running server software called "VMS Services for PCs". The client can be any DOS-based client using software entitled "DECNet PCSA Client for DOS". Client integration for OS/2 and Macintosh PCs are also planned.

**Sun Solaris 2**

Solaris provides improved distribution services, heterogeneity and high security. It also contains standardized Federated Service Interfaces (FSI) which allow integration of SUN's ONC+ package with filing, naming and security services of other vendors.

**POSIX**

POSIX is not an operating system but a set of standards. If a user wants software to run on the widest set of hardware and operating systems, POSIX compliance is essential. The standards emerged in the 1980s from the Institute of Electrical and Electronic Engineers (IEEE) and many are endorsed by ANSI and ISO.

### 4.3.2.2 Server database management software

All organizations keep and use data in the form of information. It forms the basis of enterprise-wide information management, decision-making and transaction processing. Until the 1980s, structured programming and 3GL languages were the standard programming environments used. That implied a programming language such as COBOL, accessing a hierarchical or network data model, storing data on sequential files. These database models required a program written in, for instance, COBOL by a programmer to access them. With the development of relational and object-oriented data models, the database's logical design has been segregated from the details of the physical design and storage. Therefore, even nonprogrammers can easily design and use them by means of relational applications such as Access, Paradox and Approach. Database access was illustrated in figure 3.4.

The relational model (Codd,1969) sees all data as tables, consisting of rows (records) and columns (fields). Object-oriented database management systems (ODBMS) also gained momentum. Dominating database management systems are shown in table 4.4 as identified by Bear Sterns and Co.

| Database | Market share |
|---|---|
| Oracle | 43.8% |
| Sybase | 20% |
| Informix | 13.9 |
| Comp Assoc | 11.1% |
| ASK/Ingress | 7.7% |
| Progress | 2.1% |
| Gupta | 1.5% |

**Table** *4.4 Market share of DBMS in 1993*

SQL is a non-procedural query language designed to work with relational databases, for retrieval and similar actions. SQL is a mathematical formalization based on first-order predicate logic of the relational algebra operations. Unfortunately there is a host of SQL languages. Standards organizations, such as X/Open, the American Standards Institute (ANSI), and the International Standards Organization (ISO), are constantly updating their versions of SQL standards. SQL will also be discussed under the heading middleware (APIs).

**Important DBMS Engine features are:**

*Concurrency and locking*: Multi-user database systems need a type of locking scheme that prevents users from interfering with each other. Various locks are used, such as database locks, file locks, table locks, page locks. Shared locks allow concurrent reads, while exclusive locks allow access to only one user at one time.

*Data types*: Despite the SQL standard, DBMS still varies several key areas, namely date and time, money or currency, and Binary Large Objects (BLOB).

*Data integrity*: Integrity is related to accurate, reliable data. Various levels of integrity are sustained while a database is being used.

*Indexes* are defined on unique key fields and used to enhance query performance. According to the relational table, key fields should never contain null values. However, not all servers support non-null key fields and composite keys.

*Domain integrity* refers to the range of legal and logical values defined for a field. Domain integrity is not enforced in a standard way. In other words it will have to be maintained in multiple applications.

*Referential integrity* precludes your deleting a record in the master order table unless all related rows in the tables are also deleted. There are two strategies for enforcing referential integrity: by special stored procedures called triggers or by the data dictionary (system catalogue). The SQL standard supports both approaches.

*Backup and recovery facilities*: These are standard features of database servers, but implementation varies widely for several reasons. The procedures depend on the operating system, reflect robustness for OLTP, and furthermore, they reflect backup and recovery design decisions made by DBMS designers.

## 4.4 The Network Platform

The network was introduced in Chapter 3, section 3.2.3. As previously mentioned, the network transmits requests for data from client to server, routes requests for data among servers, and supports internetwork communication. Networks are complex systems. Networking will be discussed under the headings hardware, including the standard architectures, topologies and protocols, and networking software, including middleware.

### 4.4.1 The Network Hardware Platform

A network architecture defines the protocol, message formats, and standards used within that architecture. Standard architectures, network topologies, and major protocols will be discussed.

#### 4.4.1.1 Standard architectures

A network combines different computing systems, be they stand-alone personal computers or large mainframe computer systems. In a network, heterogeneous equipment containing different architectures have to be connected. Many standard architectures have developed over the years. The OSI model, TCP/IP, SNA and other models will be discussed.

*OSI Reference model*

The reference model of Open Systems Interconnect (OSI) is a published standard for networks. Developed by the International Standards Organization (ISO), the seven-layer model covers all aspects of networking from physical connection to sophisticated application support. Table 4.5 illustrates the levels with their functions:

- level 1 specifies the physical access protocols and connection required, for example Ethernet, FDDI, ATM, token ring and repeaters

- level 2 supports the packet level addressing, bridges, ODI and NDIS (NIC multiprotocol support)
- level 3 addresses the network operating systems and routers
- level 4 address the gateways and transport protocols such as TCP/IP, SPX/IPX, and X.25
- level 5 supports various protocols, named pipes, NetBios emulation, and most of IBM's APPC
- level 6 namely the presentation layer, handles network security, character-code translations, operating systems, RPCs (Remote Procedure Calls) and CORBA
- level 7 contains tools and languages that support the application programs, such as SQL and APIs.

| | Level | Function |
|---|---|---|
| 7 | Application | Support for application programs |
| 6 | Presentation | Code and format translations |
| 5 | Session | Dialogue management between users |
| 4 | Transport | Quality control of packet transmission |
| 3 | Network | Internetwork routing |
| 2 | Data link | Creation of frames |
| 1 | Physical | Transmission of signals |

**Table 4.5** *OSI reference model.*

The OSI model deals primarily with the interconnection between systems and provides a generalized view of a layered network architecture. Various LAN architectures use the OSI model as a reference.

*SNA*

Mainframe vendors developed their own networks with protocols before LANs existed. Two important ones are SNA and DECnet. Binary Large Objects (BLOB) from IBM uses a seven-layer architecture similar to the OSI model, although without the one-to-one correlation between the layers. The goal of the OSI model is to provide standard information exchange protocols for communication between autonomous, possibly different architectures. SNA, on the other hand, is designed for the exchange of information between network nodes that belong to a single architecture. The architecture is one on which IBM builds its various program offerings. SNA handles connections between users in a network in such a way that the underlying physical aspects of how the information unit is routed between users throughout the network are transparent to the user. The existence of a single architecture allows IBM to tailor both hardware and software network components to achieve maximum efficiency and performance. SNA supports networking facilities including distributed processing for IBM systems only.

*TCP/IP*

The Transmission Control Protocol/Internet Protocol (TCP/IP) includes a set of networking standards that specifies details of computer communications as well as conventions for networks interconnection and network traffic routing. TCP/IP has resulted from research funded by the Defense Advanced Research Projects Agency (DARPA). This network protocol supports multivendor interoperability. It differs from the previous network protocols in the following ways:

- Network technology independence: TCP/IP is vendor-independent and specifies how information transmission units are to be transmitted on a particular network.
- Any-to-any interconnection: Communication between computer systems are facilitated by giving a unique address, recognized throughout the Internet, to each system.
- Source-destination acknowledgement: TCP/IP protocols support end-to-end achnowledgement between the source and the destination of the message.

TCP/IP is a four-layer communication architecture consisting of 4 levels of implementation as illustrated in table 4.6

| |
|---|
| Application layer |
| Reliable Stream (TCP) User datagram (UDP) |
| Internet Protocol |
| Network Interface services. |

**Table 4.6** *TCP/IP architecture*

TCP/IP can be described as a hierarchy, built on a physical network interface, that provides communication interfaces to the network hardware. The IP protocol in figure 4.6 is a connectionless service that deals with the delivery of information packages. The IP protocol includes rules on how packets should be processed, as well as aspects such as the handling of error messages. The TCP layer guarantees delivery, handles data concurrency and sequencing, error checking, retransmission and connection to other applications on other systems.

***Other Models***

There are various other proprietary models, such as Digital Network Architecture (DNA) which is part of a suite of communication products called DECnet from Digital, Sun Microsystems's Network File System (NFS), and Xerox Network Systems's Internet Packet Exchange (IPX) and Sequenced Packet Exchange (SPX). For more information on this subject refer to Berson (1992).

#### 4.4.1.2 Network classification

A networks may be classified by its communication method, the method used to switch data between the nodes and their structure (network topology).

***Communication method***

Point-to-point communication allows one node to communicate with another directly connected node. Ethernet is a point-to-point network. In multipoint or multidrop networks, all nodes share one line and time is allocated to each node. Networks also use different message forwarding methods. LANs are broadcasting networks where each node is connected to a common transmission channel so that a single message can reach every node. WANs are store-and-forward networks that receive a completed message into a buffer before transmitting it to its destination.

*Network switching*

Data must be switched between links and nodes by networks. Two techniques are commonly used, namely packet switching and message switching.

Packet switching networks (X.25 networks) divide the data into segments called packets which are transported across the network, and reassembled at the destination by software.

Message switching writes messages to the node's storage area. The destination must inform the sender that it wishes to receive the message, before it will be forwarded. E-mail uses this implementation technique.

*Network topologies*

The topology of the network consists of the structure of the nodes and links. Links may take a virtual or physical form, for example telephone lines, satellite channels and private links. Three popular topologies are illustrated in figure 4.7.

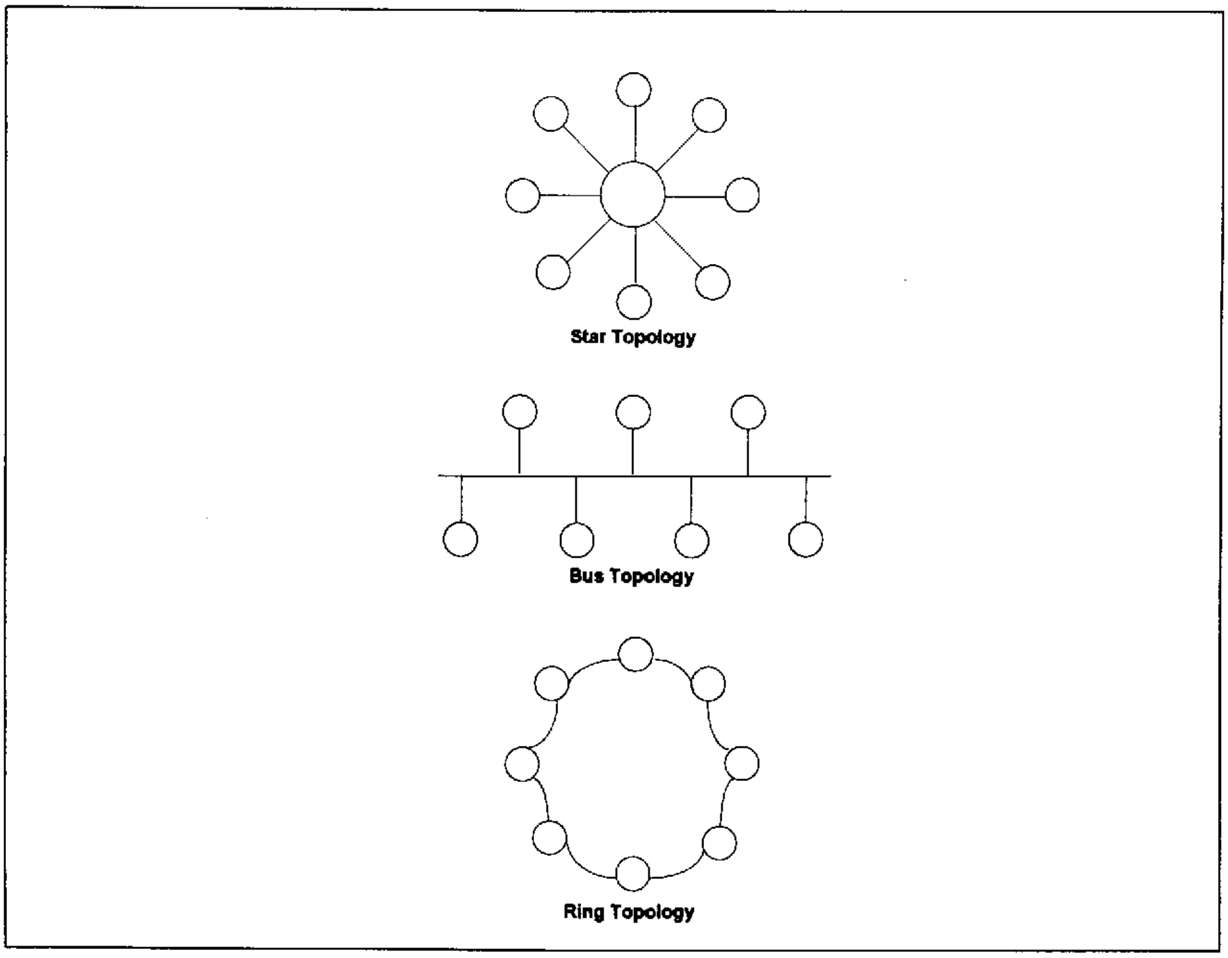


**Figure 4.7** *Network topologies*

#### 4.4.1.3 Local area networks.

The client in the client/server system will probably reside on a PC connected in a LAN system. The LAN consists of software, an operating system, networking software and hardware components, such as the network node (PC) , and connectors in the form of cables and NIC or LAN interfaces. LAN configurations are basically peer-to-peer and large-scale traditional file-server LANs. Peer-to-peer configurations are used for small workgroups and the traditional type of LAN for enterprise networks. Furthermore the LAN uses a protocol which can be described as a set of rules or standard methods for transmitting computer messages. Ethernet is an example of a protocol which uses packet switching. Traditionally Ethernet moved data at a rate of 10 Mbps (megabits or million bits per second), currently it is being revised for faster transmission, namely 100 Mbps and a full-duplex Ethernet at rates of 20 Mbds. Other standards include token ring (100 Mbps) and ARCnet (with 20 Mbps).

Watterson also refers to these protocols as the electrical topology, in addition to the physical topology illustrated in figure 4.7

### 4.4.2 The Network Software Platform

The network is supported by its own software consisting of an operating system, middleware, and other network software, i. e. network computing environment and network management environment. The PCs traditionally use a single-user operating system. It is therefore the LAN that must enable multitasking and provide multi-user facilities. These are provided by the operating system of the LAN. The network operating systems will be discussed below.

#### 4.4.2.1 Network operating systems

Network operating systems are complex software applications that include a host of utilities to control the network environment with functions such as multiprocessing, multithreading, and to assist the LAN administrator in adding and deleting users, configuring network resources, setting up logon routines, managing network security and monitoring network traffic.

*Novell NetWare*

According to a March 1993 survey, NetWare controlled 60% of the LAN operating system market. The product initially provided file and print-sharing services, but is evolving to supply more than just resource sharing. It also integrates diverse hardware and software technologies. However, NetWare 3.1 provides no memory protection, no preemptive scheduling and no virtual memory.

*Microsoft LAN Manager*

The same model was used for building LAN Manager, IBM's LAN Server and DEC's Pathworks. With OS/2 as the underlying platform, it supports multitasking and offers memory protection. On account of OS/2's strong GUI and extensive tool availability, building applications for a LAN Manager configuration is easy on this platform. DEC Pathworks provides additional network interfaces. Other Microsoft products previously discussed include Windows NT, Windows for Workgroups.

***Banyan VINES***

This product uses Unix as a platform and provides strong management services. Network capability is provided to diverse environments such as Ethernet, ARCNET, Token-Ring, Net BIOS and Omninet. It has, however, not fully penetrated the commercial market.

### 4.4.2.2 Middleware

Middleware was introduced briefly in section 3.2.4. As middleware span the client, the server and the network, it is referred to in each section, but is discussed in more detail in this section. Middleware is described as a glue, a mask, a route map, directories and dictionaries (Forge, 1995). It can also be described as key functions grouped together as illustrated in figure 4.8, to connect databases, applications, user interfaces and shared services.

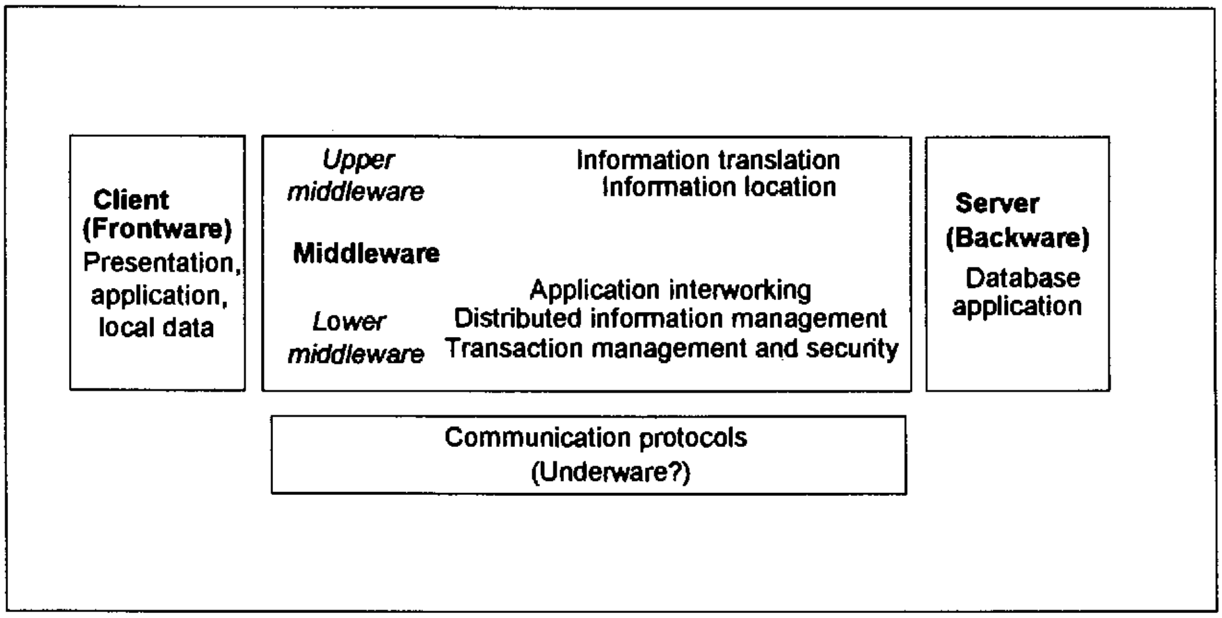


**Figure 4.8** *Middleware (Forge, 1995)*

Middleware will :

- locate remote resources, in the form of applications, databases, or data or documents and handle the interfacing between them
- manage the interactions and transactions between applications on different machines, including low-level shared services, such as printing, filing and security

- isolate applications from the network details and all mechanics of cross-platform transaction management.

The middleware component relieves the application programmer of the task of establishing the physical and logical connection to the target data source. Watterson (1995) defines three types of middleware :

- Network or messaging middleware which includes remote procedure calls (RPC) and interprocess communications (IPC), or tools to shield the programmer from the network details. A standard RPC has been defined by OSF (Open Systems Foundation) in conjunction with its Distributed Computing Environment (DCE).
- Database and applications middleware which constitutes a broad category including application programming interfaces (APIs) and gateways. Database middleware translates the incoming requests into database-specific commands. An API is a set of standard functions for interaction between applications and other applications, PC packages, services such as communications and operating systems. SQL gateways are proprietary software and also provide links between client programs and back-end databases.
- Copy management/replication server middleware is a growing category of middleware that is used to copy databases or parts of databases.

Middleware tries to solve the problem of many different desktop applications trying to access many different data sources. Without middleware, programmers have to write special code for each desktop application to access back-end data.

Figure 4.9 illustrates the functions of middleware in a client/server system.

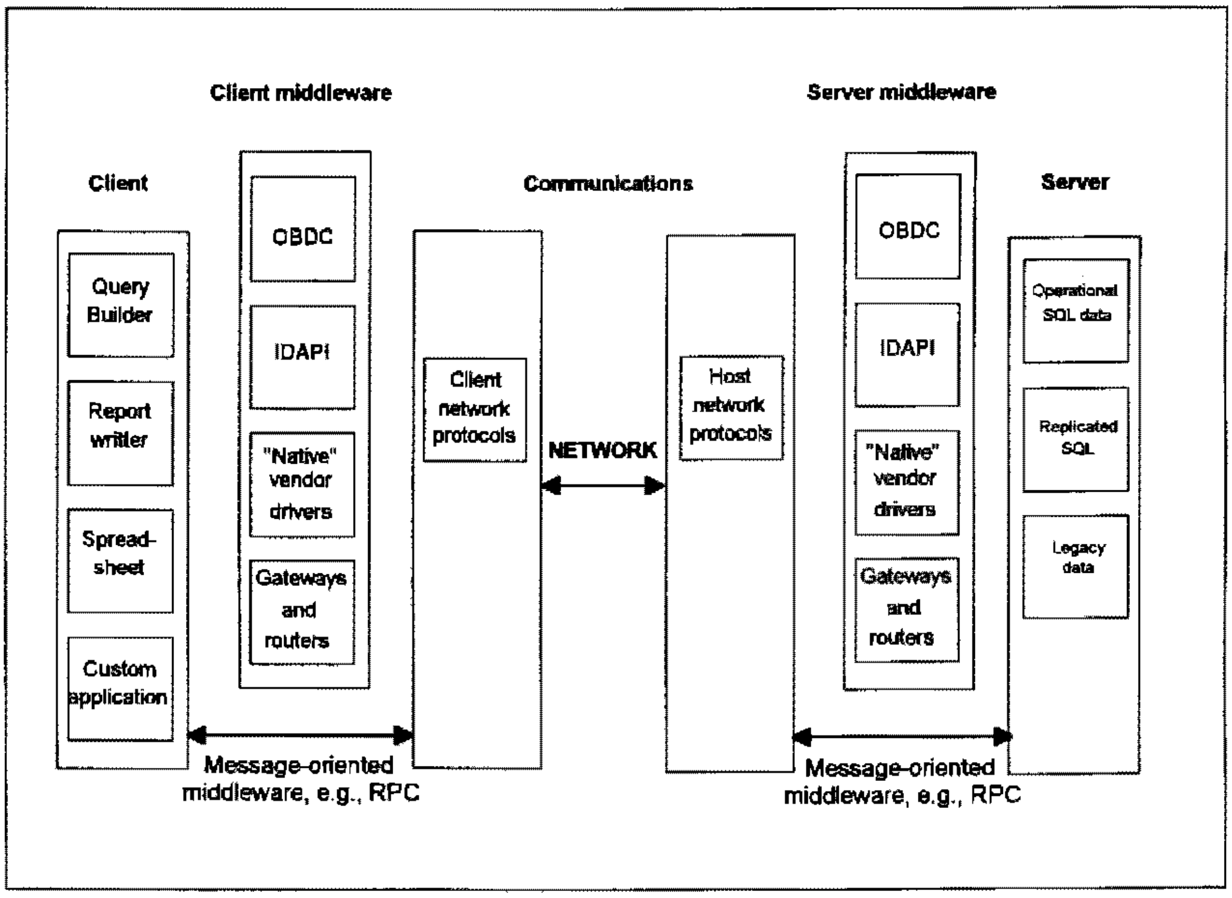


**Figure 4.9** *Generic Middleware model*

***Database middleware***

Requests are coded in a language understandable by the back-end database. The database manipulation language (DML) commands are passed to the back-end using a database API (application programming interface). The request may be coded in the form of a SQL statement as a remote procedure call. Using SQL gateways, the middleware black box is a SQL translator. ODBC (Object Database Connectivity) is described as a kind of generic SQL API popularized by Microsoft for Windows, but it is also available for UNIX and Macintosh computer systems. An ODBC driver is necessary to implement the connection.

Information Builders' EDA/SQL is another SQL gateway, that that consists out of both mainframe and front-end components, through which to over 100 back-end SQL and legacy data sources can be accessed. Most SQL vendors, such as Oracle and Sybase, sell SQL gateways that provide access either to SQL databases, or both SQL and non-SQL data sources.

***Network and messaging middleware***

This type of middleware establishes database connectivity using a traditional programming language (Watterson, 1995). It is also called application middleware. A standard RPC has been defined by OSF (Open Systems Foundation) in conjunction with its Distributed Computing Environment (DCE). This software routes messages between different platforms.

These middleware components pass messages to the remote systems through a host-language interface. The two kinds of RPC's are both database-specific, and constitute message-oriented and generic application-level RPCs. The application level RPCs are code-intensive, as they are written in a 3GL. They are, however, supported by the object management group in the form of CORBA. RPC technology relies on synchronous messaging.

Message-passing systems are asynchronous. Message-oriented middleware (MOM) are based on the same store-and-forward method as e-mail-systems. Figure 4.10 illustrates this.

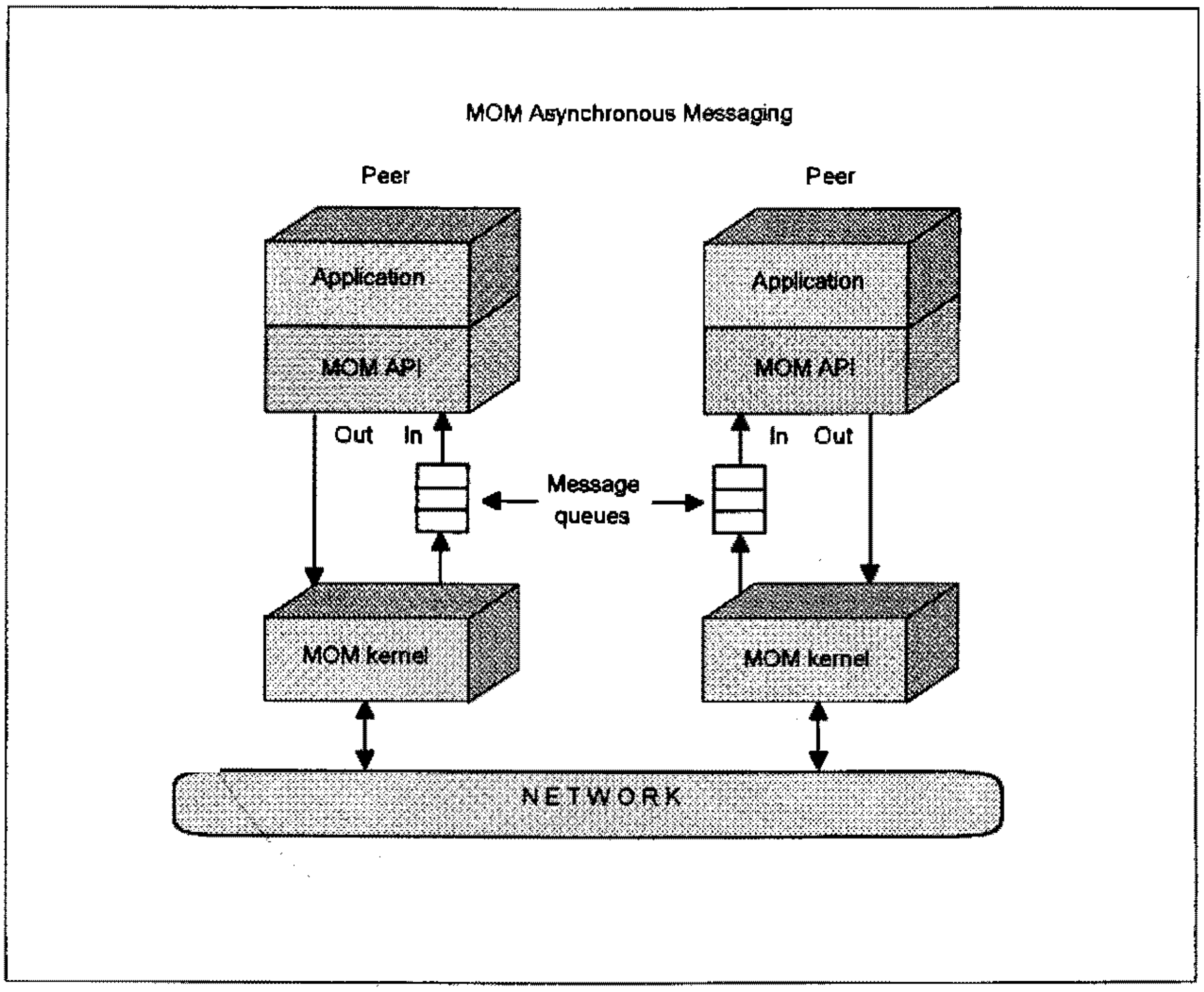


**Figure 4.10** *Message-oriented middleware (MOM)*

***Copy management / Data warehousing middleware***

This type of middleware is used to copy the whole or parts of databases when an event or threshold occurs (Watterson, 1995). The software can be programmed to replicate the copies at fixed intervals. Data warehousing is a strong growing trend in client/server systems.

Middleware can also be categorized into lower middleware, such as low-level software for sending messages from one PC to another, and upper middleware, such as high-performance TP-monitor middleware for OLTP. Two significant architectural approaches are:

- OLE / COM (component object model) from Microsoft, described in section 4.2.2.2
- OpenDoc, CI Lab's architecture for document-centric computing, with DSOM and CORBA (Common Object Request Broker Architecture).

Middleware has emerged as a piece of strategic industrial software, particularly of client/server systems. Desktop middleware, combined with compound document infrastructure will play a leading role in industrial software. Figure 4.11 illustrates the middleware layers in a database front-end architecture for a Windows environment.

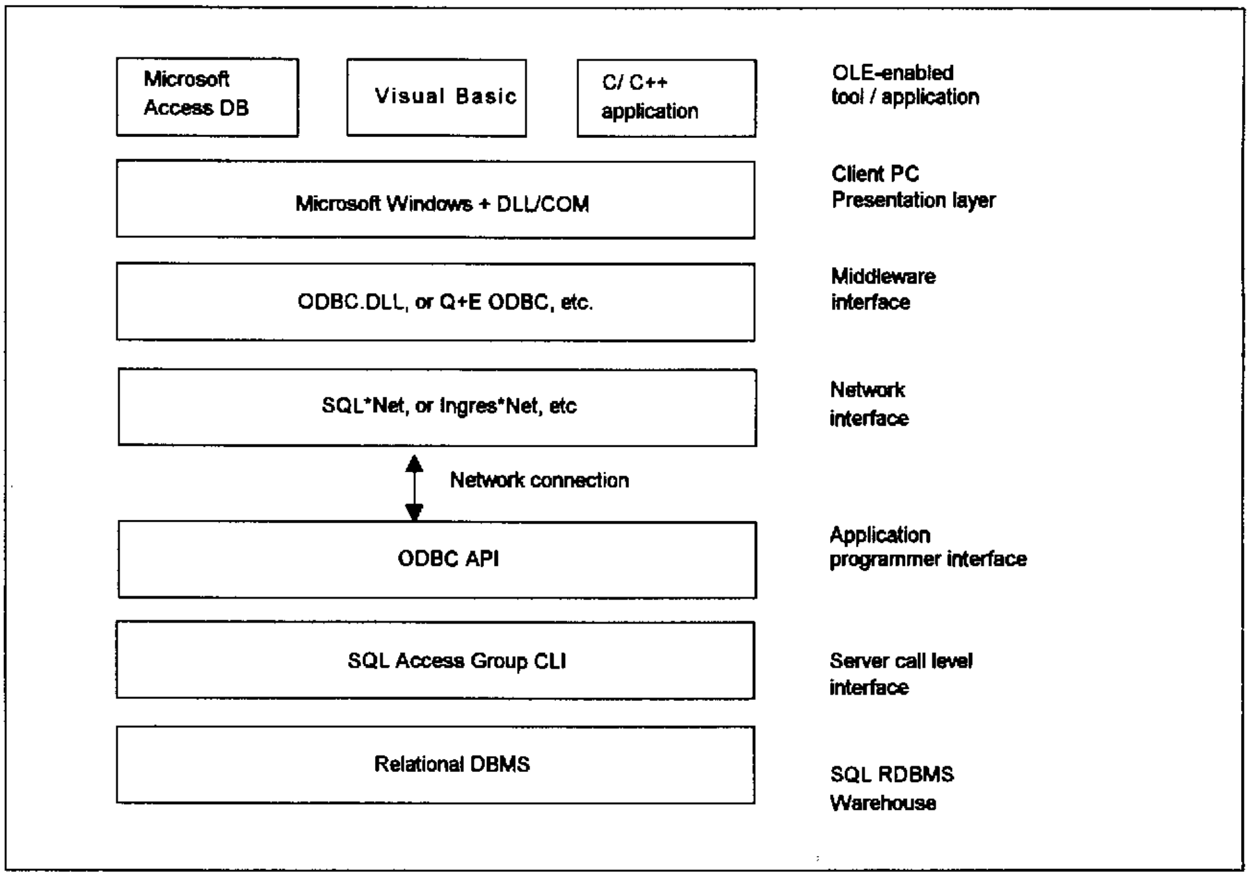

Microsoft Access DB
Visual Basic
C/ C++ application
OLE-enabled tool / application
Microsoft Windows + DLL/COM
Client PC Presentation layer
ODBC.DLL, or Q+E ODBC, etc.
Middleware interface
SQL*Net, or Ingres*Net, etc
Network interface
Network connection
ODBC API
Application programmer interface
SQL Access Group CLI
Server call level interface
Relational DBMS
SQL RDBMS Warehouse

**Figure 4.11** *Middleware layers in a database front-end architecture in a Windows environment.*

The most important middleware will be discussed in the section below.

## *Applications Programming Interfaces (APIs).*

Applications need standard ways of communicing. To accommodate interlinking of applications, software programs and user interfaces vendor, standards bodies and user groups have formulated service interfaces called applications programming interfaces.

The API on the called service utility or application receives standard requests, as requested by the requesting application for a specific requirement. The API then provides the standard response. APIs are available for user interfaces, databases, information representations such as graphics, network and security management services, data communication services and operating system interactions. The goal is a complete set of commonly accepted standard APIs. Forge (1995) identifies two types of APIs, namely high-level interfaces for application programming and low-level interfaces.

*Linking between PC packages*

Microsoft Corporation provides two packaged interfaces, namely DDE and OLE for linking packages in the same PC, relying on shared memory. This approach opens up the enormous and readily available power and range of PC users. DDE and OLE have been discussed in section 4.2.2.2 in the section on operating systems.

Other interfaces for package interfacing are OpenDoc (supported by Apple, Novell, IBM, Borland, Sun, Taligent, Claris and WordPerfect) and Lotus Embedding and Linking (LEL) from Lotus Development Corporation.

All of these allow packages to exchange data and to embed document objects in other documents.

*Interfacing to database management systems*

The client/server model is particularly useful for database access in a distributed environment. The client/server model is based on a standard access language, SQL, which has standardized to some degree the general way in which databases are accessed. The SQL queries send a request from the user, from the DBMS client-interface, to the server in the form of an SQL query. At the server the request is received by an application level API, translated by a low-level DBMS API (for example ODBC or IDAPI) and the data are retrieved from the relational database.

ODBC is a collection of drivers for standard database formats that provides a common programming interface to an array of data sources. This application allows the developer to work in one environment and write applications that access data in many different file formats. For instance, the application can be developed in a PC-based format and then the code may be used against server-based data.

All relational database vendors support SQL, but unfortunately there are many variations and extensions of SQL. So far the effort of the SQL Access Group (SAG) to impose an industry standard SQL on vendors for access to a particular database, has been met with mixed success.

Vendor approaches to interfacing with DBMS' include the following:

- versions of DBMS are offered that can run without the conventional operating system, but with a network-loadable module instead
- stored procedures are offered to allow some of the data preparation to be done on the DBMS server
- named pipes, a form of interprocess communication, can be used. The mechanism permits various applications performing librarian functions to implement a client/server configuration between SQL databases and applications across a NOS
- Microsoft's Open Database Connectivity API allows PC users to access a range of relational databases, including those from ORACLE, Informix, Ingres, Gupta, CCA, Microsoft, Prograss, Sybase, IBM, Hewlett Packard and Digital equipment. An alternative API is Integrated Database API (IDAPI) supported by Oracle, Borland, Informix, IBM, Sybase, WordPerfect and 50 other software suppliers.

***Transaction processing for client/server systems***

On-line transaction processing with a database is managed by an OLTP or TP monitor. These have traditionally been mainframe utilities, for example IBM's CICS (Customer Information Control System). Other types include USL's Tuxedo with the X/Open XA API for interfacing, NCR's Top End, Encina, CICS/6000 for IBMs AIX UNIX.

#### 4.4.2.3 Other cross-platform software

Complex LANs and WANs require staff to manage and monitor them, using specially developed tools. IBM's NetView or HP's OpenView runs on UNIX workstations and offers visual data displays ranging from network maps to data traffic. These types of software also include a host of utilities for managing everything from router configuration and network addresses to basic network security.

***Distributed management environment***

Open Software Foundation is currently focusing on developing distributed computing and management software and developing a new role as manager of middleware technologies. OSF's Distributed Management Environment (DME) provides a framework for a vendor-neutral, object-oriented environment that can be used by hardware and software vendors to develop products. DME provides:

- an object-oriented application interface for the administration and management of multivendor objects such as network devices, systems, applications, files and databases
- a standard API to manage a wide range of networked systems via either OSI's Common Management Information Protocol (CMIP) or Simple Network Management Protocol (SNMP)
- a common GUI across network management applications.

***Network computing environment***

Network computing environments allow the distribution of applications over heterogeneous platforms and software. OSF's standard to support this architecture is DCE or the Distributed Computing Environment. DCE provides a framework of services for distributing applications in heterogeneous hardware and software environments. DCE supports OSI standards and protocols and Internet standards, such as TCP/IP protocols. The NCE model is illustrated in figure 4.12.

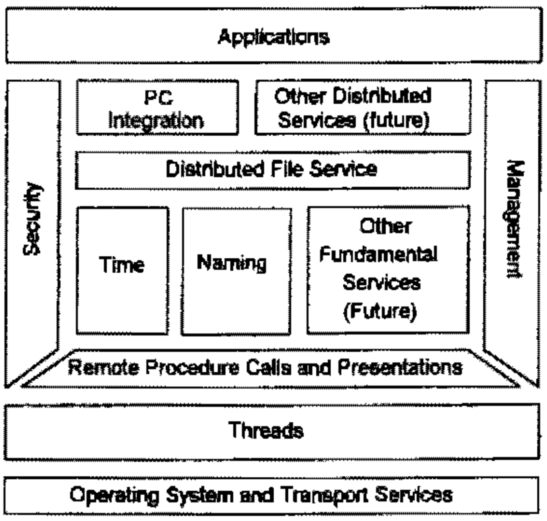

Applications
PC Integration
Other Distributed Services (future)
Distributed File Service
Security
Time
Naming
Other Fundamental Services (Future)
Management
Remote Procedure Calls and Presentations
Threads
Operating System and Transport Services

**Figure 4.12** *Distributed Computing Environment Model. (Open Software Foundation.)*

## *CORBA (Common Object Request Broker Architecture)*

CORBA is another important standard which was developed for distributed environments by OMG. CORBA overlaps with DCE with respect to some of the components, as well as with Microsoft (OLE) , IBM (SOM/DSOM). CORBA provides a means of abstractly describing applications and their relationships, as well as services for locating and activating those objects in a multivendor, networked environment. CORBA defines object request brokers (ORB) that communicate with other vendors' ORBs, using RPCs. CORBA ensures that distributed objects can intercommunicate, thus acting as middleware to objects. The OMG defined and supports CORBA as an open standard for application interoperability.

## *OLE*

Microsoft's OLE 2 (Object Linking and Embedding) is another set of standard specifications and specification implementation commonly used in the industry. The basic architectural model and its implementation differs from OpenDoc's (the standard of OMG), but in other respects the overall compound document and component software functionality is consistent and comparable with that of OpenDoc.

OLE supports object linking and embedding as discussed in section 4.2.2.2. Microsoft reportedly plans Macintosh and UNIX versions of OLE, and compatibility with CORBA standards.

The compound document technology of OLE is based on a component software architecture, based on a Component Object Model (COM) standard that ensures binary-level interoperabiltity across different applications. The central units of the COM model are sets of related functions, such as drag-and-drop, implemented as interfaces. All OLE objects implement the component object interface. The OLE compound document interfaces are organized in the following functional groups:

- compound object
- compound document
- linking
- data transfer/caching
- drag and drop
- persistent storage
- in-place activation and
- automation.

**Hierarchical storage model**

The hierarchical model uses storages to organize compound documents in a directory-like structure. Storage units contain streams, one for each object (the same as a file). Compound files are used for data access and manipulation.

**Data exchange model**

The OLE model, called uniform data exchange, allows data to be uniformly exchanged through drag and drop, copy and paste or API calls.

**Automation**

This facility permits applications to reveal their command and functional interfaces. It differs from that of OpenDoc as it uses OLE custom controls which replicate Visual Basic controls for OLE.

***Opendoc***

OpenDoc has an open non-proprietary architecture and supports Windows, Macintosh, OS/2 UNIX and Apple platforms. It is supported by Apple, IBM, Taligent, Novell/Wordperfect and Sunsoft.

OpenDoc consists of the following set of standards:

**Compound document services:** The parts of the document are organised by end users and may contain data of one or more types, including multimedia formats. The document contains a root or top-level part into which parts are placed which can be accessed and manipulated by end users. Editor part handlers and viewer part handlers provide interfaces for accessing and manipulation, while the parts will be displayed in frames.

**Control infrastructure** permits component handlers to share the compound document and interface facilities. A document shell creates and initializes document containers, assembles user interface events and sends them to the dispatcher. The dispatcher delivers events, such as mouse clicks, to the relevant application handlers. A window manager controls visible windows and frames.

**Open scripting architecture**

The automation technology is responsible for manipulating the document parts and programmatically coordinated these to work together. OSA is content-centred as each architecture has its own content model which specifies its data and operations. OSA defines a standard vocabulary of semantic events.

**Opendoc's component services** provide the underlying infrastructure for managing OpenDoc component documents, providing cross-platform portability and interoperability with non-OpenDoc applications. Key services include component registration, persistent storage (Bento), data exchange and resource negotiation.

The structured storage system is called a Bento container which represents a compound document as a collection of data streams. Bento maintains indexes that track complex relationships among document parts. The data exchange model is based on the Bento model. The same calls as those used to store documents are used to transport data within and across documents with drag-and-drop, copy-and-paste and linking.

**Object management services**

SOM (IBM's System Object Model) is a CORBA-compliant Object Request Broker that supports remote and local interoperability.

Another perspective, however, is multimedia requirements for client/server systems.

#### 4.4.2.4 Multimedia networking

The network layer is an important component as it controls the communication, accessing and delivery of the client requests, and vice versa. Another perspective, however, is multimedia requirements for client/server systems. Users at distant corporate sites have to communicate with each other. Communications occur daily in the form of phone calls, meetings, reports and text-based electronic mail documents. With desktop workstations as a gateway, networked multimedia give users access to video, text, application-based and audio information. Multimedia conferencing, e-mail and telecommuting are also made possible.

Significant technological advances that facilitates multimedia networking include asynchronous transfer mode (ATM) and synchronous optical network (SONET). Compression techniques that minimize data storage are also becoming increasingly effective.

Software manufacturers are introducing multimedia to the desktop, for instance Microsoft's Multimedia Extensions for Windows and Apple's Quicktime. These applications are quickly becoming embedded both in the interface itself and in the software used, such as presentation graphics and database management systems.

Multimedia can also be used for interactive training when applied to a WAN. Multimedia training modules allow users to access the information on demand, thus reducing classroom training time. At present, a considerable amount of multimedia training software is available on CD-ROM, which gives users access to large amounts of information at a single desktop workstation. The advantage of networking is that it allows corporations to have remote access to centralized multimedia databases containing training and corporate data.

***Technologies for multimedia networking***

Two network models offer methods for delivering multimedia information, namely the OSI-based layer model and the multimedia server model. The OSI-based layer model focuses on the organized transmission of data over networks and is illustrated in figure 4.5. The interoperability ensures that different systems platforms have equal access to multimedia information distributed over networks regardless of incompatibilities that may exist in their operation.

Multimedia has unique networking requirements. These include high bandwidth and isochronous services. The transmittance of multimedia information is sensitive to network delays and requires isochronous services. New technologies are becoming available to handle the flow of multimedia information. The foremost of these are fibre distributed data interface (FDDI), asynchronous data interface (ATM) and the synchronous optical network (SONET).

*Fibre distributed data interface (FDDI):* FDDI offers markedly higher data transfer rates than current networking technologies. FDDI offers a 100 b/s data transfer rate and FDDI LANs can be spread over larger distances than bus-type networks.

*Synchronous optical network (SONET):* will further empower the ATM by offering a high capacity on the physical level, taking multimedia beyond the LAN into the WAN. SONET is a packet technology that transfers data in packets or virtual containers of varying sizes, allowing for the efficient transfer of data. ATM and SONET use a four-layer model.

*Asynchronous data interface* (ATM) is a networking technology that offers both high-bandwidth and high-distance data transfer. In ATM, data is transferred in packets of fixed size, 53 bytes per cell. ATM can operate on a LAN or WAN or can be scaled down to the desktop.

## 4.5 The Application

The application is the program that must be developed to execute on the hardware components. An application uses the client's user interface for presentation to the client and the server for data services and processing. The application spans client and server using "application partitioning". Most of the logic of a large percentage of today's client/server systems reside on the client station and uses the server for data accessing and storing.

Typically, a client program running on a user workstation or PC will request a service, the network will transmit the message, using middleware; the server program will receive the message, execute the request and return the results. To summarize:

- user submits a request
- client application packages the request to the server
- request is transported through the network to the server
- server processes the request, returns the results
- results are transmitted along the network to the PC or workstation
- end user or client receives the results.

Gradations of client server computing developed by the Gartner Group is illustrated in figure 4.13.

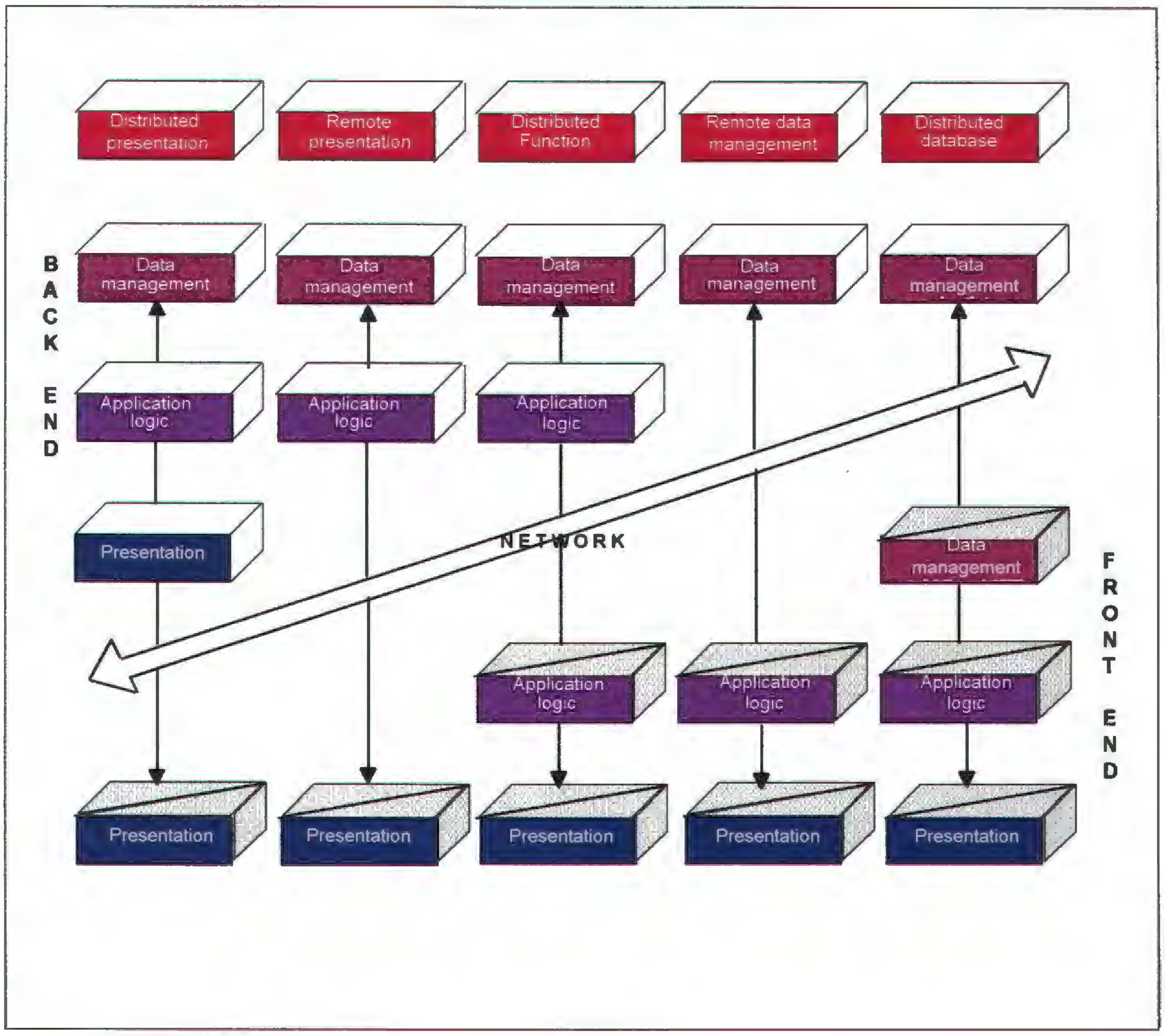


**Figure 4.13** *Gradations of client/server computing*

Five distinct portions of the application are identified, namely graphic user interface, SQL operations, business rules, connections between GUI and business rules and connections between SQL-operations and business rules. As is shown in table 4.11 the application uses six basic elements that interact with one another in a computing process (Marion, 1994).

Each of these elements is explained briefly.

| |
|---|
| Data storage |
| Database Management System |
| Application Software |
| Client and Network Operating System |
| User Interface |
| Display Device |

**Table 4.7** *Basic Application Elements*

***Data Storage:***

The requests of the client are often only a request for data. The function of the data storage unit is to provide a data store and to allow high-order processes to access the data, using data storage media, a storage control system and interfaces. Typical media include magnetic tapes, disks or optical disks. The storage control system comprises of the logic to access the data and controls the data flow to and from the storage unit.

***Database Management System***

The DBMS organizes the data for accessibility by application programs. The physical organization of data is managed by the data storage system, and the logical storage is managed by the DBMS. The DBMS organizes, stores, retrieves and relates data components.

***Application Software***

The application software can be written in a high-level language such as C++ or Visual Basic or it can take the form of more general types of application packages such as QuatrroPro. Application development environments, such as Powerbuilder, Forte, Visual Basic etc., can also be used to develop the application. A summary of relevant development environments are attached in Appendix B.

*Operating System*

The client operating system controls the resources of the computer system. The network operating system controls functions such as job scheduling, priorities, access to devices and security.

*User Interface*

The end user uses a user interface to communicate with the system. This interface could take the form of a character-based menu or a GUI. Windows and OS/2 environments support a more user-friendly environment consisting of a WIMP interface. The GUI uses an applications programming interface (API) to link a GUI with an existing application.

*Display Device*

The display device could take the form of a computer terminal, a PC or a workstation. In the earliest computer systems all the components, except the display device, were located on a central processor. Display units in the form of terminals were connected to the central processor. This is known as a time-sharing computing system. Another configuration, known as resource sharing, is the LAN system in which only the data storage element is stored on the server and the other components reside on a microcomputer.

The client/server system combines these approaches and exploits the advantages of each. The computing elements are divided among the platforms that are best suited to each element. In this configuration the display device, user interface, operating system and application software are all located on the client platform, while the database management system and the data storage are best located on the host platform as illustrated in table 4.12.

| **Processes** |
|---|
| Data Storage |
| Database Management System |

| Application Software |
|---|
| Operating System |
| User Interface |
| Display Device |

**Table 4.8** *Application Distribution*

Middleware is used in application development in the construction of client/server systems. In the process of developing an end user tool such as Microsoft Access, Visual Basic or a simple C++ application generator is employed at the desktop.

Secondly, reusable components from third-party parts vendors are added to the dynamic link library (DLL) to provide low-level client/server connectivity in addition to GUI and processing functionality, for example Microsoft's ODBC.DLL and SQL .

Off-the shelf components are assembled at the client and server ends of the system until everything works together and the application solves the business process problem.

## 4.6 Summary

In this chapter, the technology components of the client, the server, the network. and the application are reviewed. The preliminary model in figure 1.1 in chapter 1 can now be given in detail and a technology reference model is presented in figure 4.14 showing the essential components of the model.

In the next chapter, key aspects in developing client/server applications will be reviewed including management, business perspectives and critical success factors. The development process must be guided by a formal methodology, comprising a process model, techniques and tools. Personnel with skills are also identified as role players in the development process.

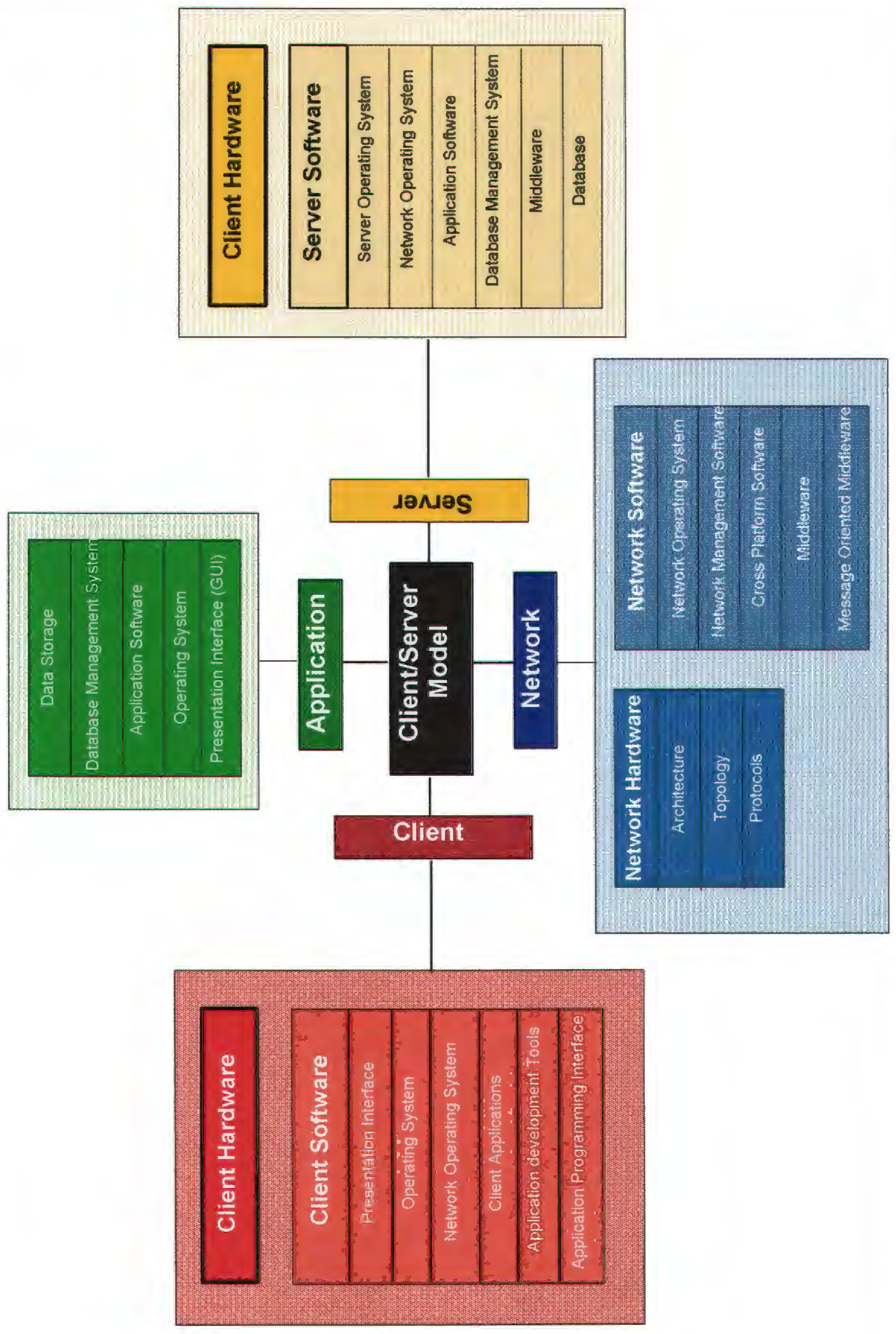


**Figure 4.14** *Client/server technology model*

# CHAPTER 5

# Key Aspects in Client/Server Application Development

## 5.1 Introduction

Client/server applications are complex to develop and design because of the scope of the application. To simplify complexity planning is essential. The application must be managed, monitored and controlled at all times. The basic design issues for information system design still hold, but additional design issues are applicable to client/server systems.

This chapter will define client/server-specific design issues. The development of a client/server system must primarily be driven by business needs must and be undertaken only for the sake of technology. Whitten et al. (1993) defines the traditional information system as a subsystem of the business:

"It is an arrangement of interdependent human and machine components that interact to support the operational, managerial and decision-making information needs of the business end-users."

The information system, and in this case the client/server system, must support the business perspective. This chapter reviews the importance of the business perspective and project management aspects in the development process, including planning, monitoring, control and critical success factors.

Analysis, design and implementation must be controlled by a prescribed methodology comprising a process model, techniques and tools. The development process is performed using a commercially available application development environment, e.g. Forte, DAIS, or Delphi and employing CASE tools where viable.

Another essential element in the development of a client/server system is the selection of the appropriate role players with the necessary skills. If the necessary skills are not available in the organization, outsourcing may be considered. A team will have to be selected comprising a networking and data communications specialist, an analyst, a database specialist and client/server specialist.

Client/server applications are by nature an open computing platform comprising a number of diverse components across a diverse, distributed architecture. A variety of elements must be considered together when creating a client/server solution, for instance the basic hardware, network, application software, database management system, support and implementation services.

This chapter reviews the aspects of importance in creating a client/server application, namely a generic strategy, business perspectives, a methodology comprising a process model, techniques and tools.

## 5.2 Business Perspectives and Strategic Planning

Information technology can be regarded as a tool in the tool kit used by an organization to keep itself profitable. The business perspective is of the utmost importance in the design of the enterprise-wide information system. The strategic planning process is a mechanism that can identify the areas of investigation required to support the process of client/server integration.

The information of an organization should be seen and used as a strategic resource (Nienaber and Redelinghuys,1995). According to this perspective, the information system will relate to corporate goals and strategies and add value to the organisation's products and services.

Strategic IT planning in the contemporary context includes various processes which determine how information technology can contribute to the implementation of the corporate strategy, so that the organization may gain a competitive advantage.

Smit (1994) identifies the following major steps in creating a strategic plan:

- assessment of corporate strategy and policies
- analysis of strategic information requirements
- creating an application plan
- creating a technology plan
- reviewing the mission, critical success factors, objectives and structure of the IT department.

**Assessment of corporate strategy:** Effective IT planning requires a clear understanding of corporate vision, mission and the driving force behind these. A well-defined set of objectives must be identified by the organization. The core strengths and the way in which the IT can be used in exploiting them to a competitive advantage must be identified. An assessment must be made of whether the development of the client/server system will have an impact on all aspects of the enterprise.

**Analysis of strategic information requirements:** Strategic information is information that is used to help the organization achieve and manage its objectives. The organization first has to identify the strategic information to be able to integrate the information by means of a particular IT configuration. The client/server system may be used as replacement for the mainframe, but client/server applications can often be more profitably deployed to extend the functionality of existing systems. Existing hardware may be downsized or upsized. The nature of the problem will serve to shape the type of client/server solution, the tools selected, and the approach used by the project. The primary purpose of such a system may be any one of the following:

- *Decision Support System.* In such a system the primary purpose is to provide the organization's decision makers with easy access to information concerning the business in order to support analysis and determine what happened in the past.
- *Departmental Support System.* The primary purpose of such a system is to automate one or more of the operational processes of a group of users. This category of application typically incorporates some form of transaction processing and operational reporting.
- *Transaction Processing (OLTP).* This is the typical 'core' application that serves to support the day-by-day operations of the business.
- *Electronic Performance Support Systems (EPSS).* These applications combine elements of DSS, OLTP, computer-based training (CBT) and often real-time systems.

**Creating an application plan:** The plan should include the mission, objectives and critical success factors, as well as high level business processes for each system or potential system. It should also contain a responsibility or involvement matrix. The scope of the impact must be assessed.

**Creating a technology plan:** The technology plan incorporates the above aspects into a technological realization. This deals with hardware, software and the infrastructure. The common factor here should be that of standardization and openness to ensure interoperability, flexibility and functionality. However, the current technological foundation must also take into account the integration of the new client/server system with the current systems. The sources of information residing in legacy systems are of the utmost importance. A strategy for converting this information to the new client/server system will be essential.

**Reviewing the mission, critical success factors, objectives and structure** of the IT department: Throughout the planning process care must be taken that the initial goals are achieved. The mission and objectives must be reviewed and critical success factors tested. Critical success factors have been identified and generally include the following:

- effective project management
- business process redesign
- managing the change process
- managing the necessary changes
- training and re-training.

The organization is the basis for structuring a client/server system. Information technology must be aligned with the business strategy of the organization. De Kok (1994) identifies a few guidelines and emphasizes the importance of a future vision:

- The guideline is to clearly understand the business processes and the integration of business processes. The potential levers of IT should also be understood as early as possible.
- A clear understanding of how systems support the current business must be attained.

- Opportunities, gaps and shortcomings must be clearly defined.
- Visions of the future process must be developed which focus on the future and do not only rectify current weaknesses.

Insight gained from strategic planning is vital to further implementation planning.Figure 5.1 illustrates the steps in defining the strategy.

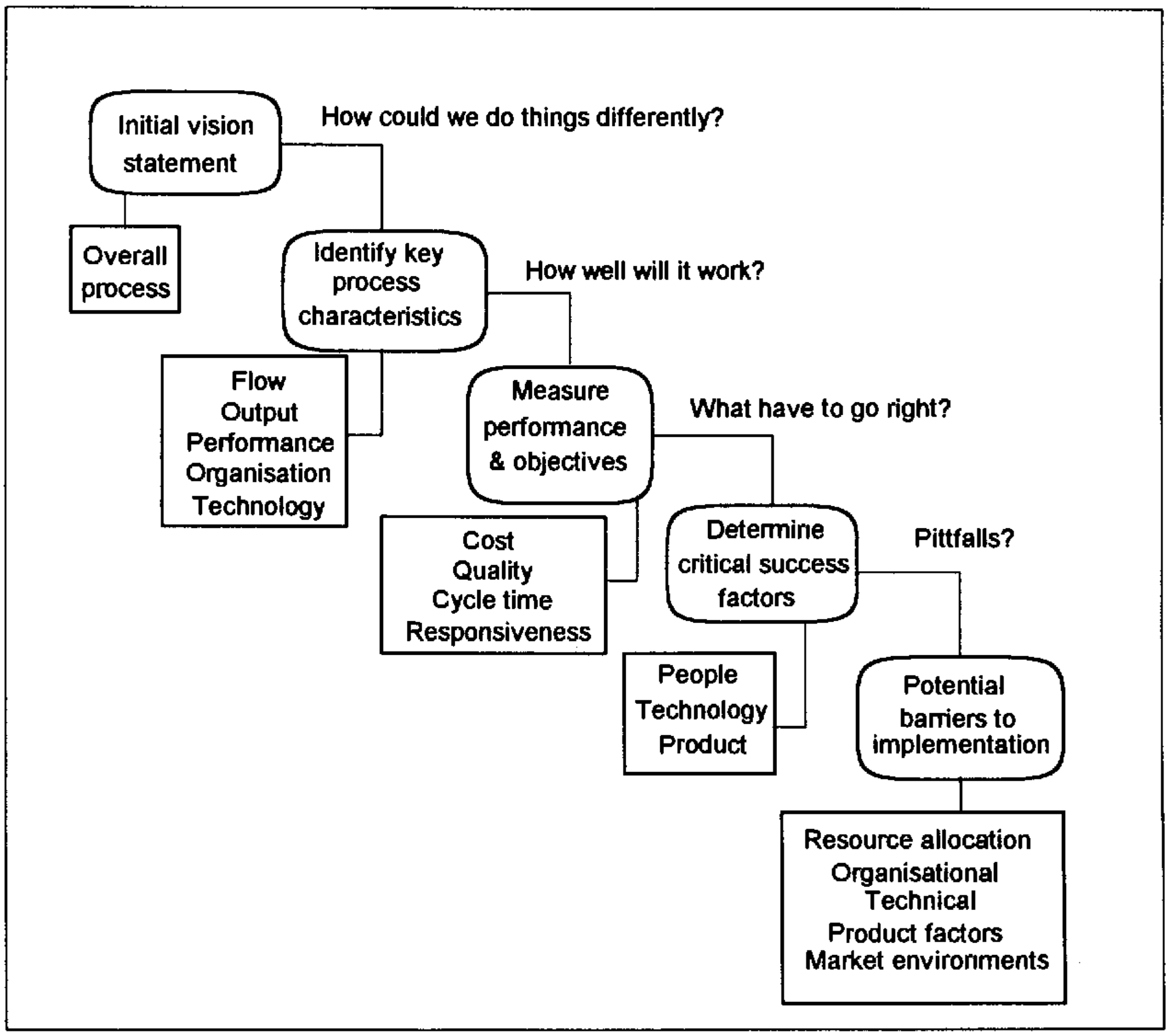


**Figure 5.1** *Steps towards an IT business strategy.*

The steps are:

*Initial vision statement:* Defining the overall process and assessing how it can be changed for the better.

*Identify key process characteristics*: The key characteristics are identified. Aspects include flow, output, performance, organization and technology.

*Measure performance and objectives*: The cost, quality, cycle time and responsiveness are identified.

*Determine critical success factors:* Quality, flexibility, and cost control are of importance.

*Potential barriers to implementation*: All negative aspects must be identified.

Vaughn LT (1994) identifies a three-level planning framework for client/server application development to sustain business perspectives, which includes strategic, tactical and operational planning:

**Strategic planning** includes the identification of goals and objectives. The strategic planning of an organization seeks to determine the objectives of the organization with regard to products and services, operating policies, growth targets, market definition and organizational reorganization. It is essential for the developer to understand the objectives the organization hopes to achieve with the deployment of client/server applications. If the objective is to migrate to more open, flexible and less expensive client/server environments, then an extensive enterprise-wide planning effort should be initiated to define the long-term objectives and technological directions of the organization. On the other hand, if it is simply to establish a cost-effective foundation for providing departmental system solutions of limited scope, extensive enterprise-wide planning will not be needed. The type of solution, for example DSS, OLTP or departmental support systems must also be determined.

**Tactical planning** determines how these objectives will be achieved. Tactical planning will include budget and time limits. On the organization's side, it will deal with the definition of requirements, financial forecasting, new product introduction, project prioritizing and budget formulation. On the systems' side it will deal with the evaluation and selection of technologies, the implementation of technologies, identification of new projects and development and training of personnel resources. The current technological foundation of the organization must be assessed, including aspects such as openness of technology and standards. The degree of integration that will be needed between current systems and the client/server systems being contemplated must be determined.

**Operational planning** will implement tactical plans while maintaining current activities. This planning phase includes the allocation of resources, support of daily activities, the production of products and the rendering of services. On the system's side, it deals with maintaining current services at acceptable performance levels, the allocation of resources to projects in progress, the management of projects in process, the maintenance of current, installed technologies, the installation and integration of new technologies and the resolution and management of problems. The skills and expertise of the organization must also be assessed to identify necessary training or outsourcing if the required skills are not available in the organization.

It is extremely important that the information technology of the organization forms part of the enterprise and is not merely an add-on. The information systems of an organization should not be seen as a separate department of computerized applications. It must be linked with the business needs, or better even, it should develop as a key technology of the organization and support the organization in gaining a competitive advantage over its competitors. The effects of the implementation of the client/server systems cannot be overestimated and will furthermore have an impact on the organization as a whole. Recognizing the importance of this factor and planning for change must be a critically important component of all strategic, tactical and operational plans.

Forge (1995) suggests using eight principles when designing client/server applications. These principles will be discussed in detail in section 5.3.3. However, the second principle of his methodology states that applications must be designed with the enterprise architecture based on the core business processes and standards. He emphasizes that a holistic view of the applications and their infrastructure is required, rather than individual developments by platform. An architectural approach is necessary where the design is led by core business processes and business problems.

*Critical success factors for client/server applications*

The introduction of a client/server system in the organization should be undertaken as a key technology and in compliance with the business perspectives mentioned. Forge (1995) defines five critical success factors, specifically for building client/server systems, in the following order of priority:

- business sponsorship
- people - skills, cultures and attitudes
- development methodology
- the enterprise architecture
- effective use of tools for rapid application prototyping.

Each of these factors are discussed briefly.

**Business sponsorship**

As emphasized in section 5.2, the move to client/server systems must be driven by a business demand. It must give the firm a competitive approach or sustain core business processes, or both. To gain this, business sponsors will be prepared to allow major changes to be introduced and high budgets which may be necessary for successful implementation.

**People - skills, cultures and attitudes**

Another essential element is the availability of the right mix of skills, either within the information systems department, or recruited from outside the organisation. The development of client/server systems requires a wide variety of skills which may not always be available. Critical areas, such as Network Operating Systems, PC LANs, and interapplication communications, may need reinforcement of skills. Outsourcing may also be a good option to consider. Figure 5.2 illustrates skills to be recruited, trained or outsourced. Skills and role players will be discussed in section 5.6.

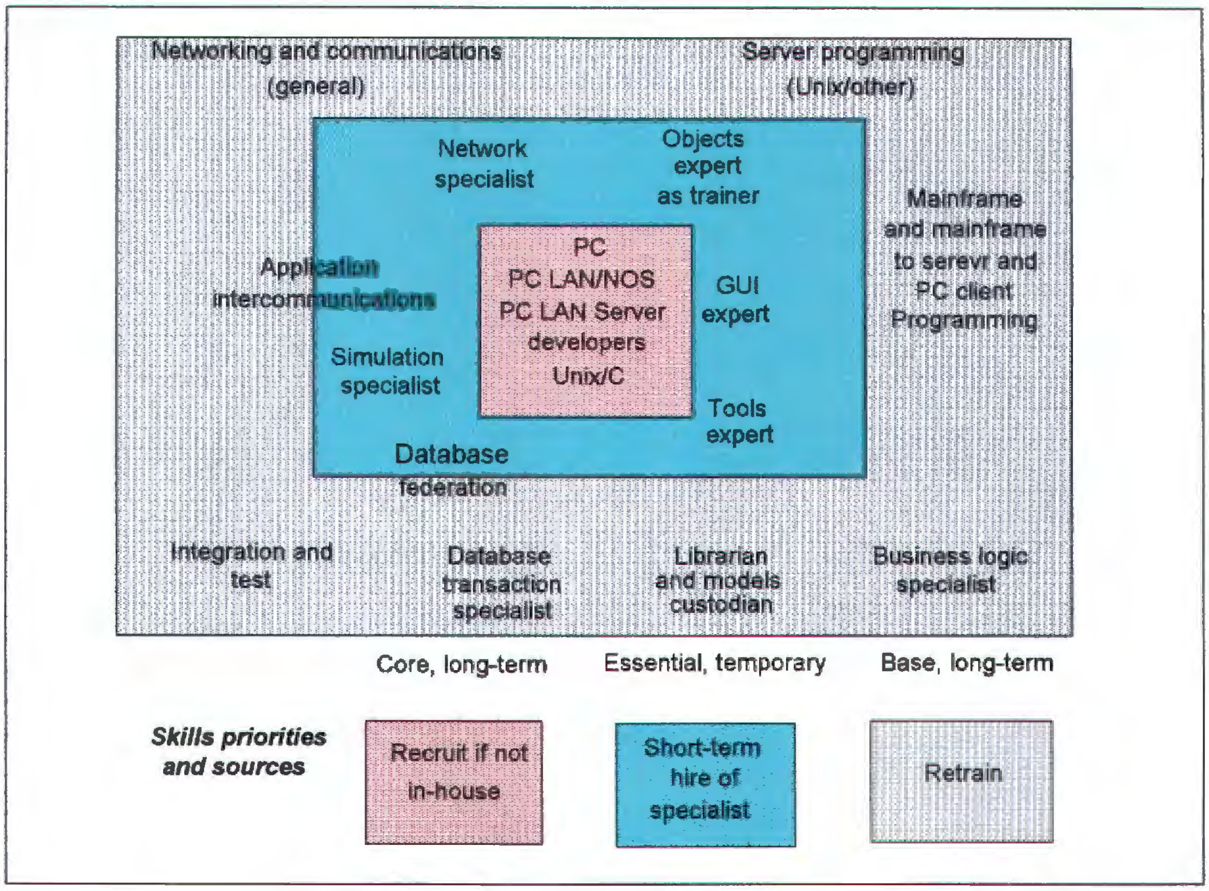

Networking and communications (general)
Server programming (Unix/other)
Network specialist
Objects expert as trainer
Mainframe and mainframe to serevr and PC client Programming
Application intercommunications
PC
PC LAN/NOS
PC LAN Server developers
Unix/C
GUI expert
Simulation specialist
Tools expert
Database federation
Integration and test
Database transaction specialist
Librarian and models custodian
Business logic specialist
Core, long-term
Essential, temporary
Base, long-term
Skills priorities and sources
Recruit if not in-house
Short-term hire of specialist
Retrain

**Figure 5.2** *Skills needed for client/server development.*

## Development methodologies

A strong vision of the business direction must be used to identify business processes. From these business processes, information flows, activities, tasks and staff involvement can be identified at a business level. Various methodologies comprising a variety of methods already exist, and more are being developed to sustain the development process. Traditional methods are generally not suitable for the development of client/server applications and more recent methods, such as object-oriented methods, rapid application prototyping and rapid application development methods must be investigated. Platforms can be developed in parallel using simulators to compensate for development not being synchronised with various platforms. Development methodologies will be discussed in detail in section 5.3.2.

## The enterprise architecture

In the development of client/server applications, major technological issues must be reviewed and, to simplify the development process, a general technical infrastructure needs to be in place. The infrastructure can be based on reusable application components, such as communications libraries and applications services. In addition, general technology decisions regarding platform types, networking and software packages as shown in Figure 5.3 should be incorporated. The enterprise-wide architecture is also be addressed in section 5.3.3 in the discussion on rapid application development frameworks and Forge's methodology.

The enterprise architecture will determine the common technical standards of the enterprise, their interactions and the range of vendors.

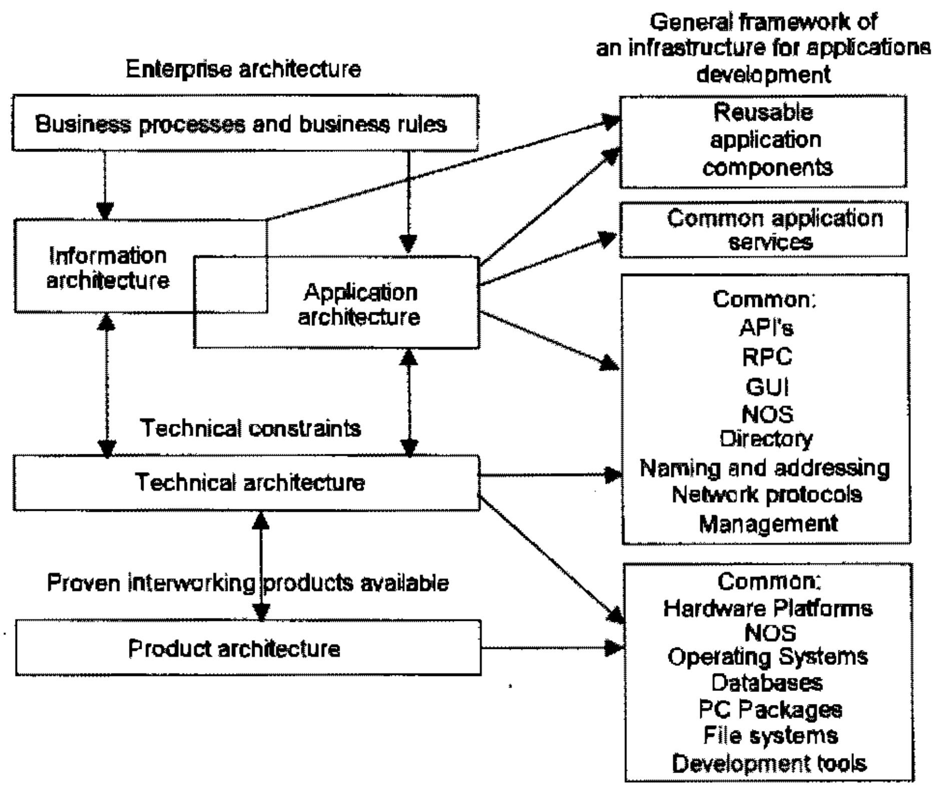


**Figure 5.3** *Application-wide Architecture*

**Effective use of tools for rapid application prototyping**

The effective use of available tools and techniques that work together will support the developers in their task. The DBMS and OLTP utilities can provide ways of federating different databases and new applications interfaces. Tools which have been found essential for success are those that support rapid application development and prototyping. Tools and techniques will also be discussed in section 5.4 and 5.5. A selected number of client/server application development tools are listed in Appendix B.

However, planning and identifying critical success factors alone will not suffice. The execution of the development process must be monitored and managed. The management of client/server systems is reviewed in the following section.

***Management of Client/Server applications***

Client/server systems are still in their infancy and a limited number of tools is available to support project management of client/server applications from an enterprise perspective. This dearth of appropriate tools have contributed to a lack of progress in client/server implementations. Project management is the process of directing the development of an acceptable system at a minimum cost within a specified time frame (Schach, 1994). The management of software development projects primarily involves:

- planning of project tasks and staffing the team
- organizing and scheduling the project effort
- directing and controlling the project.

All of these tasks will benefit by having an appropriate model of the software development process or software process model, which will give it a standard structure, development discipline and measurable and controlled units.

According to Cashin (1993), JP Morgan & Co of New York has produced a client/server management agenda in the form of a set of management-related issues to use as evaluation issues. These issues are categorized into four areas: architecture, methodology, organization and support. A diagram of the issues related to each area is shown in figure 5.4.

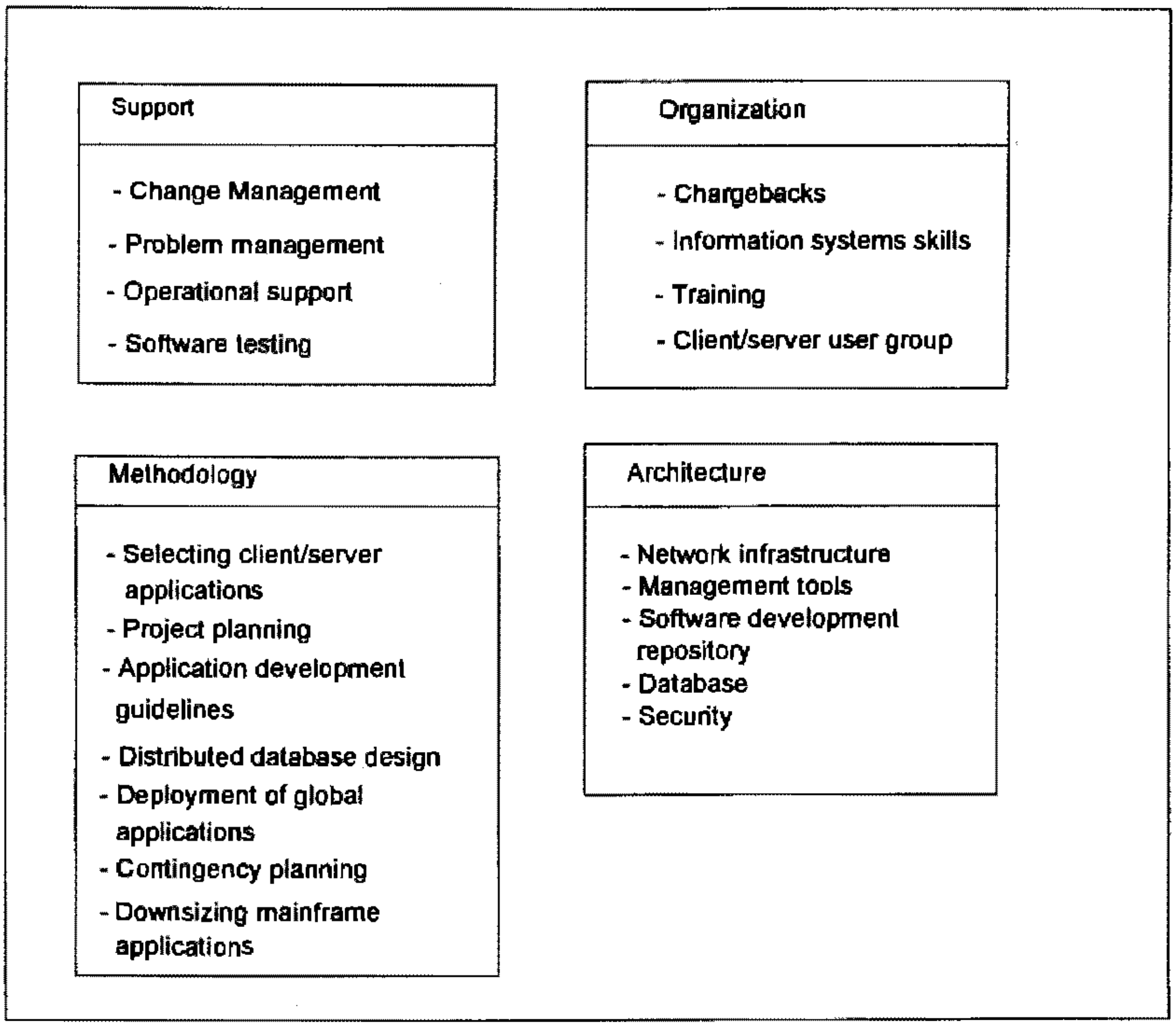

Support
- Change Management
- Problem management
- Operational support
- Software testing
Organization
- Chargebacks
- Information systems skills
- Training
- Client/server user group
Methodology
- Selecting client/server applications
- Project planning
- Application development guidelines
- Distributed database design
- Deployment of global applications
- Contingency planning
- Downsizing mainframe applications
Architecture
- Network infrastructure
- Management tools
- Software development repository
- Database
- Security

**Figure 5.4** *Client/Server Management Agenda*

Today's computer software, such as Harvard's Project Manager, ABT's Project Management Workbench and Microsoft's Project Manager, is being used to support project managers. CASE tools also provide useful project management capabilities.

## 5.3 Development Process

The series of steps required in a development process to yield a product may be modelled according to a process model. For software, such a process model is the software development life-cycle model, consisting of a number of development phases or cycles. Various methods and techniques have been proposed to support the tasks of the development phases (Falkenberg, et al., 1992). Life-cycle methodologies comprising methods for the complete life cycle have also been proposed (Steinholt, 1993).

Traditional design methods may be categorized into process-oriented and data-oriented methods (Pressman, 1992). Process-oriented methods concentrate on analysing the processes of the project, using i.e. data flow analysis, workflow analysis, and the structure of the problem, whereas data-oriented systems concentrate on analysing the data.

Traditional methods do not satisfy all design needs and newer developments such as object-oriented analysis methods and rapid application development have been added to the list (Yourdon, 1994). However, client/server systems have their own characteristics and therefore need specific methods for analysis and design. In addition to the standard data and process analysis techiques, techniques are needed to model concurrency, partitioning, serialization, and the multiple components of a client/server environment. Various tools are available to support these techniques. In addition to the existing general software development tools and add-on products, vendors of client/server products are offering numerous development tools. In this section, the traditional process models will be discussed in short. Various methods comprising techniques and tools for the development of client/server systems will also be reviewed. Additional principles and techniques that have proven useful in the analysis of the processes, determination of problems, identification of solutions and modelling of multicomponent environments will also be reviewed, although some are considered traditional in systems development methodologies.

## 5.3.1 Process Models

Different software development models and life-cycle models are based on different views on software development (Steinholt, 1993). This section describes several software development models, such as the waterfall model, the rapid prototype model, the incremental model and the spiral model.

### *The Waterfall Model*

An advantage of this traditional model is the integration of testing in each phase. Documentation is not a separate phase but should be performed in every phase.

A disadvantage of using the waterfall model for client/server development is that the model is documentation-driven (Schach,1994). If the specifications are faulty, incomplete, inconsistent or ambiguous, the whole system will be constructed incorrectly. The model also does not support iteration which is an integral part of the process. The model has also been criticized for its lack of descriptive power, and its failure to integrate activities such as resource management, quality assurance, configuration management and verification and validation (Charette, 1987).

Attempts have been made to enhance the Waterfall model. Resulting models are the rapid prototype model, the incremental model, and Boehms' (1988) spiral model. These will be discussed in short.

***The Rapid Prototype Model***

The rapid prototype model rapidly builds a working model of only a subset of the final product. An advantage of this model is the early feedback to the client, developer and management. It also includes aspects such as resource management, configuration management and verification and validation.

This model on its own is not very appropriate for client/server systems as it is built on an incremental development process and does not support an integrated environment (Schach,1994). It may, however, be used in combination with other models, such as the spiral model to achieve effective results.

***The Incremental Model***

The realization that software is developed incrementally has led to a software process model that exploits this aspect of software development.

An advantage of this method is the rapid development of the parts. The client will have a subset of the system to experiment with at an early stage of the development. However, for large systems such as client/server systems, the incorporation of the project as a whole may become very difficult (Charette, 1986).

***Spiral Model***

The spiral model improves on the basic waterfall model by adding risk analysis, prototyping, iteration, and project management aspects to the development cycles.

Linear models are not really suited to the development of complex multicomponent client/server systems. The client/server model consists of separate models that must be intergrated but which also function correctly as separate elements. The spiral model builds a series of prototypes that are designed and tested before arriving at the final operational version.

A disadvantage of the spiral model is its lack of explicit process guidance in determining the prospective system's objectives, constraints, and alternatives. An extension of the spiral model is proposed by Boehm (1994), called the Next Generation Process Model (NGPM), which uses the Theory W approach (Boehm-Ross, 1989). This refined spiral model addresses the shortcomings of the basic model.

### 5.3.2 Methodologies, Methods and Techniques

Traditional design methodologies may be categorized into process-oriented and data-oriented methodologies (Somerville, 1994). A methodology comprises a number of methods which use appropriate techniques and tools to achieve a general goal. Process-oriented methodologies model processes, data flow between processes, data stores, and their interaction with external entities. Methods used are data flow analysis, workflow analysis, structure charts, and transaction analysis diagrams. These methods model the structure of the processes in a linear fashion, and do not include changes to the state of the system as a result of events or time-related events. Data-oriented methodologies design the product according to the structure of the data on which it is to operate. Methods used include Jackson System Development, Entity-Relationship diagrams and Warnier-Orr diagrams.

Traditional methodologies do not satisfy the design needs for client/server systems comprising a complex environment. These methods separate the process model from the data model allowing contradictions and incompatibilities in the design. As current systems differ radically from traditional ones, new methodologies are needed to support the larger integrated environments (Yourdon,1994). Developers need an approach that will support the complications of using a combination of mainframes, mini's, LANs and PCs.

***Object-oriented methodologies***

Object-oriented methododologies are proving to be quite useful in several areas which have not been served well by more conventional methodologies (Nelson, 1992). The main advantages of replacing conventional methodologies with OO methodologies are productivity, rapid systems development, increased quality and maintainability (Coad, 1990). Furthermore object-orientation supports reusability, inheritance, abstraction, data encapsulation and polymorphism (Rumbaugh, et al. 1991). The principles of object-orientation are explicitly suitable for client/server applications, for instance the encapsulation of complexity is achieved by dividing the entire system into smaller pieces.

In the next section four additional development methodologies for client/server systems will be discussed. Three of the four methods specifically uses object-orientation, while the fourth does not preclude its use.

### 5.3.3 Client/Server Development Methodologies.

Traditional types of development methodologies are not appropriate for the development of client/server systems (Forge,1995). At this time, there is no standard methodology for the development of client/server applications, however, guidelines have been identified and proposals published. In the next section alternative published methodologies, comprising methods and techniques proposed by different authors, are listed and discussed in short. Vaughn (1994) incorporates quality management and continuous improvement techniques into the development methodology, Powers, et al. (1990) includes rapid application development and other techniques into their methodology, Seybold (1993) supports a workflow methodology and Forge (1995) suggests using a methodology consisting of eight basic key design principles, specifically designed for client /server systems.

**Vaughn's methodology**

Vaughn (1994) suggests a methodology that is based on research by Rapaille, et al. of Archetype Studies International, of the University of Chicago. The methods incorporate proven quality management and continuous improvement techniques, as illustrated in figure 5.5.

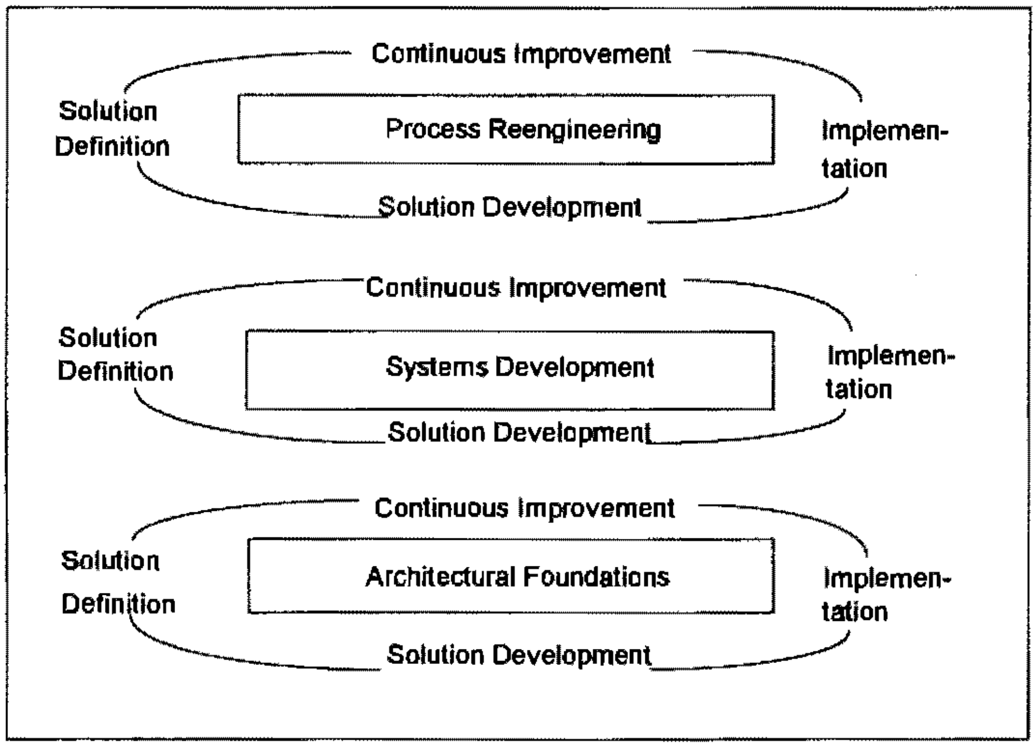


**Figure 5.5** *Vaughn's development methodology(1994)*

**The process reengineering phase** of the project consists of identifying and defining of problems to be addressed, gaining a broad understanding of current situations and identifying the changes to be made, and defining and prototyping different solutions until an optimal solution is found. It also comprises the development of the solution, the implementation and monitoring thereof and the continuous improvement of it.

**The systems development phase** consists of modelling the information and workflow necessary to implement the solutions, documenting current functions, technologies and information structures, determining user needs and functional requirements, prototyping various solutions until the optimal solution is found and the requirements of that solution have been specified, designed and implemented.

**The architectural phase** of the project is concerned with reviewing current technologies, selecting of tools to be used in building the system, selecting and prototyping the client, network and server technologies to be used in both prototyping and building the application. Defining the interaction of components, benchmarking and implementing data structures, determining data distribution across the network, and designing external interfaces also form part of this phase.

Each of the phases is cyclic, and consists of four steps, namely solution definition, solution development, implementation and continuous improvement. The methodology is flexible and can be seen as a guideline allowing the developer a choice of appropriate techniques. One of the techniques that may be used is the cause-and-effect diagram.

***Cause-and-effect diagrams***

Cause-and-effect diagrams are effective in aiding the identification and classification of problems and their causes. These diagrams assist in the efforts to systematically gather and organize people's thoughts on the potential cause of a problem and provide a framework within which these causes can be structured. These diagrams may be used as frameworks within which more information can be gathered. Figure 5.6 illustrates the notation.

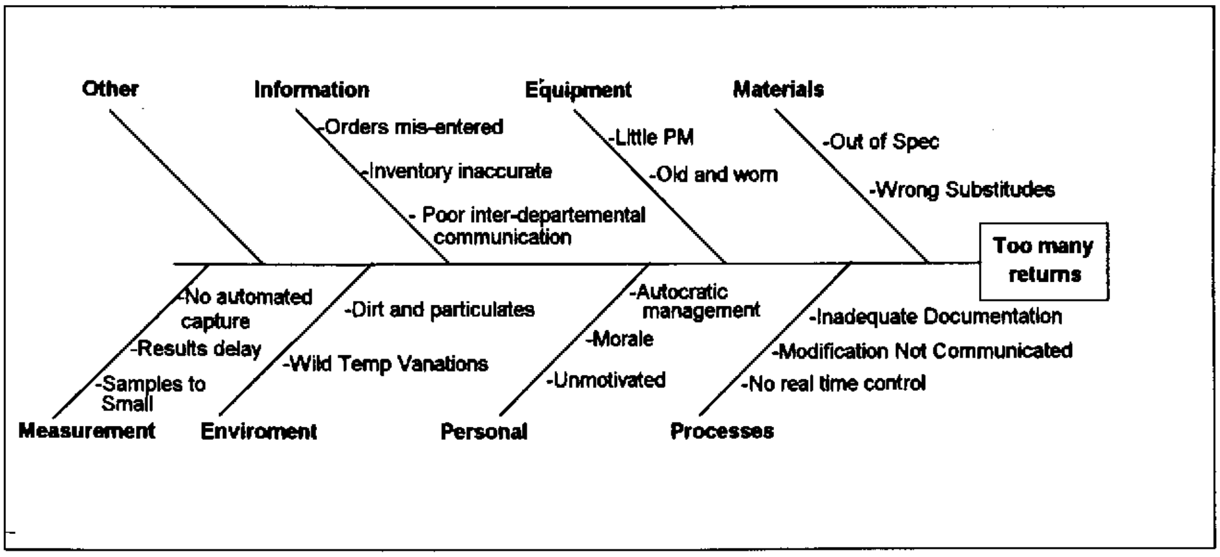


**Figure 5.6** *Cause/Effect diagram (Vaughn, 1994)*

**Framework for Rapid Application Development**

Powers, et al. (1990) proposes a development methodology comprising three steps. The first step is the construction on an IS framework, the second step the selection and use of a set of life-cycle techniques and the third the selection of various techniques outside the framework of the traditional life cycle. Each will be discussed briefly:

*Step 1: The IS framework for rapid development* constitutes the identification of the enterprise-wide architecture as well as the creation of an effective development infrastructure. The enterprise-wide IS architecture comprises the following:

- information architecture
- application architecture
- technical architecture
- management architecture.

The development infrastructure consists of the business objectives, methods, techniques, developers that possess certain skills, and automated tools. This infrastructure must be executed within a set of project management disciplines, as illustrated in figure 5.7 .

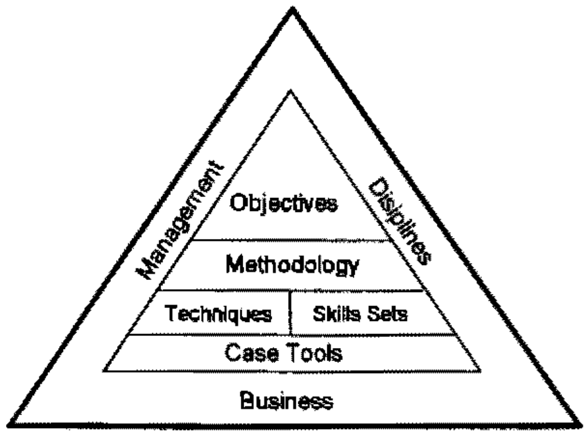


**Figure 5.7** *Components of a development infrastructure (Powers & Cheney, 1990)*

*The basic life-cycle techniques* applied are scope control, joint application design (JAD), prototyping, version development, application software packages, application generation tools, I-CASE, life-cycle tailoring, as illustrated in figure 5.8.

Figure 5.8 illustrates these components.

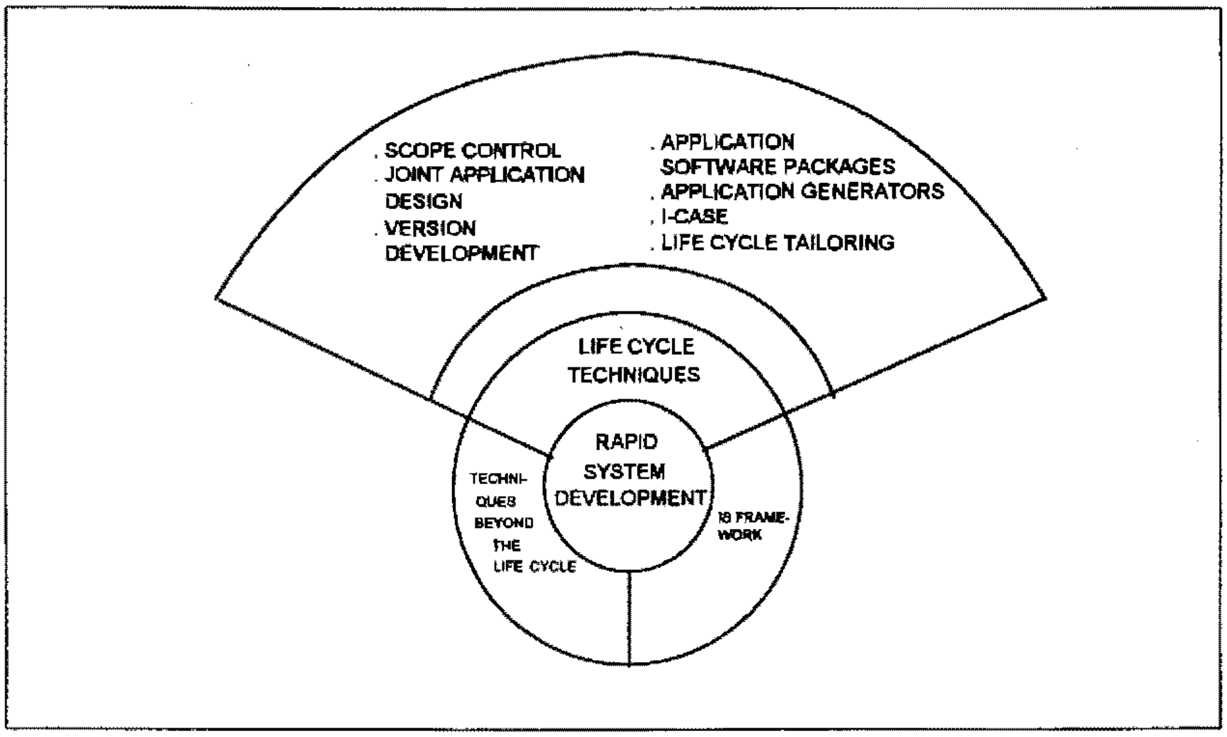


**Figure 5.8** *Life-cycle techniques for rapid development (Powers & Cheney, 1990)*

*The techniques used for rapid application development are discussed below:*

*Scope control* provides a base system of absolutely critical functionality to be installed first. Then, as experience is gained, additional functionality can be judged on its own merit - benefit versus cost. In order to identify essential elements, three steps are suggested, namely eliminate everything possible, simplify the remainder and automate simplified functions where possible.

*Joint application design (JAD)* is a technique where intense working sessions lasting about four days are used for application design. A leader will head the sessions, attended by 6 to 10 key persons who have the necessary knowledge about the field of study and the necessary authority to make decisions in that field.

*Prototyping*, already discussed in this chapter.

*Version development* follows a very simple principle, namely to develop a large complex system by breaking it into a series of parts, versions, that can be separately designed and implemented.

*Application generation tools* can be used for saving time. The following facilities are available: relational database systems, screen painters, report automators, conversation generators, fourth generation languages, code generators, automated testing tools.

*I-CASE* refers to integrated CASE tools to assist in the process.

*Life cycle tailoring* implies that life cycles can be used as a base and can then be tailored to the given situation.

*Additional techniques* for rapid development beyond the SDLC are rapid iterative prototyping, reengineering and object-oriented development.

Rapid application development supports the architecture of the client/server system, and also allows for techniques beyond the conventional techniques.

**Seybold Workflow Methodology**

A methodology which also incorporates rapid application development has been suggested by Seybold (1993). The methodology is executed in four steps:

*Step 1 Define business processes*

The procedure is as follows:

- identify the target application area and shared models
- identify process owners and their responsibilities
- develop a clear process philosophy
- define the role of IS in the reengineering process
- reengineer the IS to be able to operate iterative OO analysis and design.

*Step 2 Define business object and rules*

The procedure is as follows:

- define business processes, object definitions, rules and process definitions
- define business objects
- capture the rules of the business, policies and procedures
- store in electronic, reusable form.

*Step 3 Develop user interface prototypes*

The procedure is as follows:

- use rapid application development
- develop prototypes of the user interface
- supply client with a copy for experimentation
- redesign for additional client requirements as suggested.

*Step 4 Develop an application prototype*

The procedure is as follows:

- develop a prototype
- develop application model in a multi-user environment
- test and verify
- develop the delivery model.

Workflow analysis which can be used in analysing processes will be discussed briefly.

*Workflow analysis*

Workflow analysis specifically targets actual work being performed in terms of interactions between people within the problem space. This examination is done with the intent of identifying bottlenecks, disconnections between related activities, and unnecessary or redundant steps which may be deleted The basic element of this method is the workflow loop. The elements of the workflow diagram can be combined to model workflow processes of great complexity. It can also be used with success to analyse and optimize reliable processes within the problem space. Figure 5.9 illustrates workflow analysis.

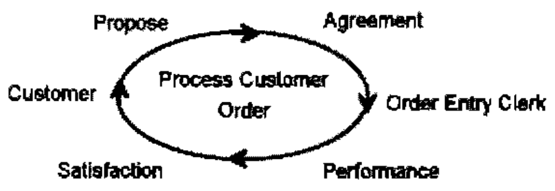


**Figure 5.9** *Workflow Analysis (Vaughn, 1994)*

Advantages of this methodology are the immediate furnishing of results, easy maintainability and the fact that information system processes are being initiated by business processes. This methodology incorporates essential elements for the development of client/server applications, such as the business perspective, object-oriented development, rapid application development and prototyping

**Forge's Methodology : Key Principles in Client/Server Application Development.**

From extensive practical experience, Forge (1995) proposes a flexible development methodology consisting of eight key principles:

- sample object-based methodologies and exploit the general principles of object-based design
- design applications with an enterprise architecture based on the core business processes and standards

- design at high-level using three basic concepts: shared application services; network design for LAN or WAN bottlenecks; client server pairs
- understand the business logic, plus technico-economics and politics
- size in two steps
- select server platforms according to their roles
- design for management
- roll-out with care.

Each of these design principles will be discussed in the subsequent section:

*Principle 1: Sample object-based methodologies and exploit the general principles of object-oriented design.*

Forge suggests using a spiral or incremental life-cycle model, combined with an object-oriented approach and rapid application development methodology. New methodologies based on an object-oriented approach, such as Wirfs-Brock, Rumbaugh et al, Grady Booch, Schlaer Mellor and Jacobson's objectory approach can be explored and the most appropriate or a combination of these, can be selected and used.

*Principle 2: Design applications with an enterprise architecture based on the core business processes and standards.*

A holistic view of the applications and their infrastructure is required, rather than individual developments by platform. An architectural approach is necessary where the design is led by core business processes and business problems. Figure 5.10 defines an enterprise architecture of how to design using standards, which allows the straightforward definition of the

the application set and the architecture best suited to the enterprise.

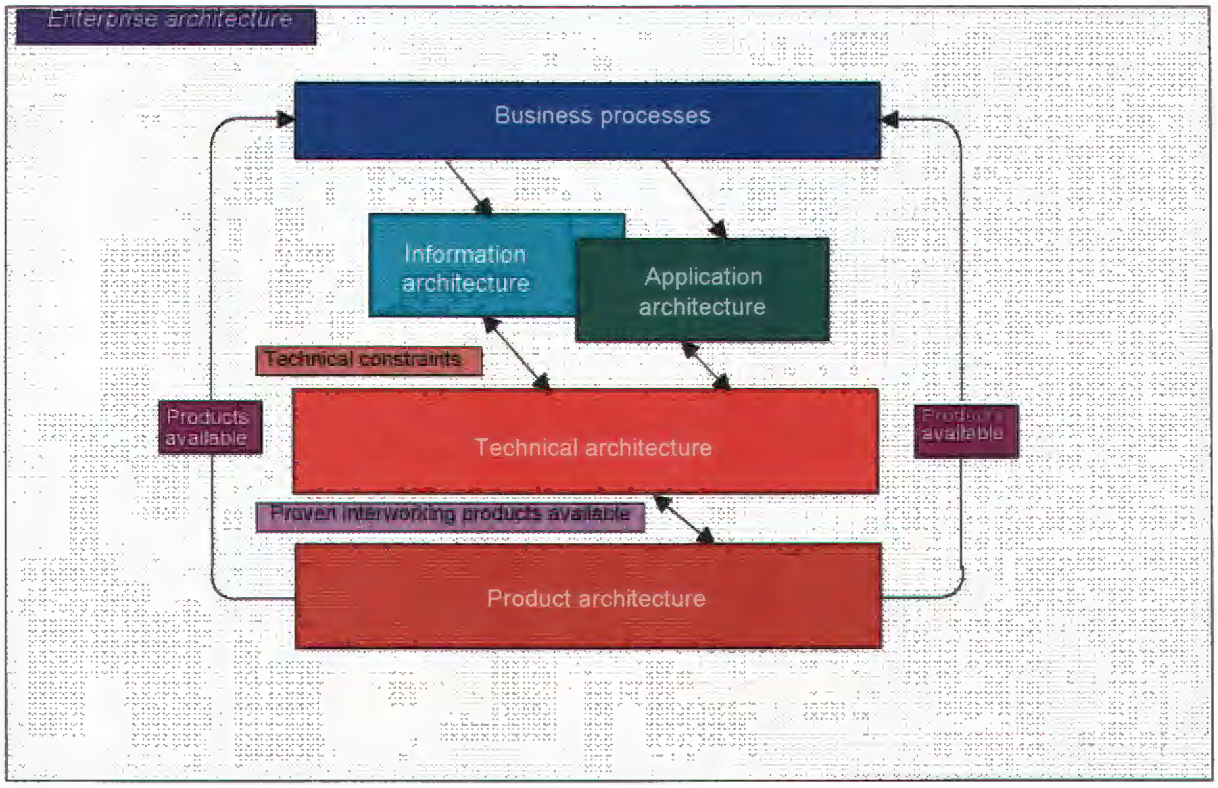


**Figure 5.10** *An enterprise architecture (Forge, 1995)*

*Defining the terms:*

Enterprise architecture - an IT architecture based on business principles.

Business processes - the core company activities for producing profit.

Information architecture - the organization of data, documents, images throughout the enterprise, according to business needs and end user needs.

Application architecture - the configuration and general types of applications and their interrelations, according to business needs and end user needs.

Technical architecture - the selection of standards for interfaces and configurations of all hardware and software and of the major types of each.

Product architecture - the pragmatic selection of software and hardware products with a view to achieving interconnectivity with each other and with the hardware platforms supplied.

The following decisions must be taken at a technical level:

- major processing platforms (e.g. mainframe at centre, servers at branch offices, 486 PC on desks)
- preferred systems environment by processing platform (e.g. Unix on servers, Windows on Pcs)
- networking and communication architecture and topology (e.g. linked LANs with high-speed routers connect branch offices via communication servers to national centre. LAN servers use Ethernet in branches)
- systems management responsibility at technical and operational levels (e.g. local responsibility and maintenance with central multilingual help-desks and management tools based on a product with multivendor support)
- a database and data administrator (e.g. new databases will be relational, from certain suppliers and coordinated by a central database administrator, with a local administrator).

A technical architecture can define a closed set of interworking standards, plus interworking systems components. A list of de facto standards are attached in Appendix A. In practice, vendor choice is often an important constraint today. Various market leaders are all expanding and are moving towards client/server support, for instance Microsoft, Lotus, relational database management systems, IBM, DEC, HP, NCR and Unisys.

*Principle 3: Design at high-level using three basic concepts.*

*A four-layer model with modular services*: Design the overall system with four layers, as shown in figure 5.11.

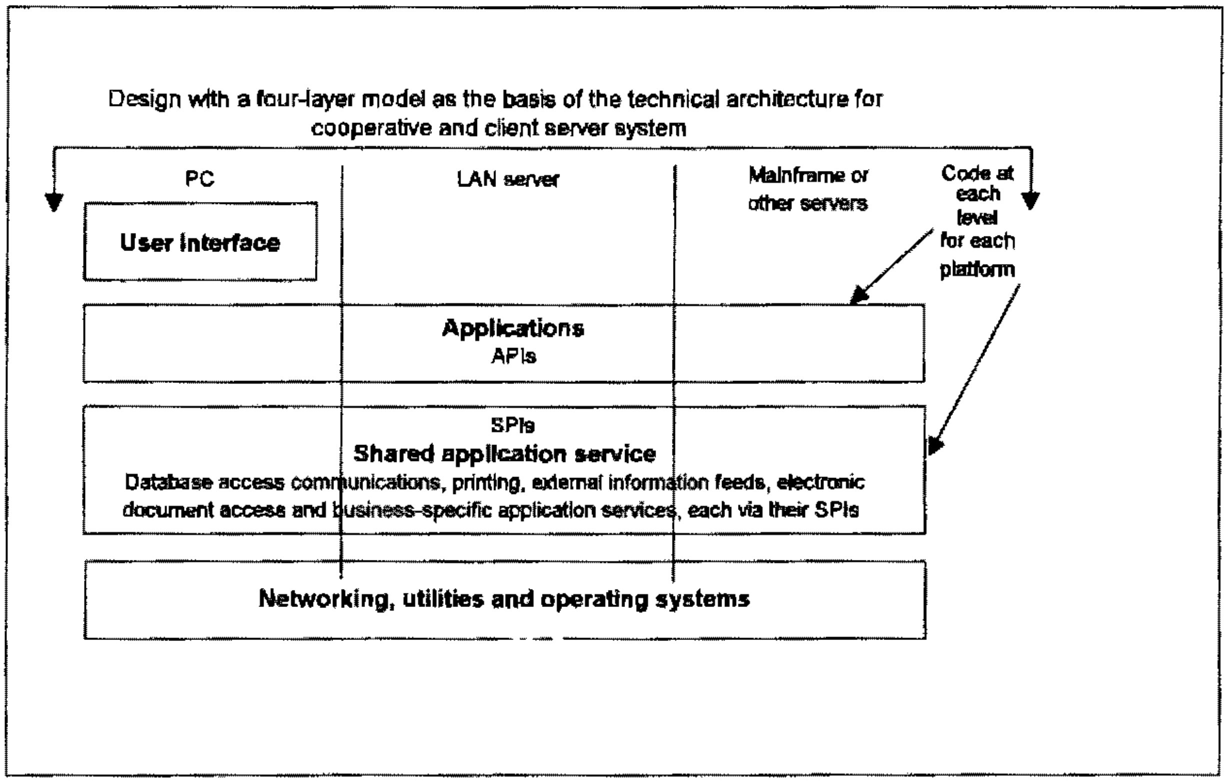

Design with a four-layer model as the basis of the technical architecture for cooperative and client server system
PC
LAN server
Mainframe or other servers
Code at each level for each platform
User Interface
Applications
APIs
SPIs
Shared application service
Database access communications, printing, external information feeds, electronic document access and business-specific application services, each via their SPIs
Networking, utilities and operating systems

**Figure 5.11** *Four-layer model for the technical architecture (Dewire, 1993)*

*Design from the point of view of a network of nodes*: Design of distributed processing requires the examination of the data flow traffic across the network, and the identification of the range of performance factors. Problems such as time-outs, routing and line quality, security and contingency plans must all be allowed for. Relevant design parameters include:

- desired throughput (transactions and data rates)
- effective delays, including overheads
- commit and rollback points and protocols for data flows
- security measures and reliability
- failure modes and their effects

- likely saturated data paths
- bottleneck identification.

*Use the client/server relation for any number of platforms:* The client/server relation is a powerful conceptual model, and can be used for each application-piece interaction which is spread over several machines to provide a simple way of assigning functions and management responsibilities.

*Principle 4: Understand the business logic, plus technco-economics.*

Splitting applications and data across many platforms is a difficult art. Business logic should determine splitting decisions. Forge identifies a few guidelines:

- discern clearly on what grounds the splitting decisions will be made (cost, management or political?)
- apply business logic to splitting
- organizational or political pressures may be the major splitting constraint
- techno-economic considerations relate to balancing or optimizing various performance and capability criteria.

When splitting applications, the split may be made the closest to either the data owner, the responsible person for entry and integrity, or the person who created it.

When splitting data, the most general rule is to examine sessions and traffic which form database queries, and then to place application logic and data to optimize cost/performance across the transaction chain for the highest number of users while minimizing network loads. Other criteria are:

- place data where it adds the most value
- place data according to the degree of sharing

- place data according to the update rate to minimize network traffic
- give local data copies to users if there are response time problems.

The alternatives to data distribution are:

*Data separation:* Different databases are placed at different places where they are needed and used most regularly.

*Data partitioning:* The data are split vertically (fields) or horizontally (records) and placed at different points, where they are used and needed most, but the whole is shared by a distributed community of users.

*Data replication:* The same data are placed at several sites, and problems with updates and network traffic must be considered.

Distribution of applications and data is a whole study on its own, and falls beyond the scope of this study. It is of importance for this study in terms of its relevance to the building of a conceptual model. Refer to relevant literature on the subject.

*Principle 5: Size in two steps*

Although no entirely satisfactory method for sizing has been found, some rules of thumb are supplied. The sizing of servers, client and database machines may best be done in two iterations.

**In the first step** the apparent appropriate size of each unit may be formed in terms of key parameters for all applications, as in table 5.1.

| System Component | Load parameters | Performance depends on |
|---|---|---|
| PC client | Number of applications<br>Processing load<br>RAM and disk demand per applic<br>data format conversions | Processing power and storage size<br>Network input/output speeds<br>Disk access rate |
| Servers and mainframes | Number of applications<br>Load per application<br>Number of users per application<br>Transactions per user<br>Concurrent access demand<br>Data format conversion load<br>Utility and overhead demands<br>Failure nodes and counter-measures | Net transaction processing rate<br>Input/output processing rate<br>Processing power<br>RAM cache size<br>RAM size<br>Disk access rate<br>Disk storage size<br>Measures for backup and concurrency |
| if data held | Database requirements<br>Partitioning/replication/update policy<br>Data access protection & security | Database storage size, access rate<br><br>Data placement and protection |
| Network | File and interactive transaction traffic<br>Database update traffic<br>Data protocols overhead and routing algorithms<br>Gateways and bridger translation delays<br>Network management protocols<br>Physical support quality | Net throughput end to end<br>Data line speeds and number of lines<br><br>Protocol efficiency<br>Topology and gateways used<br><br>Network load and service level. |

**Table 5.1** *Sizing parameters and performance (Forge, 1995)*

The server type strongly influences network traffic. Communication traffic depends on the distribution of processing tasks between client and server. A network or communication specialist can also be consulted.

**In a second step** the design may be optimized. Bottlenecks and overloaded elements can then be removed from the design, or additional capacity added. Check network sizing early by prototyping the design and test horizontally across platforms.

*Principle 6: Select server platforms according to their roles.*

The essence of client/server design is to select a processor streamlined for its service. There are different types of servers, varying by technology, size of users supported and services they are best suited for. A growing trend is to use a high-power PC for workgroup servers. However, machines dedicated to server operations have appeared over the past few years.

Two key factors must be examined when studying the role of the server:

- the operational loading (number of users, network traffic peaks, processing, etc.)
- functional position in a hierarchy of servers.

Server operating systems dictate many facets of performance and capabilities. The operating system dictates the types and size of applications, number of users, communications interfaces and network architecture, as well as the client operating systems that are best interfaced with. Key comments concerning operating systems are provided in Appendix A exhibit A.3.

*Principle 7 Design for management.*

Management problems have been identified in the areas of the application, transactions, data administration, system software and networking. As there are few cross-platform management support tools, users generally have to build their own management infrastructure. At a basic network level, some useful standards for network management protocols have been appearing. A fully distributed environment management tool for the five management areas may come with the Open Software Foundations distributed management environment (DME) as shown in figure 4.11.

*Principle 8: Roll-out with care.*

According to Forge (1994), the biggest challenge in client/server and cooperative systems is high-quality integration across platforms, with full error checkout. A four-step process is recommended:

- internal department integration test : ISD test team tests according to test schedule drawn up with end-users.
- internal department user test : Key end users test system for inconsistencies, bugs, performance problems, and any poor working practices.
- pilot tests : few departments test the system for several months with full remote networking. diagnostics, and help desk running.
- roll-out across the entire organization.

The overall approach must be to design as if there is only one system, and then to split the application logic for each platform. Development must be an iteration of steps, creating prototypes as early as possible. The tasks can be summarized as follows:

- analyse business requirements with end users, deduce business logic, business objects and organizational constraints
- simulate the design of GUI together with the design of application logic, communications, data structures, services with splitting and sizing in iterations
- perform the application programming for each platform of the creation of application services for database access routines, test
- integration tests for platforms, transactions, separately and integrated.

A complication that cannot be ignored is that of legacy systems. Most organizations will have a legacy system that will have to be either converted to the new configuration, or left on its platform which is then integrated into the client/server system. Progressive moving of legacy systems are recommended. Alternatively encapsulation can be used by structuring the required parts of code or data as objects and linking them to the new applications. Digitalk supports this, using its Partswrapper function. Refer to Redelinghuys (1996) for further details on this matter.

### 5.3.4 Client/server Specific Design Methods

In the previous sections various methodologies for the development of client/server systems were reviewed. The development methods used should support the developer in creating decomposable applications and support the functional distribution and data distribution of the system. Each of these will be discussed briefly.

***Decomposable applications***

When designing a client/server system, the method must support scalability, performance and efficiency, which can be achieved by creating decomposable applications (Eckerson,1995).

In a two-tier architecture the GUI, application logic, and services reside at the client PC. The client issues SQL calls across a local area network to a relational database to retrieve data which reside at the server. However, a two-tier architecture is not sufficient to support enterprise-wide client/server applications. A middle tier is added to provide location and migration transparency of distributed resources, as well as a variety of services required to provide reliable, secure and efficient distributed computing.

The key to decomposable applications is to ensure that each of the components functions as a separate, independent component within the application. The three-tier application can be decomposed into presentation, functional logic and data. If the application cannot be decomposed, applications cannot easily be extracted and ported to another platform, or changed without affecting other parts of the application. Three methods of creating decomposable applications are identified by Eckerson (1994).

*Method 1:* Use a traditional programming language to segment monolithic applications into callable procedures.

*Method 2*: Link programs running presentation, logic, and data components via common interfaces, such as remote procedure calls (RPC's) or application programming interfaces (API).

*Method 3*: Build presentation, function and data components from a series of autonomous objects. A hybrid interface can also be created to force vendor interfaces to link up components that may be distributed across platforms.

***Functional distribution***

To take full advantage of the client/server paradigm, the developer needs to address issues of functional distribution of their application across client, server and network nodes. The issues surrounding this functional distribution constitute the single greatest difference between the physical design of mainframe multi-user and client/server applications.

To illustrate some of the issues and possibilities, take the example of a simplified order entry system, incorporating on-line transaction processing, off-cycle batch update processing, scheduled and ad-hoc production reports, and other special-purpose functionality. The simplest implementation is illustrated in figure 5.12, with all processing performed on the client stations, and the database server providing basic data management selecting and updating functionality.

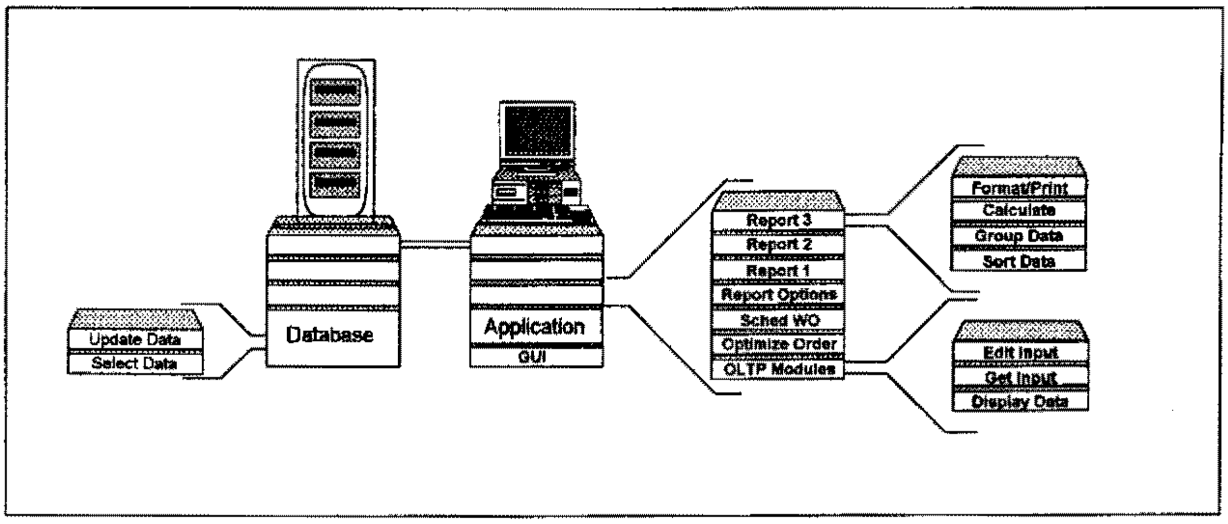


**Figure 5.12** *Data management distribution (Vaughn, 1994)*

The various distribution strategies which may be considered, are:

*Option 1* The edit input functions of the OLTP modules may be moved to the server which will place referential integrity enforcing on the server.

*Option 2* The sorting, grouping, and calculation functions of the report may be moved to the server.

*Option 3* All report functionality may be moved to the server if it is not heavily loaded, as illustrated in figure 5.13. This can be an effective strategy if the report does not require user interaction.

*Option 4* One client may be dedicated as a stand-alone report-server, with possibly a printer directly attached.

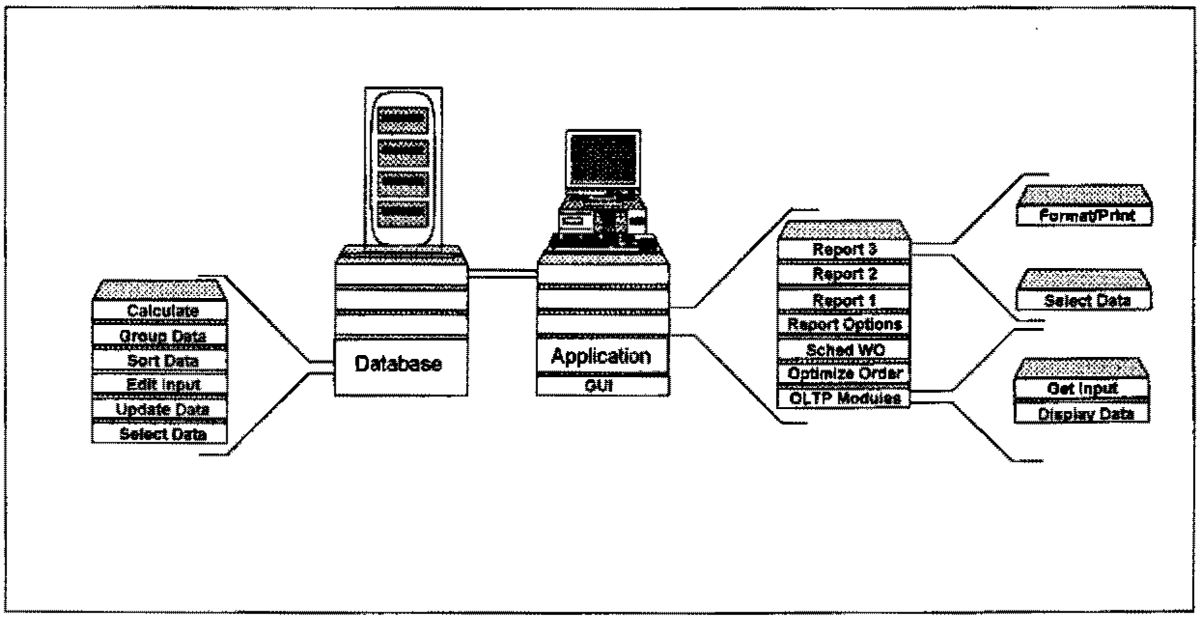


**Figure 5.13** *Query distribution (Vaughn, 1994)*

A few rules of thumb for functional distribution are proposed *(Vaughn, 1994)*:

- Locate the functionality as close as possible to the source of that functionality's input and the target of that function's output. Where source and target reside at different locations, consider decomposing the functionality further.
- Locate the functionality on the platform that provides the most appropriate resources to support it.
- Locate the functionality where it will act to conserve resources in the following order: conserve shared server resources, shared network bandwidth and client resources.

***Data distribution***

A distributed database is a collection of data distributed across multiple different platforms that are connected by a communication network, which provides localized application access to the data. A number of factors influence the distribution strategy (Eckerson, 1995):

*Capacity:* DASD and throughput capacity of the server platform can be exceeded, resulting in poor query performance. Segmenting and distributing data across multiple platforms reduce the demand processing and data storage requirements.

*Transmission media limitations:* In the process of accessing distributed data, the primary performance constraint becomes the bandwidth of the telecommunication medium used to establish the link. New technologies promise to increase the speed, but are expensive and not widely available.

*Organizational structure:* The organizational structure must be reflected in the data distribution structure.

*Combining heterogeneous data sources:* This scenario is the result of the fact that many large organizations have used two or maybe more database management systems, with incompatible data format.

***Data distribution constraints*:** There are many different ways of data distribution. However, the nature of the data distribution imposes a number of constraints:

- location transparency for application processes
- performance for distributed queries over WANs and relatively slow transmission lines
- fully distributed security that encompasses all servers
- full transaction management, distributed update commit protocols for failure detection and backout, distributed concurrency control
- allowance for failures of localized resources
- organizational ownership of distributed data.

Database distribution takes three basic forms today, namely distributed update, distributed query and data replication. An intensive study of data distribution falls beyond the scope of this field of study.

### 5.3.5 CASE TOOLS

Various tools are needed to assist the developer in the design process. Computer-Aided Software Engineering is an enabling technology that provides the user with a set of tools to aid the development process. CASE tools that help the developer during the earlier phases of the development process, namely the requirements, specification, planning and design phases are sometimes termed upperCASE or front-end tools, whereas those that assist with implementation, integration and maintenance are termed lowerCASE or back-end tools.

Various CASE tools are available on the market today, some vendor-specific. A CASE environment is a collection of tools that together support one or perhaps two phases of the development process. An integrated project support environment (IPSE) supports every phase of the process. A software engineering environment (SEE) is defined as the process, methods and automation required to produce a software system (Charette, 1987). Basic capabilities include:

- graphics tools for example data flow diagrams, flowcharts, E-R diagrams, structure charts, status transition diagrams
- data dictionary tools to record, maintain, and report on system details
- prototyping tools for external designs of inputs, screens, forms, and outputs. Some CASE tools can be interfaced with 4GL and application generators
- automatic quality checking for graphics and dictionary specifications. Consistency and completeness can be checked
- code generators to reduce programming effort required to transform specifications into systems
- cost-benefit analysis tools
- project management tools
- documentation assemblers to combine various specifications into reports.

Figure 5.14 illustrates the standard components which are supported.

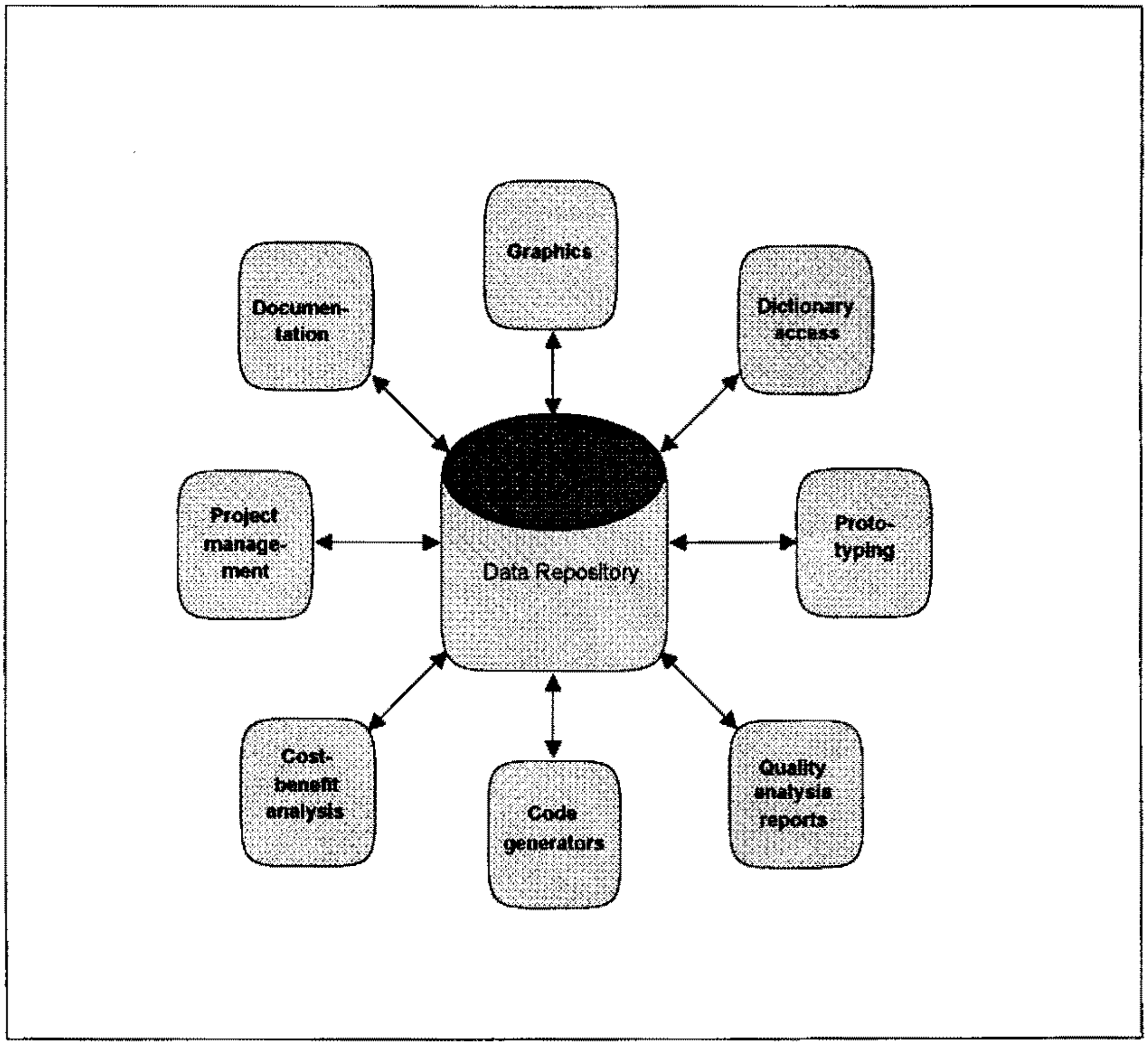


**Figure 5.14** *Standard CASE tool components (Whitten, 1993)*

These traditional tools may be and are used during the development process but, generally, they do not provide all the functionality needed for the development of client/server systems. Section 5.5 discusses client/server development tools.

### 5.3.6 Client/Server Application Development Tools

Effective tools for developing cross-platform applications are still rare. Major gaps exist, as many tools cover only one section of the application, such as GUI painting and SQL access to the common database. Specialized advanced tools are not used widely yet, perhaps because of their recent emergence and the consequent lack of user awareness and confidence. CASE support for a distributed environment is still limited and complete support for simulation, cross-platform execution and middleware has not realized yet.

The principal areas where gaps exist are as follows (Forge, 1995):

**Simulation and modelling:** especially in the area of performance modelling with cross-platform simulation for design, network and systems management.

**Cross-platform development**: holistic cross-platform management tools in the areas of application transactions, communications and PC LAN networking.

**Middleware tools and frameworks:** Only a few general products are available for interapplication communications, such as those from Netwise, Peer-to-Peer Logic and System Strategies.

Further limitations are the fact that many tools are environment-specific. An example is Entire from Software AG, a powerful tool set for building client/server applications with support for their own 4GL, Natural an Adabas database.

*Phases identified in the development of tools are:*

*First generation development tools* supply the developer with a 'best-of-the-breed' approach, but not necessarily using only one package. Enterprise-wide integration is therefore a problem. Components can be connected using interfaces supported by the different tools, although it will be more difficult to use several tools than one tool only. However, using the best-of-the-breed approach will prevent the developer from becoming vendor-dependent.

*Second generation tools* provide the developer with a set of tools to create an enterprise-wide client/server system in decomposable components. These tools try to permit the developer to design the entire application: the GUI, business logic, data access routines, communications code, and then store the designs in a repository with version control facilities. The tools usually contain a GUI tool kit for painting screens, a proprietary 4GL for writing business logic, and a point-and-click interface for defining services, such as SQL generation and translation, data access, communication modes, security, management and transaction controls. The tools generally support development in a Windows environment and generate code for a range of client and server platforms, databases, networks, and database gateways. Key features (Nienaber & Redelinghuys, 1995) to be supported by tools are partitioning, concurrency and serialization.

Where gaps still exist in second generation development tools, they may be filled by 3GL coding and extensions. For instance, users may end up programming in 3GL with SQL access and using traditional network protocols. Some PC packages have a programming interface accessible from the 3GLs, such as C as a library call, or from a PC development language such as Microsoft Visual Basic. This trend is expanding with a new generation of programming interfaces. Thus, PC packages can be integrated as a key building block into the final application using PC software development languages, the PC's linking and embedding standards, and the PC package's programming interface and macros.

However, major client/server and cooperative tools are being developed. Some tools are developed specifically for client/server, but many others may be PC front-end development tools with add-ons. Table 5.2 categorizes the types of tools available.

| CATEGORY | FUNCTIONS |
|---|---|
| *Traditional support* | |
| 4GLs and 3GLs | Languages with compilers<br>DBMS SQL Interface with tools to configure/manage the database on PC, server or mainframe. |
| *Client/server support* | |
| Client/server front-end | GUI screen painters for Windows 3.X, Presentation Manager and X/Motif screen generators for PCs, for client on PC, some with SQL query writers for database access routines. |
| Client/server cross-platform | 4GL application generators, database tools and OLTP transaction processing monitors for distributed processing |
| Configuration management | Version control with integration of modules. |
| *Support becoming available* | |
| Multi-platform management | Tools for cross-platform systems and network management |
| RPC generators and libraries | Middleware tools - hides communication complexity from programmers |
| Interapplication message queue | Middleware tools - hides communication complexity from programmers |
| Librarians | Middleware tools to locate information in files and databases across the network using directories. |
| Transaction managers | Manage transaction across several platforms |
| Database integrators | Link distributed different databases into one entity |
| Cross-platform tool linkers | Share information for tools on several platforms |
| Revamping tools | Add new interfaces to legacy applications, perhaps reverse engineer. |

**Table 5.2** *Types of client/server tools (Forge, 1995)*

For data distribution and partitioning, new concepts in tools and support are appearing, such as the Information Warehouse in conjunction with EDA (Enterprise Data Architecture), which integrates SQL and non-SQL, non-relational databases and file systems. This is presented together with the Distributed Relational Database Architecture (DRDA) as blueprint for the data structuring framework.

Commercial systems differ in their approach to client/server systems. In Appendix B selected commercially available client/server development tools are reviewed. However, the discussion and the tools discussed are by no means complete, as this falls beyond the scope of this dissertation.

A plethora of client/server development tools is available and it is impossible to give an in-depth discussion of all of them. Exhibit A4 in Appendix A gives a selection of client/server development tools for applications, database, front-end and management functions.

Basic requirements for application environments are:

- GUI tool kit/screen painter
- 3GL and 4GL for writing business logic
- point-and-click interface for defining services
- SQL generation and translation
- data access software
- communication modes
- security
- management
- transaction controls.

Communication tools that shield the developer from communication details are being developed.. Transaction management for multiplatform tracking and information optimizing for reducing network traffic are also developing fast.

Network management systems support the client/server application in performing the following essential functions:

- performance monitoring
- fault tracking with diagnostics
- configuration management
- accounting and billing
- security and user administration
- planning and modeling.

Tools for global naming and addressing, version control tools, and tools for legacy systems conversion are also required. The following criteria for front-end and database interface builders are suggested.

**Front-end screens building**

GUI support with icons, buttons and menus

forms generation and report generator

PC interfaces with OLE and DDE

support for porting changes

data representation.

**Database interface builders**

Control over record locks, error codes and messages

Does it support access to non-relational databases and VSAM, ISAM, ASCII and PC DOS files?

Does it support multiple sessions?

Does it mask SQL differences between different databases?

Does database interfacing take place direct or via a gateway?

Currently client/server development tools are appearing which support different levels of functionality. When selecting a tool, all of the above-mentioned criteria for selection must be consulted. However, additional factors such as cost per workstation, budget allocated, user training and available personnel are also of importance. For further detail refer to the list of selected client/server application development tools compiled in Appendix B.

## 5.4 Role Players

On account of the fact that client/server applications frequently involve multiple platforms, heterogeneous databases and multiple application programs, they are usually more complex than a single system application. Therefore, it may be difficult to find a single person with the necessary skills and expertise for such a project. It may be necessary to form a project team. Marion (1994) suggests that a team be formed for the following reasons:

**Need for diverse skills:** Not only programming skills are needed, but also skills from other areas such as data communications, local area networking, database design and management and PC hardware.

**Scope of the project:** The client/server application may span many platforms and involve multiple applications. The size and scope of the project will warrant the use of a team.

**To build consensus:** Technical as well as organizational aspects will have to be considered. The project team consisting of representatives of all the business units involved can help build consensus and acceptance of the project. A client/server application may also cross many organizational boundaries and this team can play a role in determining "ownership" of the project.

Given the rationale for forming the project team, persons with the necessary skills must be appointed to the team. A team can be organized as a democratic team, or according to the chief-programmer concept. As it is not improbable to find a group of people of the same age and with the necessary experience to form a democratic team, the chief-programmer team will be a better choice. A team for developing client/server systems can typically consist of a project manager, programmer, programmer-analyst, database analyst, data communications specialist, and end user representative. Each of these job categories are reviewed briefly.

## 5.4.1 Project Manager

The project manager is responsible for the management and direction of the project. Ideally the project manager should have managerial and technical skills. However, a basic understanding of the technologies will also suffice, provided that sufficient expertise is available from other team members. The project manager should be skilled in project management techniques and the use of project management tools. In addition, the person should have interpersonal skills to allow him/her to deal with team members, end users and upper management. Skills in the following areas are desirable:

- project management
- the architecture of client/server computing
- graphic user interfaces (Windows, OS/2)
- use of client applications development tools (vendor specific)
- use of Server databases
- knowledge of Local Area Networks.

### 5.4.2 Programmer/programmer-analyst

The programmer/programmer-analyst is responsible for the design and development of the application. This may involve the development of system requirements and specifications, prototypes, developing and testing of applications, documentation and maintenance. Experience of GUI's, design tools, SQL and database management systems is essential. A basic knowledge of local area networks, gateways and data communications are also required. Skills in the following areas are desirable:

- the architecture of client/server computing
- use of client operating systems (vendor specific DOS, OS/2, UNIX, Windows NT)
- use of graphic user interfaces (Windows, OS/2)
- use of client applications development tools or languages (vendor-specific i.e. Presentation Manager, C, Visual Basic, Delphi, etc. )
- knowledge of server databases (Gupta, ORACLE, DB2, Sybase)
- knowledge of structured query language (as specified)
- local area networks (basic principles)
- data communications.

### 5.4.3 Database Analyst

It may be necessary to incorporate existing databases into the system or organize large volumes of data. The services of a database specialist in the form of database analyst/database administrator may be required. Responsibilities may include the design and implementation of the new database, development of new server database applications, testing, documentation and database maintenance and administration. Most host database systems require a database administrator for support and maintenance. Skills in the following areas are desirable:

- the architecture of client/server computing

- server operating systems (vendor specific OS/2, UNIX, Novell, Windows NT)
- server databases (Gupta, ORACLE, DB2, Sybase)
- database design tools (vendor specific, i.e. Gupta, ORACLE, DB2, Sybase, etc.)
- structured query languages (as specified)
- database administration and security (vendor-specific, i.e., Gupta, ORACLE, DB2, Sybase).

### 5.4.4 Data Communications Specialist

The design and implementation of the required communication links between physically separated client/server platforms will be the responsibility of the data communication specialist. Tasks will include the design, and installation of new communication systems, reviewing existing communication systems, capacity planning, installation of communication hardware and software, configuration of host gateways, documentation and ongoing support and maintenance. These functions may be performed by the central IS staff or a LAN specialist.

The checklist for skills reads:

- the architecture of client/server computing
- client operating systems (vendor-specific DOS, OS/2, UNIX, Windows )
- network operating systems (vendor specific, i.e. Novell, IBM, Microsoft)
- client and server hardware installation and support (vendor-specific)
- communications hardware installation and support (vendor-specific )
- local area networks (design and installation).

### 5.4.5 End user representative

The end user is often overlooked but is a key member of the team. The responsibilities of the end user will be participation in defining the requirements and specification for the system, the development of user procedures, testing of applications, implementation of the application and training. The end user plays a major role in gaining acceptance for the client/server project and can help overcome organizational resistance to change.

Skills in the following areas are desirable:

- basic operation of PCs / workstations
- graphic user interfaces and their usage
- knowledge of local area networks
- client application software (application-specific)
- reporting tools.

The team will consist essentially of the above-mentioned role players. Depending on the size and complexity of the system it may be necessary to have additional members on the team. Focus groups may be added if the project is large and encompasses many areas of the organization. For instance, in a project to implement a financial system, separate focus groups may be used to define the requirements for accounts payable, accounts receivable.

Support from upper management is a key factor, since it is necessary to obtain resources in the form of equipment, staff time and funding. As the scope of the project expands and crosses organizational boundaries, the need for management support increases. As the goals of decentralization and enterprise computing are often contradictory, the need for broad management is crucial.

Two common approaches are suggested by Marion W (1994), namely a steering committee and an executive committee. The steering committee approach comprises a standing committee of top managers who review and approve all IS projects. According to the degree of centralization, the committee will be able to approve or disapprove a project. The executive committee is similar to the steering committee but is not a standing committee and is formed for each project. The success of a project can depend on how well it is managed and planned.

### 5.4.6 Training

Specialists in client/server systems are rare at present. When training is needed, how can it be obtained? The following options are available:

**Vendor training** is given by the vendor of the specific software that is used. These training programs can range from simple, end user training to in-depth application development training.

**Third-party training** is given by a third-party organization. Training on general topics as well as specific technical training are given.

**Courses** offered by public/private educational institutions.

**In-house training** or self-study.

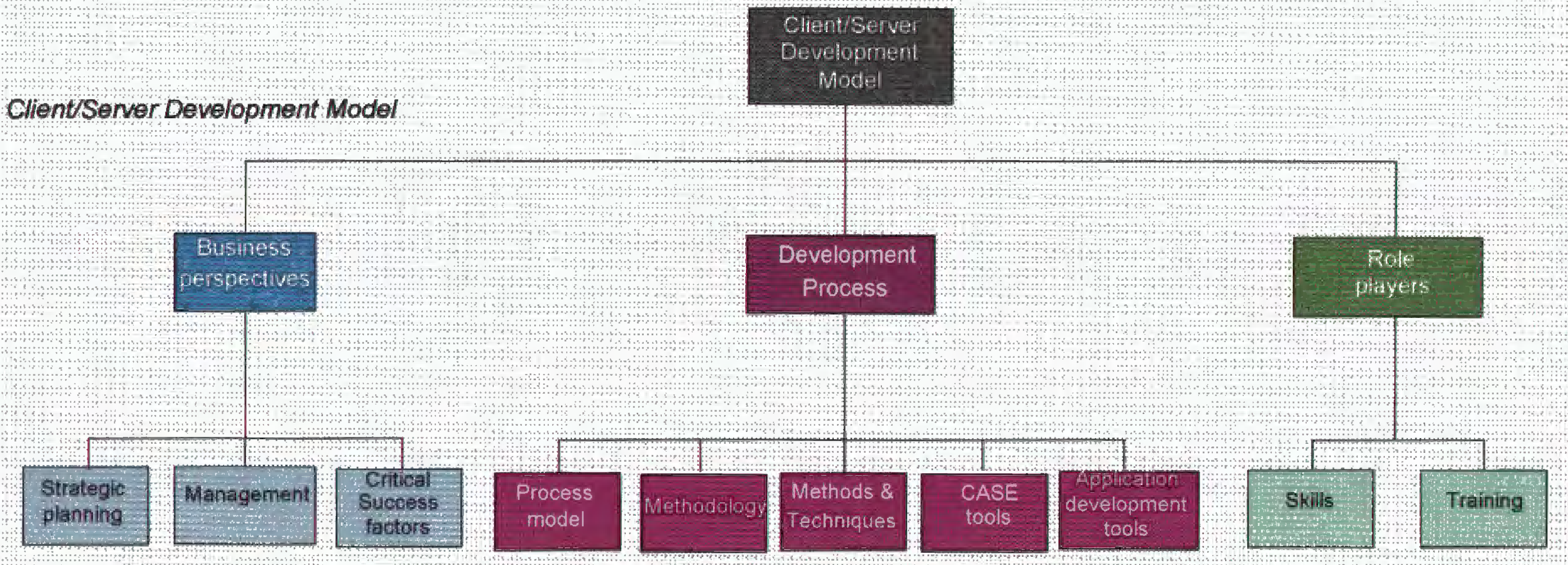
## 5.5 Summary

The client/server system is complex as client/server applications have more components than the traditional type of system. In this chapter, development aspects for client/server applications have been identified. The business perspective, management issues and critical success factors have been discussed. Traditional life-cycle processes, development methods, and techniques have been identified and a few methods suitable for client/server application development were reviewed. Design issues specifically important to the client/server environment were also identified. CASE tools as well as client/server development tools were discussed. The different role players in the development process and their skills were addressed. A model of the relevant components in the development process are illustrated in figure 5.15.

A summary of the investigation and various conclusions will be stated in the next chapter.

**Figure 5.15**

***Client/Server Development Model***



# CHAPTER 6

# SUMMARY and CONCLUSION

## 6.1 Introduction

This chapter reviews the investigation and relates the work done in each chapter to achieving the objectives of the study. A summary of the research results of this investigation is presented. Conclusions are drawn and assumptions are validated in the light of the results. A road map is compiled in the form of a technology reference model for client/server software development. The chapter concludes with proposed areas for further investigation.

## 6.2 Summary of Investigation

A thorough investigation of the area of research was executed. The research topic was introduced by illustrating the state of the art of client/server software development, including problems and definitions. The basic meta-primitives of a client/server software development application were identified and the results were presented in the preliminary client/server model in figure 1.1. The preliminary model was used throughout the study to illustrate research results.

The relevant issues which have bearing on the investigation were identified next. The rationale for the development of client/server software was investigated and this resulted in the identification of the driving forces for client/server application development, which were conceptualized in figure 2.4. The components of the client/server software development environment were introduced briefly and a taxonomy of the different modes of classification and classes of client/server systems was given. The main technologies of client/server systems were surveyed and the main components of the environment described. Meta-primitives of the technology model were derived and resulted in the instantiation of the client/server technology model presented in figure 4.14.

Another important aspect of the research was the identification of the essential components in the development process. Key aspects of the process of client/server software development were identified. Generic development process models were reviewed, as well as various methodologies, methods, techniques and tools. Relevant client/server development methodologies were investigated and listed. The role players, including their skills and training, were also identified. The specific design issues that are of importance concerning client/server systems were also reviewed. The result of this investigation was instantiated in the model that illustrates the key aspects of the development process represented in figure 5.15.

## 6.3 Conclusion

The aspects identified in the model are essential for the development of client/server software. As conclusion of the research the technology reference model can be used as road map in the process of client/server software development. The investigation led to various conclusions which will be stated briefly.

In the selection of the clients technology components, the main options available for hardware are compiled in table 4.1. The software components are summarized in table 4.2. Chapter 4 reviewed each of these components. The main components of the server are summarized in figure 4.4. Guidelines for selecting a database server is made available to developers who have to decide on a database server.

| |
|---|
| • Determine the existing infrastructure |
| • Describe the applications that is driving the move to client/server, the scope and target timeline |
| • Establish the compatibility and fit with the enterprise architecture |
| • Review standards, openness and required compatibilities |
| • Establish performance requirements and specifications. |
| • Evaluate server engine features for performance and capacity |
| • Determine purchase costs, costs of upgrade and application development. |
| • Have developers and database administrators evaluate the tools available. |
| • Investigate other companies, who have implemented a successful client/server system. |
| • Get an evaluation copy and develop a prototype |
| • Determine service and support available at remote sites |

**Table 6.1** *Guidelines for server selection*

Pragmatic standards across the enterprise must be defined and used as a basis. A list of de facto standards are compiled in Appendix A. Various available standard APIs are also listed in Appendix A, exhibit A2. For a client/server system the technical architecture is crucial. In the selection of the network, the hardware platform was investigated and discussed in section 4.4.1. A comparison of major approaches to networking is given in table 6.2.

| Proprietary protocol (SNA, DECnet, SQLNet | TCP/IP | Gateways or protocol converters | PCs with multi -protocol support |
|---|---|---|---|
| Provides optimal performance with host system | De facto standard | Has fewer protocols to load into PC memory | Is flexible, allowing communications with a variety of hosts |
| Offers reduced training and configuration options | Provides easy access to Internet resources | Can support dumb terminals<br>Some processing can be done on gateway CPU | May avoid costly host or gateway upgrades |
| Interconnectivity with other systems may be costly and complex | May not deliver high performance | May not deliver high performance | Has higher PC memory overhead |
| | Does not offer sufficient security | Provides exposure to single point of failure | PC configuration can be a nightmare |
| | | Additional node to manage | |

**Table 6.2** *Major approaches to networking*

Generic development methodologies were reviewed, as well as a number of published methodologies for client/server development. Forge's methodology is the most promising of these new methodologies and combines traditional methods using the spiral model and an object-oriented approach, rapid application development and prototyping. The investigation led to the identification of the key aspects of the development process which are shown in figure 5.15. A plethora of development tools is available for the development process. The most relevant tools are listed in Appendix A, exhibit A4. Appendix B review the approach, architecture and basic methodology used by a few relevant application development environments. Guidelines for the selection of application development tools are listed in table 6.3.

| **Product** | **Candidate 1** | **Candidate 2** |
|---|---|---|
| Unit cost | | |
| Runtime fees if any | | |
| Annual support/licensing fees | | |
| Training costs/developer | | |
| Track record of current users? | | |
| Suitable for: | | |
| programmer | | |
| developer | | |
| power user | | |
| end user | | |
| Development platforms | | |

| | | |
|---|---|---|
| Deployment platforms | | |
| Databases supported | | |
| ODBC support? | | |
| RPC and communication support? | | |
| Gateways supported | | |
| General integration with environment | | |
| Team development support | | |
| Platform independence | | |
| **End user features** | | |
| Built-in report writer? | | |
| Ease of learning and use? | | |
| Editable SQL | | |
| Built-in support for cm | | |
| **Developer features** | | |
| Development language | | |
| Team programming support | | |
| Object repository | | |
| Supports Inheritance | | |
| Interactive debugger | | |
| Prototyping facilities | | |
| **Other** | | |
| Vendor reputation | | |
| Market share | | |
| Market momentum | | |
| | | |

**Table 6.3** *Guidelines for client/server application development tools.*

The aim of the investigation was to compile a road map to aid the developer in the process of client/server software development. Such a road map was compiled. Together the two main models, figures 4.14 and 5.15 constitute the road map in the form of a technology reference model for client/server software development. The technology model can be used for future reference regarding client/server application implementation.

## 6.4 Areas for further Investigation

Various areas were identified for further investigation. The complexity of the client/server software development environment makes it possible for researchers to pursue the following:

- *Project management for client/server software development*, in which the complexities of managing the development process are studied in a cross-platform environment (Forge, 1995)
- *Client/server architectures*, where the distributed environment and its complex architecture is studied (Berson,1994)
- *Integration of legacy systems in a client/server environment*, reviewing the process of integrating legacy systems with client/server systems (Redelinghuys, 1996)
- *The role of standards and open systems in client/server software development*, in which standards and open systems are reviewed relating to the client/server environment (Dewire, 1994)
- *Application and data distribution in a client/server environment*, where the process of splitting applications and data across different platforms is studied ( Forge, 1995).

# APPENDIX A

| **De facto standards** | |
|---|---|
| DBMS | SQL - many varieties |
| GUI for PCs | Windows X3, CUA 91 |
| GUI for Unix PCs and workstations | Motif/X-Windows |
| PC NOS | Novell Netware for PC client to PC server with IPX |
| Unix NOS for Unix server | Novell Netware or NFS |
| Communications | TCP/IP |
| Standards in RPC | most common are Sun, Netwise, Novell, but are still emerging |
| IBM interworking communications | APPC/LU6.2 protocols |
| IBM GUI | Presentation Manager, CUA91 |
| LAN server OS | OS/2 |
| DBMS | DB2 |

**Exhibit A1** *De facto standards (Forge, 1995)*

| **De Facto standards** | |
|---|---|
| APPC | Original IBM SNA interfacing protocol for peer-to-peer operations |
| HLLAPI | IBM high-level language API |
| SNMP | Simple network management protocol for TCP/IP network management |
| SQL | Standard query language for database services access |
| Berkeley sockets | Communications for Unix (and OS) for TCP/IP and XNS |
| **Emerging standards** | |
| XA | X/Open interface for TP monitor to DB communications |
| VIM | Vendor-independent messaging (an E-mail API) |
| IDAPI | Integrated database API |
| XFTAM | X/Open API for FTAM, file transfer and management |
| XTI | X/Open transport interface for communications |
| ODBC | Microsoft Open Database Connectivity- API for for SQL-DBS with Windows |
| MAPI | Microsoft E-mail API |
| CPI_C | IBM Common Programming Interface Communications for LU 6.2 SAA communications |
| Novell Appware | API libraries for the Novell Netware NOS |
| **Open Standards Systems** | |
| POSIC | Operating systems API (standard call set for applications) |
| X400 | E-mail API |
| X500 | Directory services - API for naming and addressing |
| CMIP | Common Management information protocol |

**Exhibit A2** *Strategic APIs*

| Operating system | Key factors |
|---|---|
| UNIX (USL, SCO and vendors like IBM, HP, NCR , Digital, uniting with COSE and OSF/1 | Unix scales well with 1 to 1000 users<br>Is 'open'<br>Available with standard applications interface and Motif GUI.<br>User interface improving<br>not always suitable for non-professional users. |
| MS-DOS (Microsoft) | Limited in memory space, not multitasking but dominates the PC world. |
| OS/2 (IBM & Microsoft) | Multitasking but low-end performance only<br>20 - 50 heavy users |
| Network-loadable module, NLM (Novell) | NLM is designed for a server.<br>Minimal operating system, designed to run an application as service on a network.<br>Open to the extent that it supports some industry standards for protocols. |
| Microsoft Windows NT Advanced Server | NT is intended as server complement to Windows 3.<br>50 - 100 users<br>Support from systems vendors. |

**Exhibit A3** *Operating systems*

| **Cross-platform c/s application tools** | **PC screen painters/ SQL/ 4GL/5GL** | **RDBMS with SQL and tools** |
|---|---|---|
| Forte | Powersoft Powerbuilder | |
| Crossroads | Easel Workbench | Gupta SQLBase |
| Cooperative solutions Ellipse | Gupta SQLWindows | IBM DB2 (MVS, AIX, OS/2) |
| Enfin | Revelation | Informix |
| INTERSOLV APS | QBE Vision | Ingres |
| Knowledgeware Objectview | JYACC JAM | Oracle |
| MDNS Object/1 | NeXT NeXTSTEP | Progress |
| Softwright | Digitalk Parts | Sybase SQL Server |
| Uniface | Asymetrix toolbook | Microsoft SQL server |
| | Mozart | HP Allbase/SQL |
| | Datacase | |
| **Distributed OLTP monitors** | **RPC tools/ interfacing/ communication** | **Configuration managers** |
| USL Tuxedo | Netwise RPC | SEMA lifespan |
| Transarc Encina | NobleNet EZ-RPC | INTERSOLV PVCS |
| NCR TopEnd | Softwright Showcase | Cooperative solutions |
| Encina/9000NCR | Gupta SQLnetwork | Ellipse |
| | MS Comms Server | Forte |
| **Transaction managers** | IBM Comms manager | |
| Cooperative solutions Ellipse | DEC DNS/Pathworks | |
| Easel transaction server | | |
| **Cross-platform database linkers** | **Network/systems management & performance** | **3 GLS useful compilers and languages** |
| Constellation HyperSTAR | Novell NMS | Microfocus OO Cobol |
| IBM/IBE EDA | CA Unicentre/Star | MetaWare c++ |
| Uniface | HP Open NEtView | Rogue Wave tools |
| | IBM AIX NetView/6000 | MDBS Object/1 |
| **Cross-platform tools linkers** | OpenVision Technologies | Microsoft Visual Basic |
| HP/SAIC Softbench | Tivoli Systems | Smalltalk |
| | Legend CPE | Digital Smalltalk |
| | DEC Mcc/PolycentreEMA | |

**Exhibit A4** *A selection of client/server tools (Forge, 1995)*

# APPENDIX B

***Forte from Objective Solutions - Intellicorp***

Forte is an enterprise-wide object-oriented client/server development and deployment tool. It supports a wide range of 'open' servers and client computing and operating systems including DEC/VMS, DEC/OSF, HP/HP-UX, IBM/AIX SUN/Solaris, Apple/Mac, MS/Windows and MS/WindowsNT. It supports object-oriented development with a 4GL, a 4GL debugger, GUI, screen designer and a multi-user development repository.

*Key strategic issues are:*

The building of new systems is simplified by decoupling development from deployment. Programmers design a logical application that is independent of the physical environment.

A run-time system is then generated that manages the details of the graphic user interface, distribution of application functionality among clients and servers, communications, and strategies for high levels of reliability and performance.

Application logic is developed as if it were to run on a single machine. Forte then partitions the application and transparently manages the distribution pieces on multiple computers.

*The product architecture is divided into three components:*

1 Application definition facility that contains GUI - 4GL, repository and debugger

2 System generation facility with configuration, partitioning and code generation

3 Distributed executed facility with object manager, performance monitor, system administration.

*The development methodology*

Object modelling is used to model the business processes.

The application architecture must then be designed using the elements of the business model as a basis. Multiple applications can be designed. The use of an object-oriented business modelling approach directly supports the implementation of an application architecture also based on an object-oriented technology. The technical infrastructure comprises the current and planned hardware and software system components such as operating systems, networks, databases, and middleware.

The application architecture of Forte implements the business model in a manner independent of the underlying technical infrastructure. To realize this, the application development process is separated from the deployment process. The development environment makes it easy to construct the application, while the deployment environment supports both access to technical services required to support the application, and operations support to install, manage, and monitor the application. Forte allows the developer to build logically complete applications with a single system perspective and then partition them into client and server processes upon generation. The application can be developed on any of these platforms, and can then be arbitrarily divided into sets of objects called "application partitions" which can be deployed over the network by simply dragging and dropping in its deployment interface.

When an application partition is so deployed, its associated C source code is automatically downloaded to the target machine where it is compiled and then linked to Forte's distributed object manager (DOM). The DOM complies with the COBRA standard and automatically manages message passing between objects over a range of network protocols. All graphic user interfaces developed in Forte are portable between Motif, MS-Windows and the AppleMac.

Forte's three environments consist of the following:

- development environment: an object-oriented 4GL, GUI developer tools, repository, interpreters and debuggers
- system generation environment: configuration, partitioning, code generation
- distributed execution environment: distributed object manager, performance monitor, system administrator.

Forte includes tools for defining application functionality, generating deployment systems and managing the application in a distributed environment. The application definition facility comprises the GUI, object-oriented 4GL, repository, library routines and debugger, the system generation facility handles the configuration, partitioning and code generation and the distributed execution facility handles the system administration, performance monitor and distributed object manager.

***SAP R/3 System***

The basic architecture of SAP's R/3 product comprises the database server, application server and the PC workstations. This multi-tiered system is designed for maximum portability, modularity and scalability.

*Key strategic issues*

The basic architecture is a three-tiered architecture where the presentation and dialogue functions are performed at workstations, application logic runs on the application servers and a relational database manages and coordinates the body of shared data. However, more than one mode of operation are possible in this multi-tiered approach. It can be deployed either as a centralized system, a two-tiered or three-tiered system:

- In a centralized R/3 configuration all processing tasks are allocated to the central host computer. Terminals are connected to the central computer but their use is analogous to that of the classical centralized system.
- Two-tiered configurations are typically implemented with special presentation servers for graphic user interfaces. For example, a Windows PC can be used as presentation server or a powerful desktop PC for presentation and executing applications connected to the main server with the basic database software.
- In a three-tiered application, separate computers are used for presentation, running applications, and the underlying database.

Users can access several application servers, as a large number of different application servers can work parallel to one another, accessing data stored in the database server. The application server can be installed on either a homogeneous or a heterogeneous system.

The graphic user interface is portable and offered for a wide variety of platforms, ie Microsoft Windows, OS/2-Presentation Manager, OSF/Motif, and Apple Macintosh. The same functionality is, however, provided on all platforms of RS/3, regardless of the presentation layer.

The system provides extensive management facilities in the Computing Center Management System (CCMS), which includes system monitoring, system control and open interfaces management.

R/3 consists of standard software modules that support all the possible routines within an enterprise. Event-related work routines are systematically linked to form process chains, which are integrated to simplify changes made. If something is changed in one module, it will automatically be updated in the other linked modules.

The technological platform of R/3 is illustrated in table B1.

| | | |
|---|---|---|
| **Hardware** | UNIX systems Bull IBM Digital SNI HP SUN | AT&T, Data general, IBM Bull / Zenith, Digital Sequant, Compaq, HP(Intel), SNI |
| **Operating systems** | AIX, SINIX HP-UX Solaris OSF/1 | Windows NT |
| **Databases** | ORACLE7 Informix Online ADABAS D DB2/6000 | ORACLE7 ADABAS D MS SQL Server 95 |
| **Dialog** | Windows 3.1, Windows 95, OSF Motif Presentation Manager, Macintosh | |

**Table B1** *Technology platforms of RS/3*

## *The development methodology*

System development is done with a development environment, namely the development workbench (ABAP/4 Development Workbench). This environment comprises development tools such as: screen painter, menu painter, editor, debugger, tools for performance measurement, development tools for dialogue, repository, enterprise data model and the developer organizer.

*TUXEDO*

TUXEDO has also been called an enhanced client/server system which cooperates with the basic database server product to provide extended capabilities. The client requests services by name, and the system routes the requests to the named service. After completion of the service, the results are returned to the client as a return function. The server may reside on any platform and may be moved without programming involvement. Enhanced technology is provided by additional software from databases or file servers. TUXEDO began as an in-house transaction processing system within AT&T and has been developed into a fully mature enhanced client/server system. Release 4.0 was made available in 1990. A wide range of platforms are supported, namely UNIX, MS-DOS, OS/2, SUN SPARC, etc.

***Kappa / OMW* - Objective Solutions representatives for Intellicorp.**

Object Management Workbench is a an object-oriented full life-cycle CASE tool. OMW was created by IntelliCorp of Mountain View, Calif., in conjunction with Martin James & Co. based in Reston, Va. As it is built on top of Kappa, IntelliCorp's object-oriented application development environment, a single model can be used at all stages of development. OMW implements the Martin/Odell Object-Oriented Information Engineering Methodology. OOIE Methodology is compatible with existing application development methodologies. It provides full life-cycle support, and a rapid application development tool (KAPPA) which dynamically generates SQL to map records and tables into KAPPA objects. All screens are fully portable, access UNIX workstations running the X/Motif Window System and Microsoft Windows-based PCs.

KAPPA supports rapid iterative development, scaleable open architecture, has modelling capabilities, visual development and flexibility.

OMW provides the developer with business models that execute tasks. The workbench provides facilities such as an object diagrammer, event diagrammer, business rules modeller, scenario manager and report writer. It implements business rules in the object model and in the event model.

***Borland***

Borland's Object Component Architecture (BOCA) includes a standard database entity which will be offered with application tools such as Paradox, QuattroPro and dBase V, C++, and Delphi95. This will enable users to share data across platforms. With the exclusion of C++, these tools are used for building PC applications.

Borland is currently enhancing its products in ways that will allow them to facilitate the development of client/server applications. In addition, Borland's Object Exchange messaging module will allow users to send or receive information by e-mail, fax, voice, and other methods.

Delphi, its client/server development environment, enables developers to build compiled front-ends with a user-friendly GUI. Its capabilities combine with Borland's middleware technology to access remote servers. Application partitioning is also supported.

***GUPTA CORPORATION***

GUPTA desktop development product, SQL Windows 5.0, provides a comprehensive set of tools for developing client/server applications and an easy-to-use interface. However, this product only supports desktop databases for team programming capabilities and an interface to CASE tools add-on modules, SQLWindows Corporate, must be purchased. Gupta supply a range of client/server software, i.e. front-ends Quest, SQLWindows and TeamWindows, middleware, namely SQLRouters and SQLWindows Network and SQLBase as its back-end.

Interconnectivity with other products are supported. Microsoft's ODBC is supported, enabling communication with Microsoft products, and the Gupta Open Repository can reside on an Oracle7 database server, which allows sites with already installed Oracle7 to use their existing installed DBMS. QuickObjects permits integration with Lotus Notes.

*Microsoft*

Miscrosoft has focused on OLE as an object model of future and existing Microsoft products. OLE allows the developer to create compound documents that contain data objects from other OLE-enabled applications. Microsoft are offering a homogeneous client/server environment based on Windows and Windows NT clients. Access already supports OLE creation, and when connected with Visual Basic or Visual C++, client/server applications can be created. Underlying OLE is the object model Component Object Model (COM)

Visual Basic automates the creation of the user interface but code must be written for all but routine database transactions. In VB, developers can configure the environments for either stand-alone or client/server systems.

Microsoft is also working on an application programming interface, namely ODBC (Open Database Connectivity).

***Axiant* from Cognos Inc.**

Axiant is claimed by its distributors to create applications quickly, exploit the power of both client and server, and supply the client with fast and easy access to corporate data.

Its features include :

automated initial program development, cooperative team development, multi-user, object-based repository to store application objects, automated application models for describing standard business processes and the support of PowerHouse 4GL.

It also supports important client/server features, namely flexible application deployment, application partitioning and scalability.

It supplies the client with an integrated desktop with desktop tools Impromptu and Powerplay. Impromptu includes reporting abilities, multidimensional modelling capabilities, data analysis and reporting facilities. Watcom/SQL, is an industry standard database that allows developers to build and test applications on the development workstation which is targeted for server deployment.

Axiant incorporates a library of application-building rules from which it can create processing logic automatically. Powerplay is used as front-end for data-analysis.

# REFERENCE LIST

## Books

Berson, A. 1994. **Client/Server Architecture.** McGraw-Hill.

Cashin, J. 1993. **Client/Server Technology: The New Direction in Computer Networking Technology.** Computer Technology Research Corp.

Cerutti, D. and Pierson, D. 1993. **Distributed Computing Environments.** McGraw-Hill.

Charette, R. N. 1988. **Software Engineering Environments**. McGraw-Hill.

Critchley, T. A. and Batty, K. C. 1993. **Open Systems: The Reality.** Prentice Hall.

Coad, P. and Yourdon, E. 1990. **Object-Oriented Analysis.** 2nd ed. Yourdon Press/Prentice Hall.

Dewire, D.T. 1993. **Client/Server Computing.** McGraw-Hill Inc.

Falkenberg, E.D, Rolland, C., El-Sayed, E.N. 1992. **Information System Concepts: Improving the understanding.** North-Holland, IFIP.

Forge, S. 1995. **Developing Cooperative and Client server systems.** McGraw-Hill Inc.

Gagliardi, G. 1994. **Client/Server Computing.** Prentice Hall.

Hall, C. L. 1994. **Technical Foundations of Client/Server Systems.** John Wiley & Sons, Inc.

Kavanagh, P. 1995. **Downsizing for Client/Server applications.** AP Professionals.

Kroenke, D. M. 1995. **Database Processing: Fundamentals, Design and Implementation.** Fifth Edition. Prentice Hall.

Loftus, C.W., Sherratt, E.M., Gautier, R.J., Grandi, P.A.M., Price, D.E. & Tedd, M.D. 1995. **Distributed Software Engineering.** Prentice Hall.

Marion, W. 1994. **Client/Server Strategies.** McGraw-Hill.

McFadden, F. R. and Hoffer J. A. 1993. **Modern Database Management.** Fourth Edition. Benjamin/Cummings Publishing Company, Inc.

Powers, M. J., Cheney P. H., Crow G. 1990. **Structured Systems Development.** Thompson International Publishing.

Redelinghuys, L. 1996. **A methodology for integrating legacy systems with the client/server environment.** M.Sc. dissertation. UNISA.

Rumbaugh, J., Blaha, M. Premerlani, W., Eddy, F., Lorensen, W. 1991. **Object-Oriented Modeling and Design**. Prentice-Hall International Inc.

Schach, S.R. 1996. **Classical and Object-oriented Software Engineering.** Richard D Irwin.

Shafe, L. 1994. **A Manager's Guide to Client server**. Addison-Wesley publishing.

Vaughn, L. T. 1994. **Client/Server System Design & Implementation**. McGraw-Hill, Inc.

Watterson, K. 1995. **Client/Server Technology for Managers**. Addison-Wesley Publishers.

Whitten, J.L, Bentley, L.D., and Barlow, V.M. 1993. **Systems Analysis & Design Methods**. Third edition. Richard D. Irwin, Inc.

Yourdon, E. 1994. **Object-Oriented Systems Design: an integrated approach**, Prentice Hall.

## Journals

Appun, F. 1992. **Managing and controlling Client Server Environments.** SA Client Server Conference. October 20-21.

Aquino, A. A., and Goddard, D. 1994. **The right Tools for coding business rules.** Datamation, March.

Bergen, M. 1994. **Systems Integration: the need for change as the modern business imperative.** SA Systems Integration Conference, September 12-13.

Berquist, R. 1995. **The challenge is really this: Learn to be discriminating.** Special supplement: Client/server computing, Computing Canada, March.

Boehm, B. and Bose, P. 1994. **A collaborative spiral software process model based on Theory W.** Proceedings of Third International Conference on the Software Process, October.

Butterworth, P. 1994. **Building an application architecture to reflect the business model.** Forte software.

Cardy, J. 1994. **Implementing Systems Integration within an overall IT architecture.** SA Systems Integration Conference. Johannesburg, September 12-13.

Cox, B. J. 1994. **There is a silver bullet.** BYTE, October, pp 209.

De Kok, S. 1994. **Embargoed**, Proceedings of Systems Integration Conference, SA September 12-13.

Du Plessis, A.L. and Van der Walt, E., 1992. **Modeling the software development process**, ISCO2 Conference.

Eckerson, W.W. 1995. **Three-tier Client/Server Architecture: achieving scalability, performance and efficiency in client server applications.** Open Information Systems, Vol 10.

Lewis, T.G. 1995. **Where is client/server software headed?** IEEE Computer, April.

Jones, A.P. 1994. **The Open Systems Push: A driver for systems integration**. Proceedings of SA Systems Integration Conference, September.

Linthicum, D.S. 1994. **Client/server strategy: coping with complexity**. DBMS. v7, n4, p46. April.

Lunn, N. 1992. **Alternatives in Client/Server and Downsizing**, SA Client/Server Conference, October.

Lunn, N. 1992. **1 Mainframe, 10 Mini's or 100 Micro's:** SA Client/Server Conference, October.

Menninger, D. 1994. **Client/Server Tools go Desktop. Or is it Desktop Tools Go Client / Server.** Data Based Advisor, December, pp 101 - 107.

Mezick, D., et al. 1994. **What features do you need in a client/server front-end tool?** Data Based Advisor, September.

Myers, M. 1994. **Bugs, Fixes, and Workarounds.** Client Server Advisor, December, pp 108-112.

Myers, M., 1995. **Client/Server Technology Empowers Hospital Management.** Data Based Advisor, January, pp. 72-73.

Nelson, M.L. & Byrnes, R.B. 1992. **A spiral model of object-oriented rapid prototyping**. Proceedings of the eighth International Conference, August.

Nienaber, R.C., and Redelinghuys, L. 1995. **A Strategy for Open Systems**. Technical report for M.Sc. module on Open Systems. UNISA.

Robertson, J. A. 1994. **Managing systems integration within a framework of business process re-design.** Proceedings of Systems Integration Conference, September.

Semich, J. W. 1994. **What's the next step after Client/Server?** Datamation, March.

Smit, H. 1994. **Aligning Systems Integration with IT Strategic Planning and Business Needs.** Proceedings of Systems Integration Conference, SA September.

Smith J. M., 1995. **Why feds should stop worrying and learn to love it.** Spotlight on client/server, Government computer news, March 20, Vol14, pp73.

Steger, M. 1995. **Manage the Client/Server Monster.** Visual Basic Programmer's Journal, April, pp.42-50.

Steger, H., 1995. **End-user's dream, an IS manager's nightmare.** Client/server systems special supplement.

Steinholt, B. & Jorgensen, M. 1993. **Software development methods and life cycle models.** Telektronik, Vol 89 pp 44-51.

The. L., 1994. **Now you can Automate BPR.** Datamation.

Viljoen, Z. 1995. **The CSCW paradigm for software development.** M.Sc. dissertation. UNISA., June.

Wessels, W. 1992. **The new way demystified.** South African Client Server Conference. October 20-21.

Wilent, S. 1993. **Visualizing client/server development.** LAN Times, October 17.